

Odin

Generated by Doxygen 1.8.13

Thu Aug 10 2017 19:13:39

Contents

1	Data Structure Documentation	2
1.1	ace_cube_t Struct Reference	2
1.1.1	Detailed Description	2
1.1.2	Field Documentation	2
1.2	adder_signals Struct Reference	3
1.2.1	Detailed Description	3
1.2.2	Field Documentation	3
1.3	dp_ram_signals Struct Reference	4
1.3.1	Detailed Description	4
1.3.2	Field Documentation	4
1.4	exp_node Struct Reference	6
1.4.1	Detailed Description	6
1.4.2	Field Documentation	7
1.5	hard_block_model Struct Reference	8
1.5.1	Detailed Description	8
1.5.2	Field Documentation	8
1.6	hard_block_models Struct Reference	9
1.6.1	Detailed Description	10
1.6.2	Field Documentation	10
1.7	hard_block_pins Struct Reference	10
1.7.1	Detailed Description	10
1.7.2	Field Documentation	11
1.8	hard_block_ports Struct Reference	11
1.8.1	Detailed Description	11
1.8.2	Field Documentation	12
1.9	hashtable_node_t_t Struct Reference	13

1.9.1	Detailed Description	13
1.9.2	Field Documentation	13
1.10	hashtable_t_t Struct Reference	14
1.10.1	Detailed Description	14
1.10.2	Field Documentation	14
1.11	implicit_memory Struct Reference	16
1.11.1	Detailed Description	16
1.11.2	Field Documentation	16
1.12	line_t Struct Reference	18
1.12.1	Detailed Description	18
1.12.2	Field Documentation	18
1.13	lines_t Struct Reference	19
1.13.1	Detailed Description	19
1.13.2	Field Documentation	19
1.14	node_list_t_t Struct Reference	20
1.14.1	Detailed Description	20
1.14.2	Field Documentation	20
1.15	ParselInitRegState Struct Reference	21
1.15.1	Detailed Description	21
1.15.2	Member Function Documentation	21
1.16	pin_names Struct Reference	22
1.16.1	Detailed Description	22
1.16.2	Field Documentation	22
1.17	queue_node_t_t Struct Reference	23
1.17.1	Detailed Description	23
1.17.2	Field Documentation	23
1.18	queue_t_t Struct Reference	24
1.18.1	Detailed Description	24

1.18.2	Field Documentation	24
1.19	s_adder Struct Reference	25
1.19.1	Detailed Description	26
1.19.2	Field Documentation	26
1.20	s_memory Struct Reference	27
1.20.1	Detailed Description	27
1.20.2	Field Documentation	27
1.21	s_memory_port_sizes Struct Reference	29
1.21.1	Detailed Description	29
1.21.2	Field Documentation	29
1.22	s_multiplier Struct Reference	29
1.22.1	Detailed Description	30
1.22.2	Field Documentation	30
1.23	sp_ram_signals Struct Reference	30
1.23.1	Detailed Description	31
1.23.2	Field Documentation	31
1.24	stages_t Struct Reference	32
1.24.1	Detailed Description	32
1.24.2	Field Documentation	32
1.25	STRING_CACHE Struct Reference	34
1.25.1	Detailed Description	34
1.25.2	Field Documentation	34
1.26	test_vector Struct Reference	36
1.26.1	Detailed Description	36
1.26.2	Field Documentation	36
1.27	veri_define Struct Reference	37
1.27.1	Detailed Description	37
1.27.2	Field Documentation	37

1.28	veri_Defines Struct Reference	38
1.28.1	Detailed Description	38
1.28.2	Field Documentation	39
1.29	veri_flag_node Struct Reference	39
1.29.1	Detailed Description	39
1.29.2	Field Documentation	40
1.30	veri_flag_stack Struct Reference	40
1.30.1	Detailed Description	40
1.30.2	Field Documentation	40
1.31	veri_include Struct Reference	41
1.31.1	Detailed Description	41
1.31.2	Field Documentation	41
1.32	veri_Includes Struct Reference	42
1.32.1	Detailed Description	42
1.32.2	Field Documentation	42
2	File Documentation	43
2.1	vtr-verilog-to-routing/ODIN_II/SRC/DEBUG_TOOLS/netlist_visualizer.c File Reference	43
2.1.1	Function Documentation	43
2.2	vtr-verilog-to-routing/ODIN_II/SRC/DEBUG_TOOLS/netlist_visualizer.cpp File Reference	46
2.2.1	Function Documentation	46
2.3	vtr-verilog-to-routing/ODIN_II/SRC/DEBUG_TOOLS/netlist_visualizer.h File Reference	49
2.3.1	Function Documentation	49
2.4	vtr-verilog-to-routing/ODIN_II/SRC/DEBUG_TOOLS/print_netlist.c File Reference	50
2.4.1	Function Documentation	50
2.5	vtr-verilog-to-routing/ODIN_II/SRC/DEBUG_TOOLS/print_netlist.cpp File Reference	51
2.5.1	Function Documentation	51
2.6	vtr-verilog-to-routing/ODIN_II/SRC/DEBUG_TOOLS/print_netlist.h File Reference	52

2.6.1	Function Documentation	52
2.7	vtr-verilog-to-routing/ODIN_II/SRC/globals.h File Reference	52
2.7.1	Variable Documentation	53
2.8	vtr-verilog-to-routing/ODIN_II/SRC/lib_ast/ast_elaborate.cpp File Reference	58
2.8.1	Macro Definition Documentation	60
2.8.2	Function Documentation	61
2.8.3	Variable Documentation	79
2.9	vtr-verilog-to-routing/ODIN_II/SRC/lib_ast/ast_elaborate.h File Reference	80
2.9.1	Typedef Documentation	81
2.9.2	Function Documentation	81
2.10	vtr-verilog-to-routing/ODIN_II/SRC/lib_ast/ast_util.cpp File Reference	100
2.10.1	Function Documentation	101
2.11	vtr-verilog-to-routing/ODIN_II/SRC/lib_ast/ast_util.h File Reference	116
2.11.1	Function Documentation	117
2.12	vtr-verilog-to-routing/ODIN_II/SRC/lib_ast/parse_making_ast.cpp File Reference	132
2.12.1	Function Documentation	135
2.12.2	Variable Documentation	153
2.13	vtr-verilog-to-routing/ODIN_II/SRC/lib_ast/parse_making_ast.h File Reference	157
2.13.1	Function Documentation	159
2.14	vtr-verilog-to-routing/ODIN_II/SRC/lib_blif/output_blif.cpp File Reference	176
2.14.1	Function Documentation	177
2.15	vtr-verilog-to-routing/ODIN_II/SRC/lib_blif/output_blif.h File Reference	181
2.15.1	Function Documentation	181
2.16	vtr-verilog-to-routing/ODIN_II/SRC/lib_blif/read_blif.cpp File Reference	182
2.16.1	Macro Definition Documentation	183
2.16.2	Function Documentation	184
2.16.3	Variable Documentation	194
2.17	vtr-verilog-to-routing/ODIN_II/SRC/lib_blif/read_blif.h File Reference	194

2.17.1	Function Documentation	195
2.18	vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/adders.cpp File Reference	195
2.18.1	Function Documentation	196
2.18.2	Variable Documentation	205
2.19	vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/adders.h File Reference	208
2.19.1	Typedef Documentation	209
2.19.2	Function Documentation	209
2.19.3	Variable Documentation	217
2.20	vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/hard_blocks.cpp File Reference	218
2.20.1	Function Documentation	219
2.20.2	Variable Documentation	224
2.21	vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/hard_blocks.h File Reference	225
2.21.1	Function Documentation	225
2.21.2	Variable Documentation	229
2.22	vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/implicit_memory.cpp File Reference	230
2.22.1	Function Documentation	230
2.22.2	Variable Documentation	237
2.23	vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/implicit_memory.h File Reference	237
2.23.1	Function Documentation	238
2.23.2	Variable Documentation	243
2.24	vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/memories.cpp File Reference	244
2.24.1	Function Documentation	245
2.24.2	Variable Documentation	262
2.25	vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/memories.h File Reference	263
2.25.1	Macro Definition Documentation	264
2.25.2	Typedef Documentation	265
2.25.3	Function Documentation	265
2.25.4	Variable Documentation	278

2.26	vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/multipliers.cpp File Reference	279
2.26.1	Function Documentation	280
2.26.2	Variable Documentation	288
2.27	vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/multipliers.h File Reference	288
2.27.1	Typedef Documentation	289
2.27.2	Function Documentation	289
2.27.3	Variable Documentation	294
2.28	vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/subtractions.cpp File Reference	295
2.28.1	Function Documentation	295
2.28.2	Variable Documentation	298
2.29	vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/subtractions.h File Reference	300
2.29.1	Function Documentation	300
2.29.2	Variable Documentation	303
2.30	vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_check.cpp File Reference	304
2.30.1	Function Documentation	304
2.31	vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_check.h File Reference	309
2.31.1	Function Documentation	310
2.32	vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_cleanup.cpp File Reference	310
2.32.1	Macro Definition Documentation	311
2.32.2	Typedef Documentation	312
2.32.3	Function Documentation	312
2.32.4	Variable Documentation	315
2.33	vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_cleanup.h File Reference	318
2.33.1	Function Documentation	318
2.34	vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_create_from_ast.cpp File Reference	318
2.34.1	Macro Definition Documentation	321
2.34.2	Function Documentation	321
2.34.3	Variable Documentation	353

2.35	vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_create_from_ast.h File Reference	356
2.35.1	Function Documentation	356
2.36	vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_utils.cpp File Reference	358
2.36.1	Function Documentation	359
2.36.2	Variable Documentation	389
2.37	vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_utils.h File Reference	390
2.37.1	Function Documentation	391
2.38	vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/node_creation_library.cpp File Reference	422
2.38.1	Function Documentation	424
2.38.2	Variable Documentation	433
2.39	vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/node_creation_library.h File Reference	440
2.39.1	Function Documentation	441
2.40	vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/partial_map.cpp File Reference	449
2.40.1	Function Documentation	450
2.41	vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/partial_map.h File Reference	457
2.41.1	Function Documentation	457
2.42	vtr-verilog-to-routing/ODIN_II/SRC/odin_ii.cpp File Reference	458
2.42.1	Function Documentation	459
2.42.2	Variable Documentation	461
2.43	vtr-verilog-to-routing/ODIN_II/SRC/odin_util.cpp File Reference	462
2.43.1	Function Documentation	463
2.44	vtr-verilog-to-routing/ODIN_II/SRC/odin_util.h File Reference	480
2.44.1	Macro Definition Documentation	480
2.44.2	Function Documentation	481
2.45	vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL/read_xml_config_file.cpp File Reference	495
2.45.1	Function Documentation	496
2.45.2	Variable Documentation	499
2.46	vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL/read_xml_config_file.h File Reference	499

2.46.1	Function Documentation	500
2.47	vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL/verilog_bison_user_defined.h File Reference	500
2.47.1	Function Documentation	500
2.48	vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL/verilog_preprocessor.cpp File Reference	501
2.48.1	Function Documentation	501
2.48.2	Variable Documentation	508
2.49	vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL/verilog_preprocessor.h File Reference	508
2.49.1	Macro Definition Documentation	509
2.49.2	Typedef Documentation	510
2.49.3	Function Documentation	510
2.49.4	Variable Documentation	515
2.50	vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/ace.cpp File Reference	515
2.50.1	Macro Definition Documentation	517
2.50.2	Typedef Documentation	520
2.50.3	Function Documentation	521
2.51	vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/ace.h File Reference	527
2.51.1	Function Documentation	527
2.52	vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/queue.cpp File Reference	528
2.52.1	Function Documentation	528
2.53	vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/queue.h File Reference	531
2.53.1	Macro Definition Documentation	532
2.53.2	Function Documentation	532
2.54	vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/sim_block.h File Reference	532
2.54.1	Function Documentation	533
2.55	vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/simulate_blif.cpp File Reference	533
2.55.1	Macro Definition Documentation	535
2.55.2	Function Documentation	536
2.56	vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/simulate_blif.h File Reference	570
2.56.1	Macro Definition Documentation	573
2.56.2	Function Documentation	574
2.57	vtr-verilog-to-routing/ODIN_II/SRC/STRING_HASH_TOOL/hashtable.cpp File Reference	608
2.57.1	Function Documentation	609
2.58	vtr-verilog-to-routing/ODIN_II/SRC/STRING_HASH_TOOL/hashtable.h File Reference	614
2.58.1	Macro Definition Documentation	614
2.58.2	Function Documentation	615
2.59	vtr-verilog-to-routing/ODIN_II/SRC/STRING_HASH_TOOL/string_cache.cpp File Reference	615
2.59.1	Function Documentation	616
2.60	vtr-verilog-to-routing/ODIN_II/SRC/STRING_HASH_TOOL/string_cache.h File Reference	622
2.60.1	Function Documentation	623
2.61	vtr-verilog-to-routing/ODIN_II/SRC/types.h File Reference	627

1 Data Structure Documentation

1.1 `ace_cube_t` Struct Reference

Data Fields

- `pset cube`
- `int num_literals`
- `double static_prob`

1.1.1 Detailed Description

Definition at line 69 of file `ace.cpp`.

1.1.2 Field Documentation

1.1.2.1 `cube`

`pset cube`

Definition at line 70 of file `ace.cpp`.

1.1.2.2 `num_literals`

`int num_literals`

Definition at line 71 of file `ace.cpp`.

1.1.2.3 `static_prob`

`double static_prob`

Definition at line 72 of file `ace.cpp`.

The documentation for this struct was generated from the following file:

- `vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/ace.cpp`

1.2 adder_signals Struct Reference

```
#include <adders.h>
```

Data Fields

- `signal_list_t * a`
- `signal_list_t * b`
- `signal_list_t * cin`
- `signal_list_t * cout`
- `signal_list_t * sumout`

1.2.1 Detailed Description

Definition at line 38 of file adders.h.

1.2.2 Field Documentation

1.2.2.1 a

```
signal_list_t* a
```

Definition at line 39 of file adders.h.

1.2.2.2 b

```
signal_list_t* b
```

Definition at line 40 of file adders.h.

1.2.2.3 cin

```
signal_list_t* cin
```

Definition at line 41 of file adders.h.

1.2.2.4 cout

```
signal_list_t* cout
```

Definition at line 42 of file adders.h.

1.2.2.5 sumout

```
signal_list_t* sumout
```

Definition at line 43 of file adders.h.

The documentation for this struct was generated from the following file:

- vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/[adders.h](#)

1.3 dp_ram_signals Struct Reference

```
#include <memories.h>
```

Data Fields

- signal_list_t * [addr1](#)
- signal_list_t * [addr2](#)
- signal_list_t * [data1](#)
- signal_list_t * [data2](#)
- signal_list_t * [out1](#)
- signal_list_t * [out2](#)
- npin_t * [we1](#)
- npin_t * [we2](#)
- npin_t * [clk](#)

1.3.1 Detailed Description

Definition at line 61 of file memories.h.

1.3.2 Field Documentation

1.3.2.1 addr1

```
signal_list_t* addr1
```

Definition at line 62 of file memories.h.

1.3.2.2 addr2

```
signal_list_t* addr2
```

Definition at line 63 of file memories.h.

1.3.2.3 clk

```
npin_t* clk
```

Definition at line 70 of file memories.h.

1.3.2.4 data1

```
signal_list_t* data1
```

Definition at line 64 of file memories.h.

1.3.2.5 data2

```
signal_list_t* data2
```

Definition at line 65 of file memories.h.

1.3.2.6 out1

```
signal_list_t* out1
```

Definition at line 66 of file memories.h.

1.3.2.7 out2

```
signal_list_t* out2
```

Definition at line 67 of file memories.h.

1.3.2.8 we1

```
npin_t* we1
```

Definition at line 68 of file memories.h.

1.3.2.9 we2

```
npin_t* we2
```

Definition at line 69 of file memories.h.

The documentation for this struct was generated from the following file:

- [vtr-verilog-to-routing/ODIN-II/SRC/lib_blocks/memories.h](#)

1.4 exp_node Struct Reference

```
#include <ast_elaborate.h>
```

Data Fields

- struct {
 - short [operation](#)
 - int [data](#)
 - std::string [variable](#)
- int [id](#)
- int [flag](#)
- int [priority](#)
- struct [exp_node](#) * [next](#)
- struct [exp_node](#) * [pre](#)

1.4.1 Detailed Description

Definition at line 41 of file ast_elaborate.h.

1.4.2 Field Documentation

1.4.2.1 data

`int data`

Definition at line 46 of file ast_elaborate.h.

1.4.2.2 flag

`int flag`

Definition at line 51 of file ast_elaborate.h.

1.4.2.3 id

`int id`

Definition at line 50 of file ast_elaborate.h.

1.4.2.4 next

`struct exp_node* next`

Definition at line 53 of file ast_elaborate.h.

1.4.2.5 operation

`short operation`

Definition at line 45 of file ast_elaborate.h.

1.4.2.6 pre

`struct exp_node* pre`

Definition at line 54 of file ast_elaborate.h.

1.4.2.7 priority

```
int priority
```

Definition at line 52 of file ast_elaborate.h.

1.4.2.8 type

```
struct { ... } type
```

1.4.2.9 variable

```
std::string variable
```

Definition at line 47 of file ast_elaborate.h.

The documentation for this struct was generated from the following file:

- vtr-verilog-to-routing/ODIN_II/SRC/lib_ast/[ast_elaborate.h](#)

1.5 hard_block_model Struct Reference

Data Fields

- char * [name](#)
- [hard_block_pins](#) * [inputs](#)
- [hard_block_pins](#) * [outputs](#)
- [hard_block_ports](#) * [input_ports](#)
- [hard_block_ports](#) * [output_ports](#)

1.5.1 Detailed Description

Definition at line 73 of file read_blif.cpp.

1.5.2 Field Documentation

1.5.2.1 input_ports

```
hard_block_ports* input_ports
```

Definition at line 79 of file read_blif.cpp.

1.5.2.2 inputs

```
hard_block_pins* inputs
```

Definition at line 76 of file read_blif.cpp.

1.5.2.3 name

```
char* name
```

Definition at line 74 of file read_blif.cpp.

1.5.2.4 output_ports

```
hard_block_ports* output_ports
```

Definition at line 80 of file read_blif.cpp.

1.5.2.5 outputs

```
hard_block_pins* outputs
```

Definition at line 77 of file read_blif.cpp.

The documentation for this struct was generated from the following file:

- [vtr-verilog-to-routing/ODIN_II/SRC/lib_blif/read_blif.cpp](#)

1.6 hard_block_models Struct Reference

Data Fields

- [hard_block_model](#) ** [models](#)
- int [count](#)
- [hashtable_t](#) * [index](#)

1.6.1 Detailed Description

Definition at line 84 of file read_blif.cpp.

1.6.2 Field Documentation

1.6.2.1 count

```
int count
```

Definition at line 86 of file read_blif.cpp.

1.6.2.2 index

```
hashtable_t* index
```

Definition at line 88 of file read_blif.cpp.

1.6.2.3 models

```
hard_block_model** models
```

Definition at line 85 of file read_blif.cpp.

The documentation for this struct was generated from the following file:

- [vtr-verilog-to-routing/ODIN_II/SRC/lib_blif/read_blif.cpp](#)

1.7 hard_block_pins Struct Reference

Data Fields

- int [count](#)
- char ** [names](#)
- [hashtable_t](#) * [index](#)

1.7.1 Detailed Description

Definition at line 55 of file read_blif.cpp.

1.7.2 Field Documentation

1.7.2.1 count

`int count`

Definition at line 56 of file `read_blif.cpp`.

1.7.2.2 index

`hashtable_t* index`

Definition at line 59 of file `read_blif.cpp`.

1.7.2.3 names

`char** names`

Definition at line 57 of file `read_blif.cpp`.

The documentation for this struct was generated from the following file:

- `vtr-verilog-to-routing/ODIN_II/SRC/lib_blif/read_blif.cpp`

1.8 hard_block_ports Struct Reference

Data Fields

- `char * signature`
- `int count`
- `int * sizes`
- `char ** names`
- `hashtable_t * index`

1.8.1 Detailed Description

Definition at line 63 of file `read_blif.cpp`.

1.8.2 Field Documentation

1.8.2.1 count

```
int count
```

Definition at line 65 of file read_blif.cpp.

1.8.2.2 index

```
hashtable_t* index
```

Definition at line 69 of file read_blif.cpp.

1.8.2.3 names

```
char** names
```

Definition at line 67 of file read_blif.cpp.

1.8.2.4 signature

```
char* signature
```

Definition at line 64 of file read_blif.cpp.

1.8.2.5 sizes

```
int* sizes
```

Definition at line 66 of file read_blif.cpp.

The documentation for this struct was generated from the following file:

- [vtr-verilog-to-routing/ODIN_II/SRC/lib_blif/read_blif.cpp](#)

1.9 hashtable_node_t Struct Reference

```
#include <hashtable.h>
```

Data Fields

- `size_t` [key_length](#)
- `void *` [key](#)
- `void *` [item](#)
- `hashtable_node_t *` [next](#)

1.9.1 Detailed Description

Definition at line 55 of file hashtable.h.

1.9.2 Field Documentation

1.9.2.1 item

```
void* item
```

Definition at line 59 of file hashtable.h.

1.9.2.2 key

```
void* key
```

Definition at line 58 of file hashtable.h.

1.9.2.3 key_length

```
size_t key_length
```

Definition at line 57 of file hashtable.h.

1.9.2.4 next

```
hashtable_node_t* next
```

Definition at line 60 of file hashtable.h.

The documentation for this struct was generated from the following file:

- [vtr-verilog-to-routing/ODIN_II/SRC/STRING_HASH_TOOL/hashtable.h](#)

1.10 hashtable_t Struct Reference

```
#include <hashtable.h>
```

Data Fields

- int [count](#)
- int [store_size](#)
- [hashtable_node_t](#) ** [store](#)
- void(* [add](#))([hashtable_t](#) *h, const void *key, size_t key_length, void *item)
- void(* [remove](#))([hashtable_t](#) *h, const void *key, size_t key_length)
- void(* [get](#))([hashtable_t](#) *h, const void *key, size_t key_length)
- void **(* [get_all](#))([hashtable_t](#) *h)
- int(* [is_empty](#))([hashtable_t](#) *h)
- void(* [destroy](#))([hashtable_t](#) *h)
- void(* [destroy_free_items](#))([hashtable_t](#) *h)

1.10.1 Detailed Description

Definition at line 33 of file hashtable.h.

1.10.2 Field Documentation

1.10.2.1 add

```
void(* add) ( hashtable\_t *h, const void *key, size_t key_length, void *item)
```

Definition at line 40 of file hashtable.h.

1.10.2.2 count

```
int count
```

Definition at line 35 of file hashtable.h.

1.10.2.3 destroy

```
void(* destroy) (hashtable_t *h)
```

Definition at line 50 of file hashtable.h.

1.10.2.4 destroy_free_items

```
void(* destroy_free_items) (hashtable_t *h)
```

Definition at line 52 of file hashtable.h.

1.10.2.5 get

```
void*(* get) (hashtable_t *h, const void *key, size_t key_length)
```

Definition at line 44 of file hashtable.h.

1.10.2.6 get_all

```
void**(* get_all) (hashtable_t *h)
```

Definition at line 46 of file hashtable.h.

1.10.2.7 is_empty

```
int(* is_empty) (hashtable_t *h)
```

Definition at line 48 of file hashtable.h.

1.10.2.8 remove

```
void>(* remove) (hashtable_t *h, const void *key, size_t key_length)
```

Definition at line 42 of file hashtable.h.

1.10.2.9 store

```
hashtable_node_t** store
```

Definition at line 37 of file hashtable.h.

1.10.2.10 store_size

```
int store_size
```

Definition at line 36 of file hashtable.h.

The documentation for this struct was generated from the following file:

- [vtr-verilog-to-routing/ODIN_II/SRC/STRING_HASH_TOOL/hashtable.h](#)

1.11 implicit_memory Struct Reference

```
#include <implicit_memory.h>
```

Data Fields

- nnode_t * [node](#)
- int [data_width](#)
- int [addr_width](#)
- char [clock_added](#)
- char [output_added](#)
- char * [name](#)

1.11.1 Detailed Description

Definition at line 33 of file implicit_memory.h.

1.11.2 Field Documentation

1.11.2.1 addr_width

```
int addr_width
```

Definition at line 36 of file implicit_memory.h.

1.11.2.2 clock_added

```
char clock_added
```

Definition at line 37 of file implicit_memory.h.

1.11.2.3 data_width

```
int data_width
```

Definition at line 35 of file implicit_memory.h.

1.11.2.4 name

```
char* name
```

Definition at line 39 of file implicit_memory.h.

1.11.2.5 node

```
nnode_t* node
```

Definition at line 34 of file implicit_memory.h.

1.11.2.6 output_added

```
char output_added
```

Definition at line 38 of file implicit_memory.h.

The documentation for this struct was generated from the following file:

- [vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/implicit_memory.h](#)

1.12 line_t Struct Reference

```
#include <simulate_blif.h>
```

Data Fields

- int [number_of_pins](#)
- int [max_number_of_pins](#)
- npin_t ** [pins](#)
- int * [pin_numbers](#)
- char * [name](#)
- int [type](#)

1.12.1 Detailed Description

Definition at line 64 of file `simulate_blif.h`.

1.12.2 Field Documentation

1.12.2.1 max_number_of_pins

```
int max_number_of_pins
```

Definition at line 66 of file `simulate_blif.h`.

1.12.2.2 name

```
char* name
```

Definition at line 69 of file `simulate_blif.h`.

1.12.2.3 number_of_pins

```
int number_of_pins
```

Definition at line 65 of file `simulate_blif.h`.

1.12.2.4 pin_numbers

```
int* pin_numbers
```

Definition at line 68 of file simulate_blif.h.

1.12.2.5 pins

```
npin_t** pins
```

Definition at line 67 of file simulate_blif.h.

1.12.2.6 type

```
int type
```

Definition at line 70 of file simulate_blif.h.

The documentation for this struct was generated from the following file:

- vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/[simulate_blif.h](#)

1.13 lines_t Struct Reference

```
#include <simulate_blif.h>
```

Data Fields

- [line_t](#) ** [lines](#)
- int * [pin_numbers](#)
- int [count](#)

1.13.1 Detailed Description

Definition at line 73 of file simulate_blif.h.

1.13.2 Field Documentation

1.13.2.1 count

```
int count
```

Definition at line 76 of file `simulate_blif.h`.

1.13.2.2 lines

```
line_t** lines
```

Definition at line 74 of file `simulate_blif.h`.

1.13.2.3 pin_numbers

```
int* pin_numbers
```

Definition at line 75 of file `simulate_blif.h`.

The documentation for this struct was generated from the following file:

- `vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/simulate_blif.h`

1.14 node_list_t_t Struct Reference

Data Fields

- `nnode_t * node`
- `struct node_list_t_t * next`

1.14.1 Detailed Description

Definition at line 42 of file `netlist_cleanup.cpp`.

1.14.2 Field Documentation

1.14.2.1 next

```
struct node_list_t_t* next
```

Definition at line 44 of file `netlist_cleanup.cpp`.

1.14.2.2 node

nnode_t* node

Definition at line 43 of file netlist_cleanup.cpp.

The documentation for this struct was generated from the following file:

- vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/[netlist_cleanup.cpp](#)

1.15 ParseInitRegState Struct Reference

Public Member Functions

- int [from_str](#) (std::string str)
- std::string [to_str](#) (int val)
- std::vector< std::string > [default_choices](#) ()

1.15.1 Detailed Description

Definition at line 271 of file odin_ii.cpp.

1.15.2 Member Function Documentation

1.15.2.1 default_choices()

```
std::vector<std::string> default_choices ( ) [inline]
```

Definition at line 290 of file odin_ii.cpp.

Here is the caller graph for this function:



1.15.2.2 from_str()

```
int from_str (
    std::string str ) [inline]
```

Definition at line 272 of file `odin_ii.cpp`.

1.15.2.3 to_str()

```
std::string to_str (
    int val ) [inline]
```

Definition at line 281 of file `odin_ii.cpp`.

The documentation for this struct was generated from the following file:

- [vtr-verilog-to-routing/ODIN_II/SRC/odin_ii.cpp](#)

1.16 pin_names Struct Reference

```
#include <simulate_blif.h>
```

Data Fields

- `char **` [pins](#)
- `int` [count](#)

1.16.1 Detailed Description

Definition at line 59 of file `simulate_blif.h`.

1.16.2 Field Documentation

1.16.2.1 count

```
int count
```

Definition at line 61 of file `simulate_blif.h`.

1.16.2.2 `pins`

```
char** pins
```

Definition at line 60 of file `simulate_blif.h`.

The documentation for this struct was generated from the following file:

- `vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/simulate_blif.h`

1.17 `queue_node_t_t` Struct Reference

```
#include <queue.h>
```

Data Fields

- `queue_node_t` * `next`
- void * `item`

1.17.1 Detailed Description

Definition at line 49 of file `queue.h`.

1.17.2 Field Documentation

1.17.2.1 `item`

```
void* item
```

Definition at line 52 of file `queue.h`.

1.17.2.2 `next`

```
queue_node_t* next
```

Definition at line 51 of file `queue.h`.

The documentation for this struct was generated from the following file:

- `vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/queue.h`

1.18 queue_t Struct Reference

```
#include <queue.h>
```

Data Fields

- [queue_node_t](#) * [head](#)
- [queue_node_t](#) * [tail](#)
- int [count](#)
- void(* [add](#))(queue_t *q, void *item)
- void (* [remove](#))(queue_t *q)
- void **(* [remove_all](#))(queue_t *q)
- int(* [is_empty](#))(queue_t *q)
- void(* [destroy](#))(queue_t *q)

1.18.1 Detailed Description

Definition at line 31 of file queue.h.

1.18.2 Field Documentation

1.18.2.1 add

```
void(* add) (queue\_t *q, void *item)
```

Definition at line 38 of file queue.h.

1.18.2.2 count

```
int count
```

Definition at line 35 of file queue.h.

1.18.2.3 destroy

```
void(* destroy) (queue\_t *q)
```

Definition at line 46 of file queue.h.

1.18.2.4 head

`queue_node_t* head`

Definition at line 33 of file queue.h.

1.18.2.5 is_empty

`int (* is_empty) (queue_t *q)`

Definition at line 44 of file queue.h.

1.18.2.6 remove

`void* (* remove) (queue_t *q)`

Definition at line 40 of file queue.h.

1.18.2.7 remove_all

`void** (* remove_all) (queue_t *q)`

Definition at line 42 of file queue.h.

1.18.2.8 tail

`queue_node_t* tail`

Definition at line 34 of file queue.h.

The documentation for this struct was generated from the following file:

- [vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/queue.h](#)

1.19 s_adder Struct Reference

```
#include <adders.h>
```

Data Fields

- int [size_a](#)
- int [size_b](#)
- int [size_cin](#)
- int [size_sumout](#)
- int [size_cout](#)
- struct [s_adder](#) * [next](#)

1.19.1 Detailed Description

Definition at line 28 of file adders.h.

1.19.2 Field Documentation

1.19.2.1 next

```
struct s\_adder* next
```

Definition at line 35 of file adders.h.

1.19.2.2 size_a

```
int size_a
```

Definition at line 30 of file adders.h.

1.19.2.3 size_b

```
int size_b
```

Definition at line 31 of file adders.h.

1.19.2.4 size_cin

```
int size_cin
```

Definition at line 32 of file adders.h.

1.19.2.5 size_cout

```
int size_cout
```

Definition at line 34 of file adders.h.

1.19.2.6 size_sumout

```
int size_sumout
```

Definition at line 33 of file adders.h.

The documentation for this struct was generated from the following file:

- vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/[adders.h](#)

1.20 s_memory Struct Reference

```
#include <memories.h>
```

Data Fields

- int [size_d1](#)
- int [size_d2](#)
- int [size_addr1](#)
- int [size_addr2](#)
- int [size_out1](#)
- int [size_out2](#)
- struct [s_memory](#) * [next](#)

1.20.1 Detailed Description

Definition at line 36 of file memories.h.

1.20.2 Field Documentation

1.20.2.1 next

```
struct s\_memory* next
```

Definition at line 44 of file memories.h.

1.20.2.2 size_addr1

```
int size_addr1
```

Definition at line 40 of file memories.h.

1.20.2.3 size_addr2

```
int size_addr2
```

Definition at line 41 of file memories.h.

1.20.2.4 size_d1

```
int size_d1
```

Definition at line 38 of file memories.h.

1.20.2.5 size_d2

```
int size_d2
```

Definition at line 39 of file memories.h.

1.20.2.6 size_out1

```
int size_out1
```

Definition at line 42 of file memories.h.

1.20.2.7 size_out2

```
int size_out2
```

Definition at line 43 of file memories.h.

The documentation for this struct was generated from the following file:

- [vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/memories.h](#)

1.21 s_memory_port_sizes Struct Reference

```
#include <memories.h>
```

Data Fields

- int [size](#)
- char * [name](#)

1.21.1 Detailed Description

Definition at line 47 of file memories.h.

1.21.2 Field Documentation

1.21.2.1 name

```
char* name
```

Definition at line 50 of file memories.h.

1.21.2.2 size

```
int size
```

Definition at line 49 of file memories.h.

The documentation for this struct was generated from the following file:

- vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/[memories.h](#)

1.22 s_multiplier Struct Reference

```
#include <multipliers.h>
```

Data Fields

- int [size_a](#)
- int [size_b](#)
- int [size_out](#)
- struct [s_multiplier](#) * [next](#)

1.22.1 Detailed Description

Definition at line 28 of file multipliers.h.

1.22.2 Field Documentation

1.22.2.1 next

```
struct s_multiplier* next
```

Definition at line 33 of file multipliers.h.

1.22.2.2 size_a

```
int size_a
```

Definition at line 30 of file multipliers.h.

1.22.2.3 size_b

```
int size_b
```

Definition at line 31 of file multipliers.h.

1.22.2.4 size_out

```
int size_out
```

Definition at line 32 of file multipliers.h.

The documentation for this struct was generated from the following file:

- [vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/multipliers.h](#)

1.23 sp_ram_signals Struct Reference

```
#include <memories.h>
```

Data Fields

- signal_list_t * [addr](#)
- signal_list_t * [data](#)
- signal_list_t * [out](#)
- npin_t * [we](#)
- npin_t * [clk](#)

1.23.1 Detailed Description

Definition at line 53 of file memories.h.

1.23.2 Field Documentation

1.23.2.1 addr

```
signal_list_t* addr
```

Definition at line 54 of file memories.h.

1.23.2.2 clk

```
npin_t* clk
```

Definition at line 58 of file memories.h.

1.23.2.3 data

```
signal_list_t* data
```

Definition at line 55 of file memories.h.

1.23.2.4 out

```
signal_list_t* out
```

Definition at line 56 of file memories.h.

1.23.2.5 we

```
npin_t* we
```

Definition at line 57 of file memories.h.

The documentation for this struct was generated from the following file:

- vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/[memories.h](#)

1.24 stages_t Struct Reference

```
#include <simulate_blif.h>
```

Data Fields

- nnode_t *** [stages](#)
- int * [counts](#)
- int [count](#)
- double * [sequential_times](#)
- double * [parallel_times](#)
- int [num_nodes](#)
- int [num_connections](#)
- int * [num_children](#)
- int [num_parallel_nodes](#)

1.24.1 Detailed Description

Definition at line 79 of file simulate_blif.h.

1.24.2 Field Documentation

1.24.2.1 count

```
int count
```

Definition at line 82 of file simulate_blif.h.

1.24.2.2 counts

```
int* counts
```

Definition at line 81 of file simulate_blif.h.

1.24.2.3 num_children

```
int* num_children
```

Definition at line 89 of file simulate_blif.h.

1.24.2.4 num_connections

```
int num_connections
```

Definition at line 88 of file simulate_blif.h.

1.24.2.5 num_nodes

```
int num_nodes
```

Definition at line 87 of file simulate_blif.h.

1.24.2.6 num_parallel_nodes

```
int num_parallel_nodes
```

Definition at line 90 of file simulate_blif.h.

1.24.2.7 parallel_times

```
double* parallel_times
```

Definition at line 84 of file simulate_blif.h.

1.24.2.8 sequential_times

```
double* sequential_times
```

Definition at line 83 of file simulate_blif.h.

1.24.2.9 stages

```
nnode_t*** stages
```

Definition at line 80 of file simulate_blif.h.

The documentation for this struct was generated from the following file:

- vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/[simulate_blif.h](#)

1.25 STRING_CACHE Struct Reference

```
#include <string_cache.h>
```

Data Fields

- long [size](#)
- long [string_hash_size](#)
- long [free](#)
- long [mod](#)
- long [mul](#)
- char ** [string](#)
- void ** [data](#)
- long * [string_hash](#)
- long * [next_string](#)

1.25.1 Detailed Description

Definition at line 24 of file string_cache.h.

1.25.2 Field Documentation

1.25.2.1 data

`void** data`

Definition at line 31 of file `string_cache.h`.

1.25.2.2 free

`long free`

Definition at line 27 of file `string_cache.h`.

1.25.2.3 mod

`long mod`

Definition at line 28 of file `string_cache.h`.

1.25.2.4 mul

`long mul`

Definition at line 29 of file `string_cache.h`.

1.25.2.5 next_string

`long* next_string`

Definition at line 33 of file `string_cache.h`.

1.25.2.6 size

`long size`

Definition at line 25 of file `string_cache.h`.

1.25.2.7 string

```
char** string
```

Definition at line 30 of file string_cache.h.

1.25.2.8 string_hash

```
long* string_hash
```

Definition at line 32 of file string_cache.h.

1.25.2.9 string_hash_size

```
long string_hash_size
```

Definition at line 26 of file string_cache.h.

The documentation for this struct was generated from the following file:

- [vtr-verilog-to-routing/ODIN_II/SRC/STRING_HASH_TOOL/string_cache.h](#)

1.26 test_vector Struct Reference

```
#include <simulate_blif.h>
```

Data Fields

- signed char ** [values](#)
- int * [counts](#)
- int [count](#)

1.26.1 Detailed Description

Definition at line 93 of file simulate_blif.h.

1.26.2 Field Documentation

1.26.2.1 count

```
int count
```

Definition at line 96 of file simulate_blif.h.

1.26.2.2 counts

```
int* counts
```

Definition at line 95 of file simulate_blif.h.

1.26.2.3 values

```
signed char** values
```

Definition at line 94 of file simulate_blif.h.

The documentation for this struct was generated from the following file:

- vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/[simulate_blif.h](#)

1.27 veri_define Struct Reference

```
#include <verilog_preprocessor.h>
```

Data Fields

- char * [symbol](#)
- char * [value](#)
- int [line](#)
- [veri_include](#) * [defined_in](#)

1.27.1 Detailed Description

Definition at line 17 of file verilog_preprocessor.h.

1.27.2 Field Documentation

1.27.2.1 `defined_in`

```
veri_include* defined_in
```

Definition at line 22 of file verilog_preprocessor.h.

1.27.2.2 `line`

```
int line
```

Definition at line 21 of file verilog_preprocessor.h.

1.27.2.3 `symbol`

```
char* symbol
```

Definition at line 19 of file verilog_preprocessor.h.

1.27.2.4 `value`

```
char* value
```

Definition at line 20 of file verilog_preprocessor.h.

The documentation for this struct was generated from the following file:

- [vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL/verilog_preprocessor.h](#)

1.28 `veri_Defines` Struct Reference

```
#include <verilog_preprocessor.h>
```

Data Fields

- [veri_define](#) ** [defined_constants](#)
- int [current_size](#)
- int [current_index](#)

1.28.1 Detailed Description

Definition at line 32 of file verilog_preprocessor.h.

1.28.2 Field Documentation

1.28.2.1 current_index

```
int current_index
```

Definition at line 36 of file verilog_preprocessor.h.

1.28.2.2 current_size

```
int current_size
```

Definition at line 35 of file verilog_preprocessor.h.

1.28.2.3 defined_constants

```
veri_define** defined_constants
```

Definition at line 34 of file verilog_preprocessor.h.

The documentation for this struct was generated from the following file:

- [vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL/verilog_preprocessor.h](#)

1.29 veri_flag_node Struct Reference

```
#include <verilog_preprocessor.h>
```

Data Fields

- int [flag](#)
- struct [veri_flag_node](#) * [next](#)

1.29.1 Detailed Description

Definition at line 63 of file verilog_preprocessor.h.

1.29.2 Field Documentation

1.29.2.1 flag

```
int flag
```

Definition at line 65 of file verilog_preprocessor.h.

1.29.2.2 next

```
struct veri_flag_node* next
```

Definition at line 66 of file verilog_preprocessor.h.

The documentation for this struct was generated from the following file:

- [vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL/verilog_preprocessor.h](#)

1.30 veri_flag_stack Struct Reference

```
#include <verilog_preprocessor.h>
```

Data Fields

- [veri_flag_node](#) * [top](#)

1.30.1 Detailed Description

Definition at line 69 of file verilog_preprocessor.h.

1.30.2 Field Documentation

1.30.2.1 top

```
veri_flag_node* top
```

Definition at line 71 of file verilog_preprocessor.h.

The documentation for this struct was generated from the following file:

- [vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL/verilog_preprocessor.h](#)

1.31 veri_include Struct Reference

```
#include <verilog_preprocessor.h>
```

Data Fields

- char * [path](#)
- struct [veri_include](#) * [included_from](#)
- int [line](#)

1.31.1 Detailed Description

Definition at line 10 of file verilog_preprocessor.h.

1.31.2 Field Documentation

1.31.2.1 included_from

```
struct veri\_include* included_from
```

Definition at line 13 of file verilog_preprocessor.h.

1.31.2.2 line

```
int line
```

Definition at line 14 of file verilog_preprocessor.h.

1.31.2.3 path

```
char* path
```

Definition at line 12 of file verilog_preprocessor.h.

The documentation for this struct was generated from the following file:

- vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL/[verilog_preprocessor.h](#)

1.32 veri_Includes Struct Reference

```
#include <verilog_preprocessor.h>
```

Data Fields

- [veri_include](#) ** [included_files](#)
- int [current_size](#)
- int [current_index](#)

1.32.1 Detailed Description

Definition at line 25 of file verilog_preprocessor.h.

1.32.2 Field Documentation

1.32.2.1 current_index

```
int current_index
```

Definition at line 29 of file verilog_preprocessor.h.

1.32.2.2 current_size

```
int current_size
```

Definition at line 28 of file verilog_preprocessor.h.

1.32.2.3 included_files

```
veri_include** included_files
```

Definition at line 27 of file verilog_preprocessor.h.

The documentation for this struct was generated from the following file:

- vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL/[verilog_preprocessor.h](#)

2 File Documentation

2.1 vtr-verilog-to-routing/ODIN_II/SRC/DEBUG_TOOLS/netlist_visualizer.c File Reference

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "types.h"
#include "globals.h"
#include "netlist_utils.h"
#include "odin_util.h"
#include "vtr_memory.h"
```

Functions

- void [depth_first_traverse_visualize](#) (nnode_t *node, FILE *fp, int traverse_mark_number)
- void [depth_first_traversal_graph_display](#) (FILE *out, short marker_value, netlist_t *netlist)
- void [forward_traversal_net_graph_display](#) (FILE *out, short marker_value, nnode_t *node)
- void [backward_traversal_net_graph_display](#) (FILE *out, short marker_value, nnode_t *node)
- void [graphVizOutputNetlist](#) (char *path, const char *name, short marker_value, netlist_t *netlist)
- void [graphVizOutputCombinationalNet](#) (char *path, const char *name, short marker_value, nnode_t *current_node)

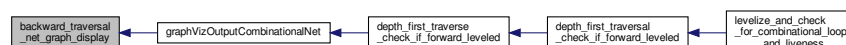
2.1.1 Function Documentation

2.1.1.1 backward_traversal_net_graph_display()

```
void backward_traversal_net_graph_display (
    FILE * out,
    short marker_value,
    nnode_t * node )
```

Definition at line 311 of file netlist_visualizer.c.

Here is the caller graph for this function:

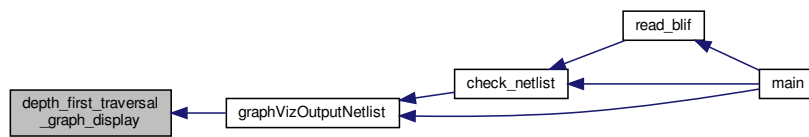


2.1.1.2 depth_first_traversal_graph_display()

```
void depth_first_traversal_graph_display (
    FILE * out,
    short marker_value,
    netlist_t * netl1ist )
```

Definition at line 68 of file netlist_visualizer.c.

Here is the caller graph for this function:

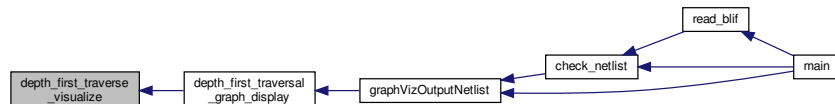


2.1.1.3 depth_first_traverse_visualize()

```
void depth_first_traverse_visualize (
    nnode_t * node,
    FILE * fp,
    int traverse_mark_number )
```

Definition at line 90 of file netlist_visualizer.c.

Here is the caller graph for this function:

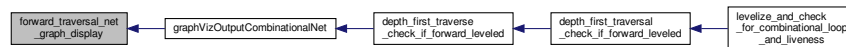


2.1.1.4 forward_traversal_net_graph_display()

```
void forward_traversal_net_graph_display (
    FILE * out,
    short marker_value,
    nnode_t * node )
```

Definition at line 208 of file netlist_visualizer.c.

Here is the caller graph for this function:



2.1.1.5 graphVizOutputCombinationalNet()

```
void graphVizOutputCombinationalNet (
    char * path,
    const char * name,
    short marker_value,
    nnode_t * current_node )
```

Definition at line 184 of file netlist_visualizer.c.

Here is the caller graph for this function:

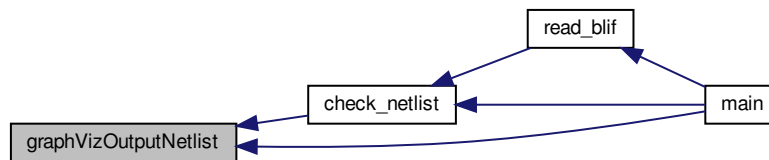


2.1.1.6 graphVizOutputNetlist()

```
void graphVizOutputNetlist (
    char * path,
    const char * name,
    short marker_value,
    netlist_t * netlist )
```

Definition at line 46 of file netlist_visualizer.c.

Here is the caller graph for this function:



2.2 vtr-verilog-to-routing/ODIN_II/SRC/DEBUG_TOOLS/netlist_visualizer.cpp File Reference

```

#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "types.h"
#include "globals.h"
#include "netlist_utils.h"
#include "odin_util.h"
#include "vtr_memory.h"

```

Functions

- void [depth_first_traverse_visualize](#) (nnode_t *node, FILE *fp, int traverse_mark_number)
- void [depth_first_traversal_graph_display](#) (FILE *out, short marker_value, netlist_t *netlist)
- void [forward_traversal_net_graph_display](#) (FILE *out, short marker_value, nnode_t *node)
- void [backward_traversal_net_graph_display](#) (FILE *out, short marker_value, nnode_t *node)
- void [graphVizOutputNetlist](#) (std::string path, const char *name, short marker_value, netlist_t *netlist)
- void [graphVizOutputCombinationalNet](#) (std::string path, const char *name, short marker_value, nnode_t *current_node)

2.2.1 Function Documentation

2.2.1.1 backward_traversal_net_graph_display()

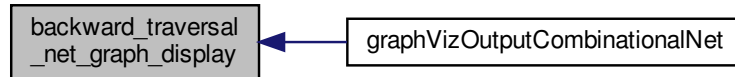
```

void backward_traversal_net_graph_display (
    FILE * out,
    short marker_value,
    nnode_t * node )

```

Definition at line 311 of file netlist_visualizer.cpp.

Here is the caller graph for this function:



2.2.1.2 depth_first_traversal_graph_display()

```
void depth_first_traversal_graph_display (  
    FILE * out,  
    short marker_value,  
    netlist_t * netllist )
```

Definition at line 68 of file `netlist_visualizer.cpp`.

Here is the caller graph for this function:

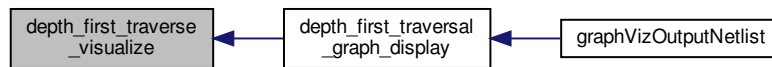


2.2.1.3 depth_first_traverse_visualize()

```
void depth_first_traverse_visualize (  
    nnode_t * node,  
    FILE * fp,  
    int traverse_mark_number )
```

Definition at line 90 of file `netlist_visualizer.cpp`.

Here is the caller graph for this function:



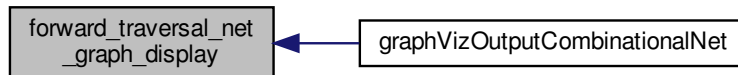
2.2.1.4 forward_traversal_net_graph_display()

```

void forward_traversal_net_graph_display (
    FILE * out,
    short marker_value,
    nnode_t * node )
  
```

Definition at line 208 of file netlist_visualizer.cpp.

Here is the caller graph for this function:



2.2.1.5 graphVizOutputCombinationalNet()

```

void graphVizOutputCombinationalNet (
    std::string path,
    const char * name,
    short marker_value,
    nnode_t * current_node )
  
```

Definition at line 184 of file netlist_visualizer.cpp.

2.2.1.6 graphVizOutputNetlist()

```
void graphVizOutputNetlist (
    std::string path,
    const char * name,
    short marker_value,
    netlist_t * netlist )
```

Definition at line 46 of file netlist_visualizer.cpp.

2.3 vtr-verilog-to-routing/ODIN_II/SRC/DEBUG_TOOLS/netlist_visualizer.h File Reference

```
#include "types.h"
```

Functions

- void [graphVizOutputNetlist](#) (std::string path, const char *name, short marker_value, netlist_t *netlist)
- void [graphVizOutputCombinationalNet](#) (std::string path, const char *name, short marker_value, nnode_t *current_node)

2.3.1 Function Documentation

2.3.1.1 graphVizOutputCombinationalNet()

```
void graphVizOutputCombinationalNet (
    std::string path,
    const char * name,
    short marker_value,
    nnode_t * current_node )
```

Definition at line 184 of file netlist_visualizer.cpp.

2.3.1.2 graphVizOutputNetlist()

```
void graphVizOutputNetlist (
    std::string path,
    const char * name,
    short marker_value,
    netlist_t * netlist )
```

Definition at line 46 of file netlist_visualizer.cpp.

2.4 vtr-verilog-to-routing/ODIN_II/SRC/DEBUG_TOOLS/print_netlist.c File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "globals.h"
#include "types.h"
#include "netlist_utils.h"
#include "print_netlist.h"
```

Functions

- void [function_to_print_node_and_its_pin](#) (npin_t *)
- void [print_netlist_for_checking](#) (netlist_t *netlist, char *name)

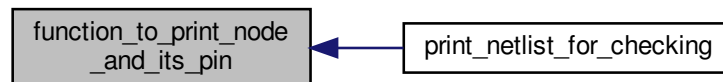
2.4.1 Function Documentation

2.4.1.1 [function_to_print_node_and_its_pin\(\)](#)

```
void function_to_print_node_and_its_pin (
    npin_t * temp_pin )
```

Definition at line 298 of file `print_netlist.c`.

Here is the caller graph for this function:



2.4.1.2 [print_netlist_for_checking\(\)](#)

```
void print_netlist_for_checking (
    netlist_t * netlist,
    char * name )
```

Definition at line 37 of file `print_netlist.c`.

2.5 vtr-verilog-to-routing/ODIN_II/SRC/DEBUG_TOOLS/print_netlist.cpp File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "globals.h"
#include "types.h"
#include "netlist_utils.h"
#include "print_netlist.h"
```

Functions

- void [function_to_print_node_and_its_pin](#) (npin_t *)
- void [print_netlist_for_checking](#) (netlist_t *netlist, char *name)

2.5.1 Function Documentation

2.5.1.1 [function_to_print_node_and_its_pin\(\)](#)

```
void function_to_print_node_and_its_pin (
    npin_t * temp_pin )
```

Definition at line 298 of file print_netlist.cpp.

Here is the caller graph for this function:



2.5.1.2 [print_netlist_for_checking\(\)](#)

```
void print_netlist_for_checking (
    netlist_t * netlist,
    char * name )
```

Definition at line 37 of file print_netlist.cpp.

2.6 vtr-verilog-to-routing/ODIN_II/SRC/DEBUG_TOOLS/print_netlist.h File Reference

Functions

- void [print_netlist_for_checking](#) (netlist_t *netlist, char *name)

2.6.1 Function Documentation

2.6.1.1 print_netlist_for_checking()

```
void print_netlist_for_checking (
    netlist_t * netlist,
    char * name )
```

Definition at line 37 of file print_netlist.c.

2.7 vtr-verilog-to-routing/ODIN_II/SRC/globals.h File Reference

```
#include "types.h"
#include "string_cache.h"
#include "read_xml_arch_file.h"
```

Variables

- t_type_descriptor * [type_descriptors](#)
- int [yylineno](#)
- short [to_view_parse](#)
- global_args_t [global_args](#)
- config_t [configuration](#)
- size_t [current_parse_file](#)
- size_t [num_modules](#)
- ast_node_t ** [ast_modules](#)
- STRING_CACHE * [module_names_to_idx](#)
- STRING_CACHE * [output_nets_sc](#)
- STRING_CACHE * [input_nets_sc](#)
- STRING_CACHE * [local_symbol_table_sc](#)
- STRING_CACHE * [global_param_table_sc](#)
- STRING_CACHE * [function_local_symbol_table_sc](#)
- netlist_t * [verilog_netlist](#)
- ast_node_t * [top_module](#)
- nnode_t ** [top_input_nodes](#)
- size_t [num_top_input_nodes](#)
- nnode_t ** [top_output_nodes](#)
- size_t [num_top_output_nodes](#)

- `nnode_t` * [gnd_node](#)
- `nnode_t` * [vcc_node](#)
- `nnode_t` * [pad_node](#)
- `nnet_t` * [zero_net](#)
- `nnet_t` * [one_net](#)
- `nnet_t` * [pad_net](#)
- `char` * [one_string](#)
- `char` * [zero_string](#)
- `char` * [pad_string](#)
- `t_arch` [Arch](#)
- `netlist_t` * [blif_netlist](#)
- `netlist_t` * [read_blif_netlist](#)
- `global_args_read_blif_t` [global_args_read_blif](#)
- `size_t` [file_line_number](#)

2.7.1 Variable Documentation

2.7.1.1 Arch

`t_arch Arch`

Definition at line 58 of file `odin_ii.cpp`.

2.7.1.2 ast_modules

`ast_node_t** ast_modules`

Definition at line 68 of file `parse_making_ast.cpp`.

2.7.1.3 blif_netlist

`netlist_t* blif_netlist`

2.7.1.4 configuration

`config_t configuration`

Definition at line 38 of file `read_xml_config_file.cpp`.

2.7.1.5 current_parse_file

`size_t current_parse_file`

Definition at line 57 of file `odin_ii.cpp`.

2.7.1.6 file_line_number

`size_t file_line_number`

Definition at line 47 of file `read_blif.cpp`.

2.7.1.7 function_local_symbol_table_sc

`STRING_CACHE* function_local_symbol_table_sc`

Definition at line 64 of file `netlist_create_from_ast.cpp`.

2.7.1.8 global_args

`global_args_t global_args`

Definition at line 59 of file `odin_ii.cpp`.

2.7.1.9 global_args_read_blif

`global_args_read_blif_t global_args_read_blif`

2.7.1.10 global_param_table_sc

`STRING_CACHE* global_param_table_sc`

Definition at line 65 of file `netlist_create_from_ast.cpp`.

2.7.1.11 gnd_node

`nnode_t* gnd_node`

2.7.1.12 input_nets_sc

```
STRING_CACHE* input_nets_sc
```

Definition at line 61 of file netlist_create_from_ast.cpp.

2.7.1.13 local_symbol_table_sc

```
STRING_CACHE* local_symbol_table_sc
```

Definition at line 63 of file netlist_create_from_ast.cpp.

2.7.1.14 module_names_to_idx

```
STRING_CACHE* module_names_to_idx
```

Definition at line 53 of file parse_making_ast.cpp.

2.7.1.15 num_modules

```
size_t num_modules
```

Definition at line 67 of file parse_making_ast.cpp.

2.7.1.16 num_top_input_nodes

```
size_t num_top_input_nodes
```

2.7.1.17 num_top_output_nodes

```
size_t num_top_output_nodes
```

2.7.1.18 one_net

```
nnet_t* one_net
```


2.7.1.19 one_string

```
char* one_string
```

Definition at line 75 of file netlist_create_from_ast.cpp.

2.7.1.20 output_nets_sc

```
STRING_CACHE* output_nets_sc
```

Definition at line 60 of file netlist_create_from_ast.cpp.

2.7.1.21 pad_net

```
nnet_t* pad_net
```

2.7.1.22 pad_node

```
nnode_t* pad_node
```

2.7.1.23 pad_string

```
char* pad_string
```

Definition at line 77 of file netlist_create_from_ast.cpp.

2.7.1.24 read_blif_netlist

```
netlist_t* read_blif_netlist
```

2.7.1.25 to_view_parse

```
short to_view_parse
```

Definition at line 75 of file parse_making_ast.cpp.

2.7.1.26 top_input_nodes

```
nnode_t** top_input_nodes
```

2.7.1.27 top_module

```
ast_node_t* top_module
```

Definition at line 79 of file netlist_create_from_ast.cpp.

2.7.1.28 top_output_nodes

```
nnode_t** top_output_nodes
```

2.7.1.29 type_descriptors

```
t_type_descriptor* type_descriptors
```

Definition at line 60 of file odin_ii.cpp.

2.7.1.30 vcc_node

```
nnode_t* vcc_node
```

2.7.1.31 verilog_netlist

```
netlist_t* verilog_netlist
```

Definition at line 81 of file netlist_create_from_ast.cpp.

2.7.1.32 yylineno

```
int yylineno
```

2.7.1.33 zero_net

```
nnet_t* zero_net
```

2.7.1.34 zero_string

```
char* zero_string
```

Definition at line 76 of file netlist_create_from_ast.cpp.

2.8 vtr-verilog-to-routing/ODIN_II/SRC/lib_ast/ast_elaborate.cpp File Reference

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "types.h"
#include "ast_util.h"
#include "ast_elaborate.h"
#include "parse_making_ast.h"
#include "verilog_bison.h"
#include "netlist_create_from_ast.h"
#include "ctype.h"
#include "vtr_util.h"
#include "vtr_memory.h"
#include <string>
#include <iostream>
#include <vector>
#include <stack>
```

Macros

- #define [read_node](#) 1
- #define [write_node](#) 2
- #define [e_data](#) 1
- #define [e_operation](#) 2
- #define [e_variable](#) 3
- #define [N](#) 64
- #define [Max_size](#) 64

Functions

- int [simplify_ast](#) ()
- void [optimize_for_tree](#) ()
- void [search_for_node](#) (ast_node_t *root, std::vector< ast_node_t *> list_for_node, std::vector< ast_node_t *> list_parent)
- ast_node_t * [get_copy_tree](#) (ast_node_t *node, long long virtual_value, std::string virtual_name)
- void [record_expression](#) (ast_node_t *node, std::vector< std::string > expressions, bool *found_statement)
- long long [calculation](#) (std::vector< std::string > post_exp)
- void [translate_expression](#) (std::vector< std::string > infix_exp, std::vector< std::string > postfix_exp)
- void [reallocate_node](#) (ast_node_t *node, int idx)
- void [find_most_unique_count](#) ()
- void [mark_node_write](#) (ast_node_t *node, std::vector< std::string > list)
- void [mark_node_read](#) (ast_node_t *node, std::vector< std::string > list)
- void [remove_intermediate_variable](#) (ast_node_t *node, std::vector< std::string > list, long long virtual_value, std::string virtual_name)
- ast_node_t * [search_marked_node](#) (ast_node_t *node, int is, std::string temp)
- void [reduce_assignment_expression](#) ()
- void [find_assign_node](#) (ast_node_t *node, std::vector< ast_node_t *> list)
- bool [simplify_expression](#) ()
- enode * [find_tail](#) (enode *node)
- void [reduce_enode_list](#) ()
- void [store_exp_list](#) (ast_node_t *node)
- void [record_tree_info](#) (ast_node_t *node)
- void [create_enode](#) (ast_node_t *node)
- bool [adjoin_constant](#) ()
- enode * [replace_enode](#) (int data, enode *t, int mark)
- bool [combine_constant](#) ()
- void [construct_new_tree](#) (enode *tail, ast_node_t *node, int line_num, int file_num)
- int [check_exp_list](#) (enode *tail)
- bool [delete_continuous_multiply](#) ()
- void [create_ast_node](#) (enode *temp, ast_node_t *node, int line_num, int file_num)
- void [create_op_node](#) (ast_node_t *node, enode *temp, int line_num, int file_num)
- void [free_exp_list](#) ()
- bool [deal_with_bracket](#) (ast_node_t *node)
- bool [check_mult_bracket](#) (std::vector< int > list)
- void [recursive_tree](#) (ast_node_t *node, std::vector< int > list_bracket)
- void [find_leaf_node](#) (ast_node_t *node, std::vector< int > list_bracket, int ids2)
- void [delete_bracket](#) (int begin, int end)
- void [delete_bracket_head](#) (enode *begin, enode *end)
- void [change_exp_list](#) (enode *begin, enode *end, enode *s, int flag)
- enode * [copy_enode_list](#) (enode *new_head, enode *begin, enode *end)
- void [copy_enode](#) (enode *node, enode *new_node)
- void [delete_bracket_tail](#) (enode *begin, enode *end)
- void [delete_bracket_body](#) (enode *begin, enode *end)
- bool [check_tree_operation](#) (ast_node_t *node)
- void [check_operation](#) (enode *begin, enode *end)
- void [reduce_parameter](#) ()
- void [find_parameter](#) (ast_node_t *node, std::vector< ast_node_t *> para)
- void [remove_para_node](#) (ast_node_t *top, std::vector< ast_node_t *> para)
- void [change_para_node](#) (ast_node_t *node, std::string name, long long value)

- void `shift_operation` ()
- void `search_certain_operation` (ast_node_t *node)
- void `check_binary_operation` (ast_node_t *node)
- void `check_node_number` (ast_node_t *parent, ast_node_t *child, int flag)
- short `has_intermediate_variable` (ast_node_t *node)
- void `keep_all_branch` (ast_node_t *temp_node, ast_node_t *for_parent, int mark)

Variables

- size_t `count_id` = 0
- size_t `count_assign` = 0
- size_t `count`
- size_t `count_write`
- `enode` * `head`
- `enode` * `p`

2.8.1 Macro Definition Documentation

2.8.1.1 e_data

```
#define e_data 1
```

Definition at line 46 of file `ast_elaborate.cpp`.

2.8.1.2 e_operation

```
#define e_operation 2
```

Definition at line 47 of file `ast_elaborate.cpp`.

2.8.1.3 e_variable

```
#define e_variable 3
```

Definition at line 48 of file `ast_elaborate.cpp`.

2.8.1.4 Max_size

```
#define Max_size 64
```

Definition at line 51 of file ast_elaborate.cpp.

2.8.1.5 N

```
#define N 64
```

Definition at line 50 of file ast_elaborate.cpp.

2.8.1.6 read_node

```
#define read_node 1
```

Definition at line 43 of file ast_elaborate.cpp.

2.8.1.7 write_node

```
#define write_node 2
```

Definition at line 44 of file ast_elaborate.cpp.

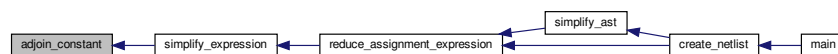
2.8.2 Function Documentation

2.8.2.1 adjoin_constant()

```
bool adjoin_constant ( )
```

Definition at line 692 of file ast_elaborate.cpp.

Here is the caller graph for this function:

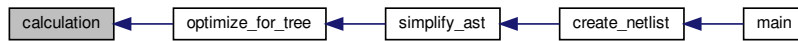


2.8.2.2 calculation()

```
long long calculation (
    std::vector< std::string > post_exp )
```

Definition at line 261 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.8.2.3 change_exp_list()

```
void change_exp_list (
    enode * begin,
    enode * end,
    enode * s,
    int flag )
```

Definition at line 1226 of file ast_elaborate.cpp.

Here is the caller graph for this function:

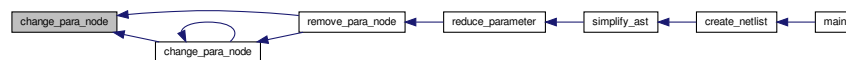


2.8.2.4 change_para_node()

```
void change_para_node (
    ast_node_t * node,
    std::string name,
    long long value )
```

Definition at line 1486 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.8.2.5 check_binary_operation()

```
void check_binary_operation (
    ast_node_t * node )
```

Definition at line 1526 of file ast_elaborate.cpp.

Here is the caller graph for this function:

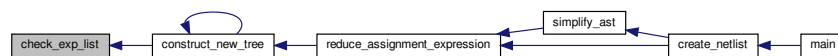


2.8.2.6 check_exp_list()

```
int check_exp_list (
    enode * tail )
```

Definition at line 939 of file ast_elaborate.cpp.

Here is the caller graph for this function:

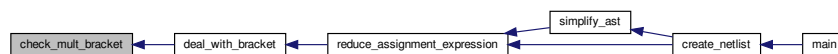


2.8.2.7 check_mult_bracket()

```
bool check_mult_bracket (
    std::vector< int > list )
```

Definition at line 1109 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.8.2.8 check_node_number()

```
void check_node_number (
    ast_node_t * parent,
    ast_node_t * child,
    int flag )
```

Definition at line 1550 of file ast_elaborate.cpp.

Here is the caller graph for this function:

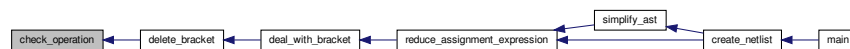


2.8.2.9 check_operation()

```
void check_operation (
    enode * begin,
    enode * end )
```

Definition at line 1403 of file ast_elaborate.cpp.

Here is the caller graph for this function:

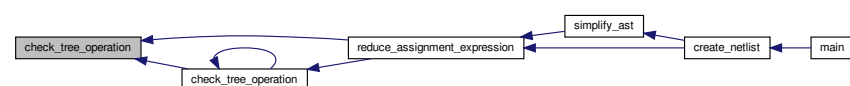


2.8.2.10 check_tree_operation()

```
bool check_tree_operation (
    ast_node_t * node )
```

Definition at line 1388 of file ast_elaborate.cpp.

Here is the caller graph for this function:

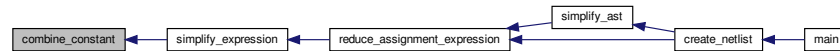


2.8.2.11 combine_constant()

```
bool combine_constant ( )
```

Definition at line 804 of file ast_elaborate.cpp.

Here is the caller graph for this function:

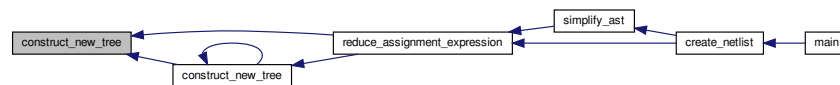


2.8.2.12 construct_new_tree()

```
void construct_new_tree (
    enode * tail,
    ast_node_t * node,
    int line_num,
    int file_num )
```

Definition at line 883 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.8.2.13 copy_enode()

```
void copy_enode (
    enode * node,
    enode * new_node )
```

Definition at line 1338 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.8.2.14 copy_enode_list()

```
enode* copy_enode_list (
    enode * new_head,
    enode * begin,
    enode * end )
```

Definition at line 1319 of file ast_elaborate.cpp.

Here is the caller graph for this function:

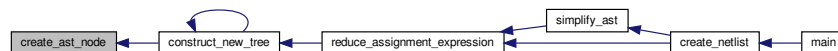


2.8.2.15 create_ast_node()

```
void create_ast_node (
    enode * temp,
    ast_node_t * node,
    int line_num,
    int file_num )
```

Definition at line 1019 of file ast_elaborate.cpp.

Here is the caller graph for this function:

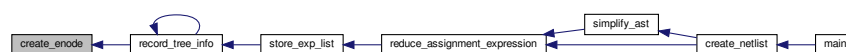


2.8.2.16 create_enode()

```
void create_enode (
    ast_node_t * node )
```

Definition at line 622 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.8.2.17 create_op_node()

```

void create_op_node (
    ast_node_t * node,
    enode * temp,
    int line_num,
    int file_num )

```

Definition at line 1046 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.8.2.18 deal_with_bracket()

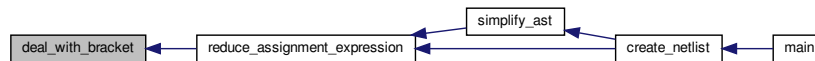
```

bool deal_with_bracket (
    ast_node_t * node )

```

Definition at line 1090 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.8.2.19 delete_bracket()

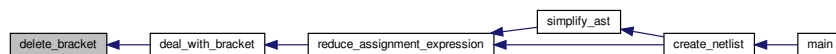
```

void delete_bracket (
    int begin,
    int end )

```

Definition at line 1170 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.8.2.20 delete_bracket_body()

```
void delete_bracket_body (
    enode * begin,
    enode * end )
```

Definition at line 1364 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.8.2.21 delete_bracket_head()

```
void delete_bracket_head (
    enode * begin,
    enode * end )
```

Definition at line 1206 of file ast_elaborate.cpp.

Here is the caller graph for this function:

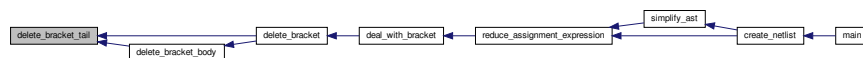


2.8.2.22 delete_bracket_tail()

```
void delete_bracket_tail (
    enode * begin,
    enode * end )
```

Definition at line 1349 of file ast_elaborate.cpp.

Here is the caller graph for this function:

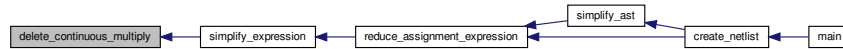


2.8.2.23 delete_continuous_multiply()

```
bool delete_continuous_multiply ( )
```

Definition at line 956 of file ast_elaborate.cpp.

Here is the caller graph for this function:

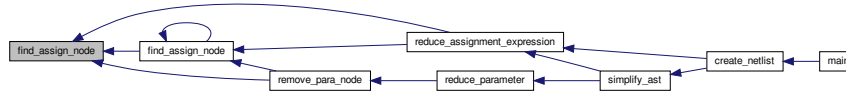


2.8.2.24 find_assign_node()

```
void find_assign_node (
    ast_node_t * node,
    std::vector< ast_node_t *> list )
```

Definition at line 482 of file ast_elaborate.cpp.

Here is the caller graph for this function:

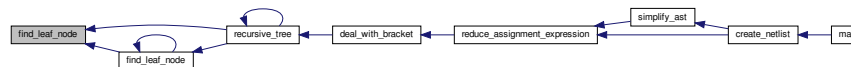


2.8.2.25 find_leaf_node()

```
void find_leaf_node (
    ast_node_t * node,
    std::vector< int > list_bracket,
    int ids2 )
```

Definition at line 1156 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.8.2.26 find_most_unique_count()

```
void find_most_unique_count ( )
```

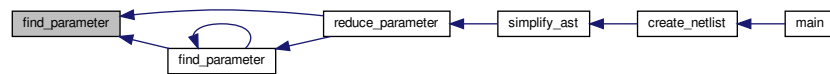
Definition at line 340 of file ast_elaborate.cpp.

2.8.2.27 find_parameter()

```
void find_parameter (
    ast_node_t * node,
    std::vector< ast_node_t *> para )
```

Definition at line 1444 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.8.2.28 find_tail()

```
enode* find_tail (
    enode * node )
```

Definition at line 513 of file ast_elaborate.cpp.

Here is the caller graph for this function:

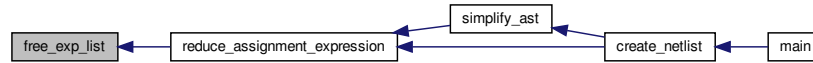


2.8.2.29 free_exp_list()

```
void free_exp_list ( )
```

Definition at line 1077 of file ast_elaborate.cpp.

Here is the caller graph for this function:

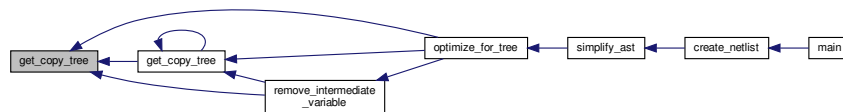


2.8.2.30 get_copy_tree()

```
ast_node_t* get_copy_tree (
    ast_node_t * node,
    long long virtual_value,
    std::string virtual_name )
```

Definition at line 177 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.8.2.31 has_intermediate_variable()

```
short has_intermediate_variable (
    ast_node_t * node )
```

Definition at line 1582 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.8.2.32 keep_all_branch()

```
void keep_all_branch (
    ast_node_t * temp_node,
    ast_node_t * for_parent,
    int mark )
```

Definition at line 1599 of file ast_elaborate.cpp.

Here is the caller graph for this function:

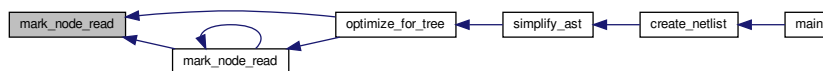


2.8.2.33 mark_node_read()

```
void mark_node_read (
    ast_node_t * node,
    std::vector< std::string > list )
```

Definition at line 367 of file ast_elaborate.cpp.

Here is the caller graph for this function:

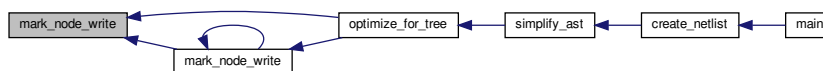


2.8.2.34 mark_node_write()

```
void mark_node_write (
    ast_node_t * node,
    std::vector< std::string > list )
```

Definition at line 351 of file ast_elaborate.cpp.

Here is the caller graph for this function:

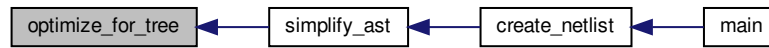


2.8.2.35 optimize_for_tree()

```
void optimize_for_tree ( )
```

Definition at line 80 of file ast_elaborate.cpp.

Here is the caller graph for this function:

**2.8.2.36 reallocate_node()**

```
void reallocate_node (
    ast_node_t * node,
    int idx )
```

Definition at line 326 of file ast_elaborate.cpp.

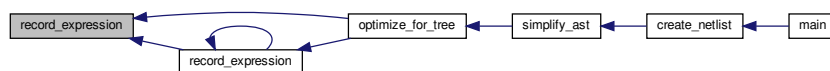
Here is the caller graph for this function:

**2.8.2.37 record_expression()**

```
void record_expression (
    ast_node_t * node,
    std::vector< std::string > expressions,
    bool * found_statement )
```

Definition at line 205 of file ast_elaborate.cpp.

Here is the caller graph for this function:

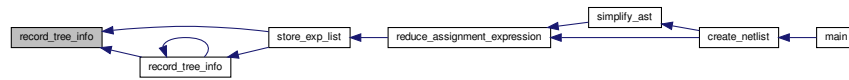


2.8.2.38 record_tree_info()

```
void record_tree_info (
    ast_node_t * node )
```

Definition at line 603 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.8.2.39 recursive_tree()

```
void recursive_tree (
    ast_node_t * node,
    std::vector< int > list_bracket )
```

Definition at line 1133 of file ast_elaborate.cpp.

Here is the caller graph for this function:

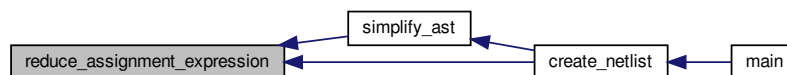


2.8.2.40 reduce_assignment_expression()

```
void reduce_assignment_expression ( )
```

Definition at line 439 of file ast_elaborate.cpp.

Here is the caller graph for this function:

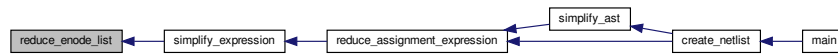


2.8.2.41 reduce_enode_list()

```
void reduce_enode_list ( )
```

Definition at line 527 of file ast_elaborate.cpp.

Here is the caller graph for this function:

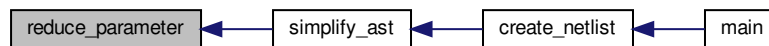


2.8.2.42 reduce_parameter()

```
void reduce_parameter ( )
```

Definition at line 1428 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.8.2.43 remove_intermediate_variable()

```
void remove_intermediate_variable (
    ast_node_t * node,
    std::vector< std::string > list,
    long long virtual_value,
    std::string virtual_name )
```

Definition at line 382 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.8.2.44 remove_para_node()

```
void remove_para_node (
    ast_node_t * top,
    std::vector< ast_node_t *> para )
```

Definition at line 1458 of file ast_elaborate.cpp.

Here is the caller graph for this function:

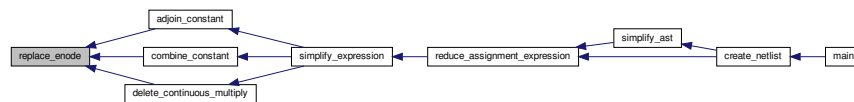


2.8.2.45 replace_enode()

```
enode* replace_enode (
    int data,
    enode * t,
    int mark )
```

Definition at line 760 of file ast_elaborate.cpp.

Here is the caller graph for this function:

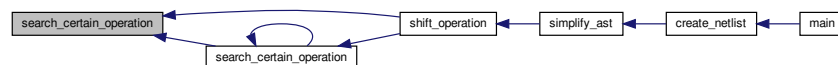


2.8.2.46 search_certain_operation()

```
void search_certain_operation (
    ast_node_t * node )
```

Definition at line 1514 of file ast_elaborate.cpp.

Here is the caller graph for this function:

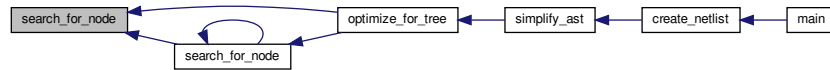


2.8.2.47 search_for_node()

```
void search_for_node (
    ast_node_t * root,
    std::vector< ast_node_t *> list_for_node,
    std::vector< ast_node_t *> list_parent )
```

Definition at line 158 of file ast_elaborate.cpp.

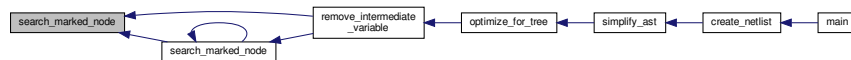
Here is the caller graph for this function:

**2.8.2.48 search_marked_node()**

```
ast_node_t* search_marked_node (
    ast_node_t * node,
    int is,
    std::string temp )
```

Definition at line 419 of file ast_elaborate.cpp.

Here is the caller graph for this function:

**2.8.2.49 shift_operation()**

```
void shift_operation ( )
```

Definition at line 1499 of file ast_elaborate.cpp.

Here is the caller graph for this function:

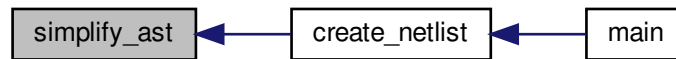


2.8.2.50 `simplify_ast()`

```
int simplify_ast ( )
```

Definition at line 59 of file `ast_elaborate.cpp`.

Here is the caller graph for this function:

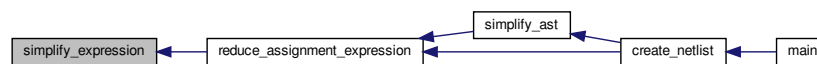


2.8.2.51 `simplify_expression()`

```
bool simplify_expression ( )
```

Definition at line 495 of file `ast_elaborate.cpp`.

Here is the caller graph for this function:



2.8.2.52 `store_exp_list()`

```
void store_exp_list (
    ast_node_t * node )
```

Definition at line 586 of file `ast_elaborate.cpp`.

Here is the caller graph for this function:



2.8.2.53 translate_expression()

```
void translate_expression (
    std::vector< std::string > infix_exp,
    std::vector< std::string > postfix_exp )
```

Definition at line 297 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.8.3 Variable Documentation

2.8.3.1 count

```
size_t count
```

Definition at line 55 of file ast_elaborate.cpp.

2.8.3.2 count_assign

```
size_t count_assign = 0
```

Definition at line 54 of file ast_elaborate.cpp.

2.8.3.3 count_id

```
size_t count_id = 0
```

Definition at line 53 of file ast_elaborate.cpp.

2.8.3.4 count_write

```
size_t count_write
```

Definition at line 56 of file ast_elaborate.cpp.

2.8.3.5 head

```
enode* head
```

Definition at line 57 of file ast_elaborate.cpp.

2.8.3.6 p

```
enode * p
```

Definition at line 57 of file ast_elaborate.cpp.

2.9 vtr-verilog-to-routing/ODIN_II/SRC/lib_ast/ast_elaborate.h File Reference

Data Structures

- struct [exp_node](#)

Typedefs

- typedef struct [exp_node](#) [enode](#)

Functions

- int [simplify_ast](#) ()
- void [optimize_for_tree](#) ()
- void [search_for_node](#) (ast_node_t *root, std::vector< ast_node_t *> list_for_node, std::vector< ast_node_t *> list_parent)
- ast_node_t * [get_copy_tree](#) (ast_node_t *node, long long virtual_value, std::string virtual_name)
- void [record_expression](#) (ast_node_t *node, std::vector< std::string > expressions, bool *found_statement)
- long long [calculation](#) (std::vector< std::string > post_exp)
- void [translate_expression](#) (std::vector< std::string > infix_exp, std::vector< std::string > postfix_exp)
- void [reallocate_node](#) (ast_node_t *node, int idx)
- void [find_most_unique_count](#) ()
- void [mark_node_write](#) (ast_node_t *node, std::vector< std::string > list)
- void [mark_node_read](#) (ast_node_t *node, std::vector< std::string > list)
- void [remove_intermediate_variable](#) (ast_node_t *node, std::vector< std::string > list, long long virtual_value, std::string virtual_name)
- ast_node_t * [search_marked_node](#) (ast_node_t *node, int is, std::string temp)
- void [reduce_assignment_expression](#) ()
- void [find_assign_node](#) (ast_node_t *t, std::vector< ast_node_t *> list)
- ast_node_t * [find_top_module](#) ()
- void [record_tree_info](#) (ast_node_t *node)
- void [store_exp_list](#) (ast_node_t *node)
- void [create_enode](#) (ast_node_t *node)
- bool [simplify_expression](#) ()

- bool `adjoin_constant` ()
- `enode` * `replace_enode` (int data, `enode` *t, int mark)
- bool `combine_constant` ()
- bool `delete_continuous_multiply` ()
- void `construct_new_tree` (`enode` *tail, `ast_node_t` *node, int line_num, int file_num)
- void `reduce_enode_list` ()
- `enode` * `find_tail` (`enode` *node)
- int `check_exp_list` (`enode` *tail)
- void `create_ast_node` (`enode` *temp, `ast_node_t` *node, int line_num, int file_num)
- void `create_op_node` (`ast_node_t` *node, `enode` *temp, int line_num, int file_num)
- void `free_exp_list` ()
- bool `deal_with_bracket` (`ast_node_t` *node)
- void `recursive_tree` (`ast_node_t` *node, std::vector< int > list_bracket)
- void `find_leaf_node` (`ast_node_t` *node, std::vector< int > list_bracket, int ids2)
- void `delete_bracket` (int beign, int end)
- void `delete_bracket_head` (`enode` *begin, `enode` *end)
- void `change_exp_list` (`enode` *beign, `enode` *end, `enode` *s, int flag)
- `enode` * `copy_enode_list` (`enode` *new_head, `enode` *begin, `enode` *end)
- void `copy_enode` (`enode` *node, `enode` *new_node)
- void `delete_bracket_tail` (`enode` *begin, `enode` *end)
- void `delete_bracket_body` (`enode` *begin, `enode` *end)
- bool `check_tree_operation` (`ast_node_t` *node)
- void `reduce_parameter` ()
- void `find_parameter` (`ast_node_t` *top, std::vector< `ast_node_t` *> para)
- void `remove_para_node` (`ast_node_t` *top, std::vector< `ast_node_t` *> para)
- void `change_para_node` (`ast_node_t` *node, std::string name, long long value)
- void `check_operation` (`enode` *begin, `enode` *end)
- void `shift_operation` ()
- void `search_certain_operation` (`ast_node_t` *node)
- void `check_binary_operation` (`ast_node_t` *node)
- void `check_node_number` (`ast_node_t` *parent, `ast_node_t` *child, int flag)
- bool `check_mult_bracket` (std::vector< int > list)
- short `has_intermediate_variable` (`ast_node_t` *node)
- void `keep_all_branch` (`ast_node_t` *temp_node, `ast_node_t` *for_parent, int mark)

2.9.1 Typedef Documentation

2.9.1.1 `enode`

```
typedef struct exp_node enode
```

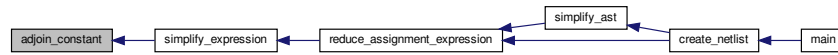
2.9.2 Function Documentation

2.9.2.1 `adjoin_constant()`

```
bool adjoin_constant ( )
```

Definition at line 692 of file `ast_elaborate.cpp`.

Here is the caller graph for this function:

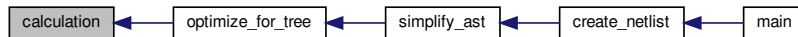


2.9.2.2 `calculation()`

```
long long calculation (
    std::vector< std::string > post_exp )
```

Definition at line 261 of file `ast_elaborate.cpp`.

Here is the caller graph for this function:



2.9.2.3 `change_exp_list()`

```
void change_exp_list (
    enode * begin,
    enode * end,
    enode * s,
    int flag )
```

Definition at line 1226 of file `ast_elaborate.cpp`.

Here is the caller graph for this function:

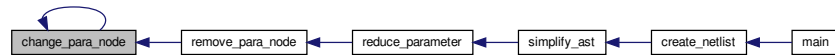


2.9.2.4 change_para_node()

```
void change_para_node (
    ast_node_t * node,
    std::string name,
    long long value )
```

Definition at line 1486 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.9.2.5 check_binary_operation()

```
void check_binary_operation (
    ast_node_t * node )
```

Definition at line 1526 of file ast_elaborate.cpp.

Here is the caller graph for this function:

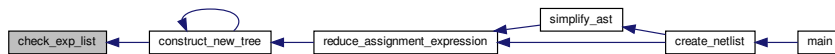


2.9.2.6 check_exp_list()

```
int check_exp_list (
    enode * tail )
```

Definition at line 939 of file ast_elaborate.cpp.

Here is the caller graph for this function:

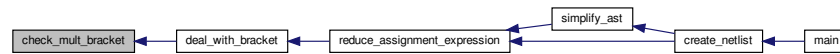


2.9.2.7 check_mult_bracket()

```
bool check_mult_bracket (
    std::vector< int > list )
```

Definition at line 1109 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.9.2.8 check_node_number()

```
void check_node_number (
    ast_node_t * parent,
    ast_node_t * child,
    int flag )
```

Definition at line 1550 of file ast_elaborate.cpp.

Here is the caller graph for this function:

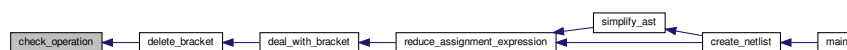


2.9.2.9 check_operation()

```
void check_operation (
    enode * begin,
    enode * end )
```

Definition at line 1403 of file ast_elaborate.cpp.

Here is the caller graph for this function:

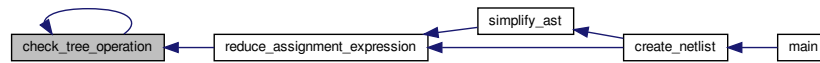


2.9.2.10 check_tree_operation()

```
bool check_tree_operation (
    ast_node_t * node )
```

Definition at line 1388 of file ast_elaborate.cpp.

Here is the caller graph for this function:

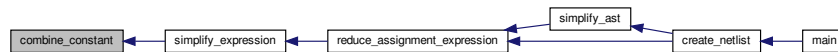


2.9.2.11 combine_constant()

```
bool combine_constant ( )
```

Definition at line 804 of file ast_elaborate.cpp.

Here is the caller graph for this function:

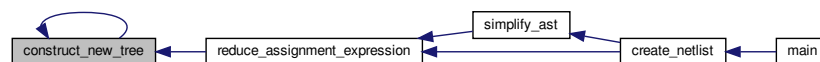


2.9.2.12 construct_new_tree()

```
void construct_new_tree (
    enode * tail,
    ast_node_t * node,
    int line_num,
    int file_num )
```

Definition at line 883 of file ast_elaborate.cpp.

Here is the caller graph for this function:

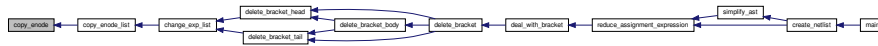


2.9.2.13 copy_enode()

```
void copy_enode (
    enode * node,
    enode * new_node )
```

Definition at line 1338 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.9.2.14 copy_enode_list()

```
enode* copy_enode_list (
    enode * new_head,
    enode * begin,
    enode * end )
```

Definition at line 1319 of file ast_elaborate.cpp.

Here is the caller graph for this function:

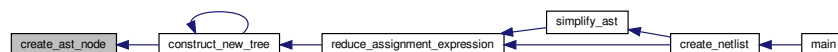


2.9.2.15 create_ast_node()

```
void create_ast_node (
    enode * temp,
    ast_node_t * node,
    int line_num,
    int file_num )
```

Definition at line 1019 of file ast_elaborate.cpp.

Here is the caller graph for this function:

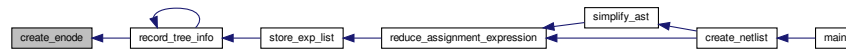


2.9.2.16 create_enode()

```
void create_enode (
    ast_node_t * node )
```

Definition at line 622 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.9.2.17 create_op_node()

```
void create_op_node (
    ast_node_t * node,
    enode * temp,
    int line_num,
    int file_num )
```

Definition at line 1046 of file ast_elaborate.cpp.

Here is the caller graph for this function:

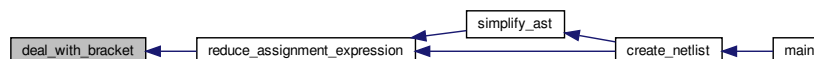


2.9.2.18 deal_with_bracket()

```
bool deal_with_bracket (
    ast_node_t * node )
```

Definition at line 1090 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.9.2.19 delete_bracket()

```
void delete_bracket (
    int begin,
    int end )
```

Definition at line 1170 of file ast_elaborate.cpp.

Here is the caller graph for this function:

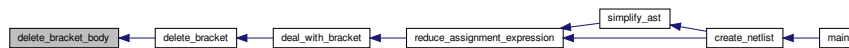


2.9.2.20 delete_bracket_body()

```
void delete_bracket_body (
    enode * begin,
    enode * end )
```

Definition at line 1364 of file ast_elaborate.cpp.

Here is the caller graph for this function:

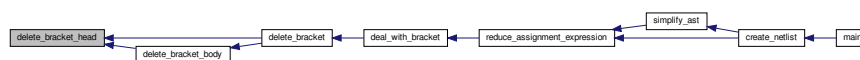


2.9.2.21 delete_bracket_head()

```
void delete_bracket_head (
    enode * begin,
    enode * end )
```

Definition at line 1206 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.9.2.22 delete_bracket_tail()

```
void delete_bracket_tail (
    enode * begin,
    enode * end )
```

Definition at line 1349 of file ast_elaborate.cpp.

Here is the caller graph for this function:

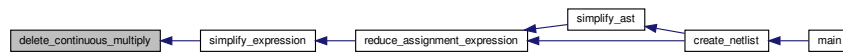


2.9.2.23 delete_continuous_multiply()

```
bool delete_continuous_multiply ( )
```

Definition at line 956 of file ast_elaborate.cpp.

Here is the caller graph for this function:

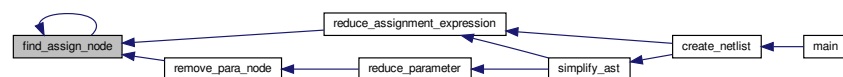


2.9.2.24 find_assign_node()

```
void find_assign_node (
    ast_node_t * t,
    std::vector< ast_node_t *> list )
```

Definition at line 482 of file ast_elaborate.cpp.

Here is the caller graph for this function:

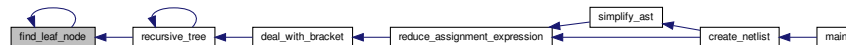


2.9.2.25 find_leaf_node()

```
void find_leaf_node (
    ast_node_t * node,
    std::vector< int > list_bracket,
    int ids2 )
```

Definition at line 1156 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.9.2.26 find_most_unique_count()

```
void find_most_unique_count ( )
```

Definition at line 340 of file ast_elaborate.cpp.

2.9.2.27 find_parameter()

```
void find_parameter (
    ast_node_t * top,
    std::vector< ast_node_t *> para )
```

Definition at line 1444 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.9.2.28 find_tail()

```
enode* find_tail (
    enode * node )
```

Definition at line 513 of file ast_elaborate.cpp.

Here is the caller graph for this function:

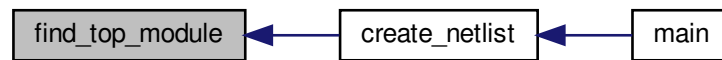


2.9.2.29 find_top_module()

```
ast_node_t* find_top_module ( )
```

Definition at line 419 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:

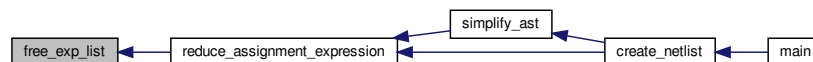


2.9.2.30 free_exp_list()

```
void free_exp_list ( )
```

Definition at line 1077 of file ast_elaborate.cpp.

Here is the caller graph for this function:

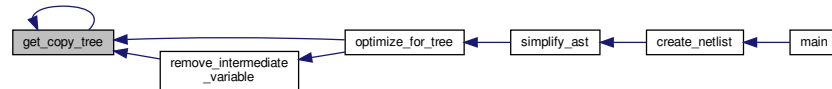


2.9.2.31 get_copy_tree()

```
ast_node_t* get_copy_tree (
    ast_node_t * node,
    long long virtual_value,
    std::string virtual_name )
```

Definition at line 177 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.9.2.32 has_intermediate_variable()

```
short has_intermediate_variable (
    ast_node_t * node )
```

Definition at line 1582 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.9.2.33 keep_all_branch()

```
void keep_all_branch (
    ast_node_t * temp_node,
    ast_node_t * for_parent,
    int mark )
```

Definition at line 1599 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.9.2.34 mark_node_read()

```
void mark_node_read (
    ast_node_t * node,
    std::vector< std::string > list )
```

Definition at line 367 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.9.2.35 mark_node_write()

```
void mark_node_write (
    ast_node_t * node,
    std::vector< std::string > list )
```

Definition at line 351 of file ast_elaborate.cpp.

Here is the caller graph for this function:

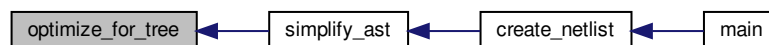


2.9.2.36 optimize_for_tree()

```
void optimize_for_tree ( )
```

Definition at line 80 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.9.2.37 `reallocate_node()`

```
void reallocate_node (
    ast_node_t * node,
    int idx )
```

Definition at line 326 of file `ast_elaborate.cpp`.

Here is the caller graph for this function:



2.9.2.38 `record_expression()`

```
void record_expression (
    ast_node_t * node,
    std::vector< std::string > expressions,
    bool * found_statement )
```

Definition at line 205 of file `ast_elaborate.cpp`.

Here is the caller graph for this function:

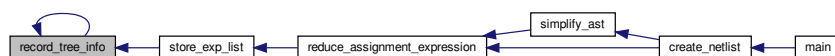


2.9.2.39 `record_tree_info()`

```
void record_tree_info (
    ast_node_t * node )
```

Definition at line 603 of file `ast_elaborate.cpp`.

Here is the caller graph for this function:

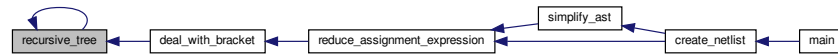


2.9.2.40 recursive_tree()

```
void recursive_tree (
    ast_node_t * node,
    std::vector< int > list_bracket )
```

Definition at line 1133 of file ast_elaborate.cpp.

Here is the caller graph for this function:

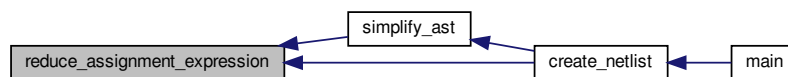


2.9.2.41 reduce_assignment_expression()

```
void reduce_assignment_expression ( )
```

Definition at line 439 of file ast_elaborate.cpp.

Here is the caller graph for this function:

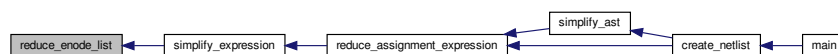


2.9.2.42 reduce_enode_list()

```
void reduce_enode_list ( )
```

Definition at line 527 of file ast_elaborate.cpp.

Here is the caller graph for this function:

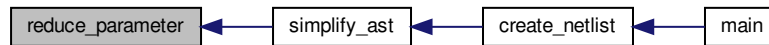


2.9.2.43 reduce_parameter()

```
void reduce_parameter ( )
```

Definition at line 1428 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.9.2.44 remove_intermediate_variable()

```
void remove_intermediate_variable (
    ast_node_t * node,
    std::vector< std::string > list,
    long long virtual_value,
    std::string virtual_name )
```

Definition at line 382 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.9.2.45 remove_para_node()

```
void remove_para_node (
    ast_node_t * top,
    std::vector< ast_node_t *> para )
```

Definition at line 1458 of file ast_elaborate.cpp.

Here is the caller graph for this function:

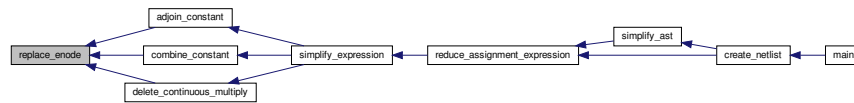


2.9.2.46 replace_enode()

```
enode* replace_enode (
    int data,
    enode * t,
    int mark )
```

Definition at line 760 of file ast_elaborate.cpp.

Here is the caller graph for this function:

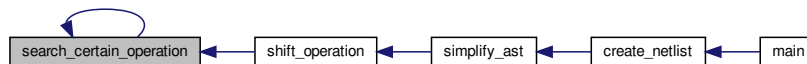


2.9.2.47 search_certain_operation()

```
void search_certain_operation (
    ast_node_t * node )
```

Definition at line 1514 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.9.2.48 search_for_node()

```
void search_for_node (
    ast_node_t * root,
    std::vector< ast_node_t *> list_for_node,
    std::vector< ast_node_t *> list_parent )
```

Definition at line 158 of file ast_elaborate.cpp.

Here is the caller graph for this function:

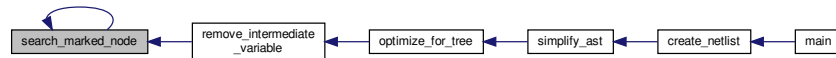


2.9.2.49 search_marked_node()

```
ast_node_t* search_marked_node (
    ast_node_t * node,
    int is,
    std::string temp )
```

Definition at line 419 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.9.2.50 shift_operation()

```
void shift_operation ( )
```

Definition at line 1499 of file ast_elaborate.cpp.

Here is the caller graph for this function:

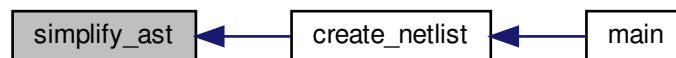


2.9.2.51 simplify_ast()

```
int simplify_ast ( )
```

Definition at line 59 of file ast_elaborate.cpp.

Here is the caller graph for this function:

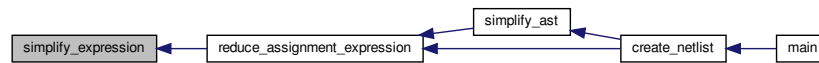


2.9.2.52 simplify_expression()

```
bool simplify_expression ( )
```

Definition at line 495 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.9.2.53 store_exp_list()

```
void store_exp_list (
    ast_node_t * node )
```

Definition at line 586 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.9.2.54 translate_expression()

```
void translate_expression (
    std::vector< std::string > infix_exp,
    std::vector< std::string > postfix_exp )
```

Definition at line 297 of file ast_elaborate.cpp.

Here is the caller graph for this function:



2.10 vtr-verilog-to-routing/ODIN_II/SRC/lib_ast/ast_util.cpp File Reference

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <stdarg.h>
#include <math.h>
#include "globals.h"
#include "types.h"
#include "ast_util.h"
#include "odin_util.h"
#include "vtr_util.h"
#include "vtr_memory.h"
```

Functions

- char ** [get_name_of_pins_number](#) (ast_node_t *var_node, int start, int width)
- void [update_tree_tag](#) (ast_node_t *node, int cases, int tagged)
- void [add_tag_data](#) ()
- ast_node_t * [create_node_w_type](#) (ids id, int line_number, int file_number)
- void [free_assignment_of_node_keep_tree](#) (ast_node_t *node)
- ast_node_t * [free_single_node](#) (ast_node_t *node)
- ast_node_t * [free_whole_tree](#) (ast_node_t *node)
- ast_node_t * [create_tree_node_id](#) (char *string, int line_number, int)
- ast_node_t * [create_tree_node_long_long_number](#) (long long number, int constant_bit_size, int line_number, int)
- ast_node_t * [create_tree_node_number](#) (char *number, int line_number, int)
- void [allocate_children_to_node](#) (ast_node_t *node, int num_children,...)
- void [add_child_to_node](#) (ast_node_t *node, ast_node_t *child)
- void [add_child_at_the_beginning_of_the_node](#) (ast_node_t *node, ast_node_t *child)
- int [get_range](#) (ast_node_t *first_node)
- void [make_concat_into_list_of_strings](#) (ast_node_t *concat_top, char *instance_name_prefix)
- void [change_to_number_node](#) (ast_node_t *node, long long value)
- char * [get_name_of_var_declare_at_bit](#) (ast_node_t *var_declare, int bit)
- char * [get_name_of_pin_at_bit](#) (ast_node_t *var_node, int bit, char *instance_name_prefix)
- char_list_t * [get_name_of_pins](#) (ast_node_t *var_node, char *instance_name_prefix)
- char_list_t * [get_name_of_pins_with_prefix](#) (ast_node_t *var_node, char *instance_name_prefix)
- ast_node_t * [resolve_node](#) (STRING_CACHE *local_param_table_sc, short initial, char *module_name, ast_node_t *node)
- char * [make_module_param_name](#) (ast_node_t *module_param_list, char *module_name)
- void [move_ast_node](#) (ast_node_t *src, ast_node_t *dest, ast_node_t *node)
- ast_node_t * [ast_node_deep_copy](#) (ast_node_t *node)
- ast_node_t * [fold_unary](#) (ast_node_t *child_0, operation_list op_id)
- ast_node_t * [fold_binary](#) (ast_node_t *child_0, ast_node_t *child_1, operation_list op_id)
- ast_node_t * [node_is_constant](#) (ast_node_t *node)
- void [initial_node](#) (ast_node_t *new_node, ids id, int line_number, int file_number, int unique_counter)

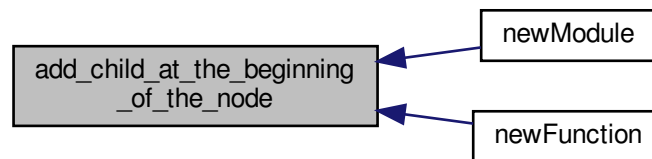
2.10.1 Function Documentation

2.10.1.1 add_child_at_the_beginning_of_the_node()

```
void add_child_at_the_beginning_of_the_node (  
    ast_node_t * node,  
    ast_node_t * child )
```

Definition at line 387 of file ast_util.cpp.

Here is the caller graph for this function:

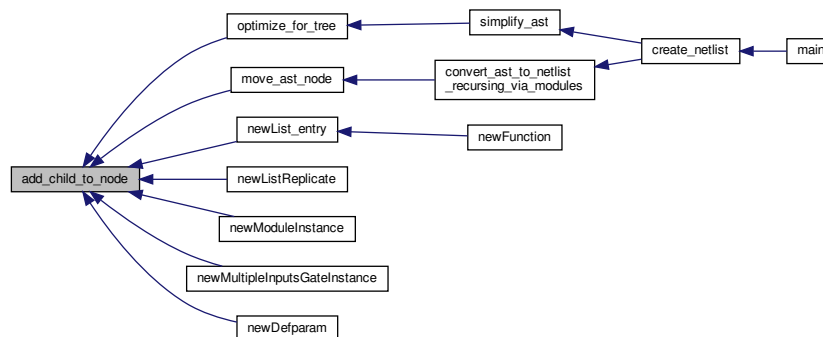


2.10.1.2 add_child_to_node()

```
void add_child_to_node (  
    ast_node_t * node,  
    ast_node_t * child )
```

Definition at line 372 of file ast_util.cpp.

Here is the caller graph for this function:



2.10.1.3 add_tag_data()

```
void add_tag_data ( )
```

Definition at line 70 of file ast_util.cpp.

Here is the caller graph for this function:

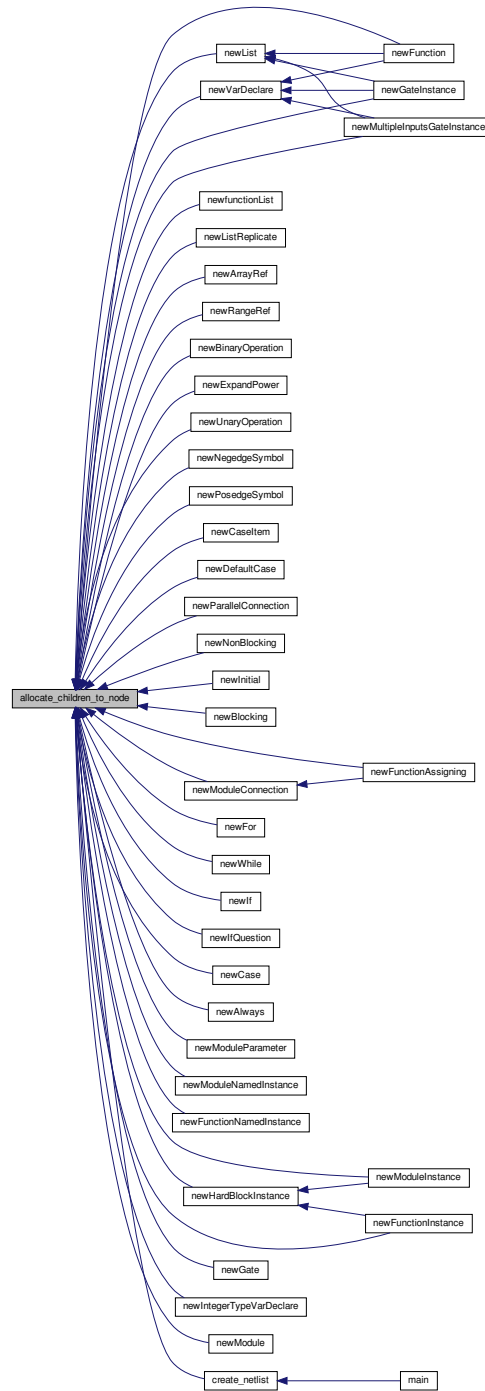


2.10.1.4 allocate_children_to_node()

```
void allocate_children_to_node (
    ast_node_t * node,
    int num_children,
    ... )
```

Definition at line 351 of file ast_util.cpp.

Here is the caller graph for this function:



2.10.1.5 ast_node_deep_copy()

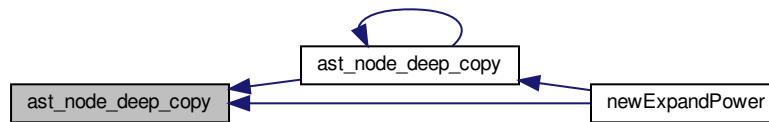
```
ast_node_t* ast_node_deep_copy (
```



```
ast_node_t * node )
```

Definition at line 1040 of file ast_util.cpp.

Here is the caller graph for this function:

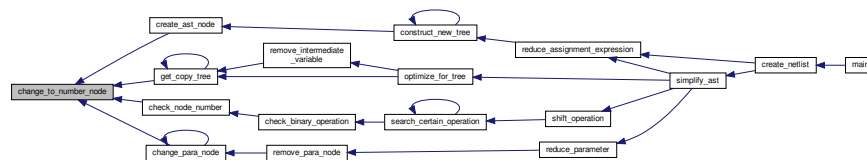


2.10.1.6 change_to_number_node()

```
void change_to_number_node (
    ast_node_t * node,
    long long value )
```

Definition at line 563 of file ast_util.cpp.

Here is the caller graph for this function:

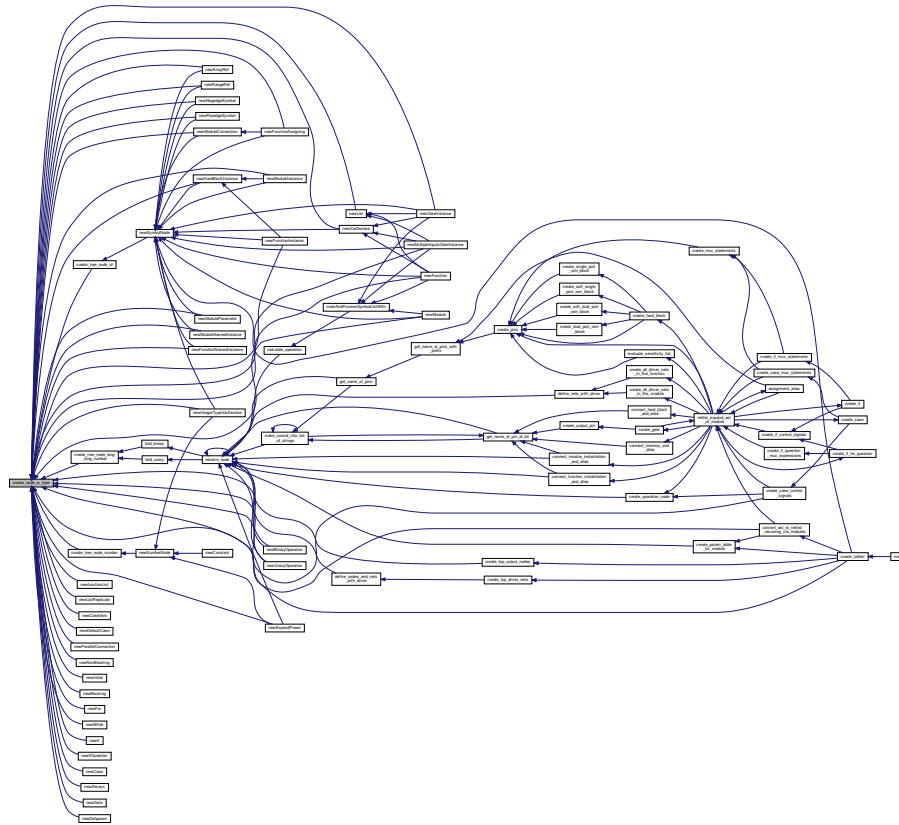


2.10.1.7 create_node_w_type()

```
ast_node_t* create_node_w_type (
    ids id,
    int line_number,
    int file_number )
```

Definition at line 95 of file ast_util.cpp.

Here is the caller graph for this function:

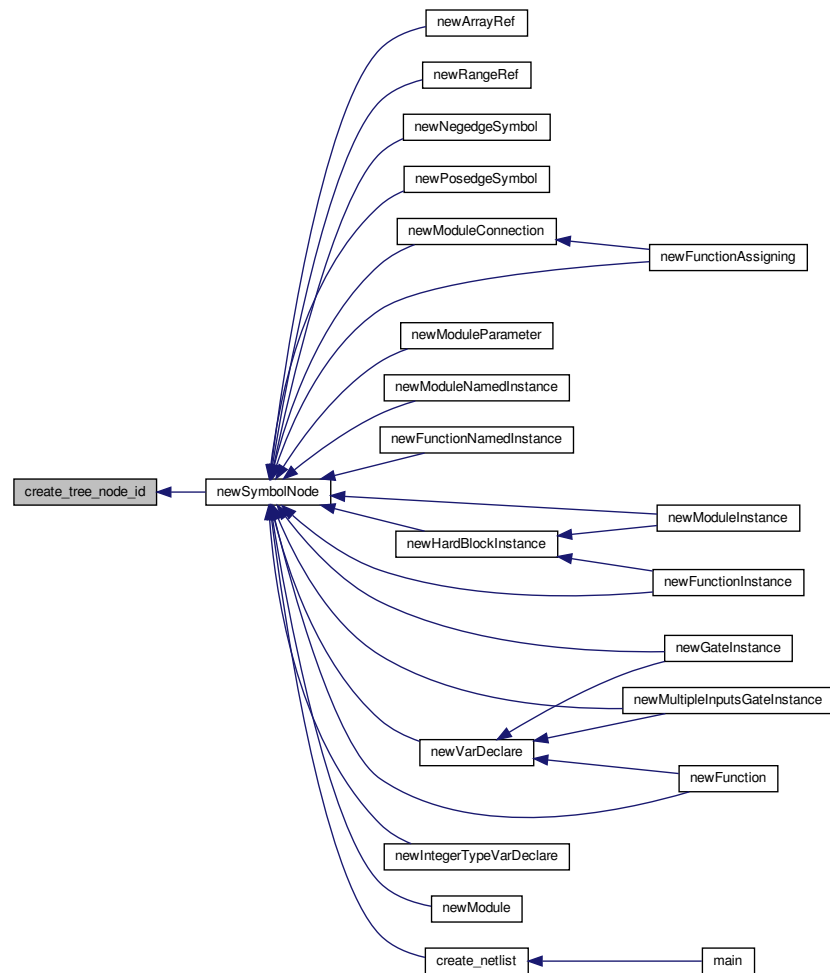


2.10.1.8 create_tree_node_id()

```
ast_node_t* create_tree_node_id (
    char * string,
    int line_number,
    int )
```

Definition at line 175 of file ast_util.cpp.

Here is the caller graph for this function:



2.10.1.9 create_tree_node_long_long_number()

```

ast_node_t* create_tree_node_long_long_number (
    long long number,
    int constant_bit_size,
    int line_number,
    int )

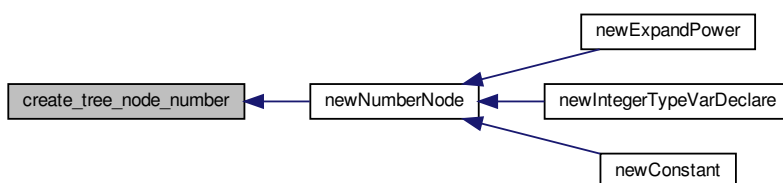
```

Definition at line 186 of file `ast_util.cpp`.

The diagram illustrates a complex dependency graph for a data pipeline. The graph is composed of numerous nodes, each representing a data source or a processing step, connected by directed edges that show the flow of data and dependencies. The nodes are organized into several layers, with the input data sources at the top and the final output nodes at the bottom. The graph is highly interconnected, with many nodes having multiple incoming and outgoing edges, indicating a complex dependency structure. The nodes are labeled with various identifiers, including 'input_data', 'input_data_2', 'input_data_3', 'get_data', 'get_data_2', 'get_data_3', 'output_data', 'output_data_2', 'output_data_3', and many others. The edges represent the flow of data and dependencies between these nodes, showing how the final output is derived from the input data through a series of processing steps.

```
ast_node_t* create_tree_node_number (
    char * number,
    int line_number,
    int )
```

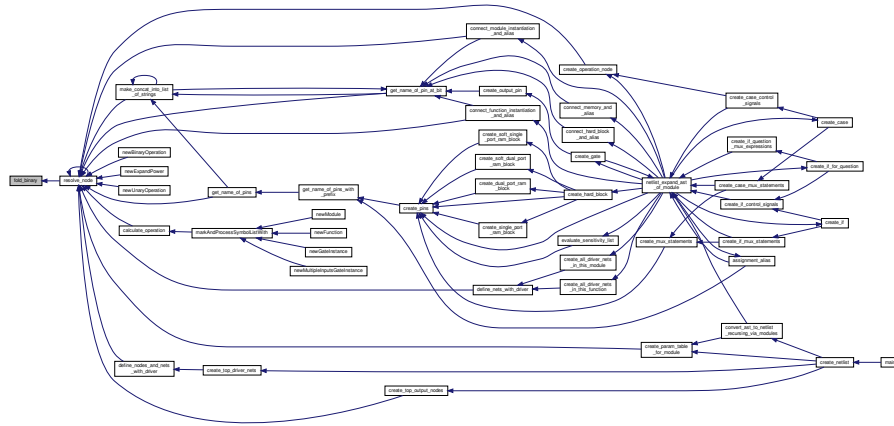
Here is the caller graph for this function:



```
ast_node_t* fold_binary (
    ast_node_t * child_0,
    ast_node_t * child_1,
    operation_list op_id )
```

Definition at line 1158 of file ast_util.cpp.

Here is the caller graph for this function:

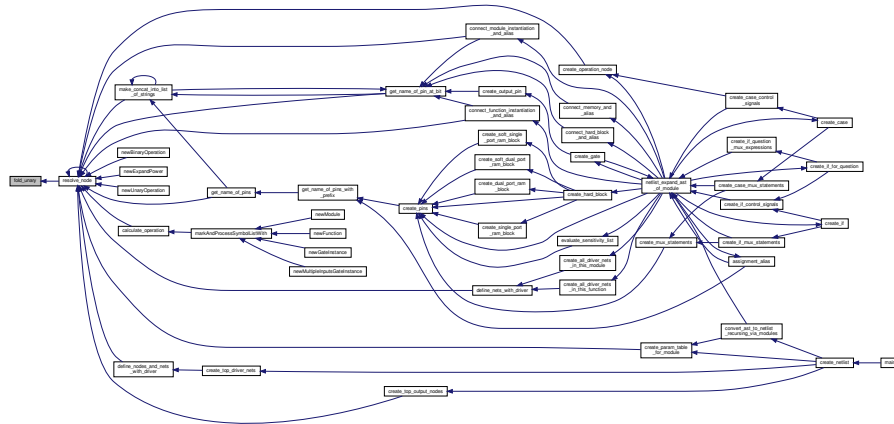


2.10.1.12 fold_unary()

```
ast_node_t* fold_unary (
    ast_node_t * child_0,
    operation_list op_id )
```

Definition at line 1071 of file ast_util.cpp.

Here is the caller graph for this function:

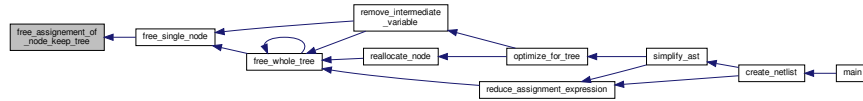


2.10.1.13 free_assignment_of_node_keep_tree()

```
void free_assignment_of_node_keep_tree (
    ast_node_t * node )
```

Definition at line 122 of file ast_util.cpp.

Here is the caller graph for this function:

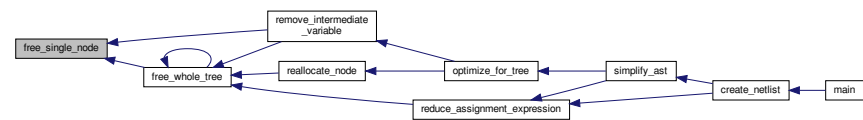


2.10.1.14 free_single_node()

```
ast_node_t* free_single_node (
    ast_node_t * node )
```

Definition at line 147 of file ast_util.cpp.

Here is the caller graph for this function:

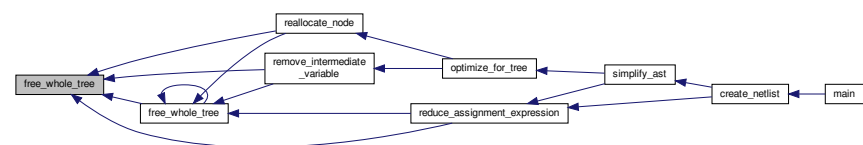


2.10.1.15 free_whole_tree()

```
ast_node_t* free_whole_tree (
    ast_node_t * node )
```

Definition at line 161 of file ast_util.cpp.

Here is the caller graph for this function:

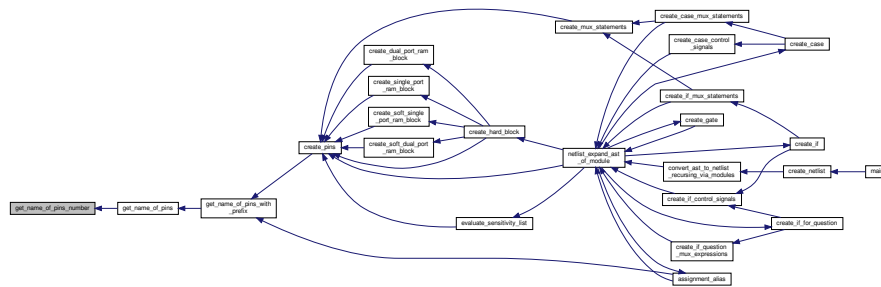


2.10.1.18 get_name_of_pins_number()

```
char ** get_name_of_pins_number (
    ast_node_t * var_node,
    int start,
    int width )
```

Definition at line 721 of file ast_util.cpp.

Here is the caller graph for this function:

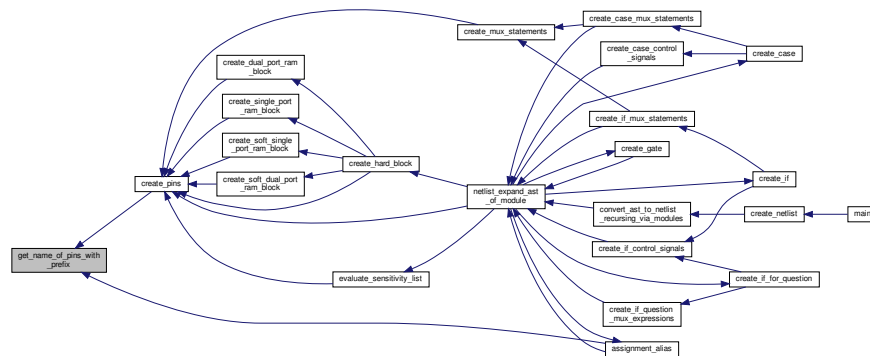


2.10.1.19 get_name_of_pins_with_prefix()

```
char_list_t* get_name_of_pins_with_prefix (
    ast_node_t * var_node,
    char * instance_name_prefix )
```

Definition at line 897 of file ast_util.cpp.

Here is the caller graph for this function:

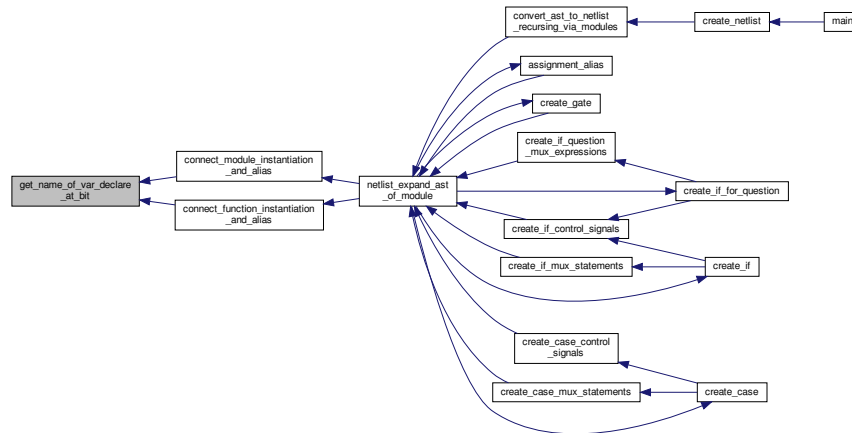


2.10.1.20 get_name_of_var_declare_at_bit()

```
char* get_name_of_var_declare_at_bit (
    ast_node_t * var_declare,
    int bit )
```

Definition at line 589 of file ast_util.cpp.

Here is the caller graph for this function:

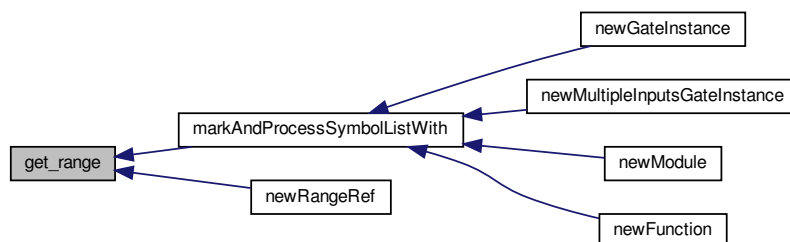


2.10.1.21 get_range()

```
int get_range (
    ast_node_t * first_node )
```

Definition at line 417 of file ast_util.cpp.

Here is the caller graph for this function:



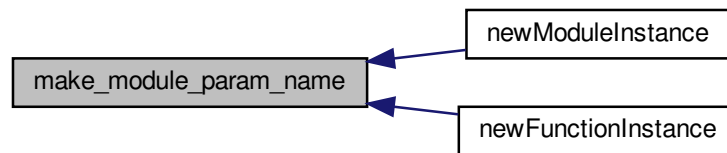
2.10.1.24 make_module_param_name()

```
char* make_module_param_name (
    ast_node_t * module_param_list,
    char * module_name )
```

Make a unique name for a module based on its parameter list e.g. for a "mod #(0,1,2,3) a(b,c,d)" instantiation you get name__0_1_2_3

Definition at line 988 of file ast_util.cpp.

Here is the caller graph for this function:

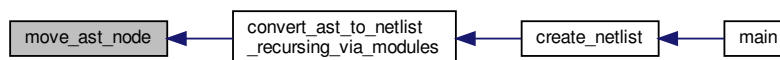


2.10.1.25 move_ast_node()

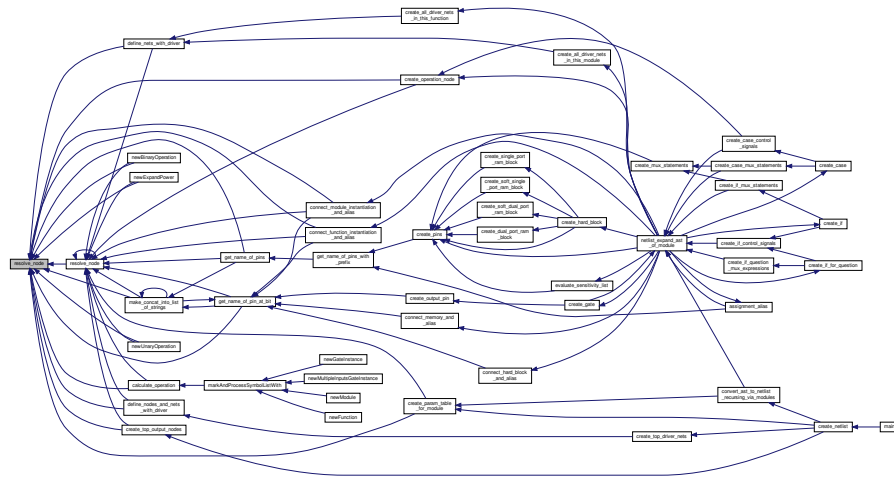
```
void move_ast_node (
    ast_node_t * src,
    ast_node_t * dest,
    ast_node_t * node )
```

Definition at line 1016 of file ast_util.cpp.

Here is the caller graph for this function:



Here is the caller graph for this function:



2.10.1.28 update_tree_tag()

```
void update_tree_tag (
    ast_node_t * node,
    int cases,
    int tagged )
```

Definition at line 48 of file ast_util.cpp.

2.11 vtr-verilog-to-routing/ODIN_II/SRC/lib_ast/ast_util.h File Reference

```
#include "types.h"
```

Functions

- void [add_tag_data](#) ()
- ast_node_t * [create_node_w_type](#) (ids id, int line_number, int file_number)
- ast_node_t * [create_tree_node_id](#) (char *string, int line_number, int file_number)
- ast_node_t * [create_tree_node_long_long_number](#) (long long number, int constant_bit_size, int line_number, int file_number)
- ast_node_t * [create_tree_node_number](#) (char *number, int line_number, int file_number)
- void [initial_node](#) (ast_node_t *node, ids id, int line_number, int file_number, int counter)
- void [allocate_children_to_node](#) (ast_node_t *node, int num_children,...)
- void [add_child_to_node](#) (ast_node_t *node, ast_node_t *child)
- void [add_child_at_the_beginning_of_the_node](#) (ast_node_t *node, ast_node_t *child)
- void [move_ast_node](#) (ast_node_t *src, ast_node_t *dest, ast_node_t *node)

- `ast_node_t * ast_node_deep_copy (ast_node_t *node)`
- `ast_node_t * free_whole_tree (ast_node_t *node)`
- `ast_node_t * free_single_node (ast_node_t *node)`
- `void free_assignment_of_node_keep_tree (ast_node_t *node)`
- `char * make_module_param_name (ast_node_t *module_param_list, char *module_name)`
- `void make_concat_into_list_of_strings (ast_node_t *concat_top, char *instance_name_prefix)`
- `void change_to_number_node (ast_node_t *node, long long number)`
- `int get_range (ast_node_t *first_node)`
- `char * get_name_of_pin_at_bit (ast_node_t *var_node, int bit, char *instance_name_prefix)`
- `char * get_name_of_var_declare_at_bit (ast_node_t *var_declare, int bit)`
- `char_list_t * get_name_of_pins (ast_node_t *var_node, char *instance_name_prefix)`
- `char_list_t * get_name_of_pins_with_prefix (ast_node_t *var_node, char *instance_name_prefix)`
- `ast_node_t * resolve_node (STRING_CACHE *local_param_table_sc, short initial, char *module_name, ast_node_t *node)`
- `ast_node_t * node_is_constant (ast_node_t *node)`
- `ast_node_t * fold_binary (ast_node_t *child_0, ast_node_t *child_1, operation_list op_id)`
- `ast_node_t * fold_unary (ast_node_t *child_0, operation_list op_id)`

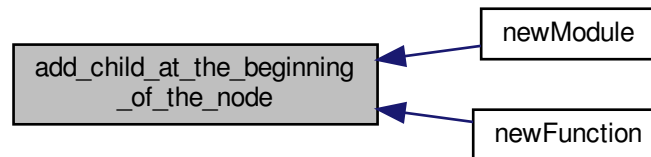
2.11.1 Function Documentation

2.11.1.1 add_child_at_the_beginning_of_the_node()

```
void add_child_at_the_beginning_of_the_node (
    ast_node_t * node,
    ast_node_t * child )
```

Definition at line 387 of file `ast_util.cpp`.

Here is the caller graph for this function:

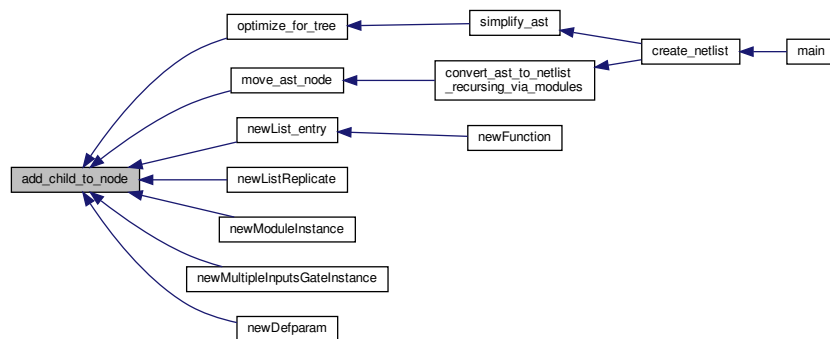


2.11.1.2 add_child_to_node()

```
void add_child_to_node (
    ast_node_t * node,
    ast_node_t * child )
```

Definition at line 372 of file ast_util.cpp.

Here is the caller graph for this function:



2.11.1.3 add_tag_data()

```
void add_tag_data ( )
```

Definition at line 70 of file ast_util.cpp.

Here is the caller graph for this function:

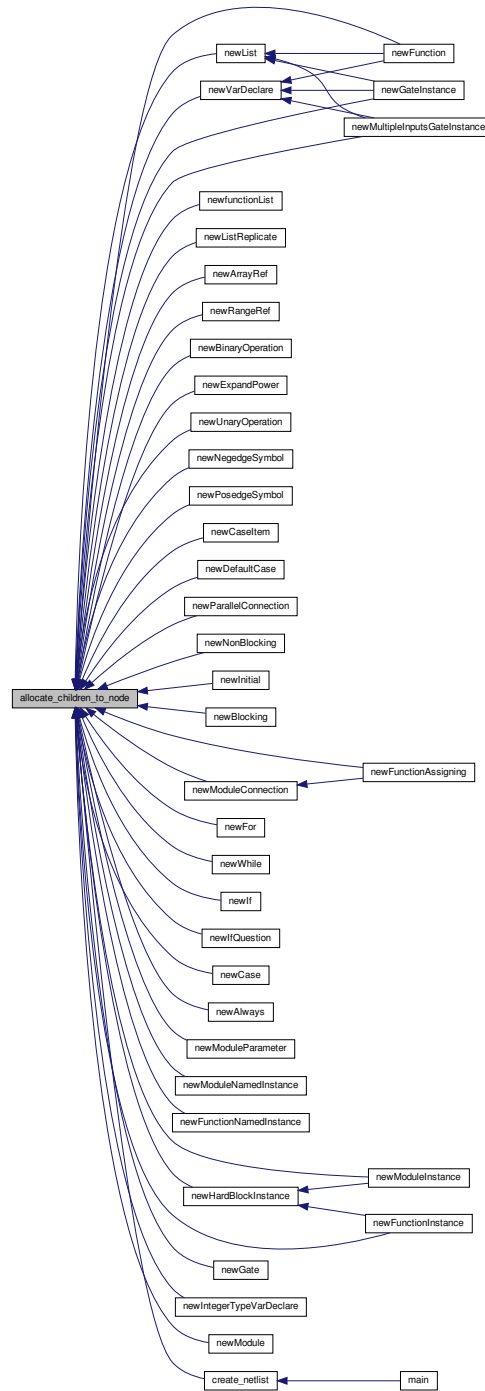


2.11.1.4 allocate_children_to_node()

```
void allocate_children_to_node (  
    ast_node_t * node,  
    int num_children,  
    ... )
```

Definition at line 351 of file ast_util.cpp.

Here is the caller graph for this function:



2.11.1.5 ast_node_deep_copy()

```
ast_node_t* ast_node_deep_copy (
```

Definition at line 1040 of file ast_util.cpp.

```
graph LR; newExpandPower --> ast_node_deep_copy; ast_node_deep_copy --> ast_node_deep_copy;
```

```
void change_to_number_node (
    ast_node_t * node,
    long long number )
```

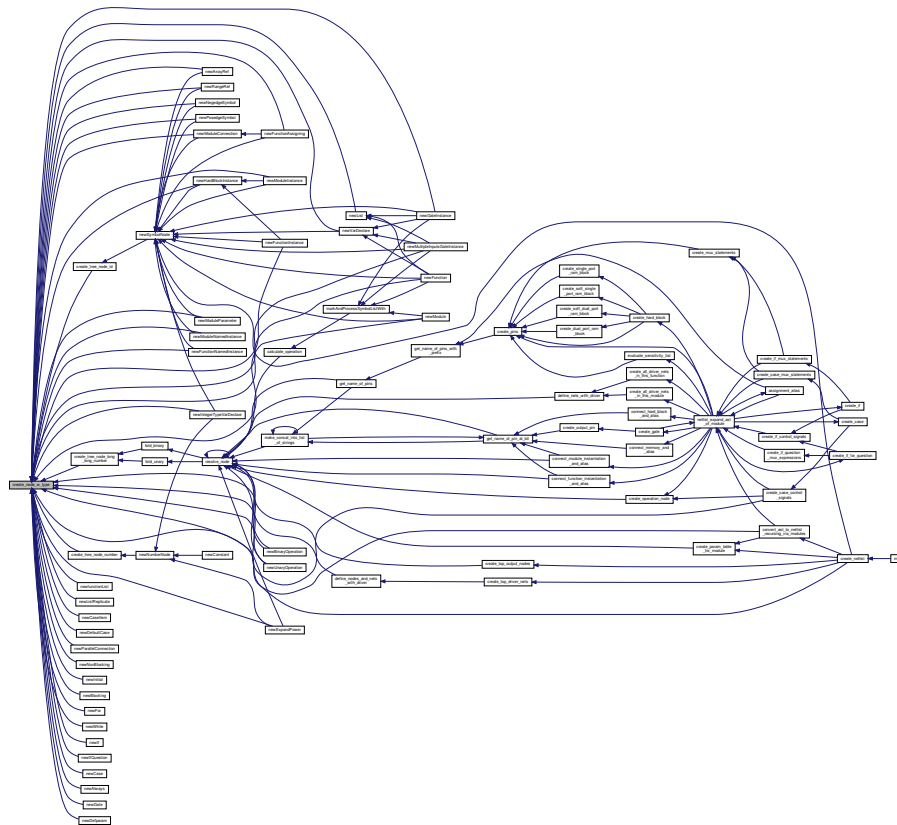
```

graph LR
    main --> create_ast[create_ast]
    create_ast --> construct_new_tree[construct_new_tree]
    construct_new_tree --> reduce_assignment_expression[reduce_assignment_expression]
    reduce_assignment_expression --> simplify_ast[simplify_ast]
    simplify_ast --> create_reflist[create_reflist]
    simplify_ast --> shift_operation[shift_operation]
    simplify_ast --> reduce_parameter[reduce_parameter]
    shift_operation --> search_certain_operation[search_certain_operation]
    search_certain_operation --> check_binary_operation[check_binary_operation]
    check_binary_operation --> check_node_number[check_node_number]
    check_node_number --> change_to_number_node[change_to_number_node]
    check_binary_operation --> remove_param_node[remove_param_node]
    remove_param_node --> change_param_node[change_param_node]
    search_certain_operation --> optimize_for_tree[optimize_for_tree]
    optimize_for_tree --> get_copy_tree[get_copy_tree]
    get_copy_tree --> create_ast_node[create_ast_node]
    create_ast_node --> change_to_number_node
    create_ast_node --> remove_intermediate_variable[remove_intermediate_variable]
    remove_intermediate_variable --> optimize_for_tree
    optimize_for_tree --> create_ast_node
    change_to_number_node --> change_to_number_node
    change_param_node --> change_to_number_node
    
```

```
ast_node_t* create_node_w_type (
    ids id,
    int line_number,
    int file_number )
```

Generated on Thu Aug 10 2017 19:13:39 for Odin by Doxygen

Here is the caller graph for this function:

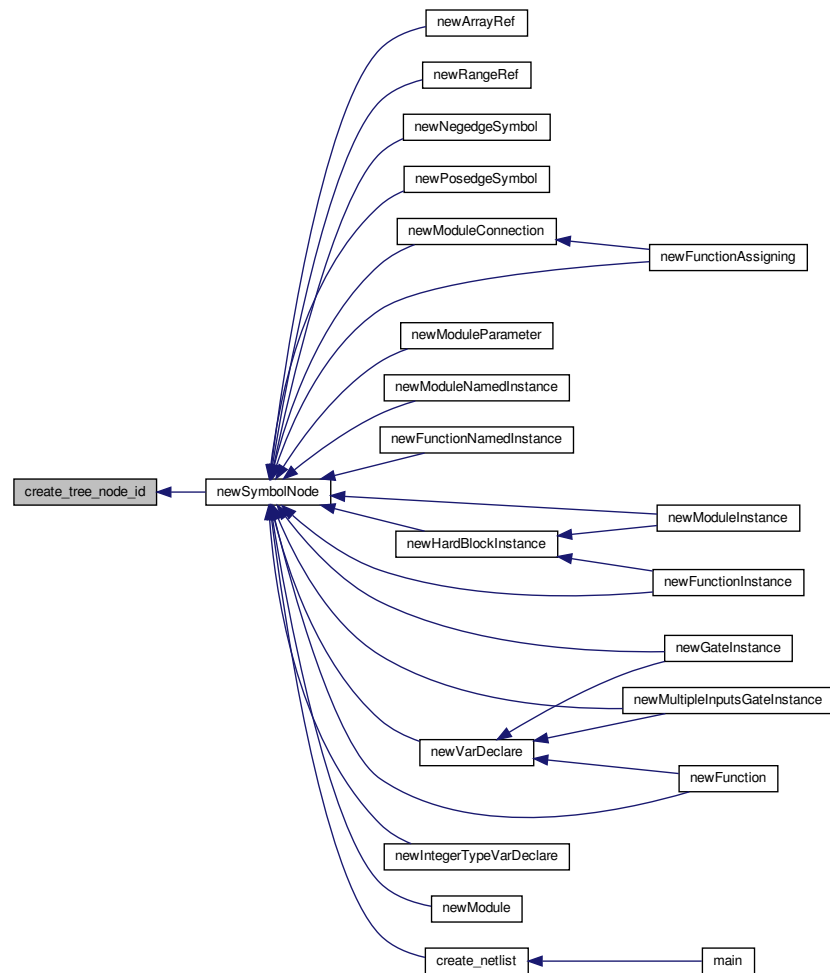


2.11.1.8 `create_tree_node_id()`

```
ast_node_t* create_tree_node_id (
    char * string,
    int line_number,
    int file_number )
```

Definition at line 175 of file `ast_util.cpp`.

Here is the caller graph for this function:



2.11.1.9 create_tree_node_long_long_number()

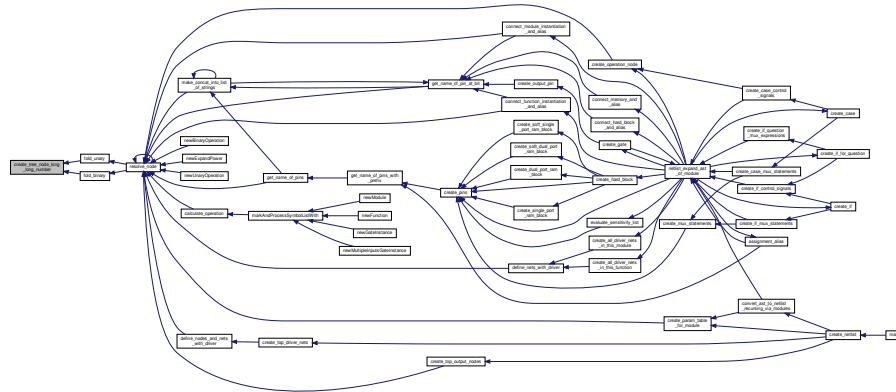
```

ast_node_t* create_tree_node_long_long_number (
    long long number,
    int constant_bit_size,
    int line_number,
    int file_number )

```

Definition at line 186 of file ast_util.cpp.

Here is the caller graph for this function:

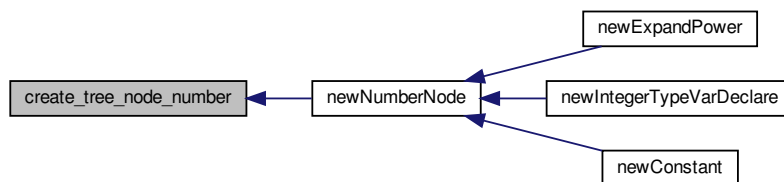


2.11.1.10 create_tree_node_number()

```
ast_node_t* create_tree_node_number (
    char * number,
    int line_number,
    int file_number )
```

Definition at line 212 of file ast_util.cpp.

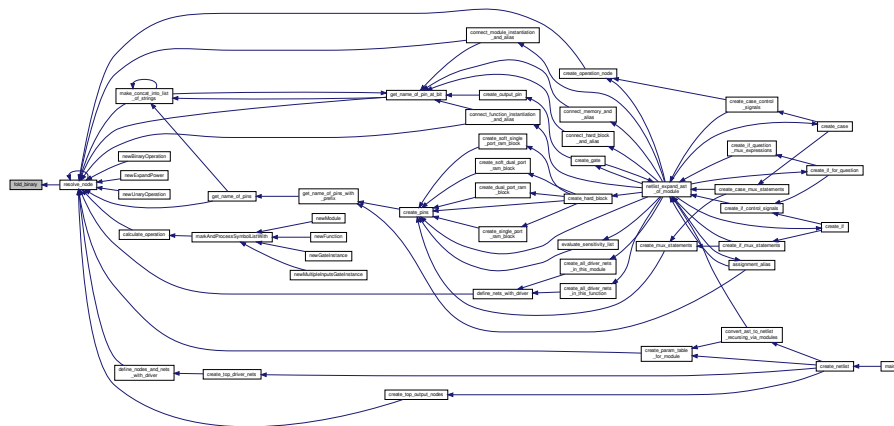
Here is the caller graph for this function:



2.11.1.11 fold_binary()

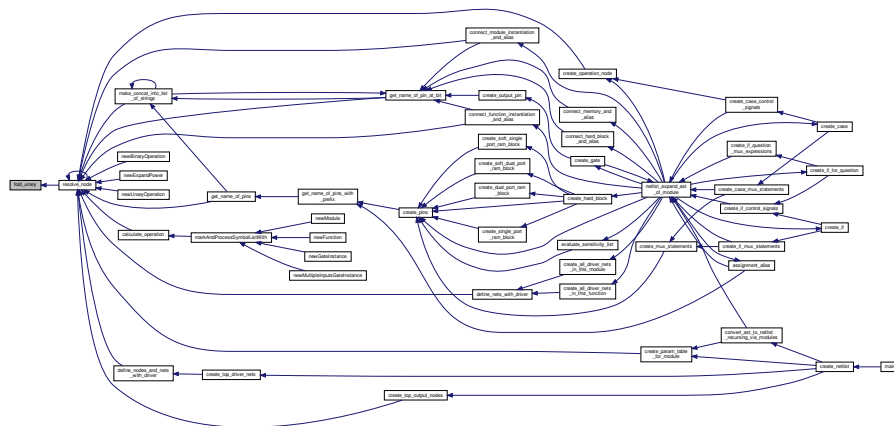
```
ast_node_t* fold_binary (
    ast_node_t * child_0,
    ast_node_t * child_1,
    operation_list op_id )
```

Here is the caller graph for this function:



```
ast_node_t* fold_unary (
    ast_node_t * child_0,
    operation_list op_id )
```

Here is the caller graph for this function:

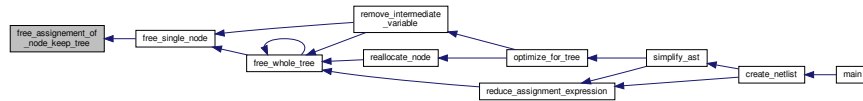


2.11.1.13 free_assignment_of_node_keep_tree()

```
void free_assignment_of_node_keep_tree (
    ast_node_t * node )
```

Definition at line 122 of file ast_util.cpp.

Here is the caller graph for this function:

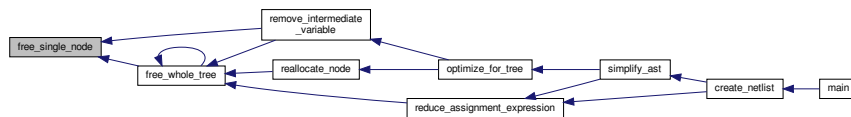


2.11.1.14 free_single_node()

```
ast_node_t* free_single_node (
    ast_node_t * node )
```

Definition at line 147 of file ast_util.cpp.

Here is the caller graph for this function:

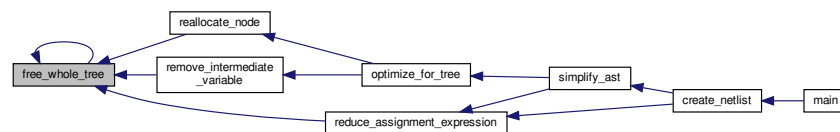


2.11.1.15 free_whole_tree()

```
ast_node_t* free_whole_tree (
    ast_node_t * node )
```

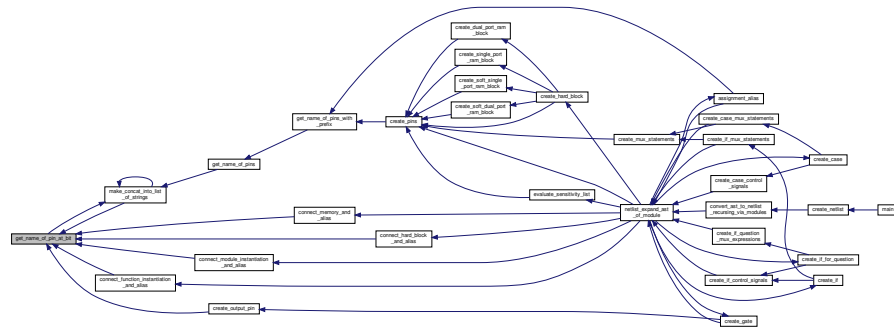
Definition at line 161 of file ast_util.cpp.

Here is the caller graph for this function:



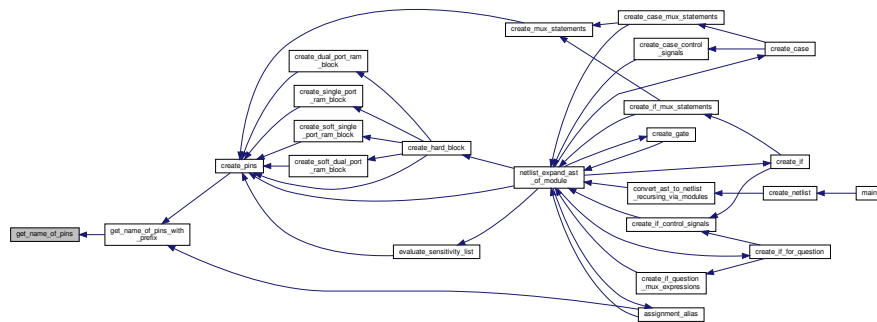
```
char* get_name_of_pin_at_bit (
    ast_node_t * var_node,
    int bit,
    char * instance_name_prefix )
```

Here is the caller graph for this function:



```
char_list_t* get_name_of_pins (
    ast_node_t * var_node,
    char * instance_name_prefix )
```

Here is the caller graph for this function:

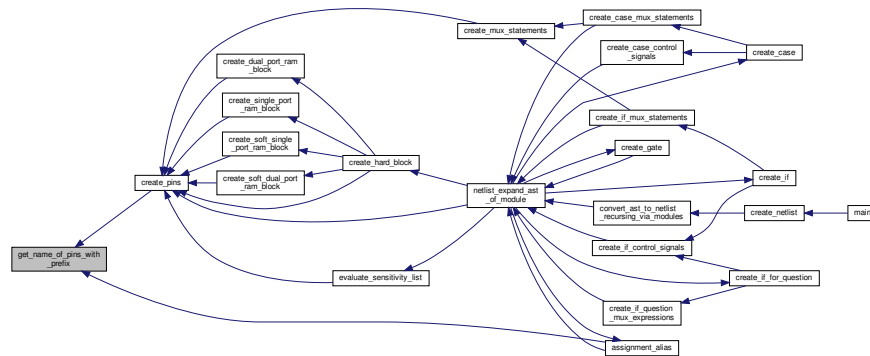


2.11.1.18 get_name_of_pins_with_prefix()

```
char_list_t* get_name_of_pins_with_prefix (
    ast_node_t * var_node,
    char * instance_name_prefix )
```

Definition at line 897 of file ast_util.cpp.

Here is the caller graph for this function:

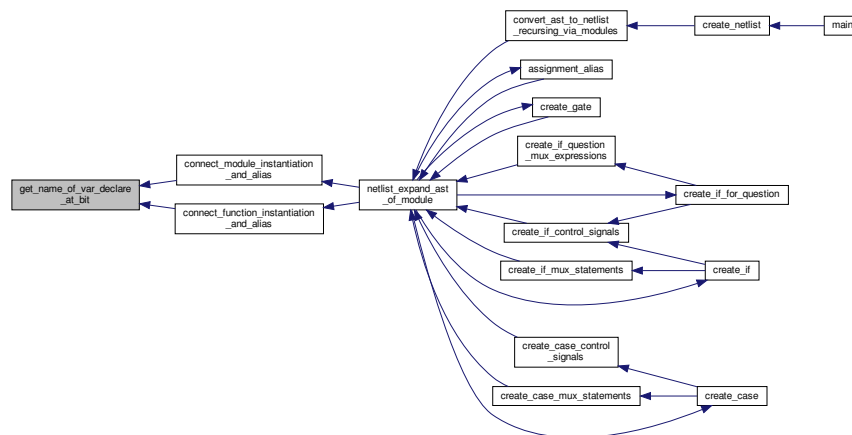


2.11.1.19 get_name_of_var_declare_at_bit()

```
char* get_name_of_var_declare_at_bit (
    ast_node_t * var_declare,
    int bit )
```

Definition at line 589 of file ast_util.cpp.

Here is the caller graph for this function:

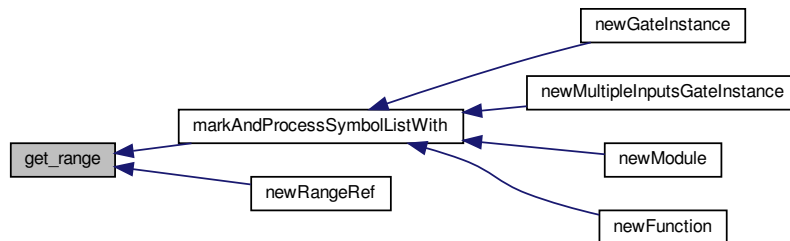


2.11.1.20 get_range()

```
int get_range (
    ast_node_t * first_node )
```

Definition at line 417 of file ast_util.cpp.

Here is the caller graph for this function:



2.11.1.21 initial_node()

```
void initial_node (
    ast_node_t * node,
    ids id,
    int line_number,
    int file_number,
    int counter )
```

Definition at line 1358 of file ast_util.cpp.

Here is the caller graph for this function:



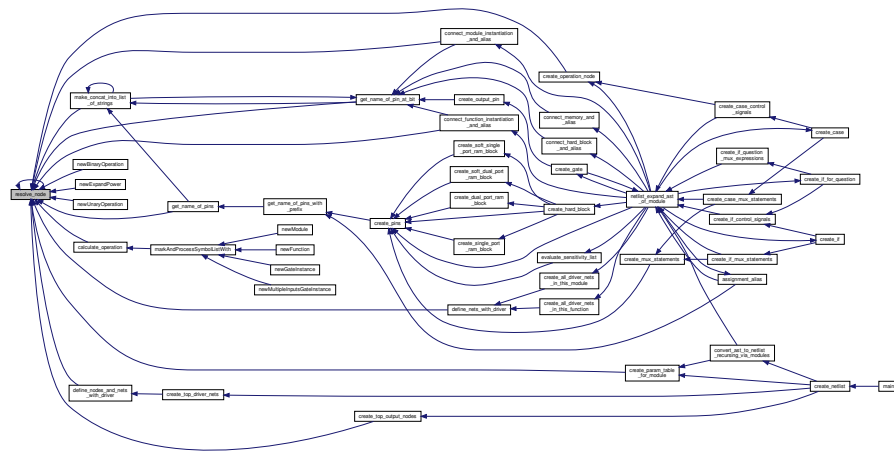
2.11.1.26 resolve_node()

```
ast_node_t* resolve_node (
    STRING_CACHE * local_param_table_sc,
    short initial,
    char * module_name,
    ast_node_t * node )
```

Recursively resolves an IDENTIFIER to a parameter into its actual value, by looking it up in the global_param_table_sc
Also try and fold any BINARY_OPERATIONS now that an IDENTIFIER has been resolved

Definition at line 923 of file ast_util.cpp.

Here is the caller graph for this function:



2.12 vtr-verilog-to-routing/ODIN_II/SRC/lib_ast/parse_making_ast.cpp File Reference

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <assert.h>
#include <math.h>
#include "globals.h"
#include "types.h"
#include "ast_util.h"
#include "parse_making_ast.h"
#include "string_cache.h"
#include "verilog_bison_user_defined.h"
#include "verilog_preprocessor.h"
#include "hard_blocks.h"
#include "vtr_util.h"
#include "vtr_memory.h"
```

Functions

- void [graphVizOutputPreproc](#) (FILE *yyin, std::string path, char *file)
- ast_node_t * [newFunctionAssigning](#) (ast_node_t *expression1, ast_node_t *expression2, int line_number)
- ast_node_t * [newHardBlockInstance](#) (char *module_ref_name, ast_node_t *module_named_instance, int line_number)
- void [parse_to_ast](#) ()
- void [cleanup_hard_blocks](#) ()
- void [init_parser](#) ()
- void [cleanup_parser](#) ()
- void [init_parser_for_file](#) ()
- void [clean_up_parser_for_file](#) ()
- void [next_parsed_verilog_file](#) (ast_node_t *file_items_list)
- ast_node_t * [newSymbolNode](#) (char *id, int line_number)
- ast_node_t * [newNumberNode](#) (char *num, int line_number)
- ast_node_t * [newList](#) (ids node_type, ast_node_t *child)
- ast_node_t * [newfunctionList](#) (ids node_type, ast_node_t *child)
- ast_node_t * [newList_entry](#) (ast_node_t *list, ast_node_t *child)
- ast_node_t * [newListReplicate](#) (ast_node_t *exp, ast_node_t *child)
- ast_node_t * [markAndProcessSymbolListWith](#) (ids top_type, ids id, ast_node_t *symbol_list)
- ast_node_t * [newArrayRef](#) (char *id, ast_node_t *expression, int line_number)
- ast_node_t * [newRangeRef](#) (char *id, ast_node_t *expression1, ast_node_t *expression2, int line_number)
- ast_node_t * [newBinaryOperation](#) (operation_list op_id, ast_node_t *expression1, ast_node_t *expression2, int line_number)
- ast_node_t * [newExpandPower](#) (operation_list op_id, ast_node_t *expression1, ast_node_t *expression2, int line_number)
- ast_node_t * [newUnaryOperation](#) (operation_list op_id, ast_node_t *expression, int line_number)
- ast_node_t * [newNegedgeSymbol](#) (char *symbol, int line_number)
- ast_node_t * [newPosedgeSymbol](#) (char *symbol, int line_number)
- ast_node_t * [newCaseItem](#) (ast_node_t *expression, ast_node_t *statement, int line_number)
- ast_node_t * [newDefaultCase](#) (ast_node_t *statement, int line_number)
- ast_node_t * [newParallelConnection](#) (ast_node_t *expression1, ast_node_t *expression2, int line_number)
- ast_node_t * [newNonBlocking](#) (ast_node_t *expression1, ast_node_t *expression2, int line_number)
- ast_node_t * [newInitial](#) (ast_node_t *expression1, int line_number)
- ast_node_t * [newBlocking](#) (ast_node_t *expression1, ast_node_t *expression2, int line_number)
- ast_node_t * [newFor](#) (ast_node_t *initial, ast_node_t *compare_expression, ast_node_t *terminal, ast_node_t *statement, int line_number)
- ast_node_t * [newWhile](#) (ast_node_t *compare_expression, ast_node_t *statement, int line_number)
- ast_node_t * [newIf](#) (ast_node_t *compare_expression, ast_node_t *true_expression, ast_node_t *false_expression, int line_number)
- ast_node_t * [newIfQuestion](#) (ast_node_t *compare_expression, ast_node_t *true_expression, ast_node_t *false_expression, int line_number)
- ast_node_t * [newCase](#) (ast_node_t *compare_expression, ast_node_t *case_list, int line_number)
- ast_node_t * [newAlways](#) (ast_node_t *delay_control, ast_node_t *statement, int line_number)
- ast_node_t * [newModuleConnection](#) (char *id, ast_node_t *expression, int line_number)
- ast_node_t * [newModuleParameter](#) (char *id, ast_node_t *expression, int line_number)
- ast_node_t * [newModuleNamedInstance](#) (char *unique_name, ast_node_t *module_connect_list, ast_node_t *module_parameter_list, int line_number)
- ast_node_t * [newFunctionNamedInstance](#) (ast_node_t *module_connect_list, ast_node_t *module_parameter_list, int line_number)

- `ast_node_t * newModuleInstance` (char *module_ref_name, ast_node_t *module_named_instance, int line_number)
- `ast_node_t * newFunctionInstance` (char *function_ref_name, ast_node_t *function_named_instance, int line_number)
- `ast_node_t * newGateInstance` (char *gate_instance_name, ast_node_t *expression1, ast_node_t *expression2, ast_node_t *expression3, int line_number)
- `ast_node_t * newMultipleInputsGateInstance` (char *gate_instance_name, ast_node_t *expression1, ast_node_t *expression2, ast_node_t *expression3, int line_number)
- `ast_node_t * newGate` (operation_list op_id, ast_node_t *gate_instance, int line_number)
- `ast_node_t * newVarDeclare` (char *symbol, ast_node_t *expression1, ast_node_t *expression2, ast_node_t *expression3, ast_node_t *expression4, ast_node_t *value, int line_number)
- `ast_node_t * newIntegerTypeVarDeclare` (char *symbol, ast_node_t *, ast_node_t *, ast_node_t *expression3, ast_node_t *expression4, ast_node_t *value, int line_number)
- `ast_node_t * newModule` (char *module_name, ast_node_t *list_of_ports, ast_node_t *list_of_module_items, int line_number)
- `ast_node_t * newFunction` (ast_node_t *list_of_ports, ast_node_t *list_of_module_items, int line_number)
- void `next_function` ()
- void `next_module` ()
- `ast_node_t * newDefparam` (ids, ast_node_t *val, int line_number)
- void `newConstant` (char *id, char *number, int line_number)
- void `graphVizOutputAst` (std::string path, ast_node_t *top)
- void `graphVizOutputAst_traverse_node` (FILE *fp, ast_node_t *node, ast_node_t *from, int from_num)
- long `calculate_operation` (ast_node_t *node)

Variables

- int `yylineno`
- `STRING_CACHE * defines_for_file_sc`
- `STRING_CACHE ** defines_for_module_sc`
- `STRING_CACHE * modules_inputs_sc`
- `STRING_CACHE * modules_outputs_sc`
- `STRING_CACHE ** defines_for_function_sc`
- `STRING_CACHE * functions_inputs_sc`
- `STRING_CACHE * functions_outputs_sc`
- `STRING_CACHE * module_names_to_idx`
- `ast_node_t ** block_instantiations_instance`
- int `size_block_instantiations`
- `ast_node_t ** module_instantiations_instance`
- int `size_module_instantiations`
- `ast_node_t ** module_variables_not_defined`
- int `size_module_variables_not_defined`
- `ast_node_t ** function_instantiations_instance`
- int `size_function_instantiations`
- `ast_node_t ** function_instantiations_instance_by_module`
- int `size_function_instantiations_by_module`
- `size_t num_modules`
- `ast_node_t ** ast_modules`
- int `num_functions`
- `ast_node_t ** ast_functions`
- `ast_node_t ** all_file_items_list`
- int `size_all_file_items_list`
- short `to_view_parse`
- int `unique_label_count`

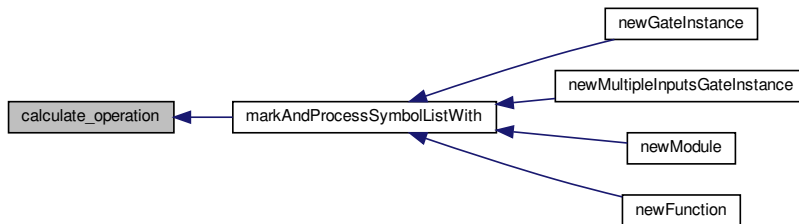
2.12.1 Function Documentation

2.12.1.1 calculate_operation()

```
long calculate_operation (
    ast_node_t * node )
```

Definition at line 2151 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.12.1.2 clean_up_parser_for_file()

```
void clean_up_parser_for_file ( )
```

Definition at line 314 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.12.1.3 cleanup_hard_blocks()

```
void cleanup_hard_blocks ( )
```

Definition at line 222 of file parse_making_ast.cpp.

Here is the caller graph for this function:

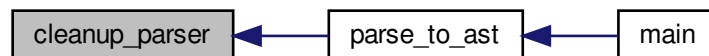


2.12.1.4 cleanup_parser()

```
void cleanup_parser ( )
```

Definition at line 279 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.12.1.5 graphVizOutputAst()

```
void graphVizOutputAst (
    std::string path,
    ast_node_t * top )
```

Definition at line 1805 of file parse_making_ast.cpp.

Here is the caller graph for this function:



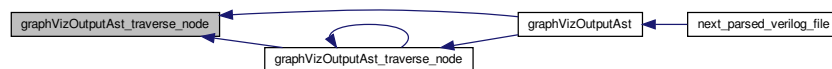
2.12.1.6 graphVizOutputAst_traverse_node()

```

void graphVizOutputAst_traverse_node (
    FILE * fp,
    ast_node_t * node,
    ast_node_t * from,
    int from_num )
  
```

Definition at line 1830 of file `parse_making_ast.cpp`.

Here is the caller graph for this function:



2.12.1.7 graphVizOutputPreproc()

```

void graphVizOutputPreproc (
    FILE * yyin,
    std::string path,
    char * file )
  
```

Definition at line 87 of file `parse_making_ast.cpp`.

Here is the caller graph for this function:

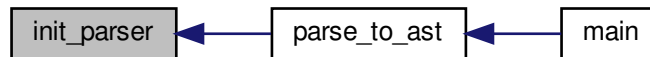


2.12.1.8 init_parser()

```
void init_parser ( )
```

Definition at line 247 of file parse_making_ast.cpp.

Here is the caller graph for this function:

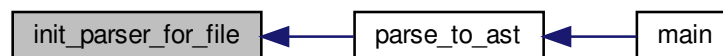


2.12.1.9 init_parser_for_file()

```
void init_parser_for_file ( )
```

Definition at line 298 of file parse_making_ast.cpp.

Here is the caller graph for this function:

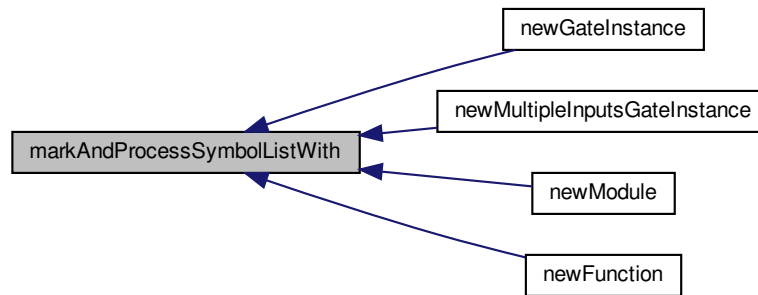


2.12.1.10 markAndProcessSymbolListWith()

```
ast_node_t* markAndProcessSymbolListWith (
    ids top_type,
    ids id,
    ast_node_t * symbol_list )
```

Definition at line 450 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.12.1.11 newAlways()

```
ast_node_t* newAlways (
    ast_node_t * delay_control,
    ast_node_t * statement,
    int line_number )
```

Definition at line 1157 of file `parse_making_ast.cpp`.

2.12.1.12 newArrayRef()

```
ast_node_t* newArrayRef (
    char * id,
    ast_node_t * expression,
    int line_number )
```

Definition at line 847 of file `parse_making_ast.cpp`.

2.12.1.13 newBinaryOperation()

```
ast_node_t* newBinaryOperation (
    operation_list op_id,
    ast_node_t * expression1,
    ast_node_t * expression2,
    int line_number )
```

Definition at line 879 of file `parse_making_ast.cpp`.

2.12.1.14 newBlocking()

```
ast_node_t* newBlocking (
    ast_node_t * expression1,
    ast_node_t * expression2,
    int line_number )
```

Definition at line 1052 of file parse_making_ast.cpp.

2.12.1.15 newCase()

```
ast_node_t* newCase (
    ast_node_t * compare_expression,
    ast_node_t * case_list,
    int line_number )
```

Definition at line 1144 of file parse_making_ast.cpp.

2.12.1.16 newCaseItem()

```
ast_node_t* newCaseItem (
    ast_node_t * expression,
    ast_node_t * statement,
    int line_number )
```

Definition at line 989 of file parse_making_ast.cpp.

2.12.1.17 newConstant()

```
void newConstant (
    char * id,
    char * number,
    int line_number )
```

Definition at line 1776 of file parse_making_ast.cpp.

2.12.1.18 newDefaultCase()

```
ast_node_t* newDefaultCase (
    ast_node_t * statement,
    int line_number )
```

Definition at line 1002 of file parse_making_ast.cpp.

2.12.1.19 newDefparam()

```
ast_node_t* newDefparam (
    ids ,
    ast_node_t * val,
    int line_number )
```

Definition at line 1707 of file parse_making_ast.cpp.

2.12.1.20 newExpandPower()

```
ast_node_t* newExpandPower (
    operation_list op_id,
    ast_node_t * expression1,
    ast_node_t * expression2,
    int line_number )
```

Definition at line 893 of file parse_making_ast.cpp.

2.12.1.21 newFor()

```
ast_node_t* newFor (
    ast_node_t * initial,
    ast_node_t * compare_expression,
    ast_node_t * terminal,
    ast_node_t * statement,
    int line_number )
```

Definition at line 1089 of file parse_making_ast.cpp.

2.12.1.22 newFunction()

```
ast_node_t* newFunction (
    ast_node_t * list_of_ports,
    ast_node_t * list_of_module_items,
    int line_number )
```

Definition at line 1572 of file parse_making_ast.cpp.

2.12.1.23 newFunctionAssigning()

```
ast_node_t * newFunctionAssigning (
    ast_node_t * expression1,
    ast_node_t * expression2,
    int line_number )
```

Definition at line 1064 of file `parse_making_ast.cpp`.

2.12.1.24 newFunctionInstance()

```
ast_node_t* newFunctionInstance (
    char * function_ref_name,
    ast_node_t * function_named_instance,
    int line_number )
```

Definition at line 1344 of file `parse_making_ast.cpp`.

2.12.1.25 newfunctionList()

```
ast_node_t* newfunctionList (
    ids node_type,
    ast_node_t * child )
```

Definition at line 404 of file `parse_making_ast.cpp`.

2.12.1.26 newFunctionNamedInstance()

```
ast_node_t* newFunctionNamedInstance (
    ast_node_t * module_connect_list,
    ast_node_t * module_parameter_list,
    int line_number )
```

Definition at line 1241 of file `parse_making_ast.cpp`.

2.12.1.27 newGate()

```
ast_node_t* newGate (
    operation_list op_id,
    ast_node_t * gate_instance,
    int line_number )
```

Definition at line 1455 of file `parse_making_ast.cpp`.

2.12.1.28 newGateInstance()

```
ast_node_t* newGateInstance (
    char * gate_instance_name,
    ast_node_t * expression1,
    ast_node_t * expression2,
    ast_node_t * expression3,
    int line_number )
```

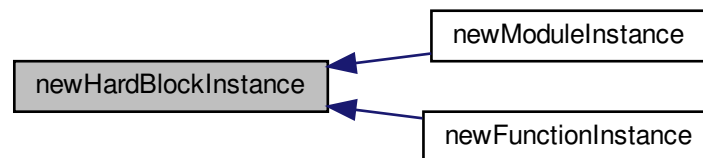
Definition at line 1378 of file parse_making_ast.cpp.

2.12.1.29 newHardBlockInstance()

```
ast_node_t * newHardBlockInstance (
    char * module_ref_name,
    ast_node_t * module_named_instance,
    int line_number )
```

Definition at line 1264 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.12.1.30 newIf()

```
ast_node_t* newIf (
    ast_node_t * compare_expression,
    ast_node_t * true_expression,
    ast_node_t * false_expression,
    int line_number )
```

Definition at line 1119 of file parse_making_ast.cpp.

2.12.1.31 newIfQuestion()

```
ast_node_t* newIfQuestion (
    ast_node_t * compare_expression,
    ast_node_t * true_expression,
    ast_node_t * false_expression,
    int line_number )
```

Definition at line 1132 of file `parse_making_ast.cpp`.

2.12.1.32 newInitial()

```
ast_node_t* newInitial (
    ast_node_t * expression1,
    int line_number )
```

Definition at line 1040 of file `parse_making_ast.cpp`.

2.12.1.33 newIntegerTypeVarDeclare()

```
ast_node_t* newIntegerTypeVarDeclare (
    char * symbol,
    ast_node_t * ,
    ast_node_t * ,
    ast_node_t * expression3,
    ast_node_t * expression4,
    ast_node_t * value,
    int line_number )
```

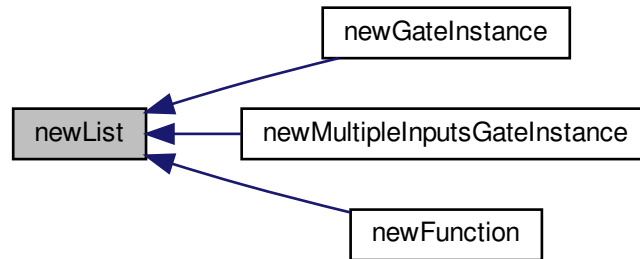
Definition at line 1489 of file `parse_making_ast.cpp`.

2.12.1.34 newList()

```
ast_node_t* newList (
    ids node_type,
    ast_node_t * child )
```

Definition at line 394 of file `parse_making_ast.cpp`.

Here is the caller graph for this function:



2.12.1.35 newList_entry()

```
ast_node_t* newList_entry (  
    ast_node_t * list,  
    ast_node_t * child )
```

Definition at line 418 of file `parse_making_ast.cpp`.

Here is the caller graph for this function:



2.12.1.36 newListReplicate()

```
ast_node_t* newListReplicate (  
    ast_node_t * exp,  
    ast_node_t * child )
```

Definition at line 432 of file `parse_making_ast.cpp`.

2.12.1.37 newModule()

```
ast_node_t* newModule (  
    char * module_name,  
    ast_node_t * list_of_ports,  
    ast_node_t * list_of_module_items,  
    int line_number )
```

Definition at line 1516 of file parse_making_ast.cpp.

2.12.1.38 newModuleConnection()

```
ast_node_t* newModuleConnection (  
    char * id,  
    ast_node_t * expression,  
    int line_number )
```

Definition at line 1170 of file parse_making_ast.cpp.

Here is the caller graph for this function:

**2.12.1.39 newModuleInstance()**

```
ast_node_t* newModuleInstance (  
    char * module_ref_name,  
    ast_node_t * module_named_instance,  
    int line_number )
```

Definition at line 1284 of file parse_making_ast.cpp.

2.12.1.40 newModuleNamedInstance()

```
ast_node_t* newModuleNamedInstance (
    char * unique_name,
    ast_node_t * module_connect_list,
    ast_node_t * module_parameter_list,
    int line_number )
```

Definition at line 1226 of file `parse_making_ast.cpp`.

2.12.1.41 newModuleParameter()

```
ast_node_t* newModuleParameter (
    char * id,
    ast_node_t * expression,
    int line_number )
```

Definition at line 1193 of file `parse_making_ast.cpp`.

2.12.1.42 newMultipleInputsGateInstance()

```
ast_node_t* newMultipleInputsGateInstance (
    char * gate_instance_name,
    ast_node_t * expression1,
    ast_node_t * expression2,
    ast_node_t * expression3,
    int line_number )
```

Definition at line 1409 of file `parse_making_ast.cpp`.

2.12.1.43 newNegedgeSymbol()

```
ast_node_t* newNegedgeSymbol (
    char * symbol,
    int line_number )
```

Definition at line 959 of file `parse_making_ast.cpp`.

2.12.1.44 newNonBlocking()

```
ast_node_t* newNonBlocking (
    ast_node_t * expression1,
    ast_node_t * expression2,
    int line_number )
```

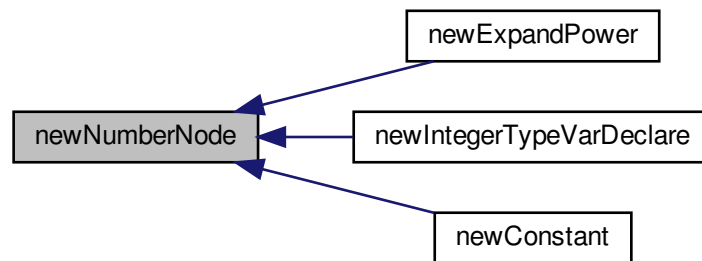
Definition at line 1027 of file `parse_making_ast.cpp`.

2.12.1.45 newNumberNode()

```
ast_node_t* newNumberNode (  
    char * num,  
    int line_number )
```

Definition at line 385 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.12.1.46 newParallelConnection()

```
ast_node_t* newParallelConnection (  
    ast_node_t * expression1,  
    ast_node_t * expression2,  
    int line_number )
```

Definition at line 1015 of file parse_making_ast.cpp.

2.12.1.47 newPosedgeSymbol()

```
ast_node_t* newPosedgeSymbol (  
    char * symbol,  
    int line_number )
```

Definition at line 974 of file parse_making_ast.cpp.

2.12.1.48 newRangeRef()

```
ast_node_t* newRangeRef (
    char * id,
    ast_node_t * expression1,
    ast_node_t * expression2,
    int line_number )
```

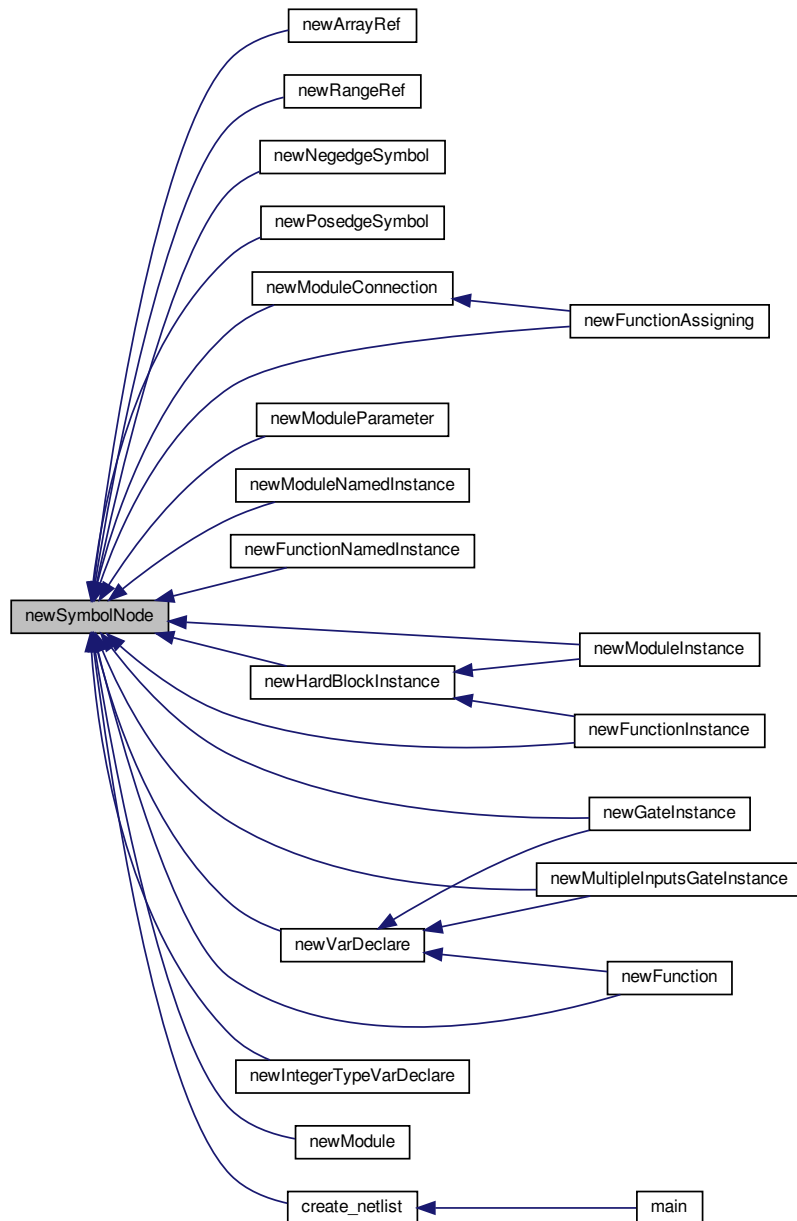
Definition at line 862 of file parse_making_ast.cpp.

2.12.1.49 newSymbolNode()

```
ast_node_t* newSymbolNode (
    char * id,
    int line_number )
```

Definition at line 348 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.12.1.50 newUnaryOperation()

```

ast_node_t* newUnaryOperation (
    operation_list op_id,

```

```
ast_node_t * expression,  
int line_number )
```

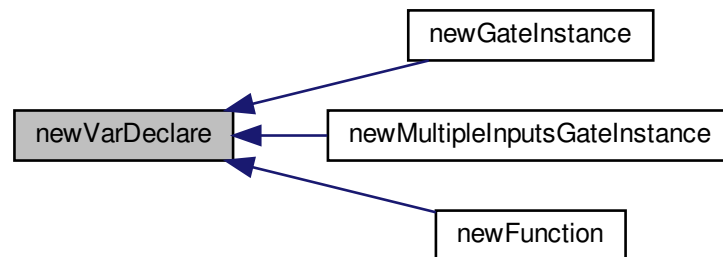
Definition at line 941 of file parse_making_ast.cpp.

2.12.1.51 newVarDeclare()

```
ast_node_t* newVarDeclare (  
    char * symbol,  
    ast_node_t * expression1,  
    ast_node_t * expression2,  
    ast_node_t * expression3,  
    ast_node_t * expression4,  
    ast_node_t * value,  
    int line_number )
```

Definition at line 1472 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.12.1.52 newWhile()

```
ast_node_t* newWhile (  
    ast_node_t * compare_expression,  
    ast_node_t * statement,  
    int line_number )
```

Definition at line 1102 of file parse_making_ast.cpp.

2.12.1.53 next_function()

```
void next_function ( )
```

Definition at line 1655 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.12.1.54 next_module()

```
void next_module ( )
```

Definition at line 1678 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.12.1.55 next_parsed_verilog_file()

```
void next_parsed_verilog_file (
    ast_node_t * file_items_list )
```

Definition at line 323 of file parse_making_ast.cpp.

2.12.1.56 parse_to_ast()

```
void parse_to_ast ( )
```

Definition at line 116 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.12.2 Variable Documentation

2.12.2.1 all_file_items_list

```
ast_node_t** all_file_items_list
```

Definition at line 72 of file parse_making_ast.cpp.

2.12.2.2 ast_functions

```
ast_node_t** ast_functions
```

Definition at line 71 of file parse_making_ast.cpp.

2.12.2.3 ast_modules

```
ast_node_t** ast_modules
```

Definition at line 68 of file parse_making_ast.cpp.

2.12.2.4 block_instantiations_instance

```
ast_node_t** block_instantiations_instance
```

Definition at line 55 of file parse_making_ast.cpp.

2.12.2.5 defines_for_file_sc

```
STRING_CACHE* defines_for_file_sc
```

Definition at line 43 of file parse_making_ast.cpp.

2.12.2.6 defines_for_function_sc

```
STRING_CACHE** defines_for_function_sc
```

Definition at line 49 of file parse_making_ast.cpp.

2.12.2.7 defines_for_module_sc

```
STRING_CACHE** defines_for_module_sc
```

Definition at line 45 of file parse_making_ast.cpp.

2.12.2.8 function_instantiations_instance

```
ast_node_t** function_instantiations_instance
```

Definition at line 62 of file parse_making_ast.cpp.

2.12.2.9 function_instantiations_instance_by_module

```
ast_node_t** function_instantiations_instance_by_module
```

Definition at line 64 of file parse_making_ast.cpp.

2.12.2.10 functions_inputs_sc

`STRING_CACHE*` functions_inputs_sc

Definition at line 50 of file parse_making_ast.cpp.

2.12.2.11 functions_outputs_sc

`STRING_CACHE*` functions_outputs_sc

Definition at line 51 of file parse_making_ast.cpp.

2.12.2.12 module_instantiations_instance

`ast_node_t**` module_instantiations_instance

Definition at line 58 of file parse_making_ast.cpp.

2.12.2.13 module_names_to_idx

`STRING_CACHE*` module_names_to_idx

Definition at line 53 of file parse_making_ast.cpp.

2.12.2.14 module_variables_not_defined

`ast_node_t**` module_variables_not_defined

Definition at line 60 of file parse_making_ast.cpp.

2.12.2.15 modules_inputs_sc

`STRING_CACHE*` modules_inputs_sc

Definition at line 46 of file parse_making_ast.cpp.

2.12.2.16 modules_outputs_sc

```
STRING_CACHE* modules_outputs_sc
```

Definition at line 47 of file parse_making_ast.cpp.

2.12.2.17 num_functions

```
int num_functions
```

Definition at line 70 of file parse_making_ast.cpp.

2.12.2.18 num_modules

```
size_t num_modules
```

Definition at line 67 of file parse_making_ast.cpp.

2.12.2.19 size_all_file_items_list

```
int size_all_file_items_list
```

Definition at line 73 of file parse_making_ast.cpp.

2.12.2.20 size_block_instantiations

```
int size_block_instantiations
```

Definition at line 56 of file parse_making_ast.cpp.

2.12.2.21 size_function_instantiations

```
int size_function_instantiations
```

Definition at line 63 of file parse_making_ast.cpp.

2.12.2.22 size_function_instantiations_by_module

```
int size_function_instantiations_by_module
```

Definition at line 65 of file parse_making_ast.cpp.

2.12.2.23 size_module_instantiations

```
int size_module_instantiations
```

Definition at line 59 of file parse_making_ast.cpp.

2.12.2.24 size_module_variables_not_defined

```
int size_module_variables_not_defined
```

Definition at line 61 of file parse_making_ast.cpp.

2.12.2.25 to_view_parse

```
short to_view_parse
```

Definition at line 75 of file parse_making_ast.cpp.

2.12.2.26 unique_label_count

```
int unique_label_count
```

Definition at line 1801 of file parse_making_ast.cpp.

2.12.2.27 yylineno

```
int yylineno
```

2.13 vtr-verilog-to-routing/ODIN_II/SRC/lib_ast/parse_making_ast.h File Reference

```
#include "types.h"
```

Functions

- void [parse_to_ast](#) ()
- void [init_parser](#) ()
- void [cleanup_parser](#) ()
- void [cleanup_hard_blocks](#) ()
- void [init_parser_for_file](#) ()
- void [clean_up_parser_for_file](#) ()
- ast_node_t * [newSymbolNode](#) (char *id, int line_number)
- ast_node_t * [newNumberNode](#) (char *num, int line_number)
- ast_node_t * [newList](#) (ids type_id, ast_node_t *expression)
- ast_node_t * [newList_entry](#) (ast_node_t *concat_node, ast_node_t *expression)
- ast_node_t * [newListReplicate](#) (ast_node_t *exp, ast_node_t *child)
- ast_node_t * [markAndProcessSymbolListWith](#) (ids top_type, ids id, ast_node_t *symbol_list)
- ast_node_t * [newArrayRef](#) (char *id, ast_node_t *expression, int line_number)
- ast_node_t * [newRangeRef](#) (char *id, ast_node_t *expression1, ast_node_t *expression2, int line_number)
- ast_node_t * [newBinaryOperation](#) (operation_list op_id, ast_node_t *expression1, ast_node_t *expression2, int line_number)
- ast_node_t * [newExpandPower](#) (operation_list op_id, ast_node_t *expression1, ast_node_t *expression2, int line_number)
- ast_node_t * [newUnaryOperation](#) (operation_list op_id, ast_node_t *expression, int line_number)
- ast_node_t * [newPosedgeSymbol](#) (char *symbol, int line_number)
- ast_node_t * [newNegedgeSymbol](#) (char *symbol, int line_number)
- ast_node_t * [newCaseItem](#) (ast_node_t *expression, ast_node_t *statement, int line_number)
- ast_node_t * [newDefaultCase](#) (ast_node_t *statement, int line_number)
- ast_node_t * [newNonBlocking](#) (ast_node_t *expression1, ast_node_t *expression2, int line_number)
- ast_node_t * [newInitial](#) (ast_node_t *expression1, int line_number)
- ast_node_t * [newBlocking](#) (ast_node_t *expression1, ast_node_t *expression2, int line_number)
- ast_node_t * [newIf](#) (ast_node_t *compare_expression, ast_node_t *true_expression, ast_node_t *false_↵ expression, int line_number)
- ast_node_t * [newIfQuestion](#) (ast_node_t *compare_expression, ast_node_t *true_expression, ast_node_↵ t *false_expression, int line_number)
- ast_node_t * [newCase](#) (ast_node_t *compare_expression, ast_node_t *case_list, int line_number)
- ast_node_t * [newAlways](#) (ast_node_t *delay_control, ast_node_t *statements, int line_number)
- ast_node_t * [newFor](#) (ast_node_t *initial, ast_node_t *compare_expression, ast_node_t *terminal, ast_node_t *statement, int line_number)
- ast_node_t * [newWhile](#) (ast_node_t *compare_expression, ast_node_t *statement, int line_number)
- ast_node_t * [newModuleConnection](#) (char *id, ast_node_t *expression, int line_number)
- ast_node_t * [newModuleNamedInstance](#) (char *unique_name, ast_node_t *module_connect_list, ast_node_t *module_parameter_list, int line_number)
- ast_node_t * [newFunctionNamedInstance](#) (ast_node_t *module_connect_list, ast_node_t *module_parameter_↵ _list, int line_number)
- ast_node_t * [newModuleInstance](#) (char *module_ref_name, ast_node_t *module_named_instance, int line_↵ number)
- ast_node_t * [newFunctionInstance](#) (char *function_ref_name, ast_node_t *function_named_instance, int line_↵ _number)
- ast_node_t * [newModuleParameter](#) (char *id, ast_node_t *expression, int line_number)
- ast_node_t * [newfunctionList](#) (ids node_type, ast_node_t *child)
- ast_node_t * [newParallelConnection](#) (ast_node_t *expression1, ast_node_t *expression2, int line_number)
- ast_node_t * [newGateInstance](#) (char *gate_instance_name, ast_node_t *expression1, ast_node_t *expression2, ast_node_t *expression3, int line_number)

- `ast_node_t * newMultipleInputsGateInstance` (`char *gate_instance_name`, `ast_node_t *expression1`, `ast_node_t *expression2`, `ast_node_t *expression3`, `int line_number`)
- `ast_node_t * newGate` (`operation_list gate_type`, `ast_node_t *gate_instance`, `int line_number`)
- `ast_node_t * newAssign` (`ast_node_t *statement`, `int line_number`)
- `ast_node_t * newVarDeclare` (`char *symbol`, `ast_node_t *expression1`, `ast_node_t *expression2`, `ast_node_t *expression3`, `ast_node_t *expression4`, `ast_node_t *value`, `int line_number`)
- `ast_node_t * newIntegerTypeVarDeclare` (`char *symbol`, `ast_node_t *expression1`, `ast_node_t *expression2`, `ast_node_t *expression3`, `ast_node_t *expression4`, `ast_node_t *value`, `int line_number`)
- `ast_node_t * newModule` (`char *module_name`, `ast_node_t *list_of_ports`, `ast_node_t *list_of_module_items`, `int line_number`)
- `ast_node_t * newFunction` (`ast_node_t *list_of_ports`, `ast_node_t *list_of_module_items`, `int line_number`)
- `void next_module` ()
- `void next_function` ()
- `void newConstant` (`char *id`, `char *number`, `int line_number`)
- `ast_node_t * newDefparam` (`ids id`, `ast_node_t *val`, `int line_number`)
- `void next_parsed_verilog_file` (`ast_node_t *file_items_list`)
- `void graphVizOutputAst` (`std::string path`, `ast_node_t *top`)
- `void graphVizOutputAst_traverse_node` (`FILE *fp`, `ast_node_t *node`, `ast_node_t *from`, `int from_num`)
- `long calculate_operation` (`ast_node_t *node`)

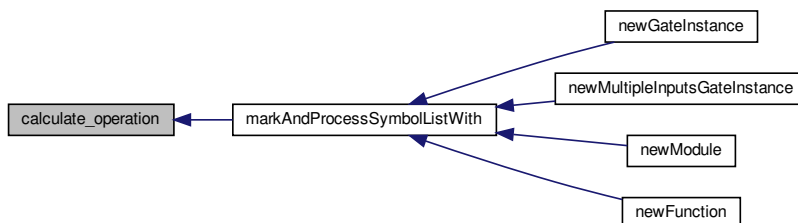
2.13.1 Function Documentation

2.13.1.1 calculate_operation()

```
long calculate_operation (
    ast_node_t * node )
```

Definition at line 2151 of file `parse_making_ast.cpp`.

Here is the caller graph for this function:



2.13.1.2 `clean_up_parser_for_file()`

```
void clean_up_parser_for_file ( )
```

Definition at line 314 of file `parse_making_ast.cpp`.

Here is the caller graph for this function:



2.13.1.3 `cleanup_hard_blocks()`

```
void cleanup_hard_blocks ( )
```

Definition at line 222 of file `parse_making_ast.cpp`.

Here is the caller graph for this function:

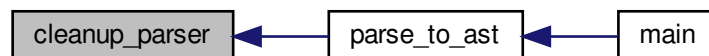


2.13.1.4 `cleanup_parser()`

```
void cleanup_parser ( )
```

Definition at line 279 of file `parse_making_ast.cpp`.

Here is the caller graph for this function:



2.13.1.5 graphVizOutputAst()

```
void graphVizOutputAst (
    std::string path,
    ast_node_t * top )
```

Definition at line 1805 of file parse_making_ast.cpp.

Here is the caller graph for this function:

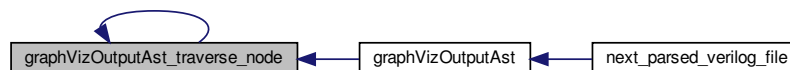


2.13.1.6 graphVizOutputAst_traverse_node()

```
void graphVizOutputAst_traverse_node (
    FILE * fp,
    ast_node_t * node,
    ast_node_t * from,
    int from_num )
```

Definition at line 1830 of file parse_making_ast.cpp.

Here is the caller graph for this function:

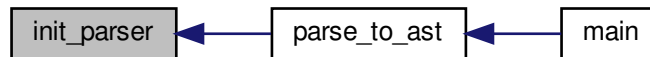


2.13.1.7 init_parser()

```
void init_parser ( )
```

Definition at line 247 of file parse_making_ast.cpp.

Here is the caller graph for this function:

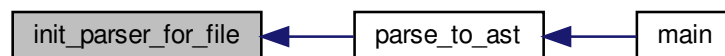


2.13.1.8 init_parser_for_file()

```
void init_parser_for_file ( )
```

Definition at line 298 of file parse_making_ast.cpp.

Here is the caller graph for this function:

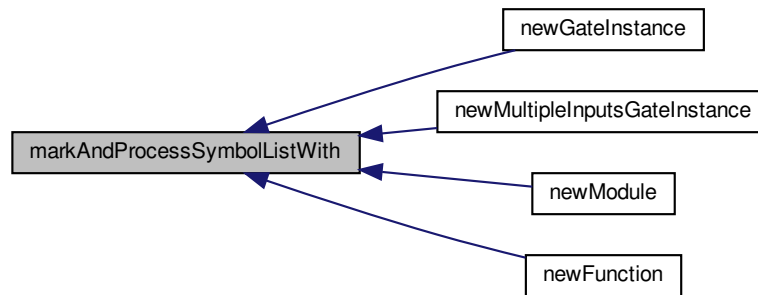


2.13.1.9 markAndProcessSymbolListWith()

```
ast_node_t* markAndProcessSymbolListWith (
    ids top_type,
    ids id,
    ast_node_t * symbol_list )
```

Definition at line 450 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.13.1.10 `newAlways()`

```
ast_node_t* newAlways (
    ast_node_t * delay_control,
    ast_node_t * statements,
    int line_number )
```

Definition at line 1157 of file `parse_making_ast.cpp`.

2.13.1.11 `newArrayRef()`

```
ast_node_t* newArrayRef (
    char * id,
    ast_node_t * expression,
    int line_number )
```

Definition at line 847 of file `parse_making_ast.cpp`.

2.13.1.12 `newAssign()`

```
ast_node_t* newAssign (
    ast_node_t * statement,
    int line_number )
```

2.13.1.13 newBinaryOperation()

```
ast_node_t* newBinaryOperation (
    operation_list op_id,
    ast_node_t * expression1,
    ast_node_t * expression2,
    int line_number )
```

Definition at line 879 of file parse_making_ast.cpp.

2.13.1.14 newBlocking()

```
ast_node_t* newBlocking (
    ast_node_t * expression1,
    ast_node_t * expression2,
    int line_number )
```

Definition at line 1052 of file parse_making_ast.cpp.

2.13.1.15 newCase()

```
ast_node_t* newCase (
    ast_node_t * compare_expression,
    ast_node_t * case_list,
    int line_number )
```

Definition at line 1144 of file parse_making_ast.cpp.

2.13.1.16 newCaseItem()

```
ast_node_t* newCaseItem (
    ast_node_t * expression,
    ast_node_t * statement,
    int line_number )
```

Definition at line 989 of file parse_making_ast.cpp.

2.13.1.17 newConstant()

```
void newConstant (
    char * id,
    char * number,
    int line_number )
```

Definition at line 1776 of file parse_making_ast.cpp.

2.13.1.18 newDefaultCase()

```
ast_node_t* newDefaultCase (
    ast_node_t * statement,
    int line_number )
```

Definition at line 1002 of file `parse_making_ast.cpp`.

2.13.1.19 newDefparam()

```
ast_node_t* newDefparam (
    ids id,
    ast_node_t * val,
    int line_number )
```

Definition at line 1707 of file `parse_making_ast.cpp`.

2.13.1.20 newExpandPower()

```
ast_node_t* newExpandPower (
    operation_list op_id,
    ast_node_t * expression1,
    ast_node_t * expression2,
    int line_number )
```

Definition at line 893 of file `parse_making_ast.cpp`.

2.13.1.21 newFor()

```
ast_node_t* newFor (
    ast_node_t * initial,
    ast_node_t * compare_expression,
    ast_node_t * terminal,
    ast_node_t * statement,
    int line_number )
```

Definition at line 1089 of file `parse_making_ast.cpp`.

2.13.1.22 newFunction()

```
ast_node_t* newFunction (
    ast_node_t * list_of_ports,
    ast_node_t * list_of_module_items,
    int line_number )
```

Definition at line 1572 of file `parse_making_ast.cpp`.

2.13.1.23 newFunctionInstance()

```
ast_node_t* newFunctionInstance (
    char * function_ref_name,
    ast_node_t * function_named_instance,
    int line_number )
```

Definition at line 1344 of file `parse_making_ast.cpp`.

2.13.1.24 newfunctionList()

```
ast_node_t* newfunctionList (
    ids node_type,
    ast_node_t * child )
```

Definition at line 404 of file `parse_making_ast.cpp`.

2.13.1.25 newFunctionNamedInstance()

```
ast_node_t* newFunctionNamedInstance (
    ast_node_t * module_connect_list,
    ast_node_t * module_parameter_list,
    int line_number )
```

Definition at line 1241 of file `parse_making_ast.cpp`.

2.13.1.26 newGate()

```
ast_node_t* newGate (
    operation_list gate_type,
    ast_node_t * gate_instance,
    int line_number )
```

Definition at line 1455 of file `parse_making_ast.cpp`.

2.13.1.27 newGateInstance()

```
ast_node_t* newGateInstance (
    char * gate_instance_name,
    ast_node_t * expression1,
    ast_node_t * expression2,
    ast_node_t * expression3,
    int line_number )
```

Definition at line 1378 of file `parse_making_ast.cpp`.

2.13.1.28 newIf()

```
ast_node_t* newIf (
    ast_node_t * compare_expression,
    ast_node_t * true_expression,
    ast_node_t * false_expression,
    int line_number )
```

Definition at line 1119 of file `parse_making_ast.cpp`.

2.13.1.29 newIfQuestion()

```
ast_node_t* newIfQuestion (
    ast_node_t * compare_expression,
    ast_node_t * true_expression,
    ast_node_t * false_expression,
    int line_number )
```

Definition at line 1132 of file `parse_making_ast.cpp`.

2.13.1.30 newInitial()

```
ast_node_t* newInitial (
    ast_node_t * expression1,
    int line_number )
```

Definition at line 1040 of file `parse_making_ast.cpp`.

2.13.1.31 newIntegerTypeVarDeclare()

```
ast_node_t* newIntegerTypeVarDeclare (
    char * symbol,
    ast_node_t * expression1,
    ast_node_t * expression2,
    ast_node_t * expression3,
    ast_node_t * expression4,
    ast_node_t * value,
    int line_number )
```

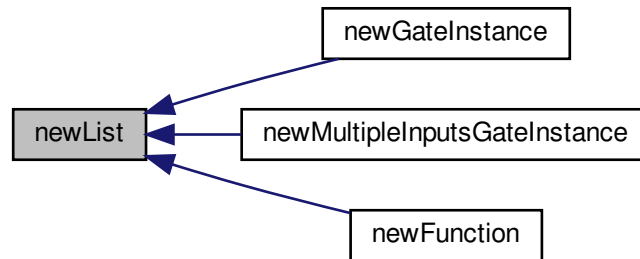
Definition at line 1489 of file `parse_making_ast.cpp`.

2.13.1.32 newList()

```
ast_node_t* newList (
    ids type_id,
    ast_node_t * expression )
```

Definition at line 394 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.13.1.33 newList_entry()

```
ast_node_t* newList_entry (
    ast_node_t * concat_node,
    ast_node_t * expression )
```

Definition at line 418 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.13.1.34 newListReplicate()

```
ast_node_t* newListReplicate (
    ast_node_t * exp,
    ast_node_t * child )
```

Definition at line 432 of file parse_making_ast.cpp.

2.13.1.35 newModule()

```
ast_node_t* newModule (
    char * module_name,
    ast_node_t * list_of_ports,
    ast_node_t * list_of_module_items,
    int line_number )
```

Definition at line 1516 of file parse_making_ast.cpp.

2.13.1.36 newModuleConnection()

```
ast_node_t* newModuleConnection (
    char * id,
    ast_node_t * expression,
    int line_number )
```

Definition at line 1170 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.13.1.37 newModuleInstance()

```
ast_node_t* newModuleInstance (
    char * module_ref_name,
    ast_node_t * module_named_instance,
    int line_number )
```

Definition at line 1284 of file parse_making_ast.cpp.

2.13.1.38 newModuleNamedInstance()

```
ast_node_t* newModuleNamedInstance (
    char * unique_name,
    ast_node_t * module_connect_list,
    ast_node_t * module_parameter_list,
    int line_number )
```

Definition at line 1226 of file `parse_making_ast.cpp`.

2.13.1.39 newModuleParameter()

```
ast_node_t* newModuleParameter (
    char * id,
    ast_node_t * expression,
    int line_number )
```

Definition at line 1193 of file `parse_making_ast.cpp`.

2.13.1.40 newMultipleInputsGateInstance()

```
ast_node_t* newMultipleInputsGateInstance (
    char * gate_instance_name,
    ast_node_t * expression1,
    ast_node_t * expression2,
    ast_node_t * expression3,
    int line_number )
```

Definition at line 1409 of file `parse_making_ast.cpp`.

2.13.1.41 newNegedgeSymbol()

```
ast_node_t* newNegedgeSymbol (
    char * symbol,
    int line_number )
```

Definition at line 959 of file `parse_making_ast.cpp`.

2.13.1.42 newNonBlocking()

```
ast_node_t* newNonBlocking (
    ast_node_t * expression1,
    ast_node_t * expression2,
    int line_number )
```

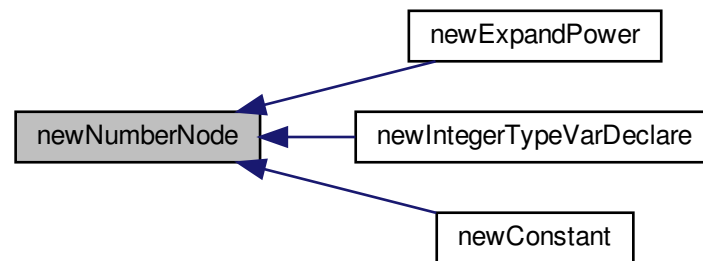
Definition at line 1027 of file `parse_making_ast.cpp`.

2.13.1.43 newNumberNode()

```
ast_node_t* newNumberNode (  
    char * num,  
    int line_number )
```

Definition at line 385 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.13.1.44 newParallelConnection()

```
ast_node_t* newParallelConnection (  
    ast_node_t * expression1,  
    ast_node_t * expression2,  
    int line_number )
```

Definition at line 1015 of file parse_making_ast.cpp.

2.13.1.45 newPosedgeSymbol()

```
ast_node_t* newPosedgeSymbol (  
    char * symbol,  
    int line_number )
```

Definition at line 974 of file parse_making_ast.cpp.

2.13.1.46 newRangeRef()

```
ast_node_t* newRangeRef (
    char * id,
    ast_node_t * expression1,
    ast_node_t * expression2,
    int line_number )
```

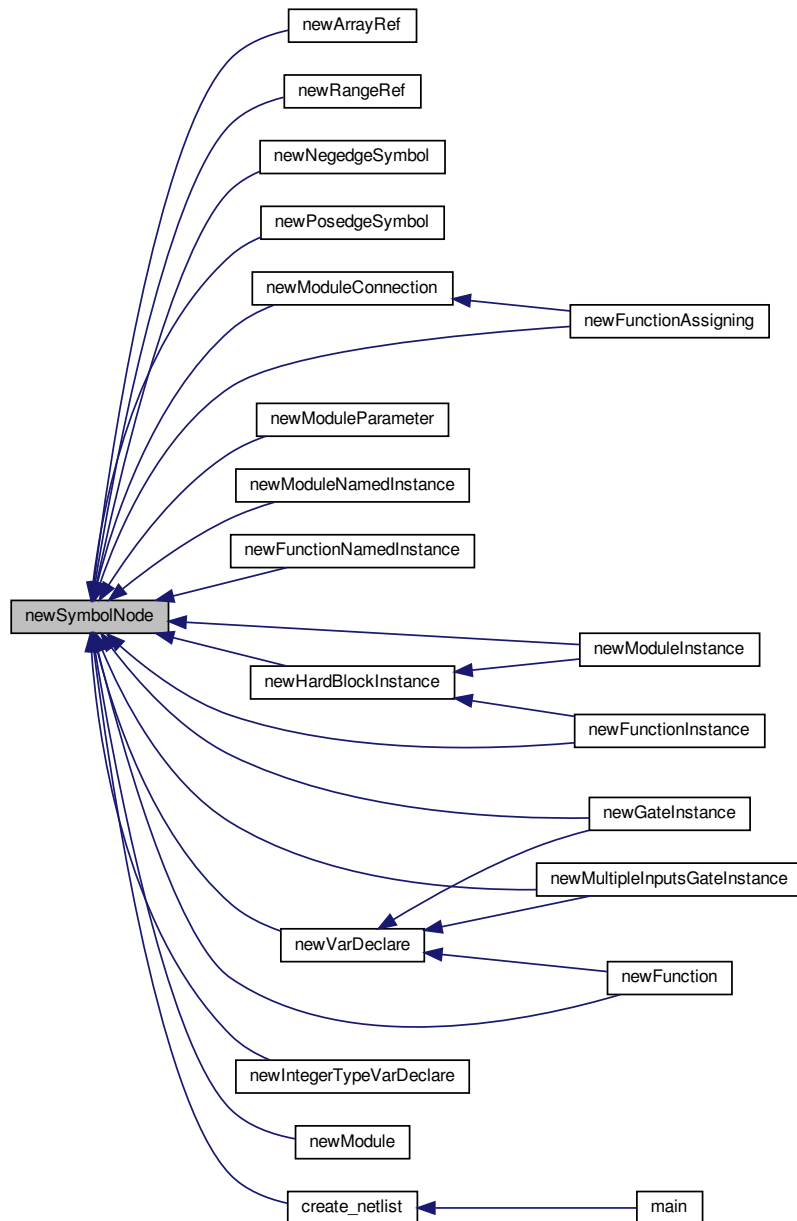
Definition at line 862 of file parse_making_ast.cpp.

2.13.1.47 newSymbolNode()

```
ast_node_t* newSymbolNode (
    char * id,
    int line_number )
```

Definition at line 348 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.13.1.48 newUnaryOperation()

```

ast_node_t* newUnaryOperation (
    operation_list op_id,

```

```
ast_node_t * expression,  
int line_number )
```

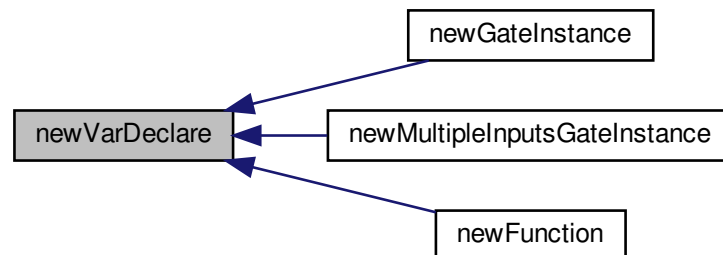
Definition at line 941 of file parse_making_ast.cpp.

2.13.1.49 newVarDeclare()

```
ast_node_t* newVarDeclare (  
    char * symbol,  
    ast_node_t * expression1,  
    ast_node_t * expression2,  
    ast_node_t * expression3,  
    ast_node_t * expression4,  
    ast_node_t * value,  
    int line_number )
```

Definition at line 1472 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.13.1.50 newWhile()

```
ast_node_t* newWhile (  
    ast_node_t * compare_expression,  
    ast_node_t * statement,  
    int line_number )
```

Definition at line 1102 of file parse_making_ast.cpp.

2.13.1.51 next_function()

```
void next_function ( )
```

Definition at line 1655 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.13.1.52 next_module()

```
void next_module ( )
```

Definition at line 1678 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.13.1.53 next_parsed_verilog_file()

```
void next_parsed_verilog_file (
    ast_node_t * file_items_list )
```

Definition at line 323 of file parse_making_ast.cpp.

2.13.1.54 parse_to_ast()

```
void parse_to_ast ( )
```

Definition at line 116 of file parse_making_ast.cpp.

Here is the caller graph for this function:



2.14 vtr-verilog-to-routing/ODIN_II/SRC/lib_blif/output_blif.cpp File Reference

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include "types.h"
#include "globals.h"
#include "netlist_utils.h"
#include "odin_util.h"
#include "multipliers.h"
#include "hard_blocks.h"
#include "adders.h"
#include "subtractions.h"
#include "vtr_util.h"
#include "vtr_memory.h"
```

Functions

- void [depth_first_traversal_to_output](#) (short marker_value, FILE *fp, netlist_t *netlist)
- void [depth_traverse_output_blif](#) (nnode_t *node, int traverse_mark_number, FILE *fp)
- void [output_node](#) (nnode_t *node, short traverse_number, FILE *fp)
- void [define_logical_function](#) (nnode_t *node, short type, FILE *out)
- void [define_set_input_logical_function](#) (nnode_t *node, const char *bit_output, FILE *out)
- void [define_ff](#) (nnode_t *node, FILE *out)
- void [define_decoded_mux](#) (nnode_t *node, FILE *out)
- void [output_blif_pin_connect](#) (nnode_t *node, FILE *out)
- void [output_blif](#) (char *file_name, netlist_t *netlist)

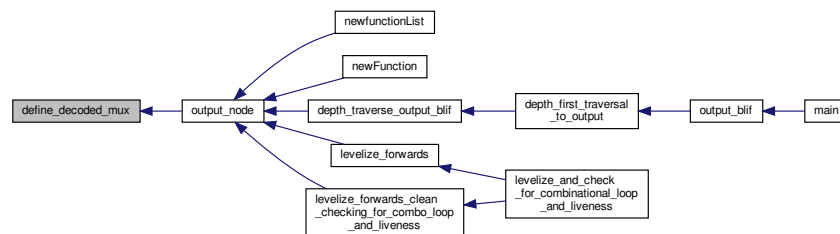
2.14.1 Function Documentation

2.14.1.1 define_decoded_mux()

```
void define_decoded_mux (  
    nnode_t * node,  
    FILE * out )
```

Definition at line 703 of file output_blif.cpp.

Here is the caller graph for this function:

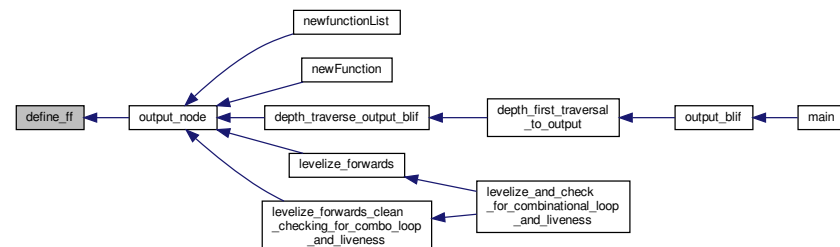


2.14.1.2 define_ff()

```
void define_ff (  
    nnode_t * node,  
    FILE * out )
```

Definition at line 663 of file output_blif.cpp.

Here is the caller graph for this function:

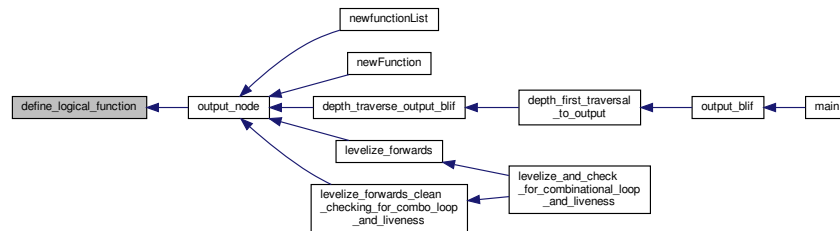


2.14.1.3 define_logical_function()

```
void define_logical_function (
    nnode_t * node,
    short type,
    FILE * out )
```

Definition at line 402 of file output_blif.cpp.

Here is the caller graph for this function:

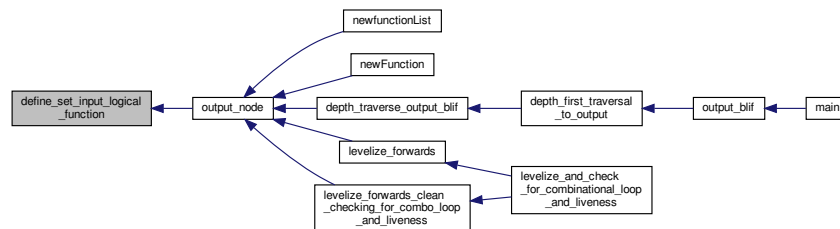


2.14.1.4 define_set_input_logical_function()

```
void define_set_input_logical_function (
    nnode_t * node,
    const char * bit_output,
    FILE * out )
```

Definition at line 575 of file output_blif.cpp.

Here is the caller graph for this function:

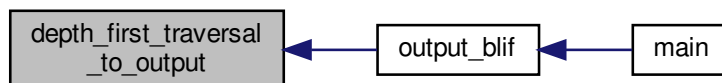


2.14.1.5 depth_first_traversal_to_output()

```
void depth_first_traversal_to_output (
    short marker_value,
    FILE * fp,
    netlist_t * netlist )
```

Definition at line 228 of file output_blif.cpp.

Here is the caller graph for this function:

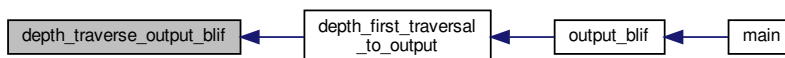


2.14.1.6 depth_traverse_output_blif()

```
void depth_traverse_output_blif (
    nnode_t * node,
    int traverse_mark_number,
    FILE * fp )
```

Definition at line 253 of file output_blif.cpp.

Here is the caller graph for this function:



2.14.1.7 output_blif()

```
void output_blif (
    char * file_name,
    netlist_t * netlist )
```

Definition at line 55 of file output_blif.cpp.

Here is the caller graph for this function:

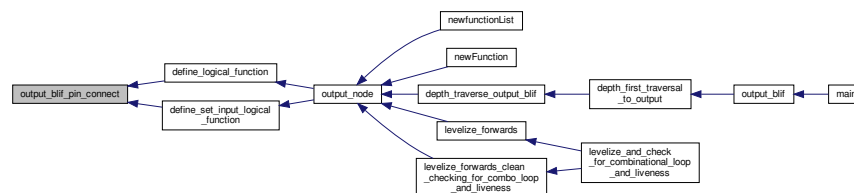


2.14.1.8 output_blif_pin_connect()

```
void output_blif_pin_connect (
    nnode_t * node,
    FILE * out )
```

Definition at line 803 of file output_blif.cpp.

Here is the caller graph for this function:

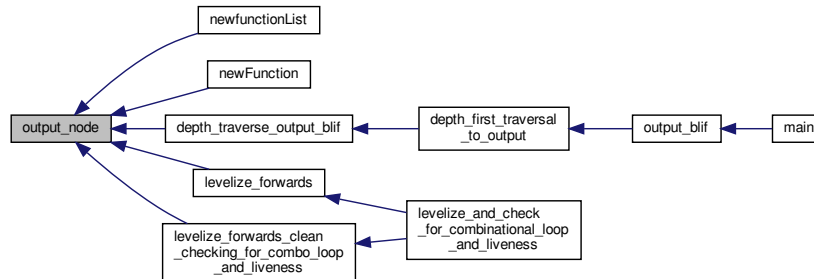


2.14.1.9 output_node()

```
void output_node (
    nnode_t * node,
    short traverse_number,
    FILE * fp )
```

Definition at line 299 of file output_blif.cpp.

Here is the caller graph for this function:



2.15 vtr-verilog-to-routing/ODIN_II/SRC/lib_blif/output_blif.h File Reference

Functions

- void `output_blif` (char *file_name, netlist_t *netlist)

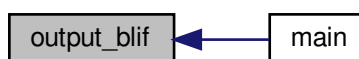
2.15.1 Function Documentation

2.15.1.1 output_blif()

```
void output_blif (
    char * file_name,
    netlist_t * netlist )
```

Definition at line 55 of file output_blif.cpp.

Here is the caller graph for this function:



2.16 vtr-verilog-to-routing/ODIN_II/SRC/lib_blif/read_blif.cpp File Reference

```
#include <stdlib.h>
#include <string.h>
#include <stdio.h>
#include "globals.h"
#include "read_blif.h"
#include "string_cache.h"
#include "netlist_utils.h"
#include "types.h"
#include "hashtable.h"
#include "netlist_check.h"
#include "simulate_blif.h"
#include "vtr_util.h"
#include "vtr_memory.h"
```

Data Structures

- struct [hard_block_pins](#)
- struct [hard_block_ports](#)
- struct [hard_block_model](#)
- struct [hard_block_models](#)

Macros

- #define [TOKENS](#) "\t\n"
- #define [GND_NAME](#) "gnd"
- #define [VCC_NAME](#) "vcc"
- #define [HBPAD_NAME](#) "unconn"
- #define [READ_BLIF_BUFFER](#) 1048576

Functions

- void [rb_create_top_driver_nets](#) (const char *instance_name_prefix, [hashtable_t](#) *output_nets_hash)
- void [rb_look_for_clocks](#) ()
- void [add_top_input_nodes](#) (FILE *file, [hashtable_t](#) *output_nets_hash)
- void [rb_create_top_output_nodes](#) (FILE *file)
- int [read_tokens](#) (char *buffer, [hard_block_models](#) *models, FILE *file, [hashtable_t](#) *output_nets_hash)
- void [create_internal_node_and_driver](#) (FILE *file, [hashtable_t](#) *output_nets_hash)
- short [assign_node_type_from_node_name](#) (char *output_name)
- short [read_bit_map_find_unknown_gate](#) (int input_count, [nnode_t](#) *node, FILE *file)
- void [create_latch_node_and_driver](#) (FILE *file, [hashtable_t](#) *output_nets_hash)
- void [create_hard_block_nodes](#) ([hard_block_models](#) *models, FILE *file, [hashtable_t](#) *output_nets_hash)
- void [hook_up_nets](#) ([hashtable_t](#) *output_nets_hash)
- void [hook_up_node](#) ([nnode_t](#) *node, [hashtable_t](#) *output_nets_hash)
- char * [search_clock_name](#) (FILE *file)
- void [free_hard_block_model](#) ([hard_block_model](#) *model)
- char * [get_hard_block_port_name](#) (char *name)

- long `get_hard_block_pin_number` (char *original_name)
- `hard_block_ports` * `get_hard_block_ports` (char **pins, int count)
- `hashtable_t` * `index_names` (char **names, int count)
- `hashtable_t` * `associate_names` (char **names1, char **names2, int count)
- void `free_hard_block_pins` (`hard_block_pins` *p)
- void `free_hard_block_ports` (`hard_block_ports` *p)
- `hard_block_model` * `get_hard_block_model` (char *name, `hard_block_ports` *ports, `hard_block_models` *models)
- void `add_hard_block_model` (`hard_block_model` *m, `hard_block_ports` *ports, `hard_block_models` *models)
- char * `generate_hard_block_ports_signature` (`hard_block_ports` *ports)
- int `verify_hard_block_ports_against_model` (`hard_block_ports` *ports, `hard_block_model` *model)
- `hard_block_model` * `read_hard_block_model` (char *name_subckt, `hard_block_ports` *ports, FILE *file)
- void `free_hard_block_models` (`hard_block_models` *models)
- `hard_block_models` * `create_hard_block_models` ()
- int `count_blif_lines` (FILE *file)
- void `read_blif` (char *blif_file)

Variables

- size_t `file_line_number`
- const char * `BLIF_ONE_STRING` = "ONE_VCC_CNS"
- const char * `BLIF_ZERO_STRING` = "ZERO_GND_ZERO"
- const char * `BLIF_PAD_STRING` = "ZERO_PAD_ZERO"
- const char * `DEFAULT_CLOCK_NAME` = "top^clock"

2.16.1 Macro Definition Documentation

2.16.1.1 GND_NAME

```
#define GND_NAME "gnd"
```

Definition at line 41 of file read_blif.cpp.

2.16.1.2 HBPAD_NAME

```
#define HBPAD_NAME "unconn"
```

Definition at line 43 of file read_blif.cpp.

2.16.1.3 READ_BLIF_BUFFER

```
#define READ_BLIF_BUFFER 1048576
```

Definition at line 45 of file read_blif.cpp.

2.16.1.4 TOKENS

```
#define TOKENS " \t\n"
```

Definition at line 40 of file read_blif.cpp.

2.16.1.5 VCC_NAME

```
#define VCC_NAME "vcc"
```

Definition at line 42 of file read_blif.cpp.

2.16.2 Function Documentation

2.16.2.1 add_hard_block_model()

```
void add_hard_block_model (  
    hard_block_model * m,  
    hard_block_ports * ports,  
    hard_block_models * models )
```

Definition at line 1572 of file read_blif.cpp.

2.16.2.2 add_top_input_nodes()

```
void add_top_input_nodes (  
    FILE * file,  
    hashtable_t * output_nets_hash )
```

Definition at line 1021 of file read_blif.cpp.

2.16.2.3 assign_node_type_from_node_name()

```
short assign_node_type_from_node_name (  
    char * output_name )
```

Definition at line 249 of file read_blif.cpp.

2.16.2.4 associate_names()

```
hashtable_t * associate_names (  
    char ** names1,  
    char ** names2,  
    int count )
```

Definition at line 1400 of file read_blif.cpp.

Here is the caller graph for this function:

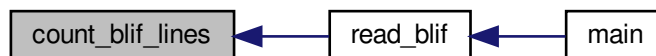


2.16.2.5 count_blif_lines()

```
int count_blif_lines (  
    FILE * file )
```

Definition at line 1613 of file read_blif.cpp.

Here is the caller graph for this function:



2.16.2.6 create_hard_block_models()

```
hard_block_models * create_hard_block_models ( )
```

Definition at line 1599 of file read_blif.cpp.

Here is the caller graph for this function:



2.16.2.7 create_hard_block_nodes()

```
void create_hard_block_nodes (
    hard_block_models * models,
    FILE * file,
    hashtable_t * output_nets_hash )
```

Definition at line 498 of file read_blif.cpp.

2.16.2.8 create_internal_node_and_driver()

```
void create_internal_node_and_driver (
    FILE * file,
    hashtable_t * output_nets_hash )
```

Definition at line 650 of file read_blif.cpp.

2.16.2.9 create_latch_node_and_driver()

```
void create_latch_node_and_driver (
    FILE * file,
    hashtable_t * output_nets_hash )
```

Definition at line 319 of file read_blif.cpp.

2.16.2.10 free_hard_block_model()

```
void free_hard_block_model (  
    hard_block_model * model )
```

Definition at line 1646 of file read_blif.cpp.

Here is the caller graph for this function:



2.16.2.11 free_hard_block_models()

```
void free_hard_block_models (  
    hard_block_models * models )
```

Definition at line 1631 of file read_blif.cpp.

Here is the caller graph for this function:



2.16.2.12 free_hard_block_pins()

```
void free_hard_block_pins (  
    hard_block_pins * p )
```

Definition at line 1660 of file read_blif.cpp.

Here is the caller graph for this function:



2.16.2.13 free_hard_block_ports()

```
void free_hard_block_ports (  
    hard_block_ports * p )
```

Definition at line 1674 of file read_blif.cpp.

Here is the caller graph for this function:

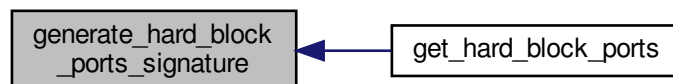


2.16.2.14 generate_hard_block_ports_signature()

```
char * generate_hard_block_ports_signature (  
    hard_block_ports * ports )
```

Definition at line 1508 of file read_blif.cpp.

Here is the caller graph for this function:



2.16.2.15 get_hard_block_model()

```
hard_block_model * get_hard_block_model (  
    char * name,  
    hard_block_ports * ports,  
    hard_block_models * models )
```

Definition at line 1587 of file read_blif.cpp.

2.16.2.16 get_hard_block_pin_number()

```
long get_hard_block_pin_number (
    char * original_name )
```

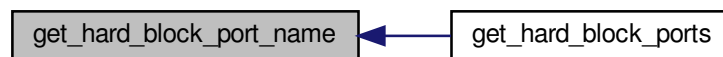
Definition at line 1550 of file read_blif.cpp.

2.16.2.17 get_hard_block_port_name()

```
char * get_hard_block_port_name (
    char * name )
```

Definition at line 1533 of file read_blif.cpp.

Here is the caller graph for this function:



2.16.2.18 get_hard_block_ports()

```
hard_block_ports * get_hard_block_ports (
    char ** pins,
    int count )
```

Definition at line 1416 of file read_blif.cpp.

2.16.2.19 hook_up_nets()

```
void hook_up_nets (
    hashtable_t * output_nets_hash )
```

Definition at line 1212 of file read_blif.cpp.

2.16.2.20 hook_up_node()

```
void hook_up_node (
    nnode_t * node,
    hashtable_t * output_nets_hash )
```

Definition at line 1235 of file read_blif.cpp.

Here is the caller graph for this function:



2.16.2.21 index_names()

```
hashtable_t * index_names (
    char ** names,
    int count )
```

Definition at line 1384 of file read_blif.cpp.

Here is the caller graph for this function:



2.16.2.22 rb_create_top_driver_nets()

```
void rb_create_top_driver_nets (
    const char * instance_name_prefix,
    hashtable_t * output_nets_hash )
```

Definition at line 1136 of file read_blif.cpp.

Here is the caller graph for this function:



2.16.2.23 rb_create_top_output_nodes()

```
void rb_create_top_output_nodes (
    FILE * file )
```

Definition at line 1073 of file read_blif.cpp.

2.16.2.24 rb_look_for_clocks()

```
void rb_look_for_clocks ( )
```

Definition at line 1114 of file read_blif.cpp.

Here is the caller graph for this function:



2.16.2.25 read_bit_map_find_unknown_gate()

```
short read_bit_map_find_unknown_gate (
    int input_count,
    nnode_t * node,
    FILE * file )
```

Definition at line 781 of file read_blif.cpp.

2.16.2.26 read_blif()

```
void read_blif (
    char * blif_file )
```

Definition at line 138 of file read_blif.cpp.

Here is the caller graph for this function:



2.16.2.27 read_hard_block_model()

```
hard_block_model * read_hard_block_model (
    char * name_subckt,
    hard_block_ports * ports,
    FILE * file )
```

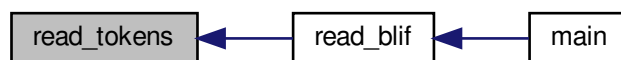
Definition at line 1256 of file read_blif.cpp.

2.16.2.28 read_tokens()

```
int read_tokens (
    char * buffer,
    hard_block_models * models,
    FILE * file,
    hashtable_t * output_nets_hash )
```

Definition at line 191 of file read_blif.cpp.

Here is the caller graph for this function:



2.16.2.29 search_clock_name()

```
char * search_clock_name (
    FILE * file )
```

Definition at line 430 of file read_blif.cpp.

Here is the caller graph for this function:



2.16.2.30 verify_hard_block_ports_against_model()

```
int verify_hard_block_ports_against_model (
    hard_block_ports * ports,
    hard_block_model * model )
```

Definition at line 1453 of file read_blif.cpp.

2.16.3 Variable Documentation

2.16.3.1 BLIF_ONE_STRING

```
const char* BLIF_ONE_STRING = "ONE_VCC_CNS"
```

Definition at line 49 of file read_blif.cpp.

2.16.3.2 BLIF_PAD_STRING

```
const char* BLIF_PAD_STRING = "ZERO_PAD_ZERO"
```

Definition at line 51 of file read_blif.cpp.

2.16.3.3 BLIF_ZERO_STRING

```
const char* BLIF_ZERO_STRING = "ZERO_GND_ZERO"
```

Definition at line 50 of file read_blif.cpp.

2.16.3.4 DEFAULT_CLOCK_NAME

```
const char* DEFAULT_CLOCK_NAME = "top^clock"
```

Definition at line 52 of file read_blif.cpp.

2.16.3.5 file_line_number

```
size_t file_line_number
```

Definition at line 47 of file read_blif.cpp.

2.17 vtr-verilog-to-routing/ODIN_II/SRC/lib_blif/read_blif.h File Reference

Functions

- void [read_blif](#) (char *blif_file)

2.17.1 Function Documentation

2.17.1.1 read_blif()

```
void read_blif (  
    char * blif_file )
```

Definition at line 138 of file read_blif.cpp.

Here is the caller graph for this function:



2.18 vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/adders.cpp File Reference

```
#include <stdlib.h>  
#include <assert.h>  
#include <stdio.h>  
#include <string.h>  
#include "types.h"  
#include "node_creation_library.h"  
#include "adders.h"  
#include "netlist_utils.h"  
#include "partial_map.h"  
#include "read_xml_arch_file.h"  
#include "globals.h"  
#include "subtractions.h"  
#include "vtr_memory.h"  
#include "vtr_list.h"
```

Functions

- void [record_add_distribution](#) (nnode_t *node)
- void [init_split_adder](#) (nnode_t *node, nnode_t *ptr, int a, int sizea, int b, int sizeb, int cin, int cout, int index, int flag, netlist_t *netlist)
- void [init_add_distribution](#) ()
- void [report_add_distribution](#) ()
- void [find_hard_adders](#) ()

- void `declare_hard_adder` (nnode_t *node)
- void `instantiate_hard_adder` (nnode_t *node, short mark, netlist_t *)
- void `add_the_blackbox_for_adds` (FILE *out)
- void `define_add_function` (nnode_t *node, short, FILE *out)
- void `split_adder` (nnode_t *nodeo, int a, int b, int sizea, int sizeb, int cin, int cout, int `count`, netlist_t *netlist)
- void `iterate_adders` (netlist_t *netlist)
- void `clean_adders` ()
- void `reduce_operations` (netlist_t *, operation_list op)
- void `traverse_list` (operation_list oper, t_linked_vptr *place)
- void `match_node` (t_linked_vptr *place, operation_list oper)
- int `match_ports` (nnode_t *node, nnode_t *next_node, operation_list oper)
- void `traverse_operation_node` (ast_node_t *node, char *component[], operation_list op, int *mark)
- void `merge_nodes` (nnode_t *node, nnode_t *next_node)
- void `remove_list_node` (t_linked_vptr *pre, t_linked_vptr *next)
- void `remove_fanout_pins` (nnode_t *node)
- void `reallocate_pins` (nnode_t *node, nnode_t *next_node)
- void `free_op_nodes` (nnode_t *node)
- int `match_pins` (nnode_t *node, nnode_t *next_node)

Variables

- t_model * `hard_adders` = NULL
- t_linked_vptr * `add_list` = NULL
- t_linked_vptr * `processed_adder_list` = NULL
- t_linked_vptr * `chain_list` = NULL
- int `total` = 0
- int * `adder` = NULL
- int `min_add` = 0
- int `min_threshold_adder` = 0
- netlist_t * `the_netlist`
- int `adder_chain_count`
- int `longest_adder_chain`
- int `total_adders`
- double `geomean_addsub_length`
- double `sum_of_addsub_logs`
- int `total_addsub_chain_count`

2.18.1 Function Documentation

2.18.1.1 add_the_blackbox_for_adds()

```
void add_the_blackbox_for_adds (  
    FILE * out )
```

Definition at line 230 of file adders.cpp.

Here is the caller graph for this function:



2.18.1.2 clean_adders()

```
void clean_adders ( )
```

Definition at line 971 of file adders.cpp.

Here is the caller graph for this function:



2.18.1.3 declare_hard_adder()

```
void declare_hard_adder (  
    nnode_t * node )
```

Definition at line 152 of file adders.cpp.

Here is the caller graph for this function:

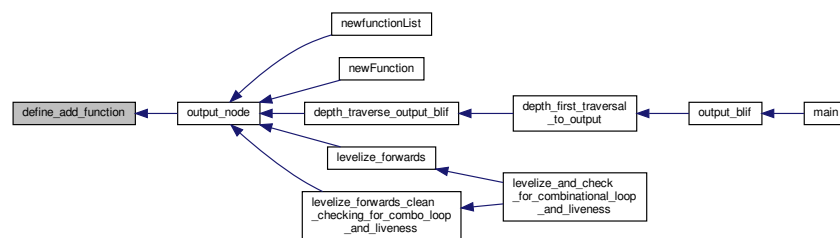


2.18.1.4 define_add_function()

```
void define_add_function (
    nnode_t * node,
    short ,
    FILE * out )
```

Definition at line 329 of file adders.cpp.

Here is the caller graph for this function:

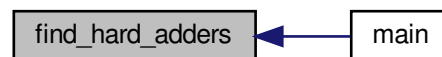


2.18.1.5 find_hard_adders()

```
void find_hard_adders ( )
```

Definition at line 126 of file adders.cpp.

Here is the caller graph for this function:



2.18.1.6 free_op_nodes()

```
void free_op_nodes (  
    nnode_t * node )
```

Definition at line 1252 of file adders.cpp.

Here is the caller graph for this function:

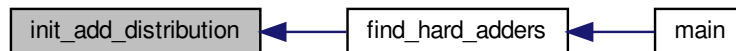


2.18.1.7 init_add_distribution()

```
void init_add_distribution ( )
```

Definition at line 63 of file adders.cpp.

Here is the caller graph for this function:



2.18.1.8 init_split_adder()

```
void init_split_adder (  
    nnode_t * node,  
    nnode_t * ptr,  
    int a,  
    int sizea,  
    int b,  
    int sizeb,  
    int cin,  
    int cout,  
    int index,  
    int flag,  
    netlist_t * netlist )
```


Definition at line 413 of file adders.cpp.

Here is the caller graph for this function:

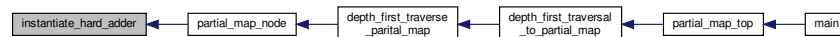


2.18.1.9 `instantiate_hard_adder()`

```
void instantiate_hard_adder (  
    nnode_t * node,  
    short mark,  
    netlist_t * )
```

Definition at line 190 of file adders.cpp.

Here is the caller graph for this function:



2.18.1.10 `iterate_adders()`

```
void iterate_adders (  
    netlist_t * netlist )
```

Definition at line 911 of file adders.cpp.

Here is the caller graph for this function:



2.18.1.11 match_node()

```
void match_node (  
    t_linked_vptr * place,  
    operation_list oper )
```

Definition at line 1032 of file adders.cpp.

Here is the caller graph for this function:

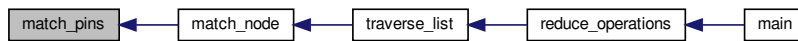


2.18.1.12 match_pins()

```
int match_pins (  
    nnode_t * node,  
    nnode_t * next_node )
```

Definition at line 1268 of file adders.cpp.

Here is the caller graph for this function:

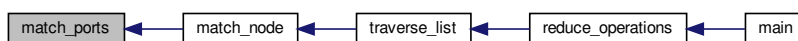


2.18.1.13 match_ports()

```
int match_ports (  
    nnode_t * node,  
    nnode_t * next_node,  
    operation_list oper )
```

Definition at line 1078 of file adders.cpp.

Here is the caller graph for this function:

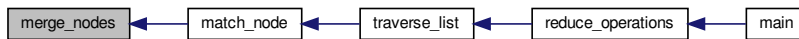


2.18.1.14 merge_nodes()

```
void merge_nodes (  
    nnode_t * node,  
    nnode_t * next_node )
```

Definition at line 1175 of file adders.cpp.

Here is the caller graph for this function:



2.18.1.15 reallocate_pins()

```
void reallocate_pins (  
    nnode_t * node,  
    nnode_t * next_node )
```

Definition at line 1225 of file adders.cpp.

Here is the caller graph for this function:



2.18.1.16 record_add_distribution()

```
void record_add_distribution (  
    nnode_t * node )
```

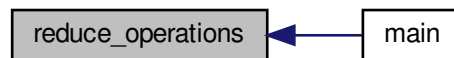
Definition at line 77 of file adders.cpp.

2.18.1.17 reduce_operations()

```
void reduce_operations (
    netlist_t * ,
    operation_list op )
```

Definition at line 984 of file adders.cpp.

Here is the caller graph for this function:

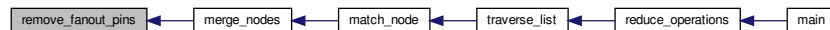


2.18.1.18 remove_fanout_pins()

```
void remove_fanout_pins (
    nnode_t * node )
```

Definition at line 1200 of file adders.cpp.

Here is the caller graph for this function:

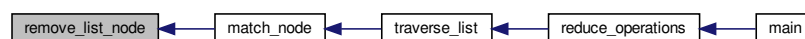


2.18.1.19 remove_list_node()

```
void remove_list_node (
    t_linked_vptr * pre,
    t_linked_vptr * next )
```

Definition at line 1187 of file adders.cpp.

Here is the caller graph for this function:

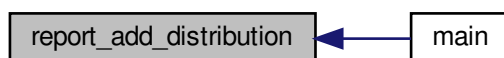


2.18.1.20 report_add_distribution()

```
void report_add_distribution ( )
```

Definition at line 98 of file adders.cpp.

Here is the caller graph for this function:

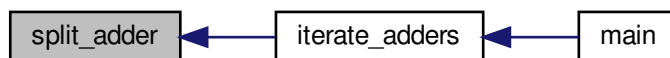


2.18.1.21 split_adder()

```
void split_adder (
    nnode_t * nodeo,
    int a,
    int b,
    int sizea,
    int sizeb,
    int cin,
    int cout,
    int count,
    netlist_t * netlist )
```

Definition at line 707 of file adders.cpp.

Here is the caller graph for this function:

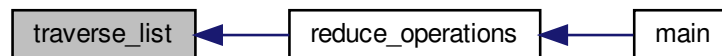


2.18.1.22 traverse_list()

```
void traverse_list (
    operation_list oper,
    t_linked_vptr * place )
```

Definition at line 1020 of file adders.cpp.

Here is the caller graph for this function:

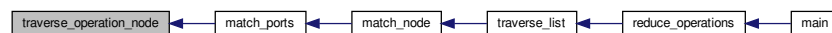


2.18.1.23 traverse_operation_node()

```
void traverse_operation_node (
    ast_node_t * node,
    char * component[],
    operation_list op,
    int * mark )
```

Definition at line 1141 of file adders.cpp.

Here is the caller graph for this function:



2.18.2 Variable Documentation

2.18.2.1 add_list

```
t_linked_vptr* add_list = NULL
```

Definition at line 45 of file adders.cpp.

2.18.2.2 adder

```
int* adder = NULL
```

Definition at line 49 of file adders.cpp.

2.18.2.3 adder_chain_count

```
int adder_chain_count
```

Definition at line 180 of file netlist_cleanup.cpp.

2.18.2.4 chain_list

```
t_linked_vptr* chain_list = NULL
```

Definition at line 47 of file adders.cpp.

2.18.2.5 geomean_addsub_length

```
double geomean_addsub_length
```

Definition at line 188 of file netlist_cleanup.cpp.

2.18.2.6 hard_adders

```
t_model* hard_adders = NULL
```

Definition at line 44 of file adders.cpp.

2.18.2.7 longest_adder_chain

```
int longest_adder_chain
```

Definition at line 181 of file netlist_cleanup.cpp.

2.18.2.8 min_add

```
int min_add = 0
```

Definition at line 50 of file adders.cpp.

2.18.2.9 min_threshold_adder

```
int min_threshold_adder = 0
```

Definition at line 51 of file adders.cpp.

2.18.2.10 processed_adder_list

```
t_linked_vptr* processed_adder_list = NULL
```

Definition at line 46 of file adders.cpp.

2.18.2.11 sum_of_addsub_logs

```
double sum_of_addsub_logs
```

Definition at line 189 of file netlist_cleanup.cpp.

2.18.2.12 the_netlist

```
netlist_t* the_netlist
```

Definition at line 53 of file adders.cpp.

2.18.2.13 total

```
int total = 0
```

Definition at line 48 of file adders.cpp.

2.18.2.14 total_adders

```
int total_adders
```

Definition at line 182 of file netlist_cleanup.cpp.

2.18.2.15 total_addsub_chain_count

```
int total_addsub_chain_count
```

Definition at line 190 of file netlist_cleanup.cpp.

2.19 vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/adders.h File Reference

```
#include "read_xml_arch_file.h"
```

Data Structures

- struct [s_adder](#)
- struct [adder_signals](#)

Typedefs

- typedef struct [s_adder](#) [t_adder](#)

Functions

- void [init_add_distribution](#) ()
- void [report_add_distribution](#) ()
- void [declare_hard_adder](#) (nnode_t *node)
- void [instantiate_hard_adder](#) (nnode_t *node, short mark, netlist_t *netlist)
- void [instantiate_simple_soft_adder](#) (nnode_t *node, short mark, netlist_t *netlist)
- void [find_hard_adders](#) ()
- void [add_the_blackbox_for_adds](#) (FILE *out)
- void [define_add_function](#) (nnode_t *node, short type, FILE *out)
- void [split_adder](#) (nnode_t *node, int a, int b, int sizea, int sizeb, int cin, int cout, int [count](#), netlist_t *netlist)
- void [iterate_adders](#) (netlist_t *netlist)
- void [clean_adders](#) ()
- void [reduce_operations](#) (netlist_t *netlist, operation_list op)
- void [traverse_list](#) (operation_list oper, vtr::t_linked_vptr *place)
- void [match_node](#) (vtr::t_linked_vptr *place, operation_list oper)
- int [match_ports](#) (nnode_t *node, nnode_t *next_node, operation_list oper)
- void [traverse_operation_node](#) (ast_node_t *node, char *component[], operation_list op, int *mark)
- void [merge_nodes](#) (nnode_t *node, nnode_t *next_node)
- void [remove_list_node](#) (vtr::t_linked_vptr *node, vtr::t_linked_vptr *place)
- void [remove_fanout_pins](#) (nnode_t *node)
- void [reallocate_pins](#) (nnode_t *node, nnode_t *next_node)
- void [free_op_nodes](#) (nnode_t *node)
- int [match_pins](#) (nnode_t *node, nnode_t *next_node)

Variables

- `t_model` * [hard_adders](#)
- `vtr::t_linked_vptr` * [add_list](#)
- `vtr::t_linked_vptr` * [chain_list](#)
- `vtr::t_linked_vptr` * [processed_adder_list](#)
- `int` [total](#)
- `int` [min_add](#)
- `int` [min_threshold_adder](#)

2.19.1 Typedef Documentation

2.19.1.1 `t_adder`

```
typedef struct s\_adder t\_adder
```

2.19.2 Function Documentation

2.19.2.1 `add_the_blackbox_for_adds()`

```
void add\_the\_blackbox\_for\_adds (  
    FILE * out )
```

Definition at line 230 of file `adders.cpp`.

Here is the caller graph for this function:



2.19.2.2 `clean_adders()`

```
void clean_adders ( )
```

Definition at line 971 of file `adders.cpp`.

Here is the caller graph for this function:



2.19.2.3 `declare_hard_adder()`

```
void declare_hard_adder (
    nnode_t * node )
```

Definition at line 152 of file `adders.cpp`.

Here is the caller graph for this function:

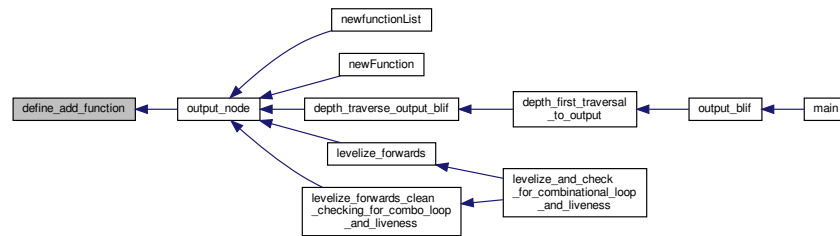


2.19.2.4 `define_add_function()`

```
void define_add_function (
    nnode_t * node,
    short type,
    FILE * out )
```

Definition at line 329 of file `adders.cpp`.

Here is the caller graph for this function:



2.19.2.5 find_hard_adders()

```
void find_hard_adders ( )
```

Definition at line 126 of file `adders.cpp`.

Here is the caller graph for this function:



2.19.2.6 free_op_nodes()

```
void free_op_nodes (
    nnode_t * node )
```

Definition at line 1252 of file `adders.cpp`.

Here is the caller graph for this function:

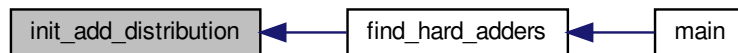


2.19.2.7 init_add_distribution()

```
void init_add_distribution ( )
```

Definition at line 63 of file adders.cpp.

Here is the caller graph for this function:

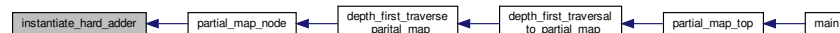


2.19.2.8 instantiate_hard_adder()

```
void instantiate_hard_adder (
    nnode_t * node,
    short mark,
    netlist_t * netlist )
```

Definition at line 190 of file adders.cpp.

Here is the caller graph for this function:



2.19.2.9 instantiate_simple_soft_adder()

```
void instantiate_simple_soft_adder (
    nnode_t * node,
    short mark,
    netlist_t * netlist )
```

2.19.2.10 iterate_adders()

```
void iterate_adders (
    netlist_t * netlist )
```

Definition at line 911 of file adders.cpp.

Here is the caller graph for this function:



2.19.2.11 match_node()

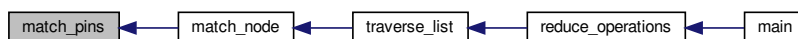
```
void match_node (
    vtr::t_linked_vptr * place,
    operation_list oper )
```

2.19.2.12 match_pins()

```
int match_pins (
    nnode_t * node,
    nnode_t * next_node )
```

Definition at line 1268 of file adders.cpp.

Here is the caller graph for this function:

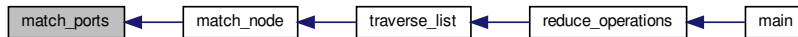


2.19.2.13 match_ports()

```
int match_ports (
    nnode_t * node,
    nnode_t * next_node,
    operation_list oper )
```

Definition at line 1078 of file adders.cpp.

Here is the caller graph for this function:

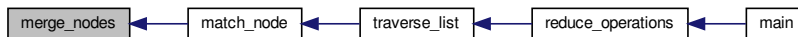


2.19.2.14 merge_nodes()

```
void merge_nodes (
    nnode_t * node,
    nnode_t * next_node )
```

Definition at line 1175 of file adders.cpp.

Here is the caller graph for this function:



2.19.2.15 reallocate_pins()

```
void reallocate_pins (
    nnode_t * node,
    nnode_t * next_node )
```

Definition at line 1225 of file adders.cpp.

Here is the caller graph for this function:

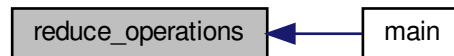


2.19.2.16 reduce_operations()

```
void reduce_operations (
    netlist_t * netlist,
    operation_list op )
```

Definition at line 984 of file adders.cpp.

Here is the caller graph for this function:

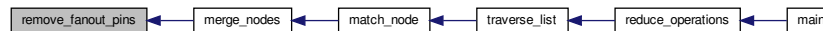


2.19.2.17 remove_fanout_pins()

```
void remove_fanout_pins (
    nnode_t * node )
```

Definition at line 1200 of file adders.cpp.

Here is the caller graph for this function:



2.19.2.18 remove_list_node()

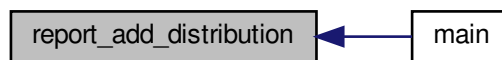
```
void remove_list_node (
    vtr::t_linked_vptr * node,
    vtr::t_linked_vptr * place )
```


2.19.2.19 report_add_distribution()

```
void report_add_distribution ( )
```

Definition at line 98 of file adders.cpp.

Here is the caller graph for this function:

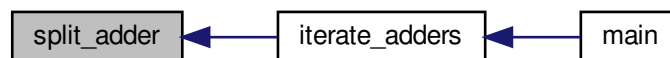


2.19.2.20 split_adder()

```
void split_adder (
    nnode_t * node,
    int a,
    int b,
    int sizea,
    int sizeb,
    int cin,
    int cout,
    int count,
    netlist_t * netlist )
```

Definition at line 707 of file adders.cpp.

Here is the caller graph for this function:



2.19.2.21 traverse_list()

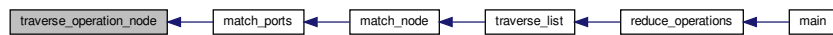
```
void traverse_list (  
    operation_list oper,  
    vtr::t_linked_vptr * place )
```

2.19.2.22 traverse_operation_node()

```
void traverse_operation_node (  
    ast_node_t * node,  
    char * component[],  
    operation_list op,  
    int * mark )
```

Definition at line 1141 of file adders.cpp.

Here is the caller graph for this function:



2.19.3 Variable Documentation

2.19.3.1 add_list

```
vtr::t_linked_vptr* add_list
```

Definition at line 45 of file adders.cpp.

2.19.3.2 chain_list

```
vtr::t_linked_vptr* chain_list
```

Definition at line 47 of file adders.cpp.

2.19.3.3 hard_adders

```
t_model* hard_adders
```

Definition at line 44 of file adders.cpp.

2.19.3.4 min_add

```
int min_add
```

Definition at line 50 of file adders.cpp.

2.19.3.5 min_threshold_adder

```
int min_threshold_adder
```

Definition at line 51 of file adders.cpp.

2.19.3.6 processed_adder_list

```
vtr::t_linked_vptr* processed_adder_list
```

Definition at line 46 of file adders.cpp.

2.19.3.7 total

```
int total
```

Definition at line 48 of file adders.cpp.

2.20 vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/hard_blocks.cpp File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "types.h"
#include "globals.h"
#include "hard_blocks.h"
#include "memories.h"
```

Functions

- void [cache_hard_block_names](#) ()
- t_model_ports * [get_model_port](#) (t_model_ports *ports, const char *name)
- void [register_hard_blocks](#) ()
- void [deregister_hard_blocks](#) ()
- t_model * [find_hard_block](#) (const char *name)
- void [define_hard_block](#) (nnode_t *node, short, FILE *out)
- void [output_hard_blocks](#) (FILE *out)
- void [instantiate_hard_block](#) (nnode_t *node, short mark, netlist_t *)
- int [hard_block_port_size](#) (t_model *hb, char *pname)
- enum PORTS [hard_block_port_direction](#) (t_model *hb, char *pname)

Variables

- [STRING_CACHE](#) * [hard_block_names](#) = NULL

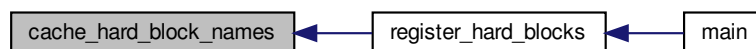
2.20.1 Function Documentation

2.20.1.1 [cache_hard_block_names](#)()

```
void cache_hard_block_names ( )
```

Definition at line 49 of file [hard_blocks.cpp](#).

Here is the caller graph for this function:

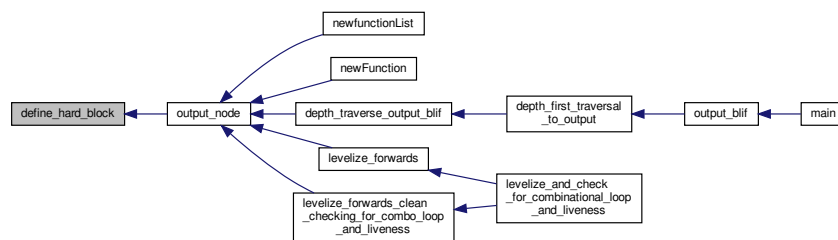


2.20.1.2 define_hard_block()

```
void define_hard_block (
    nnode_t * node,
    short ,
    FILE * out )
```

Definition at line 132 of file hard_blocks.cpp.

Here is the caller graph for this function:



2.20.1.3 deregister_hard_blocks()

```
void deregister_hard_blocks ( )
```

Definition at line 112 of file hard_blocks.cpp.

Here is the caller graph for this function:

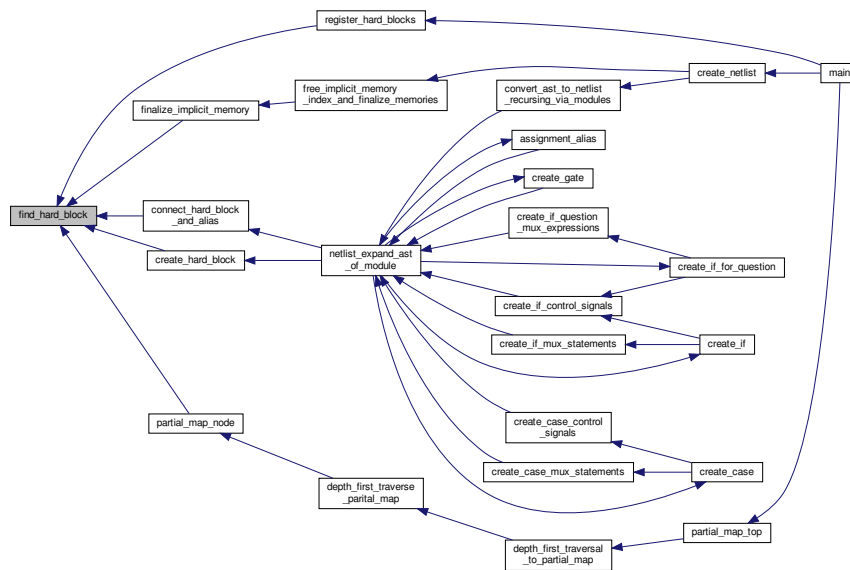


2.20.1.4 find_hard_block()

```
t_model* find_hard_block (
    const char * name )
```

Definition at line 118 of file hard_blocks.cpp.

Here is the caller graph for this function:

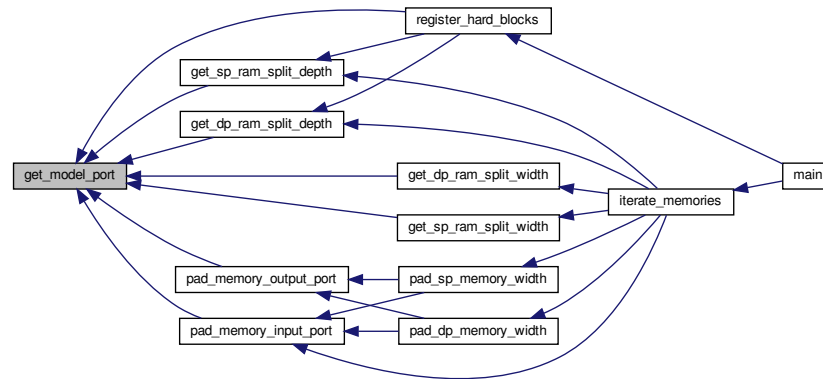


2.20.1.5 get_model_port()

```
t_model_ports* get_model_port (
    t_model_ports * ports,
    const char * name )
```

Definition at line 41 of file hard_blocks.cpp.

Here is the caller graph for this function:

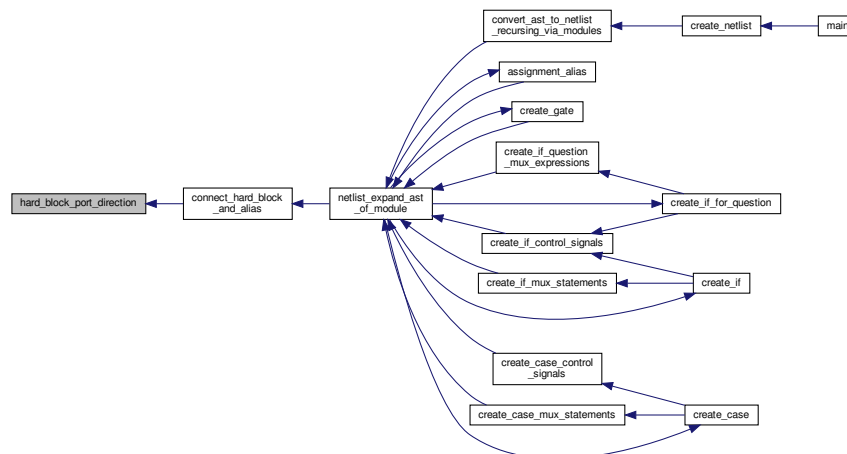


2.20.1.6 `hard_block_port_direction()`

```
enum PORTS hard_block_port_direction (
    t_model * hb,
    char * pname )
```

Definition at line 343 of file `hard_blocks.cpp`.

Here is the caller graph for this function:

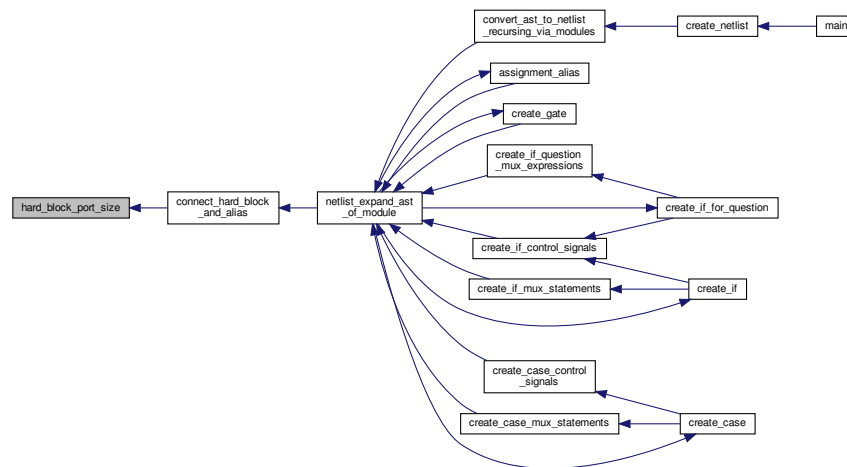


2.20.1.7 hard_block_port_size()

```
int hard_block_port_size (
    t_model * hb,
    char * pname )
```

Definition at line 308 of file hard_blocks.cpp.

Here is the caller graph for this function:

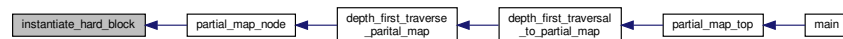


2.20.1.8 instantiate_hard_block()

```
void instantiate_hard_block (
    nnode_t * node,
    short mark,
    netlist_t * )
```

Definition at line 284 of file hard_blocks.cpp.

Here is the caller graph for this function:



2.20.1.9 output_hard_blocks()

```
void output_hard_blocks (
    FILE * out )
```

Definition at line 215 of file hard_blocks.cpp.

Here is the caller graph for this function:

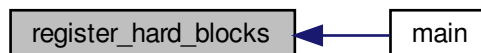


2.20.1.10 register_hard_blocks()

```
void register_hard_blocks ( )
```

Definition at line 63 of file hard_blocks.cpp.

Here is the caller graph for this function:



2.20.2 Variable Documentation

2.20.2.1 hard_block_names

```
STRING_CACHE* hard_block_names = NULL
```

Definition at line 37 of file hard_blocks.cpp.

2.21 vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/hard_blocks.h File Reference

```
#include "types.h"
```

Functions

- void [register_hard_blocks](#) ()
- void [deregister_hard_blocks](#) ()
- t_model * [find_hard_block](#) (const char *name)
- void [define_hard_block](#) (nnode_t *node, short type, FILE *out)
- void [output_hard_blocks](#) (FILE *out)
- int [hard_block_port_size](#) (t_model *hb, char *pname)
- enum PORTS [hard_block_port_direction](#) (t_model *hb, char *pname)
- void [instantiate_hard_block](#) (nnode_t *node, short mark, netlist_t *netlist)
- t_model_ports * [get_model_port](#) (t_model_ports *ports, const char *name)

Variables

- [STRING_CACHE](#) * [hard_block_names](#)

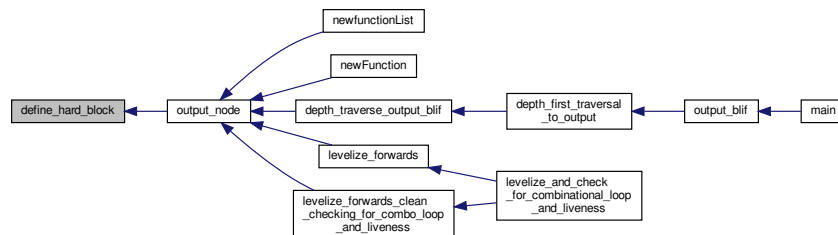
2.21.1 Function Documentation

2.21.1.1 [define_hard_block\(\)](#)

```
void define_hard_block (
    nnode_t * node,
    short type,
    FILE * out )
```

Definition at line 132 of file [hard_blocks.cpp](#).

Here is the caller graph for this function:



2.21.1.2 deregister_hard_blocks()

```
void deregister_hard_blocks ( )
```

Definition at line 112 of file hard_blocks.cpp.

Here is the caller graph for this function:

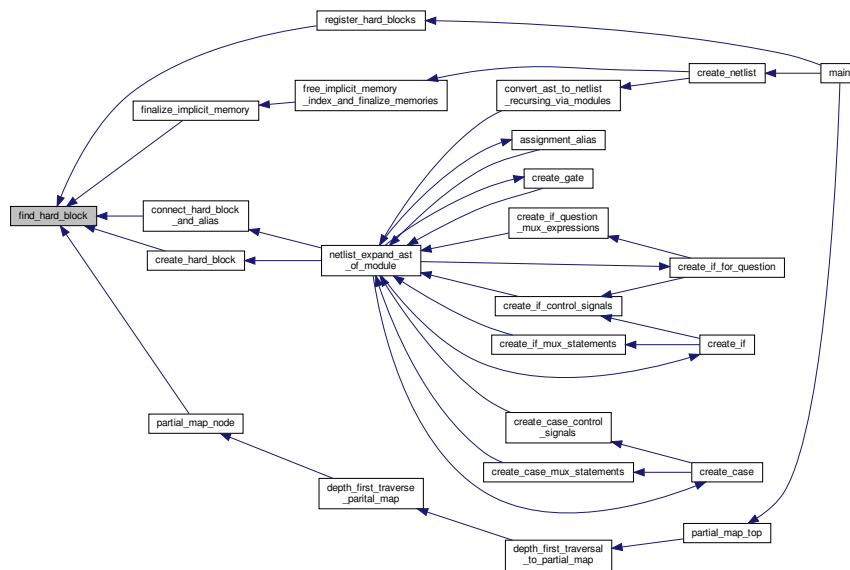


2.21.1.3 find_hard_block()

```
t_model* find_hard_block (
    const char * name )
```

Definition at line 118 of file hard_blocks.cpp.

Here is the caller graph for this function:

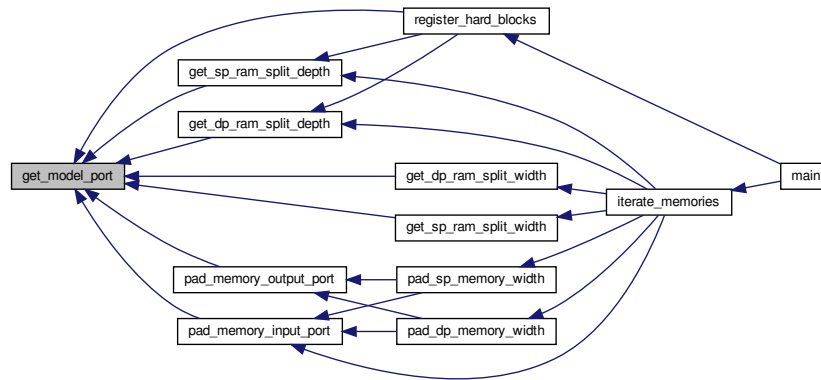


2.21.1.4 get_model_port()

```
t_model_ports* get_model_port (
    t_model_ports * ports,
    const char * name )
```

Definition at line 41 of file hard_blocks.cpp.

Here is the caller graph for this function:

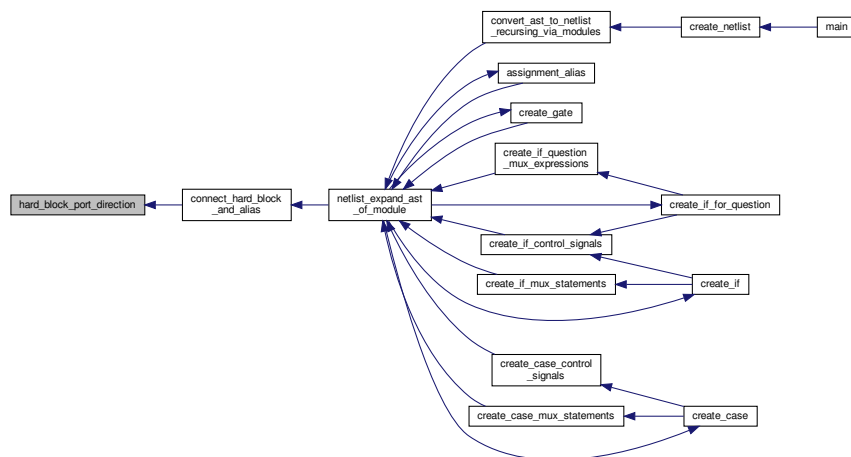


2.21.1.5 hard_block_port_direction()

```
enum PORTS hard_block_port_direction (
    t_model * hb,
    char * pname )
```

Definition at line 343 of file hard_blocks.cpp.

Here is the caller graph for this function:

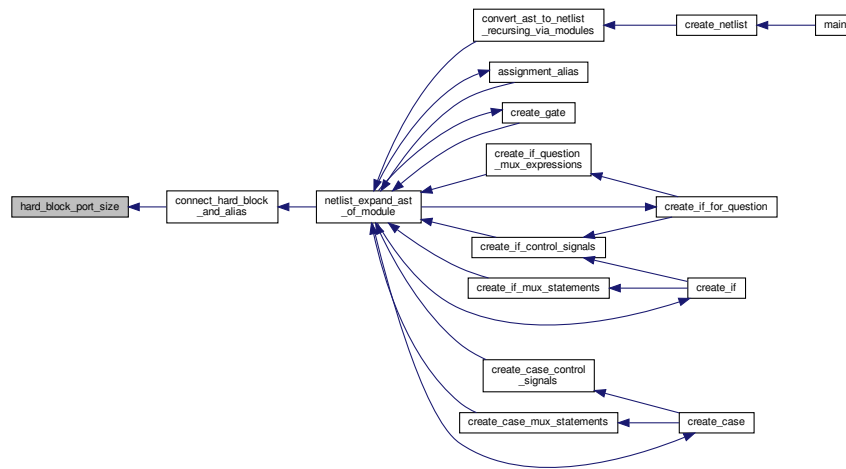


2.21.1.6 hard_block_port_size()

```
int hard_block_port_size (
    t_model * hb,
    char * pname )
```

Definition at line 308 of file hard_blocks.cpp.

Here is the caller graph for this function:

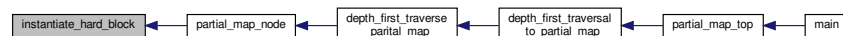


2.21.1.7 instantiate_hard_block()

```
void instantiate_hard_block (
    nnode_t * node,
    short mark,
    netlist_t * netlist )
```

Definition at line 284 of file hard_blocks.cpp.

Here is the caller graph for this function:



2.21.1.8 output_hard_blocks()

```
void output_hard_blocks (  
    FILE * out )
```

Definition at line 215 of file hard_blocks.cpp.

Here is the caller graph for this function:



2.21.1.9 register_hard_blocks()

```
void register_hard_blocks ( )
```

Definition at line 63 of file hard_blocks.cpp.

Here is the caller graph for this function:



2.21.2 Variable Documentation

2.21.2.1 hard_block_names

```
STRING_CACHE* hard_block_names
```

Definition at line 37 of file hard_blocks.cpp.

2.22 vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/implicit_memory.cpp File Reference

```
#include "types.h"
#include "hashtable.h"
#include "implicit_memory.h"
#include "node_creation_library.h"
#include "odin_util.h"
#include "vtr_util.h"
#include "vtr_memory.h"
```

Functions

- void [finalize_implicit_memory](#) ([implicit_memory](#) *memory)
- void [add_dummy_output_port_to_implicit_memory](#) ([implicit_memory](#) *memory, int size, const char *port_name)
- void [add_dummy_input_port_to_implicit_memory](#) ([implicit_memory](#) *memory, int size, const char *port_name)
- void [collapse_implicit_memory_to_single_port_ram](#) ([implicit_memory](#) *memory)
- [implicit_memory](#) * [lookup_implicit_memory](#) (char *instance_name_prefix, char *identifier)
- [implicit_memory](#) * [lookup_implicit_memory_reference_ast](#) (char *instance_name_prefix, ast_node_t *node)
- char [is_valid_implicit_memory_reference_ast](#) (char *instance_name_prefix, ast_node_t *node)
- [implicit_memory](#) * [create_implicit_memory_block](#) (int data_width, long long words, char *name, char *instance_name_prefix)
- void [add_input_port_to_implicit_memory](#) ([implicit_memory](#) *memory, signal_list_t *signals, const char *port_name)
- void [add_output_port_to_implicit_memory](#) ([implicit_memory](#) *memory, signal_list_t *signals, const char *port_name)
- [implicit_memory](#) * [lookup_implicit_memory_input](#) (char *name)
- void [register_implicit_memory_input](#) (char *name, [implicit_memory](#) *memory)
- void [init_implicit_memory_index](#) ()
- void [free_implicit_memory_index_and_finalize_memories](#) ()

Variables

- [hashtable_t](#) * [implicit_memories](#)
- [hashtable_t](#) * [implicit_memory_inputs](#)

2.22.1 Function Documentation

2.22.1.1 add_dummy_input_port_to_implicit_memory()

```
void add_dummy_input_port_to_implicit_memory (
    implicit_memory * memory,
    int size,
    const char * port_name )
```

Definition at line 194 of file implicit_memory.cpp.

Here is the caller graph for this function:



2.22.1.2 add_dummy_output_port_to_implicit_memory()

```
void add_dummy_output_port_to_implicit_memory (
    implicit_memory * memory,
    int size,
    const char * port_name )
```

Definition at line 210 of file implicit_memory.cpp.

Here is the caller graph for this function:

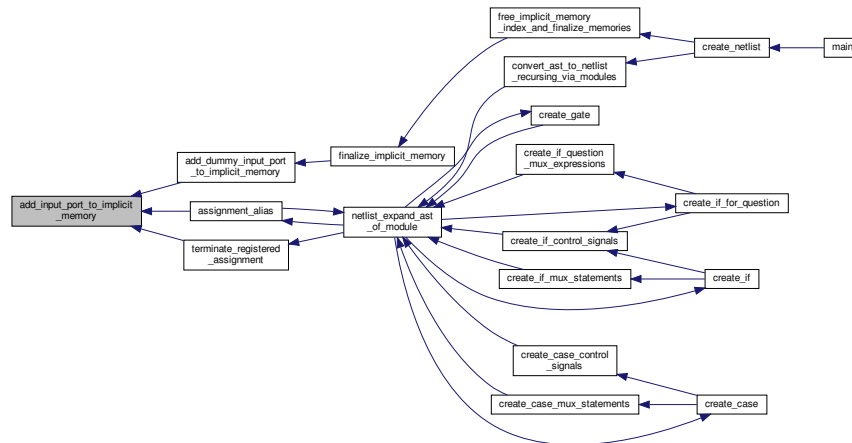


2.22.1.3 add_input_port_to_implicit_memory()

```
void add_input_port_to_implicit_memory (
    implicit_memory * memory,
    signal_list_t * signals,
    const char * port_name )
```

Definition at line 117 of file implicit_memory.cpp.

Here is the caller graph for this function:



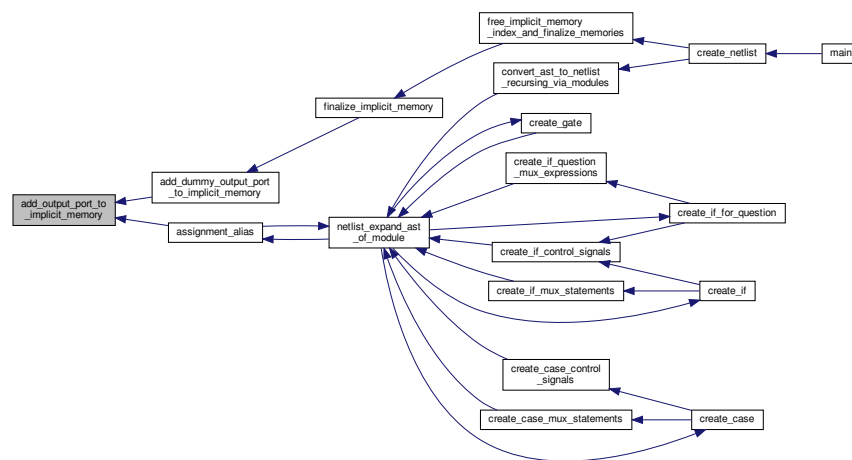
2.2.2.1.4 add_output_port_to_implicit_memory()

```

void add_output_port_to_implicit_memory (
    implicit_memory * memory,
    signal_list_t * signals,
    const char * port_name )
  
```

Definition at line 127 of file implicit_memory.cpp.

Here is the caller graph for this function:



2.22.1.5 collapse_implicit_memory_to_single_port_ram()

```
void collapse_implicit_memory_to_single_port_ram (
    implicit_memory * memory )
```

Definition at line 330 of file implicit_memory.cpp.

Here is the caller graph for this function:

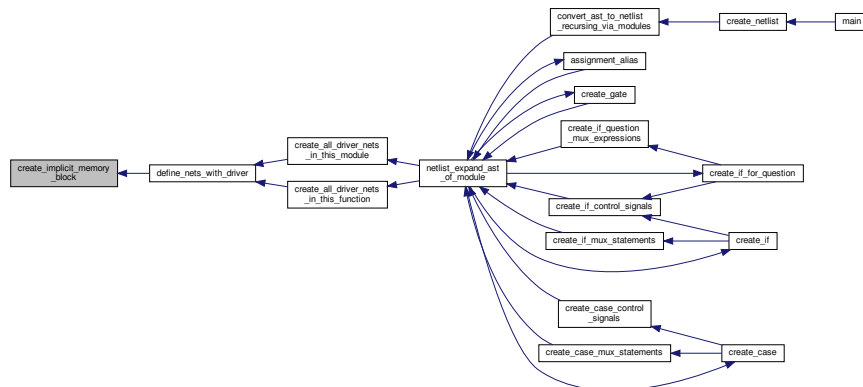


2.22.1.6 create_implicit_memory_block()

```
implicit_memory* create_implicit_memory_block (
    int data_width,
    long long words,
    char * name,
    char * instance_name_prefix )
```

Definition at line 79 of file implicit_memory.cpp.

Here is the caller graph for this function:

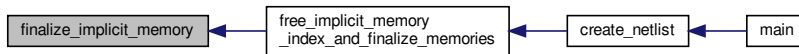


2.22.1.7 finalize_implicit_memory()

```
void finalize_implicit_memory (
    implicit_memory * memory )
```

Definition at line 233 of file implicit_memory.cpp.

Here is the caller graph for this function:

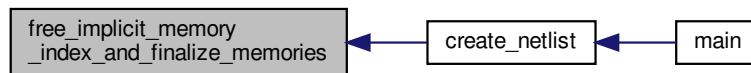


2.22.1.8 free_implicit_memory_index_and_finalize_memories()

```
void free_implicit_memory_index_and_finalize_memories ( )
```

Definition at line 168 of file implicit_memory.cpp.

Here is the caller graph for this function:

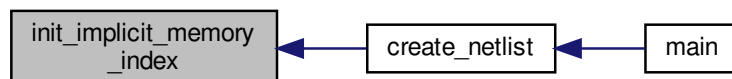


2.22.1.9 init_implicit_memory_index()

```
void init_implicit_memory_index ( )
```

Definition at line 157 of file implicit_memory.cpp.

Here is the caller graph for this function:

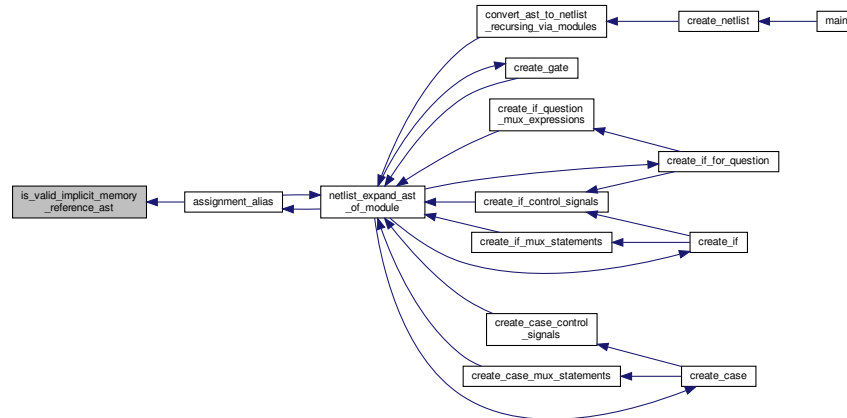


2.22.1.10 is_valid_implicit_memory_reference_ast()

```
char is_valid_implicit_memory_reference_ast (
    char * instance_name_prefix,
    ast_node_t * node )
```

Definition at line 67 of file implicit_memory.cpp.

Here is the caller graph for this function:

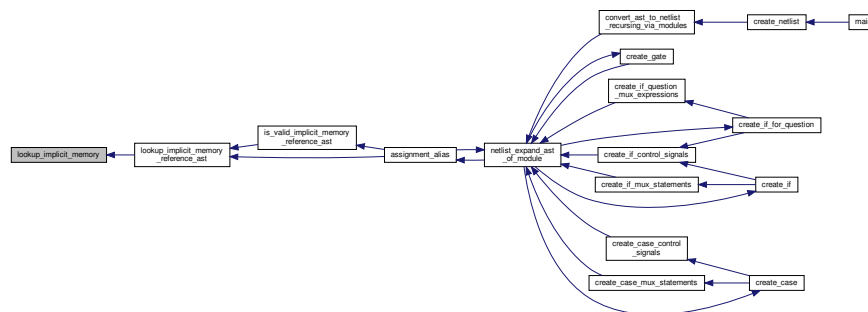


2.22.1.11 lookup_implicit_memory()

```
implicit_memory * lookup_implicit_memory (
    char * instance_name_prefix,
    char * identifier )
```

Definition at line 45 of file implicit_memory.cpp.

Here is the caller graph for this function:

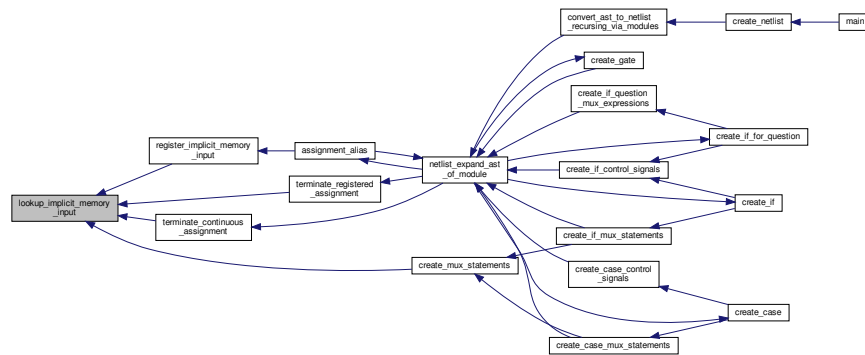


2.22.1.12 lookup_implicit_memory_input()

```
implicit_memory* lookup_implicit_memory_input (
    char * name )
```

Definition at line 137 of file implicit_memory.cpp.

Here is the caller graph for this function:

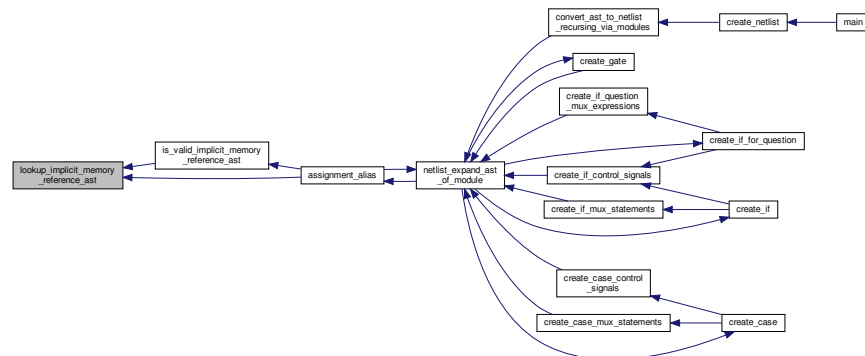


2.22.1.13 lookup_implicit_memory_reference_ast()

```
implicit_memory* lookup_implicit_memory_reference_ast (
    char * instance_name_prefix,
    ast_node_t * node )
```

Definition at line 54 of file implicit_memory.cpp.

Here is the caller graph for this function:

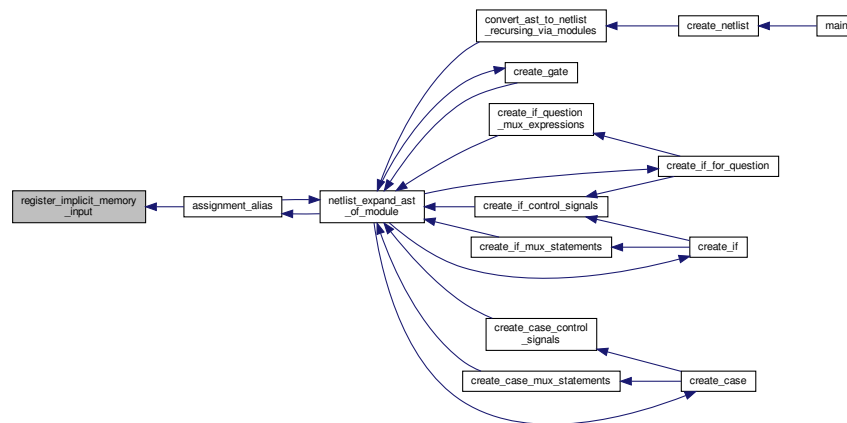


2.22.1.14 register_implicit_memory_input()

```
void register_implicit_memory_input (
    char * name,
    implicit_memory * memory )
```

Definition at line 146 of file implicit_memory.cpp.

Here is the caller graph for this function:



2.22.2 Variable Documentation

2.22.2.1 implicit_memories

```
hashtable_t* implicit_memories
```

Definition at line 32 of file implicit_memory.cpp.

2.22.2.2 implicit_memory_inputs

```
hashtable_t* implicit_memory_inputs
```

Definition at line 34 of file implicit_memory.cpp.

2.23 vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/implicit_memory.h File Reference

Data Structures

- struct `implicit_memory`

Functions

- void `add_input_port_to_implicit_memory` (`implicit_memory` *memory, `signal_list_t` *signals, const char *port_name)
- void `add_output_port_to_implicit_memory` (`implicit_memory` *memory, `signal_list_t` *signals, const char *port_name)
- `implicit_memory` * `lookup_implicit_memory_reference_ast` (char *instance_name_prefix, `ast_node_t` *node)
- char `is_valid_implicit_memory_reference_ast` (char *instance_name_prefix, `ast_node_t` *node)
- `implicit_memory` * `create_implicit_memory_block` (int data_width, long long words, char *name, char *instance_name_prefix)
- `implicit_memory` * `lookup_implicit_memory_input` (char *name)
- void `register_implicit_memory_input` (char *name, `implicit_memory` *memory)
- void `init_implicit_memory_index` ()
- void `free_implicit_memory_index_and_finalize_memories` ()

Variables

- `hashtable_t` * `implicit_memories`
- `hashtable_t` * `implicit_memory_inputs`

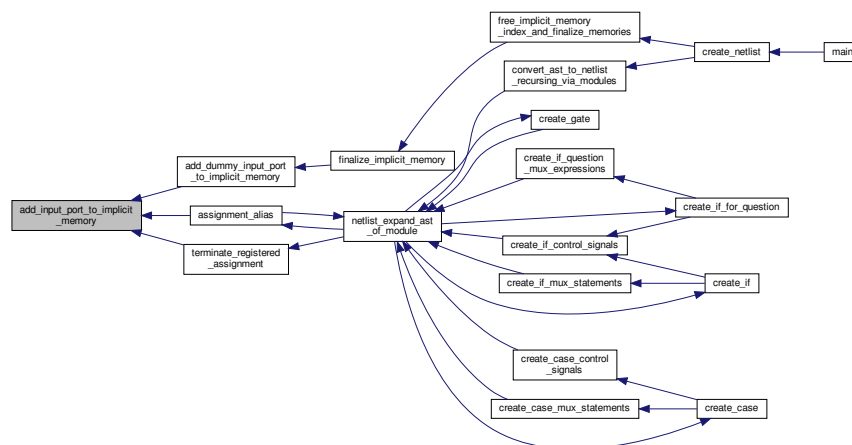
2.23.1 Function Documentation

2.23.1.1 `add_input_port_to_implicit_memory()`

```
void add_input_port_to_implicit_memory (
    implicit_memory * memory,
    signal_list_t * signals,
    const char * port_name )
```

Definition at line 117 of file `implicit_memory.cpp`.

Here is the caller graph for this function:

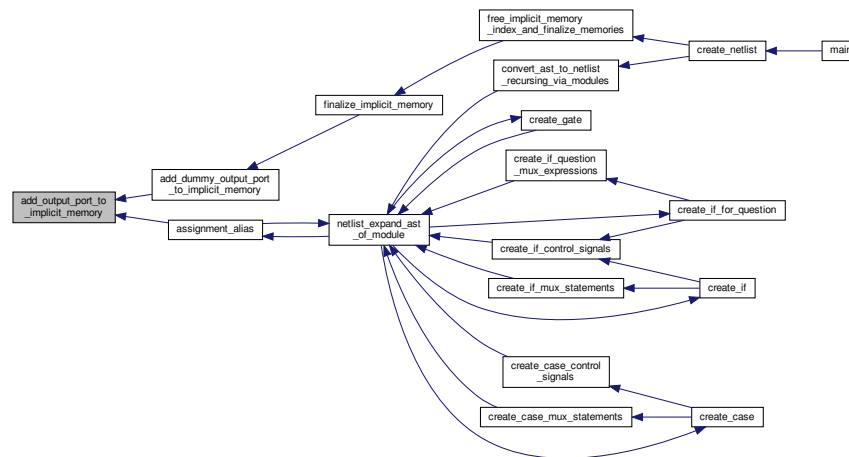


2.23.1.2 add_output_port_to_implicit_memory()

```
void add_output_port_to_implicit_memory (
    implicit_memory * memory,
    signal_list_t * signals,
    const char * port_name )
```

Definition at line 127 of file implicit_memory.cpp.

Here is the caller graph for this function:

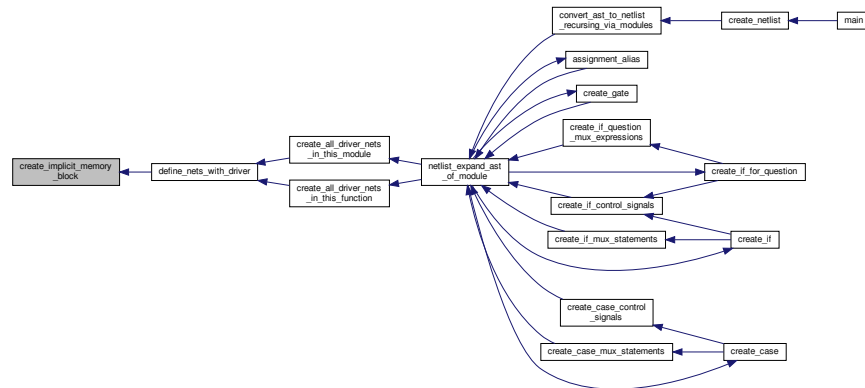


2.23.1.3 create_implicit_memory_block()

```
implicit_memory* create_implicit_memory_block (
    int data_width,
    long long words,
    char * name,
    char * instance_name_prefix )
```

Definition at line 79 of file implicit_memory.cpp.

Here is the caller graph for this function:

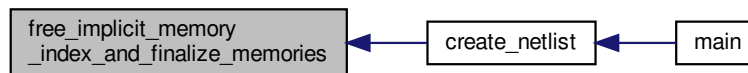


2.23.1.4 `free_implicit_memory_index_and_finalize_memories()`

```
void free_implicit_memory_index_and_finalize_memories ( )
```

Definition at line 168 of file `implicit_memory.cpp`.

Here is the caller graph for this function:

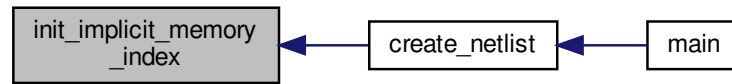


2.23.1.5 `init_implicit_memory_index()`

```
void init_implicit_memory_index ( )
```

Definition at line 157 of file `implicit_memory.cpp`.

Here is the caller graph for this function:



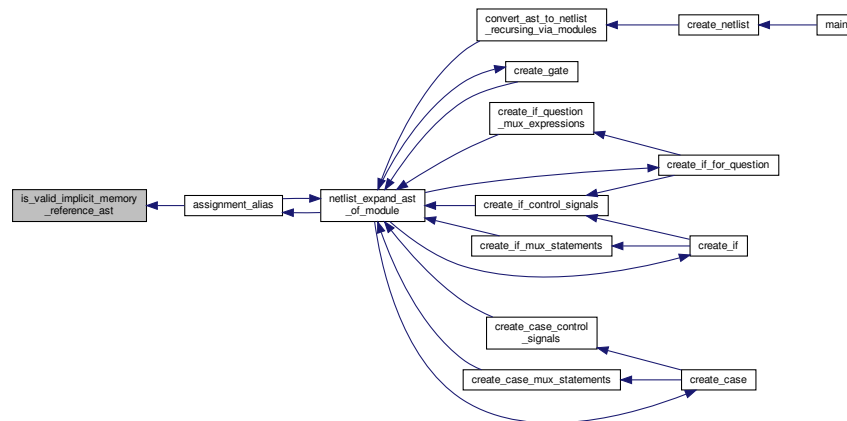
2.23.1.6 is_valid_implicit_memory_reference_ast()

```

char is_valid_implicit_memory_reference_ast (
    char * instance_name_prefix,
    ast_node_t * node )
  
```

Definition at line 67 of file `implicit_memory.cpp`.

Here is the caller graph for this function:



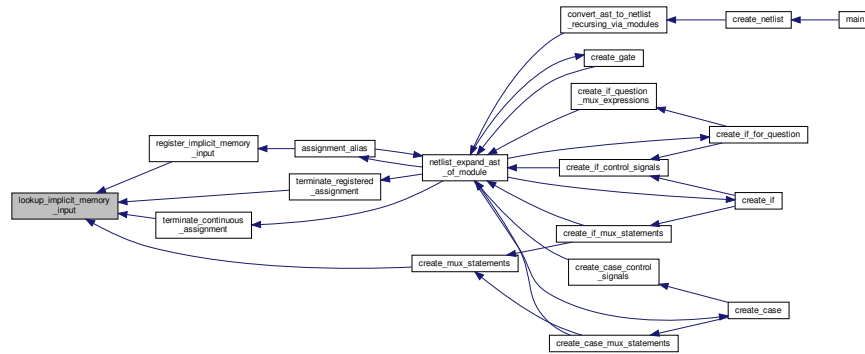
2.23.1.7 lookup_implicit_memory_input()

```

implicit_memory* lookup_implicit_memory_input (
    char * name )
  
```

Definition at line 137 of file `implicit_memory.cpp`.

Here is the caller graph for this function:



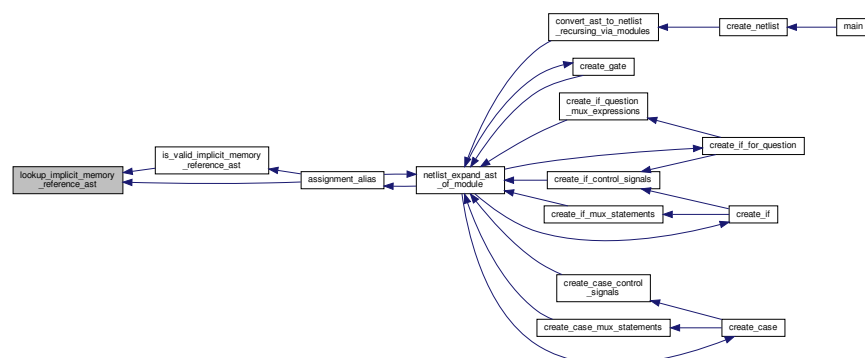
2.23.1.8 lookup_implicit_memory_reference_ast()

```

implicit_memory* lookup_implicit_memory_reference_ast (
    char * instance_name_prefix,
    ast_node_t * node )
  
```

Definition at line 54 of file implicit_memory.cpp.

Here is the caller graph for this function:

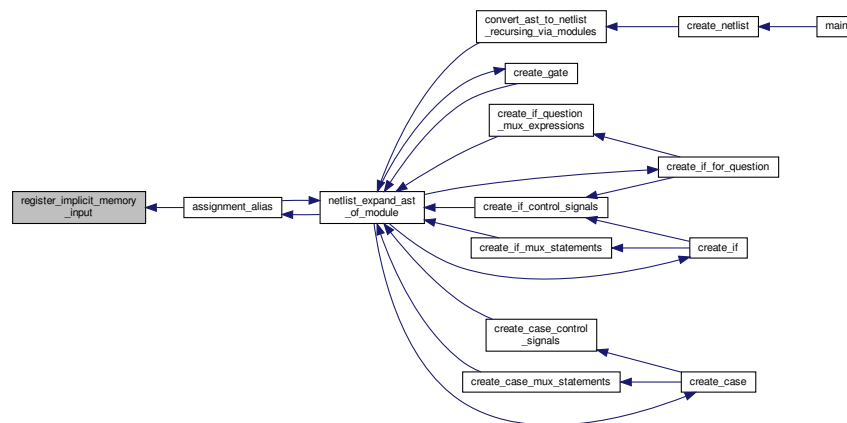


2.23.1.9 register_implicit_memory_input()

```
void register_implicit_memory_input (
    char * name,
    implicit_memory * memory )
```

Definition at line 146 of file implicit_memory.cpp.

Here is the caller graph for this function:



2.23.2 Variable Documentation

2.23.2.1 implicit_memories

```
hashtable_t* implicit_memories
```

Definition at line 32 of file implicit_memory.cpp.

2.23.2.2 implicit_memory_inputs

```
hashtable_t* implicit_memory_inputs
```

Definition at line 34 of file implicit_memory.cpp.

2.24 vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/memories.cpp File Reference

```
#include <string.h>
#include <math.h>
#include "globals.h"
#include "netlist_utils.h"
#include "node_creation_library.h"
#include "hard_blocks.h"
#include "memories.h"
#include "partial_map.h"
#include "vtr_util.h"
#include "vtr_memory.h"
```

Functions

- void [pad_dp_memory_width](#) (nnode_t *node, netlist_t *netlist)
- void [pad_sp_memory_width](#) (nnode_t *node, netlist_t *netlist)
- void [pad_memory_output_port](#) (nnode_t *node, netlist_t *netlist, t_model *model, const char *port_name)
- void [pad_memory_input_port](#) (nnode_t *node, netlist_t *netlist, t_model *model, const char *port_name)
- void [copy_input_port_to_memory](#) (nnode_t *node, signal_list_t *signals, const char *port_name)
- void [copy_output_port_to_memory](#) (nnode_t *node, signal_list_t *signals, const char *port_name)
- void [remap_input_port_to_memory](#) (nnode_t *node, signal_list_t *signals, const char *port_name)
- void [remap_output_port_to_memory](#) (nnode_t *node, signal_list_t *signalsvar, char *port_name)
- int [get_sp_ram_split_width](#) ()
- int [get_dp_ram_split_width](#) ()
- void [filter_memories_by_soft_logic_cutoff](#) ()
- int [get_sp_ram_depth](#) (nnode_t *node)
- int [get_dp_ram_depth](#) (nnode_t *node)
- int [get_sp_ram_width](#) (nnode_t *node)
- int [get_dp_ram_width](#) (nnode_t *node)
- int [get_memory_port_size](#) (const char *name)
- void [add_input_port_to_memory](#) (nnode_t *node, signal_list_t *signalsvar, const char *port_name)
- void [add_output_port_to_memory](#) (nnode_t *node, signal_list_t *signals, const char *port_name)
- void [check_memories_and_report_distribution](#) ()
- void [split_sp_memory_depth](#) (nnode_t *node, int split_size)
- void [split_dp_memory_depth](#) (nnode_t *node, int split_size)
- void [split_sp_memory_width](#) (nnode_t *node, int target_size)
- void [split_dp_memory_width](#) (nnode_t *node, int target_size)
- int [get_sp_ram_split_depth](#) ()
- int [get_dp_ram_split_depth](#) ()
- void [iterate_memories](#) (netlist_t *netlist)
- void [free_memory_lists](#) ()
- char [is_sp_ram](#) (nnode_t *node)
- char [is_dp_ram](#) (nnode_t *node)
- char [is_ast_sp_ram](#) (ast_node_t *node)
- char [is_ast_dp_ram](#) (ast_node_t *node)
- [sp_ram_signals](#) * [get_sp_ram_signals](#) (nnode_t *node)
- void [free_sp_ram_signals](#) ([sp_ram_signals](#) *signalsvar)
- [dp_ram_signals](#) * [get_dp_ram_signals](#) (nnode_t *node)
- void [free_dp_ram_signals](#) ([dp_ram_signals](#) *signalsvar)
- void [instantiate_soft_single_port_ram](#) (nnode_t *node, short mark, netlist_t *netlist)
- void [instantiate_soft_dual_port_ram](#) (nnode_t *node, short mark, netlist_t *netlist)
- signal_list_t * [create_decoder](#) (nnode_t *node, short mark, signal_list_t *input_list)

Variables

- t_model * [single_port_rams](#) = NULL
- t_model * [dual_port_rams](#) = NULL
- t_linked_vptr * [sp_memory_list](#)
- t_linked_vptr * [dp_memory_list](#)
- t_linked_vptr * [split_list](#)
- t_linked_vptr * [memory_instances](#) = NULL
- t_linked_vptr * [memory_port_size_list](#) = NULL

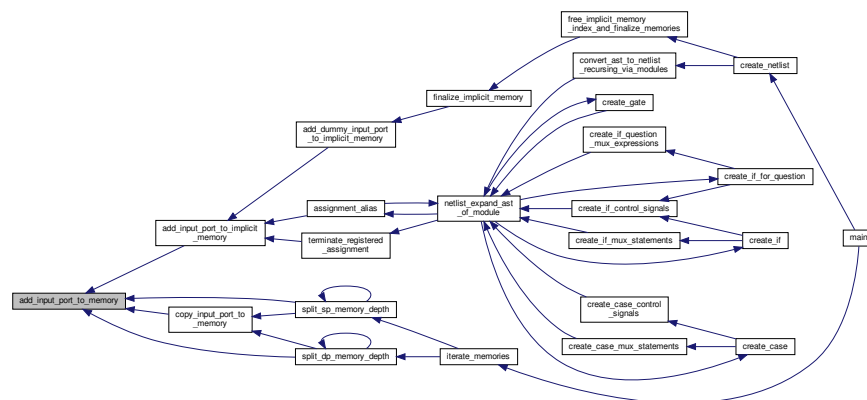
2.24.1 Function Documentation

2.24.1.1 add_input_port_to_memory()

```
void add_input_port_to_memory (
    nnode_t * node,
    signal_list_t * signalsvar,
    const char * port_name )
```

Definition at line 160 of file memories.cpp.

Here is the caller graph for this function:

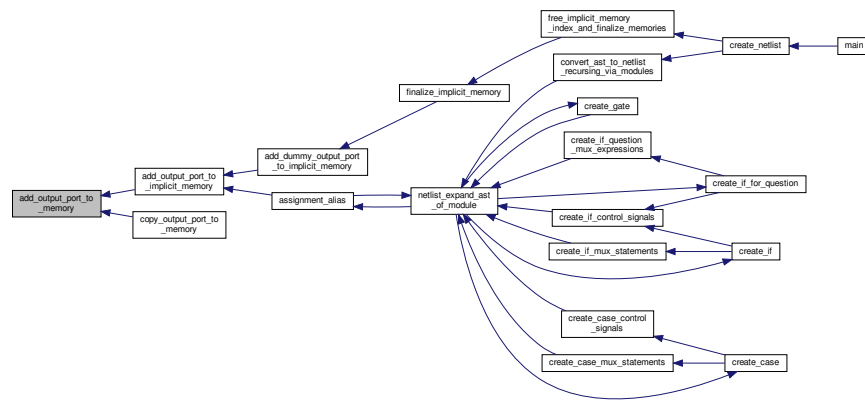


2.24.1.2 add_output_port_to_memory()

```
void add_output_port_to_memory (
    nnode_t * node,
    signal_list_t * signals,
    const char * port_name )
```

Definition at line 231 of file memories.cpp.

Here is the caller graph for this function:

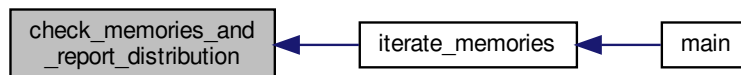


2.24.1.3 check_memories_and_report_distribution()

```
void check_memories_and_report_distribution ( )
```

Definition at line 265 of file memories.cpp.

Here is the caller graph for this function:

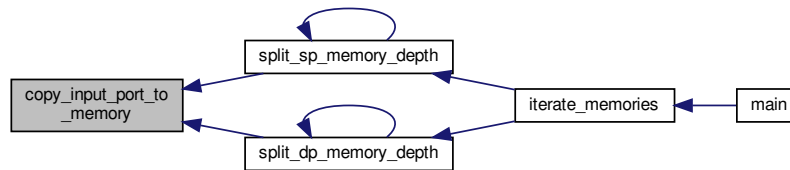


2.24.1.4 copy_input_port_to_memory()

```
void copy_input_port_to_memory (
    nnode_t * node,
    signal_list_t * signals,
    const char * port_name )
```

Definition at line 109 of file memories.cpp.

Here is the caller graph for this function:



2.24.1.5 copy_output_port_to_memory()

```
void copy_output_port_to_memory (
    nnode_t * node,
    signal_list_t * signals,
    const char * port_name )
```

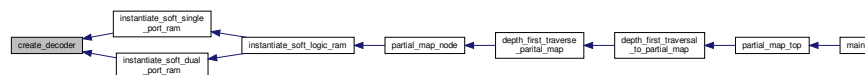
Definition at line 116 of file memories.cpp.

2.24.1.6 create_decoder()

```
signal_list_t* create_decoder (
    nnode_t * node,
    short mark,
    signal_list_t * input_list )
```

Definition at line 1676 of file memories.cpp.

Here is the caller graph for this function:

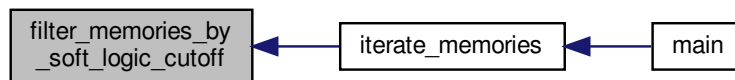


2.24.1.7 filter_memories_by_soft_logic_cutoff()

```
void filter_memories_by_soft_logic_cutoff ( )
```

Definition at line 995 of file memories.cpp.

Here is the caller graph for this function:

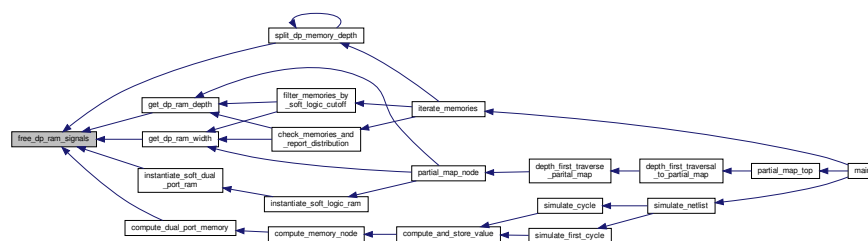


2.24.1.8 free_dp_ram_signals()

```
void free_dp_ram_signals (
    dp_ram_signals * signalsvar )
```

Definition at line 1437 of file memories.cpp.

Here is the caller graph for this function:



2.24.1.9 free_memory_lists()

```
void free_memory_lists ( )
```

Definition at line 1151 of file memories.cpp.

Here is the caller graph for this function:

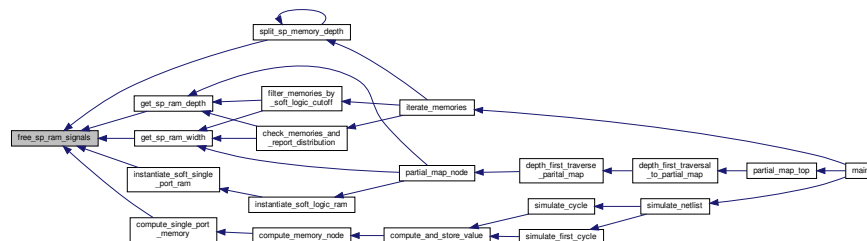


2.24.1.10 free_sp_ram_signals()

```
void free_sp_ram_signals (
    sp_ram_signals * signalsvar )
```

Definition at line 1357 of file memories.cpp.

Here is the caller graph for this function:

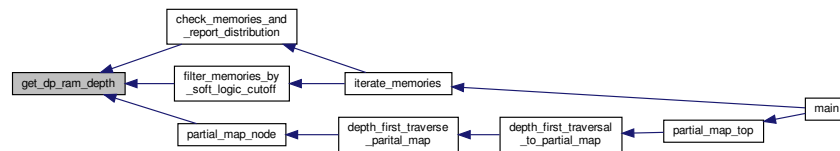


2.24.1.11 get_dp_ram_depth()

```
int get_dp_ram_depth (
    nnode_t * node )
```

Definition at line 69 of file memories.cpp.

Here is the caller graph for this function:

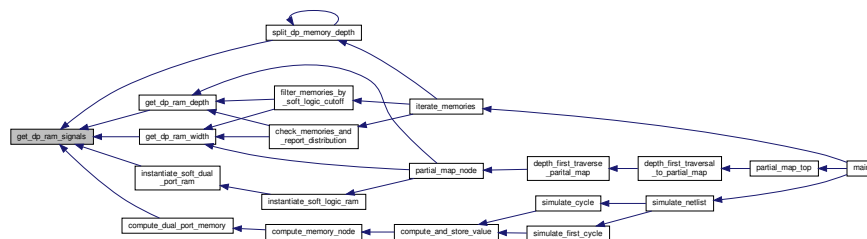


2.24.1.12 get_dp_ram_signals()

```
dp_ram_signals* get_dp_ram_signals (
    nnode_t * node )
```

Definition at line 1366 of file memories.cpp.

Here is the caller graph for this function:

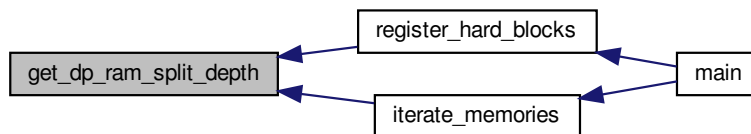


2.24.1.13 get_dp_ram_split_depth()

```
int get_dp_ram_split_depth ( )
```

Definition at line 934 of file memories.cpp.

Here is the caller graph for this function:

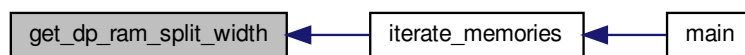


2.24.1.14 get_dp_ram_split_width()

```
int get_dp_ram_split_width ( )
```

Definition at line 975 of file memories.cpp.

Here is the caller graph for this function:

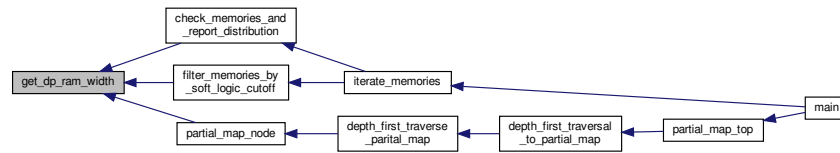


2.24.1.15 get_dp_ram_width()

```
int get_dp_ram_width (
    nnode_t * node )
```

Definition at line 86 of file memories.cpp.

Here is the caller graph for this function:

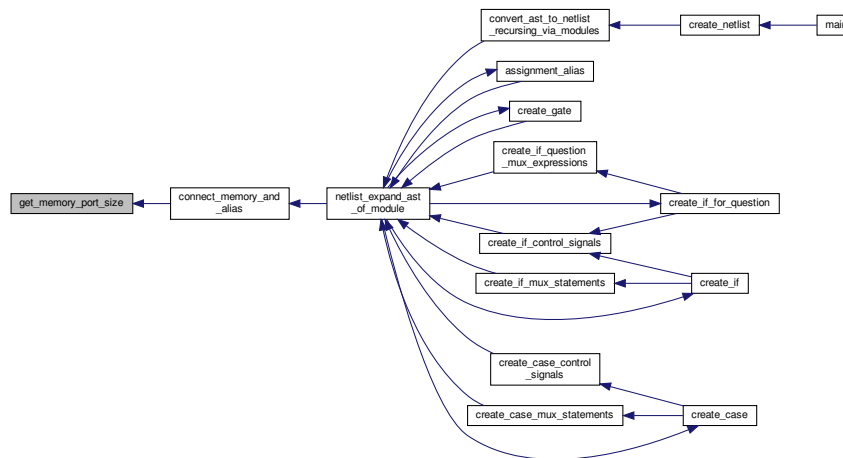


2.24.1.16 get_memory_port_size()

```
int get_memory_port_size (
    const char * name )
```

Definition at line 95 of file memories.cpp.

Here is the caller graph for this function:

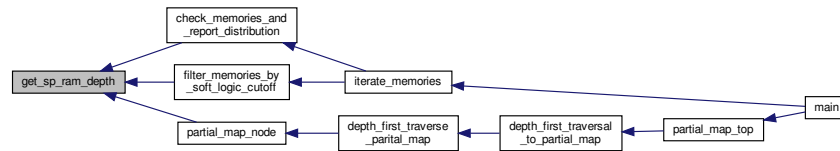


2.24.1.17 get_sp_ram_depth()

```
int get_sp_ram_depth (
    nnode_t * node )
```

Definition at line 61 of file memories.cpp.

Here is the caller graph for this function:

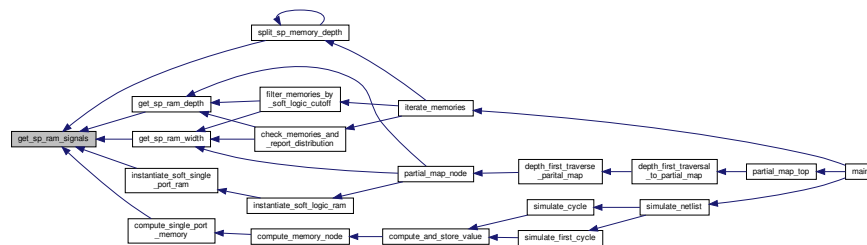


2.24.1.18 get_sp_ram_signals()

```
sp_ram_signals* get_sp_ram_signals (
    nnode_t * node )
```

Definition at line 1303 of file memories.cpp.

Here is the caller graph for this function:

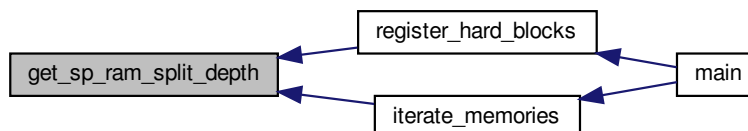


2.24.1.19 get_sp_ram_split_depth()

```
int get_sp_ram_split_depth ( )
```

Definition at line 912 of file memories.cpp.

Here is the caller graph for this function:



2.24.1.20 get_sp_ram_split_width()

```
int get_sp_ram_split_width ( )
```

Definition at line 956 of file memories.cpp.

Here is the caller graph for this function:

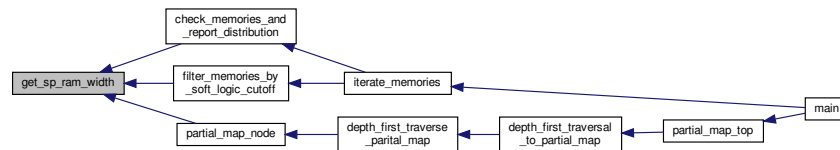


2.24.1.21 get_sp_ram_width()

```
int get_sp_ram_width (
    nnode_t * node )
```

Definition at line 78 of file memories.cpp.

Here is the caller graph for this function:



2.24.1.22 instantiate_soft_dual_port_ram()

```
void instantiate_soft_dual_port_ram (
    nnode_t * node,
    short mark,
    netlist_t * netlist )
```

Definition at line 1541 of file memories.cpp.

Here is the caller graph for this function:



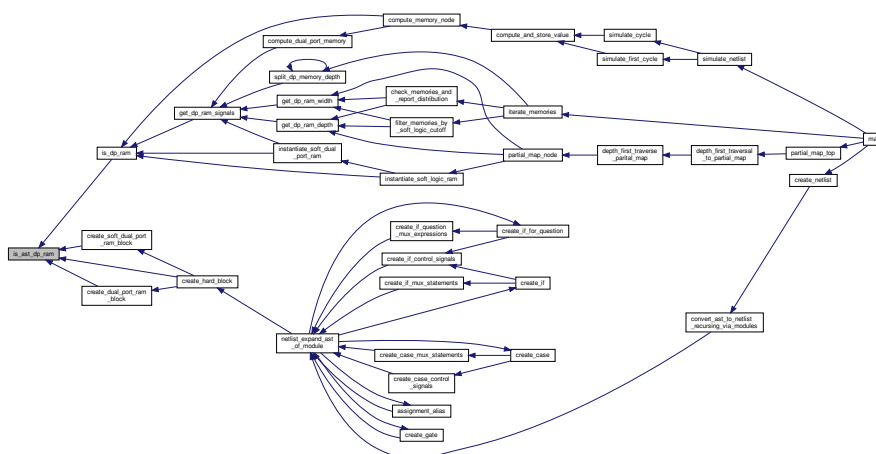
```
void instantiate_soft_single_port_ram (
    nnode_t * node,
    short mark,
    netlist_t * netlist )
```

Here is the caller graph for this function:



```
char is_ast_dp_ram (
    ast_node_t * node )
```

Here is the caller graph for this function:

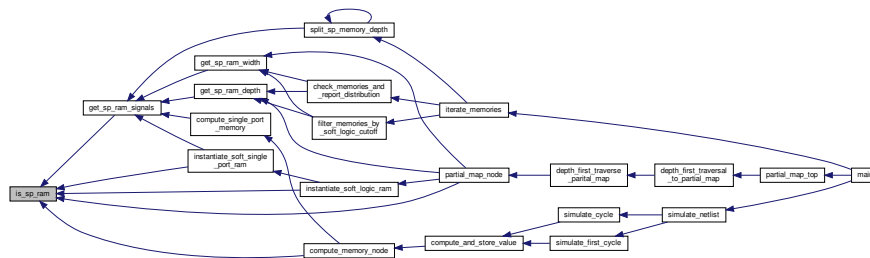


2.24.1.27 is_sp_ram()

```
char is_sp_ram (
    nnode_t * node )
```

Definition at line 1271 of file memories.cpp.

Here is the caller graph for this function:



2.24.1.28 iterate_memories()

```
void iterate_memories (
    netlist_t * netlist )
```

Definition at line 1045 of file memories.cpp.

Here is the caller graph for this function:

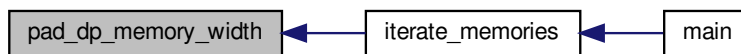


2.24.1.29 pad_dp_memory_width()

```
void pad_dp_memory_width (  
    nnode_t * node,  
    netlist_t * netlist )
```

Definition at line 1162 of file memories.cpp.

Here is the caller graph for this function:

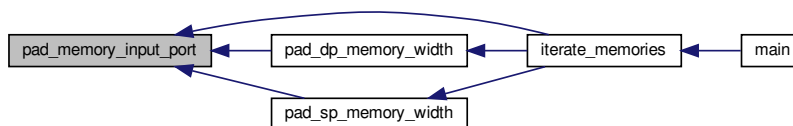


2.24.1.30 pad_memory_input_port()

```
void pad_memory_input_port (  
    nnode_t * node,  
    netlist_t * netlist,  
    t_model * model,  
    const char * port_name )
```

Definition at line 1233 of file memories.cpp.

Here is the caller graph for this function:

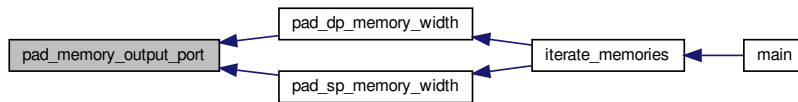


2.24.1.31 pad_memory_output_port()

```
void pad_memory_output_port (
    nnode_t * node,
    netlist_t * netlist,
    t_model * model,
    const char * port_name )
```

Definition at line 1194 of file memories.cpp.

Here is the caller graph for this function:

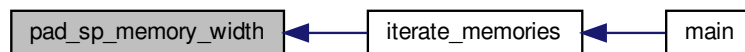


2.24.1.32 pad_sp_memory_width()

```
void pad_sp_memory_width (
    nnode_t * node,
    netlist_t * netlist )
```

Definition at line 1179 of file memories.cpp.

Here is the caller graph for this function:

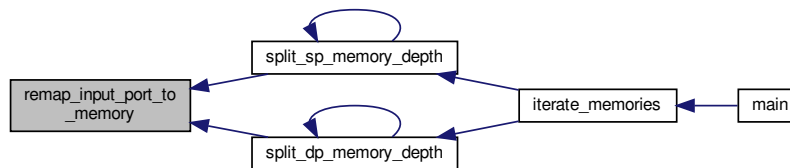


2.24.1.33 remap_input_port_to_memory()

```
void remap_input_port_to_memory (
    nnode_t * node,
    signal_list_t * signals,
    const char * port_name )
```

Definition at line 126 of file memories.cpp.

Here is the caller graph for this function:



2.24.1.34 remap_output_port_to_memory()

```
void remap_output_port_to_memory (
    nnode_t * node,
    signal_list_t * signalsvar,
    char * port_name )
```

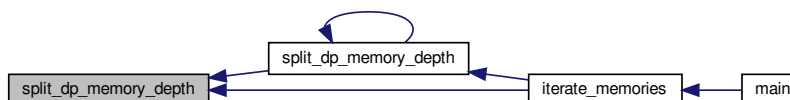
Definition at line 197 of file memories.cpp.

2.24.1.35 split_dp_memory_depth()

```
void split_dp_memory_depth (
    nnode_t * node,
    int split_size )
```

Definition at line 466 of file memories.cpp.

Here is the caller graph for this function:



2.24.1.36 split_dp_memory_width()

```
void split_dp_memory_width (  
    nnode_t * node,  
    int target_size )
```

Definition at line 736 of file memories.cpp.

Here is the caller graph for this function:

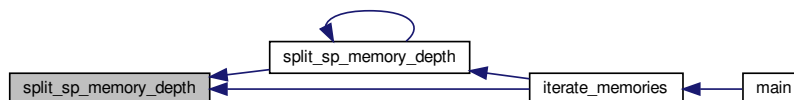


2.24.1.37 split_sp_memory_depth()

```
void split_sp_memory_depth (  
    nnode_t * node,  
    int split_size )
```

Definition at line 349 of file memories.cpp.

Here is the caller graph for this function:



2.24.1.38 split_sp_memory_width()

```
void split_sp_memory_width (  
    nnode_t * node,  
    int target_size )
```

Definition at line 632 of file memories.cpp.

Here is the caller graph for this function:



2.24.2 Variable Documentation

2.24.2.1 dp_memory_list

```
t_linked_vptr* dp_memory_list
```

Definition at line 42 of file memories.cpp.

2.24.2.2 dual_port_rams

```
t_model* dual_port_rams = NULL
```

Definition at line 39 of file memories.cpp.

2.24.2.3 memory_instances

```
t_linked_vptr* memory_instances = NULL
```

Definition at line 44 of file memories.cpp.

2.24.2.4 memory_port_size_list

```
t_linked_vptr* memory_port_size_list = NULL
```

Definition at line 45 of file memories.cpp.

2.24.2.5 single_port_rams

```
t_model* single_port_rams = NULL
```

Definition at line 38 of file memories.cpp.

2.24.2.6 sp_memory_list

```
t_linked_vptr* sp_memory_list
```

Definition at line 41 of file memories.cpp.

2.24.2.7 split_list

```
t_linked_vptr* split_list
```

Definition at line 43 of file memories.cpp.

2.25 vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/memories.h File Reference

Data Structures

- struct [s_memory](#)
- struct [s_memory_port_sizes](#)
- struct [sp_ram_signals](#)
- struct [dp_ram_signals](#)

Macros

- `#define` [MEMORY_DEPTH_LIMIT](#) 25

Typedefs

- typedef struct [s_memory](#) [t_memory](#)
- typedef struct [s_memory_port_sizes](#) [t_memory_port_sizes](#)

Functions

- [int get_sp_ram_split_depth \(\)](#)
- [int get_dp_ram_split_depth \(\)](#)
- [sp_ram_signals * get_sp_ram_signals \(nnode_t *node\)](#)
- [void free_sp_ram_signals \(sp_ram_signals *signalsvar\)](#)
- [dp_ram_signals * get_dp_ram_signals \(nnode_t *node\)](#)
- [void free_dp_ram_signals \(dp_ram_signals *signalsvar\)](#)
- [char is_sp_ram \(nnode_t *node\)](#)
- [char is_dp_ram \(nnode_t *node\)](#)
- [char is_ast_sp_ram \(ast_node_t *node\)](#)
- [char is_ast_dp_ram \(ast_node_t *node\)](#)
- [void init_memory_distribution \(\)](#)
- [void check_memories_and_report_distribution \(\)](#)
- [int get_memory_port_size \(const char *name\)](#)
- [int get_sp_ram_depth \(nnode_t *node\)](#)
- [int get_dp_ram_depth \(nnode_t *node\)](#)
- [int get_sp_ram_width \(nnode_t *node\)](#)
- [int get_dp_ram_width \(nnode_t *node\)](#)
- [void split_sp_memory_depth \(nnode_t *node, int split_size\)](#)
- [void split_dp_memory_depth \(nnode_t *node, int split_size\)](#)
- [void split_sp_memory_width \(nnode_t *node, int target_size\)](#)
- [void split_dp_memory_width \(nnode_t *node, int target_size\)](#)
- [void iterate_memories \(netlist_t *netlist\)](#)
- [void free_memory_lists \(\)](#)
- [void instantiate_soft_single_port_ram \(nnode_t *node, short mark, netlist_t *netlist\)](#)
- [void instantiate_soft_dual_port_ram \(nnode_t *node, short mark, netlist_t *netlist\)](#)
- [signal_list_t * create_decoder \(nnode_t *node, short mark, signal_list_t *input_list\)](#)
- [void add_input_port_to_memory \(nnode_t *node, signal_list_t *signalsvar, const char *port_name\)](#)
- [void add_output_port_to_memory \(nnode_t *node, signal_list_t *signalsvar, const char *port_name\)](#)

Variables

- [vtr::t_linked_vptr * sp_memory_list](#)
- [vtr::t_linked_vptr * dp_memory_list](#)
- [vtr::t_linked_vptr * memory_instances](#)
- [vtr::t_linked_vptr * memory_port_size_list](#)
- [t_model * single_port_rams](#)
- [t_model * dual_port_rams](#)

2.25.1 Macro Definition Documentation

2.25.1.1 MEMORY_DEPTH_LIMIT

```
#define MEMORY_DEPTH_LIMIT 25
```

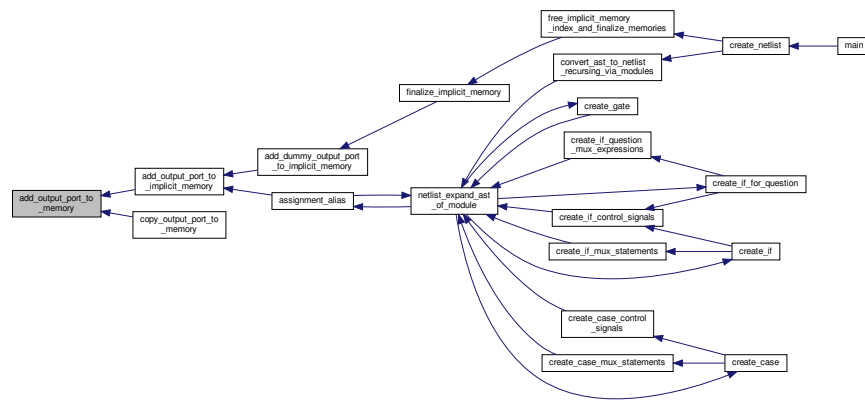
Definition at line 34 of file memories.h.

2.25.3.2 add_output_port_to_memory()

```
void add_output_port_to_memory (
    nnode_t * node,
    signal_list_t * signalsvar,
    const char * port_name )
```

Definition at line 231 of file memories.cpp.

Here is the caller graph for this function:

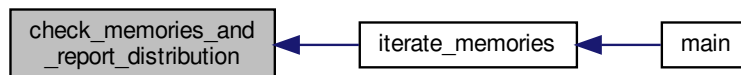


2.25.3.3 check_memories_and_report_distribution()

```
void check_memories_and_report_distribution ( )
```

Definition at line 265 of file memories.cpp.

Here is the caller graph for this function:



2.25.3.4 create_decoder()

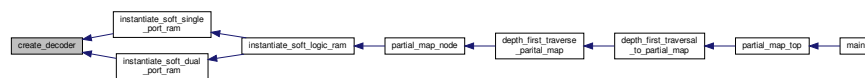
```

signal_list_t* create_decoder (
    nnode_t * node,
    short mark,
    signal_list_t * input_list )

```

Definition at line 1676 of file memories.cpp.

Here is the caller graph for this function:



2.25.3.5 free_dp_ram_signals()

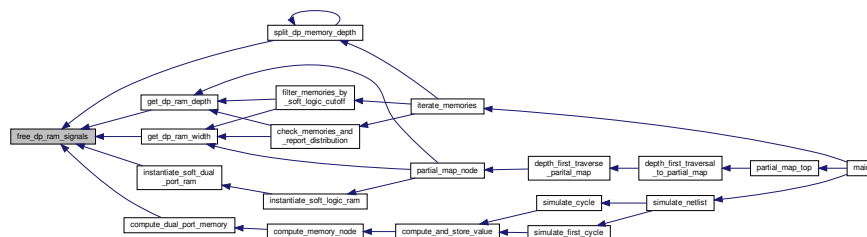
```

void free_dp_ram_signals (
    dp_ram_signals * signalsvar )

```

Definition at line 1437 of file memories.cpp.

Here is the caller graph for this function:



2.25.3.6 free_memory_lists()

```
void free_memory_lists ( )
```

Definition at line 1151 of file memories.cpp.

Here is the caller graph for this function:

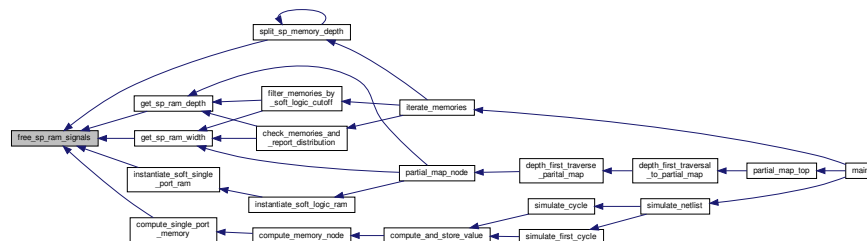


2.25.3.7 free_sp_ram_signals()

```
void free_sp_ram_signals (
    sp_ram_signals * signalsvar )
```

Definition at line 1357 of file memories.cpp.

Here is the caller graph for this function:

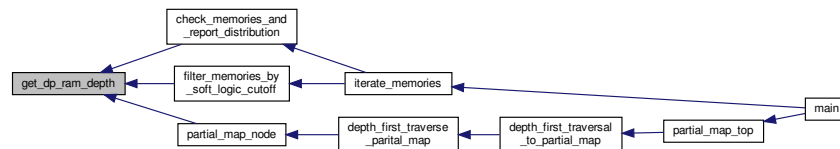


2.25.3.8 get_dp_ram_depth()

```
int get_dp_ram_depth (
    nnode_t * node )
```

Definition at line 69 of file memories.cpp.

Here is the caller graph for this function:

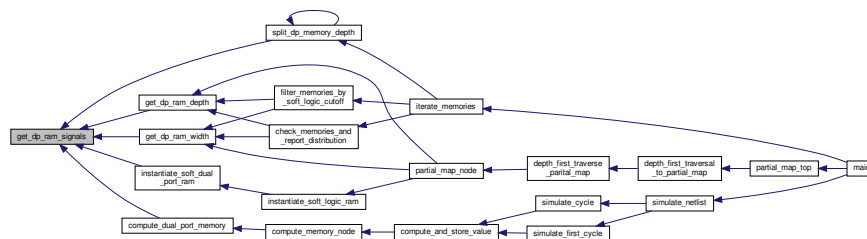


2.25.3.9 get_dp_ram_signals()

```
dp_ram_signals* get_dp_ram_signals (
    nnode_t * node )
```

Definition at line 1366 of file memories.cpp.

Here is the caller graph for this function:

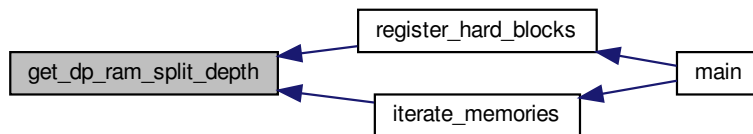


2.25.3.10 get_dp_ram_split_depth()

```
int get_dp_ram_split_depth ( )
```

Definition at line 934 of file memories.cpp.

Here is the caller graph for this function:

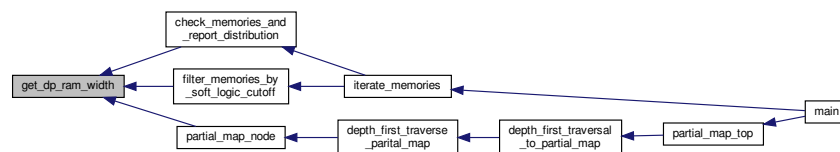


2.25.3.11 get_dp_ram_width()

```
int get_dp_ram_width (
    nnode_t * node )
```

Definition at line 86 of file memories.cpp.

Here is the caller graph for this function:

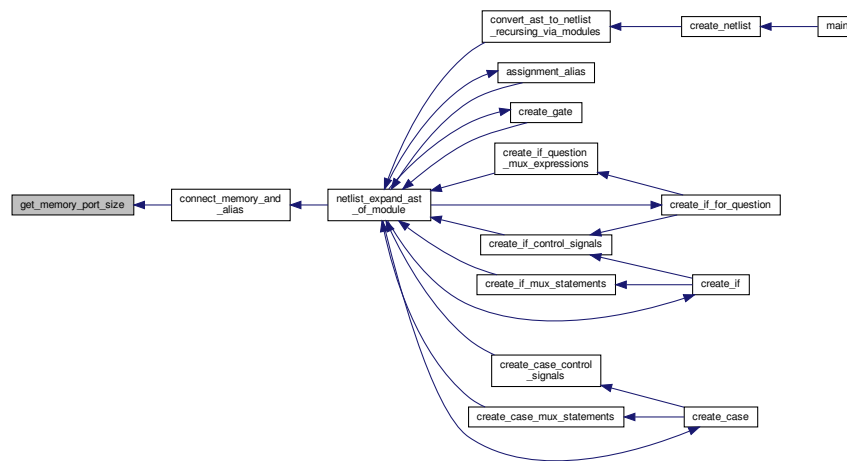


2.25.3.12 get_memory_port_size()

```
int get_memory_port_size (
    const char * name )
```

Definition at line 95 of file memories.cpp.

Here is the caller graph for this function:

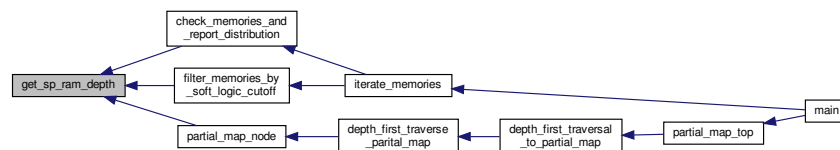


2.25.3.13 get_sp_ram_depth()

```
int get_sp_ram_depth (
    nnode_t * node )
```

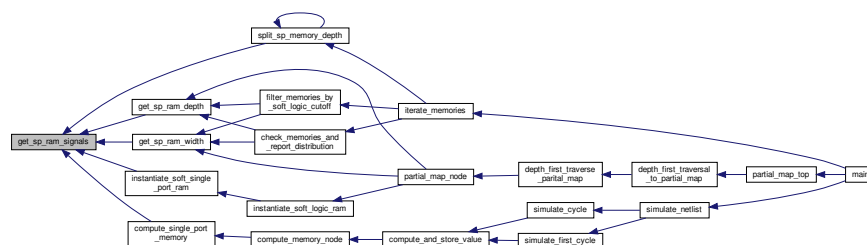
Definition at line 61 of file memories.cpp.

Here is the caller graph for this function:



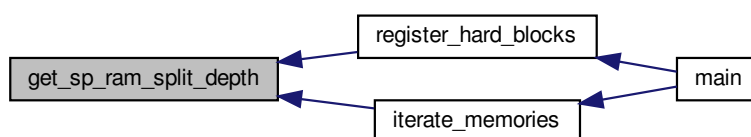

```
sp_ram_signals* get_sp_ram_signals (
    nnode_t * node )
```

Here is the caller graph for this function:



```
int get_sp_ram_split_depth ( )
```

Here is the caller graph for this function:

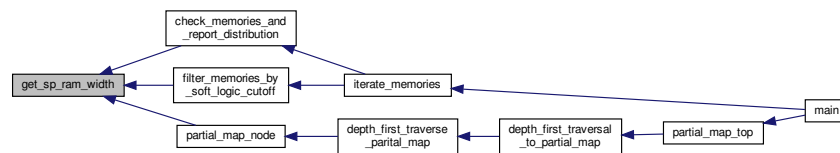


2.25.3.16 get_sp_ram_width()

```
int get_sp_ram_width (
    nnode_t * node )
```

Definition at line 78 of file memories.cpp.

Here is the caller graph for this function:



2.25.3.17 init_memory_distribution()

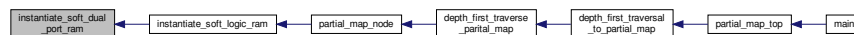
```
void init_memory_distribution ( )
```

2.25.3.18 instantiate_soft_dual_port_ram()

```
void instantiate_soft_dual_port_ram (
    nnode_t * node,
    short mark,
    netlist_t * netlist )
```

Definition at line 1541 of file memories.cpp.

Here is the caller graph for this function:



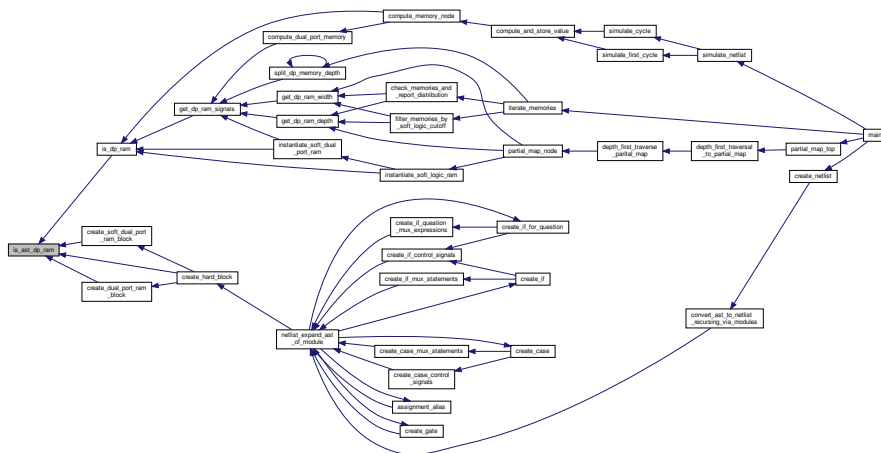
```
void instantiate_soft_single_port_ram (
    nnode_t * node,
    short mark,
    netlist_t * netlist )
```

Here is the caller graph for this function:



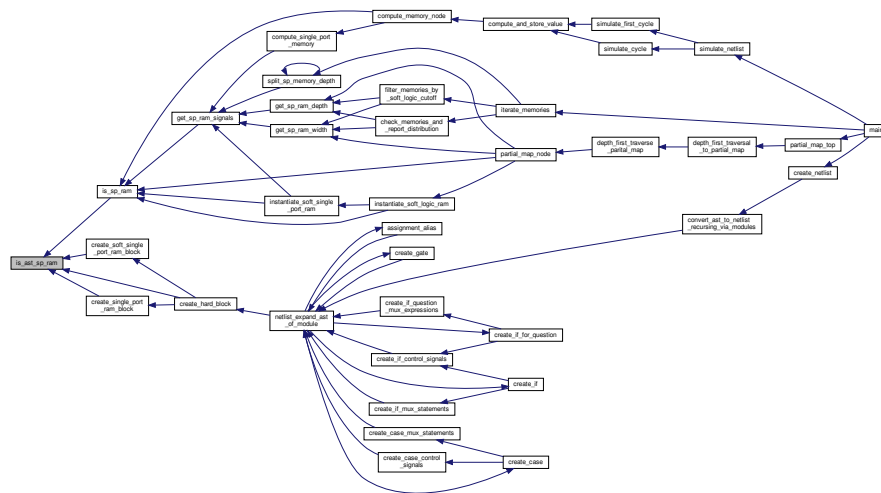
```
char is_ast_dp_ram (
    ast_node_t * node )
```

Here is the caller graph for this function:



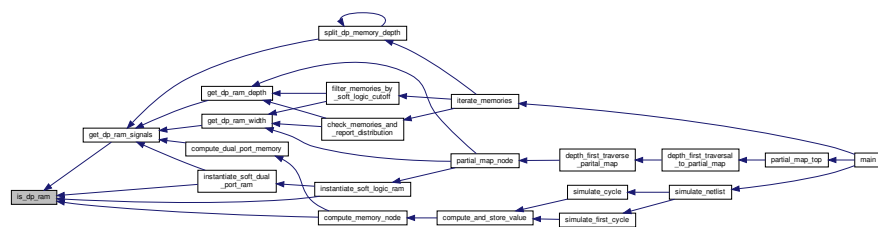
```
char is_ast_sp_ram (
    ast_node_t * node )
```

Here is the caller graph for this function:



```
char is_dp_ram (
    nnode_t * node )
```

Here is the caller graph for this function:

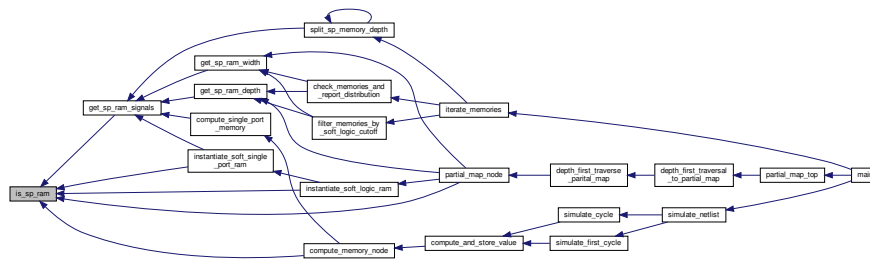


2.25.3.23 is_sp_ram()

```
char is_sp_ram (
    nnode_t * node )
```

Definition at line 1271 of file memories.cpp.

Here is the caller graph for this function:



2.25.3.24 iterate_memories()

```
void iterate_memories (
    netlist_t * netlist )
```

Definition at line 1045 of file memories.cpp.

Here is the caller graph for this function:

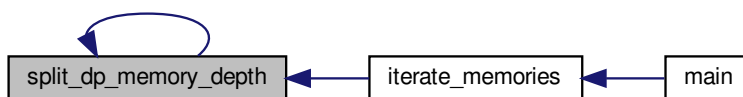


2.25.3.25 split_dp_memory_depth()

```
void split_dp_memory_depth (  
    nnode_t * node,  
    int split_size )
```

Definition at line 466 of file memories.cpp.

Here is the caller graph for this function:

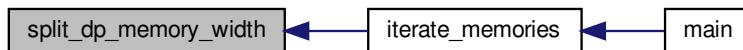


2.25.3.26 split_dp_memory_width()

```
void split_dp_memory_width (  
    nnode_t * node,  
    int target_size )
```

Definition at line 736 of file memories.cpp.

Here is the caller graph for this function:



2.25.3.27 split_sp_memory_depth()

```
void split_sp_memory_depth (  
    nnode_t * node,  
    int split_size )
```

Definition at line 349 of file memories.cpp.

Here is the caller graph for this function:

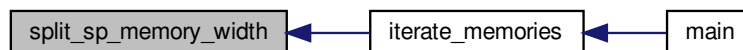


2.25.3.28 split_sp_memory_width()

```
void split_sp_memory_width (  
    nnode_t * node,  
    int target_size )
```

Definition at line 632 of file memories.cpp.

Here is the caller graph for this function:



2.25.4 Variable Documentation

2.25.4.1 dp_memory_list

```
vtr::t_linked_vptr* dp_memory_list
```

Definition at line 42 of file memories.cpp.

2.25.4.2 dual_port_rams

```
t_model* dual_port_rams
```

Definition at line 39 of file memories.cpp.

2.25.4.3 memory_instances

```
vtr::t_linked_vptr* memory_instances
```

Definition at line 44 of file memories.cpp.

2.25.4.4 memory_port_size_list

```
vtr::t_linked_vptr* memory_port_size_list
```

Definition at line 45 of file memories.cpp.

2.25.4.5 single_port_rams

```
t_model* single_port_rams
```

Definition at line 38 of file memories.cpp.

2.25.4.6 sp_memory_list

```
vtr::t_linked_vptr* sp_memory_list
```

Definition at line 41 of file memories.cpp.

2.26 vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/multipliers.cpp File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "types.h"
#include "node_creation_library.h"
#include "multipliers.h"
#include "netlist_utils.h"
#include "partial_map.h"
#include "read_xml_arch_file.h"
#include "globals.h"
#include "adders.h"
#include "vtr_memory.h"
#include "vtr_list.h"
```


Functions

- void `record_mult_distribution` (nnode_t *node)
- void `init_split_multiplier` (nnode_t *node, nnode_t *ptr, int offa, int a, int offb, int b)
- void `init_cascade_adder` (nnode_t *node, nnode_t *a, int b)
- void `split_multiplier_a` (nnode_t *node, int a0, int a1, int b)
- void `split_multiplier_b` (nnode_t *node, int a, int b1, int b0)
- void `pad_multiplier` (nnode_t *node, netlist_t *netlist)
- void `instantiate_simple_soft_multiplier` (nnode_t *node, short mark, netlist_t *netlist)
- void `init_mult_distribution` ()
- void `report_mult_distribution` ()
- void `find_hard_multipliers` ()
- void `declare_hard_multiplier` (nnode_t *node)
- void `instantiate_hard_multiplier` (nnode_t *node, short mark, netlist_t *)
- void `add_the_blackbox_for_mults` (FILE *out)
- void `define_mult_function` (nnode_t *node, short, FILE *out)
- void `split_multiplier` (nnode_t *node, int a0, int b0, int a1, int b1)
- void `iterate_multipliers` (netlist_t *netlist)
- void `clean_multipliers` ()

Variables

- t_model * `hard_multipliers` = NULL
- t_linked_vptr * `mult_list` = NULL
- int `min_mult` = 0
- int * `mults` = NULL

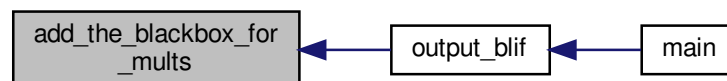
2.26.1 Function Documentation

2.26.1.1 `add_the_blackbox_for_mults()`

```
void add_the_blackbox_for_mults (
    FILE * out )
```

Definition at line 427 of file multipliers.cpp.

Here is the caller graph for this function:



2.26.1.2 clean_multipliers()

```
void clean_multipliers ( )
```

Definition at line 1166 of file multipliers.cpp.

Here is the caller graph for this function:



2.26.1.3 declare_hard_multiplier()

```
void declare_hard_multiplier (
    nnode_t * node )
```

Definition at line 343 of file multipliers.cpp.

Here is the caller graph for this function:

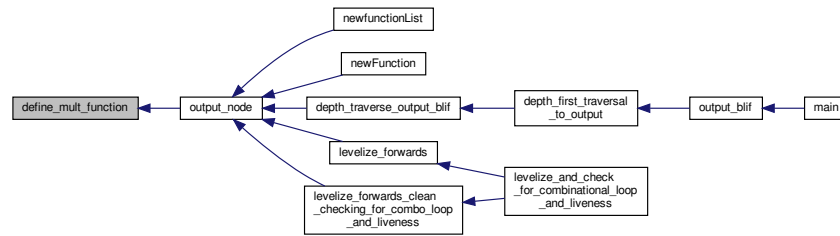


2.26.1.4 define_mult_function()

```
void define_mult_function (
    nnode_t * node,
    short ,
    FILE * out )
```

Definition at line 508 of file multipliers.cpp.

Here is the caller graph for this function:



2.26.1.5 find_hard_multipliers()

```
void find_hard_multipliers ( )
```

Definition at line 320 of file multipliers.cpp.

Here is the caller graph for this function:

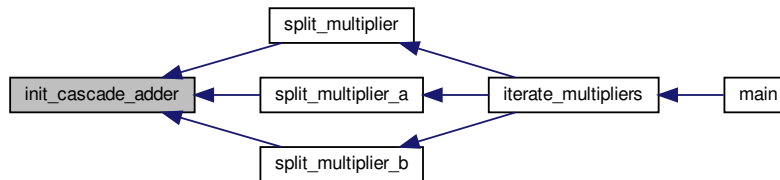


2.26.1.6 init_cascade_adder()

```
void init_cascade_adder (
    nnode_t * node,
    nnode_t * a,
    int b )
```

Definition at line 651 of file multipliers.cpp.

Here is the caller graph for this function:

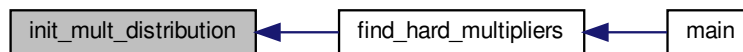


2.26.1.7 init_mult_distribution()

```
void init_mult_distribution ( )
```

Definition at line 259 of file `multipliers.cpp`.

Here is the caller graph for this function:

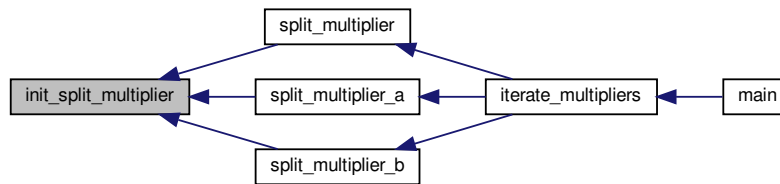


2.26.1.8 init_split_multiplier()

```
void init_split_multiplier (
    nnode_t * node,
    nnode_t * ptr,
    int offa,
    int a,
    int offb,
    int b )
```

Definition at line 603 of file `multipliers.cpp`.

Here is the caller graph for this function:



2.26.1.9 `instantiate_hard_multiplier()`

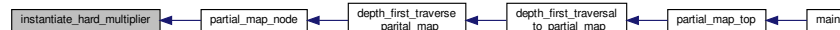
```

void instantiate_hard_multiplier (
    nnode_t * node,
    short mark,
    netlist_t * )

```

Definition at line 384 of file `multipliers.cpp`.

Here is the caller graph for this function:



2.26.1.10 `instantiate_simple_soft_multiplier()`

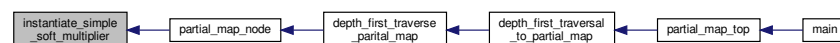
```

void instantiate_simple_soft_multiplier (
    nnode_t * node,
    short mark,
    netlist_t * netlist )

```

Definition at line 76 of file `multipliers.cpp`.

Here is the caller graph for this function:

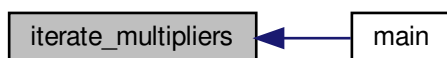


2.26.1.11 `iterate_multipliers()`

```
void iterate_multipliers (
    netlist_t * netlist )
```

Definition at line 1080 of file `multipliers.cpp`.

Here is the caller graph for this function:

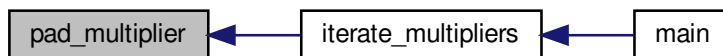


2.26.1.12 `pad_multiplier()`

```
void pad_multiplier (
    nnode_t * node,
    netlist_t * netlist )
```

Definition at line 964 of file `multipliers.cpp`.

Here is the caller graph for this function:

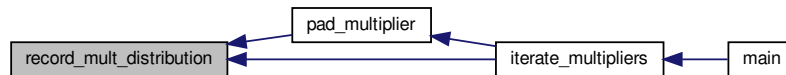


2.26.1.13 record_mult_distribution()

```
void record_mult_distribution (
    nnode_t * node )
```

Definition at line 274 of file multipliers.cpp.

Here is the caller graph for this function:

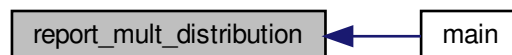


2.26.1.14 report_mult_distribution()

```
void report_mult_distribution ( )
```

Definition at line 291 of file multipliers.cpp.

Here is the caller graph for this function:

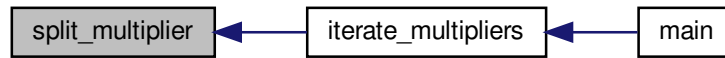


2.26.1.15 split_multiplier()

```
void split_multiplier (
    nnode_t * node,
    int a0,
    int b0,
    int a1,
    int b1 )
```

Definition at line 715 of file multipliers.cpp.

Here is the caller graph for this function:

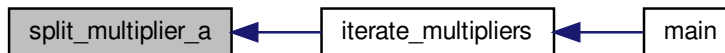


2.26.1.16 `split_multiplier_a()`

```
void split_multiplier_a (  
    nnode_t * node,  
    int a0,  
    int a1,  
    int b )
```

Definition at line 821 of file `multipliers.cpp`.

Here is the caller graph for this function:

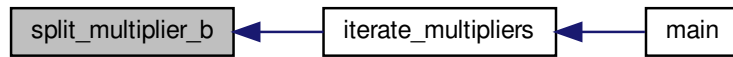


2.26.1.17 `split_multiplier_b()`

```
void split_multiplier_b (  
    nnode_t * node,  
    int a,  
    int b1,  
    int b0 )
```

Definition at line 897 of file `multipliers.cpp`.

Here is the caller graph for this function:



2.26.2 Variable Documentation

2.26.2.1 hard_multipliers

```
t_model* hard_multipliers = NULL
```

Definition at line 44 of file multipliers.cpp.

2.26.2.2 min_mult

```
int min_mult = 0
```

Definition at line 46 of file multipliers.cpp.

2.26.2.3 mult_list

```
t_linked_vptr* mult_list = NULL
```

Definition at line 45 of file multipliers.cpp.

2.26.2.4 mults

```
int* mults = NULL
```

Definition at line 47 of file multipliers.cpp.

2.27 vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/multipliers.h File Reference

```
#include "read_xml_arch_file.h"
```

Data Structures

- struct [s_multiplier](#)

Typedefs

- typedef struct [s_multiplier](#) [t_multiplier](#)

Functions

- void [init_mult_distribution](#) ()
- void [report_mult_distribution](#) ()
- void [declare_hard_multiplier](#) (nnode_t *node)
- void [instantiate_hard_multiplier](#) (nnode_t *node, short mark, netlist_t *netlist)
- void [instantiate_simple_soft_multiplier](#) (nnode_t *node, short mark, netlist_t *netlist)
- void [find_hard_multipliers](#) ()
- void [add_the_blackbox_for_mults](#) (FILE *out)
- void [define_mult_function](#) (nnode_t *node, short type, FILE *out)
- void [split_multiplier](#) (nnode_t *node, int a0, int b0, int a1, int b1)
- void [iterate_multipliers](#) (netlist_t *netlist)
- void [clean_multipliers](#) ()

Variables

- t_model * [hard_multipliers](#)
- vtr::t_linked_vptr * [mult_list](#)
- int [min_mult](#)

2.27.1 Typedef Documentation

2.27.1.1 t_multiplier

```
typedef struct s\_multiplier t\_multiplier
```

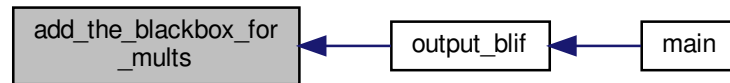
2.27.2 Function Documentation

2.27.2.1 add_the_blackbox_for_mults()

```
void add_the_blackbox_for_mults (
    FILE * out )
```

Definition at line 427 of file multipliers.cpp.

Here is the caller graph for this function:

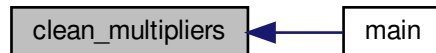


2.27.2.2 clean_multipliers()

```
void clean_multipliers ( )
```

Definition at line 1166 of file multipliers.cpp.

Here is the caller graph for this function:



2.27.2.3 declare_hard_multiplier()

```
void declare_hard_multiplier (
    nnode_t * node )
```

Definition at line 343 of file multipliers.cpp.

Here is the caller graph for this function:

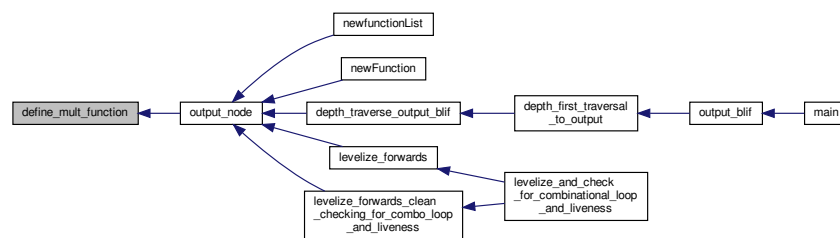


2.27.2.4 define_mult_function()

```
void define_mult_function (
    nnode_t * node,
    short type,
    FILE * out )
```

Definition at line 508 of file multipliers.cpp.

Here is the caller graph for this function:

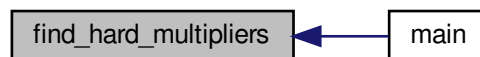


2.27.2.5 find_hard_multipliers()

```
void find_hard_multipliers ( )
```

Definition at line 320 of file multipliers.cpp.

Here is the caller graph for this function:

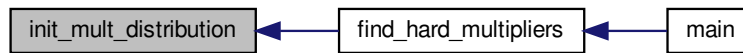


2.27.2.6 init_mult_distribution()

```
void init_mult_distribution ( )
```

Definition at line 259 of file multipliers.cpp.

Here is the caller graph for this function:

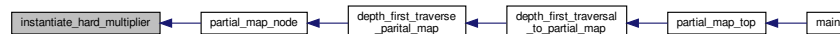


2.27.2.7 instantiate_hard_multiplier()

```
void instantiate_hard_multiplier (
    nnode_t * node,
    short mark,
    netlist_t * netlist )
```

Definition at line 384 of file multipliers.cpp.

Here is the caller graph for this function:

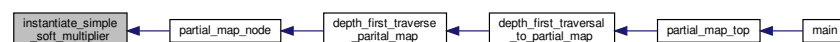


2.27.2.8 instantiate_simple_soft_multiplier()

```
void instantiate_simple_soft_multiplier (
    nnode_t * node,
    short mark,
    netlist_t * netlist )
```

Definition at line 76 of file multipliers.cpp.

Here is the caller graph for this function:

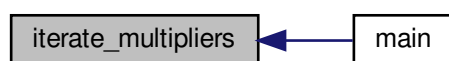


2.27.2.9 iterate_multipliers()

```
void iterate_multipliers (  
    netlist_t * netlist )
```

Definition at line 1080 of file multipliers.cpp.

Here is the caller graph for this function:

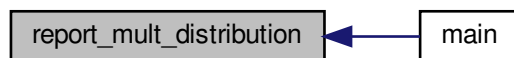


2.27.2.10 report_mult_distribution()

```
void report_mult_distribution ( )
```

Definition at line 291 of file multipliers.cpp.

Here is the caller graph for this function:

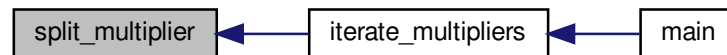


2.27.2.11 split_multiplier()

```
void split_multiplier (
    nnode_t * node,
    int a0,
    int b0,
    int a1,
    int b1 )
```

Definition at line 715 of file multipliers.cpp.

Here is the caller graph for this function:



2.27.3 Variable Documentation

2.27.3.1 hard_multipliers

```
t_model* hard_multipliers
```

Definition at line 44 of file multipliers.cpp.

2.27.3.2 min_mult

```
int min_mult
```

Definition at line 46 of file multipliers.cpp.

2.27.3.3 mult_list

```
vtr::t_linked_vptr* mult_list
```

Definition at line 45 of file multipliers.cpp.

2.28 vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/subtractions.cpp File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include "types.h"
#include "node_creation_library.h"
#include "adders.h"
#include "subtractions.h"
#include "netlist_utils.h"
#include "partial_map.h"
#include "read_xml_arch_file.h"
#include "globals.h"
#include "vtr_memory.h"
#include "vtr_util.h"
```

Functions

- void [init_split_adder_for_sub](#) (nnode_t *node, nnode_t *ptr, int a, int sizea, int b, int sizeb, int cin, int cout, int index, int flag)
- void [report_sub_distribution](#) ()
- void [declare_hard_adder_for_sub](#) (nnode_t *node)
- void [instantiate_hard_adder_subtraction](#) (nnode_t *node, short mark, netlist_t *)
- void [split_adder_for_sub](#) (nnode_t *nodeo, int a, int b, int sizea, int sizeb, int cin, int cout, int [count](#), netlist_t *netlist)
- void [iterate_adders_for_sub](#) (netlist_t *netlist)
- void [clean_adders_for_sub](#) ()

Variables

- t_linked_vptr * [sub_list](#) = NULL
- t_linked_vptr * [sub_chain_list](#) = NULL
- int [subchaintotal](#) = 0
- int * [sub](#) = NULL
- int [subtractor_chain_count](#)
- int [longest_subtractor_chain](#)
- int [total_subtractors](#)

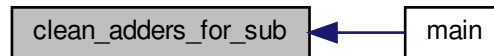
2.28.1 Function Documentation

2.28.1.1 clean_adders_for_sub()

```
void clean_adders_for_sub ( )
```

Definition at line 732 of file subtractions.cpp.

Here is the caller graph for this function:



2.28.1.2 declare_hard_adder_for_sub()

```
void declare_hard_adder_for_sub (
    nnode_t * node )
```

Definition at line 84 of file subtractions.cpp.

Here is the caller graph for this function:

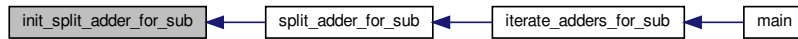


2.28.1.3 init_split_adder_for_sub()

```
void init_split_adder_for_sub (
    nnode_t * node,
    nnode_t * ptr,
    int a,
    int sizea,
    int b,
    int sizeb,
    int cin,
    int cout,
    int index,
    int flag )
```

Definition at line 164 of file subtractions.cpp.

Here is the caller graph for this function:

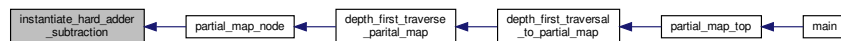


2.28.1.4 instantiate_hard_adder_subtraction()

```
void instantiate_hard_adder_subtraction (  
    nnode_t * node,  
    short mark,  
    netlist_t * )
```

Definition at line 122 of file subtractions.cpp.

Here is the caller graph for this function:

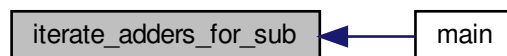


2.28.1.5 iterate_adders_for_sub()

```
void iterate_adders_for_sub (  
    netlist_t * netlist )
```

Definition at line 661 of file subtractions.cpp.

Here is the caller graph for this function:

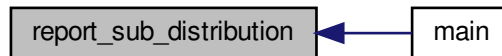


2.28.1.6 report_sub_distribution()

```
void report_sub_distribution ( )
```

Definition at line 59 of file subtractions.cpp.

Here is the caller graph for this function:

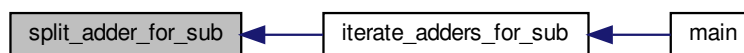


2.28.1.7 split_adder_for_sub()

```
void split_adder_for_sub (
    nnode_t * nodeo,
    int a,
    int b,
    int sizea,
    int sizeb,
    int cin,
    int cout,
    int count,
    netlist_t * netlist )
```

Definition at line 361 of file subtractions.cpp.

Here is the caller graph for this function:



2.28.2 Variable Documentation

2.28.2.1 longest_subtractor_chain

```
int longest_subtractor_chain
```

Definition at line 185 of file netlist_cleanup.cpp.

2.28.2.2 sub

```
int* sub = NULL
```

Definition at line 46 of file subtractions.cpp.

2.28.2.3 sub_chain_list

```
t_linked_vptr* sub_chain_list = NULL
```

Definition at line 44 of file subtractions.cpp.

2.28.2.4 sub_list

```
t_linked_vptr* sub_list = NULL
```

Definition at line 43 of file subtractions.cpp.

2.28.2.5 subchaintotal

```
int subchaintotal = 0
```

Definition at line 45 of file subtractions.cpp.

2.28.2.6 subtractor_chain_count

```
int subtractor_chain_count
```

Definition at line 184 of file netlist_cleanup.cpp.

2.28.2.7 total_subtractors

```
int total_subtractors
```

Definition at line 186 of file netlist_cleanup.cpp.

2.29 vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/subtractions.h File Reference

```
#include "read_xml_arch_file.h"  
#include "adders.h"
```

Functions

- void [init_sub_distribution](#) ()
- void [report_sub_distribution](#) ()
- void [declare_hard_adder_for_sub](#) (nnode_t *node)
- void [instantiate_hard_adder_subtraction](#) (nnode_t *node, short mark, netlist_t *netlist)
- void [split_adder_for_sub](#) (nnode_t *node, int a, int b, int sizea, int sizeb, int cin, int cout, int [count](#), netlist_t *netlist)
- void [iterate_adders_for_sub](#) (netlist_t *netlist)
- void [clean_adders_for_sub](#) ()

Variables

- vtr::t_linked_vptr * [sub_list](#)
- vtr::t_linked_vptr * [sub_chain_list](#)

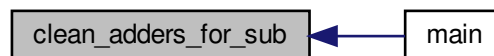
2.29.1 Function Documentation

2.29.1.1 clean_adders_for_sub()

```
void clean_adders_for_sub ( )
```

Definition at line 732 of file subtractions.cpp.

Here is the caller graph for this function:



2.29.1.2 declare_hard_adder_for_sub()

```
void declare_hard_adder_for_sub (  
    nnode_t * node )
```

Definition at line 84 of file subtractions.cpp.

Here is the caller graph for this function:



2.29.1.3 init_sub_distribution()

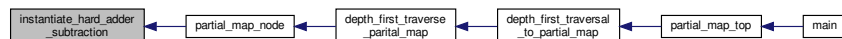
```
void init_sub_distribution ( )
```

2.29.1.4 instantiate_hard_adder_subtraction()

```
void instantiate_hard_adder_subtraction (  
    nnode_t * node,  
    short mark,  
    netlist_t * netlist )
```

Definition at line 122 of file subtractions.cpp.

Here is the caller graph for this function:

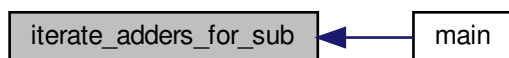


2.29.1.5 `iterate_adders_for_sub()`

```
void iterate_adders_for_sub (
    netlist_t * netlist )
```

Definition at line 661 of file `subtractions.cpp`.

Here is the caller graph for this function:

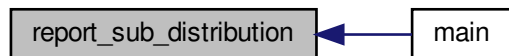


2.29.1.6 `report_sub_distribution()`

```
void report_sub_distribution ( )
```

Definition at line 59 of file `subtractions.cpp`.

Here is the caller graph for this function:

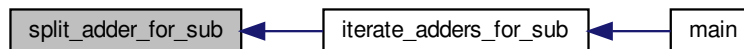


2.29.1.7 split_adder_for_sub()

```
void split_adder_for_sub (  
    nnode_t * node,  
    int a,  
    int b,  
    int sizea,  
    int sizeb,  
    int cin,  
    int cout,  
    int count,  
    netlist_t * netlist )
```

Definition at line 361 of file subtractions.cpp.

Here is the caller graph for this function:



2.29.2 Variable Documentation

2.29.2.1 sub_chain_list

```
vtr::t_linked_vptr* sub_chain_list
```

Definition at line 44 of file subtractions.cpp.

2.29.2.2 sub_list

```
vtr::t_linked_vptr* sub_list
```

Definition at line 43 of file subtractions.cpp.

2.30 vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_check.cpp File Reference

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "types.h"
#include "globals.h"
#include "netlist_utils.h"
#include "odin_util.h"
#include "ast_util.h"
#include "string_cache.h"
#include "netlist_check.h"
#include "netlist_visualizer.h"
#include "vtr_memory.h"
```

Functions

- void [levelize_backwards](#) (netlist_t *netlist)
- void [levelize_backwards_clean_checking_for_liveness](#) (short ast_based, netlist_t *netlist)
- void [levelize_forwards](#) (netlist_t *netlist)
- void [levelize_forwards_clean_checking_for_combo_loop_and_liveness](#) (short ast_based, netlist_t *netlist)
- nnode_t * [find_node_at_top_of_combo_loop](#) (nnode_t *start_node)
- void [depth_first_traversal_check_if_forward_leveled](#) (short marker_value, netlist_t *netlist)
- void [depth_first_traverse_check_if_forward_leveled](#) (nnode_t *node, int traverse_mark_number)
- void [sequential_levelized_dfs](#) (short marker_value, netlist_t *netlist)
- void [depth_first_traverse_until_next_ff_or_output](#) (nnode_t *node, nnode_t *calling_node, int traverse_mark_number, int seq_level, netlist_t *netlist)
- void [check_netlist](#) (netlist_t *netlist)
- void [depth_traverse_check_combinational_loop](#) (nnode_t *node, short start, [STRING_CACHE](#) *in_path)
- void [levelize_and_check_for_combinational_loop_and_liveness](#) (short ast_based, netlist_t *netlist)

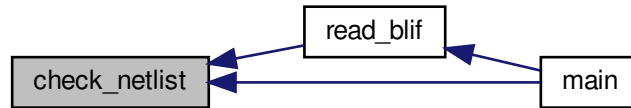
2.30.1 Function Documentation

2.30.1.1 [check_netlist\(\)](#)

```
void check_netlist (
    netlist_t * netlist )
```

Definition at line 53 of file netlist_check.cpp.

Here is the caller graph for this function:

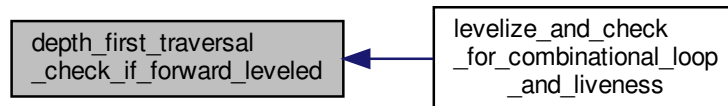


2.30.1.2 `depth_first_traversal_check_if_forward_leveled()`

```
void depth_first_traversal_check_if_forward_leveled (  
    short marker_value,  
    netlist_t * netlist )
```

Definition at line 244 of file `netlist_check.cpp`.

Here is the caller graph for this function:

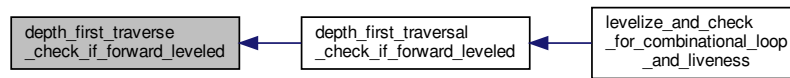


2.30.1.3 `depth_first_traverse_check_if_forward_leveled()`

```
void depth_first_traverse_check_if_forward_leveled (  
    nnode_t * node,  
    int traverse_mark_number )
```

Definition at line 266 of file `netlist_check.cpp`.

Here is the caller graph for this function:



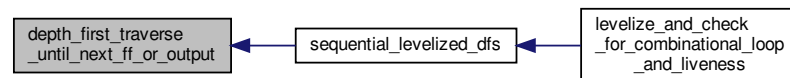
2.30.1.4 depth_first_traverse_until_next_ff_or_output()

```

void depth_first_traverse_until_next_ff_or_output (
    nnode_t * node,
    nnode_t * calling_node,
    int traverse_mark_number,
    int seq_level,
    netlist_t * netlist )
  
```

Definition at line 166 of file netlist_check.cpp.

Here is the caller graph for this function:

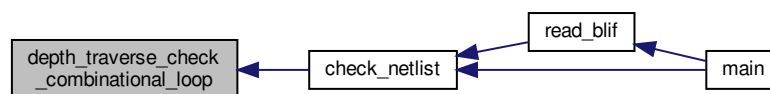


2.30.1.5 depth_traverse_check_combinational_loop()

```

void depth_traverse_check_combinational_loop (
    nnode_t * node,
    short start,
    STRING_CACHE * in_path )
  
```

Here is the caller graph for this function:

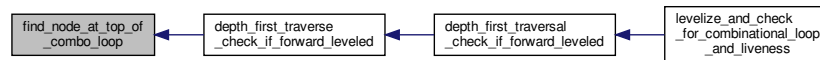


2.30.1.6 find_node_at_top_of_combo_loop()

```
nnode_t * find_node_at_top_of_combo_loop (  
    nnode_t * start_node )
```

Definition at line 793 of file netlist_check.cpp.

Here is the caller graph for this function:



2.30.1.7 levelize_and_check_for_combinational_loop_and_liveness()

```
void levelize_and_check_for_combinational_loop_and_liveness (  
    short ast_based,  
    netlist_t * netlist )
```

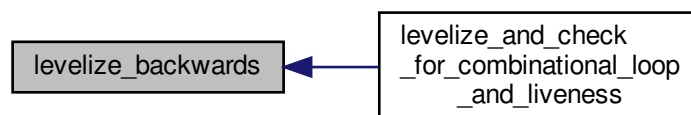
Definition at line 67 of file netlist_check.cpp.

2.30.1.8 levelize_backwards()

```
void levelize_backwards (  
    netlist_t * netlist )
```

Definition at line 577 of file netlist_check.cpp.

Here is the caller graph for this function:

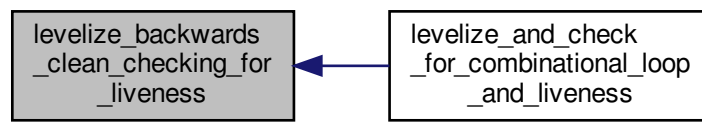


2.30.1.9 levelize_backwards_clean_checking_for_liveness()

```
void levelize_backwards_clean_checking_for_liveness (
    short ast_based,
    netlist_t * netlist )
```

Definition at line 709 of file netlist_check.cpp.

Here is the caller graph for this function:

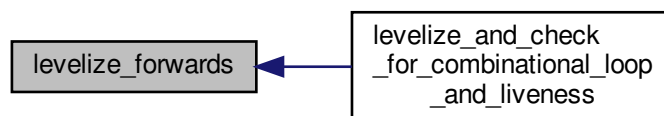


2.30.1.10 levelize_forwards()

```
void levelize_forwards (
    netlist_t * netlist )
```

Definition at line 315 of file netlist_check.cpp.

Here is the caller graph for this function:

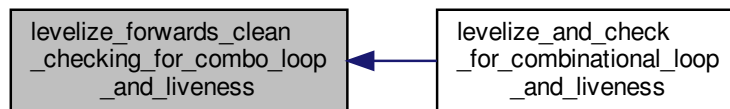


2.30.1.11 levelize_forwards_clean_checking_for_combo_loop_and_liveness()

```
void levelize_forwards_clean_checking_for_combo_loop_and_liveness (  
    short ast_based,  
    netlist_t * netlist )
```

Definition at line 476 of file netlist_check.cpp.

Here is the caller graph for this function:

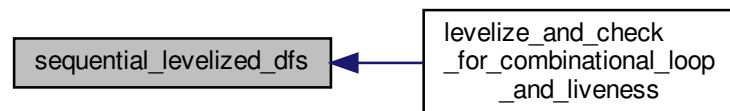


2.30.1.12 sequential_levelized_dfs()

```
void sequential_levelized_dfs (  
    short marker_value,  
    netlist_t * netlist )
```

Definition at line 90 of file netlist_check.cpp.

Here is the caller graph for this function:



2.31 vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_check.h File Reference

Functions

- void [check_netlist](#) (netlist_t *netlist)
- void [levelize_and_check_for_combinational_loop_and_liveness](#) (short ast_based, netlist_t *netlist)

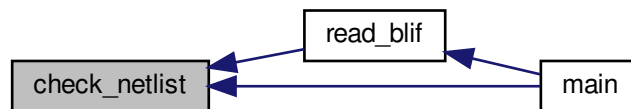
2.31.1 Function Documentation

2.31.1.1 `check_netlist()`

```
void check_netlist (
    netlist_t * netlist )
```

Definition at line 53 of file `netlist_check.cpp`.

Here is the caller graph for this function:



2.31.1.2 `levelize_and_check_for_combinational_loop_and_liveness()`

```
void levelize_and_check_for_combinational_loop_and_liveness (
    short ast_based,
    netlist_t * netlist )
```

Definition at line 67 of file `netlist_check.cpp`.

2.32 `vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_cleanup.cpp` File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "types.h"
#include "globals.h"
#include "netlist_utils.h"
#include "vtr_memory.h"
```

Data Structures

- struct [node_list_t](#)

Macros

- `#define VISITED_FORWARD ((void*)&_visited_forward)`
- `#define VISITED_BACKWARD ((void*)&_visited_backward)`
- `#define VISITED_REMOVAL ((void*)&_visited_removal)`

Typedefs

- `typedef struct node_list_t_t node_list_t`

Functions

- `node_list_t * insert_node_list (node_list_t *node_list, nnode_t *node)`
- `void traverse_backward (nnode_t *node)`
- `void traverse_forward (nnode_t *node, int toplevel, int remove_me)`
- `void mark_output_dependencies (netlist_t *netlist)`
- `void identify_unused_nodes (netlist_t *netlist)`
- `void remove_unused_nodes (node_list_t *remove)`
- `void calculate_addsub_statistics (node_list_t *addsub)`
- `void remove_unused_logic (netlist_t *netlist)`

Variables

- `int _visited_forward`
- `int _visited_backward`
- `int _visited_removal`
- `node_list_t useless_nodes`
- `node_list_t * removal_list_next = &useless_nodes`
- `node_list_t addsub_nodes`
- `node_list_t * addsub_list_next = &addsub_nodes`
- `int adder_chain_count = 0`
- `int longest_adder_chain = 0`
- `int total_adders = 0`
- `int subtractor_chain_count = 0`
- `int longest_subtractor_chain = 0`
- `int total_subtractors = 0`
- `double geomean_addsub_length = 0.0`
- `double sum_of_addsub_logs = 0.0`
- `int total_addsub_chain_count = 0`

2.32.1 Macro Definition Documentation

2.32.1.1 VISITED_BACKWARD

```
#define VISITED_BACKWARD ((void*)&_visited_backward)
```

Definition at line 37 of file netlist_cleanup.cpp.

2.32.1.2 VISITED_FORWARD

```
#define VISITED_FORWARD ((void*)&_visited_forward)
```

Definition at line 36 of file netlist_cleanup.cpp.

2.32.1.3 VISITED_REMOVAL

```
#define VISITED_REMOVAL ((void*)&_visited_removal)
```

Definition at line 38 of file netlist_cleanup.cpp.

2.32.2 Typedef Documentation

2.32.2.1 node_list_t

```
typedef struct node_list_t_t node_list_t
```

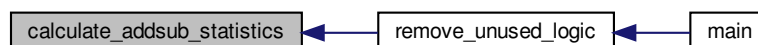
2.32.3 Function Documentation

2.32.3.1 calculate_addsub_statistics()

```
void calculate_addsub_statistics (  
    node_list_t * addsub )
```

Definition at line 192 of file netlist_cleanup.cpp.

Here is the caller graph for this function:

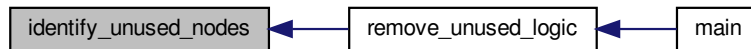


2.32.3.2 identify_unused_nodes()

```
void identify_unused_nodes (  
    netlist_t * netlist )
```

Definition at line 146 of file netlist_cleanup.cpp.

Here is the caller graph for this function:

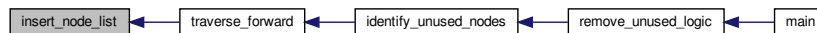


2.32.3.3 insert_node_list()

```
node_list_t * insert_node_list (  
    node_list_t * node_list,  
    nnode_t * node )
```

Definition at line 67 of file netlist_cleanup.cpp.

Here is the caller graph for this function:

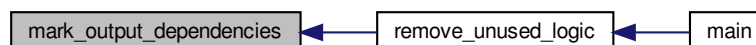


2.32.3.4 mark_output_dependencies()

```
void mark_output_dependencies (  
    netlist_t * netlist )
```

Definition at line 137 of file netlist_cleanup.cpp.

Here is the caller graph for this function:

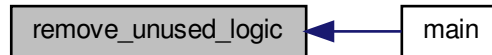


2.32.3.5 remove_unused_logic()

```
void remove_unused_logic (
    netlist_t * netlist )
```

Definition at line 232 of file netlist_cleanup.cpp.

Here is the caller graph for this function:

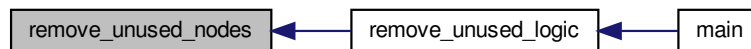


2.32.3.6 remove_unused_nodes()

```
void remove_unused_nodes (
    node_list_t * remove )
```

Definition at line 165 of file netlist_cleanup.cpp.

Here is the caller graph for this function:

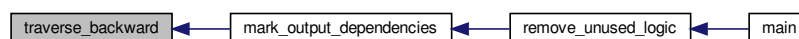


2.32.3.7 traverse_backward()

```
void traverse_backward (
    nnode_t * node )
```

Definition at line 74 of file netlist_cleanup.cpp.

Here is the caller graph for this function:

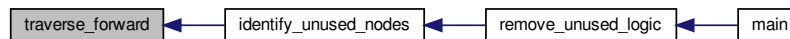


2.32.3.8 traverse_forward()

```
void traverse_forward (  
    nnode_t * node,  
    int toplevel,  
    int remove_me )
```

Definition at line 92 of file netlist_cleanup.cpp.

Here is the caller graph for this function:



2.32.4 Variable Documentation

2.32.4.1 _visited_backward

```
int _visited_backward
```

Definition at line 35 of file netlist_cleanup.cpp.

2.32.4.2 _visited_forward

```
int _visited_forward
```

Definition at line 35 of file netlist_cleanup.cpp.

2.32.4.3 _visited_removal

```
int _visited_removal
```

Definition at line 35 of file netlist_cleanup.cpp.

2.32.4.4 adder_chain_count

```
int adder_chain_count = 0
```

Definition at line 180 of file netlist_cleanup.cpp.

2.32.4.5 addsub_list_next

```
node_list_t* addsub_list_next = &addsub_nodes
```

Definition at line 51 of file netlist_cleanup.cpp.

2.32.4.6 addsub_nodes

```
node_list_t addsub_nodes
```

Definition at line 50 of file netlist_cleanup.cpp.

2.32.4.7 geomean_addsub_length

```
double geomean_addsub_length = 0.0
```

Definition at line 188 of file netlist_cleanup.cpp.

2.32.4.8 longest_adder_chain

```
int longest_adder_chain = 0
```

Definition at line 181 of file netlist_cleanup.cpp.

2.32.4.9 longest_subtractor_chain

```
int longest_subtractor_chain = 0
```

Definition at line 185 of file netlist_cleanup.cpp.

2.32.4.10 removal_list_next

```
node_list_t* removal_list_next = &useless_nodes
```

Definition at line 48 of file netlist_cleanup.cpp.

2.32.4.11 subtractor_chain_count

```
int subtractor_chain_count = 0
```

Definition at line 184 of file netlist_cleanup.cpp.

2.32.4.12 sum_of_addsub_logs

```
double sum_of_addsub_logs = 0.0
```

Definition at line 189 of file netlist_cleanup.cpp.

2.32.4.13 total_adders

```
int total_adders = 0
```

Definition at line 182 of file netlist_cleanup.cpp.

2.32.4.14 total_addsub_chain_count

```
int total_addsub_chain_count = 0
```

Definition at line 190 of file netlist_cleanup.cpp.

2.32.4.15 total_subtractors

```
int total_subtractors = 0
```

Definition at line 186 of file netlist_cleanup.cpp.

2.32.4.16 useless_nodes

`node_list_t` useless_nodes

Definition at line 47 of file netlist_cleanup.cpp.

2.33 vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_cleanup.h File Reference

Functions

- void `remove_unused_logic` (netlist_t *netlist)

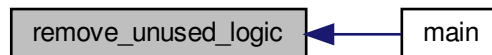
2.33.1 Function Documentation

2.33.1.1 remove_unused_logic()

```
void remove_unused_logic (  
    netlist_t * netlist )
```

Definition at line 232 of file netlist_cleanup.cpp.

Here is the caller graph for this function:



2.34 vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_create_from_ast.cpp File Reference

```
#include <string.h>  
#include <stdio.h>  
#include <stdlib.h>  
#include "types.h"  
#include "globals.h"  
#include "netlist_utils.h"  
#include "odin_util.h"  
#include "ast_util.h"  
#include "netlist_create_from_ast.h"  
#include "string_cache.h"
```

```

#include "netlist_visualizer.h"
#include "parse_making_ast.h"
#include "node_creation_library.h"
#include "multipliers.h"
#include "hard_blocks.h"
#include "memories.h"
#include "implicit_memory.h"
#include "adders.h"
#include "subtractions.h"
#include "ast_elaborate.h"
#include "vtr_util.h"
#include "vtr_memory.h"

```

Macros

- `#define INSTANTIATE_DRIVERS 1`
- `#define ALIAS_INPUTS 2`
- `#define COMBINATIONAL 1`
- `#define SEQUENTIAL 2`

Functions

- void `create_param_table_for_module` (ast_node_t *parent_parameter_list, ast_node_t *module_items, char *module_name)
- void `convert_ast_to_netlist_recurring_via_modules` (ast_node_t *current_module, char *instance_name, int level)
- signal_list_t * `netlist_expand_ast_of_module` (ast_node_t *node, char *instance_name_prefix)
- void `create_all_driver_nets_in_this_module` (char *instance_name_prefix)
- void `create_all_driver_nets_in_this_function` (char *instance_name_prefix)
- void `create_top_driver_nets` (ast_node_t *module, char *instance_name_prefix)
- void `create_top_output_nodes` (ast_node_t *module, char *instance_name_prefix)
- nnet_t * `define_nets_with_driver` (ast_node_t *var_declare, char *instance_name_prefix)
- nnet_t * `define_nodes_and_nets_with_driver` (ast_node_t *var_declare, char *instance_name_prefix)
- void `connect_hard_block_and_alias` (ast_node_t *module_instance, char *instance_name_prefix, int output_size)
- void `connect_module_instantiation_and_alias` (short PASS, ast_node_t *module_instance, char *instance_name_prefix)
- signal_list_t * `connect_function_instantiation_and_alias` (short PASS, ast_node_t *module_instance, char *instance_name_prefix)
- void `create_symbol_table_for_module` (ast_node_t *module_items, char *module_name)
- void `create_symbol_table_for_function` (ast_node_t *module_items, char *module_name)
- int `check_for_initial_reg_value` (ast_node_t *var_declare, long long *value)
- void `define_latches_initial_value_inside_initial_statement` (ast_node_t *initial_node, char *instance_name_prefix)
- signal_list_t * `concatenate_signal_lists` (signal_list_t **signal_lists, int num_signal_lists)
- signal_list_t * `create_gate` (ast_node_t *gate, char *instance_name_prefix)
- signal_list_t * `create_hard_block` (ast_node_t *block, char *instance_name_prefix)
- signal_list_t * `create_pins` (ast_node_t *var_declare, char *name, char *instance_name_prefix)
- signal_list_t * `create_output_pin` (ast_node_t *var_declare, char *instance_name_prefix)
- signal_list_t * `assignment_alias` (ast_node_t *assignment, char *instance_name_prefix)

- `signal_list_t * create_operation_node` (`ast_node_t *op`, `signal_list_t **input_lists`, `int list_size`, `char *instance_name_prefix`)
- `void terminate_continuous_assignment` (`ast_node_t *node`, `signal_list_t *assignment`, `char *instance_name_prefix`)
- `void terminate_registered_assignment` (`ast_node_t *always_node`, `signal_list_t *assignment`, `signal_list_t *potential_clocks`, `char *instance_name_prefix`)
- `signal_list_t * create_case` (`ast_node_t *case_ast`, `char *instance_name_prefix`)
- `void create_case_control_signals` (`ast_node_t *case_list_of_items`, `ast_node_t *compare_against`, `nnode_t *case_node`, `char *instance_name_prefix`)
- `signal_list_t * create_case_mux_statements` (`ast_node_t *case_list_of_items`, `nnode_t *case_node`, `char *instance_name_prefix`)
- `signal_list_t * create_if` (`ast_node_t *if_ast`, `char *instance_name_prefix`)
- `void create_if_control_signals` (`ast_node_t *if_expression`, `nnode_t *if_node`, `char *instance_name_prefix`)
- `signal_list_t * create_if_mux_statements` (`ast_node_t *if_ast`, `nnode_t *if_node`, `char *instance_name_prefix`)
- `signal_list_t * create_if_for_question` (`ast_node_t *if_ast`, `char *instance_name_prefix`)
- `signal_list_t * create_if_question_mux_expressions` (`ast_node_t *if_ast`, `nnode_t *if_node`, `char *instance_name_prefix`)
- `signal_list_t * create_mux_statements` (`signal_list_t **statement_lists`, `nnode_t *case_node`, `int num_statement_lists`, `char *instance_name_prefix`)
- `signal_list_t * create_mux_expressions` (`signal_list_t **expression_lists`, `nnode_t *mux_node`, `int num_expression_lists`, `char *instance_name_prefix`)
- `signal_list_t * evaluate_sensitivity_list` (`ast_node_t *delay_control`, `char *instance_name_prefix`)
- `int alias_output_assign_pins_to_inputs` (`char_list_t *output_list`, `signal_list_t *input_list`, `ast_node_t *node`)
- `int find_smallest_non_numerical` (`ast_node_t *node`, `signal_list_t **input_list`, `int num_input_lists`)
- `void pad_with_zeros` (`ast_node_t *node`, `signal_list_t *list`, `int pad_size`, `char *instance_name_prefix`)
- `signal_list_t * create_dual_port_ram_block` (`ast_node_t *block`, `char *instance_name_prefix`, `t_model *hb_model`)
- `signal_list_t * create_single_port_ram_block` (`ast_node_t *block`, `char *instance_name_prefix`, `t_model *hb_model`)
- `signal_list_t * create_soft_single_port_ram_block` (`ast_node_t *block`, `char *instance_name_prefix`)
- `signal_list_t * create_soft_dual_port_ram_block` (`ast_node_t *block`, `char *instance_name_prefix`)
- `void look_for_clocks` (`netlist_t *netlist`)
- `void create_netlist` ()
- `ast_node_t * find_top_module` ()
- `void connect_memory_and_alias` (`ast_node_t *hb_instance`, `char *instance_name_prefix`)

Variables

- `STRING_CACHE * output_nets_sc`
- `STRING_CACHE * input_nets_sc`
- `STRING_CACHE * local_symbol_table_sc`
- `STRING_CACHE * function_local_symbol_table_sc`
- `STRING_CACHE * global_param_table_sc`
- `ast_node_t ** local_symbol_table`
- `ast_node_t ** function_local_symbol_table`
- `int num_local_symbol_table`
- `int function_num_local_symbol_table`
- `signal_list_t * local_clock_list`
- `short local_clock_found`
- `int local_clock_idx`

- char * [one_string](#)
- char * [zero_string](#)
- char * [pad_string](#)
- ast_node_t * [top_module](#)
- netlist_t * [verilog_netlist](#)
- int [netlist_create_line_number](#) = -2
- int [type_of_circuit](#)

2.34.1 Macro Definition Documentation

2.34.1.1 ALIAS_INPUTS

```
#define ALIAS_INPUTS 2
```

Definition at line 55 of file netlist_create_from_ast.cpp.

2.34.1.2 COMBINATIONAL

```
#define COMBINATIONAL 1
```

Definition at line 57 of file netlist_create_from_ast.cpp.

2.34.1.3 INSTANTIATE_DRIVERS

```
#define INSTANTIATE_DRIVERS 1
```

Definition at line 54 of file netlist_create_from_ast.cpp.

2.34.1.4 SEQUENTIAL

```
#define SEQUENTIAL 2
```

Definition at line 58 of file netlist_create_from_ast.cpp.

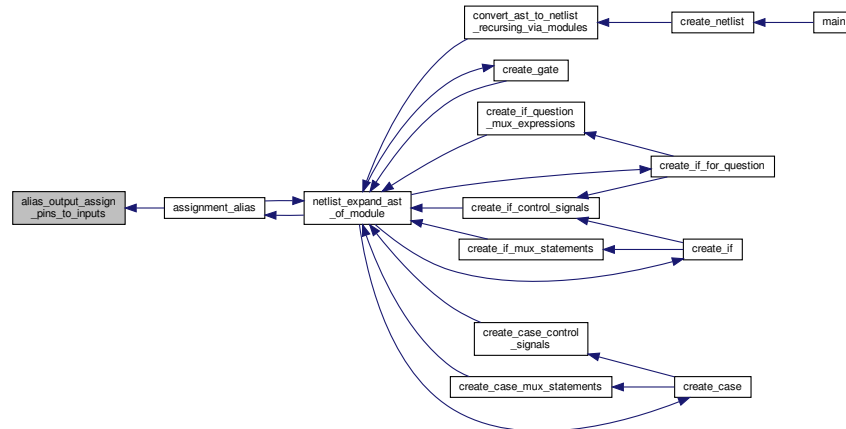
2.34.2 Function Documentation

2.34.2.1 alias_output_assign_pins_to_inputs()

```
int alias_output_assign_pins_to_inputs (
    char_list_t * output_list,
    signal_list_t * input_list,
    ast_node_t * node )
```

Definition at line 3484 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:

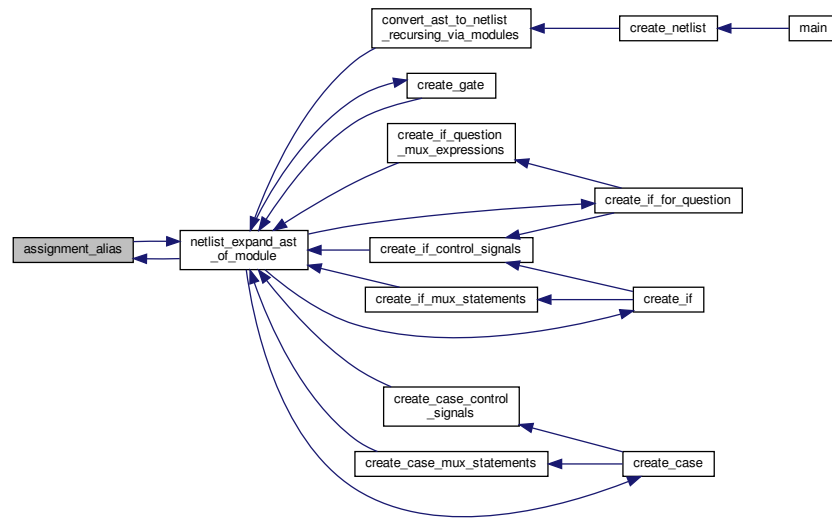


2.34.2.2 assignment_alias()

```
signal_list_t * assignment_alias (
    ast_node_t * assignment,
    char * instance_name_prefix )
```

Definition at line 2880 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



2.34.2.3 check_for_initial_reg_value()

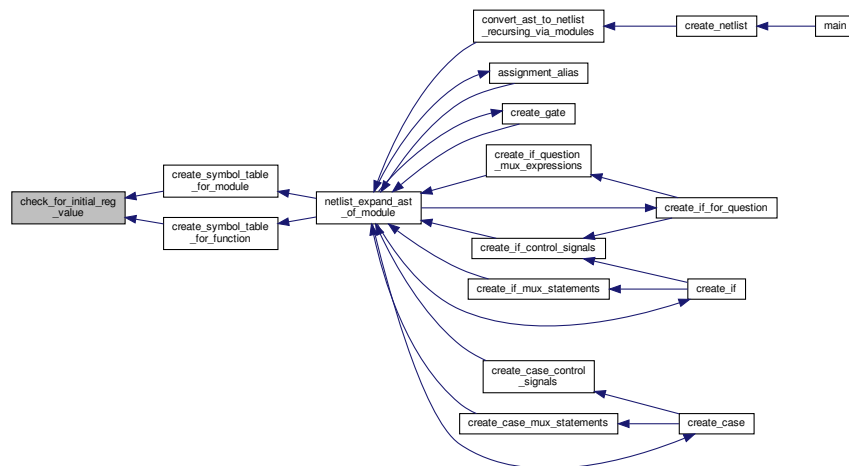
```

int check_for_initial_reg_value (
    ast_node_t * var_declare,
    long long * value )

```

Definition at line 1817 of file `netlist_create_from_ast.cpp`.

Here is the caller graph for this function:

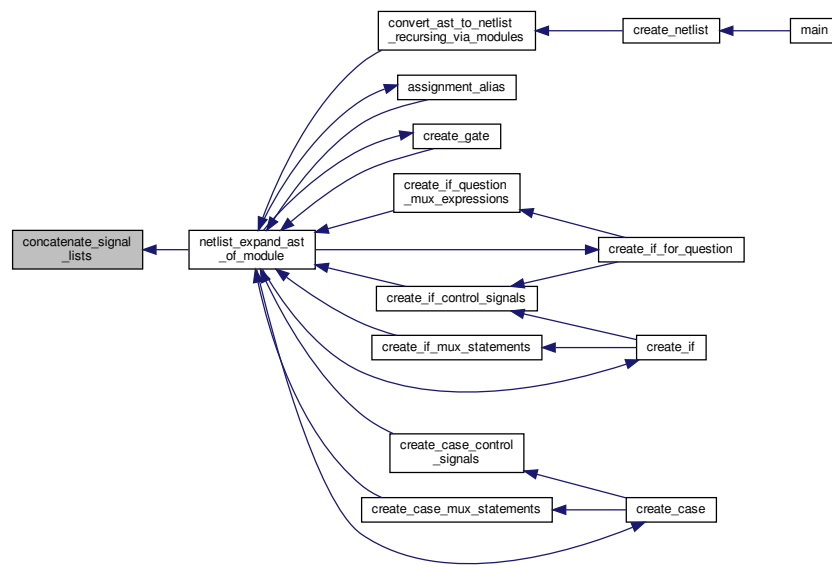


2.34.2.4 concatenate_signal_lists()

```
signal_list_t * concatenate_signal_lists (
    signal_list_t ** signal_lists,
    int num_signal_lists )
```

Definition at line 976 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:

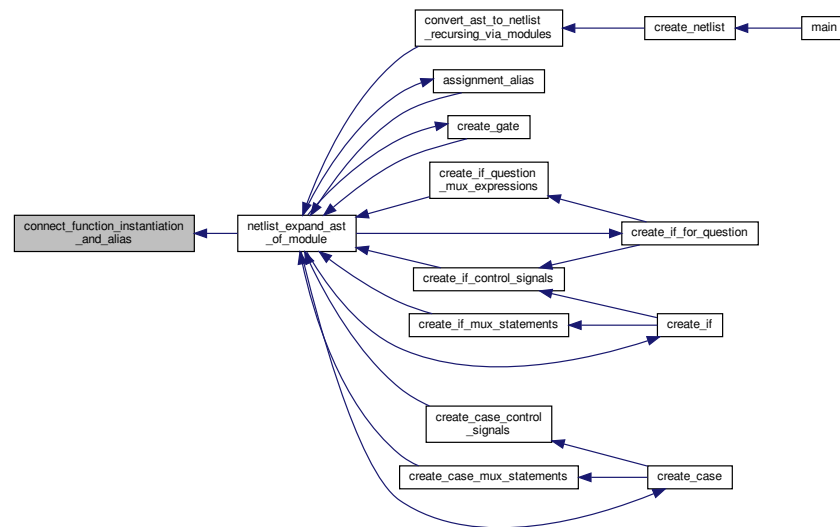


2.34.2.5 connect_function_instantiation_and_alias()

```
signal_list_t * connect_function_instantiation_and_alias (
    short PASS,
    ast_node_t * module_instance,
    char * instance_name_prefix )
```

Definition at line 2423 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



2.34.2.6 connect_hard_block_and_alias()

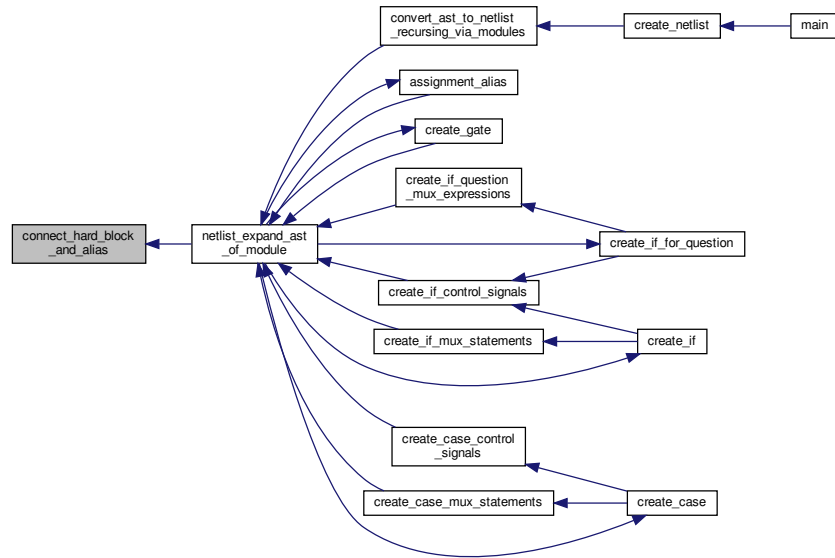
```

void connect_hard_block_and_alias (
    ast_node_t * module_instance,
    char * instance_name_prefix,
    int outport_size )

```

Definition at line 1974 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



2.34.2.7 connect_memory_and_alias()

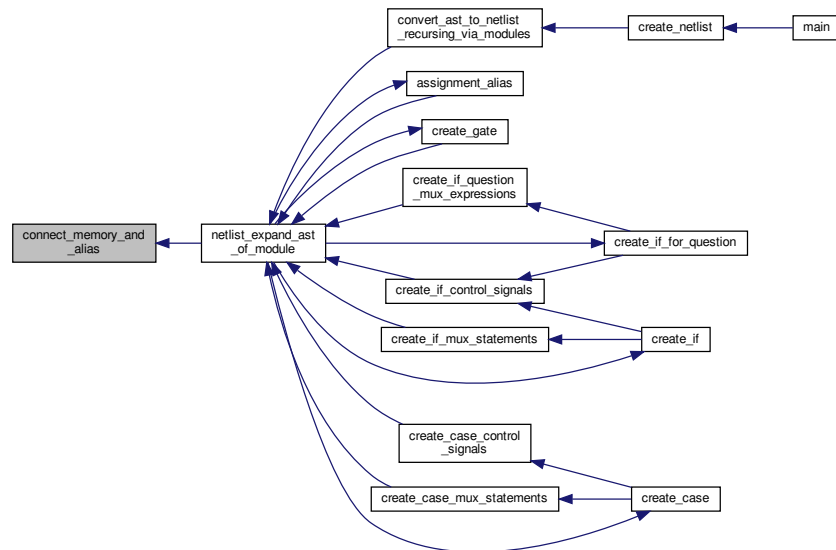
```

void connect_memory_and_alias (
    ast_node_t * hb_instance,
    char * instance_name_prefix )

```

Definition at line 1836 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



2.34.2.8 connect_module_instantiation_and_alias()

```

void connect_module_instantiation_and_alias (
    short PASS,
    ast_node_t * module_instance,
    char * instance_name_prefix )

```

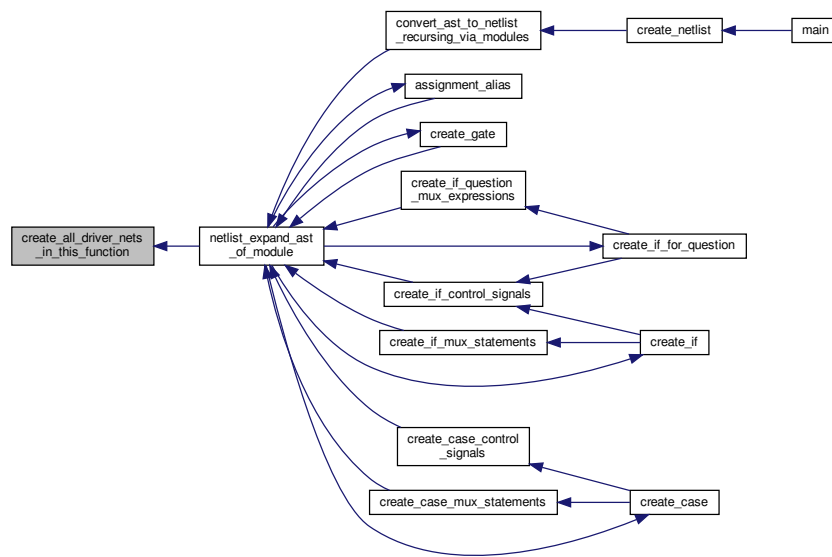
Definition at line 2127 of file `netlist_create_from_ast.cpp`.

2.34.2.10 create_all_driver_nets_in_this_function()

```
void create_all_driver_nets_in_this_function (
    char * instance_name_prefix )
```

Definition at line 1043 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:

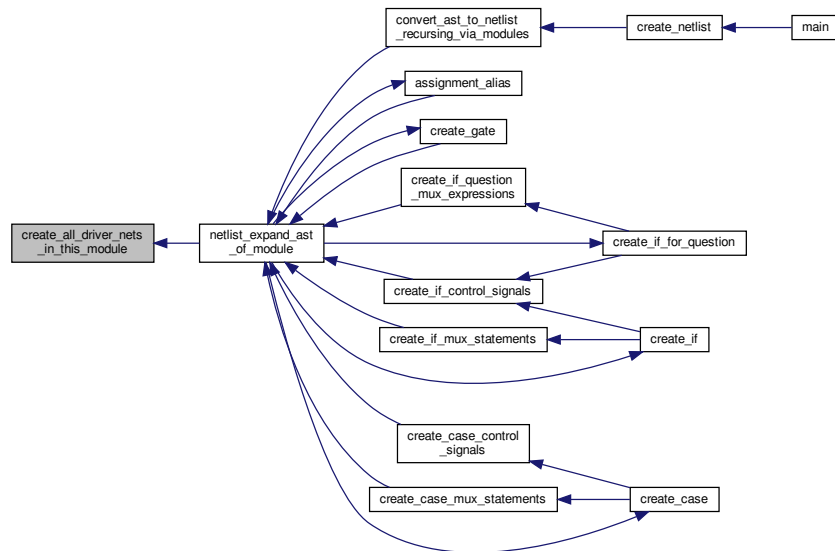


2.34.2.11 create_all_driver_nets_in_this_module()

```
void create_all_driver_nets_in_this_module (
    char * instance_name_prefix )
```

Definition at line 1005 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



2.34.2.12 create_case()

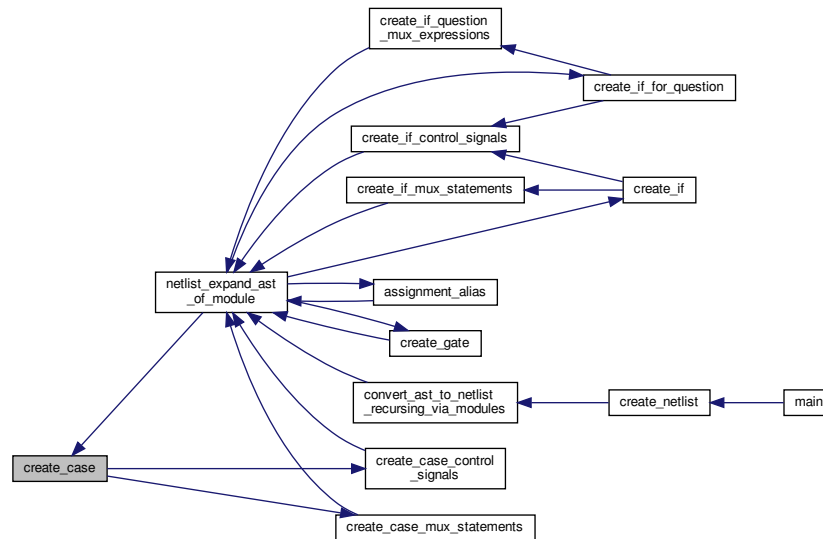
```

signal_list_t * create_case (
    ast_node_t * case_ast,
    char * instance_name_prefix )

```

Definition at line 4093 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



2.34.2.13 create_case_control_signals()

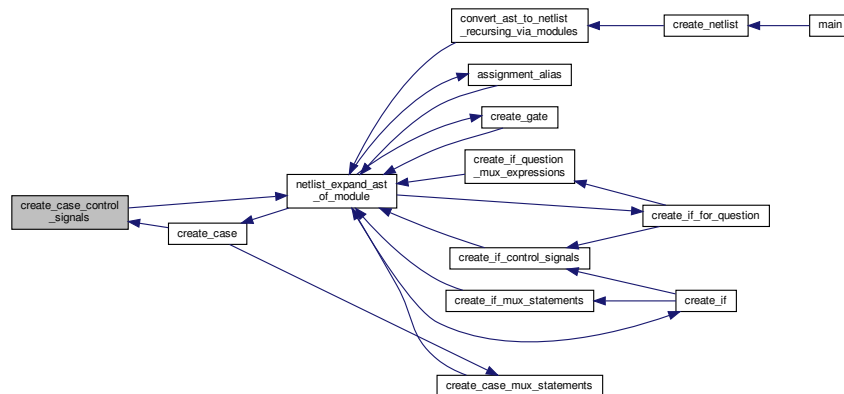
```

void create_case_control_signals (
    ast_node_t * case_list_of_items,
    ast_node_t * compare_against,
    nnode_t * case_node,
    char * instance_name_prefix )

```

Definition at line 4125 of file `netlist_create_from_ast.cpp`.

Here is the caller graph for this function:

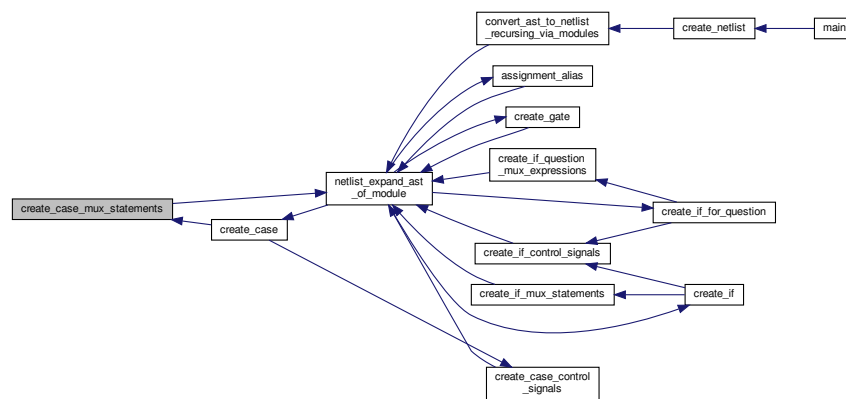


2.34.2.14 create_case_mux_statements()

```
signal_list_t * create_case_mux_statements (
    ast_node_t * case_list_of_items,
    nnode_t * case_node,
    char * instance_name_prefix )
```

Definition at line 4193 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:

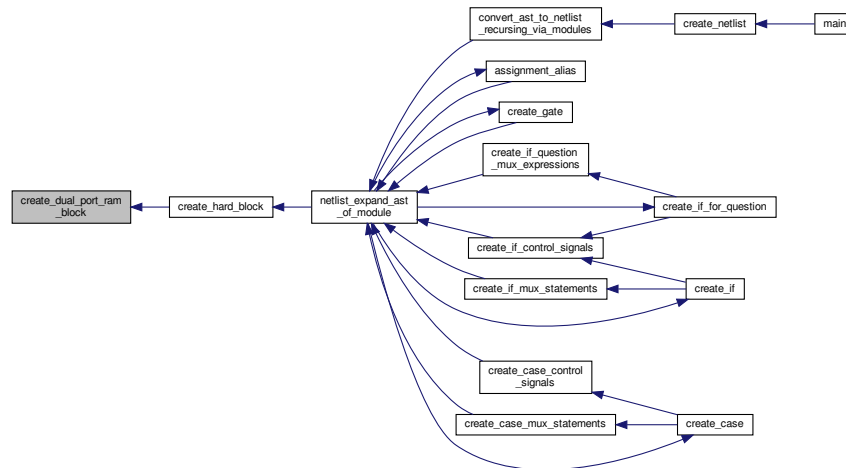


2.34.2.15 create_dual_port_ram_block()

```
signal_list_t * create_dual_port_ram_block (
    ast_node_t * block,
    char * instance_name_prefix,
    t_model * hb_model )
```

Definition at line 4511 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



2.34.2.16 create_gate()

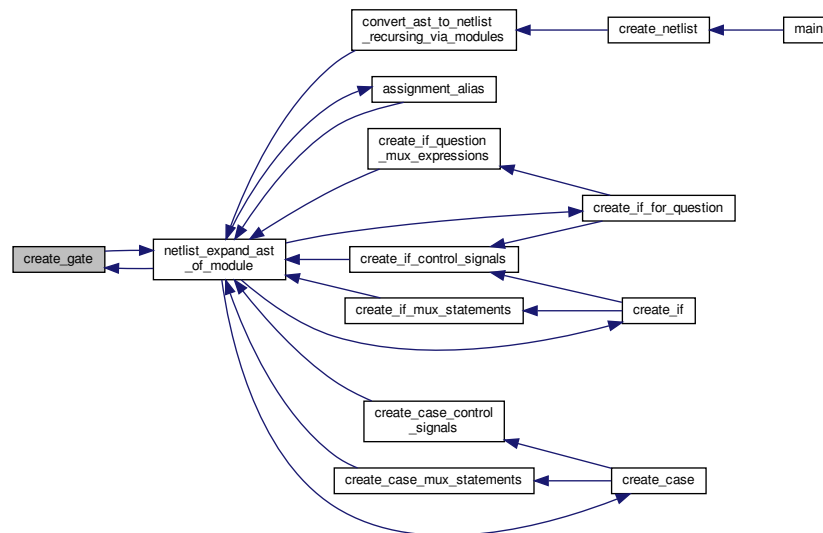
```

signal_list_t * create_gate (
    ast_node_t * gate,
    char * instance_name_prefix )

```

Definition at line 3514 of file `netlist_create_from_ast.cpp`.

Here is the caller graph for this function:

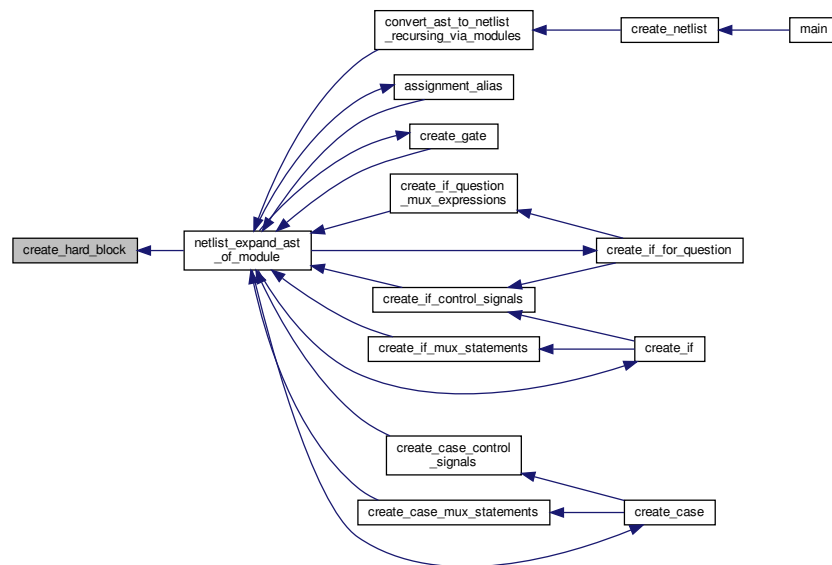


2.34.2.17 create_hard_block()

```
signal_list_t * create_hard_block (
    ast_node_t * block,
    char * instance_name_prefix )
```

Definition at line 5182 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:

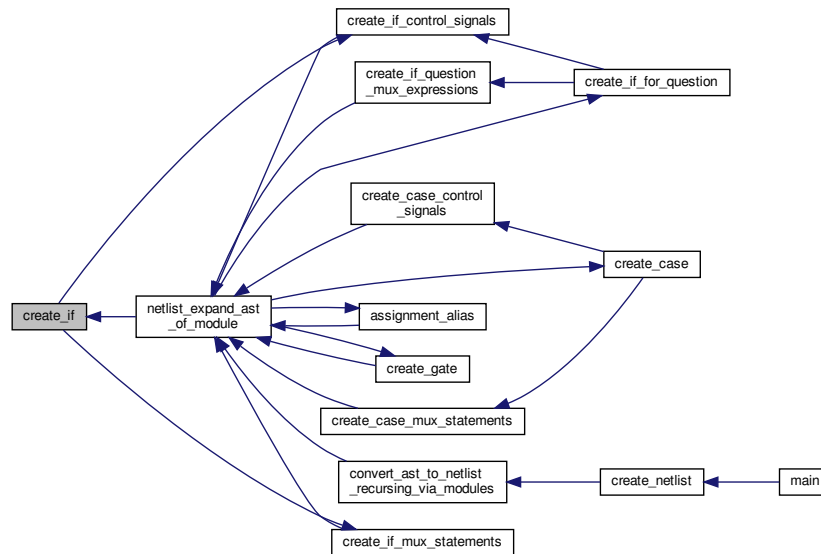


2.34.2.18 create_if()

```
signal_list_t * create_if (
    ast_node_t * if_ast,
    char * instance_name_prefix )
```

Definition at line 3975 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



2.34.2.19 create_if_control_signals()

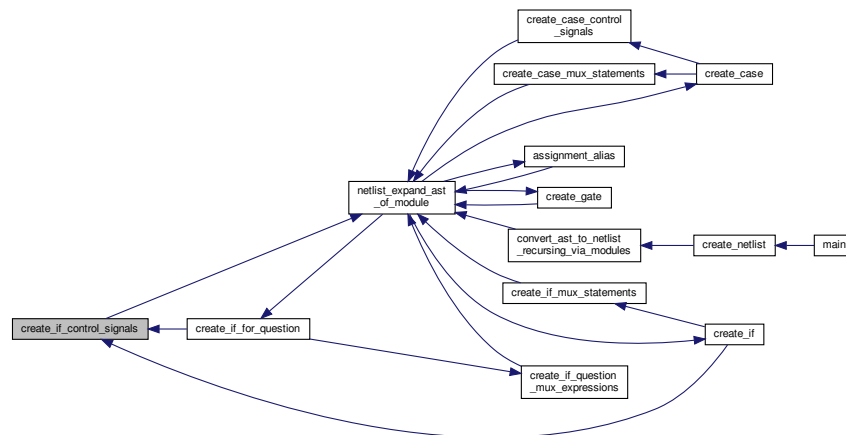
```

void create_if_control_signals (
    ast_node_t * if_expression,
    nnode_t * if_node,
    char * instance_name_prefix )

```

Definition at line 3999 of file `netlist_create_from_ast.cpp`.

Here is the caller graph for this function:

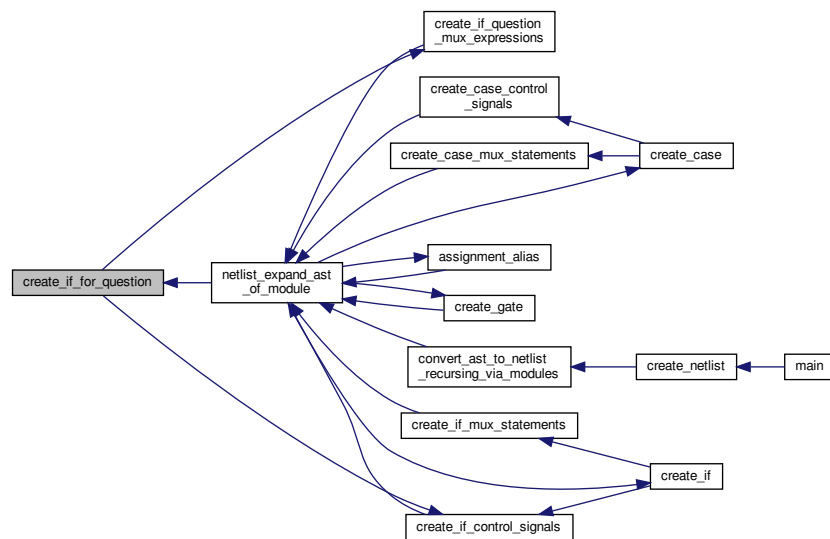


2.34.2.20 create_if_for_question()

```
signal_list_t * create_if_for_question (
    ast_node_t * if_ast,
    char * instance_name_prefix )
```

Definition at line 3919 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:

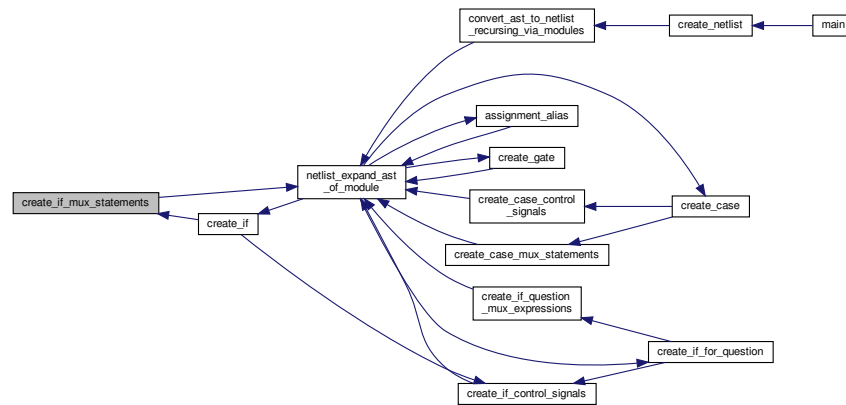


2.34.2.21 create_if_mux_statements()

```
signal_list_t * create_if_mux_statements (
    ast_node_t * if_ast,
    nnode_t * if_node,
    char * instance_name_prefix )
```

Definition at line 4059 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



2.34.2.22 create_if_question_mux_expressions()

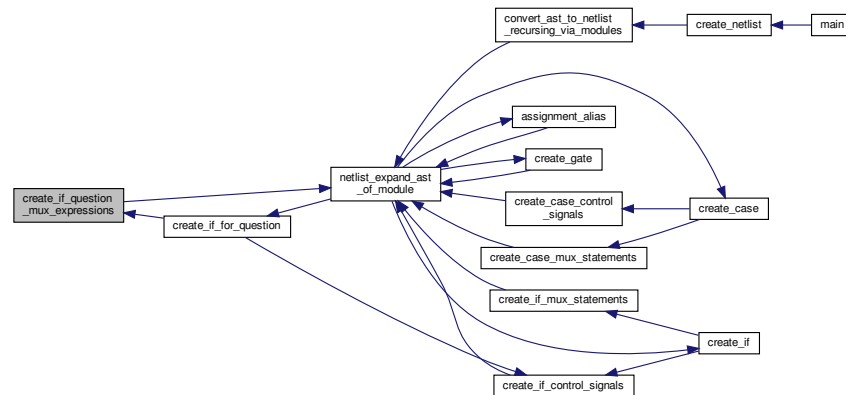
```

signal_list_t * create_if_question_mux_expressions (
    ast_node_t * if_ast,
    nnode_t * if_node,
    char * instance_name_prefix )

```

Definition at line 3943 of file `netlist_create_from_ast.cpp`.

Here is the caller graph for this function:

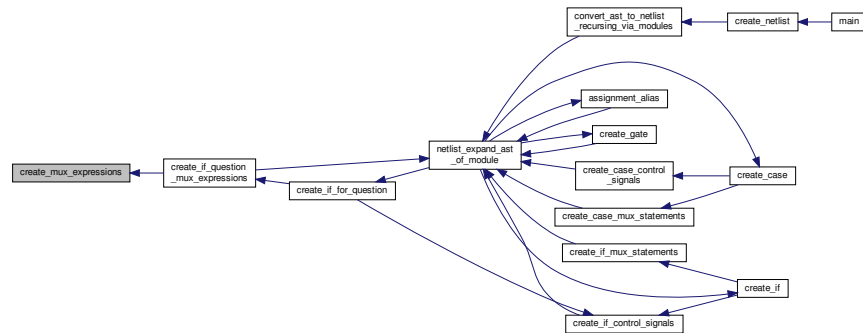


2.34.2.23 create_mux_expressions()

```
signal_list_t * create_mux_expressions (
    signal_list_t ** expression_lists,
    nnode_t * mux_node,
    int num_expression_lists,
    char * instance_name_prefix )
```

Definition at line 4346 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:

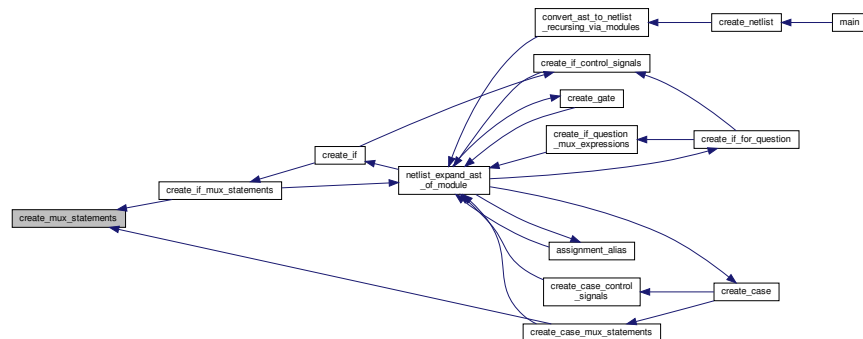


2.34.2.24 create_mux_statements()

```
signal_list_t * create_mux_statements (
    signal_list_t ** statement_lists,
    nnode_t * case_node,
    int num_statement_lists,
    char * instance_name_prefix )
```

Definition at line 4232 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



2.34.2.25 create_netlist()

```
void create_netlist ( )
```

Definition at line 319 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:

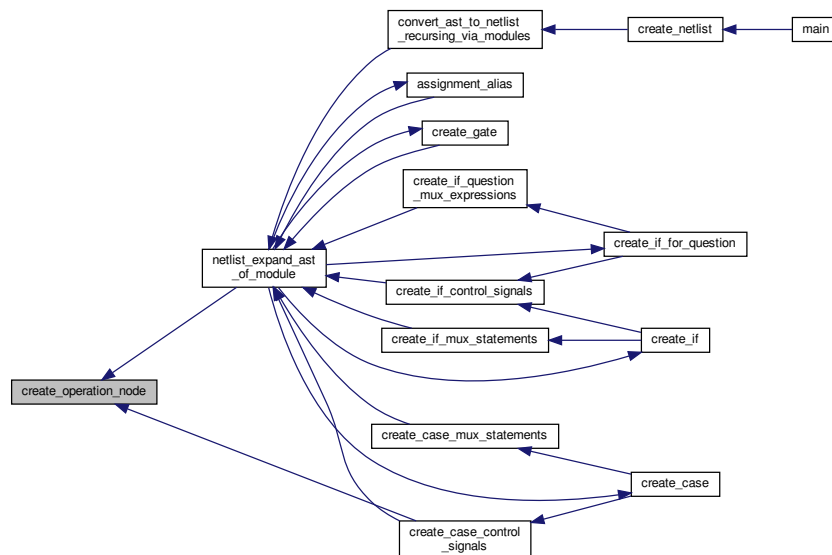


2.34.2.26 create_operation_node()

```
signal_list_t * create_operation_node (
    ast_node_t * op,
    signal_list_t ** input_lists,
    int list_size,
    char * instance_name_prefix )
```

Definition at line 3623 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:

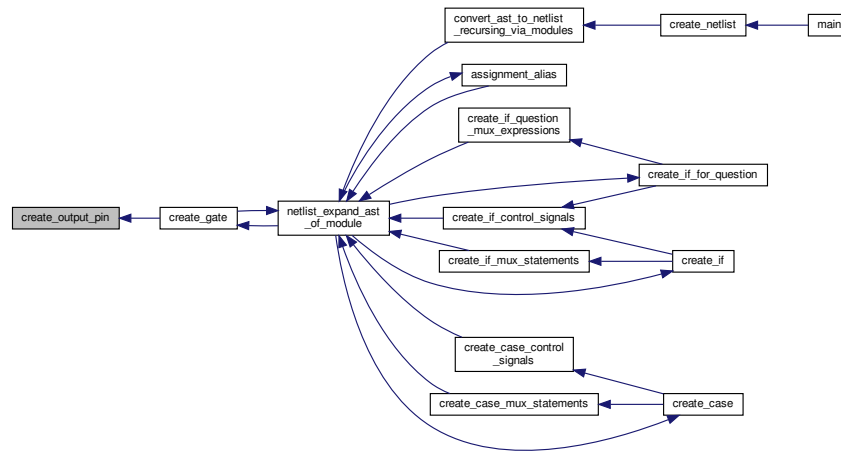


2.34.2.27 create_output_pin()

```
signal_list_t * create_output_pin (
    ast_node_t * var_declare,
    char * instance_name_prefix )
```

Definition at line 2857 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



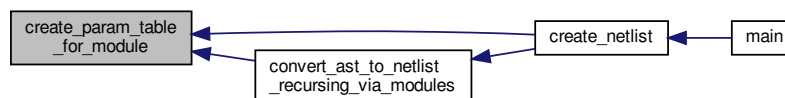
2.34.2.28 create_param_table_for_module()

```
void create_param_table_for_module (
    ast_node_t * parent_parameter_list,
    ast_node_t * module_items,
    char * module_name )
```

Scan through all VAR_DECLARE_LISTs in MODULE_ITEMS and create a hash table for all parameters in this instantiation, taking into account if they are being overridden by their parent

Definition at line 155 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



2.34.2.29 create_pins()

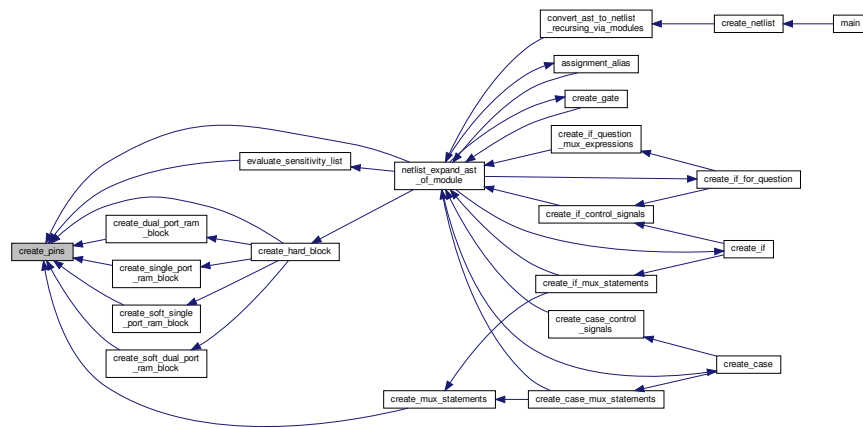
```

signal_list_t * create_pins (
    ast_node_t * var_declare,
    char * name,
    char * instance_name_prefix )

```

Definition at line 2758 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



2.34.2.30 create_single_port_ram_block()

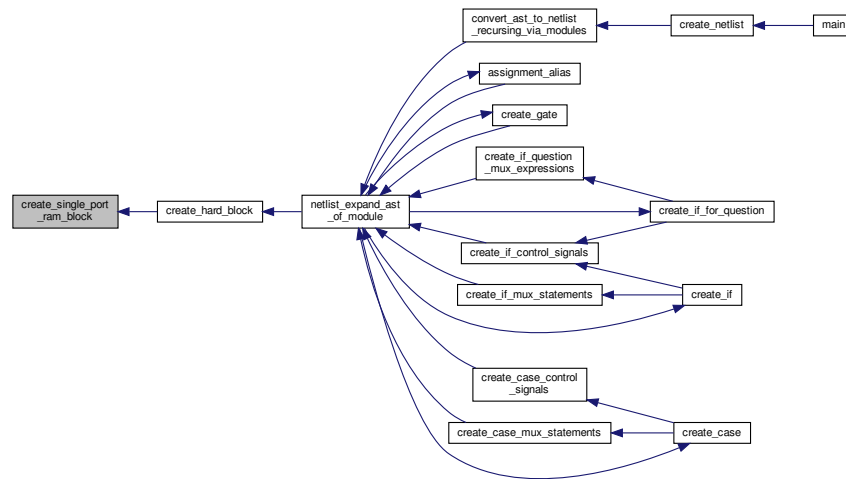
```

signal_list_t * create_single_port_ram_block (
    ast_node_t * block,
    char * instance_name_prefix,
    t_model * hb_model )

```

Definition at line 4686 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



2.34.2.31 create_soft_dual_port_ram_block()

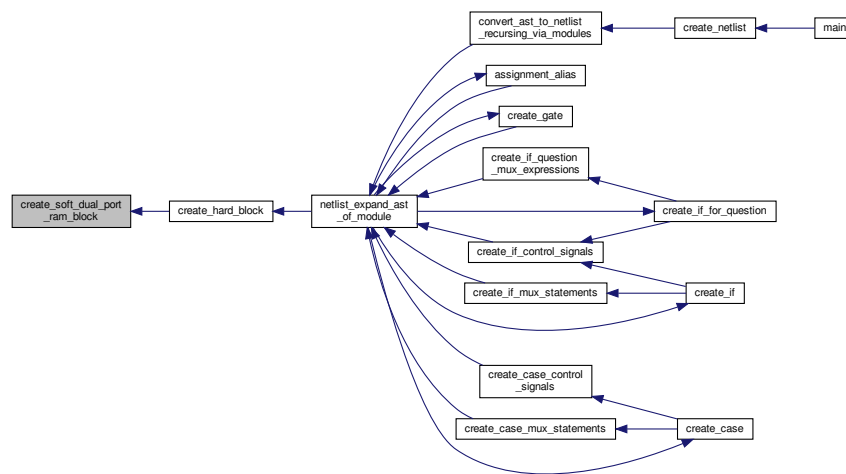
```

signal_list_t * create_soft_dual_port_ram_block (
    ast_node_t * block,
    char * instance_name_prefix )

```

Definition at line 5024 of file `netlist_create_from_ast.cpp`.

Here is the caller graph for this function:

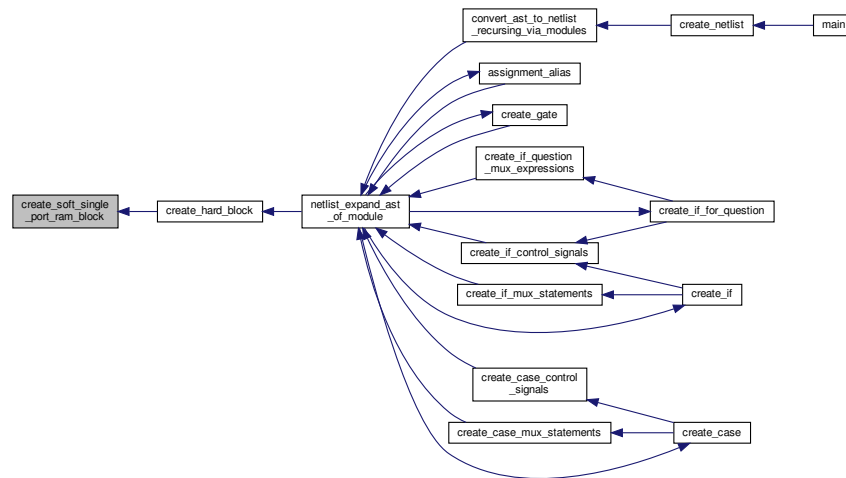


2.34.2.32 create_soft_single_port_ram_block()

```
signal_list_t * create_soft_single_port_ram_block (
    ast_node_t * block,
    char * instance_name_prefix )
```

Definition at line 4874 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:

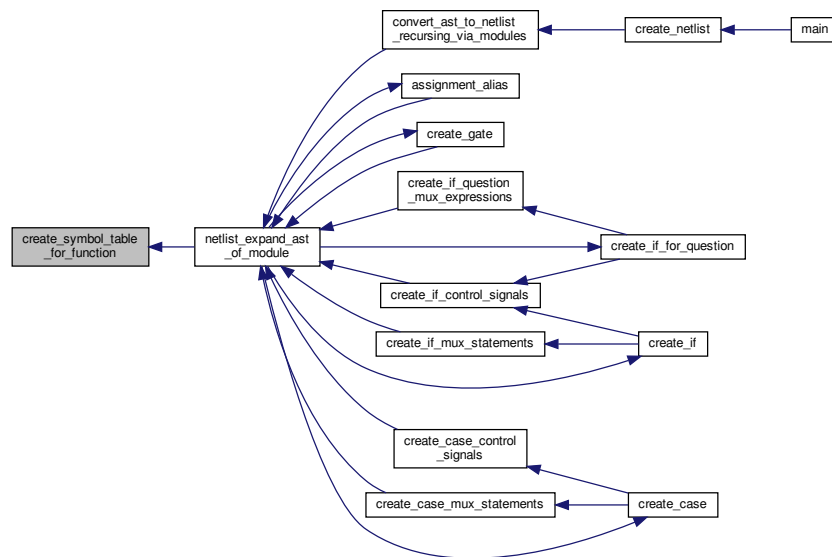


2.34.2.33 create_symbol_table_for_function()

```
void create_symbol_table_for_function (
    ast_node_t * module_items,
    char * module_name )
```

Definition at line 1699 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



2.34.2.34 create_symbol_table_for_module()

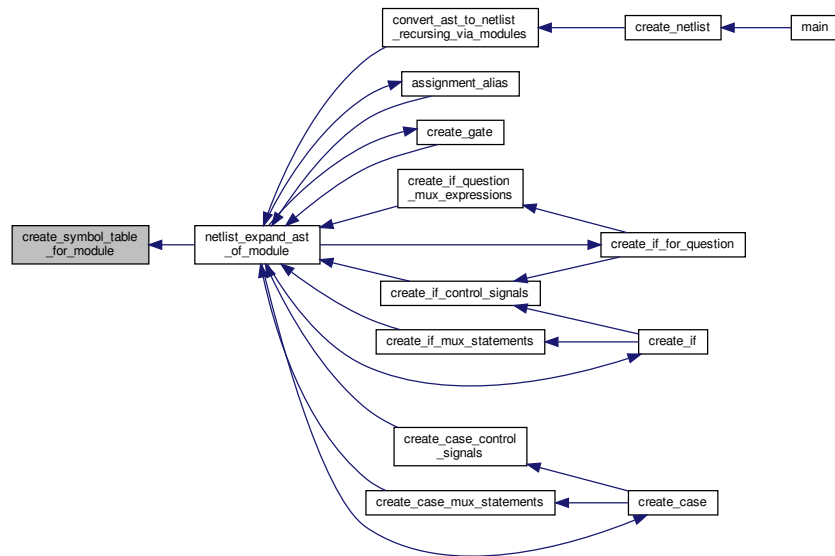
```

void create_symbol_table_for_module (
    ast_node_t * module_items,
    char * module_name )

```

Definition at line 1550 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



2.34.2.35 create_top_driver_nets()

```

void create_top_driver_nets (
    ast_node_t * module,
    char * instance_name_prefix )

```

Definition at line 1081 of file `netlist_create_from_ast.cpp`.

Here is the caller graph for this function:



2.34.2.36 create_top_output_nodes()

```
void create_top_output_nodes (
    ast_node_t * module,
    char * instance_name_prefix )
```

Definition at line 1191 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:

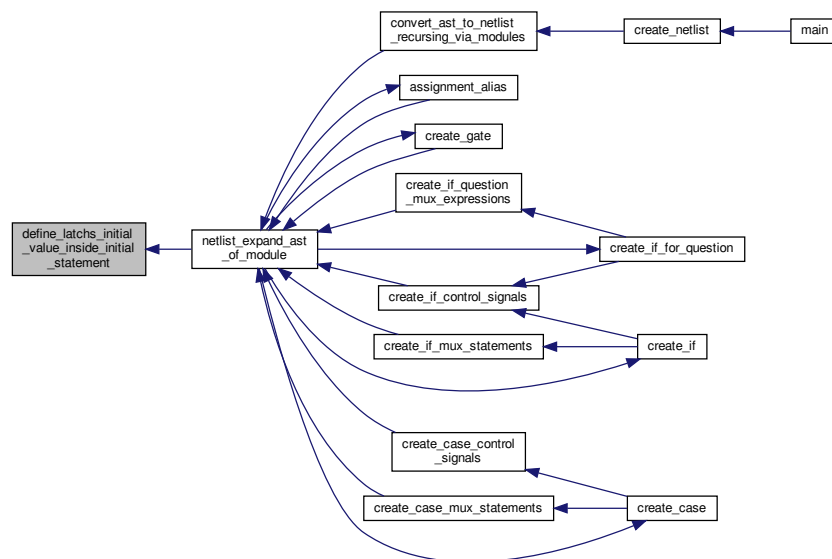


2.34.2.37 define_latches_initial_value_inside_initial_statement()

```
void define_latches_initial_value_inside_initial_statement (
    ast_node_t * initial_node,
    char * instance_name_prefix )
```

Definition at line 3149 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:

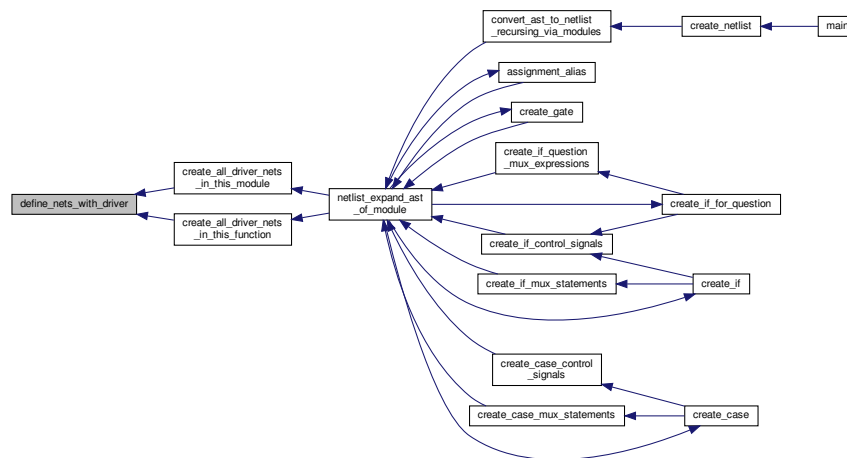


2.34.2.38 define_nets_with_driver()

```
nnet_t * define_nets_with_driver (
    ast_node_t * var_declare,
    char * instance_name_prefix )
```

Definition at line 1313 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:

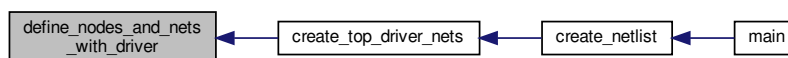


2.34.2.39 define_nodes_and_nets_with_driver()

```
nnet_t * define_nodes_and_nets_with_driver (
    ast_node_t * var_declare,
    char * instance_name_prefix )
```

Definition at line 1440 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:

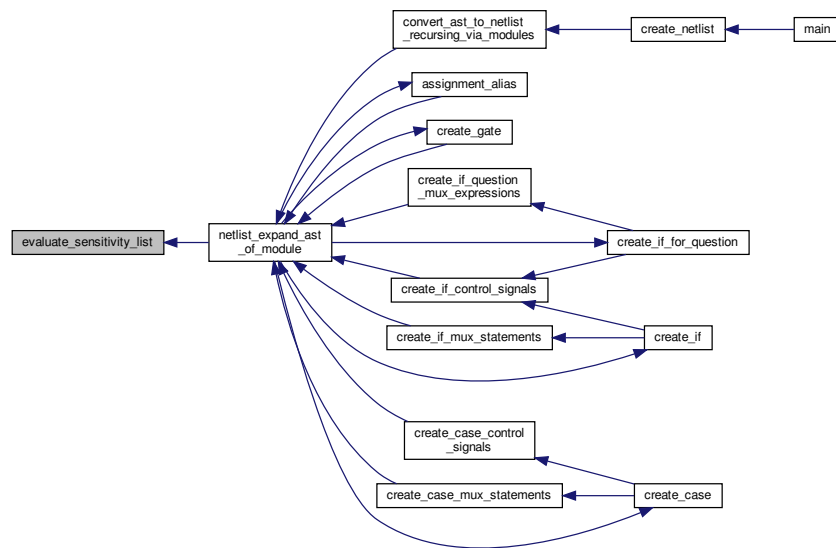


2.34.2.40 evaluate_sensitivity_list()

```
signal_list_t * evaluate_sensitivity_list (
    ast_node_t * delay_control,
    char * instance_name_prefix )
```

Definition at line 3852 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:

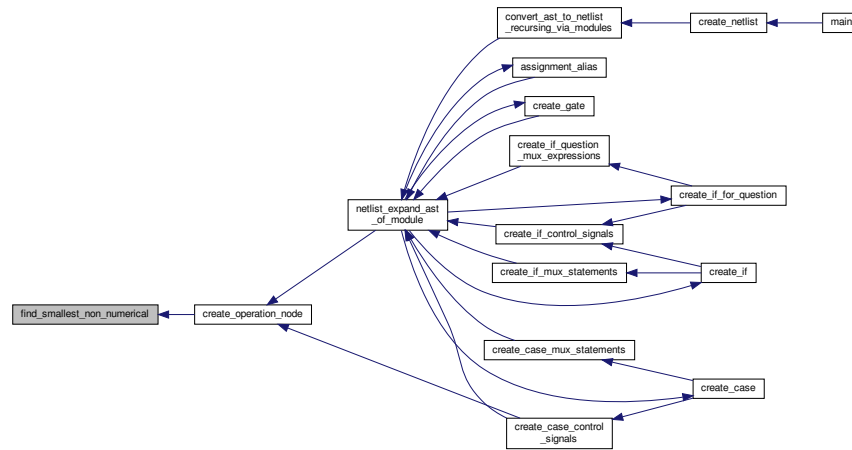


2.34.2.41 find_smallest_non_numerical()

```
int find_smallest_non_numerical (
    ast_node_t * node,
    signal_list_t ** input_list,
    int num_input_lists )
```

Definition at line 4421 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:

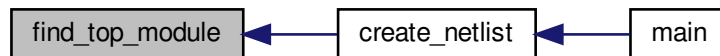


2.34.2.42 find_top_module()

```
ast_node_t* find_top_module ( )
```

Definition at line 419 of file `netlist_create_from_ast.cpp`.

Here is the caller graph for this function:

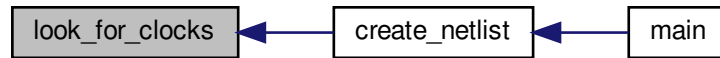


2.34.2.43 look_for_clocks()

```
void look_for_clocks (
    netlist_t * netlist )
```

Definition at line 401 of file `netlist_create_from_ast.cpp`.

Here is the caller graph for this function:



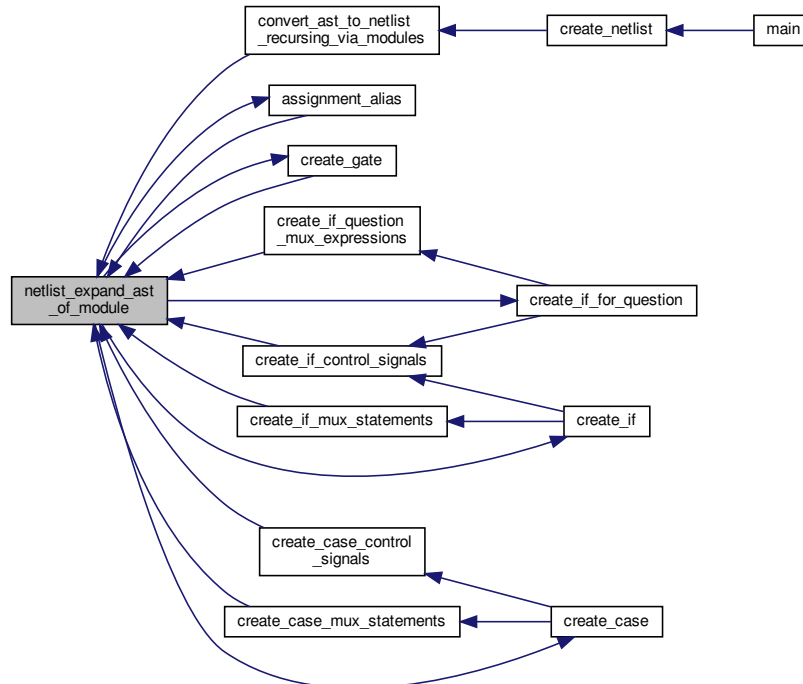
2.34.2.44 netlist_expand_ast_of_module()

```

signal_list_t * netlist_expand_ast_of_module (
    ast_node_t * node,
    char * instance_name_prefix )
  
```

Definition at line 618 of file `netlist_create_from_ast.cpp`.

Here is the caller graph for this function:

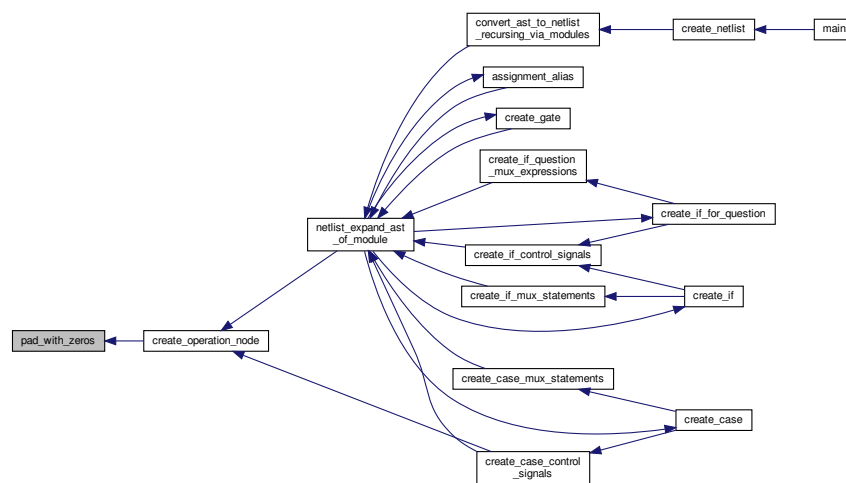


2.34.2.45 pad_with_zeros()

```
void pad_with_zeros (
    ast_node_t * node,
    signal_list_t * list,
    int pad_size,
    char * instance_name_prefix )
```

Definition at line 4484 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:

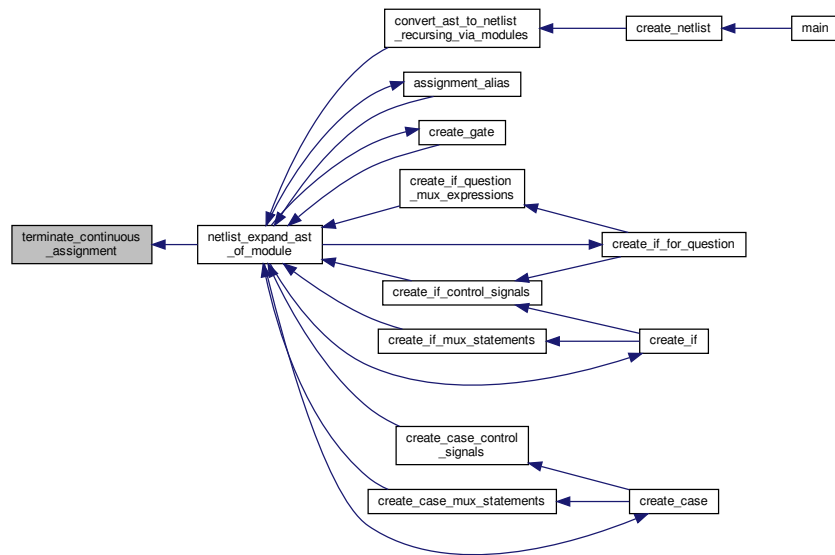


2.34.2.46 terminate_continuous_assignment()

```
void terminate_continuous_assignment (
    ast_node_t * node,
    signal_list_t * assignment,
    char * instance_name_prefix )
```

Definition at line 3401 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



2.34.2.47 terminate_registered_assignment()

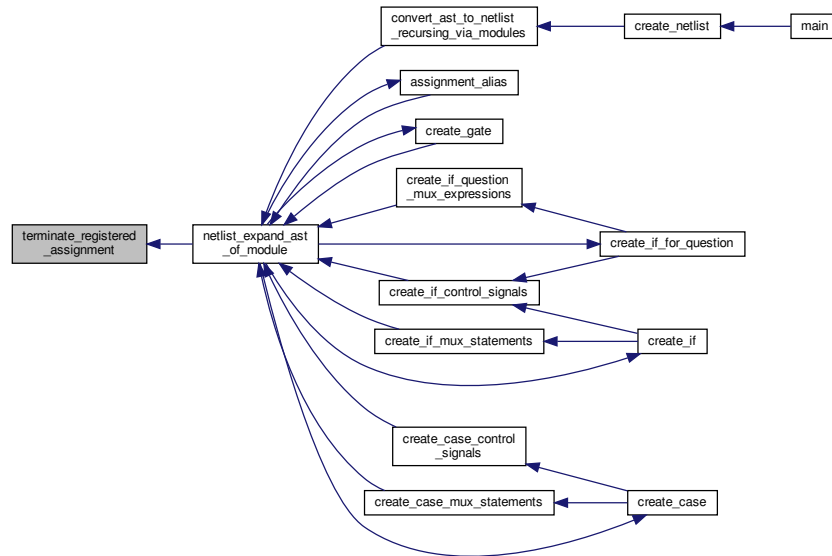
```

void terminate_registered_assignment (
    ast_node_t * always_node,
    signal_list_t * assignment,
    signal_list_t * potential_clocks,
    char * instance_name_prefix )

```

Definition at line 3185 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



2.34.3 Variable Documentation

2.34.3.1 function_local_symbol_table

```
ast_node_t** function_local_symbol_table
```

Definition at line 67 of file `netlist_create_from_ast.cpp`.

2.34.3.2 function_local_symbol_table_sc

```
STRING_CACHE* function_local_symbol_table_sc
```

Definition at line 64 of file `netlist_create_from_ast.cpp`.

2.34.3.3 function_num_local_symbol_table

```
int function_num_local_symbol_table
```

Definition at line 69 of file `netlist_create_from_ast.cpp`.

2.34.3.4 global_param_table_sc

```
STRING_CACHE* global_param_table_sc
```

Definition at line 65 of file netlist_create_from_ast.cpp.

2.34.3.5 input_nets_sc

```
STRING_CACHE* input_nets_sc
```

Definition at line 61 of file netlist_create_from_ast.cpp.

2.34.3.6 local_clock_found

```
short local_clock_found
```

Definition at line 71 of file netlist_create_from_ast.cpp.

2.34.3.7 local_clock_idx

```
int local_clock_idx
```

Definition at line 72 of file netlist_create_from_ast.cpp.

2.34.3.8 local_clock_list

```
signal_list_t* local_clock_list
```

Definition at line 70 of file netlist_create_from_ast.cpp.

2.34.3.9 local_symbol_table

```
ast_node_t** local_symbol_table
```

Definition at line 66 of file netlist_create_from_ast.cpp.

2.34.3.10 local_symbol_table_sc

```
STRING_CACHE* local_symbol_table_sc
```

Definition at line 63 of file netlist_create_from_ast.cpp.

2.34.3.11 netlist_create_line_number

```
int netlist_create_line_number = -2
```

Definition at line 83 of file netlist_create_from_ast.cpp.

2.34.3.12 num_local_symbol_table

```
int num_local_symbol_table
```

Definition at line 68 of file netlist_create_from_ast.cpp.

2.34.3.13 one_string

```
char* one_string
```

Definition at line 75 of file netlist_create_from_ast.cpp.

2.34.3.14 output_nets_sc

```
STRING_CACHE* output_nets_sc
```

Definition at line 60 of file netlist_create_from_ast.cpp.

2.34.3.15 pad_string

```
char* pad_string
```

Definition at line 77 of file netlist_create_from_ast.cpp.

2.34.3.16 top_module

```
ast_node_t* top_module
```

Definition at line 79 of file netlist_create_from_ast.cpp.

2.34.3.17 type_of_circuit

```
int type_of_circuit
```

Definition at line 85 of file netlist_create_from_ast.cpp.

2.34.3.18 verilog_netlist

```
netlist_t* verilog_netlist
```

Definition at line 81 of file netlist_create_from_ast.cpp.

2.34.3.19 zero_string

```
char* zero_string
```

Definition at line 76 of file netlist_create_from_ast.cpp.

2.35 vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_create_from_ast.h File Reference

Functions

- void [create_netlist](#) ()
- void [connect_memory_and_alias](#) (ast_node_t *hb_instance, char *instance_name_prefix)

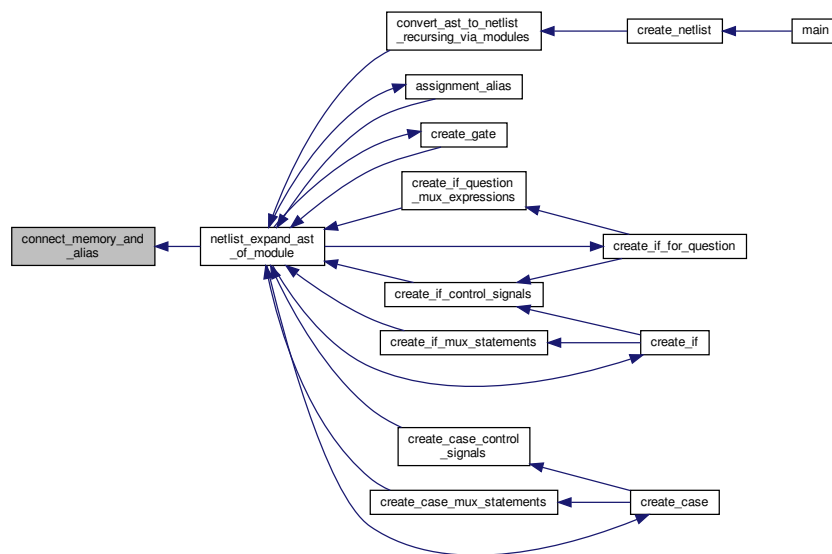
2.35.1 Function Documentation

2.35.1.1 connect_memory_and_alias()

```
void connect_memory_and_alias (
    ast_node_t * hb_instance,
    char * instance_name_prefix )
```

Definition at line 1836 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



2.35.1.2 create_netlist()

```
void create_netlist ( )
```

Definition at line 319 of file netlist_create_from_ast.cpp.

Here is the caller graph for this function:



2.36 vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_utils.cpp File Reference

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include "types.h"
#include "globals.h"
#include "netlist_utils.h"
#include "node_creation_library.h"
#include "odin_util.h"
#include "vtr_util.h"
#include "vtr_memory.h"
```

Functions

- `nnode_t * allocate_nnode ()`
- `nnode_t * free_nnode (nnode_t *to_free)`
- `void allocate_more_input_pins (nnode_t *node, int width)`
- `void allocate_more_output_pins (nnode_t *node, int width)`
- `void add_output_port_information (nnode_t *node, int port_width)`
- `void add_input_port_information (nnode_t *node, int port_width)`
- `npin_t * allocate_npin ()`
- `npin_t * copy_output_npin (npin_t *copy_pin)`
- `npin_t * copy_input_npin (npin_t *copy_pin)`
- `nnet_t * allocate_nnet ()`
- `void move_output_pin (nnode_t *node, int old_idx, int new_idx)`
- `void move_input_pin (nnode_t *node, int old_idx, int new_idx)`
- `void add_input_pin_to_node (nnode_t *node, npin_t *pin, int pin_idx)`
- `void add_fanout_pin_to_net (nnet_t *net, npin_t *pin)`
- `void add_output_pin_to_node (nnode_t *node, npin_t *pin, int pin_idx)`
- `void add_driver_pin_to_net (nnet_t *net, npin_t *pin)`
- `void combine_nets (nnet_t *output_net, nnet_t *input_net, netlist_t *netlist)`
- `void join_nets (nnet_t *join_to_net, nnet_t *other_net)`
- `void remap_pin_to_new_net (npin_t *pin, nnet_t *new_net)`
- `void remap_pin_to_new_node (npin_t *pin, nnode_t *new_node, int pin_idx)`
- `void connect_nodes (nnode_t *out_node, int out_idx, nnode_t *in_node, int in_idx)`
- `signal_list_t * init_signal_list ()`
- `void add_pin_to_signal_list (signal_list_t *list, npin_t *pin)`
- `signal_list_t * combine_lists (signal_list_t **signal_lists, int num_signal_lists)`
- `signal_list_t * combine_lists_without_freeing originals (signal_list_t **signal_lists, int num_signal_lists)`
- `signal_list_t * copy_input_signals (signal_list_t *signalsvar)`
- `signal_list_t * copy_output_signals (signal_list_t *signalsvar)`
- `void sort_signal_list_alphabetically (signal_list_t *list)`
- `signal_list_t * make_output_pins_for_existing_node (nnode_t *node, int width)`
- `void free_signal_list (signal_list_t *list)`
- `void hookup_input_pins_from_signal_list (nnode_t *node, int n_start_idx, signal_list_t *input_list, int il_start_idx, int width, netlist_t *netlist)`

- void [hookup_hb_input_pins_from_signal_list](#) (nnode_t *node, int n_start_idx, signal_list_t *input_list, int il_start←_idx, int width, netlist_t *netlist)
- void [hookup_output_pins_from_signal_list](#) (nnode_t *node, int n_start_idx, signal_list_t *output_list, int ol_start←_idx, int width)
- void [depth_traverse_count](#) (nnode_t *node, int *count, int traverse_mark_number)
- int [count_nodes_in_netlist](#) (netlist_t *netlist)
- netlist_t * [allocate_netlist](#) ()
- void [free_netlist](#) (netlist_t *to_free)
- void [add_node_to_netlist](#) (netlist_t *netlist, nnode_t *node, short special_node)
- void [mark_clock_node](#) (netlist_t *netlist, const char *clock_name)
- int [get_output_pin_index_from_mapping](#) (nnode_t *node, const char *name)
- int [get_output_port_index_from_mapping](#) (nnode_t *node, const char *name)
- int [get_input_pin_index_from_mapping](#) (nnode_t *node, const char *name)
- int [get_input_port_index_from_mapping](#) (nnode_t *node, const char *name)
- chain_information_t * [allocate_chain_info](#) ()
- void [remove_fanout_pins_from_net](#) (nnet_t *net, npin_t *, int id)

Variables

- global_args_t [global_args](#)

2.36.1 Function Documentation

2.36.1.1 add_driver_pin_to_net()

```
void add_driver_pin_to_net (
    nnet_t * net,
    npin_t * pin )
```

Definition at line 366 of file netlist_utils.cpp.


```
void add_input_pin_to_node (
```

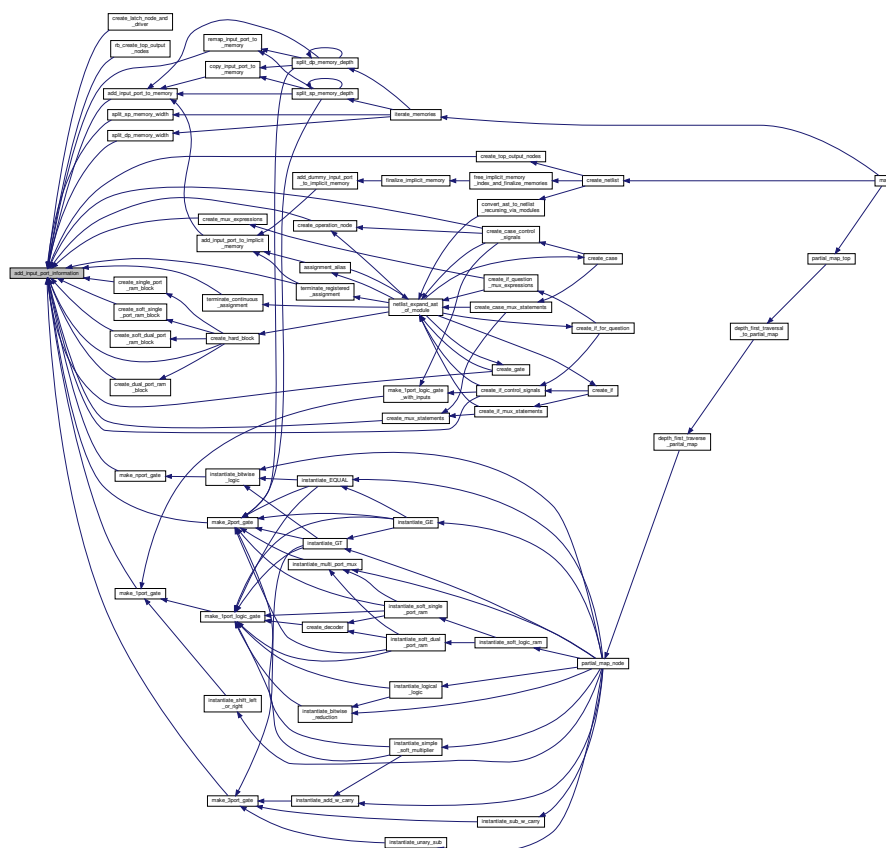
Generated on Thu Aug 10 2017 19:13:39 for Odin by Doxygen

Definition at line 316 of file netlist_utils.cpp.

[illegible]

```
void add_input_port_information (
    nnode_t * node,
    int port_width )
```

Here is the caller graph for this function:

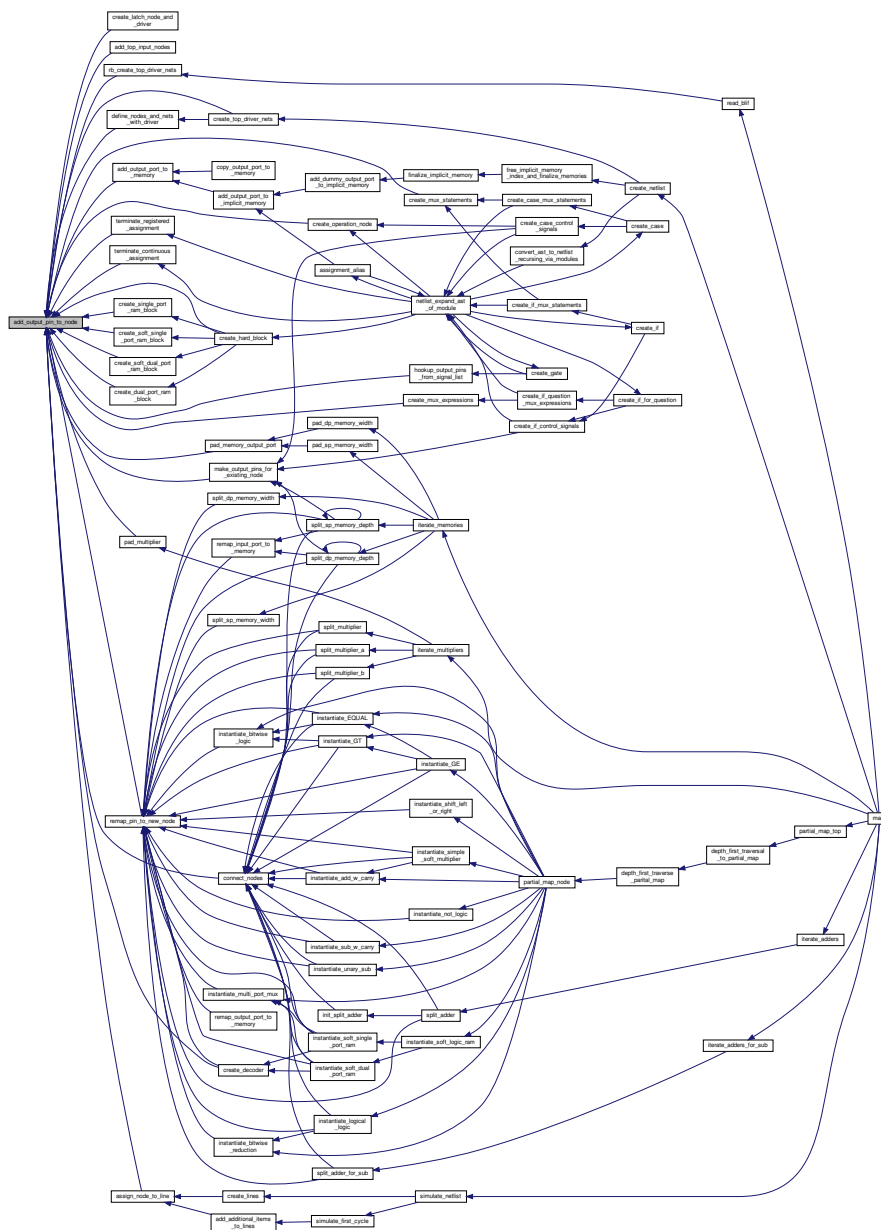


```
void add_node_to_netlist (
    netlist_t * netlist,
    nnode_t * node,
    short special_node )
```

Generated on Thu Aug 10 2017 19:13:39 for Odin by Doxygen

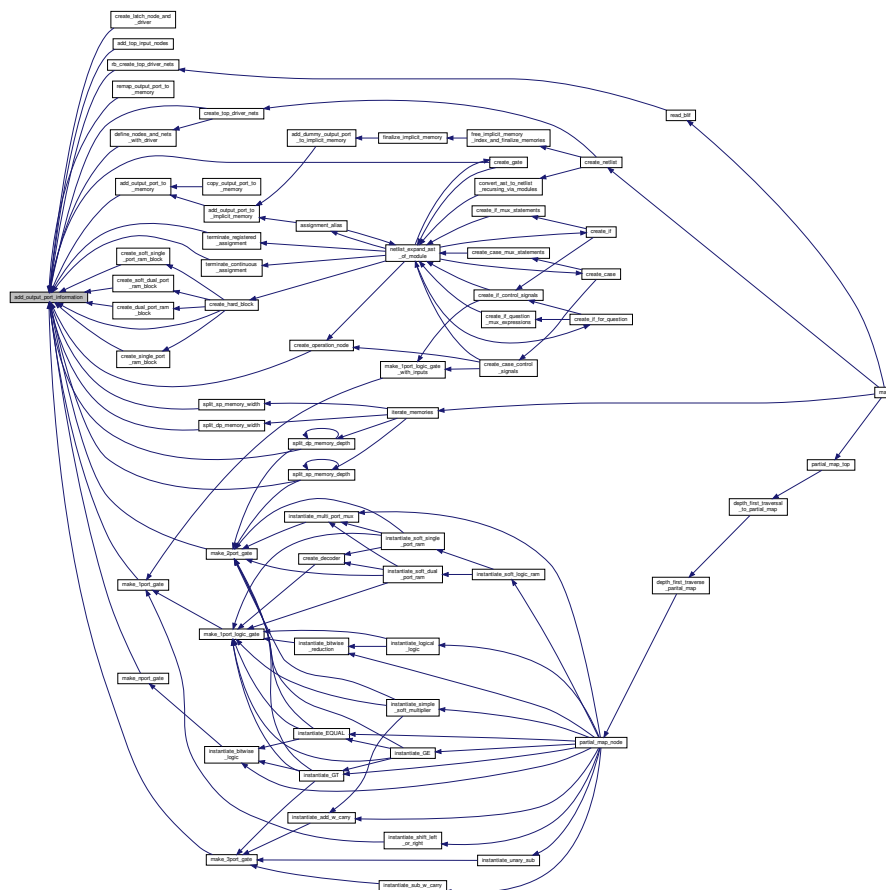
```
void add_output_pin_to_node (
    nnode_t * node,
    npin_t * pin,
    int pin_idx )
```

Here is the caller graph for this function:



```
void add_output_port_information (
    nnode_t * node,
    int port_width )
```

Here is the caller graph for this function:

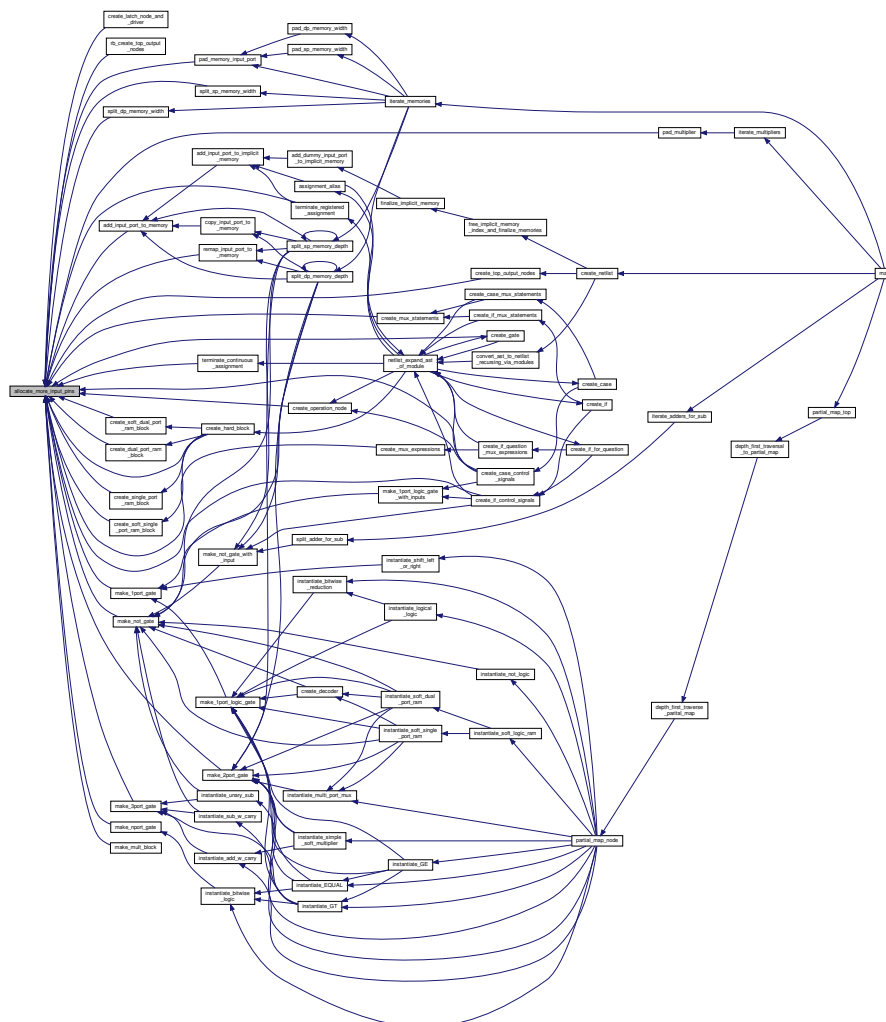


```
void add_pin_to_signal_list (
    signal_list_t * list,
    npin_t * pin )
```

Generated on Thu Aug 10 2017 19:13:39 for Odin by Doxygen

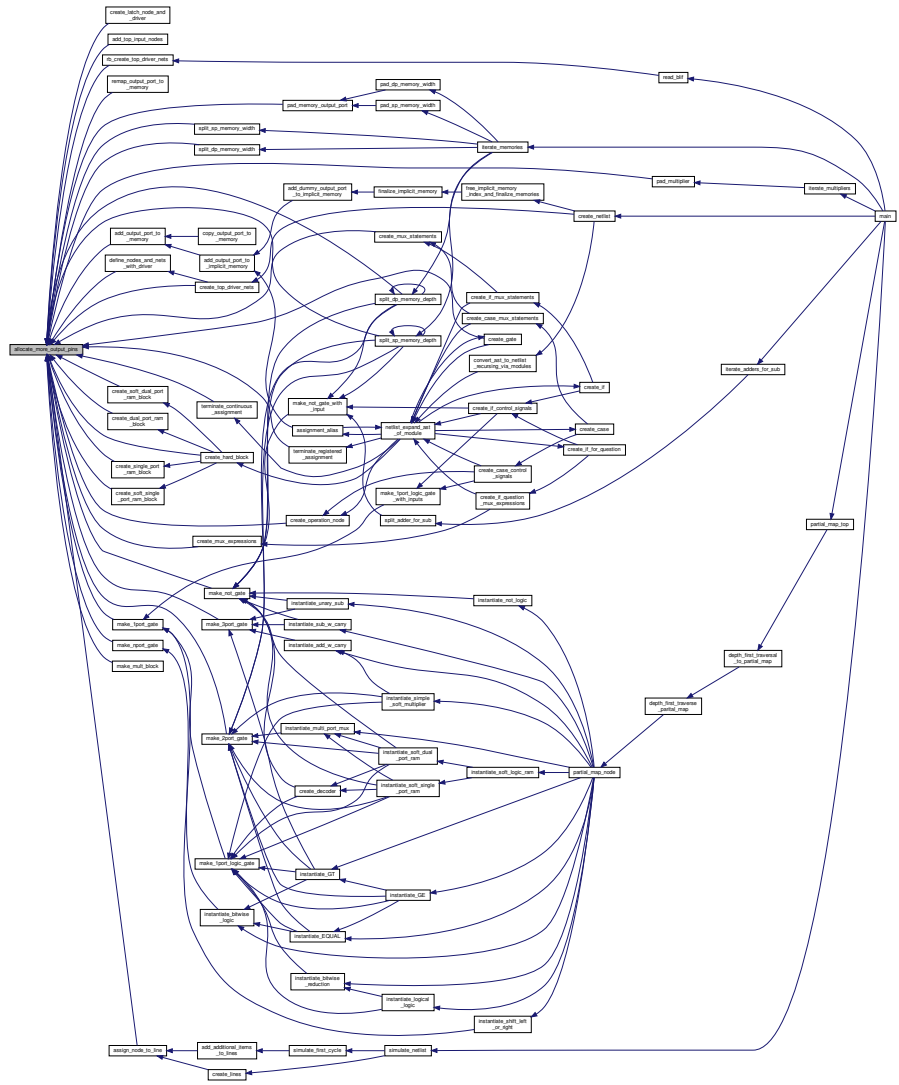

```
void allocate_more_input_pins (
    nnode_t * node,
    int width )
```

Here is the caller graph for this function:




```
void allocate_more_output_pins (
    nnode_t * node,
    int width )
```

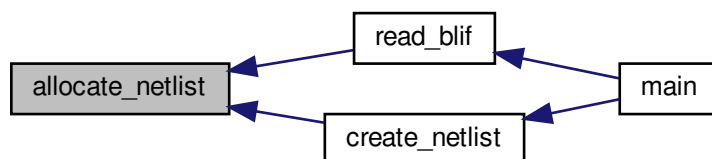
Here is the caller graph for this function:



```
netlist_t* allocate_netlist ( )
```

Definition at line 837 of file netlist_utils.cpp.

Here is the caller graph for this function:



2.36.1.13 allocate_nnet()

```
nnet_t* allocate_nnet ( )
```

Definition at line 254 of file netlist_utils.cpp.

```
nnode_t* allocate_nnode ( )
```

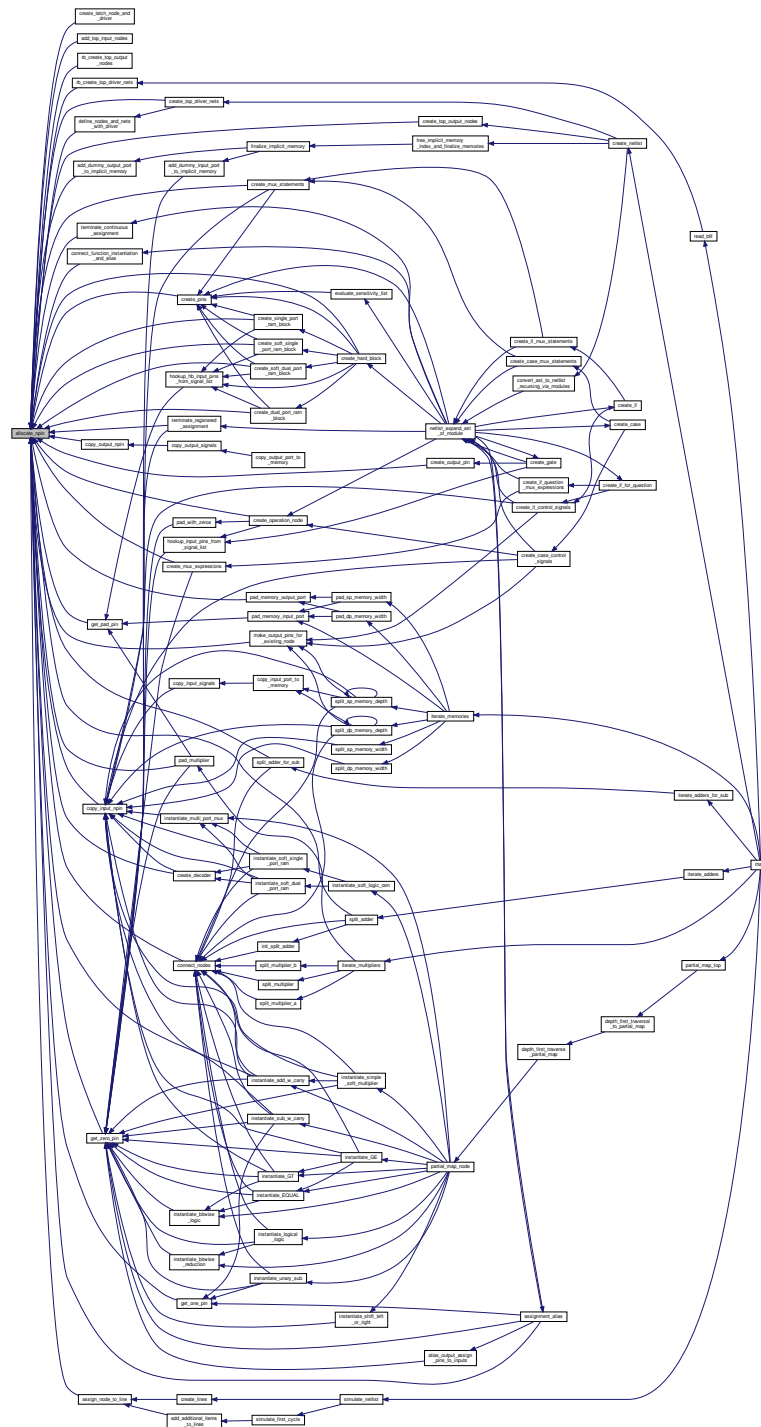
Generated on Thu Aug 10 2017 19:13:39 for Odin by Doxygen

[illegible]

```
npin_t* allocate_npin ( )
```

Definition at line 188 of file netlist_utils.cpp.

Here is the caller graph for this function:

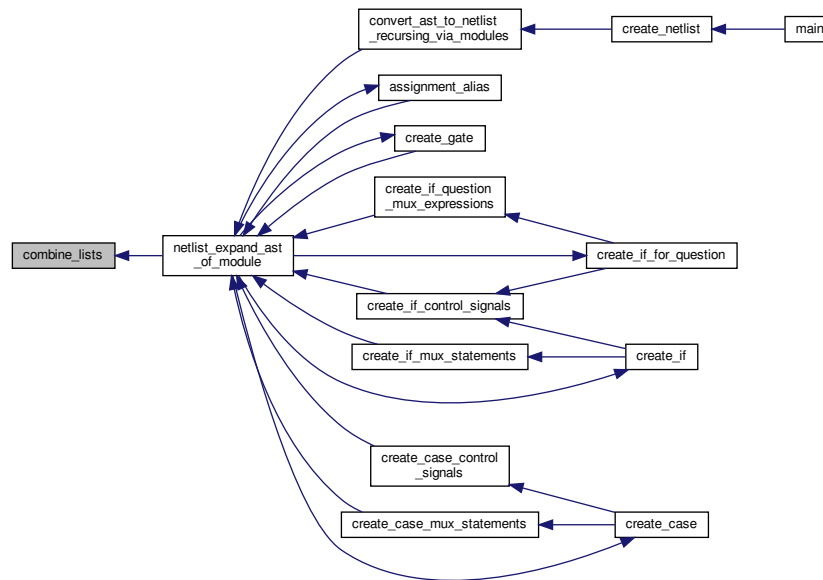


2.36.1.16 combine_lists()

```
signal_list_t* combine_lists (
    signal_list_t ** signal_lists,
    int num_signal_lists )
```

Definition at line 556 of file netlist_utils.cpp.

Here is the caller graph for this function:

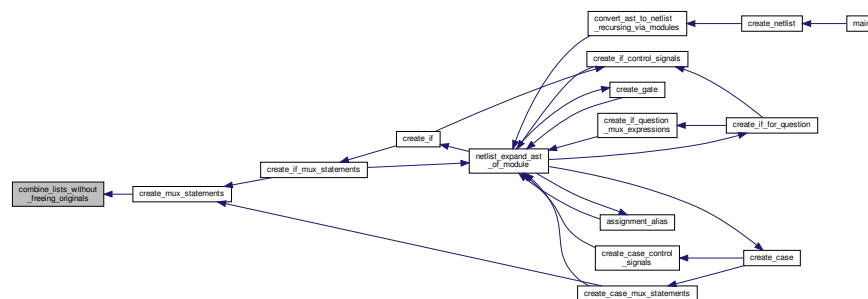


2.36.1.17 combine_lists_without_freeing_originals()

```
signal_list_t* combine_lists_without_freeing_originals (
    signal_list_t ** signal_lists,
    int num_signal_lists )
```

Definition at line 574 of file netlist_utils.cpp.

Here is the caller graph for this function:

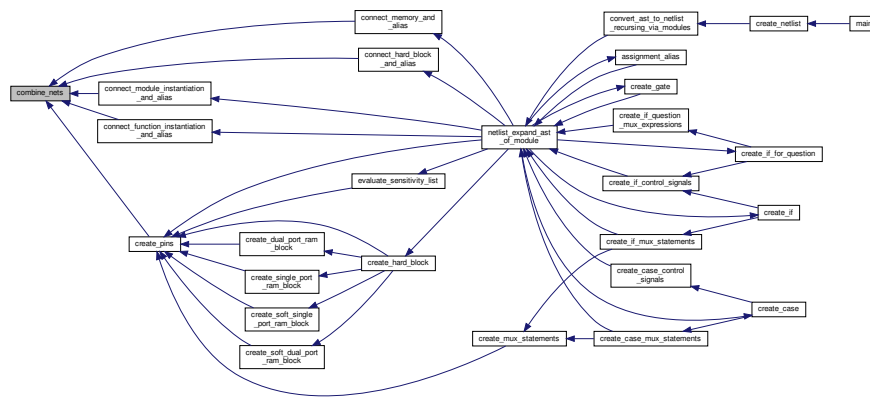


2.36.1.18 combine_nets()

```
void combine_nets (
    nnet_t * output_net,
    nnet_t * input_net,
    netlist_t * netlist )
```

Definition at line 384 of file netlist_utils.cpp.

Here is the caller graph for this function:

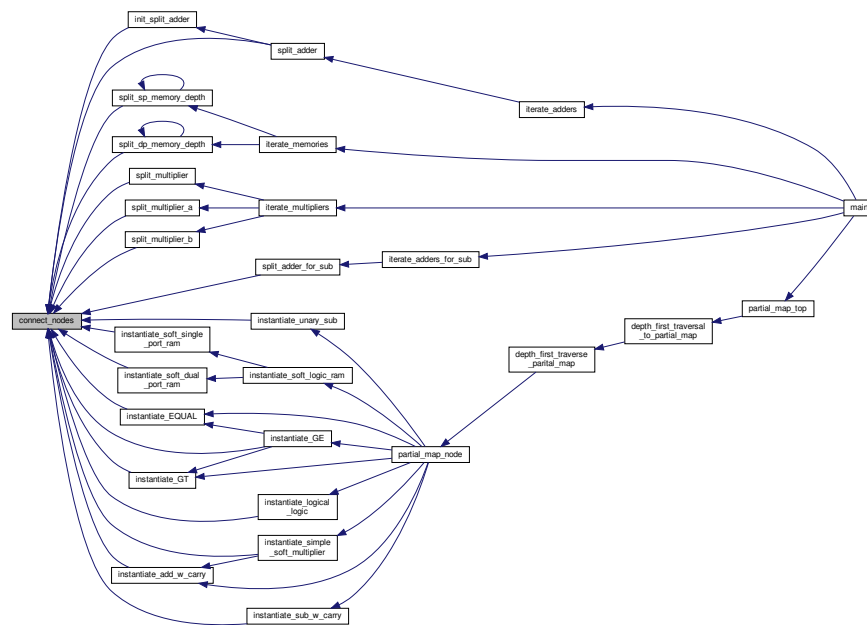


2.36.1.19 connect_nodes()

```
void connect_nodes (
    nnode_t * out_node,
    int out_idx,
    nnode_t * in_node,
    int in_idx )
```

Definition at line 489 of file netlist_utils.cpp.

Here is the caller graph for this function:



2.36.1.20 copy_input_npin()

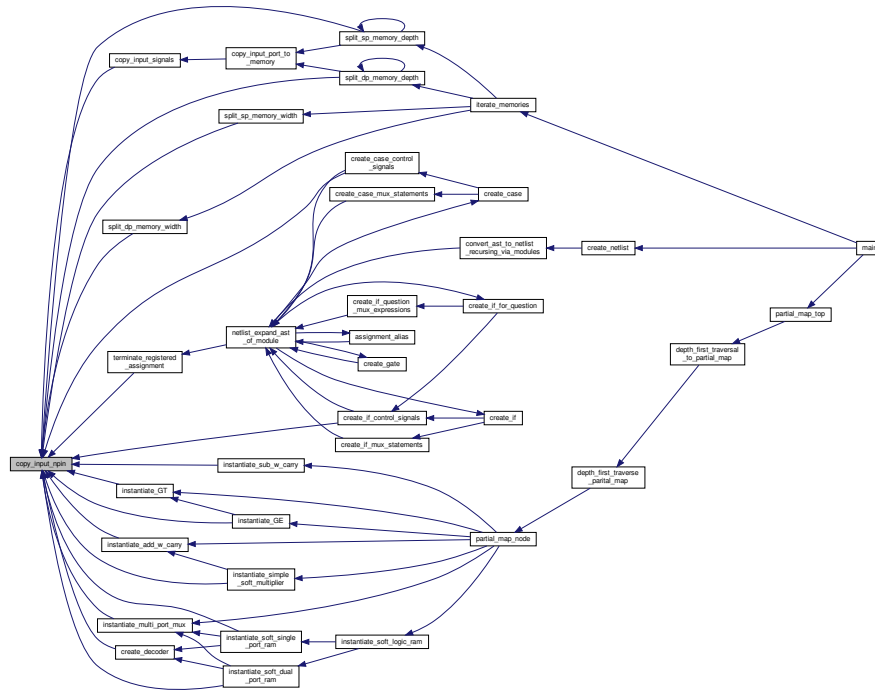
```

npin_t* copy_input_npin (
    npin_t * copy_pin )

```

Definition at line 234 of file netlist_utils.cpp.

Here is the caller graph for this function:

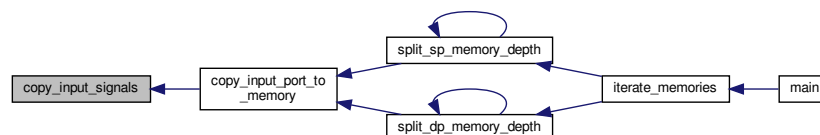


2.36.1.21 copy_input_signals()

```
signal_list_t* copy_input_signals (
    signal_list_t * signalsvar )
```

Definition at line 592 of file netlist_utils.cpp.

Here is the caller graph for this function:

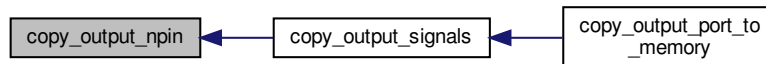


2.36.1.22 copy_output_npin()

```
npin_t* copy_output_npin (  
    npin_t * copy_pin )
```

Definition at line 217 of file netlist_utils.cpp.

Here is the caller graph for this function:

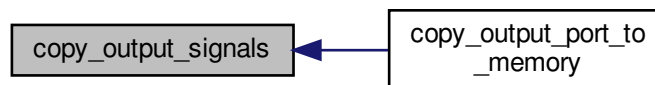


2.36.1.23 copy_output_signals()

```
signal_list_t* copy_output_signals (  
    signal_list_t * signalsvar )
```

Definition at line 606 of file netlist_utils.cpp.

Here is the caller graph for this function:



2.36.1.24 count_nodes_in_netlist()

```
int count_nodes_in_netlist (  
    netlist_t * netlist )
```

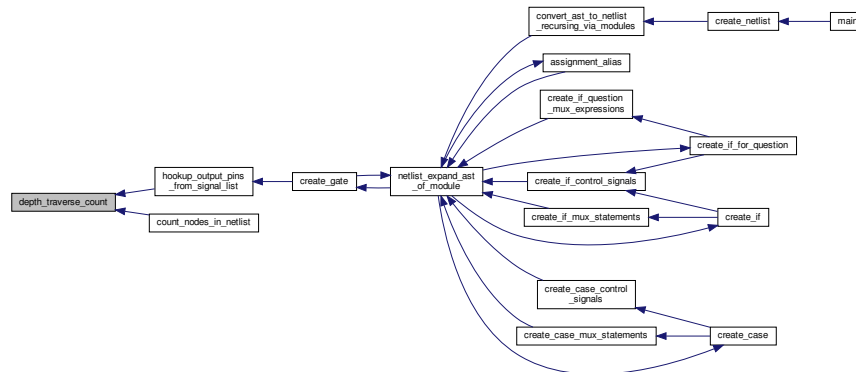
Definition at line 772 of file netlist_utils.cpp.

2.36.1.25 depth_traverse_count()

```
void depth_traverse_count (
    nnode_t * node,
    int * count,
    int traverse_mark_number )
```

Definition at line 795 of file netlist_utils.cpp.

Here is the caller graph for this function:



2.36.1.26 free_netlist()

```
void free_netlist (
    netlist_t * to_free )
```

Definition at line 886 of file netlist_utils.cpp.

2.36.1.27 free_nnode()

```
nnode_t* free_nnode (
    nnode_t * to_free )
```

Definition at line 96 of file netlist_utils.cpp.

[illegible]

```
void free_signal_list (
    signal_list_t * list )
```

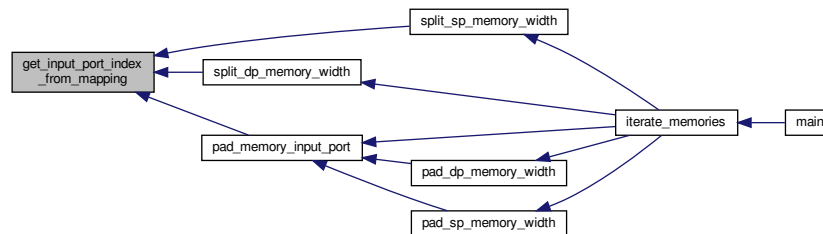
Generated on Thu Aug 10 2017 19:13:39 for Odin by Doxygen

2.36.1.30 get_input_port_index_from_mapping()

```
int get_input_port_index_from_mapping (
    nnode_t * node,
    const char * name )
```

Definition at line 1032 of file netlist_utils.cpp.

Here is the caller graph for this function:

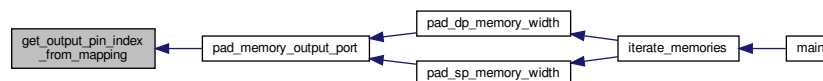


2.36.1.31 get_output_pin_index_from_mapping()

```
int get_output_pin_index_from_mapping (
    nnode_t * node,
    const char * name )
```

Definition at line 978 of file netlist_utils.cpp.

Here is the caller graph for this function:

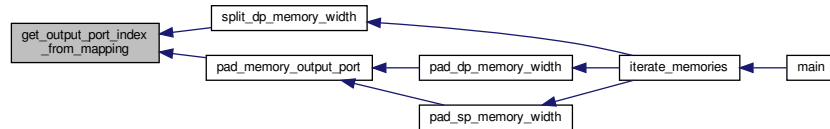


2.36.1.32 get_output_port_index_from_mapping()

```
int get_output_port_index_from_mapping (
    nnode_t * node,
    const char * name )
```

Definition at line 995 of file netlist_utils.cpp.

Here is the caller graph for this function:

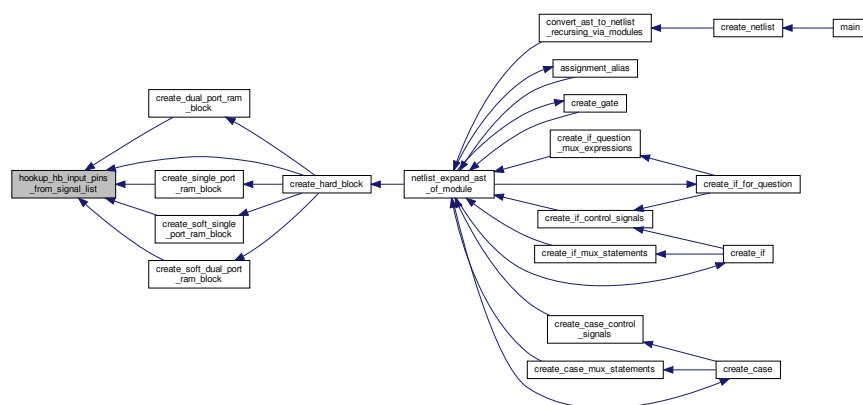


2.36.1.33 hookup_hb_input_pins_from_signal_list()

```
void hookup_hb_input_pins_from_signal_list (
    nnode_t * node,
    int n_start_idx,
    signal_list_t * input_list,
    int il_start_idx,
    int width,
    netlist_t * netlist )
```

Definition at line 717 of file netlist_utils.cpp.

Here is the caller graph for this function:



2.36.1.34 hookup_input_pins_from_signal_list()

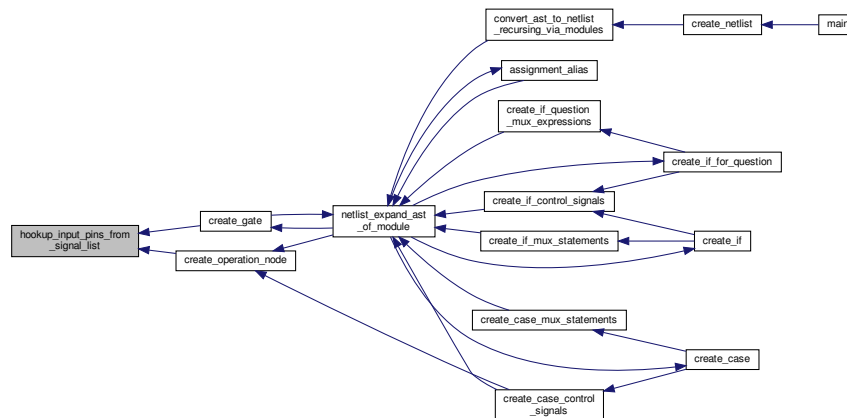
```

void hookup_input_pins_from_signal_list (
    nnode_t * node,
    int n_start_idx,
    signal_list_t * input_list,
    int il_start_idx,
    int width,
    netlist_t * netlist )

```

Definition at line 688 of file netlist_utils.cpp.

Here is the caller graph for this function:



2.36.1.35 hookup_output_pins_from_signal_list()

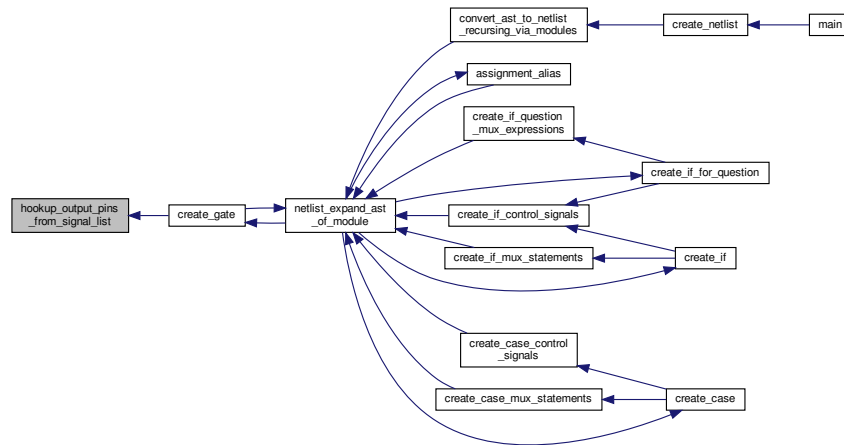
```

void hookup_output_pins_from_signal_list (
    nnode_t * node,
    int n_start_idx,
    signal_list_t * output_list,
    int ol_start_idx,
    int width )

```

Definition at line 743 of file netlist_utils.cpp.

Here is the caller graph for this function:



2.36.1.36 init_signal_list()

```
signal_list_t* init_signal_list ( )
```

Definition at line 529 of file netlist_utils.cpp.

[illegible]

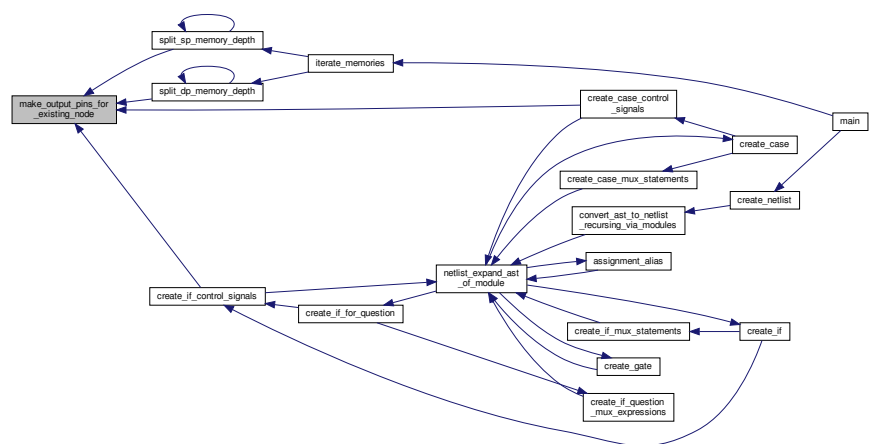
```
void join_nets (
    nnet_t * join_to_net,
    nnet_t * other_net )
```

Generated on Thu Aug 10 2017 19:13:39 for Odin by Doxygen

[illegible]

```
signal_list_t* make_output_pins_for_existing_node (
    nnode_t * node,
    int width )
```

Here is the caller graph for this function:



2.36.1.39 mark_clock_node()

```
void mark_clock_node (
    netlist_t * netlist,
    const char * clock_name )
```

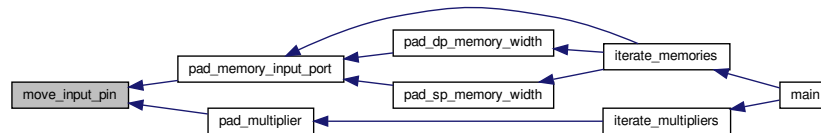
Definition at line 949 of file netlist_utils.cpp.

2.36.1.40 move_input_pin()

```
void move_input_pin (
    nnode_t * node,
    int old_idx,
    int new_idx )
```

Definition at line 296 of file netlist_utils.cpp.

Here is the caller graph for this function:

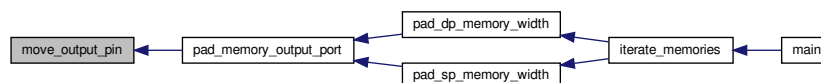


2.36.1.41 move_output_pin()

```
void move_output_pin (
    nnode_t * node,
    int old_idx,
    int new_idx )
```

Definition at line 276 of file netlist_utils.cpp.

Here is the caller graph for this function:



```
void remap_pin_to_new_net (
    npin_t * pin,
    nnet_t * new_net )
```

2.36.1.43 remap_pin_to_new_node()

Definition at line 469 of file netlist_utils.cpp.

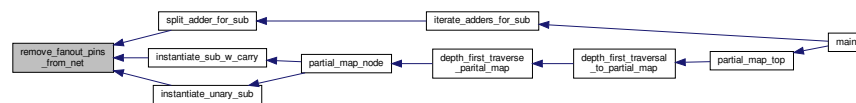
[illegible]

2.36.1.44 remove_fanout_pins_from_net()

```
void remove_fanout_pins_from_net (
    nnet_t * net,
    npin_t * ,
    int id )
```

Definition at line 1061 of file netlist_utils.cpp.

Here is the caller graph for this function:

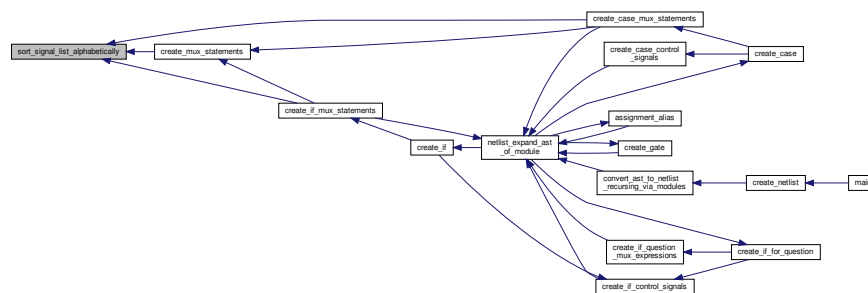


2.36.1.45 sort_signal_list_alphabetically()

```
void sort_signal_list_alphabetically (
    signal_list_t * list )
```

Definition at line 631 of file netlist_utils.cpp.

Here is the caller graph for this function:



2.36.2 Variable Documentation

2.36.2.1 global_args

```
global_args_t global_args
```

Definition at line 59 of file odin_ii.cpp.

2.37 vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_utils.h File Reference

```
#include "types.h"
```

Functions

- `nnode_t * allocate_nnode ()`
- `npin_t * allocate_npin ()`
- `npin_t * get_zero_pin (netlist_t *netlist)`
- `npin_t * get_pad_pin (netlist_t *netlist)`
- `npin_t * get_one_pin (netlist_t *netlist)`
- `npin_t * copy_input_npin (npin_t *copy_pin)`
- `npin_t * copy_output_npin (npin_t *copy_pin)`
- `nnet_t * allocate_nnet ()`
- `nnode_t * free_nnode (nnode_t *to_free)`
- `npin_t * free_npin (npin_t *to_free)`
- `void allocate_more_input_pins (nnode_t *node, int width)`
- `void allocate_more_output_pins (nnode_t *node, int width)`
- `void move_input_pin (nnode_t *node, int old_idx, int new_idx)`
- `void move_output_pin (nnode_t *node, int old_idx, int new_idx)`
- `void add_input_pin_to_node (nnode_t *node, npin_t *pin, int pin_idx)`
- `void add_fanout_pin_to_net (nnet_t *net, npin_t *pin)`
- `void add_output_pin_to_node (nnode_t *node, npin_t *pin, int pin_idx)`
- `void add_driver_pin_to_net (nnet_t *net, npin_t *pin)`
- `void add_output_port_information (nnode_t *node, int port_width)`
- `void add_input_port_information (nnode_t *node, int port_width)`
- `void combine_nets (nnet_t *output_net, nnet_t *input_net, netlist_t *netlist)`
- `void join_nets (nnet_t *net, nnet_t *input_net)`
- `void remap_pin_to_new_net (npin_t *pin, nnet_t *new_net)`
- `void remap_pin_to_new_node (npin_t *pin, nnode_t *new_node, int pin_idx)`
- `signal_list_t * init_signal_list ()`
- `void add_pin_to_signal_list (signal_list_t *list, npin_t *pin)`
- `void sort_signal_list_alphabetically (signal_list_t *list)`
- `signal_list_t * combine_lists (signal_list_t **signal_lists, int num_signal_lists)`
- `signal_list_t * combine_lists_without_freeing originals (signal_list_t **signal_lists, int num_signal_lists)`
- `signal_list_t * copy_input_signals (signal_list_t *signalsvar)`
- `signal_list_t * copy_output_signals (signal_list_t *signalsvar)`
- `void free_signal_list (signal_list_t *list)`
- `void hookup_hb_input_pins_from_signal_list (nnode_t *node, int n_start_idx, signal_list_t *input_list, int il_start_idx, int width, netlist_t *netlist)`
- `void hookup_input_pins_from_signal_list (nnode_t *node, int n_start_idx, signal_list_t *input_list, int il_start_idx, int width, netlist_t *netlist)`
- `void hookup_output_pins_from_signal_list (nnode_t *node, int n_start_idx, signal_list_t *output_list, int ol_start_idx, int width)`
- `signal_list_t * make_output_pins_for_existing_node (nnode_t *node, int width)`
- `void connect_nodes (nnode_t *out_node, int out_idx, nnode_t *in_node, int in_idx)`
- `int count_nodes_in_netlist (netlist_t *netlist)`
- `netlist_t * allocate_netlist ()`
- `void free_netlist (netlist_t *to_free)`

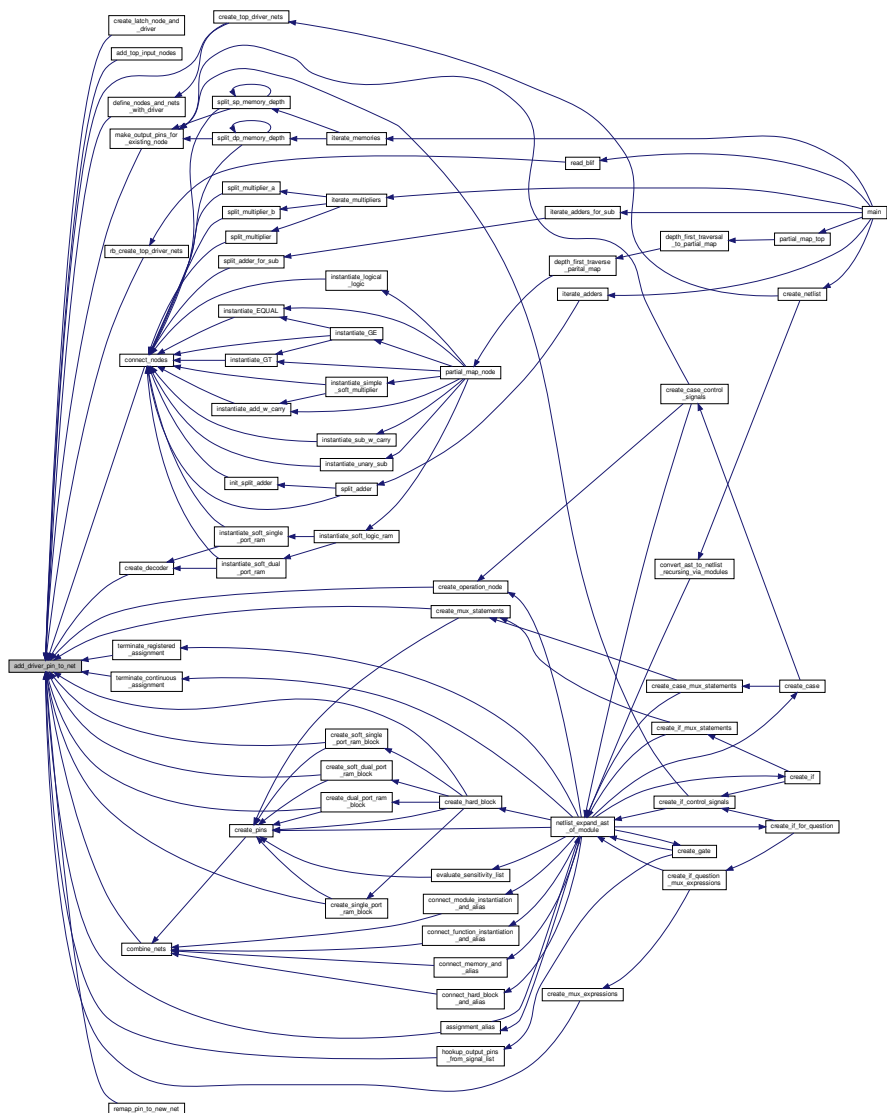
- void [add_node_to_netlist](#) (netlist_t *netlist, nnode_t *node, short special_node)
- void [mark_clock_node](#) (netlist_t *netlist, const char *clock_name)
- int [get_output_pin_index_from_mapping](#) (nnode_t *node, const char *name)
- int [get_output_port_index_from_mapping](#) (nnode_t *node, const char *name)
- int [get_input_pin_index_from_mapping](#) (nnode_t *node, const char *name)
- int [get_input_port_index_from_mapping](#) (nnode_t *node, const char *name)
- chain_information_t * [allocate_chain_info](#) ()
- void [remove_fanout_pins_from_net](#) (nnet_t *net, npin_t *pin, int id)

2.37.1 Function Documentation

2.37.1.1 add_driver_pin_to_net()

```
void add_driver_pin_to_net (  
    nnet_t * net,  
    npin_t * pin )
```

Definition at line 366 of file netlist_utils.cpp.



```
void add_input_pin_to_node (
```

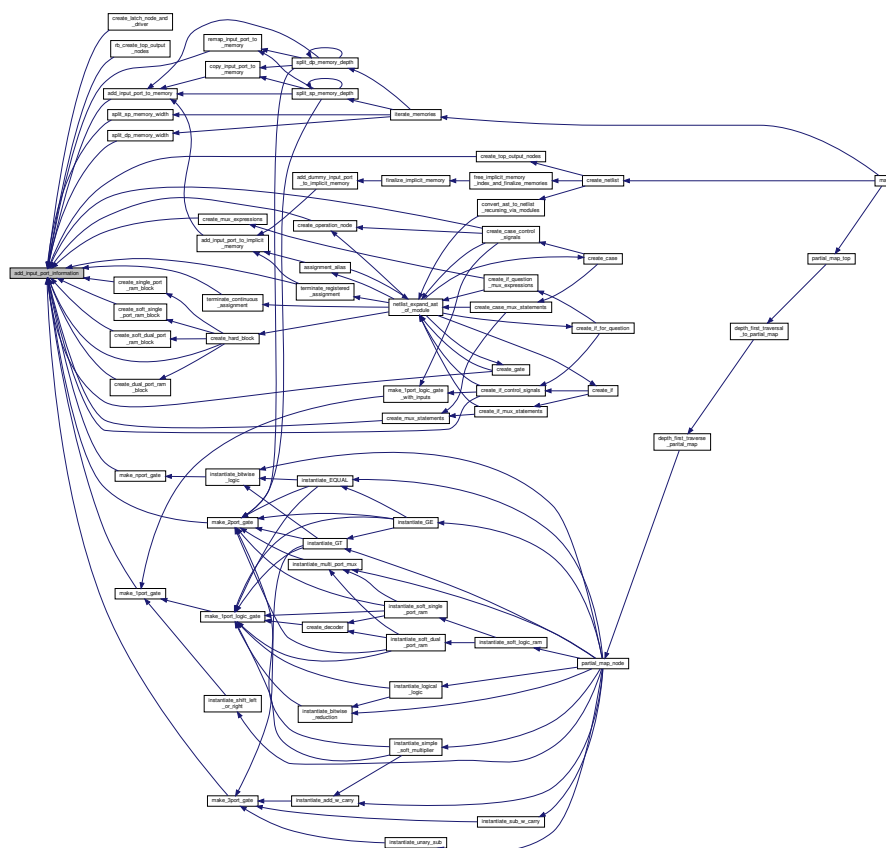
Generated on Thu Aug 10 2017 19:13:39 for Odin by Doxygen

Definition at line 316 of file netlist_utils.cpp.

[illegible]

```
void add_input_port_information (
    nnode_t * node,
    int port_width )
```

Here is the caller graph for this function:

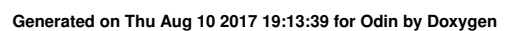


```
void add_node_to_netlist (
    netlist_t * netlist,
    nnode_t * node,
    short special_node )
```

Generated on Thu Aug 10 2017 19:13:39 for Odin by Doxygen

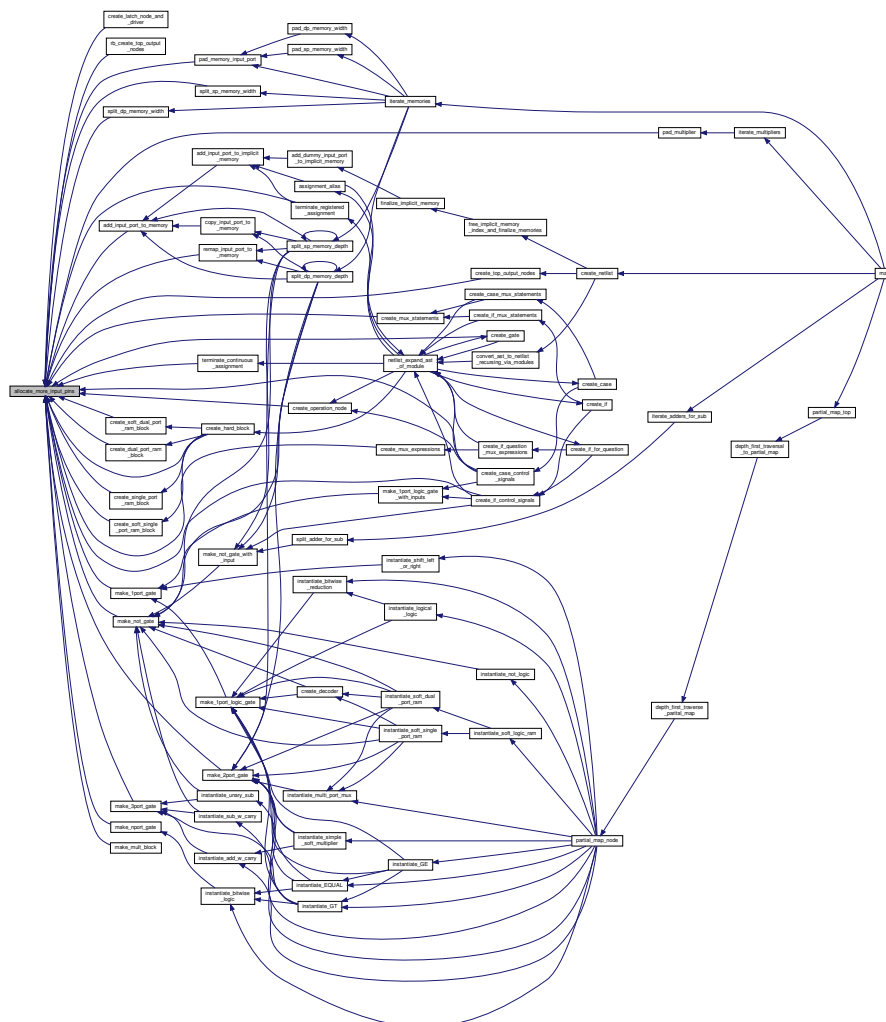
```
void add_output_pin_to_node (
    nnode_t * node,
    npin_t * pin,
    int pin_idx )
```

Here is the caller graph for this function:



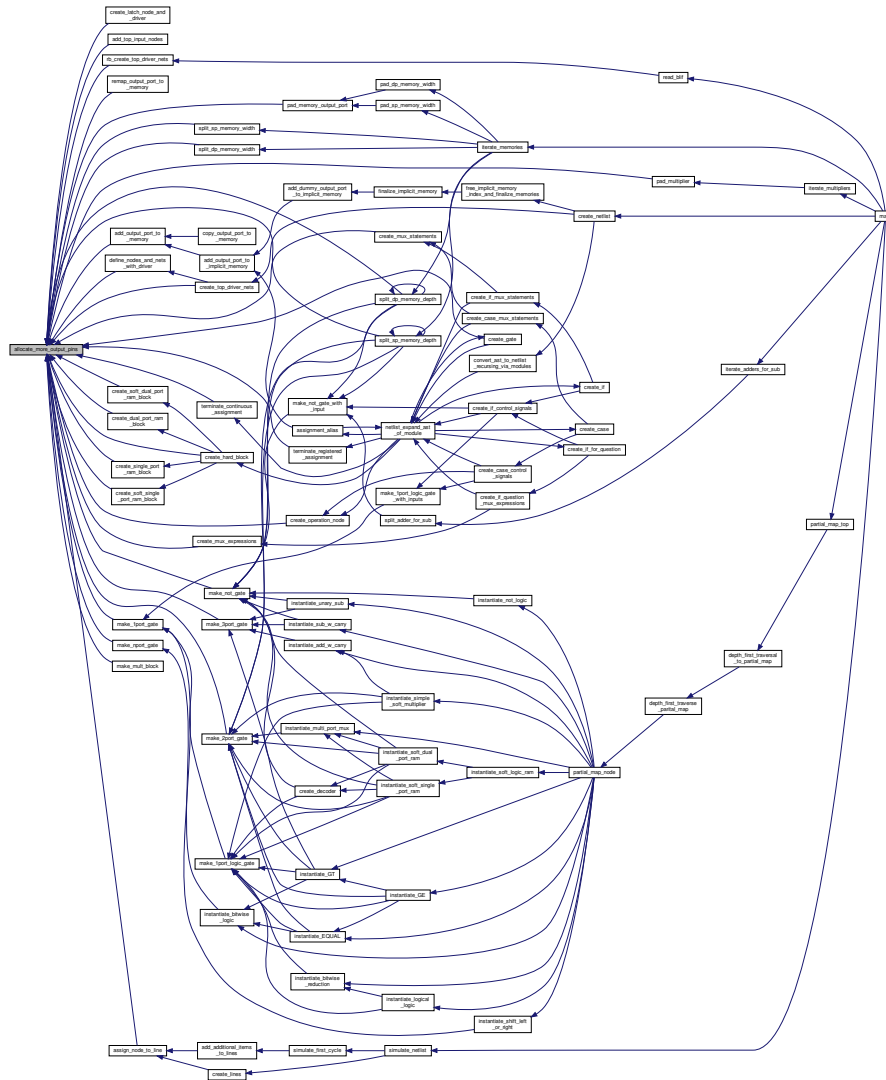

```
void allocate_more_input_pins (
    nnode_t * node,
    int width )
```

Here is the caller graph for this function:




```
void allocate_more_output_pins (
    nnode_t * node,
    int width )
```

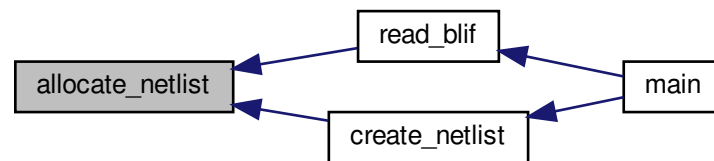
Here is the caller graph for this function:



```
netlist_t* allocate_netlist ( )
```

Definition at line 837 of file netlist_utils.cpp.

Here is the caller graph for this function:



2.37.1.13 `allocate_nnet()`

```
nnet_t* allocate_nnet ( )
```

Definition at line 254 of file netlist_utils.cpp.

[illegible]

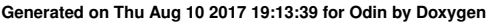
```
nnode_t* allocate_nnode ( )
```

Generated on Thu Aug 10 2017 19:13:39 for Odin by Doxygen

[illegible]

```
npin_t* allocate_npin ( )
```

Here is the caller graph for this function:

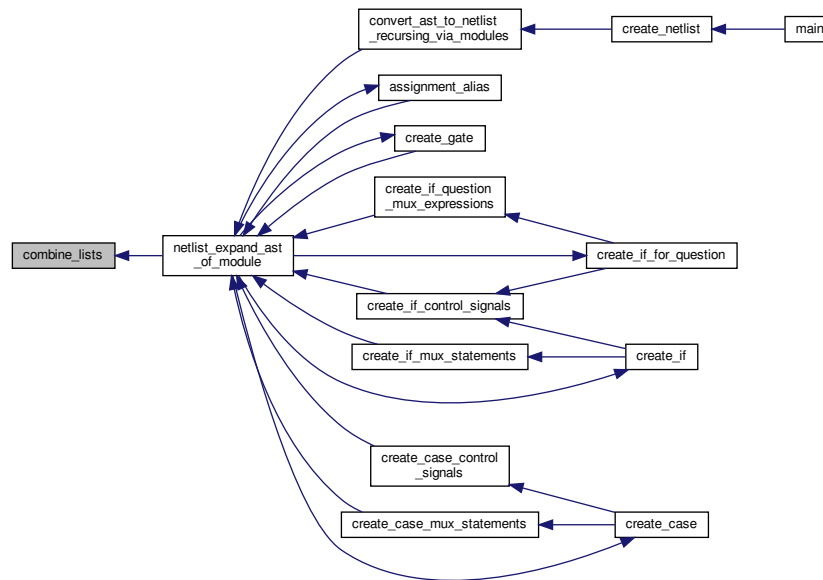


2.37.1.16 combine_lists()

```
signal_list_t* combine_lists (
    signal_list_t ** signal_lists,
    int num_signal_lists )
```

Definition at line 556 of file netlist_utils.cpp.

Here is the caller graph for this function:

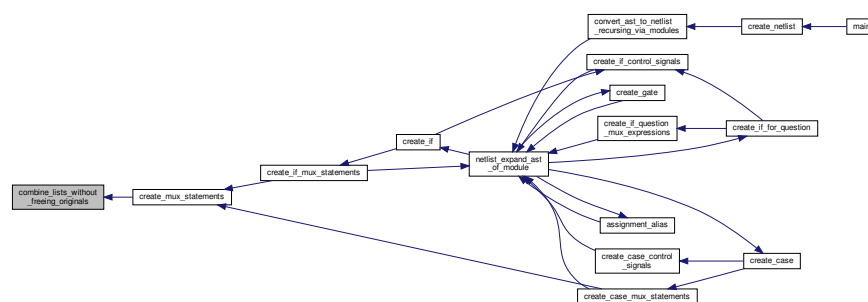


2.37.1.17 combine_lists_without_freeing_originals()

```
signal_list_t* combine_lists_without_freeing_originals (
    signal_list_t ** signal_lists,
    int num_signal_lists )
```

Definition at line 574 of file netlist_utils.cpp.

Here is the caller graph for this function:

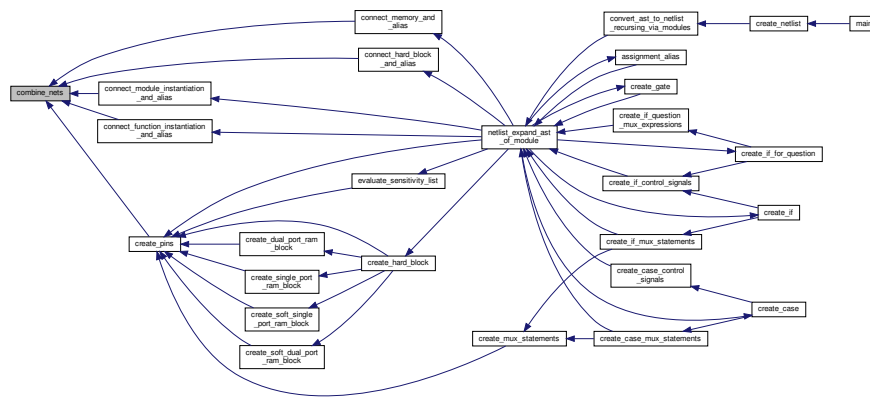


2.37.1.18 combine_nets()

```
void combine_nets (
    nnet_t * output_net,
    nnet_t * input_net,
    netlist_t * netlist )
```

Definition at line 384 of file netlist_utils.cpp.

Here is the caller graph for this function:

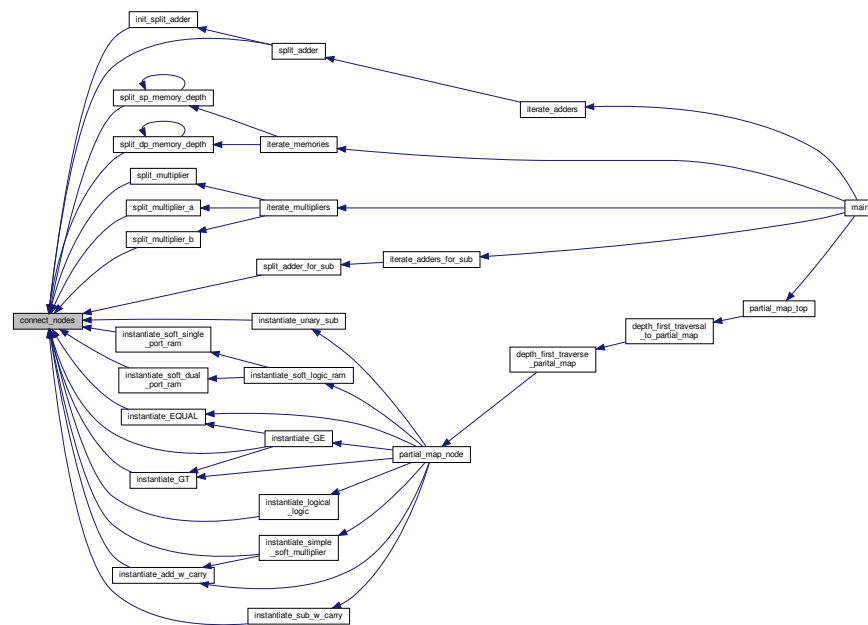


2.37.1.19 connect_nodes()

```
void connect_nodes (
    nnode_t * out_node,
    int out_idx,
    nnode_t * in_node,
    int in_idx )
```

Definition at line 489 of file netlist_utils.cpp.

Here is the caller graph for this function:



2.37.1.20 copy_input_npin()

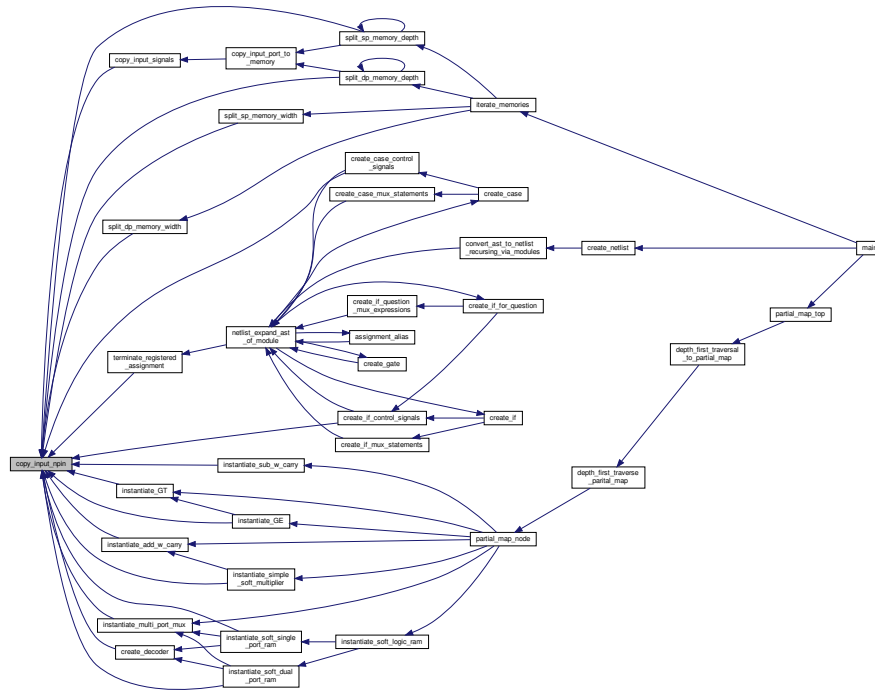
```

npin_t* copy_input_npin (
    npin_t * copy_pin )

```

Definition at line 234 of file netlist_utils.cpp.

Here is the caller graph for this function:

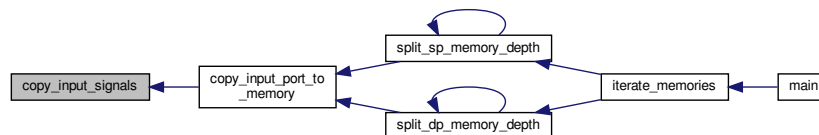


2.37.1.21 copy_input_signals()

```
signal_list_t* copy_input_signals (
    signal_list_t * signalsvar )
```

Definition at line 592 of file netlist_utils.cpp.

Here is the caller graph for this function:

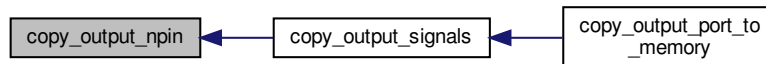


2.37.1.22 copy_output_npin()

```
npin_t* copy_output_npin (  
    npin_t * copy_pin )
```

Definition at line 217 of file netlist_utils.cpp.

Here is the caller graph for this function:

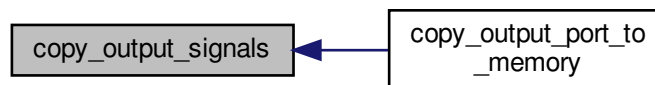


2.37.1.23 copy_output_signals()

```
signal_list_t* copy_output_signals (  
    signal_list_t * signalsvar )
```

Definition at line 606 of file netlist_utils.cpp.

Here is the caller graph for this function:



2.37.1.24 count_nodes_in_netlist()

```
int count_nodes_in_netlist (  
    netlist_t * netlist )
```

Definition at line 772 of file netlist_utils.cpp.

2.37.1.25 free_netlist()

```
void free_netlist (
    netlist_t * to_free )
```

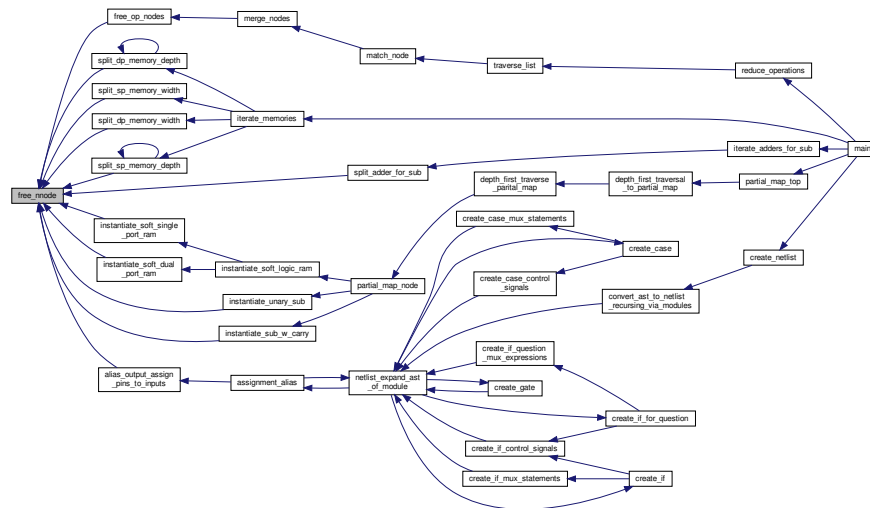
Definition at line 886 of file netlist_utils.cpp.

2.37.1.26 free_nnode()

```
nnode_t* free_nnode (
    nnode_t * to_free )
```

Definition at line 96 of file netlist_utils.cpp.

Here is the caller graph for this function:

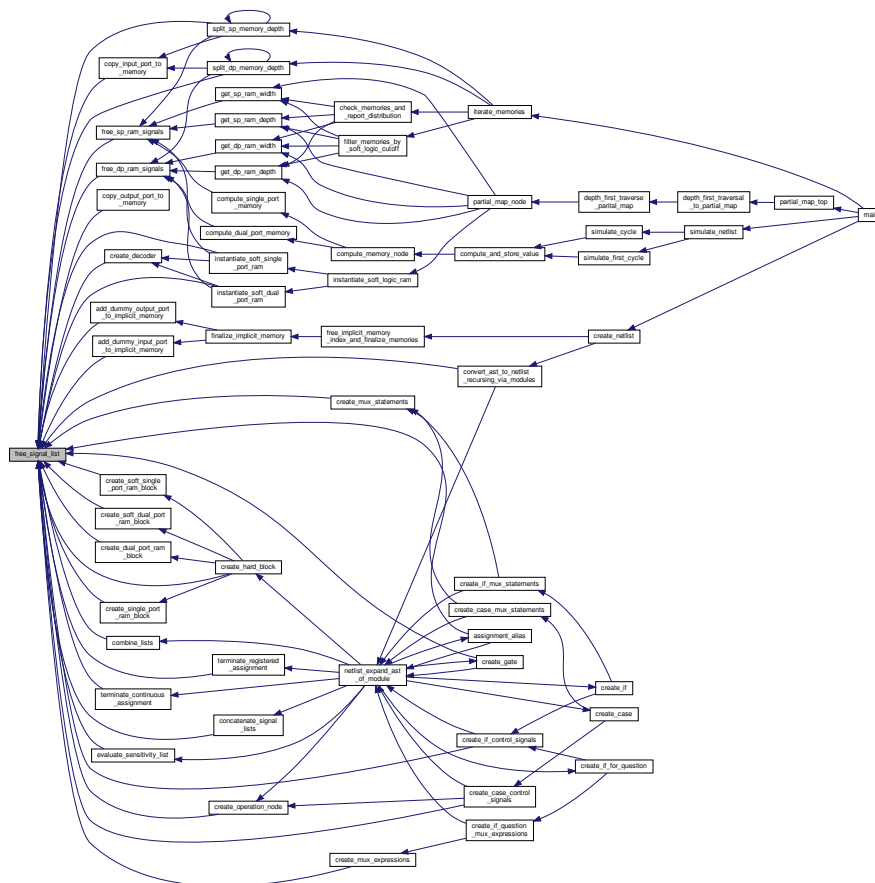


2.37.1.27 free_npin()

```
npin_t* free_npin (
    npin_t * to_free )
```

2.37.1.28 free_signal_list()

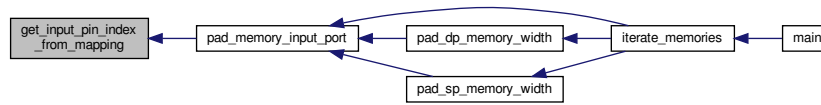
Definition at line 675 of file netlist_utils.cpp.



2.37.1.29 `get_input_pin_index_from_mapping()`

Definition at line 1015 of file netlist_utils.cpp.

Here is the caller graph for this function:



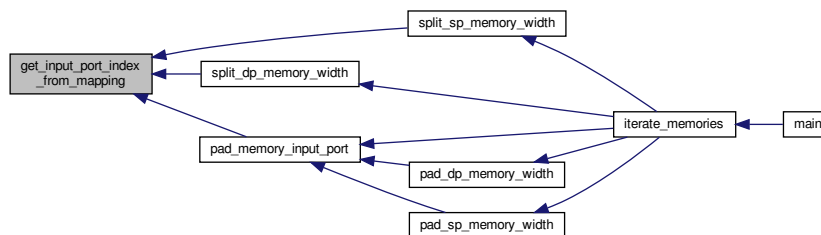
2.37.1.30 get_input_port_index_from_mapping()

```

int get_input_port_index_from_mapping (
    nnode_t * node,
    const char * name )
  
```

Definition at line 1032 of file netlist_utils.cpp.

Here is the caller graph for this function:



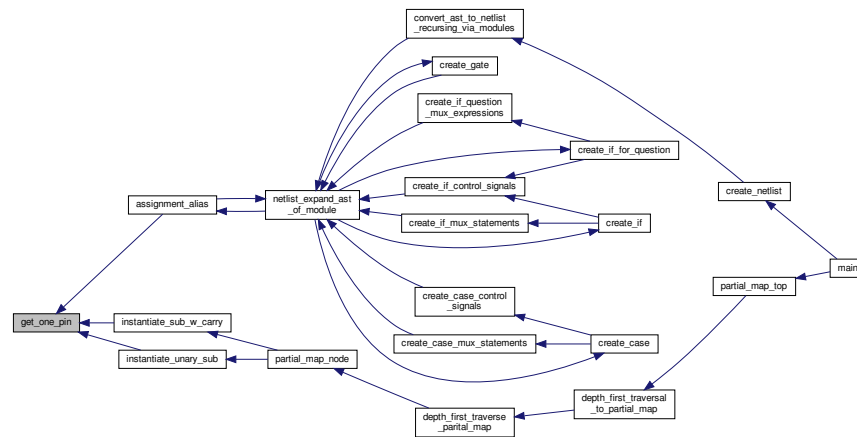
2.37.1.31 get_one_pin()

```

npin_t* get_one_pin (
    netlist_t * netlist )
  
```

Definition at line 64 of file node_creation_library.cpp.

Here is the caller graph for this function:



2.37.1.32 get_output_pin_index_from_mapping()

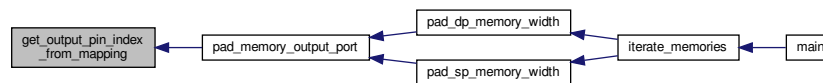
```

int get_output_pin_index_from_mapping (
    nnode_t * node,
    const char * name )

```

Definition at line 978 of file netlist_utils.cpp.

Here is the caller graph for this function:



2.37.1.33 get_output_port_index_from_mapping()

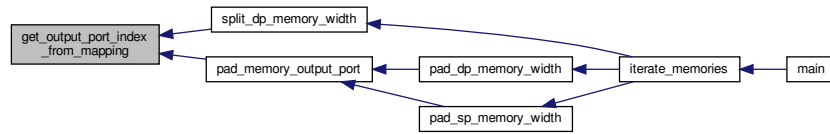
```

int get_output_port_index_from_mapping (
    nnode_t * node,
    const char * name )

```

Definition at line 995 of file netlist_utils.cpp.

Here is the caller graph for this function:



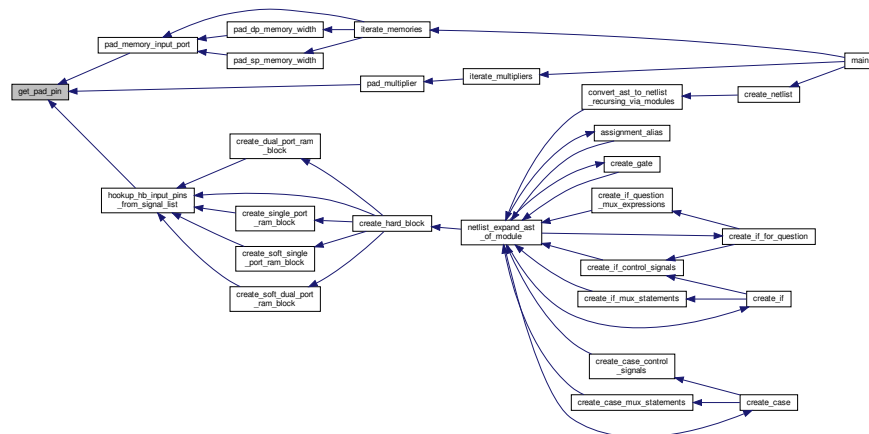
2.37.1.34 get_pad_pin()

```

npin_t* get_pad_pin (
    netlist_t * netlist )
  
```

Definition at line 40 of file node_creation_library.cpp.

Here is the caller graph for this function:



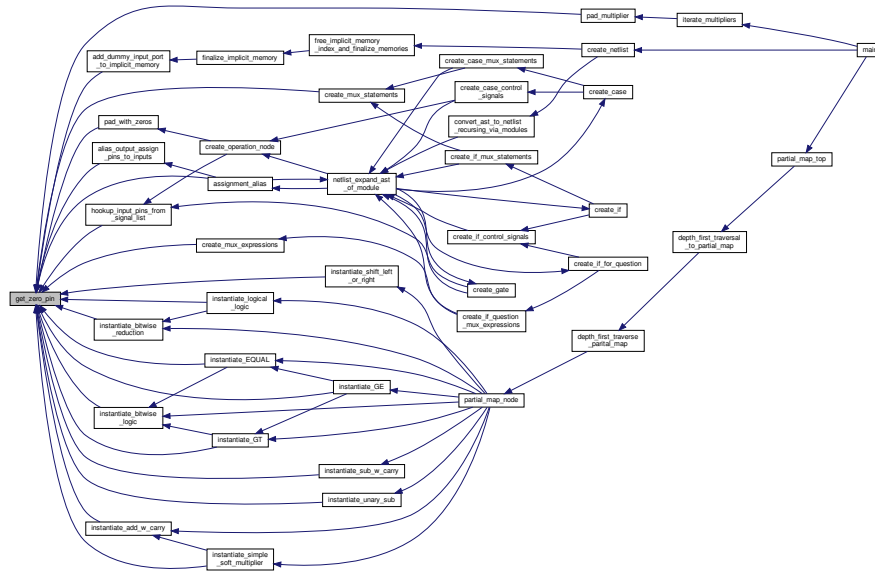
2.37.1.35 get_zero_pin()

```

npin_t* get_zero_pin (
    netlist_t * netlist )
  
```

Definition at line 52 of file node_creation_library.cpp.

Here is the caller graph for this function:

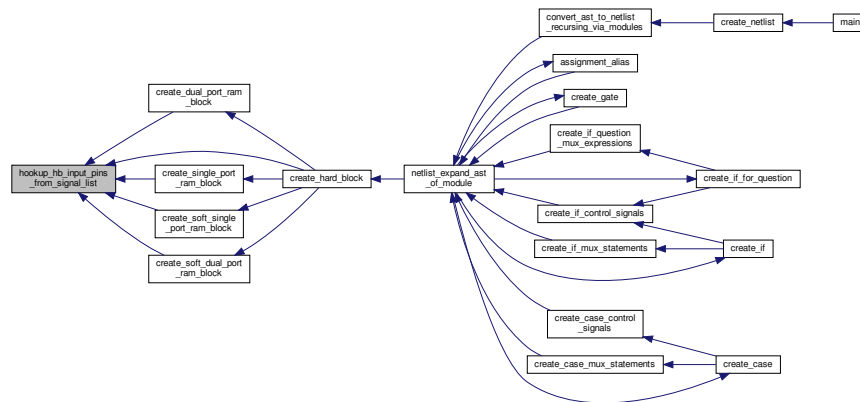


2.37.1.36 hookup_hb_input_pins_from_signal_list()

```
void hookup_hb_input_pins_from_signal_list (
    nnode_t * node,
    int n_start_idx,
    signal_list_t * input_list,
    int il_start_idx,
    int width,
    netlist_t * netlist )
```

Definition at line 717 of file `netlist_utils.cpp`.

Here is the caller graph for this function:

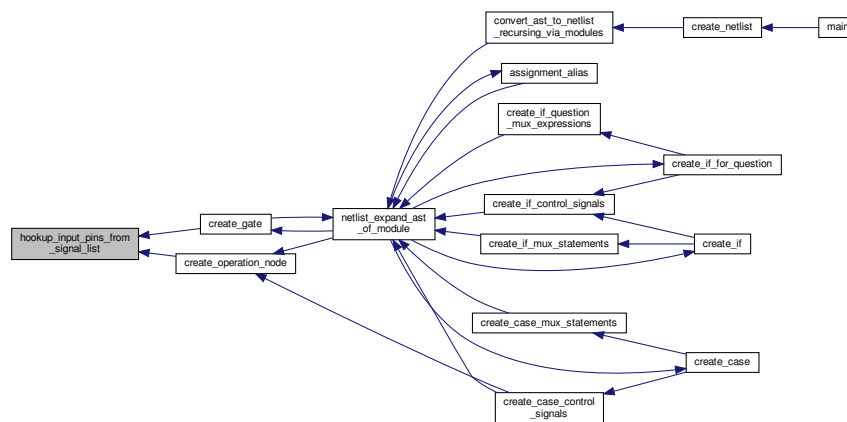


2.37.1.37 hookup_input_pins_from_signal_list()

```
void hookup_input_pins_from_signal_list (
    nnode_t * node,
    int n_start_idx,
    signal_list_t * input_list,
    int il_start_idx,
    int width,
    netlist_t * netlist )
```

Definition at line 688 of file netlist_utils.cpp.

Here is the caller graph for this function:

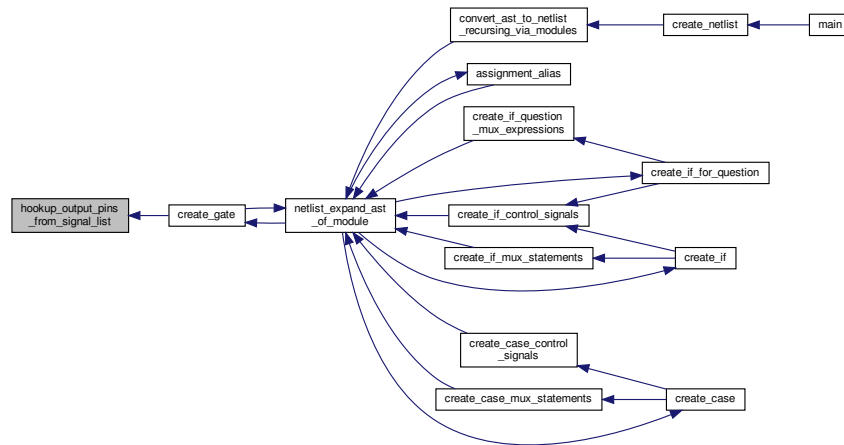


2.37.1.38 hookup_output_pins_from_signal_list()

```
void hookup_output_pins_from_signal_list (
    nnode_t * node,
    int n_start_idx,
    signal_list_t * output_list,
    int ol_start_idx,
    int width )
```

Definition at line 743 of file netlist_utils.cpp.

Here is the caller graph for this function:



2.37.1.39 init_signal_list()

```
signal_list_t* init_signal_list ( )
```

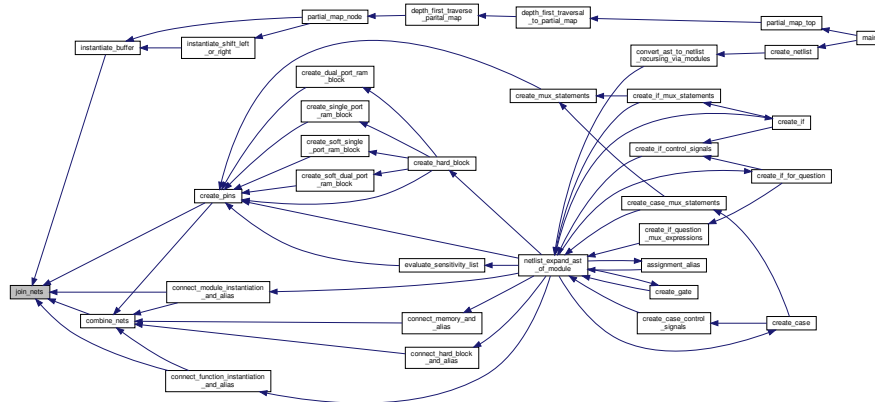
Definition at line 529 of file `netlist_utils.cpp`.

[illegible]

```
void join_nets (
    nnet_t * net,
    nnet_t * input_net )
```

Generated on Thu Aug 10 2017 19:13:39 for Odin by Doxygen

Here is the caller graph for this function:

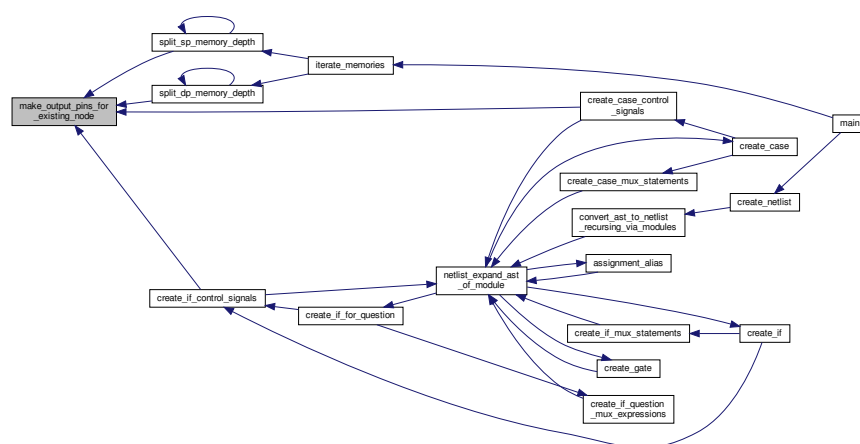


2.37.1.41 make_output_pins_for_existing_node()

```
signal_list_t* make_output_pins_for_existing_node (
    nnode_t * node,
    int width )
```

Definition at line 642 of file netlist_utils.cpp.

Here is the caller graph for this function:



2.37.1.42 mark_clock_node()

```
void mark_clock_node (
    netlist_t * netlist,
    const char * clock_name )
```

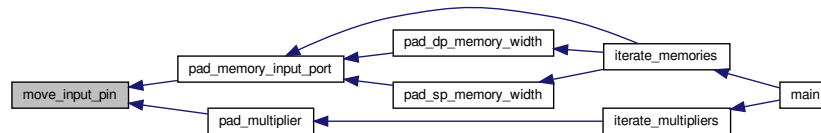
Definition at line 949 of file netlist_utils.cpp.

2.37.1.43 move_input_pin()

```
void move_input_pin (
    nnode_t * node,
    int old_idx,
    int new_idx )
```

Definition at line 296 of file netlist_utils.cpp.

Here is the caller graph for this function:

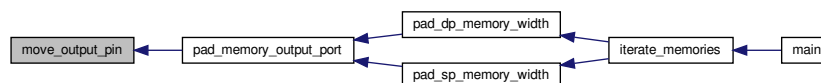


2.37.1.44 move_output_pin()

```
void move_output_pin (
    nnode_t * node,
    int old_idx,
    int new_idx )
```

Definition at line 276 of file netlist_utils.cpp.

Here is the caller graph for this function:



```
void remap_pin_to_new_net (
    npin_t * pin,
    nnet_t * new_net )
```

2.37.1.46 remap_pin_to_new_node()

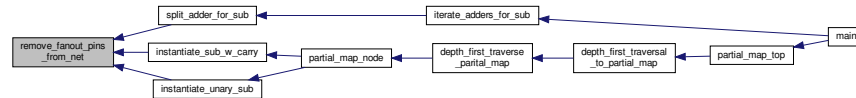
Definition at line 469 of file netlist_utils.cpp.

2.37.1.47 remove_fanout_pins_from_net()

```
void remove_fanout_pins_from_net (
    nnet_t * net,
    npin_t * pin,
    int id )
```

Definition at line 1061 of file netlist_utils.cpp.

Here is the caller graph for this function:

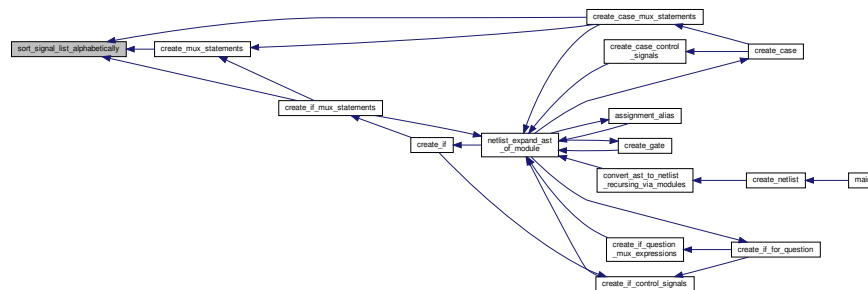


2.37.1.48 sort_signal_list_alphabetically()

```
void sort_signal_list_alphabetically (
    signal_list_t * list )
```

Definition at line 631 of file netlist_utils.cpp.

Here is the caller graph for this function:



2.38 vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/node_creation_library.cpp File Reference

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include "types.h"
#include "globals.h"
#include "netlist_utils.h"
#include "odin_util.h"
#include "node_creation_library.h"
#include "vtr_util.h"
```

Functions

- `npin_t * get_pad_pin (netlist_t *netlist)`
- `npin_t * get_zero_pin (netlist_t *netlist)`
- `npin_t * get_one_pin (netlist_t *netlist)`
- `nnode_t * make_not_gate_with_input (npin_t *input_pin, nnode_t *node, short mark)`
- `nnode_t * make_not_gate (nnode_t *node, short mark)`
- `nnode_t * make_1port_gate (operation_list type, int width_input, int width_output, nnode_t *node, short mark)`
- `nnode_t * make_1port_logic_gate (operation_list type, int width, nnode_t *node, short mark)`
- `nnode_t * make_1port_logic_gate_with_inputs (operation_list type, int width, signal_list_t *pin_list, nnode_t *node, short mark)`
- `nnode_t * make_3port_gate (operation_list type, int width_port1, int width_port2, int width_port3, int width_output, nnode_t *node, short mark)`
- `nnode_t * make_2port_gate (operation_list type, int width_port1, int width_port2, int width_output, nnode_t *node, short mark)`
- `nnode_t * make_nport_gate (operation_list type, int port_sizes, int width, int width_output, nnode_t *node, short mark)`
- `const char * node_name_based_on_op (nnode_t *node)`
- `char * hard_node_name (nnode_t *, char *instance_name_prefix, char *hb_name, char *hb_inst)`
- `char * node_name (nnode_t *node, char *instance_name_prefix)`
- `nnode_t * make_mult_block (nnode_t *node, short mark)`

Variables

- `long unique_node_name_id = 0`
- `const char * MULTI_PORT_MUX_string = "MULTI_PORT_MUX"`
- `const char * FF_NODE_string = "FF_NODE"`
- `const char * BUF_NODE_string = "BUF_NODE"`
- `const char * INPUT_NODE_string = "INPUT_NODE"`
- `const char * CLOCK_NODE_string = "CLOCK_NODE"`
- `const char * OUTPUT_NODE_string = "OUTPUT_NODE"`
- `const char * GND_NODE_string = "GND_NODE"`
- `const char * VCC_NODE_string = "VCC_NODE"`
- `const char * ADD_string = "ADD"`
- `const char * MINUS_string = "MINUS"`
- `const char * BITWISE_NOT_string = "BITWISE_NOT"`
- `const char * BITWISE_AND_string = "BITWISE_AND"`
- `const char * BITWISE_OR_string = "BITWISE_OR"`
- `const char * BITWISE_NAND_string = "BITWISE_NAND"`
- `const char * BITWISE_NOR_string = "BITWISE_NOR"`
- `const char * BITWISE_XNOR_string = "BITWISE_XNOR"`
- `const char * BITWISE_XOR_string = "BITWISE_XOR"`
- `const char * LOGICAL_NOT_string = "LOGICAL_NOT"`
- `const char * LOGICAL_OR_string = "LOGICAL_OR"`
- `const char * LOGICAL_AND_string = "LOGICAL_AND"`
- `const char * LOGICAL_NAND_string = "LOGICAL_NAND"`
- `const char * LOGICAL_NOR_string = "LOGICAL_NOR"`
- `const char * LOGICAL_XOR_string = "LOGICAL_XOR"`
- `const char * LOGICAL_XNOR_string = "LOGICAL_XNOR"`
- `const char * MULTIPLY_string = "MULTIPLY"`

- const char * [DIVIDE_string](#) = "DIVIDE"
- const char * [MODULO_string](#) = "MODULO"
- const char * [LT_string](#) = "LT"
- const char * [GT_string](#) = "GT"
- const char * [LOGICAL_EQUAL_string](#) = "LOGICAL_EQUAL"
- const char * [NOT_EQUAL_string](#) = "NOT_EQUAL"
- const char * [LTE_string](#) = "LTE"
- const char * [GTE_string](#) = "GTE"
- const char * [SR_string](#) = "SR"
- const char * [SL_string](#) = "SL"
- const char * [CASE_EQUAL_string](#) = "CASE_EQUAL"
- const char * [CASE_NOT_EQUAL_string](#) = "CASE_NOT_EQUAL"
- const char * [ADDER_FUNC_string](#) = "ADDER_FUNC"
- const char * [CARRY_FUNC_string](#) = "CARRY_FUNC"
- const char * [MUX_2_string](#) = "MUX_2"
- const char * [HARD_IP_string](#) = "HARD_IP"
- const char * [MEMORY_string](#) = "MEMORY"

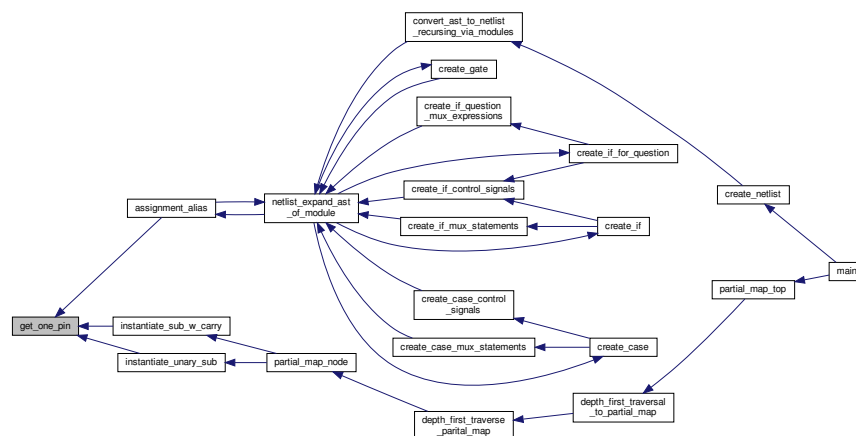
2.38.1 Function Documentation

2.38.1.1 `get_one_pin()`

```
npin_t* get_one_pin (
    netlist_t * netlist )
```

Definition at line 64 of file `node_creation_library.cpp`.

Here is the caller graph for this function:



2.38.1.2 get_pad_pin()

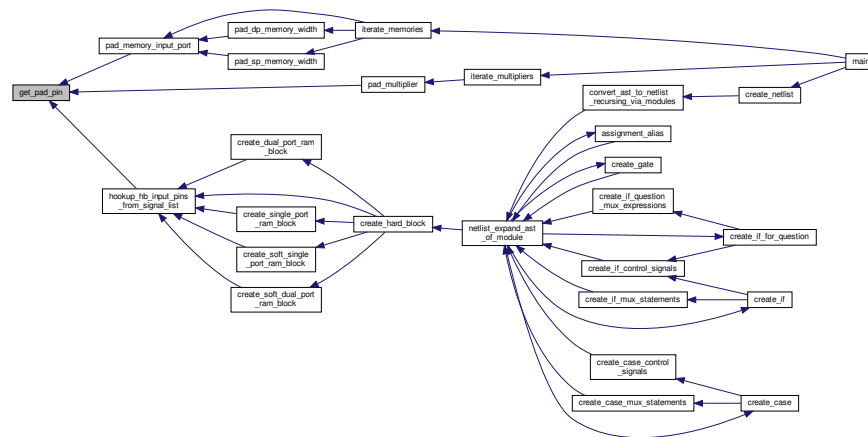
```

npin_t* get_pad_pin (
    netlist_t * netlist )

```

Definition at line 40 of file node_creation_library.cpp.

Here is the caller graph for this function:



2.38.1.3 get_zero_pin()

```

npin_t* get_zero_pin (
    netlist_t * netlist )

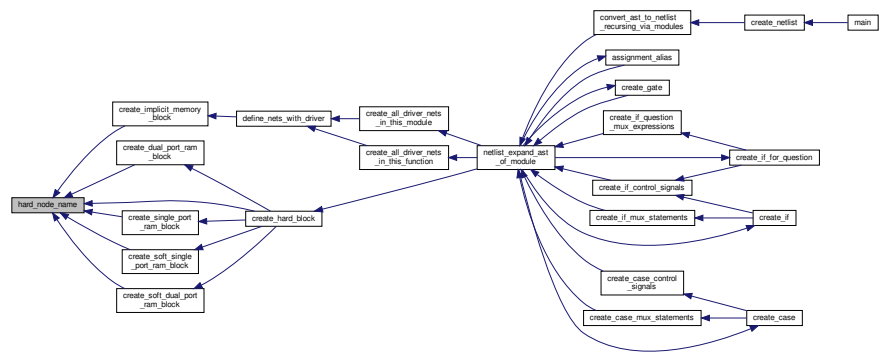
```

Definition at line 52 of file node_creation_library.cpp.

[illegible]

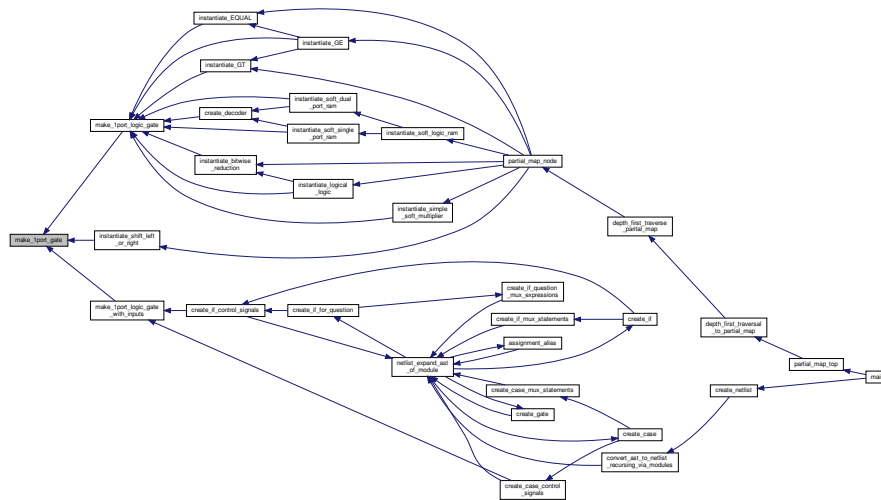
```
char* hard_node_name (
    nnode_t *,
    char * instance_name_prefix,
    char * hb_name,
    char * hb_inst )
```

Here is the caller graph for this function:



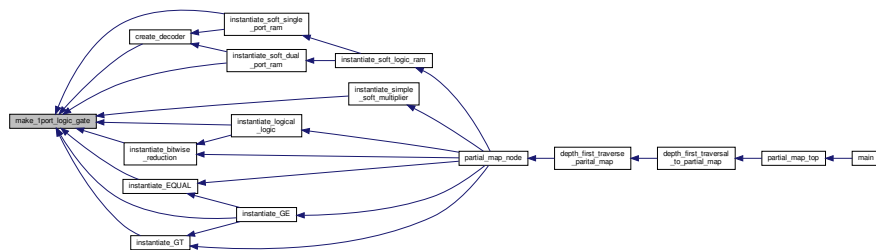
```
nnode_t* make_lport_gate (
    operation_list type,
    int width_input,
    int width_output,
    nnode_t * node,
    short mark )
```

Here is the caller graph for this function:



```
nnode_t* make_lport_logic_gate (
    operation_list type,
    int width,
    nnode_t * node,
    short mark )
```

Here is the caller graph for this function:

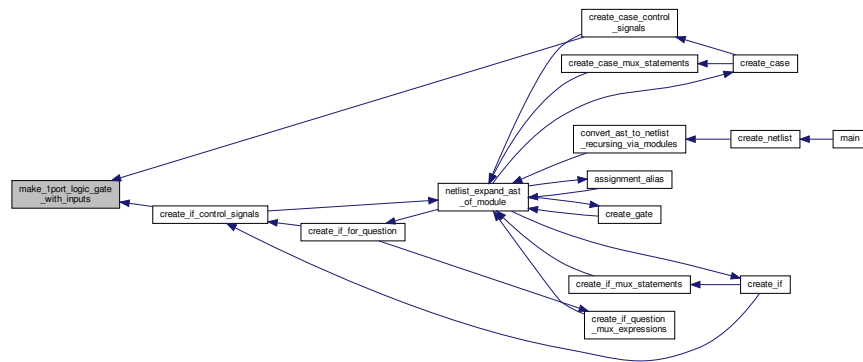


2.38.1.7 make_1port_logic_gate_with_inputs()

```
nnode_t* make_1port_logic_gate_with_inputs (
    operation_list type,
    int width,
    signal_list_t * pin_list,
    nnode_t * node,
    short mark )
```

Definition at line 149 of file node_creation_library.cpp.

Here is the caller graph for this function:

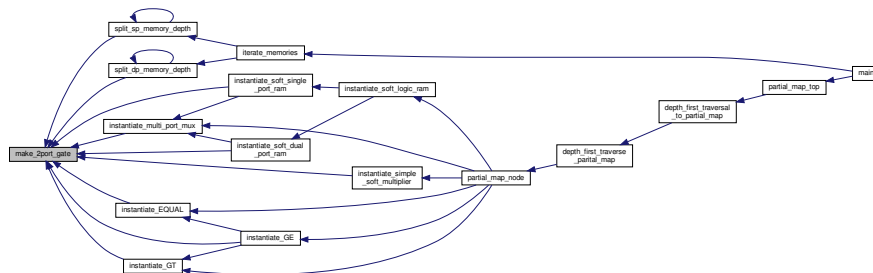


2.38.1.8 make_2port_gate()

```
nnode_t* make_2port_gate (
    operation_list type,
    int width_port1,
    int width_port2,
    int width_output,
    nnode_t * node,
    short mark )
```

Definition at line 196 of file node_creation_library.cpp.

Here is the caller graph for this function:

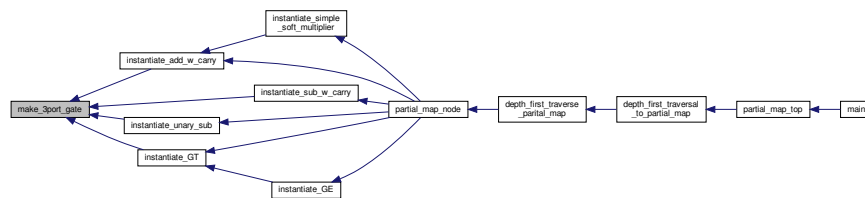


2.38.1.9 make_3port_gate()

```
nnode_t* make_3port_gate (
    operation_list type,
    int width_port1,
    int width_port2,
    int width_port3,
    int width_output,
    nnode_t * node,
    short mark )
```

Definition at line 169 of file node_creation_library.cpp.

Here is the caller graph for this function:



2.38.1.10 make_mult_block()

```
nnode_t* make_mult_block (
    nnode_t * node,
    short mark )
```

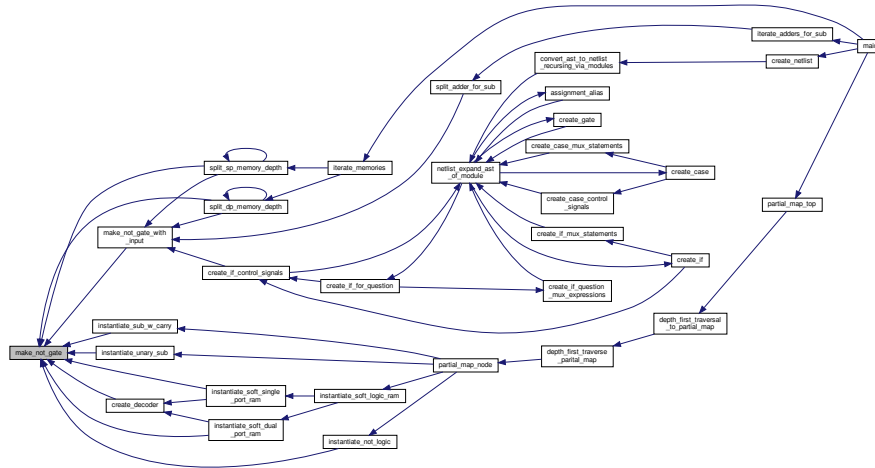
Definition at line 468 of file node_creation_library.cpp.

2.38.1.11 make_not_gate()

```
nnode_t* make_not_gate (
    nnode_t * node,
    short mark )
```

Definition at line 94 of file node_creation_library.cpp.

Here is the caller graph for this function:

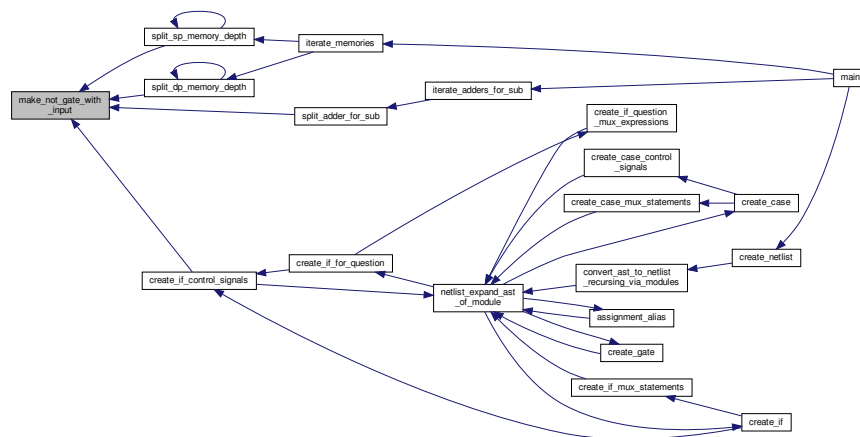


2.38.1.12 make_not_gate_with_input()

```
nnode_t* make_not_gate_with_input (
    npin_t * input_pin,
    nnode_t * node,
    short mark )
```

Definition at line 77 of file `node_creation_library.cpp`.

Here is the caller graph for this function:



2.38.1.13 make_nport_gate()

```
nnode_t* make_nport_gate (
    operation_list type,
    int port_sizes,
    int width,
    int width_output,
    nnode_t * node,
    short mark )
```

Definition at line 220 of file node_creation_library.cpp.

Here is the caller graph for this function:



2.38.1.14 node_name()

```
char* node_name (
    nnode_t * node,
    char * instance_name_prefix )
```

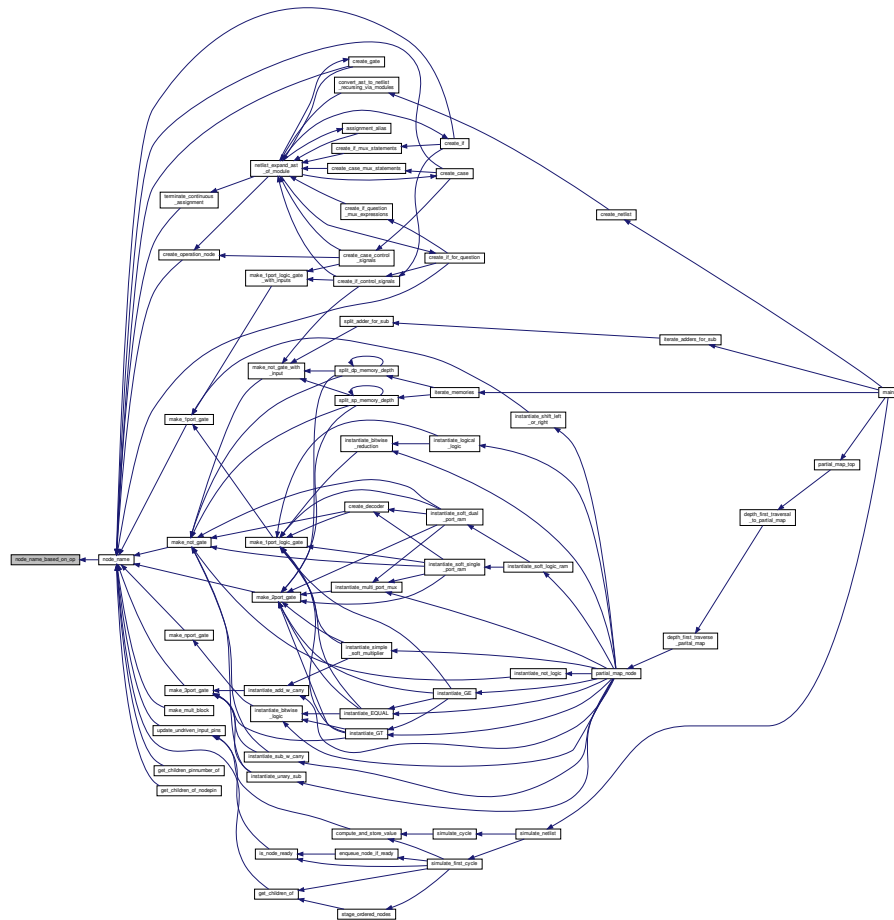
Definition at line 450 of file node_creation_library.cpp.

[illegible]

```
const char* node_name_based_on_op (
    nnode_t * node )
```

Generated on Thu Aug 10 2017 19:13:39 for Odin by Doxygen

Here is the caller graph for this function:



2.38.2 Variable Documentation

2.38.2.1 ADD_string

```
const char* ADD_string = "ADD"
```

Definition at line 251 of file node_creation_library.cpp.

2.38.2.2 ADDER_FUNC_string

```
const char* ADDER_FUNC_string = "ADDER_FUNC"
```

Definition at line 280 of file node_creation_library.cpp.

2.38.2.3 BITWISE_AND_string

```
const char* BITWISE_AND_string = "BITWISE_AND"
```

Definition at line 254 of file node_creation_library.cpp.

2.38.2.4 BITWISE_NAND_string

```
const char* BITWISE_NAND_string = "BITWISE_NAND"
```

Definition at line 256 of file node_creation_library.cpp.

2.38.2.5 BITWISE_NOR_string

```
const char* BITWISE_NOR_string = "BITWISE_NOR"
```

Definition at line 257 of file node_creation_library.cpp.

2.38.2.6 BITWISE_NOT_string

```
const char* BITWISE_NOT_string = "BITWISE_NOT"
```

Definition at line 253 of file node_creation_library.cpp.

2.38.2.7 BITWISE_OR_string

```
const char* BITWISE_OR_string = "BITWISE_OR"
```

Definition at line 255 of file node_creation_library.cpp.

2.38.2.8 BITWISE_XNOR_string

```
const char* BITWISE_XNOR_string = "BITWISE_XNOR"
```

Definition at line 258 of file node_creation_library.cpp.

2.38.2.9 BITWISE_XOR_string

```
const char* BITWISE_XOR_string = "BITWISE_XOR"
```

Definition at line 259 of file node_creation_library.cpp.

2.38.2.10 BUF_NODE_string

```
const char* BUF_NODE_string = "BUF_NODE"
```

Definition at line 245 of file node_creation_library.cpp.

2.38.2.11 CARRY_FUNC_string

```
const char* CARRY_FUNC_string = "CARRY_FUNC"
```

Definition at line 281 of file node_creation_library.cpp.

2.38.2.12 CASE_EQUAL_string

```
const char* CASE_EQUAL_string = "CASE_EQUAL"
```

Definition at line 278 of file node_creation_library.cpp.

2.38.2.13 CASE_NOT_EQUAL_string

```
const char* CASE_NOT_EQUAL_string = "CASE_NOT_EQUAL"
```

Definition at line 279 of file node_creation_library.cpp.

2.38.2.14 CLOCK_NODE_string

```
const char* CLOCK_NODE_string = "CLOCK_NODE"
```

Definition at line 247 of file node_creation_library.cpp.

2.38.2.15 DIVIDE_string

```
const char* DIVIDE_string = "DIVIDE"
```

Definition at line 268 of file node_creation_library.cpp.

2.38.2.16 FF_NODE_string

```
const char* FF_NODE_string = "FF_NODE"
```

Definition at line 244 of file node_creation_library.cpp.

2.38.2.17 GND_NODE_string

```
const char* GND_NODE_string = "GND_NODE"
```

Definition at line 249 of file node_creation_library.cpp.

2.38.2.18 GT_string

```
const char* GT_string = "GT"
```

Definition at line 271 of file node_creation_library.cpp.

2.38.2.19 GTE_string

```
const char* GTE_string = "GTE"
```

Definition at line 275 of file node_creation_library.cpp.

2.38.2.20 HARD_IP_string

```
const char* HARD_IP_string = "HARD_IP"
```

Definition at line 283 of file node_creation_library.cpp.

2.38.2.21 INPUT_NODE_string

```
const char* INPUT_NODE_string = "INPUT_NODE"
```

Definition at line 246 of file node_creation_library.cpp.

2.38.2.22 LOGICAL_AND_string

```
const char* LOGICAL_AND_string = "LOGICAL_AND"
```

Definition at line 262 of file node_creation_library.cpp.

2.38.2.23 LOGICAL_EQUAL_string

```
const char* LOGICAL_EQUAL_string = "LOGICAL_EQUAL"
```

Definition at line 272 of file node_creation_library.cpp.

2.38.2.24 LOGICAL_NAND_string

```
const char* LOGICAL_NAND_string = "LOGICAL_NAND"
```

Definition at line 263 of file node_creation_library.cpp.

2.38.2.25 LOGICAL_NOR_string

```
const char* LOGICAL_NOR_string = "LOGICAL_NOR"
```

Definition at line 264 of file node_creation_library.cpp.

2.38.2.26 LOGICAL_NOT_string

```
const char* LOGICAL_NOT_string = "LOGICAL_NOT"
```

Definition at line 260 of file node_creation_library.cpp.

2.38.2.27 LOGICAL_OR_string

```
const char* LOGICAL_OR_string = "LOGICAL_OR"
```

Definition at line 261 of file node_creation_library.cpp.

2.38.2.28 LOGICAL_XNOR_string

```
const char* LOGICAL_XNOR_string = "LOGICAL_XNOR"
```

Definition at line 266 of file node_creation_library.cpp.

2.38.2.29 LOGICAL_XOR_string

```
const char* LOGICAL_XOR_string = "LOGICAL_XOR"
```

Definition at line 265 of file node_creation_library.cpp.

2.38.2.30 LT_string

```
const char* LT_string = "LT"
```

Definition at line 270 of file node_creation_library.cpp.

2.38.2.31 LTE_string

```
const char* LTE_string = "LTE"
```

Definition at line 274 of file node_creation_library.cpp.

2.38.2.32 MEMORY_string

```
const char* MEMORY_string = "MEMORY"
```

Definition at line 284 of file node_creation_library.cpp.

2.38.2.33 MINUS_string

```
const char* MINUS_string = "MINUS"
```

Definition at line 252 of file node_creation_library.cpp.

2.38.2.34 MODULO_string

```
const char* MODULO_string = "MODULO"
```

Definition at line 269 of file node_creation_library.cpp.

2.38.2.35 MULTI_PORT_MUX_string

```
const char* MULTI_PORT_MUX_string = "MULTI_PORT_MUX"
```

Definition at line 243 of file node_creation_library.cpp.

2.38.2.36 MULTIPLY_string

```
const char* MULTIPLY_string = "MULTIPLY"
```

Definition at line 267 of file node_creation_library.cpp.

2.38.2.37 MUX_2_string

```
const char* MUX_2_string = "MUX_2"
```

Definition at line 282 of file node_creation_library.cpp.

2.38.2.38 NOT_EQUAL_string

```
const char* NOT_EQUAL_string = "NOT_EQUAL"
```

Definition at line 273 of file node_creation_library.cpp.

2.38.2.39 OUTPUT_NODE_string

```
const char* OUTPUT_NODE_string = "OUTPUT_NODE"
```

Definition at line 248 of file node_creation_library.cpp.

2.38.2.40 SL_string

```
const char* SL_string = "SL"
```

Definition at line 277 of file node_creation_library.cpp.

2.38.2.41 SR_string

```
const char* SR_string = "SR"
```

Definition at line 276 of file node_creation_library.cpp.

2.38.2.42 unique_node_name_id

```
long unique_node_name_id = 0
```

Definition at line 34 of file node_creation_library.cpp.

2.38.2.43 VCC_NODE_string

```
const char* VCC_NODE_string = "VCC_NODE"
```

Definition at line 250 of file node_creation_library.cpp.

2.39 vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/node_creation_library.h File Reference

```
#include "types.h"
```

Functions

- `nnode_t * make_not_gate_with_input` (`npin_t *input_pin`, `nnode_t *node`, short mark)
- `nnode_t * make_1port_logic_gate_with_inputs` (`operation_list` type, `int` width, `signal_list_t *pin_list`, `nnode_t *node`, short mark)
- `nnode_t * make_2port_logic_gates_with_inputs` (`operation_list` type, `int` width_port1, `signal_list_t *pin_list1`, `int` width_port2, `signal_list_t *pin_list2`, `nnode_t *node`, short mark)
- `nnode_t * make_not_gate` (`nnode_t *node`, short mark)
- `nnode_t * make_1port_logic_gate` (`operation_list` type, `int` width, `nnode_t *node`, short mark)
- `nnode_t * make_1port_gate` (`operation_list` type, `int` width_input, `int` width_output, `nnode_t *node`, short mark)
- `nnode_t * make_2port_gate` (`operation_list` type, `int` width_port1, `int` width_port2, `int` width_output, `nnode_t *node`, short mark)
- `nnode_t * make_3port_gate` (`operation_list` type, `int` width_port1, `int` width_port2, `int` width_port3, `int` width_output, `nnode_t *node`, short mark)
- `nnode_t * make_nport_gate` (`operation_list` type, `int` port_sizes, `int` width, `int` width_output, `nnode_t *node`, short mark)
- `npin_t * get_zero_pin` ()
- `npin_t * get_one_pin` ()
- `const char * node_name_based_on_op` (`nnode_t *node`)
- `char * node_name` (`nnode_t *node`, `char *instance_prefix_name`)
- `char * hard_node_name` (`nnode_t *node`, `char *instance_name_prefix`, `char *hb_name`, `char *hb_inst`)
- `nnode_t * make_mult_block` (`nnode_t *node`, short mark)

2.39.1 Function Documentation

2.39.1.1 `get_one_pin()`

```
npin_t* get_one_pin ( )
```

2.39.1.2 `get_zero_pin()`

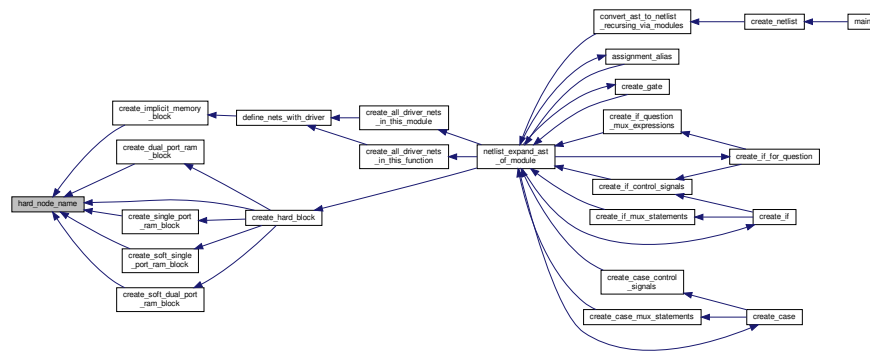
```
npin_t* get_zero_pin ( )
```

2.39.1.3 hard_node_name()

```
char* hard_node_name (
    nnode_t * node,
    char * instance_name_prefix,
    char * hb_name,
    char * hb_inst )
```

Definition at line 434 of file node_creation_library.cpp.

Here is the caller graph for this function:



2.39.1.4 make_1port_gate()

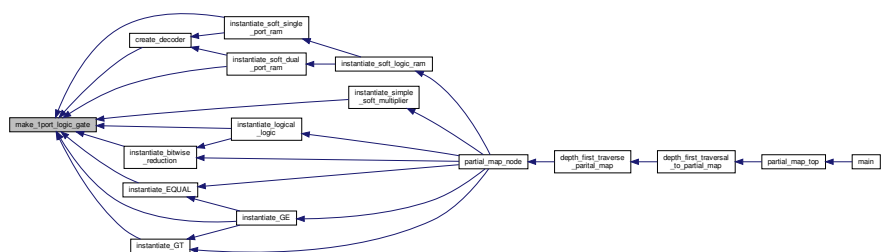
```
nnode_t* make_1port_gate (
    operation_list type,
    int width_input,
    int width_output,
    nnode_t * node,
    short mark )
```

Definition at line 115 of file node_creation_library.cpp.

[illegible]

```
nnode_t* make_lport_logic_gate (
    operation_list type,
    int width,
    nnode_t * node,
    short mark )
```

Here is the caller graph for this function:

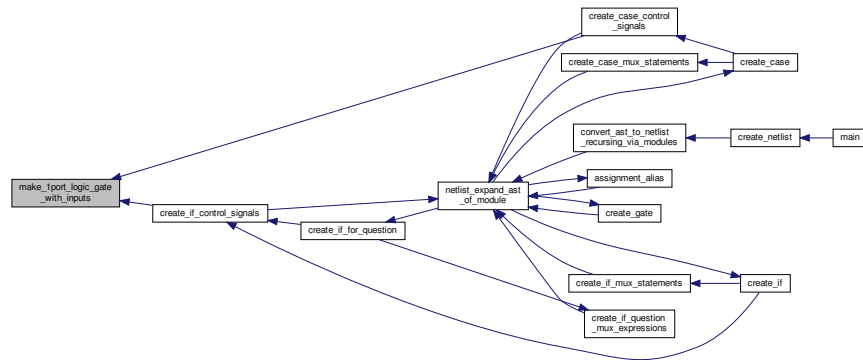


2.39.1.6 make_1port_logic_gate_with_inputs()

```
nnode_t* make_1port_logic_gate_with_inputs (
    operation_list type,
    int width,
    signal_list_t * pin_list,
    nnode_t * node,
    short mark )
```

Definition at line 149 of file node_creation_library.cpp.

Here is the caller graph for this function:

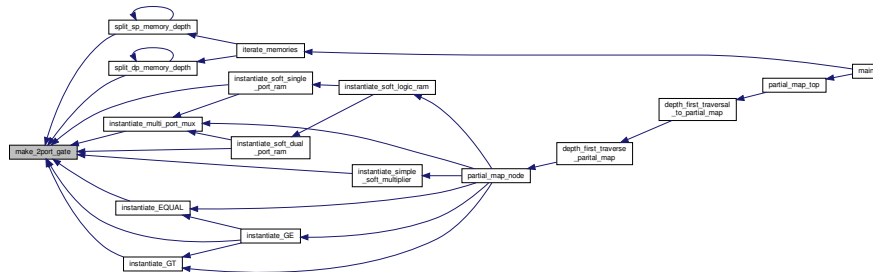


2.39.1.7 make_2port_gate()

```
nnode_t* make_2port_gate (
    operation_list type,
    int width_port1,
    int width_port2,
    int width_output,
    nnode_t * node,
    short mark )
```

Definition at line 196 of file node_creation_library.cpp.

Here is the caller graph for this function:



2.39.1.8 make_2port_logic_gates_with_inputs()

```

nnode_t* make_2port_logic_gates_with_inputs (
    operation_list type,
    int width_port1,
    signal_list_t * pin_list1,
    int width_port2,
    signal_list_t * pin_list2,
    nnode_t * node,
    short mark )

```

2.39.1.9 make_3port_gate()

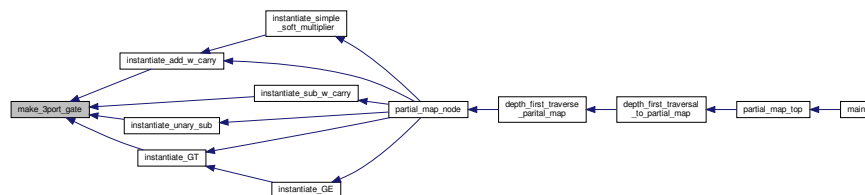
```

nnode_t* make_3port_gate (
    operation_list type,
    int width_port1,
    int width_port2,
    int width_port3,
    int width_output,
    nnode_t * node,
    short mark )

```

Definition at line 169 of file node_creation_library.cpp.

Here is the caller graph for this function:



2.39.1.10 make_mult_block()

```

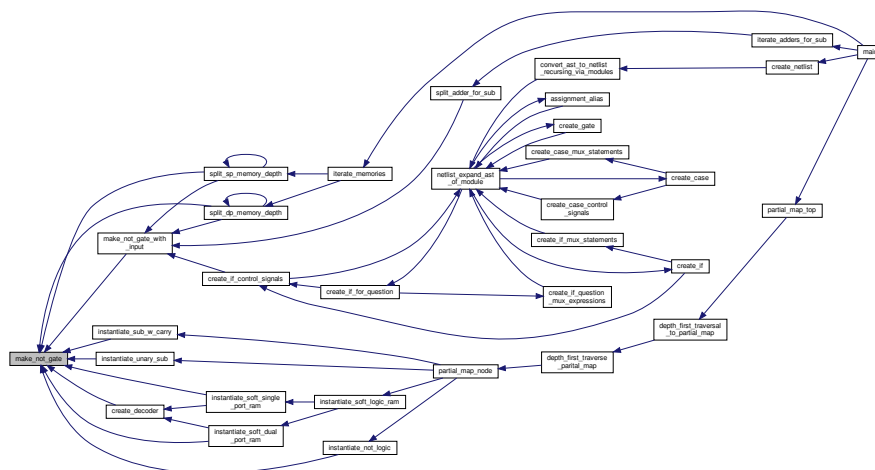
nnode_t* make_mult_block (
    nnode_t * node,
    short mark )

```

Definition at line 468 of file node_creation_library.cpp.

```
nnode_t* make_not_gate (
    nnode_t * node,
    short mark )
```

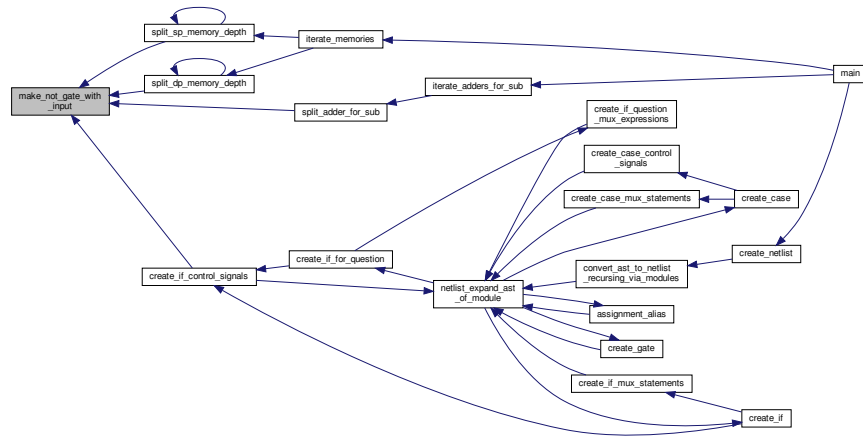
Here is the caller graph for this function:



```
nnode_t* make_not_gate_with_input (
    npin_t * input_pin,
    nnode_t * node,
    short mark )
```

Generated on Thu Aug 10 2017 19:13:39 for Odin by Doxygen

Here is the caller graph for this function:



2.39.1.13 make_nport_gate()

```

nnode_t* make_nport_gate (
    operation_list type,
    int port_sizes,
    int width,
    int width_output,
    nnode_t * node,
    short mark )

```

Definition at line 220 of file node_creation_library.cpp.

Here is the caller graph for this function:



2.39.1.14 node_name()

```

char* node_name (
    nnode_t * node,
    char * instance_prefix_name )

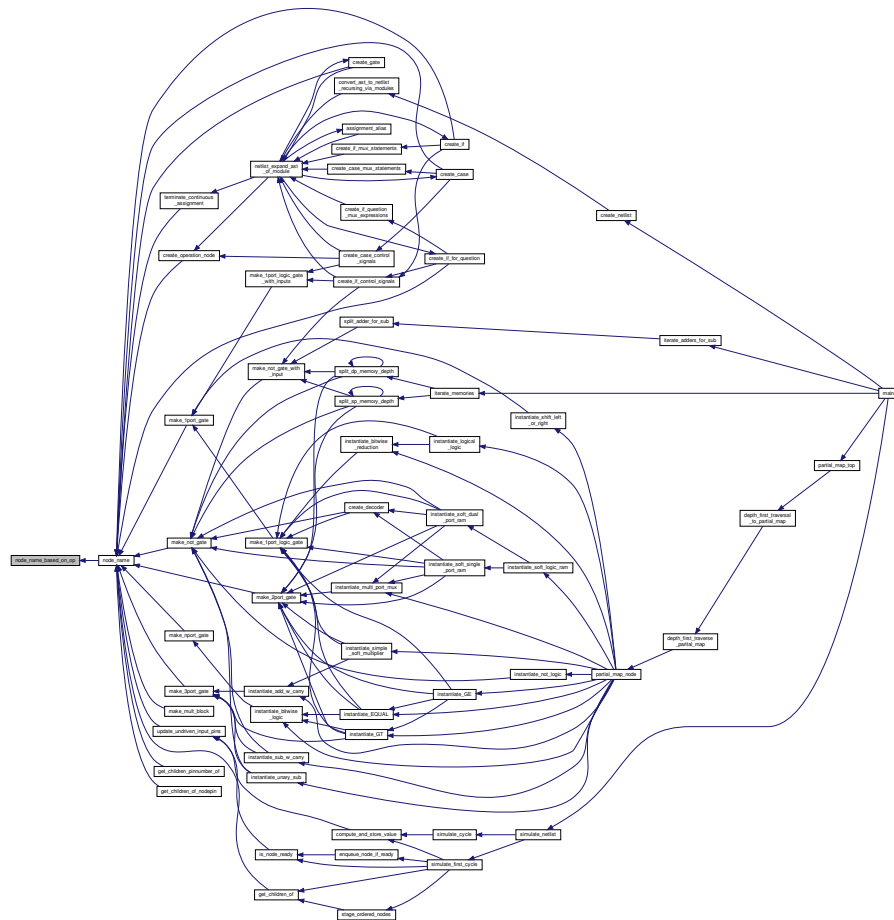
```

Definition at line 450 of file node_creation_library.cpp.


```
const char* node_name_based_on_op (
    nnode_t * node )
```

Generated on Thu Aug 10 2017 19:13:39 for Odin by Doxygen

Here is the caller graph for this function:



2.40 vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/partial_map.cpp File Reference

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include "types.h"
#include "globals.h"
#include "netlist_utils.h"
#include "node_creation_library.h"
#include "odin_util.h"
#include "partial_map.h"
#include "multipliers.h"
#include "hard_blocks.h"
#include "math.h"
#include "memories.h"
#include "adders.h"
#include "subtractions.h"
#include "vtr_memory.h"
```

Functions

- void [depth_first_traversal_to_partial_map](#) (short marker_value, netlist_t *netlist)
- void [depth_first_traverse_parital_map](#) (nnode_t *node, int traverse_mark_number, netlist_t *netlist)
- void [partial_map_node](#) (nnode_t *node, short traverse_number, netlist_t *netlist)
- void [instantiate_not_logic](#) (nnode_t *node, short mark, netlist_t *netlist)
- void [instantiate_buffer](#) (nnode_t *node, short mark, netlist_t *netlist)
- void [instantiate_bitwise_logic](#) (nnode_t *node, operation_list op, short mark, netlist_t *netlist)
- void [instantiate_bitwise_reduction](#) (nnode_t *node, operation_list op, short mark, netlist_t *netlist)
- void [instantiate_logical_logic](#) (nnode_t *node, operation_list op, short mark, netlist_t *netlist)
- void [instantiate_EQUAL](#) (nnode_t *node, short type, short mark, netlist_t *netlist)
- void [instantiate_GE](#) (nnode_t *node, short type, short mark, netlist_t *netlist)
- void [instantiate_GT](#) (nnode_t *node, short type, short mark, netlist_t *netlist)
- void [instantiate_shift_left_or_right](#) (nnode_t *node, short type, short mark, netlist_t *netlist)
- void [instantiate_unary_sub](#) (nnode_t *node, short mark, netlist_t *netlist)
- void [instantiate_sub_w_carry](#) (nnode_t *node, short mark, netlist_t *netlist)
- void [instantiate_soft_logic_ram](#) (nnode_t *node, short mark, netlist_t *netlist)
- void [partial_map_top](#) (netlist_t *netlist)
- void [instantiate_multi_port_mux](#) (nnode_t *node, short mark, netlist_t *)
- void [instantiate_add_w_carry](#) (nnode_t *node, short mark, netlist_t *netlist)

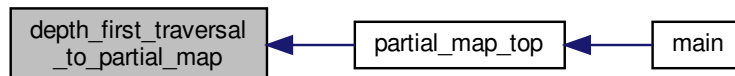
2.40.1 Function Documentation

2.40.1.1 depth_first_traversal_to_partial_map()

```
void depth_first_traversal_to_partial_map (
    short marker_value,
    netlist_t * netlist )
```

Definition at line 79 of file partial_map.cpp.

Here is the caller graph for this function:

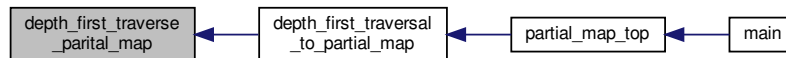


2.40.1.2 depth_first_traverse_parital_map()

```
void depth_first_traverse_parital_map (  
    nnode_t * node,  
    int traverse_mark_number,  
    netlist_t * netlist )
```

Definition at line 100 of file partial_map.cpp.

Here is the caller graph for this function:



2.40.1.3 instantiate_add_w_carry()

```
void instantiate_add_w_carry (  
    nnode_t * node,  
    short mark,  
    netlist_t * netlist )
```

Definition at line 642 of file partial_map.cpp.

Here is the caller graph for this function:

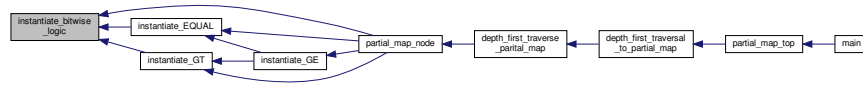


2.40.1.4 instantiate_bitwise_logic()

```
void instantiate_bitwise_logic (  
    nnode_t * node,  
    operation_list op,  
    short mark,  
    netlist_t * netlist )
```

Definition at line 554 of file partial_map.cpp.

Here is the caller graph for this function:



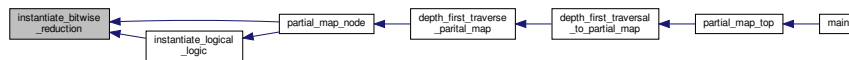
2.40.1.5 instantiate_bitwise_reduction()

```

void instantiate_bitwise_reduction (
    nnode_t * node,
    operation_list op,
    short mark,
    netlist_t * netlist )
  
```

Definition at line 485 of file partial_map.cpp.

Here is the caller graph for this function:



2.40.1.6 instantiate_buffer()

```

void instantiate_buffer (
    nnode_t * node,
    short mark,
    netlist_t * netlist )
  
```

Definition at line 401 of file partial_map.cpp.

Here is the caller graph for this function:

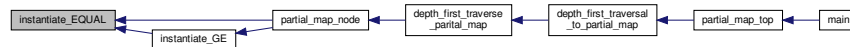


2.40.1.7 instantiate_EQUAL()

```
void instantiate_EQUAL (
    nnode_t * node,
    short type,
    short mark,
    netlist_t * netlist )
```

Definition at line 1006 of file partial_map.cpp.

Here is the caller graph for this function:

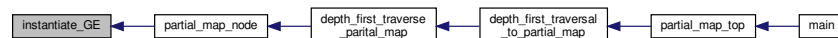


2.40.1.8 instantiate_GE()

```
void instantiate_GE (
    nnode_t * node,
    short type,
    short mark,
    netlist_t * netlist )
```

Definition at line 1242 of file partial_map.cpp.

Here is the caller graph for this function:

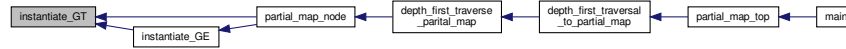


2.40.1.9 instantiate_GT()

```
void instantiate_GT (
    nnode_t * node,
    short type,
    short mark,
    netlist_t * netlist )
```

Definition at line 1092 of file partial_map.cpp.

Here is the caller graph for this function:



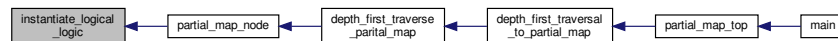
2.40.1.10 `instantiate_logical_logic()`

```

void instantiate_logical_logic (
    nnode_t * node,
    operation_list op,
    short mark,
    netlist_t * netlist )
  
```

Definition at line 422 of file `partial_map.cpp`.

Here is the caller graph for this function:



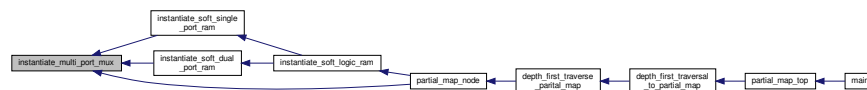
2.40.1.11 `instantiate_multi_port_mux()`

```

void instantiate_multi_port_mux (
    nnode_t * node,
    short mark,
    netlist_t * )
  
```

Definition at line 324 of file `partial_map.cpp`.

Here is the caller graph for this function:

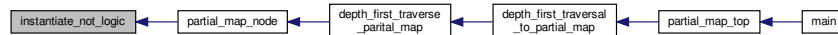


2.40.1.12 instantiate_not_logic()

```
void instantiate_not_logic (  
    nnode_t * node,  
    short mark,  
    netlist_t * netlist )
```

Definition at line 373 of file partial_map.cpp.

Here is the caller graph for this function:

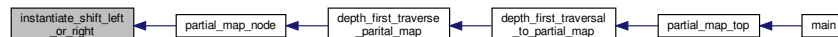


2.40.1.13 instantiate_shift_left_or_right()

```
void instantiate_shift_left_or_right (  
    nnode_t * node,  
    short type,  
    short mark,  
    netlist_t * netlist )
```

Definition at line 1323 of file partial_map.cpp.

Here is the caller graph for this function:

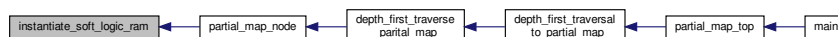


2.40.1.14 instantiate_soft_logic_ram()

```
void instantiate_soft_logic_ram (  
    nnode_t * node,  
    short mark,  
    netlist_t * netlist )
```

Definition at line 309 of file partial_map.cpp.

Here is the caller graph for this function:

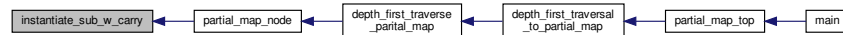


2.40.1.15 instantiate_sub_w_carry()

```
void instantiate_sub_w_carry (
    nnode_t * node,
    short mark,
    netlist_t * netlist )
```

Definition at line 765 of file partial_map.cpp.

Here is the caller graph for this function:

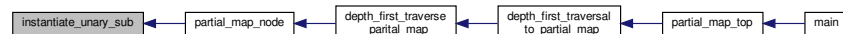


2.40.1.16 instantiate_unary_sub()

```
void instantiate_unary_sub (
    nnode_t * node,
    short mark,
    netlist_t * netlist )
```

Definition at line 896 of file partial_map.cpp.

Here is the caller graph for this function:

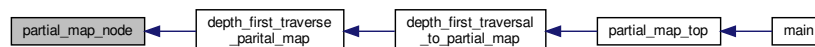


2.40.1.17 partial_map_node()

```
void partial_map_node (
    nnode_t * node,
    short traverse_number,
    netlist_t * netlist )
```

Definition at line 141 of file partial_map.cpp.

Here is the caller graph for this function:



2.40.1.18 partial_map_top()

```
void partial_map_top (
    netlist_t * netlist )
```

Definition at line 66 of file partial_map.cpp.

Here is the caller graph for this function:



2.41 vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/partial_map.h File Reference

Functions

- void [partial_map_top](#) (netlist_t *netlist)
- void [instantiate_add_w_carry](#) (nnode_t *node, short mark, netlist_t *netlist)
- void [instantiate_multi_port_mux](#) (nnode_t *node, short mark, netlist_t *netlist)

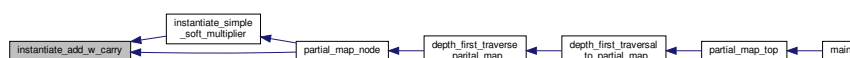
2.41.1 Function Documentation

2.41.1.1 instantiate_add_w_carry()

```
void instantiate_add_w_carry (
    nnode_t * node,
    short mark,
    netlist_t * netlist )
```

Definition at line 642 of file partial_map.cpp.

Here is the caller graph for this function:

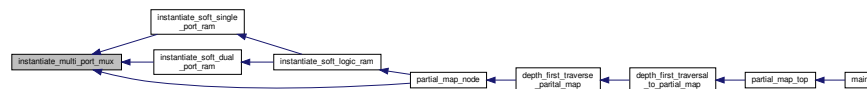


2.41.1.2 instantiate_multi_port_mux()

```
void instantiate_multi_port_mux (
    nnode_t * node,
    short mark,
    netlist_t * netlist )
```

Definition at line 324 of file partial_map.cpp.

Here is the caller graph for this function:



2.41.1.3 partial_map_top()

```
void partial_map_top (
    netlist_t * netlist )
```

Definition at line 66 of file partial_map.cpp.

Here is the caller graph for this function:



2.42 vtr-verilog-to-routing/ODIN_II/SRC/odin_ii.cpp File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <sstream>
#include "vtr_error.h"
#include "argparse.hpp"
#include "globals.h"
#include "types.h"
#include "netlist_utils.h"
```

```
#include "arch_types.h"
#include "parse_making_ast.h"
#include "netlist_create_from_ast.h"
#include "ast_util.h"
#include "read_xml_config_file.h"
#include "read_xml_arch_file.h"
#include "partial_map.h"
#include "multipliers.h"
#include "netlist_check.h"
#include "read_blif.h"
#include "output_blif.h"
#include "netlist_cleanup.h"
#include "hard_blocks.h"
#include "memories.h"
#include "simulate_blif.h"
#include "netlist_visualizer.h"
#include "adders.h"
#include "subtractions.h"
#include "vtr_util.h"
```

Data Structures

- struct [ParseInitRegState](#)

Functions

- void [set_default_config](#) ()
- void [get_options](#) (int argc, char **argv)
- void [print_usage](#) ()
- int [main](#) (int argc, char **argv)

Variables

- size_t [current_parse_file](#)
- t_arch [Arch](#)
- global_args_t [global_args](#)
- t_type_descriptor * [type_descriptors](#)
- int [block_tag](#)

2.42.1 Function Documentation

2.42.1.1 `get_options()`

```
void get_options (
    int argc,
    char ** argv )
```

Definition at line 298 of file `odin_ii.cpp`.

Here is the caller graph for this function:



2.42.1.2 `main()`

```
int main (
    int argc,
    char ** argv )
```

Definition at line 67 of file `odin_ii.cpp`.

2.42.1.3 `print_usage()`

```
void print_usage ( )
```

Definition at line 222 of file `odin_ii.cpp`.

2.42.1.4 `set_default_config()`

```
void set_default_config ( )
```

Definition at line 439 of file `odin_ii.cpp`.

Here is the caller graph for this function:



2.42.2 Variable Documentation

2.42.2.1 Arch

`t_arch Arch`

Definition at line 58 of file `odin_ii.cpp`.

2.42.2.2 block_tag

`int block_tag`

Definition at line 61 of file `odin_ii.cpp`.

2.42.2.3 current_parse_file

`size_t current_parse_file`

Definition at line 57 of file `odin_ii.cpp`.

2.42.2.4 global_args

`global_args_t global_args`

Definition at line 59 of file `odin_ii.cpp`.

2.42.2.5 type_descriptors

`t_type_descriptor* type_descriptors`

Definition at line 60 of file `odin_ii.cpp`.

2.43 vtr-verilog-to-routing/ODIN_II/SRC/odin_util.cpp File Reference

```
#include <string>
#include <sstream>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <limits.h>
#include <errno.h>
#include "types.h"
#include "globals.h"
#include "odin_util.h"
#include "vtr_util.h"
#include "vtr_memory.h"
#include <regex>
#include <stdbool.h>
```

Functions

- char * [make_signal_name](#) (char *signal_name, int bit)
- char * [make_full_ref_name](#) (const char *previous, char *module_name, char *module_instance_name, const char *signal_name, long bit)
- char * [twos_complement](#) (char *str)
- char * [convert_string_of_radix_to_bit_string](#) (char *string, int radix, int binary_size)
- char * [convert_long_long_to_bit_string](#) (long long orig_long, int num_bits)
- long long [convert_dec_string_of_size_to_long_long](#) (char *orig_string, int)
- long long [convert_string_of_radix_to_long_long](#) (char *orig_string, int radix)
- int [is_string_of_radix](#) (char *string, int radix)
- char * [convert_hex_string_of_size_to_bit_string](#) (short is_dont_care_number, char *orig_string, int binary_size)
- char * [convert_oct_string_of_size_to_bit_string](#) (char *orig_string, int binary_size)
- char * [convert_binary_string_of_size_to_bit_string](#) (short is_dont_care_number, char *orig_string, int binary_size)
- int [is_hex_string](#) (char *string)
- int [is_dont_care_string](#) (char *string)
- int [is_decimal_string](#) (char *string)
- int [is_octal_string](#) (char *string)
- int [is_binary_string](#) (char *string)
- char * [get_pin_name](#) (char *name)
- char * [get_port_name](#) (char *name)
- int [get_pin_number](#) (char *name)
- long long int [my_power](#) (long long int x, long long int y)
- char * [make_string_based_on_id](#) (nnode_t *node)
- char * [make_simple_name](#) (char *input, const char *flatten_string, char flatten_char)
- void * [my_malloc_struct](#) (size_t bytes_to_alloc)
- long long int [pow2](#) (int to_the_power)
- void [string_to_upper](#) (char *string)
- void [string_to_lower](#) (char *string)
- char * [append_string](#) (const char *string, const char *appendage,...)

- void [reverse_string](#) (char *string, int length)
- short [get_bit](#) (char in)
- void [error_message](#) (short error_type, int line_number, int file, const char *message,...)
- void [warning_message](#) (short, int line_number, int file, const char *message,...)
- char * [search_replace](#) (char *src, const char *sKey, const char *rKey, int flag)
- char * [find_substring](#) (char *src, const char *sKey, int flag)
- bool [validate_string_regex](#) (const char *str_in, const char *pattern_in)

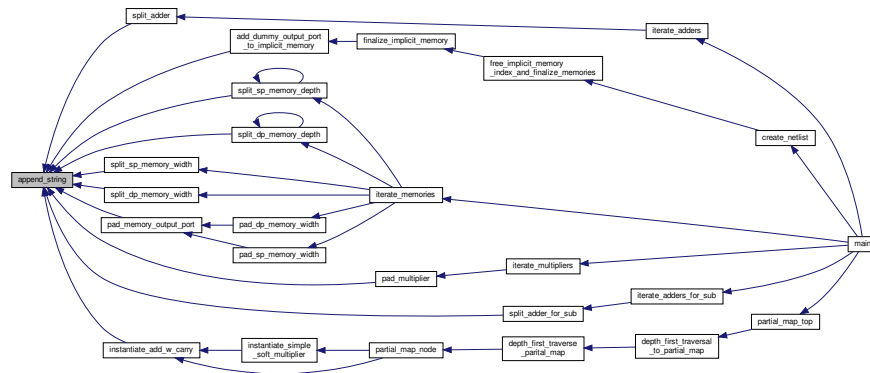
2.43.1 Function Documentation

2.43.1.1 [append_string\(\)](#)

```
char* append_string (
    const char * string,
    const char * appendage,
    ... )
```

Definition at line 647 of file `odin_util.cpp`.

Here is the caller graph for this function:

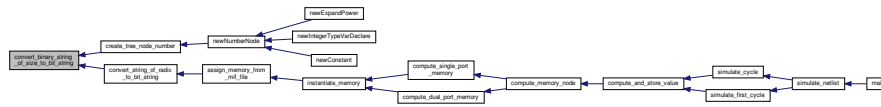


2.43.1.2 [convert_binary_string_of_size_to_bit_string\(\)](#)

```
char* convert_binary_string_of_size_to_bit_string (
    short is_dont_care_number,
    char * orig_string,
    int binary_size )
```

Definition at line 366 of file `odin_util.cpp`.

Here is the caller graph for this function:

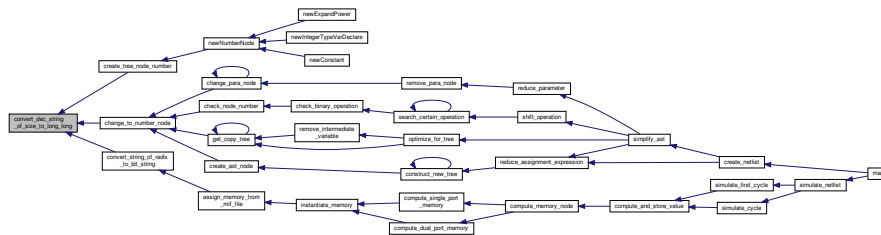


2.43.1.3 convert_dec_string_of_size_to_long_long()

```
long long convert_dec_string_of_size_to_long_long (
    char * orig_string,
    int )
```

Definition at line 158 of file `odin_util.cpp`.

Here is the caller graph for this function:

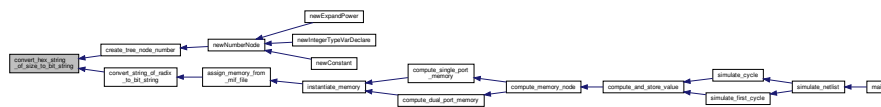


2.43.1.4 convert_hex_string_of_size_to_bit_string()

```
char* convert_hex_string_of_size_to_bit_string (
    short is_dont_care_number,
    char * orig_string,
    int binary_size )
```

Definition at line 207 of file `odin_util.cpp`.

Here is the caller graph for this function:

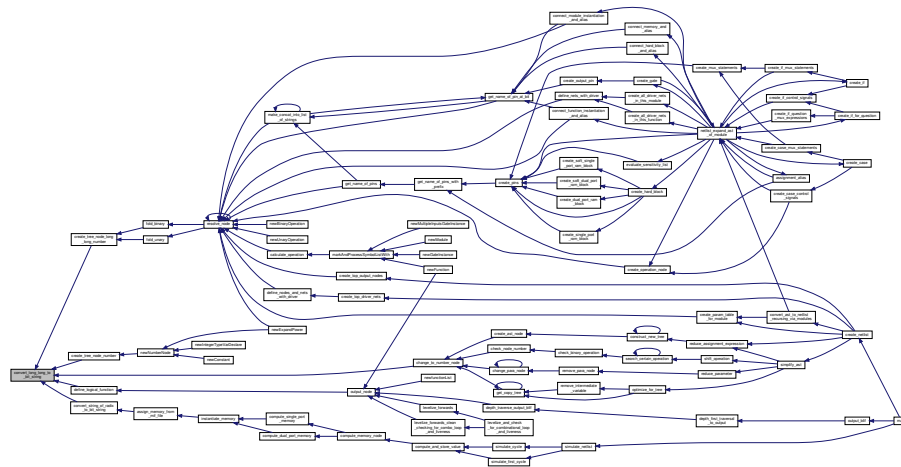


2.43.1.5 convert_long_long_to_bit_string()

```
char* convert_long_long_to_bit_string (
    long long orig_long,
    int num_bits )
```

Definition at line 137 of file `odin_util.cpp`.

Here is the caller graph for this function:

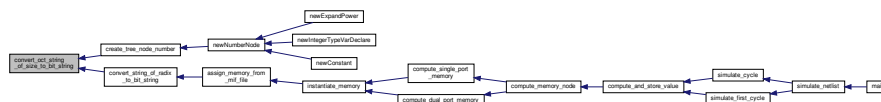


2.43.1.6 convert_oct_string_of_size_to_bit_string()

```
char* convert_oct_string_of_size_to_bit_string (
    char * orig_string,
    int binary_size )
```

Definition at line 313 of file `odin_util.cpp`.

Here is the caller graph for this function:

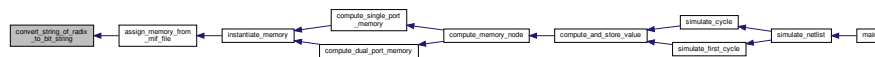


2.43.1.7 convert_string_of_radix_to_bit_string()

```
char* convert_string_of_radix_to_bit_string (
    char * string,
    int radix,
    int binary_size )
```

Definition at line 108 of file odin_util.cpp.

Here is the caller graph for this function:



2.43.1.8 convert_string_of_radix_to_long_long()

```
long long convert_string_of_radix_to_long_long (
    char * orig_string,
    int radix )
```

Definition at line 171 of file odin_util.cpp.

Here is the caller graph for this function:

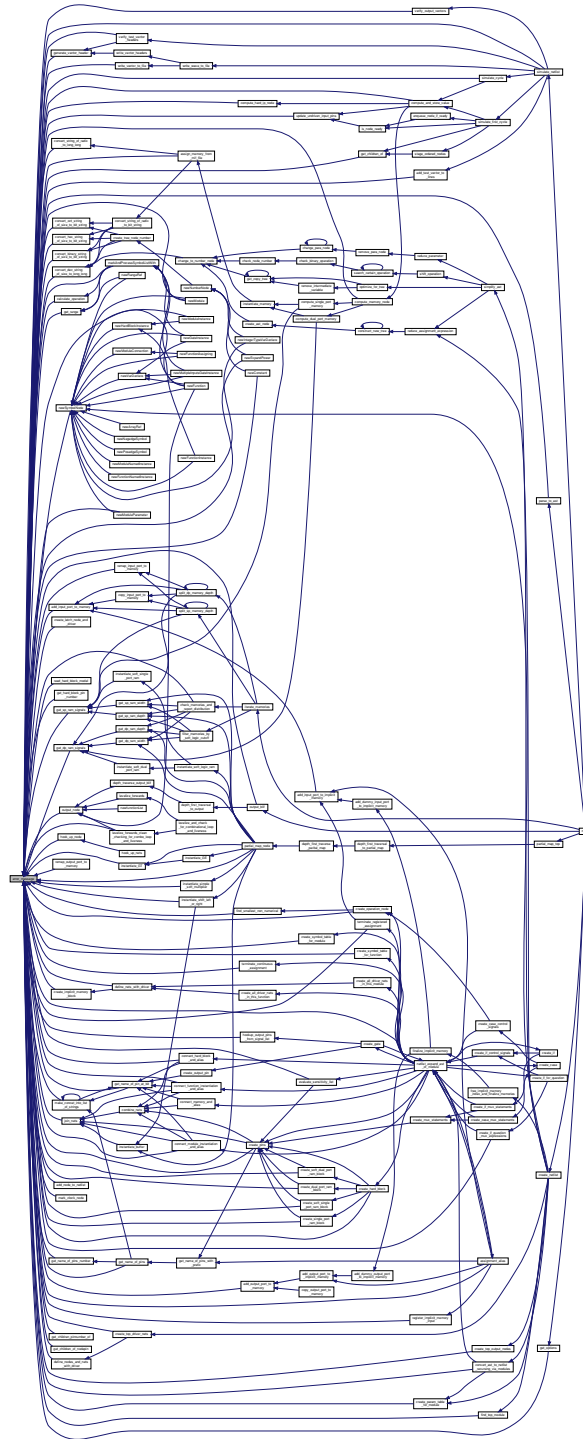


2.43.1.9 error_message()

```
void error_message (
    short error_type,
    int line_number,
    int file,
    const char * message,
    ... )
```

Definition at line 691 of file odin_util.cpp.

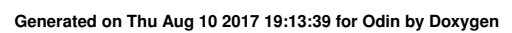
Here is the caller graph for this function:



2.43.1.10 find_substring()

```
char* find_substring (
```

Here is the caller graph for this function:

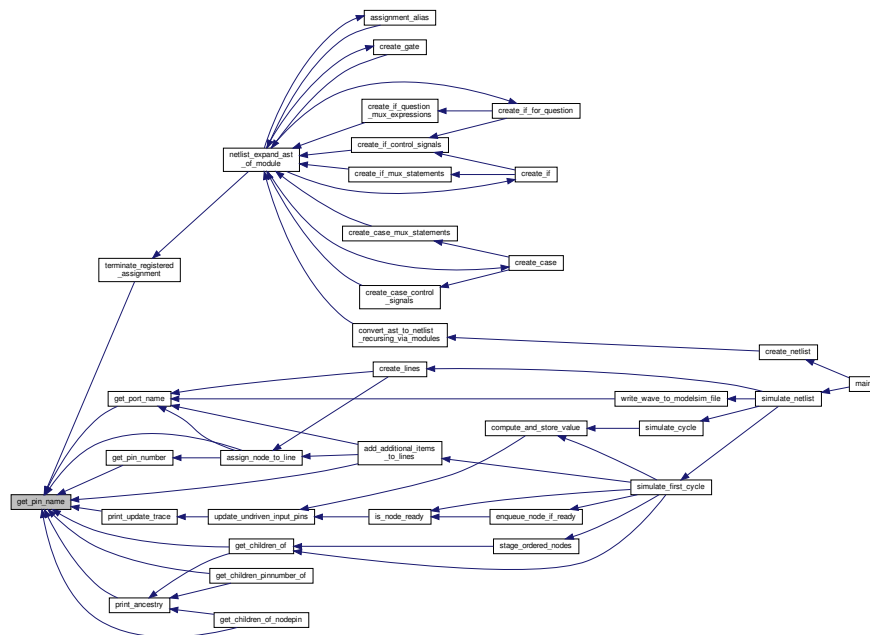


2.43.1.12 get_pin_name()

```
char* get_pin_name (
    char * name )
```

Definition at line 469 of file odin_util.cpp.

Here is the caller graph for this function:

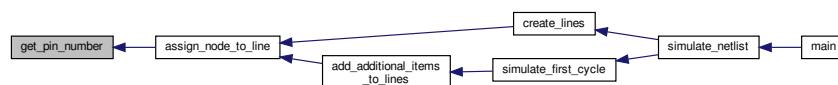


2.43.1.13 get_pin_number()

```
int get_pin_number (
    char * name )
```

Definition at line 499 of file odin_util.cpp.

Here is the caller graph for this function:

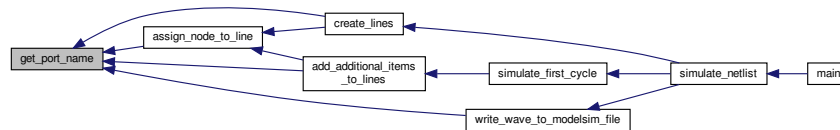


2.43.1.14 get_port_name()

```
char* get_port_name (
    char * name )
```

Definition at line 482 of file odin_util.cpp.

Here is the caller graph for this function:

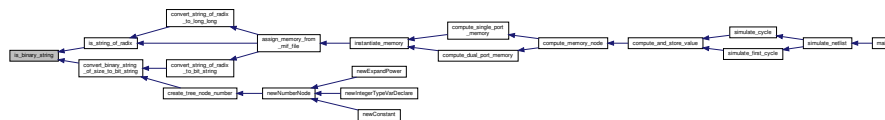


2.43.1.15 is_binary_string()

```
int is_binary_string (
    char * string )
```

Definition at line 453 of file odin_util.cpp.

Here is the caller graph for this function:

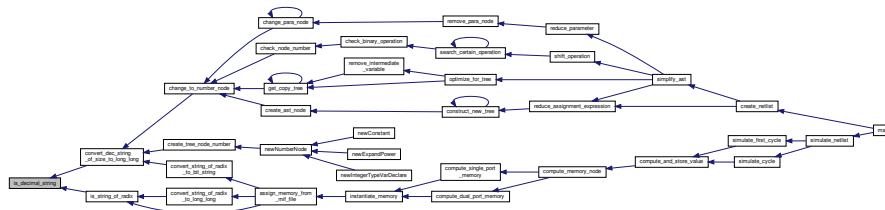


2.43.1.16 is_decimal_string()

```
int is_decimal_string (
    char * string )
```

Definition at line 427 of file odin_util.cpp.

Here is the caller graph for this function:

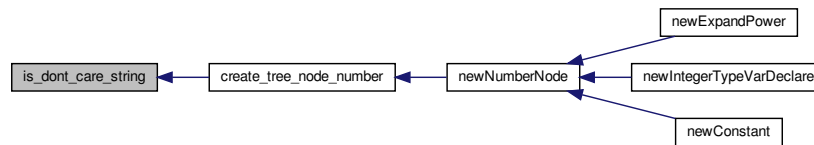


2.43.1.17 is_dont_care_string()

```
int is_dont_care_string (
    char * string )
```

Definition at line 414 of file odin_util.cpp.

Here is the caller graph for this function:

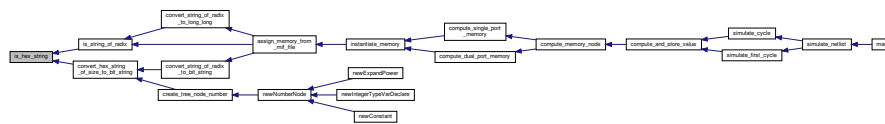


2.43.1.18 is_hex_string()

```
int is_hex_string (
    char * string )
```

Definition at line 401 of file odin_util.cpp.

Here is the caller graph for this function:

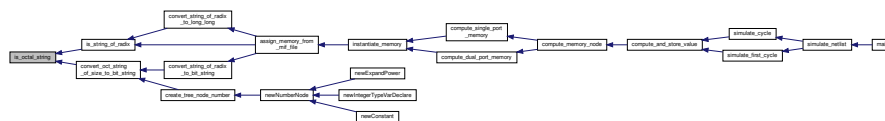


2.43.1.19 is_octal_string()

```
int is_octal_string (
    char * string )
```

Definition at line 440 of file odin_util.cpp.

Here is the caller graph for this function:

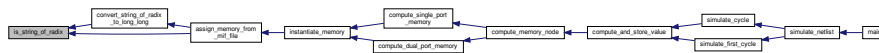


2.43.1.20 is_string_of_radix()

```
int is_string_of_radix (
    char * string,
    int radix )
```

Definition at line 189 of file odin_util.cpp.

Here is the caller graph for this function:

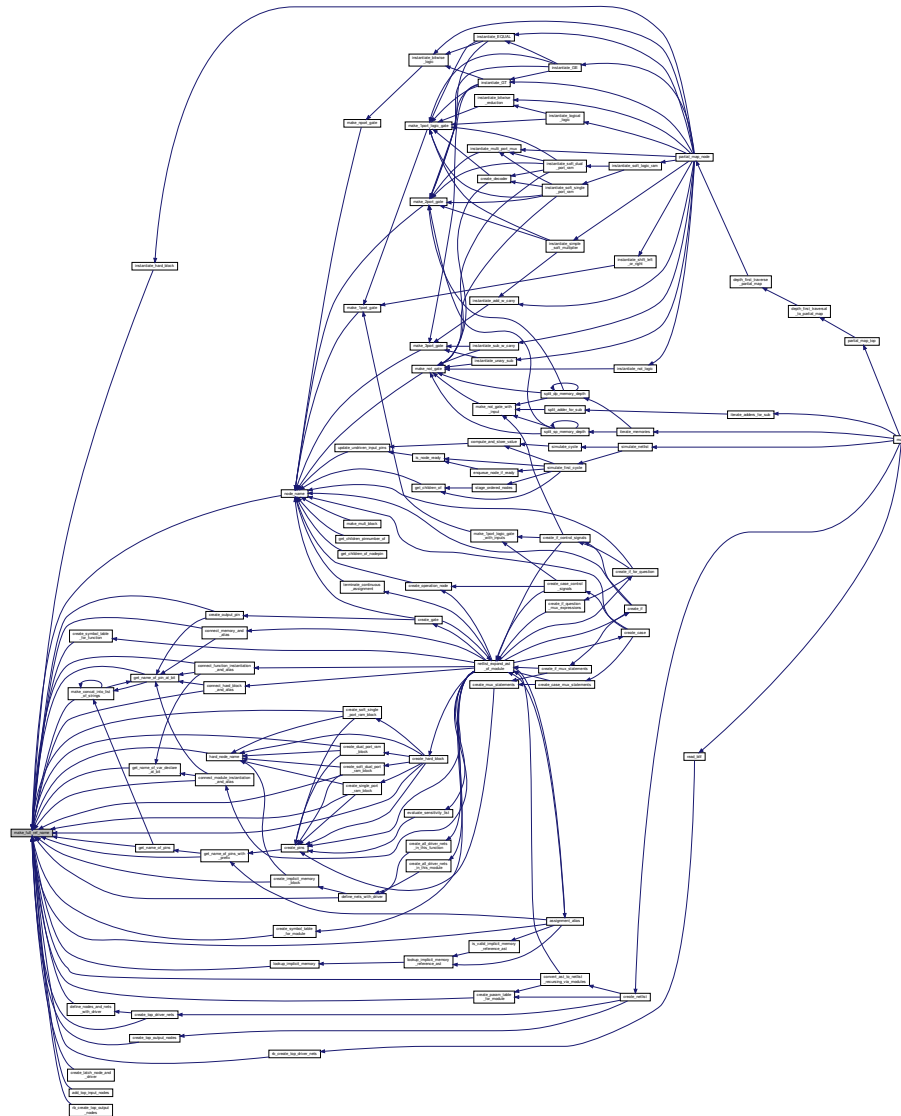


2.43.1.21 make_full_ref_name()

```
char* make_full_ref_name (
    const char * previous,
    char * module_name,
    char * module_instance_name,
    const char * signal_name,
    long bit )
```

Definition at line 61 of file odin_util.cpp.

Here is the caller graph for this function:

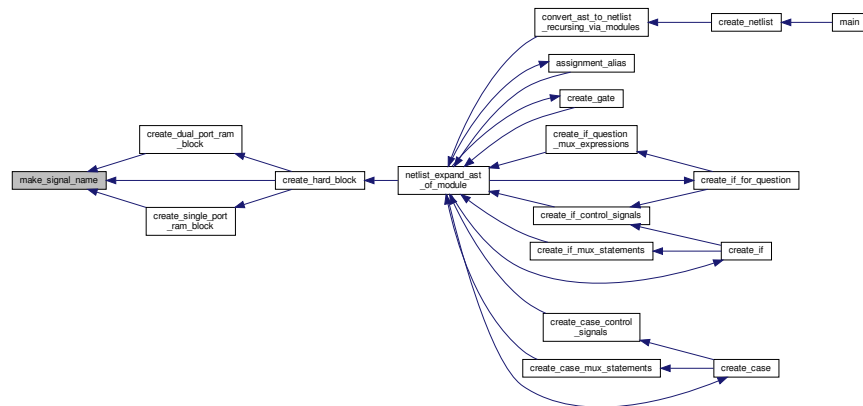


2.43.1.22 make_signal_name()

```
char* make_signal_name (
    char * signal_name,
    int bit )
```

Definition at line 46 of file odin_util.cpp.

Here is the caller graph for this function:



2.43.1.23 make_simple_name()

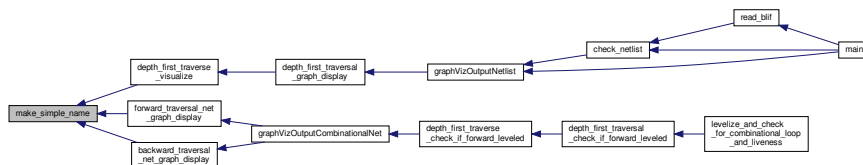
```

char* make_simple_name (
    char * input,
    const char * flatten_string,
    char flatten_char )

```

Definition at line 544 of file odin_util.cpp.

Here is the caller graph for this function:



2.43.1.24 make_string_based_on_id()

```

char* make_string_based_on_id (
    nnode_t * node )

```

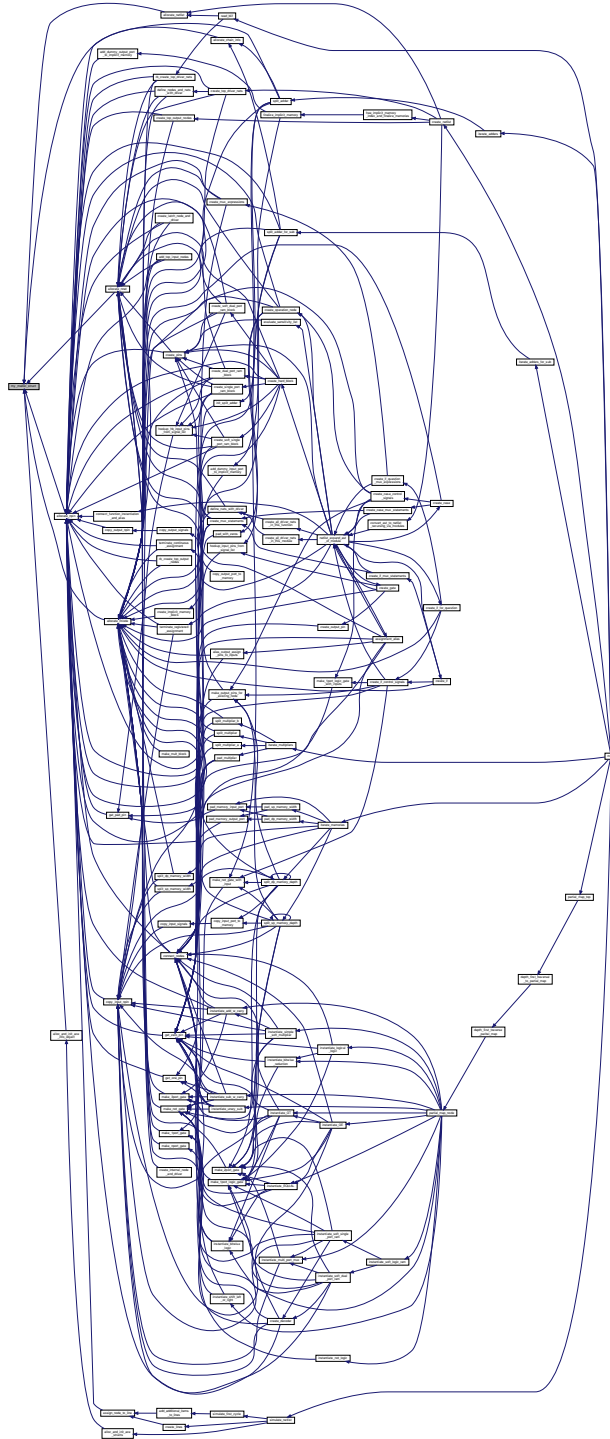
Definition at line 533 of file odin_util.cpp.

2.43.1.25 my_malloc_struct()

```
void* my_malloc_struct (
    size_t bytes_to_alloc )
```

Definition at line 574 of file odin_util.cpp.

Here is the caller graph for this function:

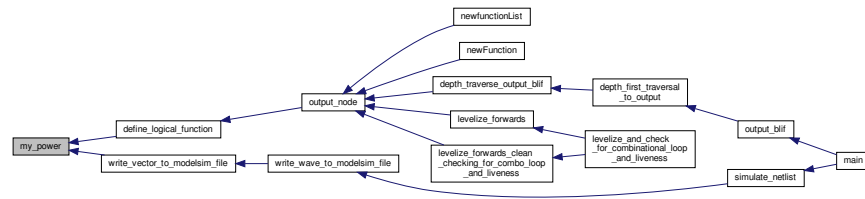


2.43.1.26 my_power()

```
long long int my_power (
    long long int x,
    long long int y )
```

Definition at line 517 of file odin_util.cpp.

Here is the caller graph for this function:



2.43.1.27 pow2()

```
long long int pow2 (
    int to_the_power )
```

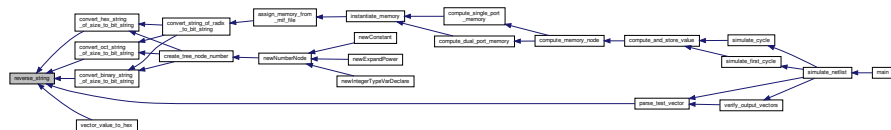
Definition at line 597 of file odin_util.cpp.

2.43.1.28 reverse_string()

```
void reverse_string (
    char * string,
    int length )
```

Definition at line 665 of file odin_util.cpp.

Here is the caller graph for this function:

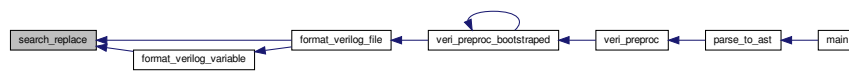


2.43.1.29 search_replace()

```
char* search_replace (
    char * src,
    const char * sKey,
    const char * rKey,
    int flag )
```

Definition at line 748 of file odin_util.cpp.

Here is the caller graph for this function:



2.43.1.30 string_to_lower()

```
void string_to_lower (
    char * string )
```

Definition at line 628 of file odin_util.cpp.

2.43.1.31 string_to_upper()

```
void string_to_upper (
    char * string )
```

Definition at line 613 of file odin_util.cpp.

Here is the caller graph for this function:

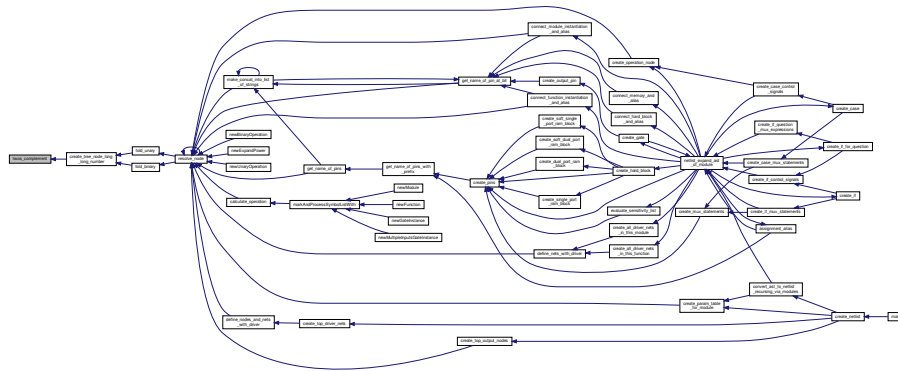


2.43.1.32 twos_complement()

```
char* twos_complement (
    char * str )
```

Definition at line 80 of file odin_util.cpp.

Here is the caller graph for this function:

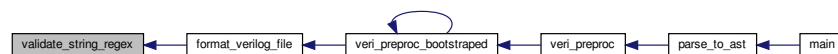


2.43.1.33 validate_string_regex()

```
bool validate_string_regex (
    const char * str_in,
    const char * pattern_in )
```

Definition at line 797 of file odin_util.cpp.

Here is the caller graph for this function:

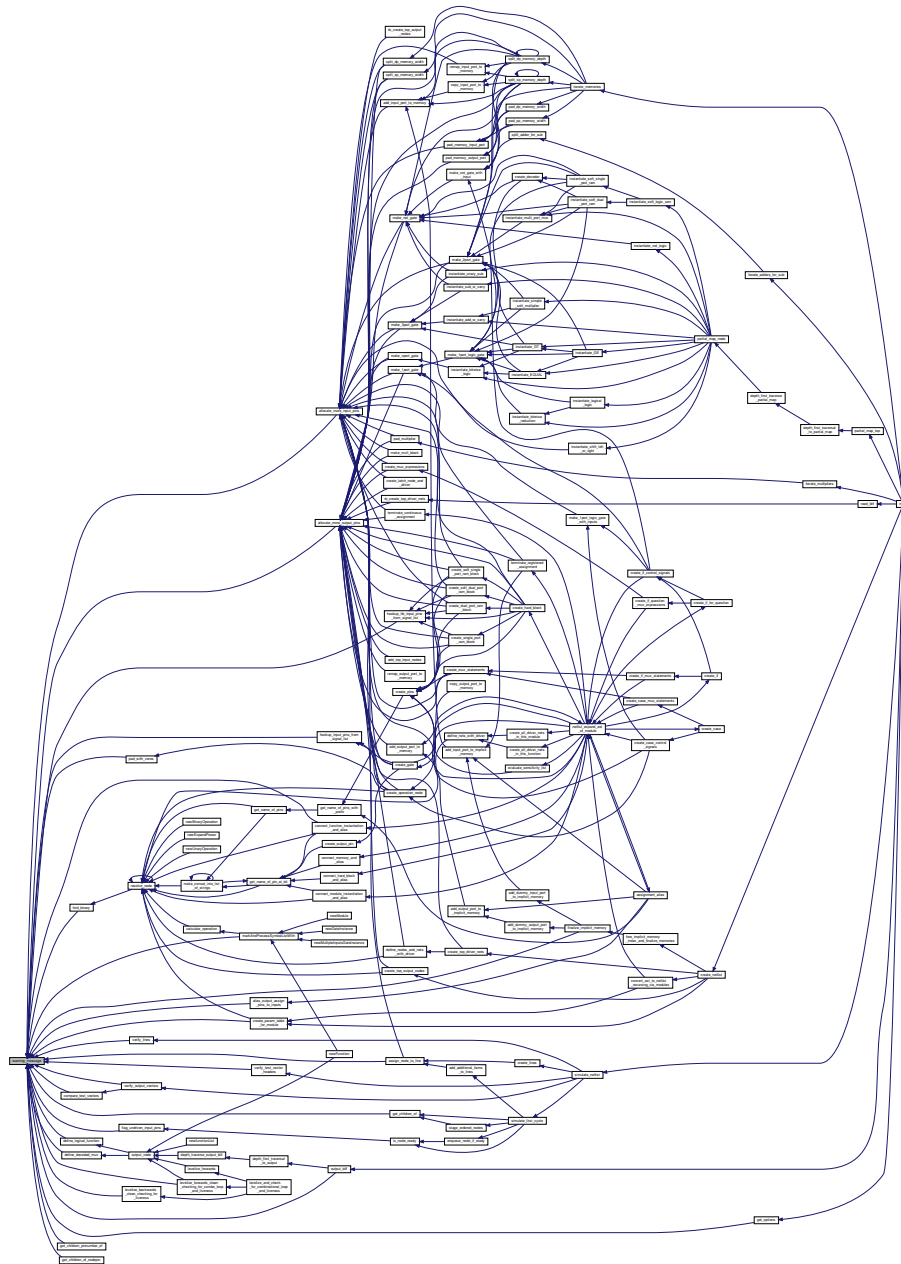


2.43.1.34 warning_message()

```
void warning_message (  
    short ,  
    int line_number,  
    int file,  
    const char * message,  
    ... )
```

Definition at line 718 of file odin_util.cpp.

Here is the caller graph for this function:



2.44 vtr-verilog-to-routing/ODIN_II/SRC/odin_util.h File Reference

```
#include "types.h"
```

Macros

- `#define MAX_BUF 256`

Functions

- `char * make_signal_name` (char *signal_name, int bit)
- `char * make_full_ref_name` (const char *previous, char *module_name, char *module_instance_name, const char *signal_name, long bit)
- `char * twos_complement` (char *str)
- `int is_string_of_radix` (char *string, int radix)
- `char * convert_string_of_radix_to_bit_string` (char *string, int radix, int binary_size)
- `long long convert_string_of_radix_to_long_long` (char *orig_string, int radix)
- `char * convert_long_long_to_bit_string` (long long orig_long, int num_bits)
- `long long convert_dec_string_of_size_to_long_long` (char *orig_string, int size)
- `char * convert_hex_string_of_size_to_bit_string` (short is_dont_care_number, char *orig_string, int size)
- `char * convert_oct_string_of_size_to_bit_string` (char *orig_string, int size)
- `char * convert_binary_string_of_size_to_bit_string` (short is_dont_care_number, char *orig_string, int binary_size)
- `long long int my_power` (long long int x, long long int y)
- `long long int pow2` (int to_the_power)
- `char * make_string_based_on_id` (nnode_t *node)
- `char * make_simple_name` (char *input, const char *flatten_string, char flatten_char)
- `void * my_malloc_struct` (size_t bytes_to_alloc)
- `void reverse_string` (char *token, int length)
- `char * append_string` (const char *string, const char *appendage,...)
- `void string_to_upper` (char *string)
- `void string_to_lower` (char *string)
- `int is_binary_string` (char *string)
- `int is_octal_string` (char *string)
- `int is_decimal_string` (char *string)
- `int is_hex_string` (char *string)
- `int is_dont_care_string` (char *string)
- `char * get_pin_name` (char *name)
- `char * get_port_name` (char *name)
- `int get_pin_number` (char *name)
- `short get_bit` (char in)
- `char * search_replace` (char *src, const char *sKey, const char *rKey, int flag)
- `bool validate_string_regex` (const char *str, const char *pattern)
- `char * find_substring` (char *src, const char *sKey, int flag)

2.44.1 Macro Definition Documentation

2.44.1.1 MAX_BUF

```
#define MAX_BUF 256
```

Definition at line 5 of file `odin_util.h`.

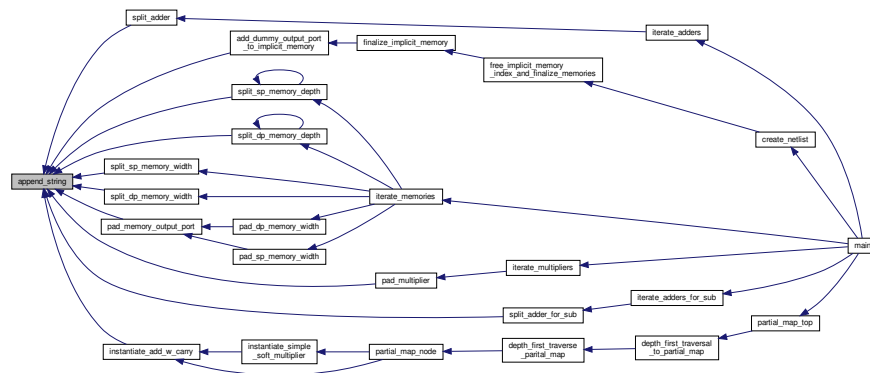
2.44.2 Function Documentation

2.44.2.1 `append_string()`

```
char* append_string (
    const char * string,
    const char * appendage,
    ... )
```

Definition at line 647 of file `odin_util.cpp`.

Here is the caller graph for this function:

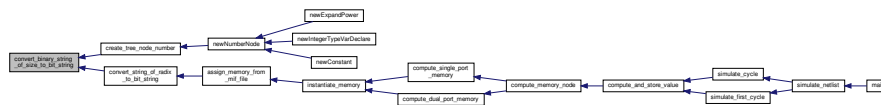


2.44.2.2 `convert_binary_string_of_size_to_bit_string()`

```
char* convert_binary_string_of_size_to_bit_string (
    short is_dont_care_number,
    char * orig_string,
    int binary_size )
```

Definition at line 366 of file `odin_util.cpp`.

Here is the caller graph for this function:

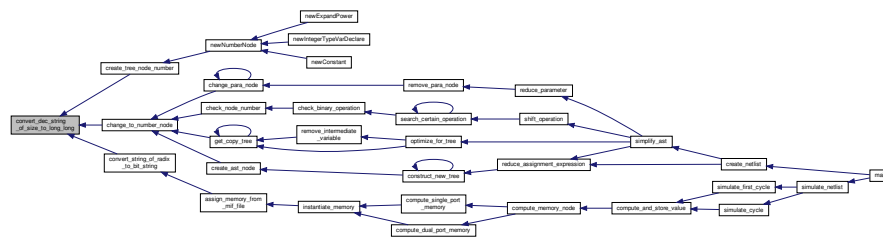


2.44.2.3 convert_dec_string_of_size_to_long_long()

```
long long convert_dec_string_of_size_to_long_long (
    char * orig_string,
    int size )
```

Definition at line 158 of file odin_util.cpp.

Here is the caller graph for this function:

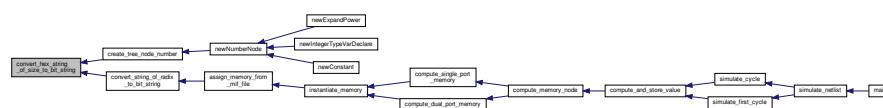


2.44.2.4 convert_hex_string_of_size_to_bit_string()

```
char* convert_hex_string_of_size_to_bit_string (
    short is_dont_care_number,
    char * orig_string,
    int size )
```

Definition at line 207 of file odin_util.cpp.

Here is the caller graph for this function:

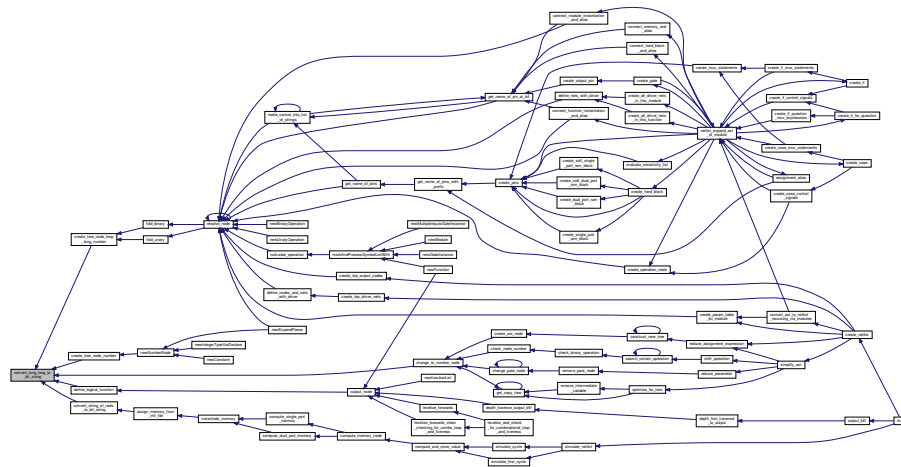


2.44.2.5 convert_long_long_to_bit_string()

```
char* convert_long_long_to_bit_string (
    long long orig_long,
    int num_bits )
```

Definition at line 137 of file `odin_util.cpp`.

Here is the caller graph for this function:

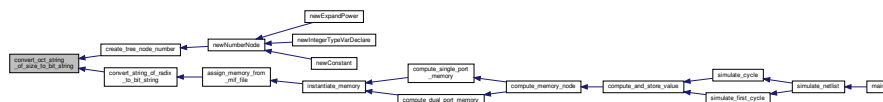


2.44.2.6 convert_oct_string_of_size_to_bit_string()

```
char* convert_oct_string_of_size_to_bit_string (
    char * orig_string,
    int size )
```

Definition at line 313 of file `odin_util.cpp`.

Here is the caller graph for this function:



2.44.2.7 convert_string_of_radix_to_bit_string()

```
char* convert_string_of_radix_to_bit_string (
    char * string,
    int radix,
    int binary_size )
```

Definition at line 108 of file odin_util.cpp.

Here is the caller graph for this function:



2.44.2.8 convert_string_of_radix_to_long_long()

```
long long convert_string_of_radix_to_long_long (
    char * orig_string,
    int radix )
```

Definition at line 171 of file odin_util.cpp.

Here is the caller graph for this function:

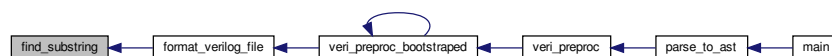


2.44.2.9 find_substring()

```
char* find_substring (
    char * src,
    const char * sKey,
    int flag )
```

Definition at line 769 of file odin_util.cpp.

Here is the caller graph for this function:

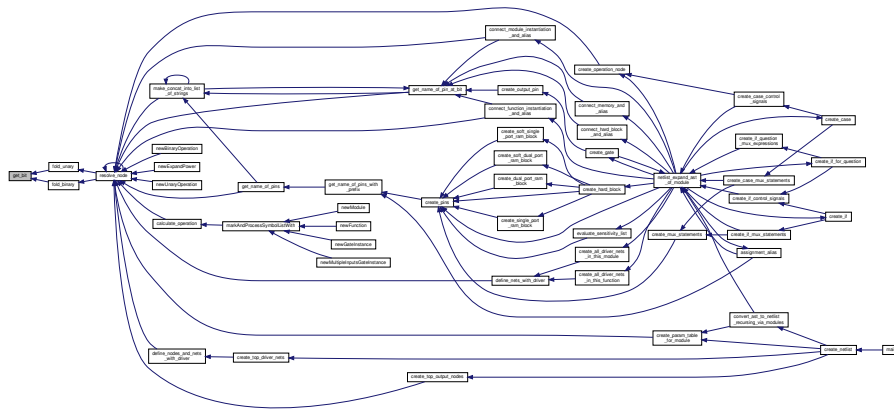


2.44.2.10 get_bit()

```
short get_bit (
    char in )
```

Definition at line 680 of file `odin_util.cpp`.

Here is the caller graph for this function:

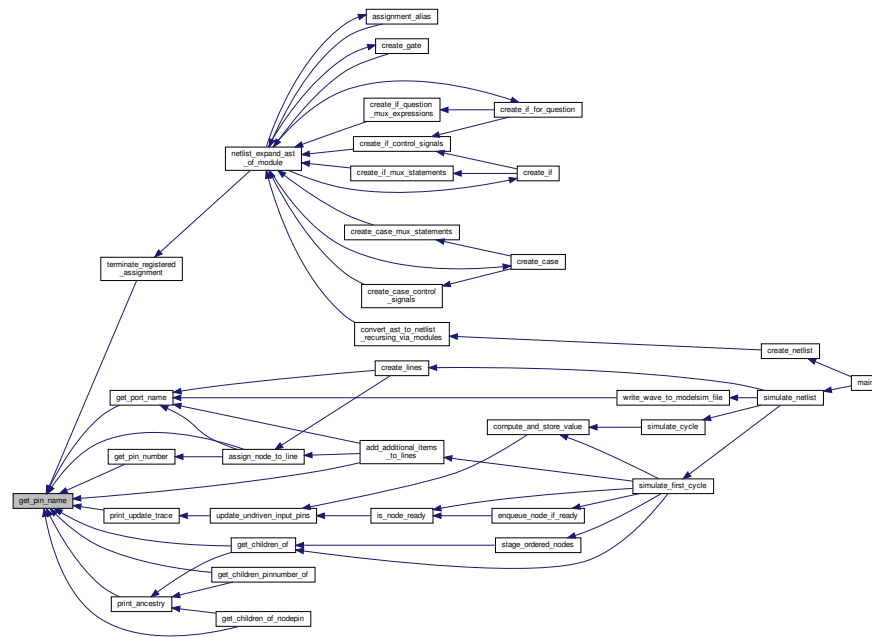


2.44.2.11 get_pin_name()

```
char* get_pin_name (
    char * name )
```

Definition at line 469 of file `odin_util.cpp`.

Here is the caller graph for this function:

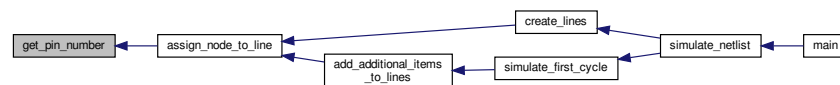


2.44.2.12 get_pin_number()

```
int get_pin_number (
    char * name )
```

Definition at line 499 of file odin_util.cpp.

Here is the caller graph for this function:

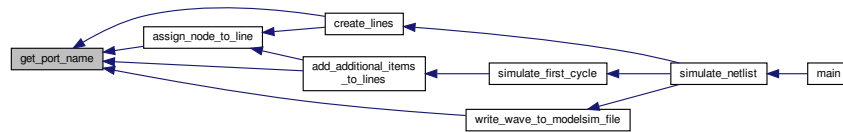


2.44.2.13 get_port_name()

```
char* get_port_name (
    char * name )
```

Definition at line 482 of file odin_util.cpp.

Here is the caller graph for this function:

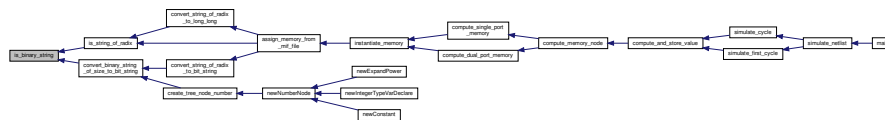


2.44.2.14 is_binary_string()

```
int is_binary_string (
    char * string )
```

Definition at line 453 of file odin_util.cpp.

Here is the caller graph for this function:

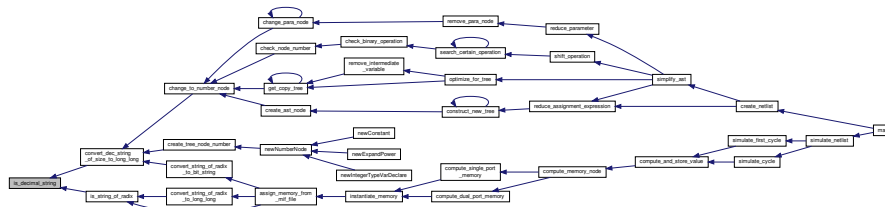


2.44.2.15 is_decimal_string()

```
int is_decimal_string (
    char * string )
```

Definition at line 427 of file odin_util.cpp.

Here is the caller graph for this function:

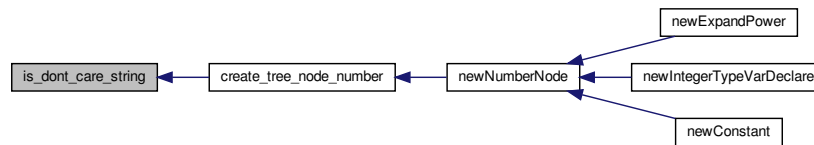


2.44.2.16 is_dont_care_string()

```
int is_dont_care_string (
    char * string )
```

Definition at line 414 of file odin_util.cpp.

Here is the caller graph for this function:

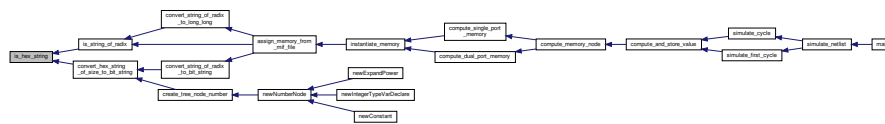


2.44.2.17 is_hex_string()

```
int is_hex_string (
    char * string )
```

Definition at line 401 of file odin_util.cpp.

Here is the caller graph for this function:



2.44.2.18 is_octal_string()

```
int is_octal_string (
    char * string )
```

Definition at line 440 of file odin_util.cpp.

Here is the caller graph for this function:

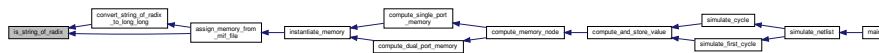


2.44.2.19 is_string_of_radix()

```
int is_string_of_radix (
    char * string,
    int radix )
```

Definition at line 189 of file `odin_util.cpp`.

Here is the caller graph for this function:

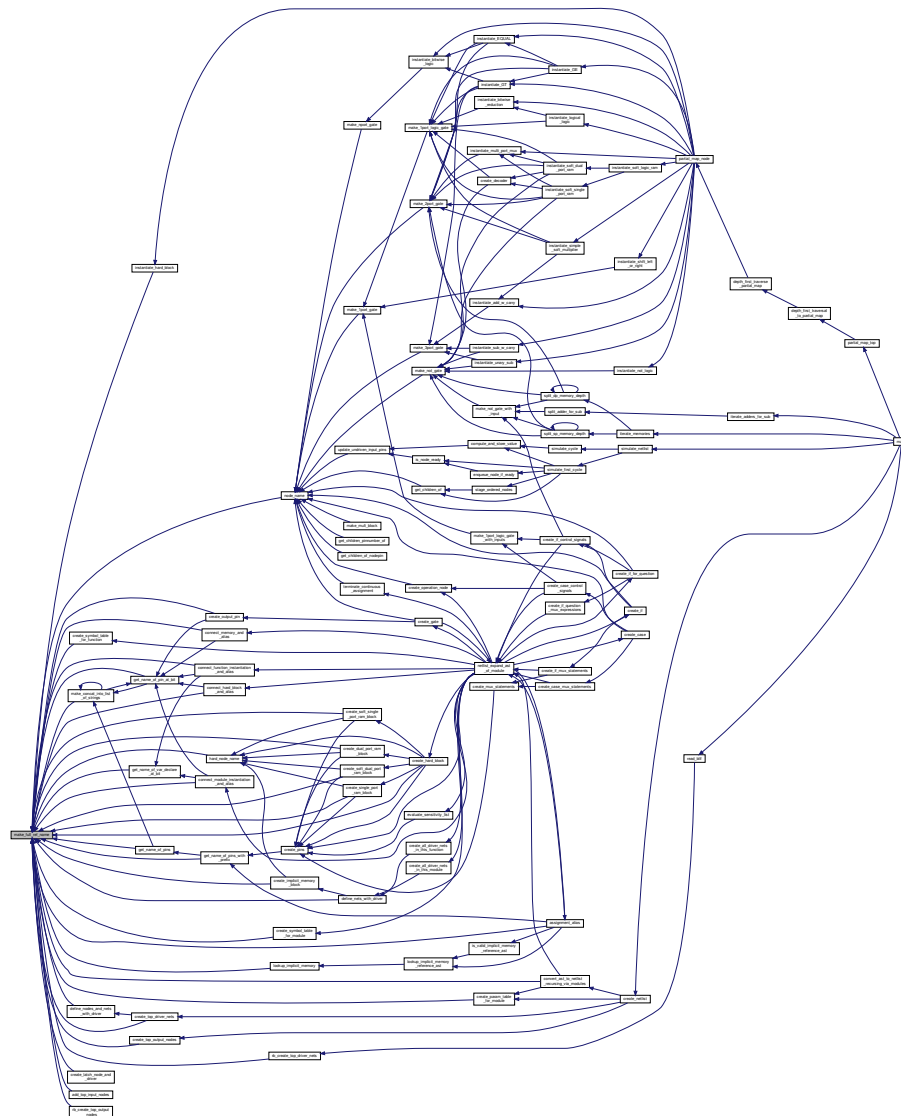


2.44.2.20 make_full_ref_name()

```
char* make_full_ref_name (
    const char * previous,
    char * module_name,
    char * module_instance_name,
    const char * signal_name,
    long bit )
```

Definition at line 61 of file `odin_util.cpp`.

Here is the caller graph for this function:

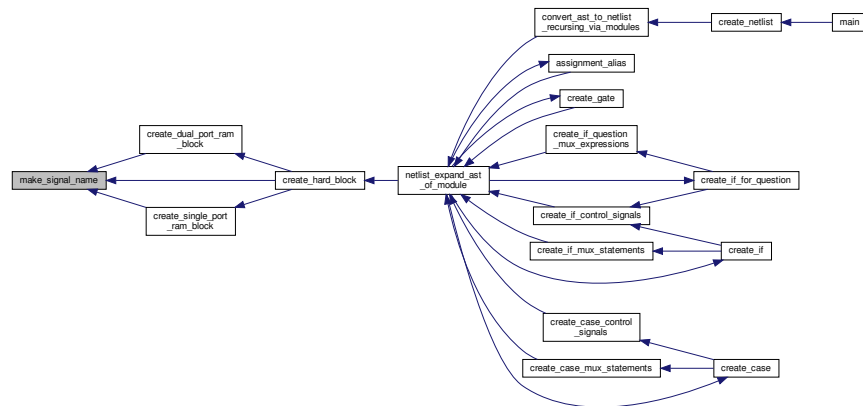


2.44.2.21 make_signal_name()

```
char* make_signal_name (
    char * signal_name,
    int bit )
```

Definition at line 46 of file odin_util.cpp.

Here is the caller graph for this function:



2.44.2.22 make_simple_name()

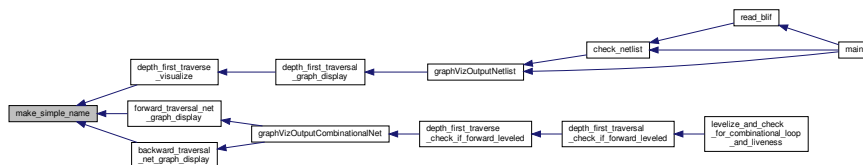
```

char* make_simple_name (
    char * input,
    const char * flatten_string,
    char flatten_char )

```

Definition at line 544 of file odin_util.cpp.

Here is the caller graph for this function:



2.44.2.23 make_string_based_on_id()

```

char* make_string_based_on_id (
    nnode_t * node )

```

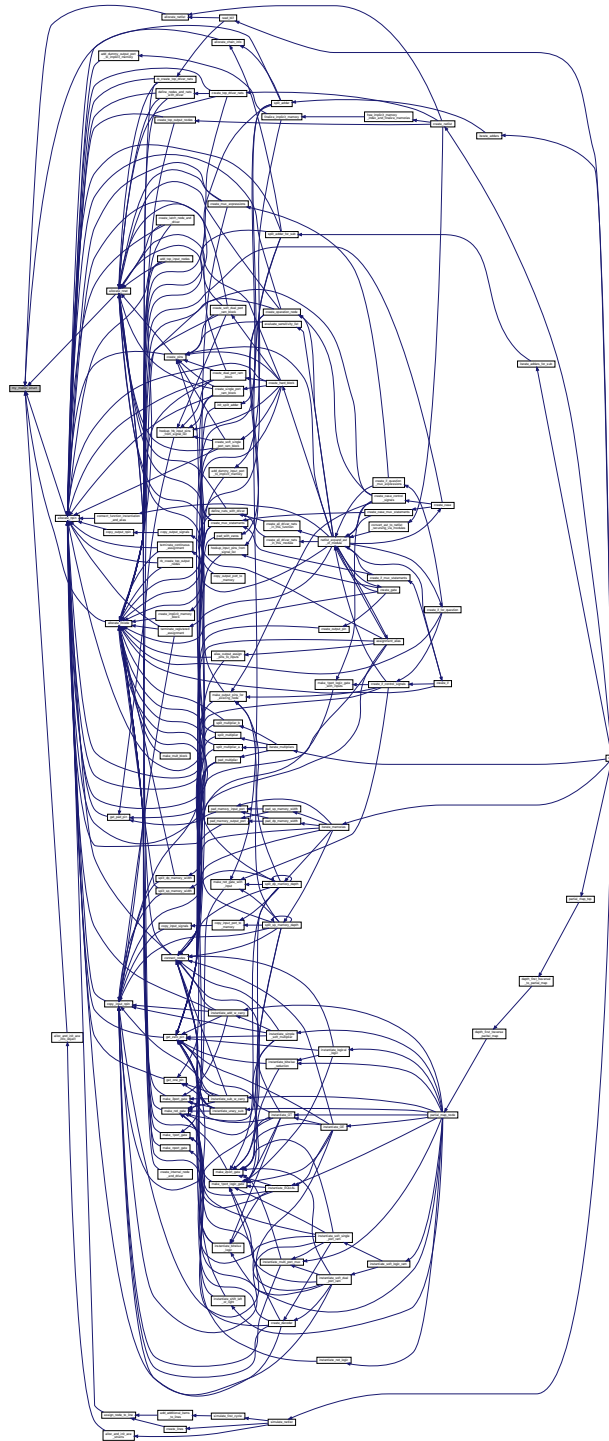
Definition at line 533 of file odin_util.cpp.

2.44.2.24 my_malloc_struct()

```
void* my_malloc_struct (
    size_t bytes_to_alloc )
```

Definition at line 574 of file `odin_util.cpp`.

Here is the caller graph for this function:

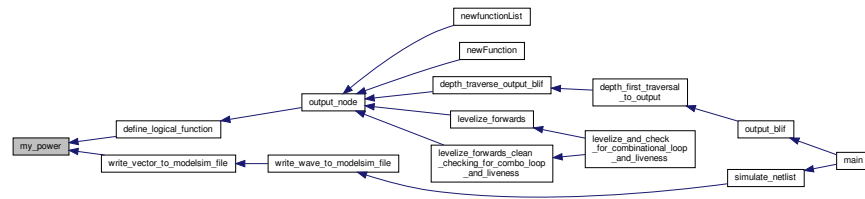


2.44.2.25 my_power()

```
long long int my_power (
    long long int x,
    long long int y )
```

Definition at line 517 of file odin_util.cpp.

Here is the caller graph for this function:



2.44.2.26 pow2()

```
long long int pow2 (
    int to_the_power )
```

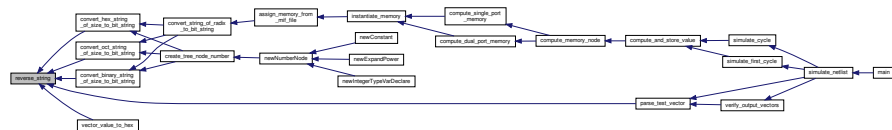
Definition at line 597 of file odin_util.cpp.

2.44.2.27 reverse_string()

```
void reverse_string (
    char * token,
    int length )
```

Definition at line 665 of file odin_util.cpp.

Here is the caller graph for this function:

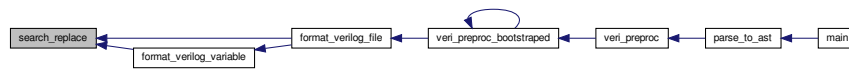


2.44.2.28 search_replace()

```
char* search_replace (
    char * src,
    const char * sKey,
    const char * rKey,
    int flag )
```

Definition at line 748 of file odin_util.cpp.

Here is the caller graph for this function:



2.44.2.29 string_to_lower()

```
void string_to_lower (
    char * string )
```

Definition at line 628 of file odin_util.cpp.

2.44.2.30 string_to_upper()

```
void string_to_upper (
    char * string )
```

Definition at line 613 of file odin_util.cpp.

Here is the caller graph for this function:

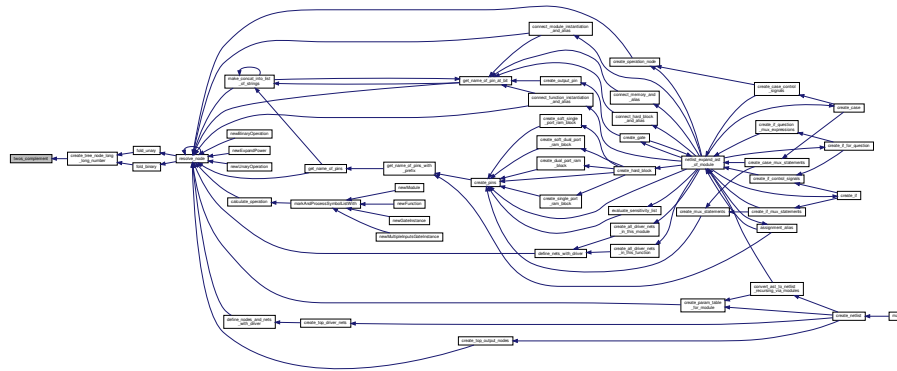


2.44.2.31 twos_complement()

```
char* twos_complement (
    char * str )
```

Definition at line 80 of file `odin_util.cpp`.

Here is the caller graph for this function:



2.44.2.32 validate_string_regex()

```
bool validate_string_regex (
    const char * str,
    const char * pattern )
```

Definition at line 797 of file `odin_util.cpp`.

Here is the caller graph for this function:



2.45 vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL/read_xml_config_file.cpp File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <stdarg.h>
#include "types.h"
#include "globals.h"
#include "read_xml_config_file.h"
#include "read_xml_util.h"
#include "pugixml.hpp"
#include "pugixml_util.hpp"
#include "vtr_util.h"
#include "vtr_memory.h"
```


Functions

- void [read_verilog_files](#) (pugi::xml_node a_node, config_t *config, const pugiutil::loc_data &loc_data)
- void [read_outputs](#) (pugi::xml_node a_node, config_t *config, const pugiutil::loc_data &loc_data)
- void [read_debug_switches](#) (pugi::xml_node a_node, config_t *config, const pugiutil::loc_data &loc_data)
- void [read_optimizations](#) (pugi::xml_node a_node, config_t *config, const pugiutil::loc_data &loc_data)
- void [set_default_optimization_settings](#) (config_t *config)
- void [read_config_file](#) (char *file_name)

Variables

- config_t [configuration](#)
- char [empty_string](#) [] = ""

2.45.1 Function Documentation

2.45.1.1 read_config_file()

```
void read_config_file (  
    char * file_name )
```

Definition at line 53 of file read_xml_config_file.cpp.

Here is the caller graph for this function:

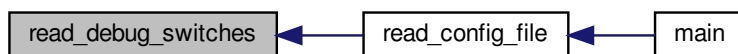


2.45.1.2 read_debug_switches()

```
void read_debug_switches (
    pugi::xml_node a_node,
    config_t * config,
    const pugiutil::loc_data & loc_data )
```

Definition at line 159 of file read_xml_config_file.cpp.

Here is the caller graph for this function:

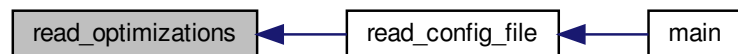


2.45.1.3 read_optimizations()

```
void read_optimizations (
    pugi::xml_node a_node,
    config_t * config,
    const pugiutil::loc_data & loc_data )
```

Definition at line 217 of file read_xml_config_file.cpp.

Here is the caller graph for this function:

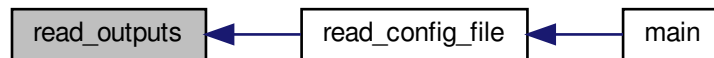


2.45.1.4 read_outputs()

```
void read_outputs (
    pugi::xml_node a_node,
    config_t * config,
    const pugiutil::loc_data & loc_data )
```

Definition at line 122 of file read_xml_config_file.cpp.

Here is the caller graph for this function:

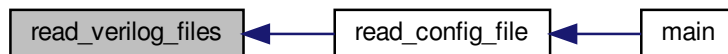


2.45.1.5 read_verilog_files()

```
void read_verilog_files (
    pugi::xml_node a_node,
    config_t * config,
    const pugiutil::loc_data & loc_data )
```

Definition at line 100 of file read_xml_config_file.cpp.

Here is the caller graph for this function:

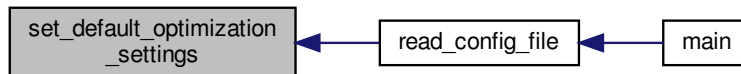


2.45.1.6 set_default_optimization_settings()

```
void set_default_optimization_settings (  
    config_t * config )
```

Definition at line 199 of file read_xml_config_file.cpp.

Here is the caller graph for this function:



2.45.2 Variable Documentation

2.45.2.1 configuration

```
config_t configuration
```

Definition at line 38 of file read_xml_config_file.cpp.

2.45.2.2 empty_string

```
char empty_string[] = ""
```

Definition at line 39 of file read_xml_config_file.cpp.

2.46 vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL/read_xml_config_file.h File Reference

```
#include "types.h"
```

Functions

- void [read_config_file](#) (char *file_name)

2.46.1 Function Documentation

2.46.1.1 read_config_file()

```
void read_config_file (  
    char * file_name )
```

Definition at line 53 of file read_xml_config_file.cpp.

Here is the caller graph for this function:



2.47 vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL/verilog_bison_user_defined.h File Reference

Functions

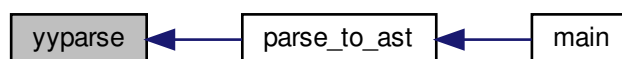
- int [yyvsparse](#) (void)

2.47.1 Function Documentation

2.47.1.1 yyparse()

```
int yyparse (  
    void )
```

Here is the caller graph for this function:



2.48 vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL/verilog_preprocessor.cpp File Reference

```
#include <ctype.h>
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "verilog_preprocessor.h"
#include "types.h"
#include "vtr_util.h"
#include "vtr_memory.h"
#include "odin_util.h"
#include <stdbool.h>
```

Functions

- FILE * [open_source_file](#) (char *filename)
- FILE * [remove_comments](#) (FILE *source)
- FILE * [format_verilog_file](#) (FILE *source)
- FILE * [format_verilog_variable](#) (FILE *src, FILE *dest)
- int [init_veri_preproc](#) ()
- int [cleanup_veri_preproc](#) ()
- void [clean_veri_define](#) ([veri_define](#) *current)
- void [clean_veri_include](#) ([veri_include](#) *current)
- int [add_veri_define](#) (char *symbol, char *value, int line, [veri_include](#) *defined_in)
- [veri_include](#) * [add_veri_include](#) (char *path, int line, [veri_include](#) *included_from)
- char * [ret_veri_definedval](#) (char *symbol)
- int [veri_is_defined](#) (char *symbol)
- FILE * [veri_preproc](#) (FILE *source)
- void [veri_preproc_bootstrapped](#) (FILE *original_source, FILE *preproc_producer, [veri_include](#) *current_include)
- char * [trim](#) (char *string)
- int [top](#) ([veri_flag_stack](#) *stack)
- int [pop](#) ([veri_flag_stack](#) *stack)
- void [push](#) ([veri_flag_stack](#) *stack, int flag)

Variables

- struct [veri_Includes](#) [veri_includes](#)
- struct [veri_Defines](#) [veri_defines](#)

2.48.1 Function Documentation

2.48.1.1 add_veri_define()

```
int add_veri_define (
    char * symbol,
    char * value,
    int line,
    veri_include * defined_in )
```

Definition at line 120 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

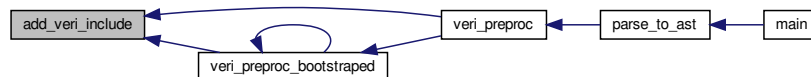


2.48.1.2 add_veri_include()

```
veri_include* add_veri_include (
    char * path,
    int line,
    veri_include * included_from )
```

Definition at line 189 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

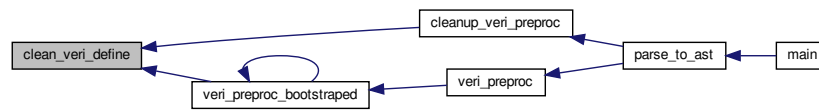


2.48.1.3 clean_veri_define()

```
void clean_veri_define (
    veri_define * current )
```

Definition at line 83 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

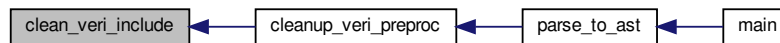


2.48.1.4 clean_veri_include()

```
void clean_veri_include (
    veri_include * current )
```

Definition at line 103 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:



2.48.1.5 cleanup_veri_preproc()

```
int cleanup_veri_preproc ( )
```

Definition at line 49 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

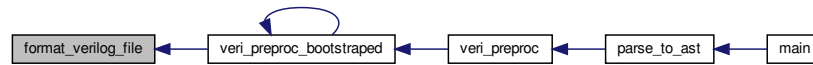


2.48.1.6 format_verilog_file()

```
FILE * format_verilog_file (
    FILE * source )
```

Definition at line 698 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:



2.48.1.7 format_verilog_variable()

```
FILE * format_verilog_variable (
    FILE * src,
    FILE * dest )
```

Definition at line 766 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

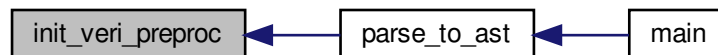


2.48.1.8 init_veri_preproc()

```
int init_veri_preproc ( )
```

Definition at line 24 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

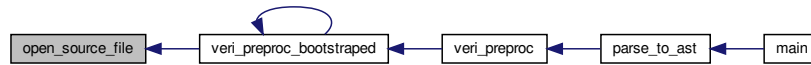


2.48.1.9 open_source_file()

```
FILE * open_source_file (  
    char * filename )
```

Definition at line 269 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

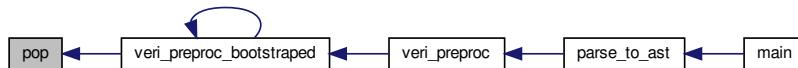


2.48.1.10 pop()

```
int pop (  
    veri_flag_stack * stack )
```

Definition at line 673 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:



2.48.1.11 push()

```
void push (  
    veri_flag_stack * stack,  
    int flag )
```

Definition at line 687 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:



2.48.1.12 remove_comments()

```
FILE * remove_comments (
    FILE * source )
```

Definition at line 358 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:



2.48.1.13 ret_veri_definedval()

```
char* ret_veri_definedval (
    char * symbol )
```

Definition at line 233 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

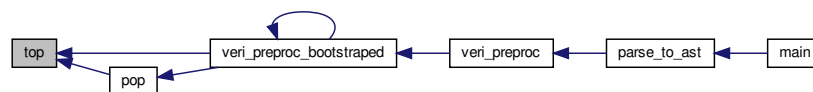


2.48.1.14 top()

```
int top (
    veri_flag_stack * stack )
```

Definition at line 663 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

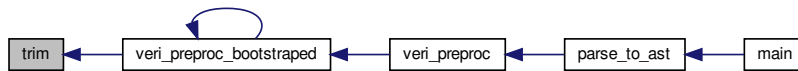


2.48.1.15 trim()

```
char* trim (  
    char * string )
```

Definition at line 636 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

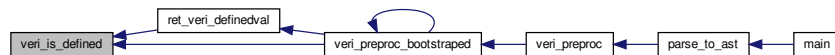


2.48.1.16 veri_is_defined()

```
int veri_is_defined (  
    char * symbol )
```

Definition at line 247 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

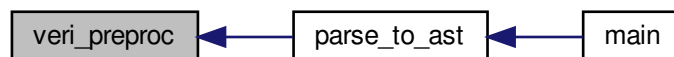


2.48.1.17 veri_preproc()

```
FILE* veri_preproc (  
    FILE * source )
```

Definition at line 320 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

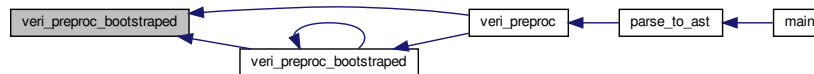


2.48.1.18 veri_preproc_bootstrapped()

```
void veri_preproc_bootstrapped (
    FILE * original_source,
    FILE * preproc_producer,
    veri_include * current_include )
```

Definition at line 406 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:



2.48.2 Variable Documentation

2.48.2.1 veri_defines

```
struct veri_Defines veri_defines
```

Definition at line 14 of file verilog_preprocessor.cpp.

2.48.2.2 veri_includes

```
struct veri_Includes veri_includes
```

Definition at line 13 of file verilog_preprocessor.cpp.

2.49 vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL/verilog_preprocessor.h File Reference

Data Structures

- struct [veri_include](#)
- struct [veri_define](#)
- struct [veri_Includes](#)
- struct [veri_Defines](#)
- struct [veri_flag_node](#)
- struct [veri_flag_stack](#)

Macros

- `#define DefaultSize 20`
- `#define MaxLine 4096`

Typedefs

- `typedef struct veri_include veri_include`
- `typedef struct veri_define veri_define`
- `typedef struct veri_flag_node veri_flag_node`

Functions

- `int init_veri_preproc ()`
- `int cleanup_veri_preproc ()`
- `void clean_veri_define (veri_define *current)`
- `void clean_veri_include (veri_include *current)`
- `int add_veri_define (char *symbol, char *value, int line, veri_include *included_from)`
- `char * ret_veri_definedval (char *symbol)`
- `int veri_is_defined (char *symbol)`
- `veri_include * add_veri_include (char *path, int line, veri_include *included_from)`
- `FILE * veri_preproc (FILE *source)`
- `void veri_preproc_bootstrapped (FILE *source, FILE *preproc_producer, veri_include *current_include)`
- `int top (veri_flag_stack *stack)`
- `int pop (veri_flag_stack *stack)`
- `void push (veri_flag_stack *stack, int flag)`
- `char * trim (char *string)`

Variables

- `struct veri_Includes veri_includes`
- `struct veri_Defines veri_defines`

2.49.1 Macro Definition Documentation

2.49.1.1 DefaultSize

```
#define DefaultSize 20
```

Definition at line 4 of file verilog_preprocessor.h.

2.49.1.2 MaxLine

```
#define MaxLine 4096
```

Definition at line 5 of file verilog_preprocessor.h.

2.49.2 Typedef Documentation

2.49.2.1 veri_define

```
typedef struct veri_define veri_define
```

2.49.2.2 veri_flag_node

```
typedef struct veri_flag_node veri_flag_node
```

2.49.2.3 veri_include

```
typedef struct veri_include veri_include
```

2.49.3 Function Documentation

2.49.3.1 add_veri_define()

```
int add_veri_define (  
    char * symbol,  
    char * value,  
    int line,  
    veri_include * included_from )
```

Definition at line 120 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

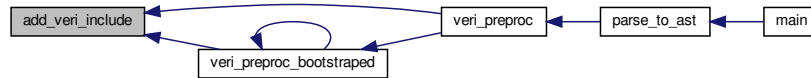


2.49.3.2 add_veri_include()

```
veri_include* add_veri_include (  
    char * path,  
    int line,  
    veri_include * included_from )
```

Definition at line 189 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

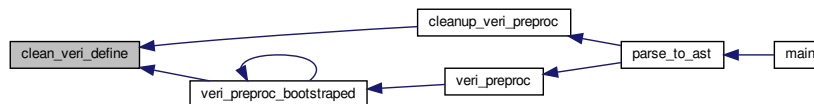


2.49.3.3 clean_veri_define()

```
void clean_veri_define (  
    veri_define * current )
```

Definition at line 83 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

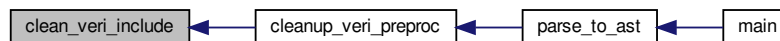


2.49.3.4 clean_veri_include()

```
void clean_veri_include (  
    veri_include * current )
```

Definition at line 103 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

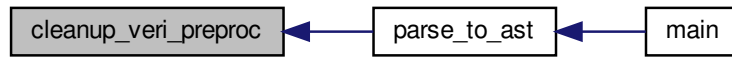


2.49.3.5 cleanup_veri_preproc()

```
int cleanup_veri_preproc ( )
```

Definition at line 49 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

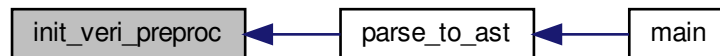


2.49.3.6 init_veri_preproc()

```
int init_veri_preproc ( )
```

Definition at line 24 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

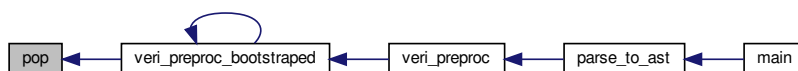


2.49.3.7 pop()

```
int pop (
    veri_flag_stack * stack )
```

Definition at line 673 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:



2.49.3.8 push()

```
void push (
    veri_flag_stack * stack,
    int flag )
```

Definition at line 687 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:



2.49.3.9 ret_veri_definedval()

```
char* ret_veri_definedval (
    char * symbol )
```

Definition at line 233 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

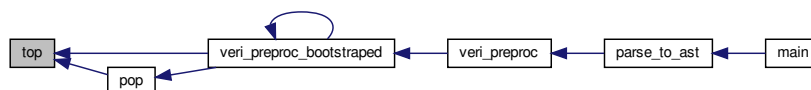


2.49.3.10 top()

```
int top (
    veri_flag_stack * stack )
```

Definition at line 663 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

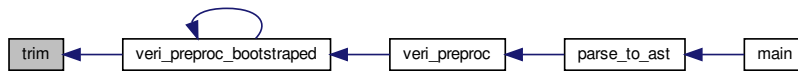


2.49.3.11 trim()

```
char* trim (
    char * string )
```

Definition at line 636 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

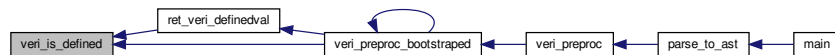


2.49.3.12 veri_is_defined()

```
int veri_is_defined (
    char * symbol )
```

Definition at line 247 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

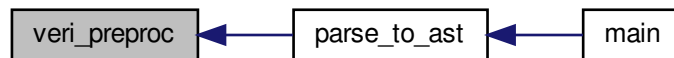


2.49.3.13 veri_preproc()

```
FILE* veri_preproc (
    FILE * source )
```

Definition at line 320 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:

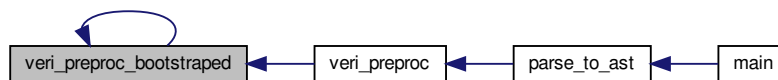


2.49.3.14 veri_preproc_bootstrapped()

```
void veri_preproc_bootstrapped (
    FILE * source,
    FILE * preproc_producer,
    veri_include * current_include )
```

Definition at line 406 of file verilog_preprocessor.cpp.

Here is the caller graph for this function:



2.49.4 Variable Documentation

2.49.4.1 veri_defines

```
struct veri_Defines veri_defines
```

Definition at line 14 of file verilog_preprocessor.cpp.

2.49.4.2 veri_includes

```
struct veri_Includes veri_includes
```

Definition at line 13 of file verilog_preprocessor.cpp.

2.50 vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/ace.cpp File Reference

```
#include <stdlib.h>
#include <stdio.h>
#include "cudd.h"
#include "ace.h"
#include "st.h"
#include "vtr_memory.h"
```

Data Structures

- struct [ace_cube_t](#)

Macros

- #define [ACE_P0TO1](#)(P1, PS) ((P1)==0.0)?0.0:(((P1)==1.0)?1.0:0.5*PS/(1.0-(P1)))
- #define [ACE_P1TO0](#)(P1, PS) ((P1)==0.0)?1.0:(((P1)==0.0)?0.0:0.5*PS/(P1))
- #define [MAX](#)(a, b) (a > b ? a : b)
- #define [TWO](#) 3
- #define [DASH](#) 3
- #define [ONE](#) 2
- #define [ZERO](#) 1
- #define [BPI](#) 32
- #define [LOGBPI](#) 5
- #define [fail](#)(why)
- #define [LOOPINIT](#)(size) ((size <= [BPI](#)) ? 1 : [WHICH_WORD](#)((size)-1))
- #define [WHICH_WORD](#)(element) (((element) >> [LOGBPI](#)) + 1)
- #define [WHICH_BIT](#)(element) ((element) & ([BPI](#)-1))
- #define [GETINPUT](#)(c, pos) ((c[[WHICH_WORD](#)(2*pos)] >> [WHICH_BIT](#)(2*pos)) & 3)
- #define [node_get_literal](#)(c_, j_) ((c_) ? [GETINPUT](#)((c_), (j_)) : [node_error](#)(2))
- #define [ALLOC](#)(type, num) ((type *) malloc(sizeof(type) * (num)))
- #define [set_remove](#)(set, e) (set[[WHICH_WORD](#)(e)] &= ~ (1 << [WHICH_BIT](#)(e)))
- #define [set_insert](#)(set, e) (set[[WHICH_WORD](#)(e)] |= 1 << [WHICH_BIT](#)(e))
- #define [set_new](#)(size) [set_clear](#)([ALLOC](#)(unsigned int, [set_size](#)(size)), size)
- #define [set_size](#)(size) ((size) <= [BPI](#) ? 2 : ([WHICH_WORD](#)((size)-1) + 1))

Typedefs

- typedef unsigned int * [pset](#)

Functions

- [ace_obj_info_t](#) * [alloc_and_init_ace_info_object](#) ()
- int [calc_node_depth](#) (nnode_t *node)
- void [compute_switching_activities](#) (netlist_t *net)
- void [prob_epsilon_fix](#) (double *d)
- void [output_ace_info](#) (netlist_t *net, FILE *act_out)
- void [output_ace_info_node](#) (char *name, [ace_obj_info_t](#) *info, FILE *act_out)
- [pset](#) [set_clear](#) ([pset](#) r, int size)
- [ace_cube_t](#) * [ace_cube_dup](#) ([ace_cube_t](#) *cube)
- [ace_cube_t](#) * [ace_cube_new_dc](#) (int num_literals)
- DdNode * [build_bdd_for_node](#) (DdManager *dd, nnode_t *node)
- void [ace_bdd_count_paths](#) (DdManager *mgr, DdNode *bdd, int *num_one_paths, int *num_zero_paths)
- double [calc_cube_switch_prob_recur](#) (DdManager *mgr, DdNode *bdd, [ace_cube_t](#) *cube, nnode_t *node, st←_table *visited, int phase)
- double [calc_cube_switch_prob](#) (DdManager *mgr, DdNode *bdd, [ace_cube_t](#) *cube, nnode_t *node, int phase)
- double [calc_switch_prob_recur](#) (DdManager *mgr, DdNode *bdd_next, DdNode *bdd, [ace_cube_t](#) *cube, nnode_t *node, double P1, int phase)
- int [node_error](#) (int code)
- void [alloc_and_init_ace_structs](#) (netlist_t *net)
- void [calculate_activity](#) (netlist_t *net, int max_cycles, FILE *act_out)

2.50.1 Macro Definition Documentation

2.50.1.1 ACE_P0T01

```
#define ACE_P0T01(  
    P1,  
    PS ) ((P1)==0.0)?0.0:(((P1)==1.0)?1.0:0.5*PS/(1.0-(P1)))
```

Definition at line 30 of file ace.cpp.

2.50.1.2 ACE_P1T00

```
#define ACE_P1T00(  
    P1,  
    PS ) ((P1)==0.0)?1.0:(((P1)==0.0)?0.0:0.5*PS/(P1))
```

Definition at line 31 of file ace.cpp.

2.50.1.3 ALLOC

```
#define ALLOC(  
    type,  
    num ) ((type *) malloc(sizeof(type) * (num)))
```

Definition at line 62 of file ace.cpp.

2.50.1.4 BPI

```
#define BPI 32
```

Definition at line 41 of file ace.cpp.

2.50.1.5 DASH

```
#define DASH 3
```

Definition at line 37 of file ace.cpp.

2.50.1.6 fail

```
#define fail(  
    why )
```

Value:

```
{\  
    (void) fprintf(stderr, "Fatal error: file %s, line %d\n%s\n",\  
        __FILE__, __LINE__, why);\br/>    (void) fflush(stdout);\br/>    abort();\  
}
```

Definition at line 44 of file ace.cpp.

2.50.1.7 GETINPUT

```
#define GETINPUT(  
    c,  
    pos ) ((c[WHICH_WORD(2*pos)] >> WHICH_BIT(2*pos)) & 3)
```

Definition at line 56 of file ace.cpp.

2.50.1.8 LOGBPI

```
#define LOGBPI 5
```

Definition at line 42 of file ace.cpp.

2.50.1.9 LOOPINIT

```
#define LOOPINIT(  
    size ) ((size <= BPI) ? 1 : WHICH_WORD((size)-1))
```

Definition at line 51 of file ace.cpp.

2.50.1.10 MAX

```
#define MAX(  
    a,  
    b ) (a > b ? a : b)
```

Definition at line 33 of file ace.cpp.

2.50.1.11 node_get_literal

```
#define node_get_literal(  
    c_,  
    j_ ) ((c_) ?  GETINPUT((c_), (j_)) :  node_error(2))
```

Definition at line 58 of file ace.cpp.

2.50.1.12 ONE

```
#define ONE 2
```

Definition at line 38 of file ace.cpp.

2.50.1.13 set_insert

```
#define set_insert(  
    set,  
    e ) (set[WHICH_WORD(e)] |= 1 << WHICH_BIT(e))
```

Definition at line 65 of file ace.cpp.

2.50.1.14 set_new

```
#define set_new(  
    size ) set_clear(ALLOC(unsigned int, set_size(size)), size)
```

Definition at line 66 of file ace.cpp.

2.50.1.15 set_remove

```
#define set_remove(  
    set,  
    e ) (set[WHICH_WORD(e)] &= ~ (1 << WHICH_BIT(e)))
```

Definition at line 64 of file ace.cpp.

2.50.1.16 set_size

```
#define set_size(  
    size ) ((size) <= BPI ? 2 : (WHICH_WORD((size)-1) + 1))
```

Definition at line 67 of file ace.cpp.

2.50.1.17 TWO

```
#define TWO 3
```

Definition at line 36 of file ace.cpp.

2.50.1.18 WHICH_BIT

```
#define WHICH_BIT(  
    element ) ((element) & (BPI-1))
```

Definition at line 54 of file ace.cpp.

2.50.1.19 WHICH_WORD

```
#define WHICH_WORD(  
    element ) (((element) >> LOGBPI) + 1)
```

Definition at line 53 of file ace.cpp.

2.50.1.20 ZERO

```
#define ZERO 1
```

Definition at line 39 of file ace.cpp.

2.50.2 Typedef Documentation

2.50.2.1 pset

```
typedef unsigned int* pset
```

Definition at line 60 of file ace.cpp.

2.50.3 Function Documentation

2.50.3.1 ace_bdd_count_paths()

```
void ace_bdd_count_paths (
    DdManager * mgr,
    DdNode * bdd,
    int * num_one_paths,
    int * num_zero_paths )
```

Definition at line 560 of file ace.cpp.

Here is the caller graph for this function:

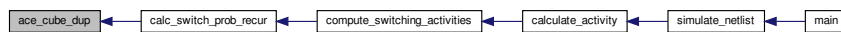


2.50.3.2 ace_cube_dup()

```
ace_cube_t * ace_cube_dup (
    ace_cube_t * cube )
```

Definition at line 464 of file ace.cpp.

Here is the caller graph for this function:



2.50.3.3 ace_cube_new_dc()

```
ace_cube_t * ace_cube_new_dc (
    int num_literals )
```

Definition at line 498 of file ace.cpp.

Here is the caller graph for this function:

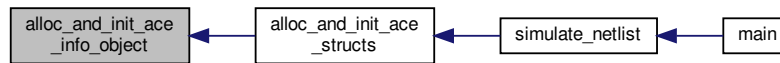


2.50.3.4 alloc_and_init_ace_info_object()

```
ace_obj_info_t * alloc_and_init_ace_info_object ( )
```

Definition at line 163 of file ace.cpp.

Here is the caller graph for this function:

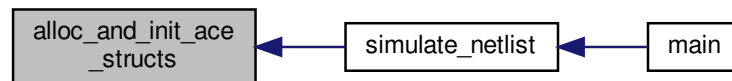


2.50.3.5 alloc_and_init_ace_structs()

```
void alloc_and_init_ace_structs (
    netlist_t * net )
```

Definition at line 101 of file ace.cpp.

Here is the caller graph for this function:



2.50.3.6 build_bdd_for_node()

```
DdNode * build_bdd_for_node (
    DdManager * dd,
    nnode_t * node )
```

Definition at line 524 of file ace.cpp.

Here is the caller graph for this function:



2.50.3.7 calc_cube_switch_prob()

```
double calc_cube_switch_prob (  
    DdManager * mgr,  
    DdNode * bdd,  
    ace_cube_t * cube,  
    nnode_t * node,  
    int phase )
```

Definition at line 650 of file ace.cpp.

Here is the caller graph for this function:



2.50.3.8 calc_cube_switch_prob_recur()

```
double calc_cube_switch_prob_recur (  
    DdManager * mgr,  
    DdNode * bdd,  
    ace_cube_t * cube,  
    nnode_t * node,  
    st_table * visited,  
    int phase )
```

Definition at line 584 of file ace.cpp.

Here is the caller graph for this function:

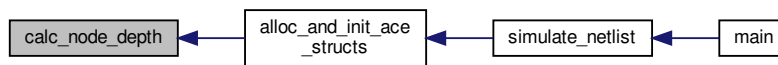


2.50.3.9 calc_node_depth()

```
int calc_node_depth (
    nnode_t * node )
```

Definition at line 184 of file ace.cpp.

Here is the caller graph for this function:



2.50.3.10 calc_switch_prob_recur()

```
double calc_switch_prob_recur (
    DdManager * mgr,
    DdNode * bdd_next,
    DdNode * bdd,
    ace_cube_t * cube,
    nnode_t * node,
    double P1,
    int phase )
```

Definition at line 668 of file ace.cpp.

Here is the caller graph for this function:

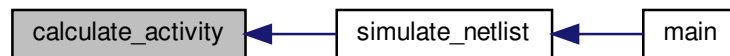


2.50.3.11 calculate_activity()

```
void calculate_activity (
    netlist_t * net,
    int max_cycles,
    FILE * act_out )
```

Definition at line 203 of file ace.cpp.

Here is the caller graph for this function:

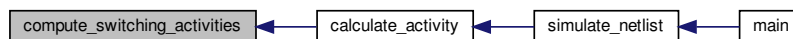


2.50.3.12 compute_switching_activities()

```
void compute_switching_activities (
    netlist_t * net )
```

Definition at line 282 of file ace.cpp.

Here is the caller graph for this function:



2.50.3.13 node_error()

```
int node_error (
    int code )
```

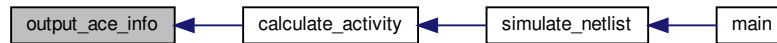
Definition at line 436 of file ace.cpp.

2.50.3.14 output_ace_info()

```
void output_ace_info (
    netlist_t * net,
    FILE * act_out )
```

Definition at line 360 of file ace.cpp.

Here is the caller graph for this function:

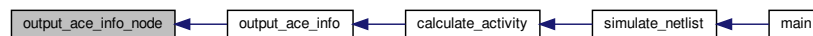


2.50.3.15 output_ace_info_node()

```
void output_ace_info_node (
    char * name,
    ace_obj_info_t * info,
    FILE * act_out )
```

Definition at line 394 of file ace.cpp.

Here is the caller graph for this function:



2.50.3.16 prob_epsilon_fix()

```
void prob_epsilon_fix (
    double * d )
```

Definition at line 344 of file ace.cpp.

Here is the caller graph for this function:



2.50.3.17 set_clear()

```
pset set_clear (
    pset r,
    int size )
```

Definition at line 424 of file ace.cpp.

2.51 vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/ace.h File Reference

```
#include "types.h"
```

Functions

- void [alloc_and_init_ace_structs](#) (netlist_t *net)
- void [calculate_activity](#) (netlist_t *net, int max_cycles, FILE *act_out)

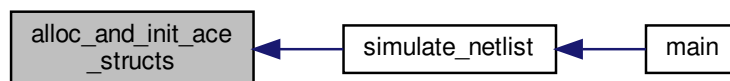
2.51.1 Function Documentation

2.51.1.1 alloc_and_init_ace_structs()

```
void alloc_and_init_ace_structs (
    netlist_t * net )
```

Definition at line 101 of file ace.cpp.

Here is the caller graph for this function:

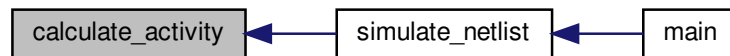


2.51.1.2 calculate_activity()

```
void calculate_activity (
    netlist_t * net,
    int max_cycles,
    FILE * act_out )
```

Definition at line 203 of file ace.cpp.

Here is the caller graph for this function:



2.52 vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/queue.cpp File Reference

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "queue.h"
#include "types.h"
#include "vtr_memory.h"
```

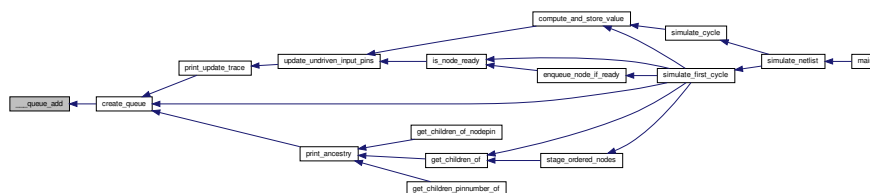
Functions

- void [__queue_add](#) ([queue_t](#) *q, void *item)
- void * [__queue_remove](#) ([queue_t](#) *q)
- void ** [__queue_remove_all](#) ([queue_t](#) *q)
- int [__queue_is_empty](#) ([queue_t](#) *q)
- void [__queue_destroy](#) ([queue_t](#) *q)
- [queue_t](#) * [create_queue](#) ()

2.52.1 Function Documentation

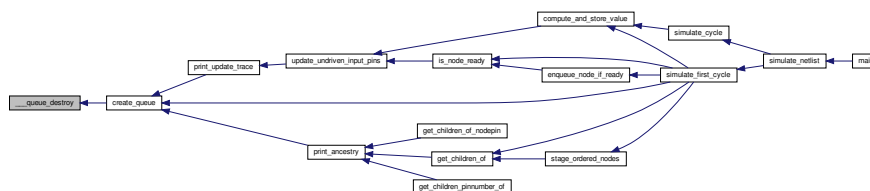
```
void __queue_add (
    queue_t * q,
    void * item )
```

Here is the caller graph for this function:



```
void __queue_destroy (
    queue_t * q )
```

Here is the caller graph for this function:

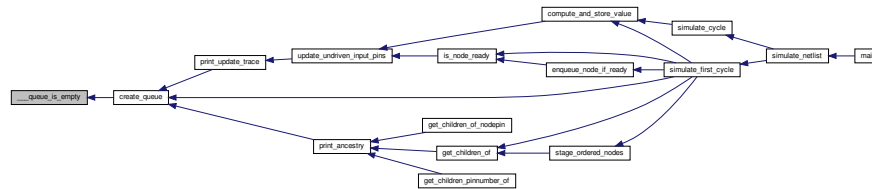


2.52.1.3 `__queue_is_empty()`

```
int __queue_is_empty (
    queue_t * q )
```

Definition at line 111 of file queue.cpp.

Here is the caller graph for this function:

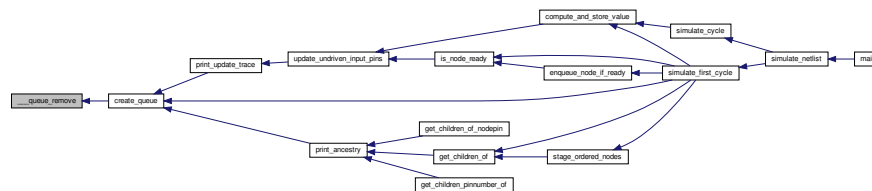


2.52.1.4 `__queue_remove()`

```
void * __queue_remove (
    queue_t * q )
```

Definition at line 77 of file queue.cpp.

Here is the caller graph for this function:

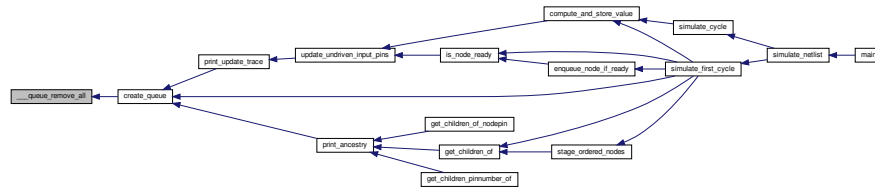


2.52.1.5 `__queue_remove_all()`

```
void ** __queue_remove_all (
    queue_t * q )
```

Definition at line 97 of file queue.cpp.

Here is the caller graph for this function:

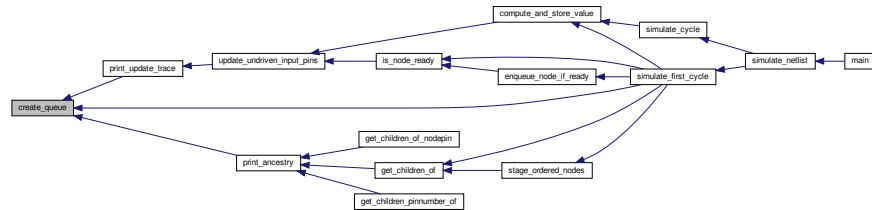


2.52.1.6 create_queue()

```
queue_t* create_queue ( )
```

Definition at line 36 of file queue.cpp.

Here is the caller graph for this function:



2.53 vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/queue.h File Reference

Data Structures

- struct [queue_t_t](#)
- struct [queue_node_t_t](#)

Macros

- `#define` [queue_t](#) struct [queue_t_t](#)
- `#define` [queue_node_t](#) struct [queue_node_t_t](#)

Functions

- [queue_t](#) * [create_queue](#) ()

2.53.1 Macro Definition Documentation

2.53.1.1 queue_node_t

```
#define queue_node_t struct queue_node_t_t
```

Definition at line 26 of file queue.h.

2.53.1.2 queue_t

```
#define queue_t struct queue_t_t
```

Definition at line 25 of file queue.h.

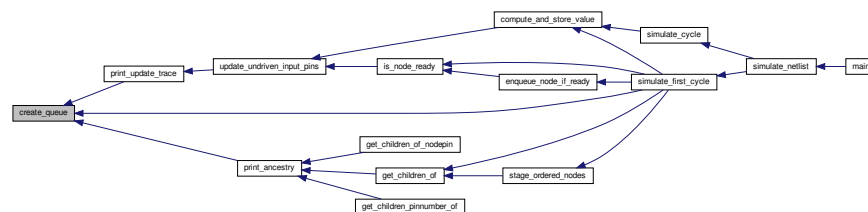
2.53.2 Function Documentation

2.53.2.1 create_queue()

```
queue_t* create_queue ( )
```

Definition at line 36 of file queue.cpp.

Here is the caller graph for this function:



2.54 vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/sim_block.h File Reference

```
#include "types.h"
```

Functions

- void [simulate_block_cycle](#) (int cycle, int num_input_pins, int *inputs, int num_output_pins, int *outputs)

2.54.1 Function Documentation

2.54.1.1 simulate_block_cycle()

```
void simulate_block_cycle (
    int cycle,
    int num_input_pins,
    int * inputs,
    int num_output_pins,
    int * outputs )
```

2.55 vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/simulate_blif.cpp File Reference

```
#include "simulate_blif.h"
#include "math.h"
#include "vtr_util.h"
#include "vtr_memory.h"
#include <string>
#include <sstream>
#include <chrono>
#include <dlfcn.h>
```

Macros

- #define [max](#)(a, b) (((a) > (b)) ? (a) : (b))
- #define [min](#)(a, b) ((a) > (b) ? (b) : (a))

Functions

- void [simulate_netlist](#) (netlist_t *netlist)
- void [simulate_cycle](#) (int cycle, stages_t *s)
- stages_t * [simulate_first_cycle](#) (netlist_t *netlist, int cycle, pin_names *p, lines_t *l)
- stages_t * [stage_ordered_nodes](#) (nnode_t **ordered_nodes, int num_ordered_nodes)
- void [compute_and_store_value](#) (nnode_t *node, int cycle)
- void [update_undriven_input_pins](#) (nnode_t *node, int cycle)
- void [flag_undriven_input_pins](#) (nnode_t *node)
- int [get_num_covered_nodes](#) (stages_t *s)
- int [enqueue_node_if_ready](#) (queue_t *queue, nnode_t *node, int cycle)
- int [is_node_complete](#) (nnode_t *node, int cycle)
- int [is_node_ready](#) (nnode_t *node, int cycle)

- int [get_clock_ratio](#) (nnode_t *node)
- void [set_clock_ratio](#) (int rat, nnode_t *node)
- nnode_t ** [get_children_of](#) (nnode_t *node, int *num_children)
- int * [get_children_pinnumber_of](#) (nnode_t *node, int *num_children)
- nnode_t ** [get_children_of_nodepin](#) (nnode_t *node, int *num_children, int output_pin)
- void [update_pin_value](#) (npin_t *pin, signed char value, int cycle)
- signed char [get_pin_value](#) (npin_t *pin, int cycle)
- int [get_values_offset](#) (int cycle)
- int [get_pin_cycle](#) (npin_t *pin)
- void [set_pin_cycle](#) (npin_t *pin, int cycle)
- int [is_even_cycle](#) (int cycle)
- void [initialize_pin](#) (npin_t *pin)
- int [is_clock_node](#) (nnode_t *node)
- int [is_posedge](#) (npin_t *pin, int cycle)
- void [compute_flipflop_node](#) (nnode_t *node, int cycle)
- void [compute_mux_2_node](#) (nnode_t *node, int cycle)
- void [compute_hard_ip_node](#) (nnode_t *node, int cycle)
- void [compute_multiply_node](#) (nnode_t *node, int cycle)
- void [compute_generic_node](#) (nnode_t *node, int cycle)
- int * [multiply_arrays](#) (int *a, int a_length, int *b, int b_length)
- void [compute_add_node](#) (nnode_t *node, int cycle, int type)
- int * [add_arrays](#) (int *a, int a_length, int *b, int b_length, int *c, int, int)
- void [compute_unary_sub_node](#) (nnode_t *node, int cycle)
- int * [unary_sub_arrays](#) (int *a, int a_length, int *c, int)
- void [compute_memory_node](#) (nnode_t *node, int cycle)
- void [compute_single_port_memory](#) (nnode_t *node, int cycle)
- void [compute_dual_port_memory](#) (nnode_t *node, int cycle)
- long [compute_memory_address](#) (signal_list_t *addr, int cycle)
- void [instantiate_memory](#) (nnode_t *node, int data_width, int addr_width)
- FILE * [preprocess_mif_file](#) (FILE *source)
- int [parse_mif_radix](#) (char *radix)
- void [assign_memory_from_mif_file](#) (FILE *mif, char *filename, int width, long depth, signed char *memory)
- void [assign_node_to_line](#) (nnode_t *node, [lines_t](#) *l, int type, int single_pin)
- void [insert_pin_into_line](#) (npin_t *pin, int pin_number, [line_t](#) *line, int type)
- [lines_t](#) * [create_lines](#) (netlist_t *netlist, int type)
- void [write_vector_headers](#) (FILE *file, [lines_t](#) *l)
- int [verify_test_vector_headers](#) (FILE *in, [lines_t](#) *l)
- int [verify_lines](#) ([lines_t](#) *l)
- int [find_portname_in_lines](#) (char *port_name, [lines_t](#) *l)
- [line_t](#) * [create_line](#) (char *name)
- char * [generate_vector_header](#) ([lines_t](#) *l)
- void [add_test_vector_to_lines](#) ([test_vector](#) *v, [lines_t](#) *l, int cycle)
- int [compare_test_vectors](#) ([test_vector](#) *v1, [test_vector](#) *v2)
- [test_vector](#) * [parse_test_vector](#) (char *buffer)
- [test_vector](#) * [generate_random_test_vector](#) ([lines_t](#) *l, int cycle, [hashtable_t](#) *hold_high_index, [hashtable_t](#) *hold_low_index)
- void [write_wave_to_file](#) ([lines_t](#) *l, FILE *file, int cycle_offset, int wave_length, int edge)
- void [write_vector_to_file](#) ([lines_t](#) *l, FILE *file, int cycle)
- void [write_wave_to_modelsim_file](#) (netlist_t *netlist, [lines_t](#) *l, FILE *modelsim_out, int cycle_offset, int wave_length)
- void [write_vector_to_modelsim_file](#) ([lines_t](#) *l, FILE *modelsim_out, int cycle)

- int [verify_output_vectors](#) (char *output_vector_file, int num_vectors)
- hashtable_t * [index_pin_name_list](#) (pin_names *list)
- pin_names * [parse_pin_name_list](#) (char *list)
- void [add_additional_items_to_lines](#) (nnode_t *node, pin_names *p, lines_t *l)
- char * [get_mif_filename](#) (nnode_t *node)
- void [trim_string](#) (char *string, const char *chars)
- int [line_has_unknown_pin](#) (line_t *line, int cycle)
- signed char [get_line_pin_value](#) (line_t *line, int pin_num, int cycle)
- char * [vector_value_to_hex](#) (signed char *value, int length)
- int [count_test_vectors](#) (FILE *in)
- int [is_vector](#) (char *buffer)
- int [get_next_vector](#) (FILE *file, char *buffer)
- void [free_lines](#) (lines_t *l)
- void [free_stages](#) (stages_t *s)
- void [free_test_vector](#) (test_vector *v)
- void [free_pin_name_list](#) (pin_names *p)
- int [print_progress_bar](#) (double completion, int position, int length, double time)
- void [print_netlist_stats](#) (stages_t *stages, int)
- void [print_simulation_stats](#) (stages_t *stages, int, double total_time, double simulation_time)
- void [print_time](#) (double time)
- void [print_ancestry](#) (nnode_t *bottom_node, int n)
- nnode_t * [print_update_trace](#) (nnode_t *bottom_node, int cycle)
- double [wall_time](#) ()
- char * [get_circuit_filename](#) ()

2.55.1 Macro Definition Documentation

2.55.1.1 max

```
#define max(  
    a,  
    b ) ((a) > (b)) ? (a) : (b)
```

Definition at line 36 of file simulate_blif.cpp.

2.55.1.2 min

```
#define min(  
    a,  
    b ) ((a) > (b)) ? (b) : (a)
```

Definition at line 37 of file simulate_blif.cpp.

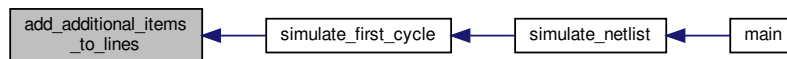
2.55.2 Function Documentation

2.55.2.1 add_additional_items_to_lines()

```
void add_additional_items_to_lines (
    nnode_t * node,
    pin_names * p,
    lines_t * l )
```

Definition at line 3138 of file simulate_blif.cpp.

Here is the caller graph for this function:



2.55.2.2 add_arrays()

```
int* add_arrays (
    int * a,
    int a_length,
    int * b,
    int b_length,
    int * c,
    int ,
    int )
```

Definition at line 1795 of file simulate_blif.cpp.

Here is the caller graph for this function:

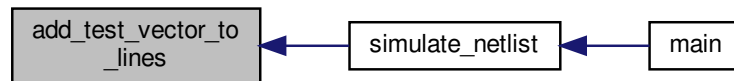


2.55.2.3 add_test_vector_to_lines()

```
void add_test_vector_to_lines (
    test_vector * v,
    lines_t * l,
    int cycle )
```

Definition at line 2638 of file simulate_blif.cpp.

Here is the caller graph for this function:



2.55.2.4 assign_memory_from_mif_file()

```
void assign_memory_from_mif_file (
    FILE * mif,
    char * filename,
    int width,
    long depth,
    signed char * memory )
```

Definition at line 2204 of file simulate_blif.cpp.

Here is the caller graph for this function:

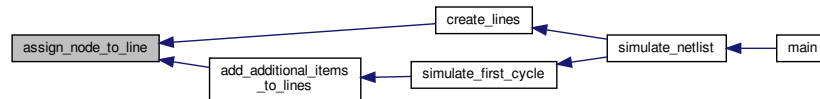


2.55.2.5 assign_node_to_line()

```
void assign_node_to_line (
    nnode_t * node,
    lines_t * l,
    int type,
    int single_pin )
```

Definition at line 2361 of file simulate_blif.cpp.

Here is the caller graph for this function:

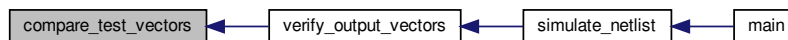


2.55.2.6 compare_test_vectors()

```
int compare_test_vectors (
    test_vector * v1,
    test_vector * v2 )
```

Definition at line 2666 of file simulate_blif.cpp.

Here is the caller graph for this function:

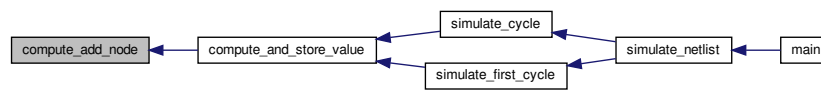


2.55.2.7 compute_add_node()

```
void compute_add_node (  
    nnode_t * node,  
    int cycle,  
    int type )
```

Definition at line 1733 of file simulate_blif.cpp.

Here is the caller graph for this function:

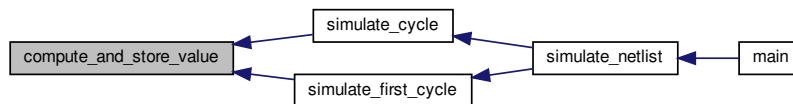


2.55.2.8 compute_and_store_value()

```
void compute_and_store_value (  
    nnode_t * node,  
    int cycle )
```

Definition at line 506 of file simulate_blif.cpp.

Here is the caller graph for this function:

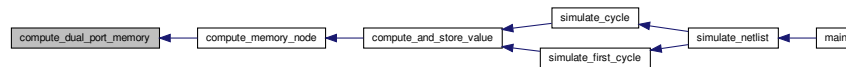


2.55.2.9 compute_dual_port_memory()

```
void compute_dual_port_memory (
    nnode_t * node,
    int cycle )
```

Definition at line 2026 of file simulate_blif.cpp.

Here is the caller graph for this function:

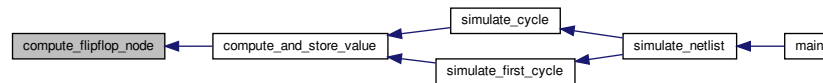


2.55.2.10 compute_flipflop_node()

```
void compute_flipflop_node (
    nnode_t * node,
    int cycle )
```

Definition at line 1458 of file simulate_blif.cpp.

Here is the caller graph for this function:

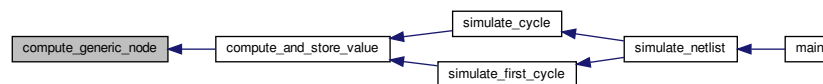


2.55.2.11 compute_generic_node()

```
void compute_generic_node (
    nnode_t * node,
    int cycle )
```

Definition at line 1658 of file simulate_blif.cpp.

Here is the caller graph for this function:

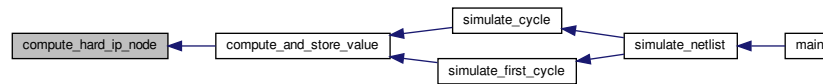


2.55.2.12 compute_hard_ip_node()

```
void compute_hard_ip_node (
    nnode_t * node,
    int cycle )
```

Definition at line 1551 of file simulate_blif.cpp.

Here is the caller graph for this function:

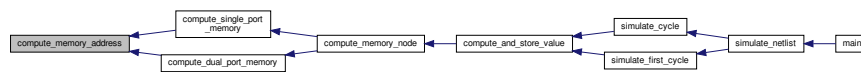


2.55.2.13 compute_memory_address()

```
long compute_memory_address (
    signal_list_t * addr,
    int cycle )
```

Definition at line 2096 of file simulate_blif.cpp.

Here is the caller graph for this function:

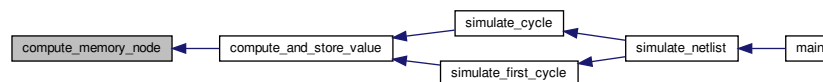


2.55.2.14 compute_memory_node()

```
void compute_memory_node (
    nnode_t * node,
    int cycle )
```

Definition at line 1958 of file simulate_blif.cpp.

Here is the caller graph for this function:

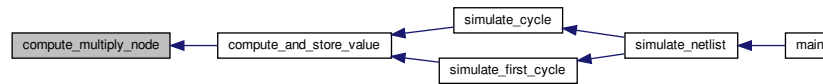


2.55.2.15 compute_multiply_node()

```
void compute_multiply_node (
    nnode_t * node,
    int cycle )
```

Definition at line 1612 of file simulate_blif.cpp.

Here is the caller graph for this function:

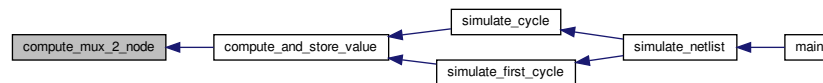


2.55.2.16 compute_mux_2_node()

```
void compute_mux_2_node (
    nnode_t * node,
    int cycle )
```

Definition at line 1478 of file simulate_blif.cpp.

Here is the caller graph for this function:

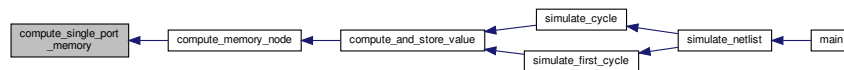


2.55.2.17 compute_single_port_memory()

```
void compute_single_port_memory (
    nnode_t * node,
    int cycle )
```

Definition at line 1972 of file simulate_blif.cpp.

Here is the caller graph for this function:

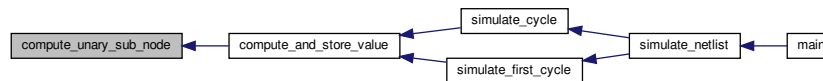


2.55.2.18 compute_unary_sub_node()

```
void compute_unary_sub_node (  
    nnode_t * node,  
    int cycle )
```

Definition at line 1873 of file simulate_blif.cpp.

Here is the caller graph for this function:

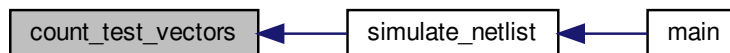


2.55.2.19 count_test_vectors()

```
int count_test_vectors (  
    FILE * in )
```

Definition at line 3317 of file simulate_blif.cpp.

Here is the caller graph for this function:

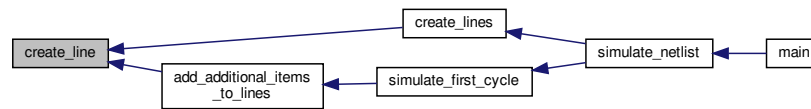


2.55.2.20 create_line()

```
line_t* create_line (  
    char * name )
```

Definition at line 2591 of file simulate_blif.cpp.

Here is the caller graph for this function:



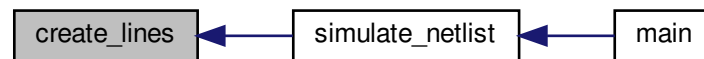
2.55.2.21 `create_lines()`

```

lines_t* create_lines (
    netlist_t * netlist,
    int type )
  
```

Definition at line 2448 of file `simulate_blif.cpp`.

Here is the caller graph for this function:



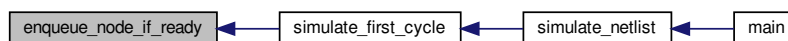
2.55.2.22 `enqueue_node_if_ready()`

```

int enqueue_node_if_ready (
    queue_t * queue,
    nnode_t * node,
    int cycle )
  
```

Definition at line 951 of file `simulate_blif.cpp`.

Here is the caller graph for this function:

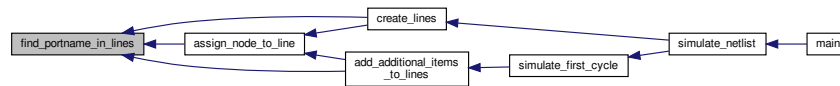


2.55.2.23 find_portname_in_lines()

```
int find_portname_in_lines (
    char * port_name,
    lines_t * l )
```

Definition at line 2578 of file simulate_blif.cpp.

Here is the caller graph for this function:

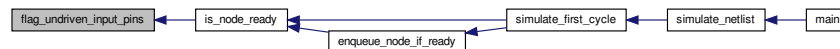


2.55.2.24 flag_undriven_input_pins()

```
void flag_undriven_input_pins (
    nnode_t * node )
```

Definition at line 885 of file simulate_blif.cpp.

Here is the caller graph for this function:

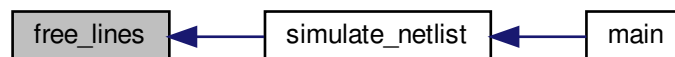


2.55.2.25 free_lines()

```
void free_lines (
    lines_t * l )
```

Definition at line 3373 of file simulate_blif.cpp.

Here is the caller graph for this function:

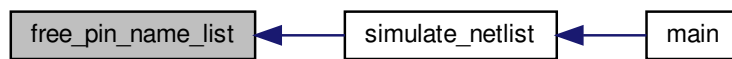


2.55.2.26 free_pin_name_list()

```
void free_pin_name_list (  
    pin_names * p )
```

Definition at line 3414 of file simulate_blif.cpp.

Here is the caller graph for this function:

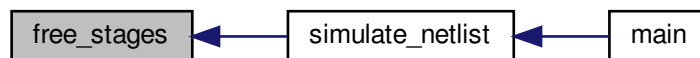


2.55.2.27 free_stages()

```
void free_stages (  
    stages_t * s )
```

Definition at line 3390 of file simulate_blif.cpp.

Here is the caller graph for this function:

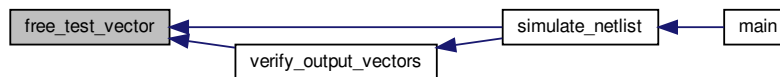


2.55.2.28 free_test_vector()

```
void free_test_vector (  
    test_vector * v )
```

Definition at line 3402 of file simulate_blif.cpp.

Here is the caller graph for this function:

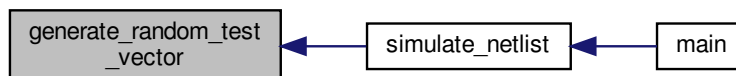


2.55.2.29 generate_random_test_vector()

```
test_vector* generate_random_test_vector (  
    lines_t * l,  
    int cycle,  
    hashtable_t * hold_high_index,  
    hashtable_t * hold_low_index )
```

Definition at line 2772 of file simulate_blif.cpp.

Here is the caller graph for this function:

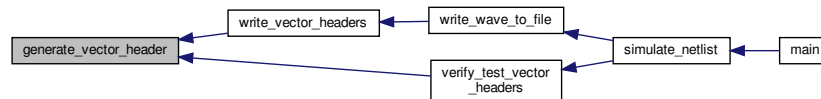


2.55.2.30 generate_vector_header()

```
char* generate_vector_header (
    lines_t * l )
```

Definition at line 2609 of file simulate_blif.cpp.

Here is the caller graph for this function:

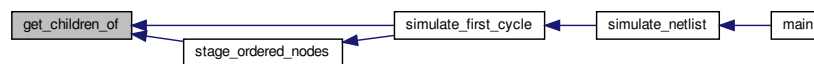


2.55.2.31 get_children_of()

```
nnode_t** get_children_of (
    nnode_t * node,
    int * num_children )
```

Definition at line 1058 of file simulate_blif.cpp.

Here is the caller graph for this function:



2.55.2.32 get_children_of_nodepin()

```
nnode_t** get_children_of_nodepin (
    nnode_t * node,
    int * num_children,
    int output_pin )
```

Definition at line 1224 of file simulate_blif.cpp.

2.55.2.33 get_children_pinnumber_of()

```
int* get_children_pinnumber_of (
    nnode_t * node,
    int * num_children )
```

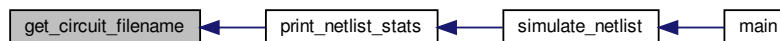
Definition at line 1141 of file simulate_blif.cpp.

2.55.2.34 get_circuit_filename()

```
char* get_circuit_filename ( )
```

Definition at line 3714 of file simulate_blif.cpp.

Here is the caller graph for this function:



2.55.2.35 get_clock_ratio()

```
int get_clock_ratio (
    nnode_t * node )
```

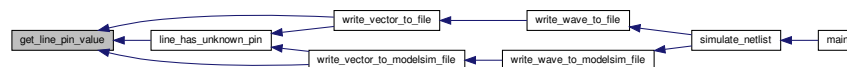
Definition at line 1036 of file simulate_blif.cpp.

2.55.2.36 get_line_pin_value()

```
signed char get_line_pin_value (
    line_t * line,
    int pin_num,
    int cycle )
```

Definition at line 3286 of file simulate_blif.cpp.

Here is the caller graph for this function:

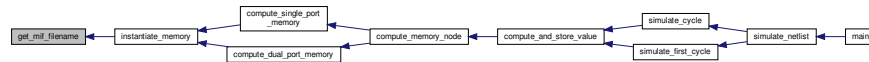


2.55.2.37 get_mif_filename()

```
char* get_mif_filename (  
    nnode_t * node )
```

Definition at line 3219 of file simulate_blif.cpp.

Here is the caller graph for this function:

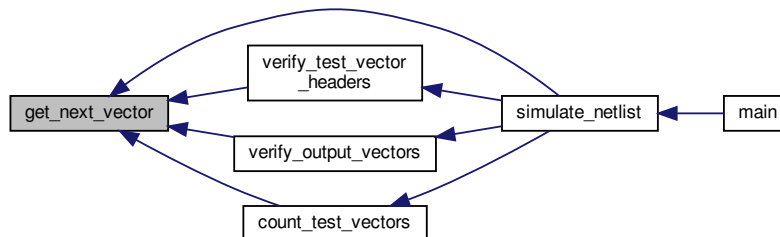


2.55.2.38 get_next_vector()

```
int get_next_vector (  
    FILE * file,  
    char * buffer )
```

Definition at line 3361 of file simulate_blif.cpp.

Here is the caller graph for this function:

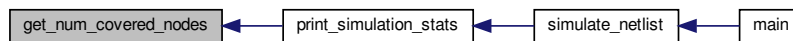


2.55.2.39 `get_num_covered_nodes()`

```
int get_num_covered_nodes (
    stages_t * s )
```

Definition at line 916 of file `simulate_blif.cpp`.

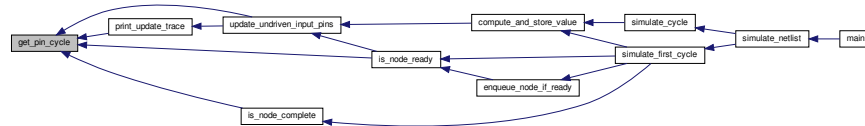
Here is the caller graph for this function:

2.55.2.40 `get_pin_cycle()`

```
int get_pin_cycle (
    npin_t * pin ) [inline]
```

Definition at line 1349 of file `simulate_blif.cpp`.

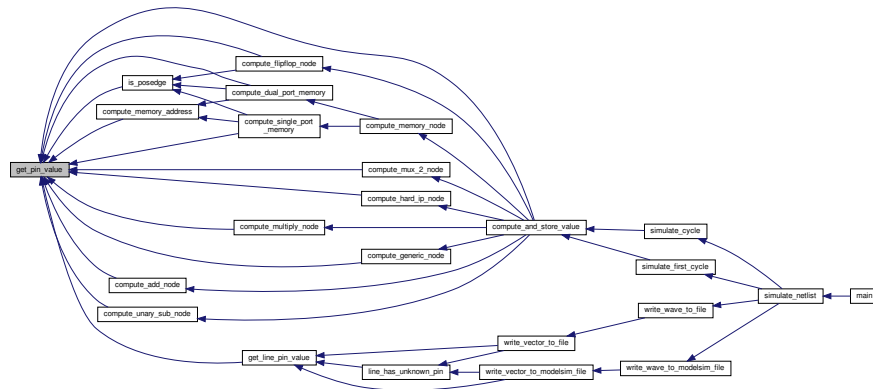
Here is the caller graph for this function:

2.55.2.41 `get_pin_value()`

```
signed char get_pin_value (
    npin_t * pin,
    int cycle )
```

Definition at line 1324 of file `simulate_blif.cpp`.

Here is the caller graph for this function:

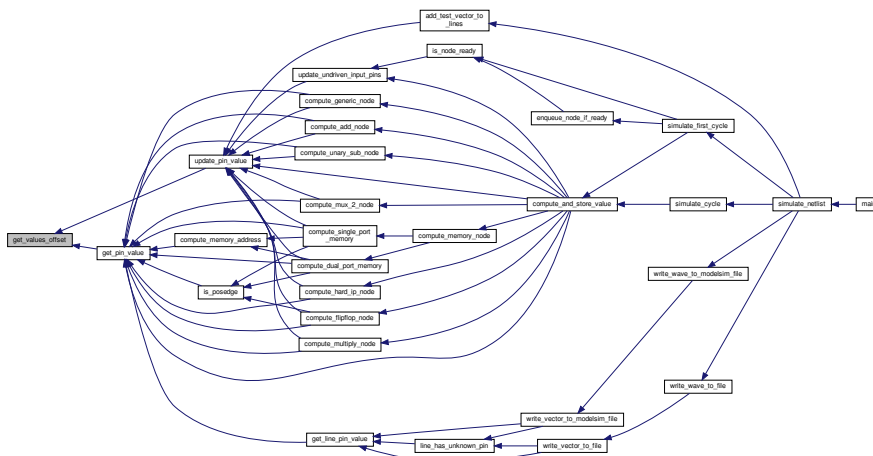


2.55.2.42 get_values_offset()

```
int get_values_offset (
    int cycle ) [inline]
```

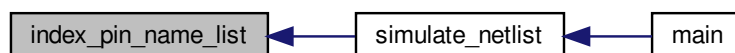
Definition at line 1341 of file `simulate_blif.cpp`.

Here is the caller graph for this function:



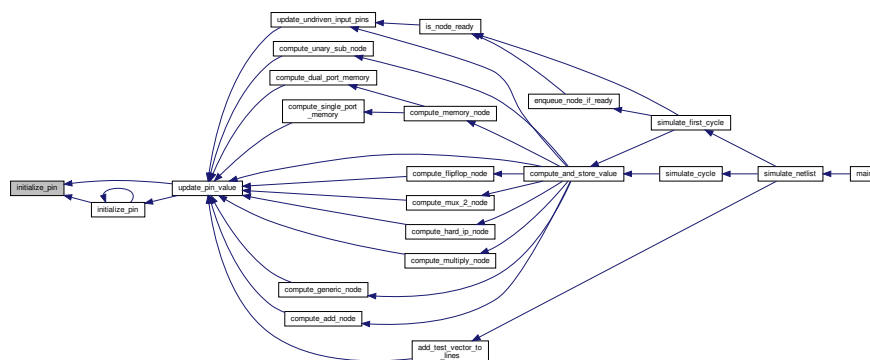
```
hashtable_t* index_pin_name_list (
    pin_names * list )
```

Here is the caller graph for this function:



```
void initialize_pin (
    npin_t * pin )
```

Here is the caller graph for this function:

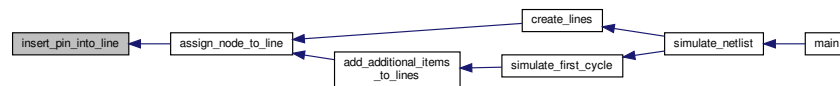


2.55.2.45 insert_pin_into_line()

```
void insert_pin_into_line (
    npin_t * pin,
    int pin_number,
    line_t * line,
    int type )
```

Definition at line 2413 of file simulate_blif.cpp.

Here is the caller graph for this function:

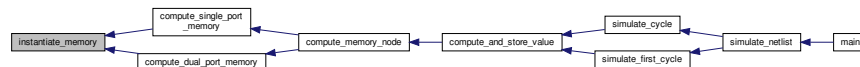


2.55.2.46 instantiate_memory()

```
void instantiate_memory (
    nnode_t * node,
    int data_width,
    int addr_width )
```

Definition at line 2116 of file simulate_blif.cpp.

Here is the caller graph for this function:

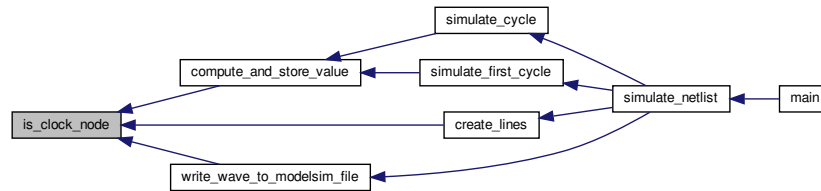


2.55.2.47 is_clock_node()

```
int is_clock_node (
    nnode_t * node ) [inline]
```

Definition at line 1434 of file simulate_blif.cpp.

Here is the caller graph for this function:

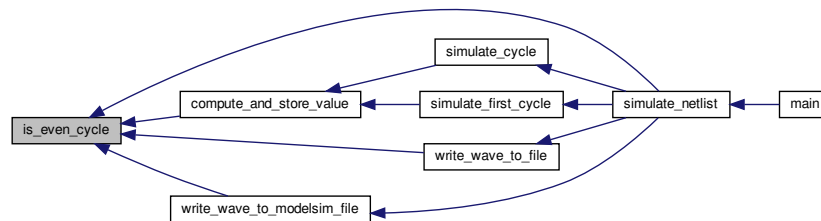


2.55.2.48 `is_even_cycle()`

```
int is_even_cycle (
    int cycle )
```

Definition at line 1370 of file `simulate_blif.cpp`.

Here is the caller graph for this function:

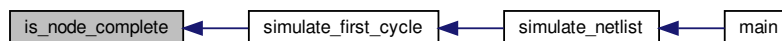


2.55.2.49 `is_node_complete()`

```
int is_node_complete (
    nnode_t * node,
    int cycle )
```

Definition at line 968 of file `simulate_blif.cpp`.

Here is the caller graph for this function:

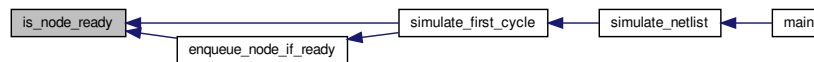


2.55.2.50 is_node_ready()

```
int is_node_ready (
    nnode_t * node,
    int cycle )
```

Definition at line 981 of file simulate_blif.cpp.

Here is the caller graph for this function:

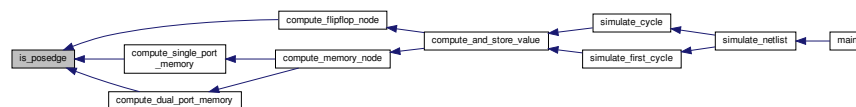


2.55.2.51 is_posedge()

```
int is_posedge (
    npin_t * pin,
    int cycle )
```

Definition at line 1447 of file simulate_blif.cpp.

Here is the caller graph for this function:

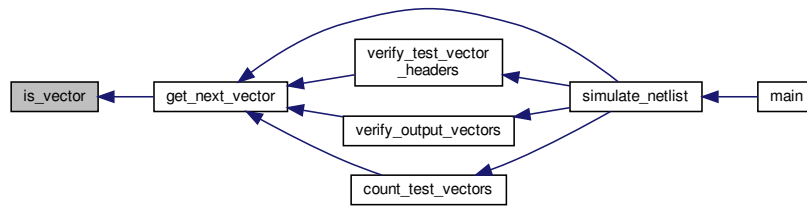


2.55.2.52 is_vector()

```
int is_vector (
    char * buffer )
```

Definition at line 3338 of file simulate_blif.cpp.

Here is the caller graph for this function:



2.55.2.53 line_has_unknown_pin()

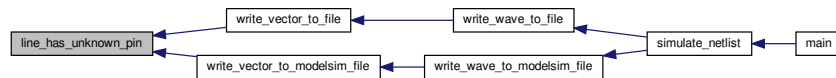
```

int line_has_unknown_pin (
    line_t * line,
    int cycle )

```

Definition at line 3267 of file `simulate_blif.cpp`.

Here is the caller graph for this function:



2.55.2.54 multiply_arrays()

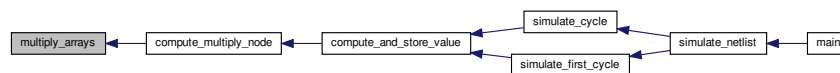
```

int* multiply_arrays (
    int * a,
    int a_length,
    int * b,
    int b_length )

```

Definition at line 1703 of file `simulate_blif.cpp`.

Here is the caller graph for this function:

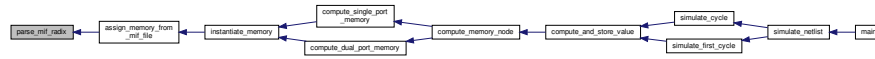


2.55.2.55 parse_mif_radix()

```
int parse_mif_radix (
    char * radix )
```

Definition at line 2183 of file simulate_blif.cpp.

Here is the caller graph for this function:

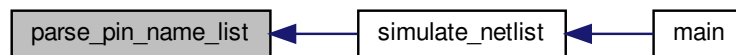


2.55.2.56 parse_pin_name_list()

```
pin_names* parse_pin_name_list (
    char * list )
```

Definition at line 3112 of file simulate_blif.cpp.

Here is the caller graph for this function:

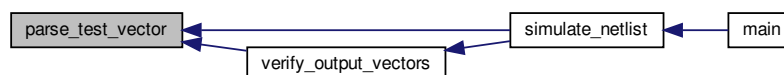


2.55.2.57 parse_test_vector()

```
test_vector* parse_test_vector (
    char * buffer )
```

Definition at line 2703 of file simulate_blif.cpp.

Here is the caller graph for this function:

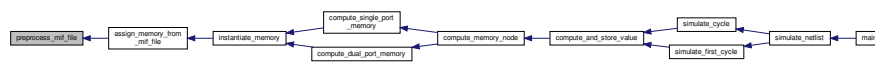


2.55.2.58 preprocess_mif_file()

```
FILE* preprocess_mif_file (
    FILE * source )
```

Definition at line 2145 of file simulate_blif.cpp.

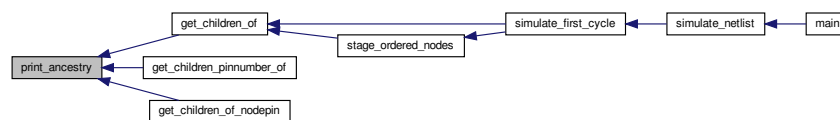
Here is the caller graph for this function:

**2.55.2.59 print_ancestry()**

```
void print_ancestry (
    nnode_t * bottom_node,
    int n )
```

Definition at line 3520 of file simulate_blif.cpp.

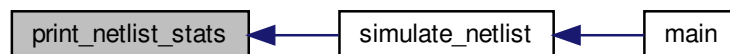
Here is the caller graph for this function:

**2.55.2.60 print_netlist_stats()**

```
void print_netlist_stats (
    stages_t * stages,
    int )
```

Definition at line 3473 of file simulate_blif.cpp.

Here is the caller graph for this function:

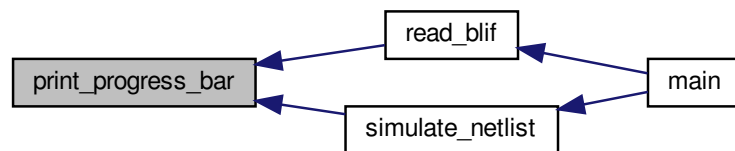


2.55.2.61 print_progress_bar()

```
int print_progress_bar (  
    double completion,  
    int position,  
    int length,  
    double time )
```

Definition at line 3431 of file simulate_blif.cpp.

Here is the caller graph for this function:

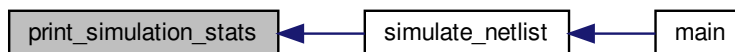


2.55.2.62 print_simulation_stats()

```
void print_simulation_stats (  
    stages_t * stages,  
    int ,  
    double total_time,  
    double simulation_time )
```

Definition at line 3491 of file simulate_blif.cpp.

Here is the caller graph for this function:

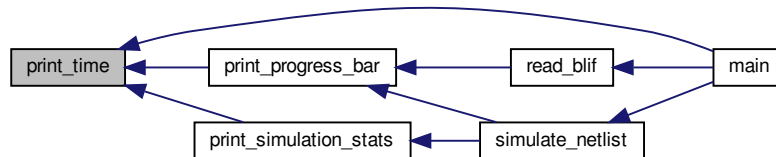


2.55.2.63 print_time()

```
void print_time (
    double time )
```

Definition at line 3507 of file simulate_blif.cpp.

Here is the caller graph for this function:

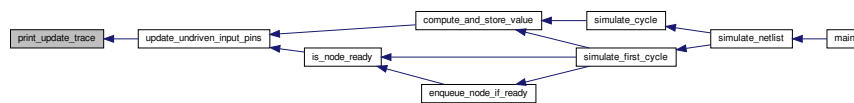


2.55.2.64 print_update_trace()

```
nnode_t* print_update_trace (
    nnode_t * bottom_node,
    int cycle )
```

Definition at line 3611 of file simulate_blif.cpp.

Here is the caller graph for this function:



2.55.2.65 set_clock_ratio()

```
void set_clock_ratio (
    int rat,
    nnode_t * node )
```

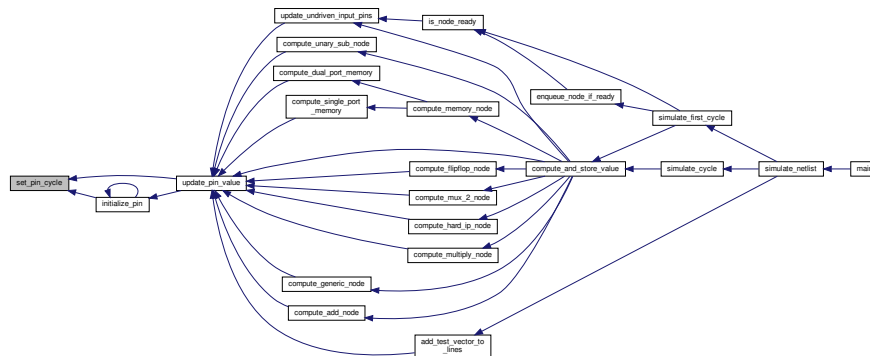
Definition at line 1044 of file simulate_blif.cpp.

2.55.2.66 set_pin_cycle()

```
void set_pin_cycle (
    npin_t * pin,
    int cycle ) [inline]
```

Definition at line 1362 of file simulate_blif.cpp.

Here is the caller graph for this function:

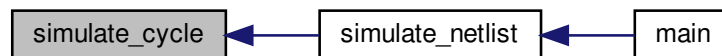


2.55.2.67 simulate_cycle()

```
void simulate_cycle (
    int cycle,
    stages_t * s )
```

Definition at line 275 of file simulate_blif.cpp.

Here is the caller graph for this function:

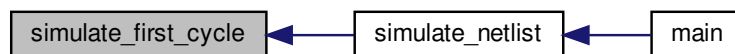


2.55.2.68 simulate_first_cycle()

```
stages_t* simulate_first_cycle (  
    netlist_t * netlist,  
    int cycle,  
    pin_names * p,  
    lines_t * l )
```

Definition at line 361 of file `simulate_blif.cpp`.

Here is the caller graph for this function:



2.55.2.69 simulate_netlist()

```
void simulate_netlist (  
    netlist_t * netlist )
```

Definition at line 43 of file `simulate_blif.cpp`.

Here is the caller graph for this function:

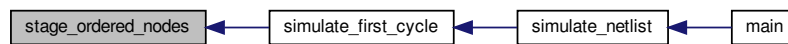


2.55.2.70 stage_ordered_nodes()

```
stages_t* stage_ordered_nodes (
    nnode_t ** ordered_nodes,
    int num_ordered_nodes )
```

Definition at line 420 of file simulate_blif.cpp.

Here is the caller graph for this function:

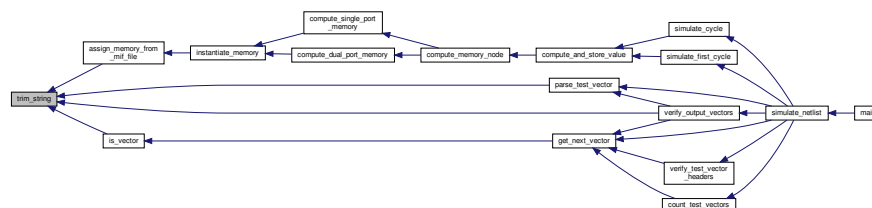


2.55.2.71 trim_string()

```
void trim_string (
    char * string,
    const char * chars )
```

Definition at line 3239 of file simulate_blif.cpp.

Here is the caller graph for this function:

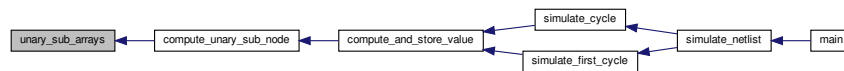


2.55.2.72 unary_sub_arrays()

```
int* unary_sub_arrays (
    int * a,
    int a_length,
    int * c,
    int )
```

Definition at line 1931 of file simulate_blif.cpp.

Here is the caller graph for this function:

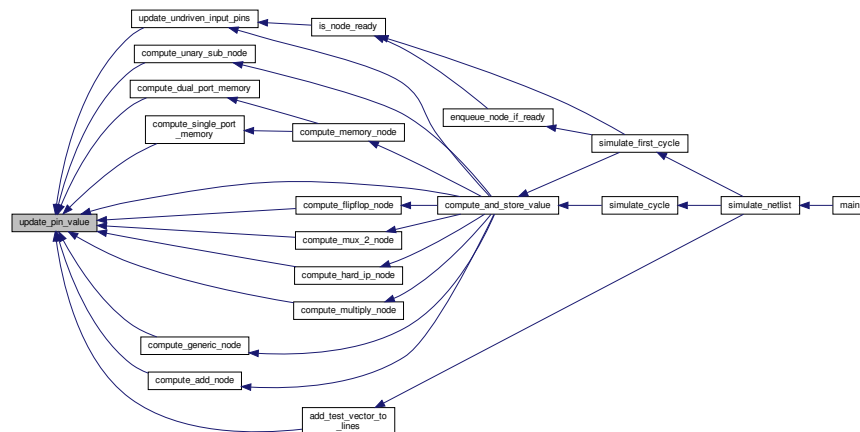


2.55.2.73 update_pin_value()

```
void update_pin_value (
    npin_t * pin,
    signed char value,
    int cycle )
```

Definition at line 1312 of file simulate_blif.cpp.

Here is the caller graph for this function:

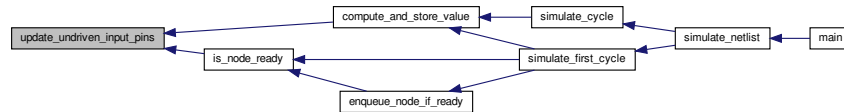


2.55.2.74 update_undriven_input_pins()

```
void update_undriven_input_pins (
    nnode_t * node,
    int cycle )
```

Definition at line 842 of file simulate_blif.cpp.

Here is the caller graph for this function:



2.55.2.75 vector_value_to_hex()

```
char* vector_value_to_hex (
    signed char * value,
    int length )
```

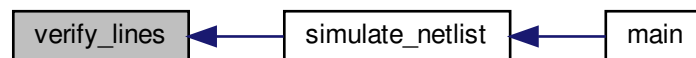
Definition at line 3295 of file simulate_blif.cpp.

2.55.2.76 verify_lines()

```
int verify_lines (
    lines_t * l )
```

Definition at line 2556 of file simulate_blif.cpp.

Here is the caller graph for this function:

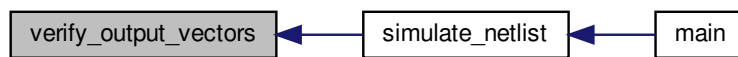


2.55.2.77 verify_output_vectors()

```
int verify_output_vectors (
    char * output_vector_file,
    int num_vectors )
```

Definition at line 2997 of file simulate_blif.cpp.

Here is the caller graph for this function:

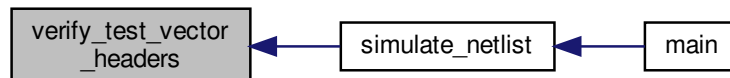


2.55.2.78 verify_test_vector_headers()

```
int verify_test_vector_headers (
    FILE * in,
    lines_t * l )
```

Definition at line 2496 of file simulate_blif.cpp.

Here is the caller graph for this function:

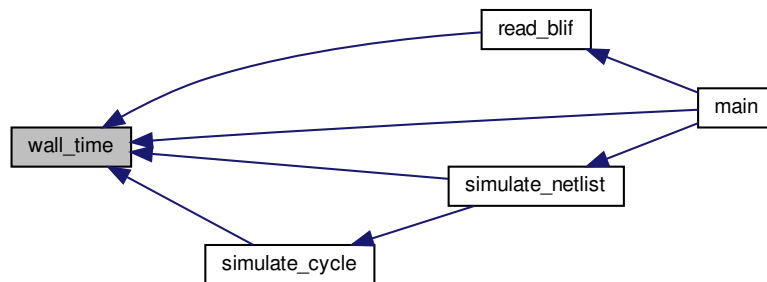


2.55.2.79 wall_time()

```
double wall_time ( )
```

Definition at line 3701 of file simulate_blif.cpp.

Here is the caller graph for this function:

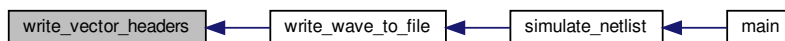


2.55.2.80 write_vector_headers()

```
void write_vector_headers (
    FILE * file,
    lines_t * l )
```

Definition at line 2482 of file simulate_blif.cpp.

Here is the caller graph for this function:

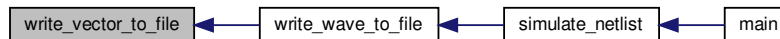


2.55.2.81 write_vector_to_file()

```
void write_vector_to_file (
    lines_t * l,
    FILE * file,
    int cycle )
```

Definition at line 2843 of file simulate_blif.cpp.

Here is the caller graph for this function:

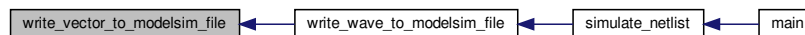


2.55.2.82 write_vector_to_modelsim_file()

```
void write_vector_to_modelsim_file (
    lines_t * l,
    FILE * modelsim_out,
    int cycle )
```

Definition at line 2945 of file simulate_blif.cpp.

Here is the caller graph for this function:

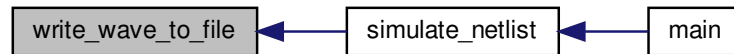


2.55.2.83 write_wave_to_file()

```
void write_wave_to_file (
    lines_t * l,
    FILE * file,
    int cycle_offset,
    int wave_length,
    int edge )
```

Definition at line 2826 of file simulate_blif.cpp.

Here is the caller graph for this function:



2.55.2.84 write_wave_to_modelsim_file()

```

void write_wave_to_modelsim_file (
    netlist_t * netlist,
    lines_t * l,
    FILE * modelsim_out,
    int cycle_offset,
    int wave_length )
  
```

Definition at line 2914 of file simulate_blif.cpp.

Here is the caller graph for this function:



2.56 vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/simulate_blif.h File Reference

```

#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <stdarg.h>
#include "queue.h"
#include "hashtable.h"
#include "sim_block.h"
#include "types.h"
#include "globals.h"
#include "netlist_utils.h"
#include "odin_util.h"
#include "multipliers.h"
#include "hard_blocks.h"
#include "memories.h"
#include "ace.h"
  
```

Data Structures

- struct [pin_names](#)
- struct [line_t](#)
- struct [lines_t](#)
- struct [stages_t](#)
- struct [test_vector](#)

Macros

- #define [SIM_WAVE_LENGTH](#) 16
- #define [BUFFER_MAX_SIZE](#) 1024
- #define [INPUT_VECTOR_FILE_NAME](#) "input_vectors"
- #define [OUTPUT_VECTOR_FILE_NAME](#) "output_vectors"
- #define [OUTPUT_ACTIVITY_FILE_NAME](#) "output_activity"
- #define [SINGLE_PORT_MEMORY_NAME](#) "single_port_ram"
- #define [DUAL_PORT_MEMORY_NAME](#) "dual_port_ram"

Functions

- void [simulate_netlist](#) (netlist_t *netlist)
- void [simulate_cycle](#) (int cycle, [stages_t](#) *s)
- [stages_t](#) * [simulate_first_cycle](#) (netlist_t *netlist, int cycle, [pin_names](#) *p, [lines_t](#) *output_lines)
- [stages_t](#) * [stage_ordered_nodes](#) (nnode_t **ordered_nodes, int num_ordered_nodes)
- void [free_stages](#) ([stages_t](#) *s)
- int [get_num_covered_nodes](#) ([stages_t](#) *s)
- int [get_clock_ratio](#) (nnode_t *node)
- void [set_clock_ratio](#) (int rat, nnode_t *node)
- nnode_t ** [get_children_of](#) (nnode_t *node, int *count)
- int * [get_children_pinnumber_of](#) (nnode_t *node, int *num_children)
- nnode_t ** [get_children_of_nodepin](#) (nnode_t *node, int *count, int output_pin)
- int [is_node_ready](#) (nnode_t *node, int cycle)
- int [is_node_complete](#) (nnode_t *node, int cycle)
- int [enqueue_node_if_ready](#) ([queue_t](#) *queue, nnode_t *node, int cycle)
- void [compute_and_store_value](#) (nnode_t *node, int cycle)
- void [compute_memory_node](#) (nnode_t *node, int cycle)
- void [compute_hard_ip_node](#) (nnode_t *node, int cycle)
- void [compute_multiply_node](#) (nnode_t *node, int cycle)
- void [compute_generic_node](#) (nnode_t *node, int cycle)
- void [compute_add_node](#) (nnode_t *node, int cycle, int type)
- void [compute_unary_sub_node](#) (nnode_t *node, int cycle)
- void [update_pin_value](#) (npin_t *pin, signed char value, int cycle)
- signed char [get_pin_value](#) (npin_t *pin, int cycle)
- int [get_values_offset](#) (int cycle)
- int [get_pin_cycle](#) (npin_t *pin)
- void [set_pin_cycle](#) (npin_t *pin, int cycle)
- void [initialize_pin](#) (npin_t *pin)
- int [is_even_cycle](#) (int cycle)
- int [is_clock_node](#) (nnode_t *node)

- signed char [get_line_pin_value](#) ([line_t](#) *line, int pin_num, int cycle)
- int [line_has_unknown_pin](#) ([line_t](#) *line, int cycle)
- void [compute_flipflop_node](#) (nnode_t *node, int cycle)
- void [compute_mux_2_node](#) (nnode_t *node, int cycle)
- int * [multiply_arrays](#) (int *a, int a_length, int *b, int b_length)
- int * [add_arrays](#) (int *a, int a_length, int *b, int b_length, int *c, int c_length, int flag)
- int * [unary_sub_arrays](#) (int *a, int a_length, int *c, int c_length)
- void [compute_single_port_memory](#) (nnode_t *node, int cycle)
- void [compute_dual_port_memory](#) (nnode_t *node, int cycle)
- long [compute_memory_address](#) (signal_list_t *addr, int cycle)
- void [instantiate_memory](#) (nnode_t *node, int data_width, int addr_width)
- char * [get_mif_filename](#) (nnode_t *node)
- FILE * [preprocess_mif_file](#) (FILE *source)
- void [assign_memory_from_mif_file](#) (FILE *file, char *filename, int width, long depth, signed char *memory)
- int [parse_mif_radix](#) (char *radix)
- int [count_test_vectors](#) (FILE *in)
- int [is_vector](#) (char *buffer)
- int [get_next_vector](#) (FILE *file, char *buffer)
- test_vector * [parse_test_vector](#) (char *buffer)
- test_vector * [generate_random_test_vector](#) ([lines_t](#) *l, int cycle, [hashtable_t](#) *hold_high_index, [hashtable_t](#) *hold_low_index)
- int [compare_test_vectors](#) (test_vector *v1, test_vector *v2)
- int [verify_test_vector_headers](#) (FILE *in, [lines_t](#) *l)
- void [free_test_vector](#) (test_vector *v)
- [line_t](#) * [create_line](#) (char *name)
- int [verify_lines](#) ([lines_t](#) *l)
- void [free_lines](#) ([lines_t](#) *l)
- int [find_portname_in_lines](#) (char *port_name, [lines_t](#) *l)
- [lines_t](#) * [create_lines](#) (netlist_t *netlist, int type)
- void [add_test_vector_to_lines](#) (test_vector *v, [lines_t](#) *l, int cycle)
- void [assign_node_to_line](#) (nnode_t *node, [lines_t](#) *l, int type, int single_pin)
- void [insert_pin_into_line](#) (npin_t *pin, int pin_number, [line_t](#) *line, int type)
- char * [generate_vector_header](#) ([lines_t](#) *l)
- void [write_vector_headers](#) (FILE *file, [lines_t](#) *l)
- void [write_vector_to_file](#) ([lines_t](#) *l, FILE *file, int cycle)
- void [write_wave_to_file](#) ([lines_t](#) *l, FILE *file, int cycle_offset, int wave_length, int both_edges)
- void [write_vector_to_modelsim_file](#) ([lines_t](#) *l, FILE *modelsim_out, int cycle)
- void [write_wave_to_modelsim_file](#) (netlist_t *netlist, [lines_t](#) *l, FILE *modelsim_out, int cycle_offset, int wave_length)
- int [verify_output_vectors](#) (char *output_vector_file, int num_test_vectors)
- void [add_additional_items_to_lines](#) (nnode_t *node, [pin_names](#) *p, [lines_t](#) *l)
- [pin_names](#) * [parse_pin_name_list](#) (char *list)
- void [free_pin_name_list](#) ([pin_names](#) *p)
- [hashtable_t](#) * [index_pin_name_list](#) ([pin_names](#) *list)
- void [trim_string](#) (char *string, const char *chars)
- char * [vector_value_to_hex](#) (signed char *value, int length)
- int [print_progress_bar](#) (double completion, int position, int length, double time)
- void [print_netlist_stats](#) ([stages_t](#) *stages, int num_vectors)
- void [print_simulation_stats](#) ([stages_t](#) *stages, int num_vectors, double total_time, double simulation_time)
- void [print_time](#) (double time)
- double [wall_time](#) ()

- char * [get_circuit_filename](#) ()
- void [update_undriven_input_pins](#) (nnode_t *node, int cycle)
- void [flag_undriven_input_pins](#) (nnode_t *node)
- void [print_ancestry](#) (nnode_t *node, int generations)
- nnode_t * [print_update_trace](#) (nnode_t *bottom_node, int cycle)
- int [is_posedge](#) (npin_t *pin, int cycle)

2.56.1 Macro Definition Documentation

2.56.1.1 BUFFER_MAX_SIZE

```
#define BUFFER_MAX_SIZE 1024
```

Definition at line 31 of file `simulate_blif.h`.

2.56.1.2 DUAL_PORT_MEMORY_NAME

```
#define DUAL_PORT_MEMORY_NAME "dual_port_ram"
```

Definition at line 57 of file `simulate_blif.h`.

2.56.1.3 INPUT_VECTOR_FILE_NAME

```
#define INPUT_VECTOR_FILE_NAME "input_vectors"
```

Definition at line 52 of file `simulate_blif.h`.

2.56.1.4 OUTPUT_ACTIVITY_FILE_NAME

```
#define OUTPUT_ACTIVITY_FILE_NAME "output_activity"
```

Definition at line 54 of file `simulate_blif.h`.

2.56.1.5 OUTPUT_VECTOR_FILE_NAME

```
#define OUTPUT_VECTOR_FILE_NAME "output_vectors"
```

Definition at line 53 of file `simulate_blif.h`.

2.56.1.6 SIM_WAVE_LENGTH

```
#define SIM_WAVE_LENGTH 16
```

Definition at line 30 of file simulate_blif.h.

2.56.1.7 SINGLE_PORT_MEMORY_NAME

```
#define SINGLE_PORT_MEMORY_NAME "single_port_ram"
```

Definition at line 56 of file simulate_blif.h.

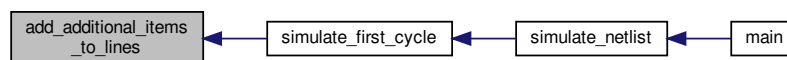
2.56.2 Function Documentation

2.56.2.1 add_additional_items_to_lines()

```
void add_additional_items_to_lines (
    nnode_t * node,
    pin_names * p,
    lines_t * l )
```

Definition at line 3138 of file simulate_blif.cpp.

Here is the caller graph for this function:



2.56.2.2 add_arrays()

```
int* add_arrays (  
    int * a,  
    int a_length,  
    int * b,  
    int b_length,  
    int * c,  
    int c_length,  
    int flag )
```

Definition at line 1795 of file simulate_blif.cpp.

Here is the caller graph for this function:

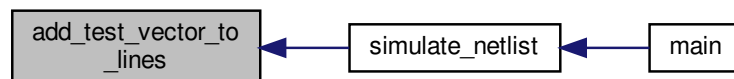


2.56.2.3 add_test_vector_to_lines()

```
void add_test_vector_to_lines (  
    test_vector * v,  
    lines_t * l,  
    int cycle )
```

Definition at line 2638 of file simulate_blif.cpp.

Here is the caller graph for this function:

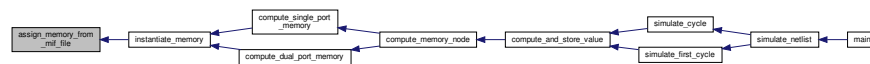


2.56.2.4 assign_memory_from_mif_file()

```
void assign_memory_from_mif_file (
    FILE * file,
    char * filename,
    int width,
    long depth,
    signed char * memory )
```

Definition at line 2204 of file simulate_blif.cpp.

Here is the caller graph for this function:

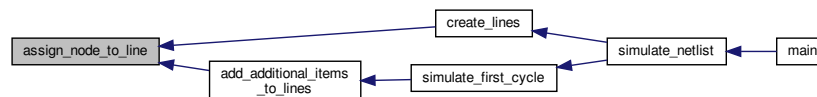


2.56.2.5 assign_node_to_line()

```
void assign_node_to_line (
    nnode_t * node,
    lines_t * l,
    int type,
    int single_pin )
```

Definition at line 2361 of file simulate_blif.cpp.

Here is the caller graph for this function:

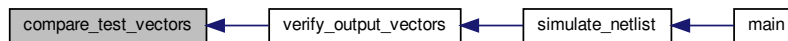


2.56.2.6 compare_test_vectors()

```
int compare_test_vectors (
    test_vector * v1,
    test_vector * v2 )
```

Definition at line 2666 of file simulate_blif.cpp.

Here is the caller graph for this function:

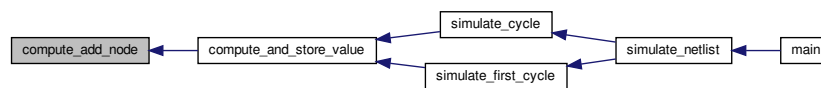


2.56.2.7 compute_add_node()

```
void compute_add_node (
    nnode_t * node,
    int cycle,
    int type )
```

Definition at line 1733 of file simulate_blif.cpp.

Here is the caller graph for this function:

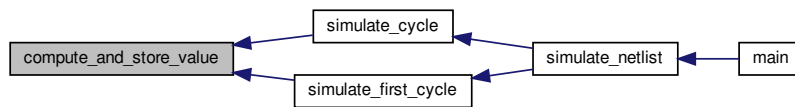


2.56.2.8 compute_and_store_value()

```
void compute_and_store_value (
    nnode_t * node,
    int cycle )
```

Definition at line 506 of file simulate_blif.cpp.

Here is the caller graph for this function:



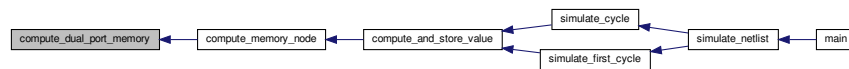
2.56.2.9 `compute_dual_port_memory()`

```

void compute_dual_port_memory (
    nnode_t * node,
    int cycle )
  
```

Definition at line 2026 of file `simulate_blif.cpp`.

Here is the caller graph for this function:



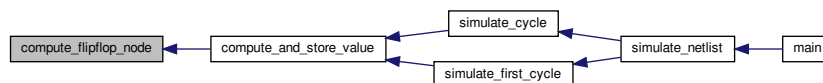
2.56.2.10 `compute_flipflop_node()`

```

void compute_flipflop_node (
    nnode_t * node,
    int cycle )
  
```

Definition at line 1458 of file `simulate_blif.cpp`.

Here is the caller graph for this function:

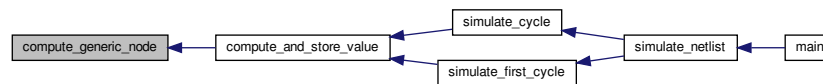


2.56.2.11 compute_generic_node()

```
void compute_generic_node (
    nnode_t * node,
    int cycle )
```

Definition at line 1658 of file simulate_blif.cpp.

Here is the caller graph for this function:

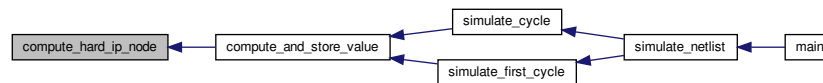


2.56.2.12 compute_hard_ip_node()

```
void compute_hard_ip_node (
    nnode_t * node,
    int cycle )
```

Definition at line 1551 of file simulate_blif.cpp.

Here is the caller graph for this function:

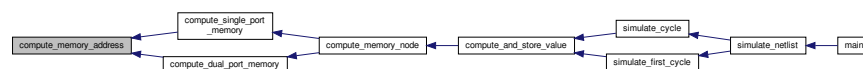


2.56.2.13 compute_memory_address()

```
long compute_memory_address (
    signal_list_t * addr,
    int cycle )
```

Definition at line 2096 of file simulate_blif.cpp.

Here is the caller graph for this function:

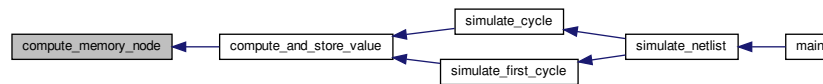


2.56.2.14 compute_memory_node()

```
void compute_memory_node (
    nnode_t * node,
    int cycle )
```

Definition at line 1958 of file simulate_blif.cpp.

Here is the caller graph for this function:

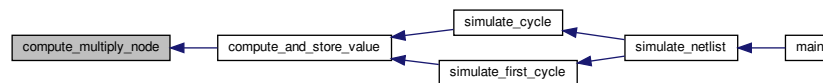


2.56.2.15 compute_multiply_node()

```
void compute_multiply_node (
    nnode_t * node,
    int cycle )
```

Definition at line 1612 of file simulate_blif.cpp.

Here is the caller graph for this function:

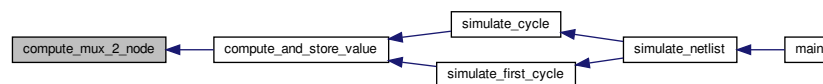


2.56.2.16 compute_mux_2_node()

```
void compute_mux_2_node (
    nnode_t * node,
    int cycle )
```

Definition at line 1478 of file simulate_blif.cpp.

Here is the caller graph for this function:

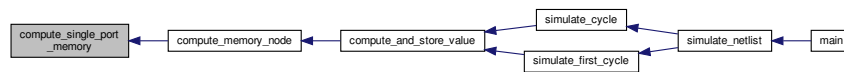


2.56.2.17 compute_single_port_memory()

```
void compute_single_port_memory (
    nnode_t * node,
    int cycle )
```

Definition at line 1972 of file simulate_blif.cpp.

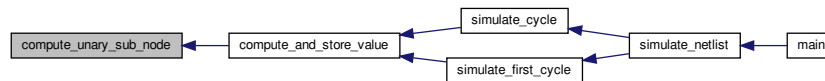
Here is the caller graph for this function:

**2.56.2.18 compute_unary_sub_node()**

```
void compute_unary_sub_node (
    nnode_t * node,
    int cycle )
```

Definition at line 1873 of file simulate_blif.cpp.

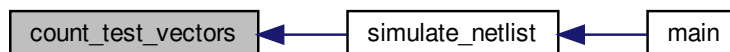
Here is the caller graph for this function:

**2.56.2.19 count_test_vectors()**

```
int count_test_vectors (
    FILE * in )
```

Definition at line 3317 of file simulate_blif.cpp.

Here is the caller graph for this function:

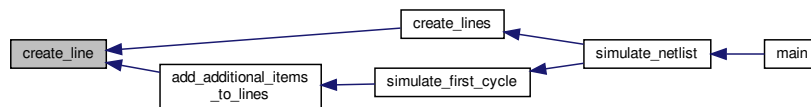


2.56.2.20 create_line()

```
line_t* create_line (  
    char * name )
```

Definition at line 2591 of file simulate_blif.cpp.

Here is the caller graph for this function:

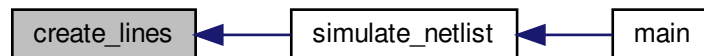


2.56.2.21 create_lines()

```
lines_t* create_lines (  
    netlist_t * netlist,  
    int type )
```

Definition at line 2448 of file simulate_blif.cpp.

Here is the caller graph for this function:

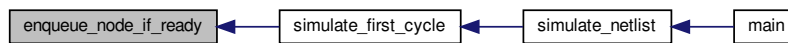


2.56.2.22 enqueue_node_if_ready()

```
int enqueue_node_if_ready (
    queue_t * queue,
    nnode_t * node,
    int cycle )
```

Definition at line 951 of file simulate_blif.cpp.

Here is the caller graph for this function:

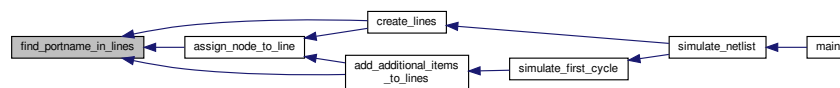


2.56.2.23 find_portname_in_lines()

```
int find_portname_in_lines (
    char * port_name,
    lines_t * l )
```

Definition at line 2578 of file simulate_blif.cpp.

Here is the caller graph for this function:

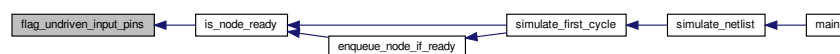


2.56.2.24 flag_undriven_input_pins()

```
void flag_undriven_input_pins (
    nnode_t * node )
```

Definition at line 885 of file simulate_blif.cpp.

Here is the caller graph for this function:

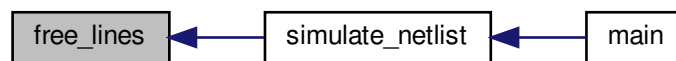


2.56.2.25 free_lines()

```
void free_lines (
    lines_t * l )
```

Definition at line 3373 of file simulate_blif.cpp.

Here is the caller graph for this function:

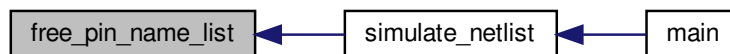


2.56.2.26 free_pin_name_list()

```
void free_pin_name_list (
    pin_names * p )
```

Definition at line 3414 of file simulate_blif.cpp.

Here is the caller graph for this function:

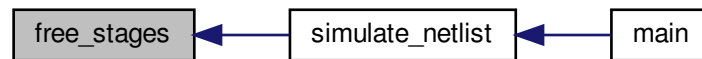


2.56.2.27 free_stages()

```
void free_stages (
    stages_t * s )
```

Definition at line 3390 of file simulate_blif.cpp.

Here is the caller graph for this function:

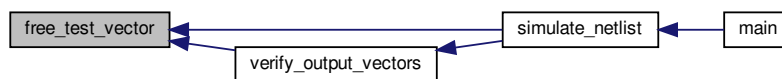


2.56.2.28 free_test_vector()

```
void free_test_vector (
    test_vector * v )
```

Definition at line 3402 of file simulate_blif.cpp.

Here is the caller graph for this function:

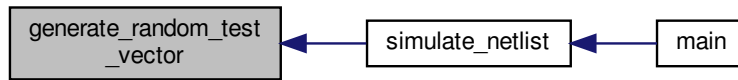


2.56.2.29 generate_random_test_vector()

```
test_vector* generate_random_test_vector (
    lines_t * l,
    int cycle,
    hashtable_t * hold_high_index,
    hashtable_t * hold_low_index )
```

Definition at line 2772 of file simulate_blif.cpp.

Here is the caller graph for this function:

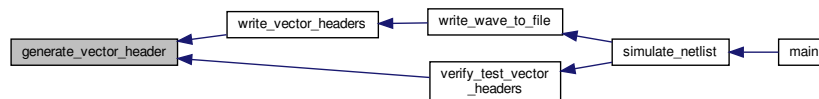


2.56.2.30 generate_vector_header()

```
char* generate_vector_header (
    lines_t * l )
```

Definition at line 2609 of file `simulate_blif.cpp`.

Here is the caller graph for this function:

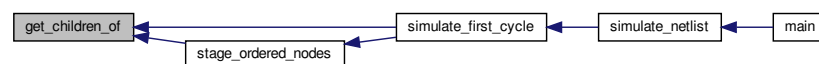


2.56.2.31 get_children_of()

```
nnode_t** get_children_of (
    nnode_t * node,
    int * count )
```

Definition at line 1058 of file `simulate_blif.cpp`.

Here is the caller graph for this function:



2.56.2.32 get_children_of_nodepin()

```
nnode_t** get_children_of_nodepin (
    nnode_t * node,
    int * count,
    int output_pin )
```

Definition at line 1224 of file simulate_blif.cpp.

2.56.2.33 get_children_pinnumber_of()

```
int* get_children_pinnumber_of (
    nnode_t * node,
    int * num_children )
```

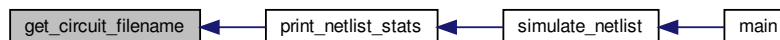
Definition at line 1141 of file simulate_blif.cpp.

2.56.2.34 get_circuit_filename()

```
char* get_circuit_filename ( )
```

Definition at line 3714 of file simulate_blif.cpp.

Here is the caller graph for this function:



2.56.2.35 get_clock_ratio()

```
int get_clock_ratio (
    nnode_t * node )
```

Definition at line 1036 of file simulate_blif.cpp.

2.56.2.36 get_line_pin_value()

```
signed char get_line_pin_value (
    line_t * line,
    int pin_num,
    int cycle )
```

Definition at line 3286 of file simulate_blif.cpp.

Here is the caller graph for this function:

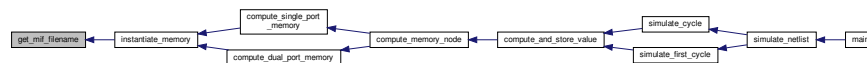


2.56.2.37 get_mif_filename()

```
char* get_mif_filename (
    nnode_t * node )
```

Definition at line 3219 of file simulate_blif.cpp.

Here is the caller graph for this function:

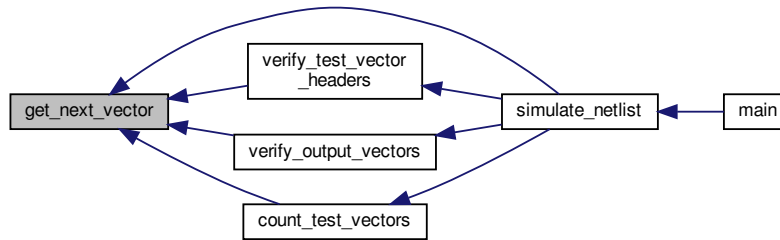


2.56.2.38 get_next_vector()

```
int get_next_vector (
    FILE * file,
    char * buffer )
```

Definition at line 3361 of file simulate_blif.cpp.

Here is the caller graph for this function:

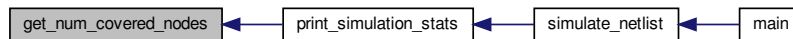


2.56.2.39 get_num_covered_nodes()

```
int get_num_covered_nodes (
    stages_t * s )
```

Definition at line 916 of file `simulate_blif.cpp`.

Here is the caller graph for this function:

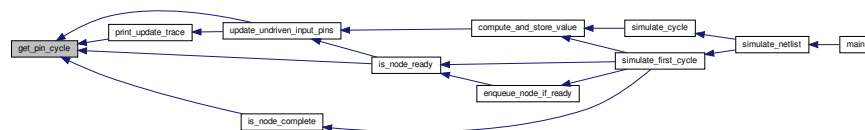


2.56.2.40 get_pin_cycle()

```
int get_pin_cycle (
    npin_t * pin ) [inline]
```

Definition at line 1349 of file `simulate_blif.cpp`.

Here is the caller graph for this function:

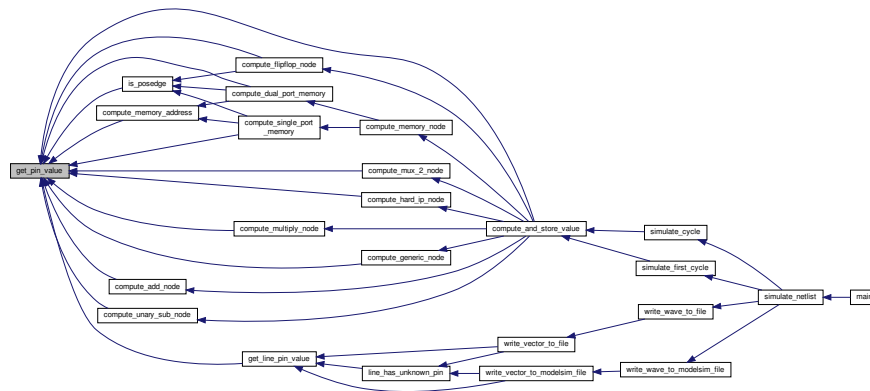


2.56.2.41 get_pin_value()

```
signed char get_pin_value (
    npin_t * pin,
    int cycle )
```

Definition at line 1324 of file simulate_blif.cpp.

Here is the caller graph for this function:

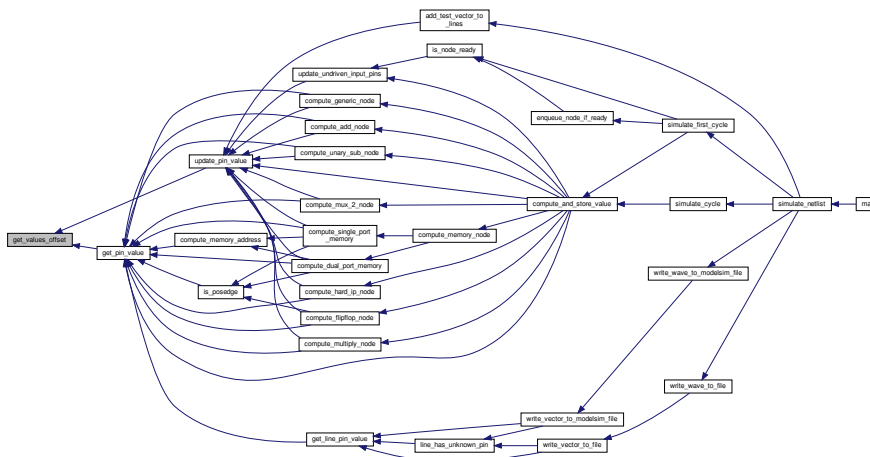


2.56.2.42 get_values_offset()

```
int get_values_offset (
    int cycle ) [inline]
```

Definition at line 1341 of file simulate_blif.cpp.

Here is the caller graph for this function:

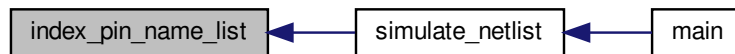


2.56.2.43 index_pin_name_list()

```
hashtable_t* index_pin_name_list (  
    pin_names * list )
```

Definition at line 3094 of file simulate_blif.cpp.

Here is the caller graph for this function:

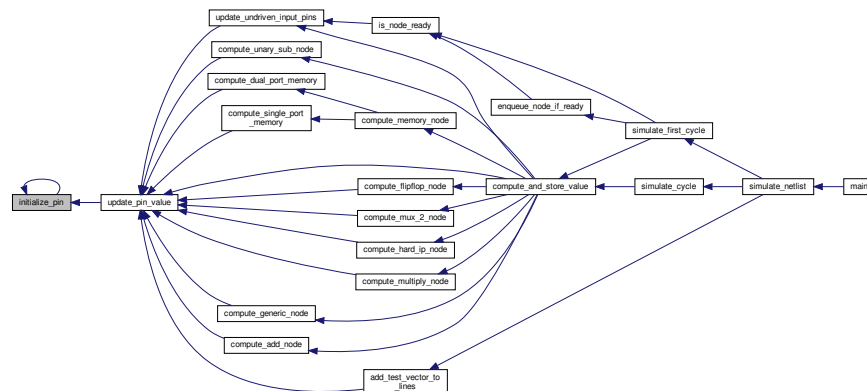


2.56.2.44 initialize_pin()

```
void initialize_pin (  
    npin_t * pin )
```

Definition at line 1385 of file simulate_blif.cpp.

Here is the caller graph for this function:

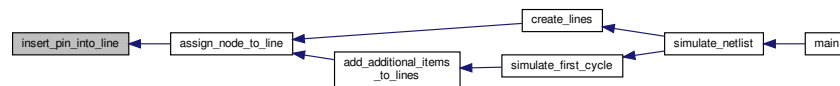


2.56.2.45 insert_pin_into_line()

```
void insert_pin_into_line (
    npin_t * pin,
    int pin_number,
    line_t * line,
    int type )
```

Definition at line 2413 of file simulate_blif.cpp.

Here is the caller graph for this function:

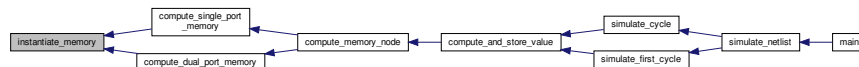


2.56.2.46 instantiate_memory()

```
void instantiate_memory (
    nnode_t * node,
    int data_width,
    int addr_width )
```

Definition at line 2116 of file simulate_blif.cpp.

Here is the caller graph for this function:

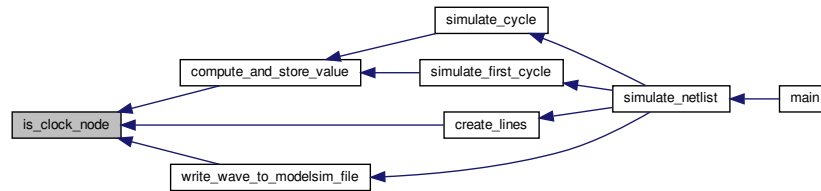


2.56.2.47 is_clock_node()

```
int is_clock_node (
    nnode_t * node ) [inline]
```

Definition at line 1434 of file simulate_blif.cpp.

Here is the caller graph for this function:

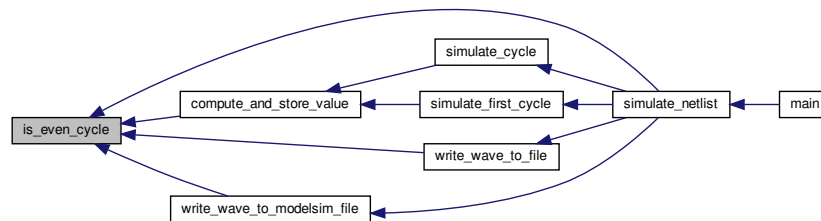


2.56.2.48 is_even_cycle()

```
int is_even_cycle (
    int cycle )
```

Definition at line 1370 of file simulate_blif.cpp.

Here is the caller graph for this function:

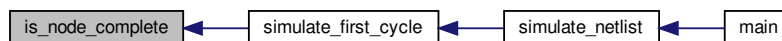


2.56.2.49 is_node_complete()

```
int is_node_complete (
    nnode_t * node,
    int cycle )
```

Definition at line 968 of file simulate_blif.cpp.

Here is the caller graph for this function:

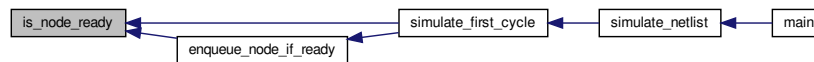


2.56.2.50 is_node_ready()

```
int is_node_ready (
    nnode_t * node,
    int cycle )
```

Definition at line 981 of file simulate_blif.cpp.

Here is the caller graph for this function:

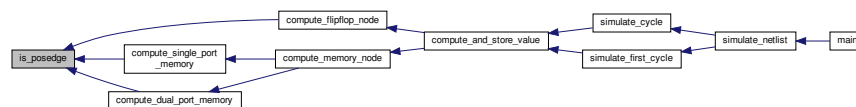


2.56.2.51 is_posedge()

```
int is_posedge (
    npin_t * pin,
    int cycle )
```

Definition at line 1447 of file simulate_blif.cpp.

Here is the caller graph for this function:

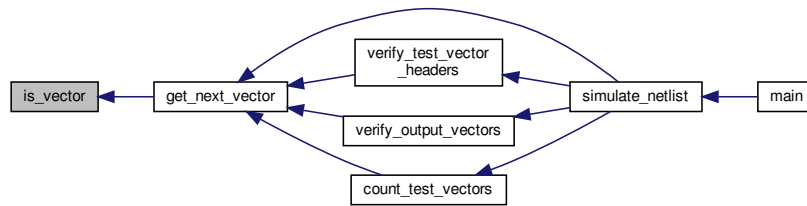


2.56.2.52 is_vector()

```
int is_vector (
    char * buffer )
```

Definition at line 3338 of file simulate_blif.cpp.

Here is the caller graph for this function:



2.56.2.53 line_has_unknown_pin()

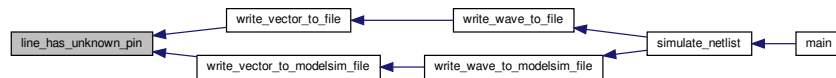
```

int line_has_unknown_pin (
    line_t * line,
    int cycle )

```

Definition at line 3267 of file `simulate_blif.cpp`.

Here is the caller graph for this function:



2.56.2.54 multiply_arrays()

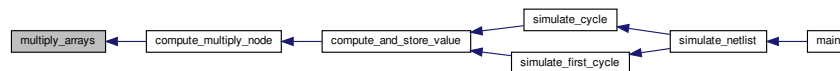
```

int* multiply_arrays (
    int * a,
    int a_length,
    int * b,
    int b_length )

```

Definition at line 1703 of file `simulate_blif.cpp`.

Here is the caller graph for this function:



2.56.2.55 parse_mif_radix()

```
int parse_mif_radix (
    char * radix )
```

Definition at line 2183 of file simulate_blif.cpp.

Here is the caller graph for this function:

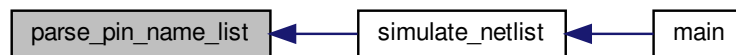


2.56.2.56 parse_pin_name_list()

```
pin_names* parse_pin_name_list (
    char * list )
```

Definition at line 3112 of file simulate_blif.cpp.

Here is the caller graph for this function:

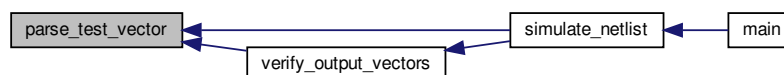


2.56.2.57 parse_test_vector()

```
test_vector* parse_test_vector (
    char * buffer )
```

Definition at line 2703 of file simulate_blif.cpp.

Here is the caller graph for this function:

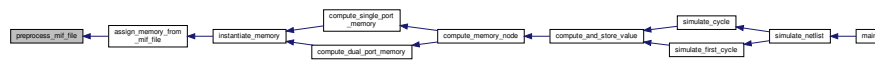


2.56.2.58 preprocess_mif_file()

```
FILE* preprocess_mif_file (
    FILE * source )
```

Definition at line 2145 of file simulate_blif.cpp.

Here is the caller graph for this function:

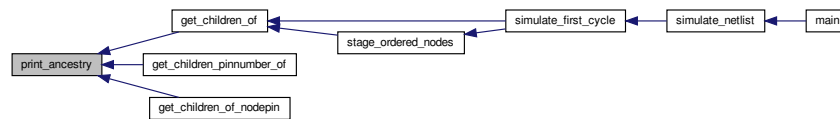


2.56.2.59 print_ancestry()

```
void print_ancestry (
    nnode_t * node,
    int generations )
```

Definition at line 3520 of file simulate_blif.cpp.

Here is the caller graph for this function:

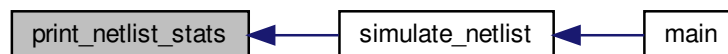


2.56.2.60 print_netlist_stats()

```
void print_netlist_stats (
    stages_t * stages,
    int num_vectors )
```

Definition at line 3473 of file simulate_blif.cpp.

Here is the caller graph for this function:

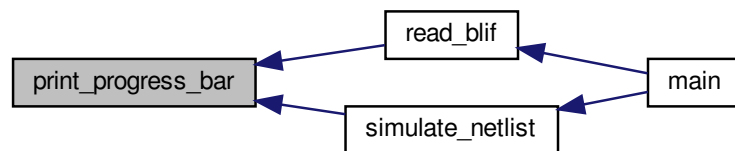


2.56.2.61 print_progress_bar()

```
int print_progress_bar (  
    double completion,  
    int position,  
    int length,  
    double time )
```

Definition at line 3431 of file simulate_blif.cpp.

Here is the caller graph for this function:

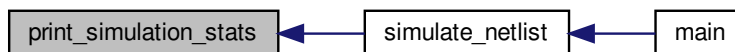


2.56.2.62 print_simulation_stats()

```
void print_simulation_stats (  
    stages_t * stages,  
    int num_vectors,  
    double total_time,  
    double simulation_time )
```

Definition at line 3491 of file simulate_blif.cpp.

Here is the caller graph for this function:

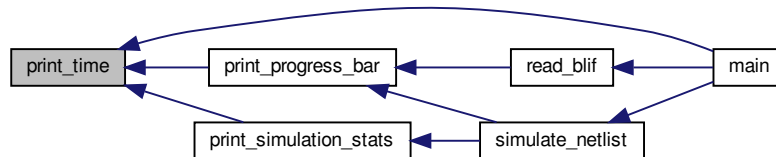


2.56.2.63 print_time()

```
void print_time (  
    double time )
```

Definition at line 3507 of file simulate_blif.cpp.

Here is the caller graph for this function:

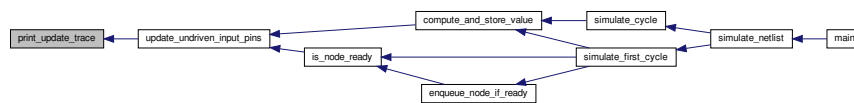


2.56.2.64 print_update_trace()

```
nnode_t* print_update_trace (  
    nnode_t * bottom_node,  
    int cycle )
```

Definition at line 3611 of file simulate_blif.cpp.

Here is the caller graph for this function:



2.56.2.65 set_clock_ratio()

```
void set_clock_ratio (  
    int rat,  
    nnode_t * node )
```

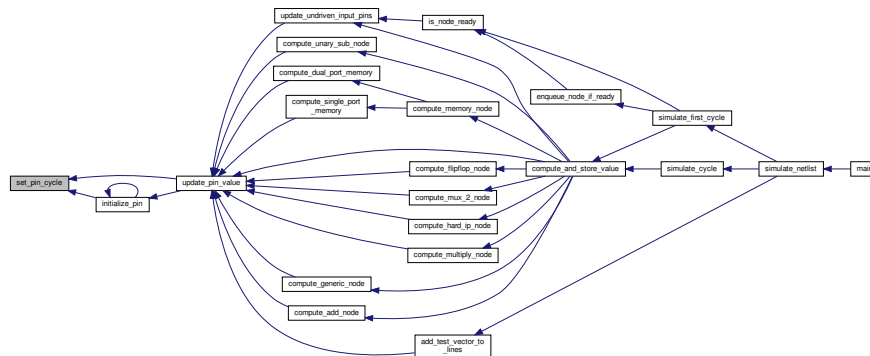
Definition at line 1044 of file simulate_blif.cpp.

2.56.2.66 set_pin_cycle()

```
void set_pin_cycle (
    npin_t * pin,
    int cycle ) [inline]
```

Definition at line 1362 of file simulate_blif.cpp.

Here is the caller graph for this function:

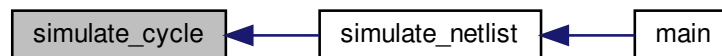


2.56.2.67 simulate_cycle()

```
void simulate_cycle (
    int cycle,
    stages_t * s )
```

Definition at line 275 of file simulate_blif.cpp.

Here is the caller graph for this function:

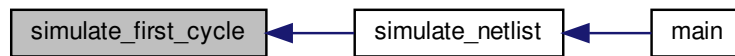


2.56.2.68 simulate_first_cycle()

```
stages_t* simulate_first_cycle (  
    netlist_t * netlist,  
    int cycle,  
    pin_names * p,  
    lines_t * output_lines )
```

Definition at line 361 of file `simulate_blif.cpp`.

Here is the caller graph for this function:



2.56.2.69 simulate_netlist()

```
void simulate_netlist (  
    netlist_t * netlist )
```

Definition at line 43 of file `simulate_blif.cpp`.

Here is the caller graph for this function:

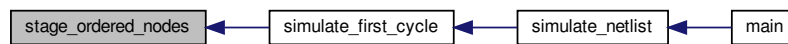


2.56.2.70 stage_ordered_nodes()

```
stages_t* stage_ordered_nodes (
    nnode_t ** ordered_nodes,
    int num_ordered_nodes )
```

Definition at line 420 of file simulate_blif.cpp.

Here is the caller graph for this function:

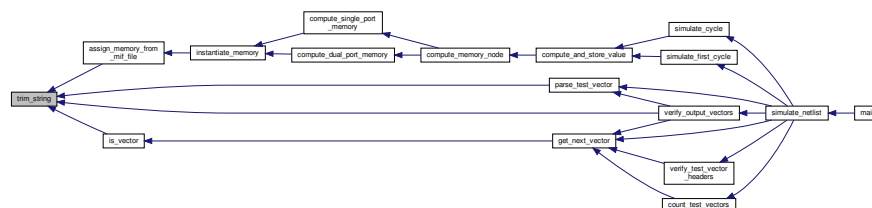


2.56.2.71 trim_string()

```
void trim_string (
    char * string,
    const char * chars )
```

Definition at line 3239 of file simulate_blif.cpp.

Here is the caller graph for this function:

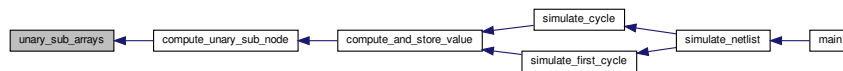


2.56.2.72 unary_sub_arrays()

```
int* unary_sub_arrays (
    int * a,
    int a_length,
    int * c,
    int c_length )
```

Definition at line 1931 of file simulate_blif.cpp.

Here is the caller graph for this function:

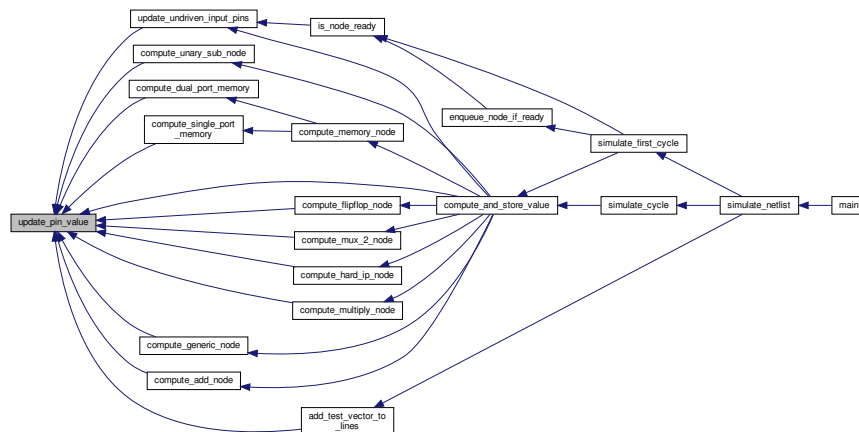


2.56.2.73 update_pin_value()

```
void update_pin_value (
    npin_t * pin,
    signed char value,
    int cycle )
```

Definition at line 1312 of file simulate_blif.cpp.

Here is the caller graph for this function:

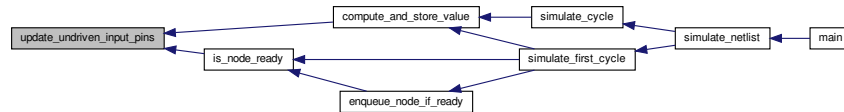


2.56.2.74 update_undriven_input_pins()

```
void update_undriven_input_pins (
    nnode_t * node,
    int cycle )
```

Definition at line 842 of file simulate_blif.cpp.

Here is the caller graph for this function:



2.56.2.75 vector_value_to_hex()

```
char* vector_value_to_hex (
    signed char * value,
    int length )
```

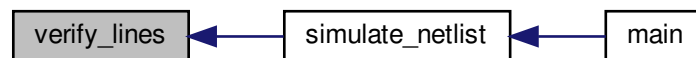
Definition at line 3295 of file simulate_blif.cpp.

2.56.2.76 verify_lines()

```
int verify_lines (
    lines_t * l )
```

Definition at line 2556 of file simulate_blif.cpp.

Here is the caller graph for this function:

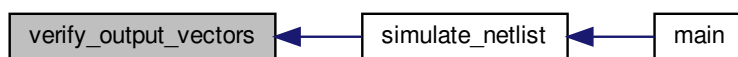


2.56.2.77 verify_output_vectors()

```
int verify_output_vectors (
    char * output_vector_file,
    int num_test_vectors )
```

Definition at line 2997 of file simulate_blif.cpp.

Here is the caller graph for this function:

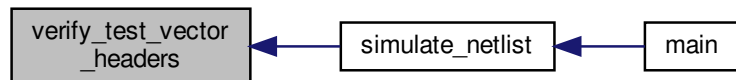


2.56.2.78 verify_test_vector_headers()

```
int verify_test_vector_headers (
    FILE * in,
    lines_t * l )
```

Definition at line 2496 of file simulate_blif.cpp.

Here is the caller graph for this function:

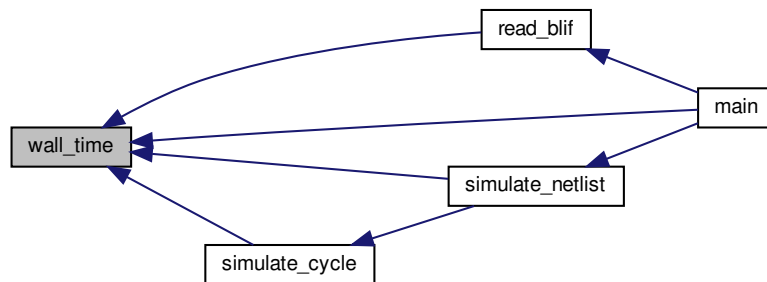


2.56.2.79 wall_time()

```
double wall_time ( )
```

Definition at line 3701 of file simulate_blif.cpp.

Here is the caller graph for this function:

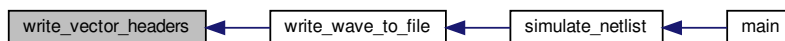


2.56.2.80 write_vector_headers()

```
void write_vector_headers (
    FILE * file,
    lines_t * l )
```

Definition at line 2482 of file simulate_blif.cpp.

Here is the caller graph for this function:

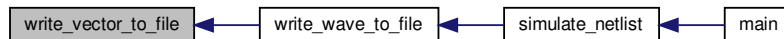


2.56.2.81 write_vector_to_file()

```
void write_vector_to_file (
    lines_t * l,
    FILE * file,
    int cycle )
```

Definition at line 2843 of file simulate_blif.cpp.

Here is the caller graph for this function:

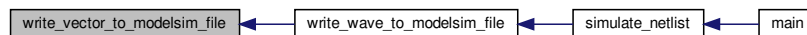


2.56.2.82 write_vector_to_modelsim_file()

```
void write_vector_to_modelsim_file (
    lines_t * l,
    FILE * modelsim_out,
    int cycle )
```

Definition at line 2945 of file simulate_blif.cpp.

Here is the caller graph for this function:

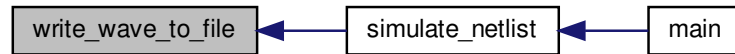


2.56.2.83 write_wave_to_file()

```
void write_wave_to_file (
    lines_t * l,
    FILE * file,
    int cycle_offset,
    int wave_length,
    int both_edges )
```

Definition at line 2826 of file simulate_blif.cpp.

Here is the caller graph for this function:



2.56.2.84 `write_wave_to_modelsim_file()`

```
void write_wave_to_modelsim_file (
    netlist_t * netlist,
    lines_t * l,
    FILE * modelsim_out,
    int cycle_offset,
    int wave_length )
```

Definition at line 2914 of file `simulate_blif.cpp`.

Here is the caller graph for this function:



2.57 `vtr-verilog-to-routing/ODIN_II/SRC/STRING_HASH_TOOL/hashtable.cpp` File Reference

```
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include "hashtable.h"
#include "types.h"
#include "vtr_memory.h"
```

Functions

- void [__hashtable_add](#) (hashtable_t *h, const void *key, size_t key_length, void *item)
- void * [__hashtable_remove](#) (hashtable_t *h, const void *key, size_t key_length)
- void * [__hashtable_get](#) (hashtable_t *h, const void *key, size_t key_length)
- void ** [__hashtable_get_all](#) (hashtable_t *h)
- int [__hashtable_is_empty](#) (hashtable_t *h)
- void [__hashtable_destroy](#) (hashtable_t *h)
- void [__hashtable_destroy_free_items](#) (hashtable_t *h)
- int [__hashtable_compare_keys](#) (const void *key, size_t key_len, const void *key1, size_t key_len1)
- unsigned int [__hashtable_hash](#) (const void *key, size_t key_len, int max_key)
- hashtable_t * [create_hashtable](#) (int store_size)

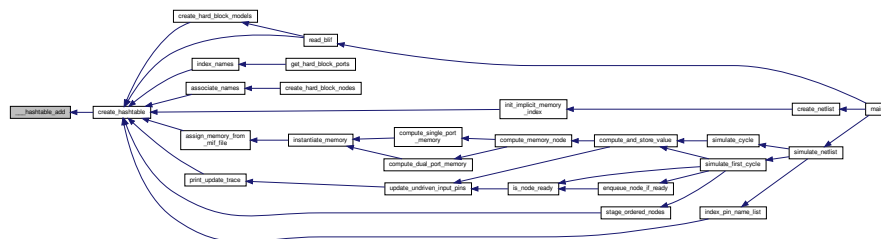
2.57.1 Function Documentation

2.57.1.1 [__hashtable_add\(\)](#)

```
void __hashtable_add (
    hashtable_t * h,
    const void * key,
    size_t key_length,
    void * item )
```

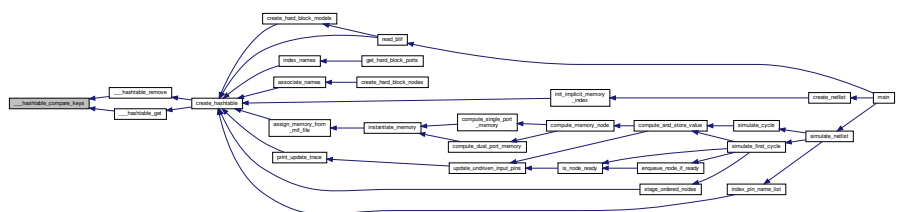
Definition at line 102 of file hashtable.cpp.

Here is the caller graph for this function:



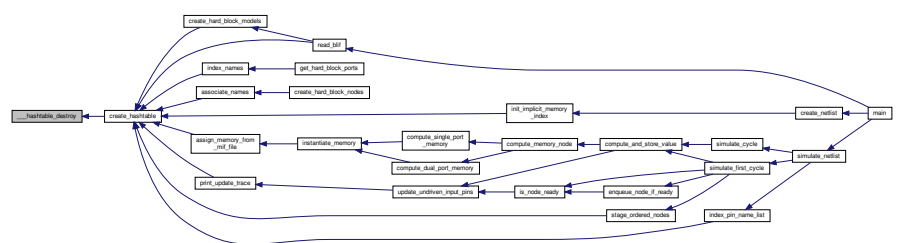
```
int __hashtable_compare_keys (
    const void * key,
    size_t key_len,
    const void * key1,
    size_t key_len1 )
```

Here is the caller graph for this function:



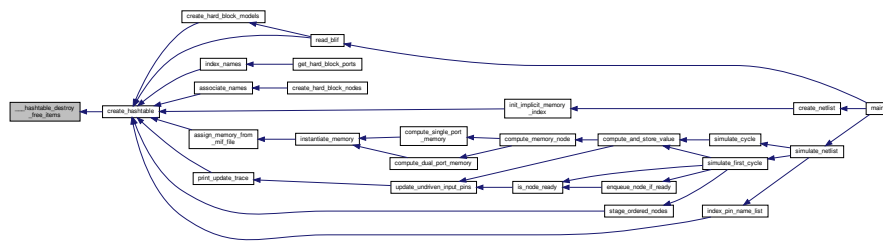
```
void __hashtable_destroy (
    hashtable_t * h )
```

Here is the caller graph for this function:



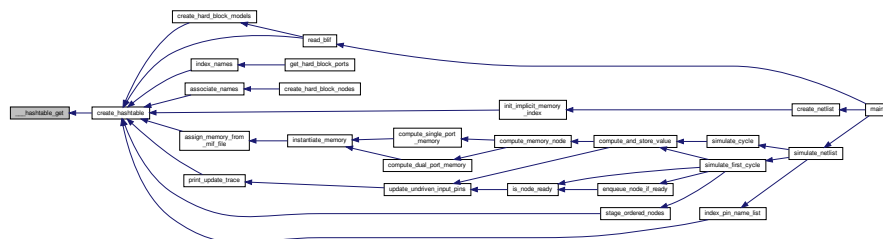
```
void __hashtable_destroy_free_items (
    hashtable_t * h )
```

Here is the caller graph for this function:



```
void * __hashtable_get (
    hashtable_t * h,
    const void * key,
    size_t key_length )
```

Here is the caller graph for this function:

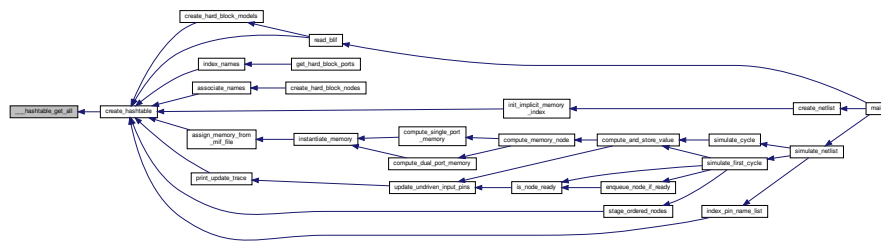


2.57.1.6 ____hashtable_get_all()

```
void ** ____hashtable_get_all (
    hashtable_t * h )
```

Definition at line 164 of file hashtable.cpp.

Here is the caller graph for this function:

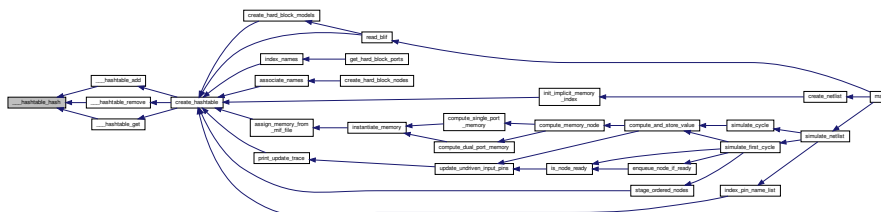


2.57.1.7 ____hashtable_hash()

```
unsigned int ____hashtable_hash (
    const void * key,
    size_t key_len,
    int max_key )
```

Definition at line 192 of file hashtable.cpp.

Here is the caller graph for this function:

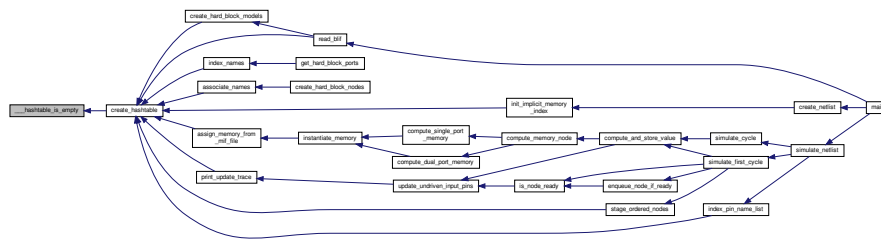


2.57.1.8 ____hashtable_is_empty()

```
int ____hashtable_is_empty (
    hashtable_t * h )
```

Definition at line 181 of file hashtable.cpp.

Here is the caller graph for this function:

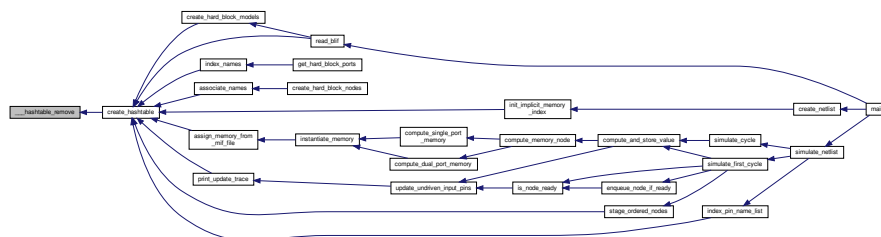


2.57.1.9 ____hashtable_remove()

```
void * ____hashtable_remove (
    hashtable_t * h,
    const void * key,
    size_t key_length )
```

Definition at line 124 of file hashtable.cpp.

Here is the caller graph for this function:

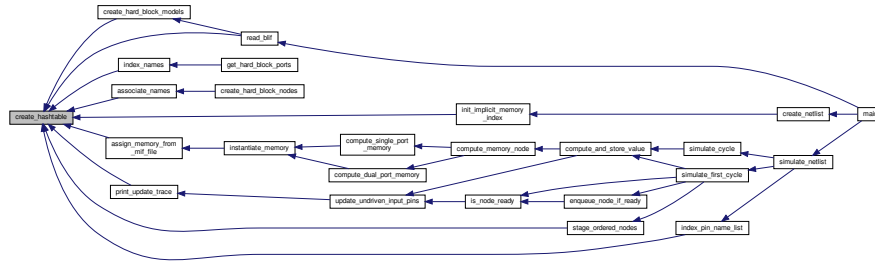


2.57.1.10 create_hashtable()

```
hashtable_t* create_hashtable (
    int store_size )
```

Definition at line 40 of file hashtable.cpp.

Here is the caller graph for this function:



2.58 vtr-verilog-to-routing/ODIN_II/SRC/STRING_HASH_TOOL/hashtable.h File Reference

```
#include <sys/types.h>
#include <stdint.h>
```

Data Structures

- struct [hashtable_t_t](#)
- struct [hashtable_node_t_t](#)

Macros

- `#define hashtable_t struct hashtable_t_t`
- `#define hashtable_node_t struct hashtable_node_t_t`

Functions

- `hashtable_t * create_hashtable (int store_size)`

2.58.1 Macro Definition Documentation

2.58.1.1 hashtable_node_t

```
#define hashtable_node_t struct hashtable_node_t_t
```

Definition at line 26 of file hashtable.h.

2.58.1.2 hashtable_t

```
#define hashtable_t struct hashtable_t_t
```

Definition at line 25 of file hashtable.h.

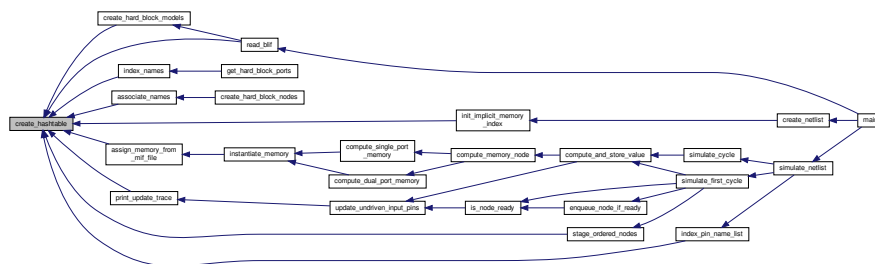
2.58.2 Function Documentation

2.58.2.1 create_hashtable()

```
hashtable_t* create_hashtable (
    int store_size )
```

Definition at line 40 of file hashtable.cpp.

Here is the caller graph for this function:

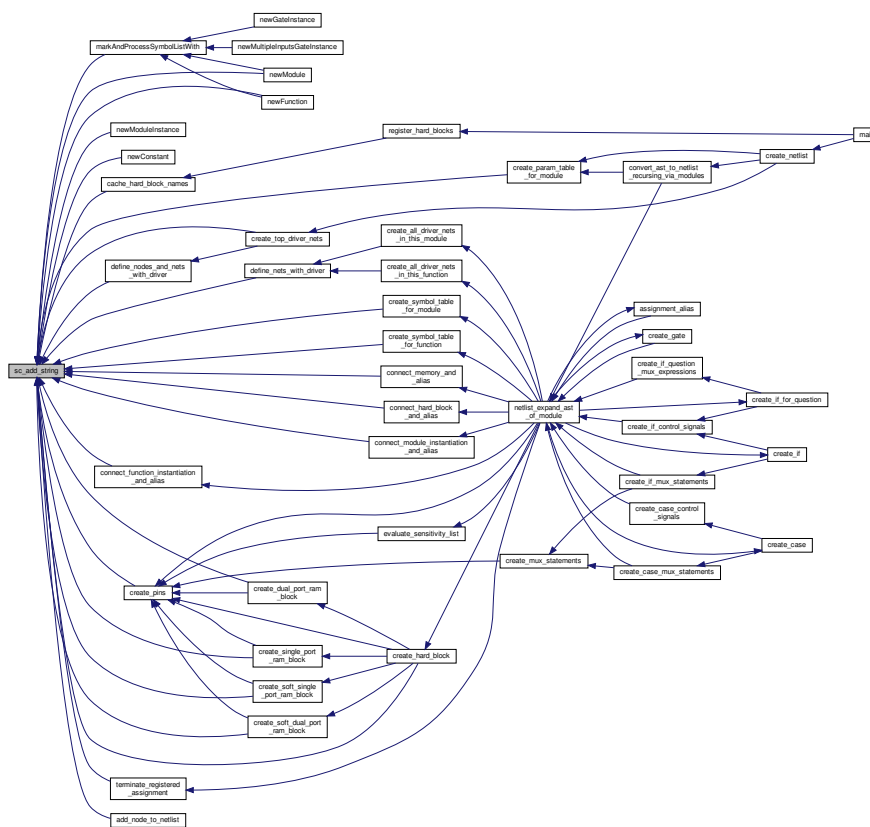


2.59 vtr-verilog-to-routing/ODIN_II/SRC/STRING_HASH_TOOL/string_cache.cpp File Reference

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "vtr_util.h"
#include "vtr_memory.h"
#include "string_cache.h"
```


2.59.1.2 sc_add_string()

Definition at line 95 of file string_cache.cpp.



2.59.1.3 sc_do_alloc()

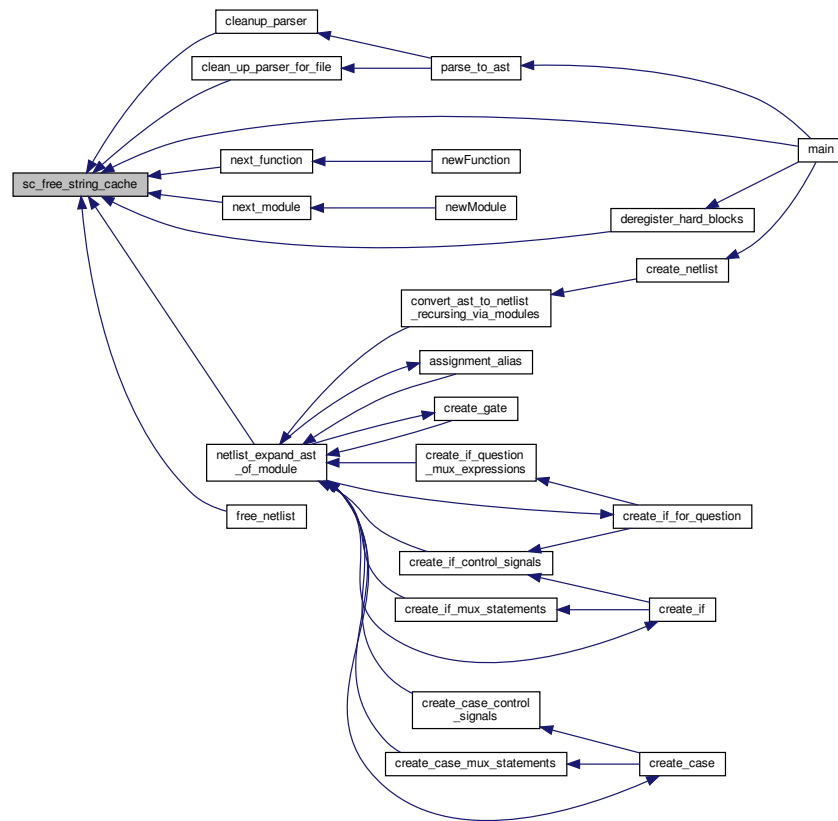
Definition at line 145 of file string_cache.cpp.

[illegible]

```
STRING_CACHE* sc_free_string_cache (
    STRING_CACHE * sc )
```

Generated on Thu Aug 10 2017 19:13:39 for Odin by Doxygen

Here is the caller graph for this function:



2.59.1.5 sc_lookup_string()

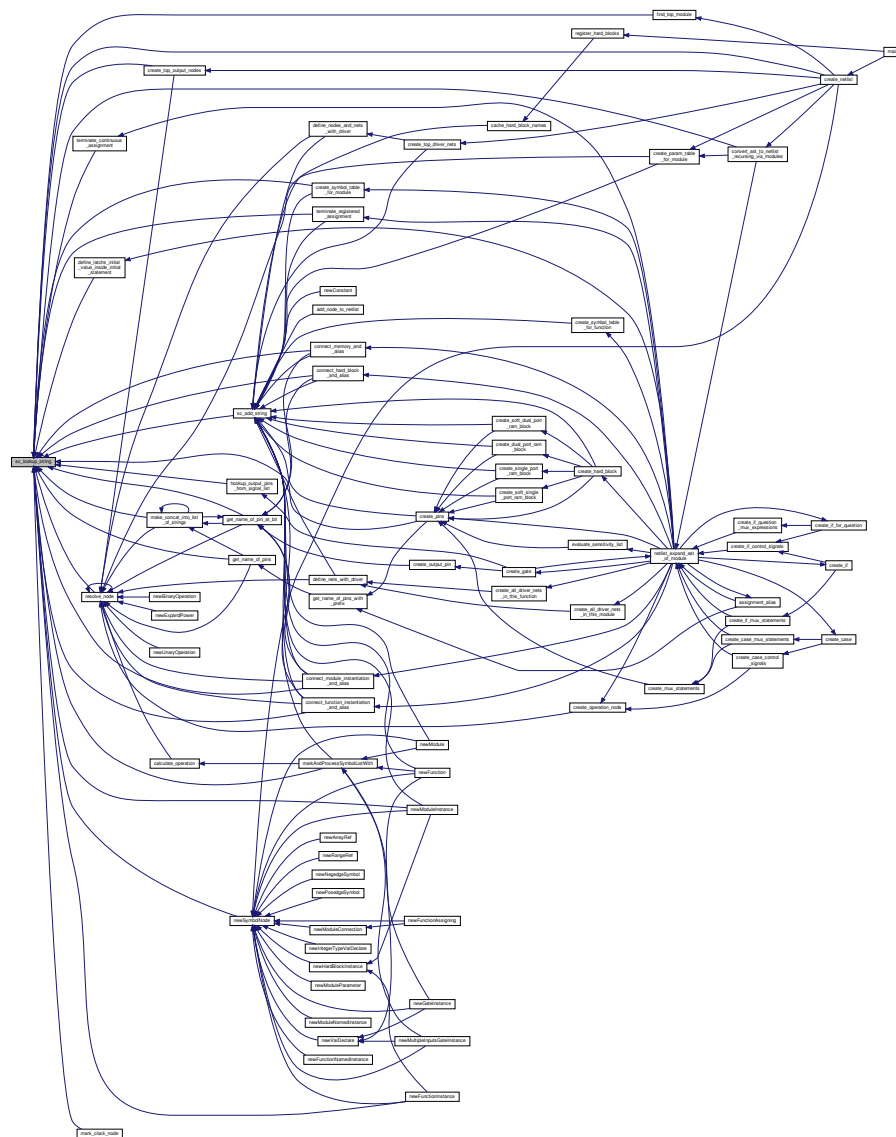
```

long sc_lookup_string (
    STRING_CACHE * sc,
    const char * string )

```

Definition at line 73 of file string_cache.cpp.

Here is the caller graph for this function:

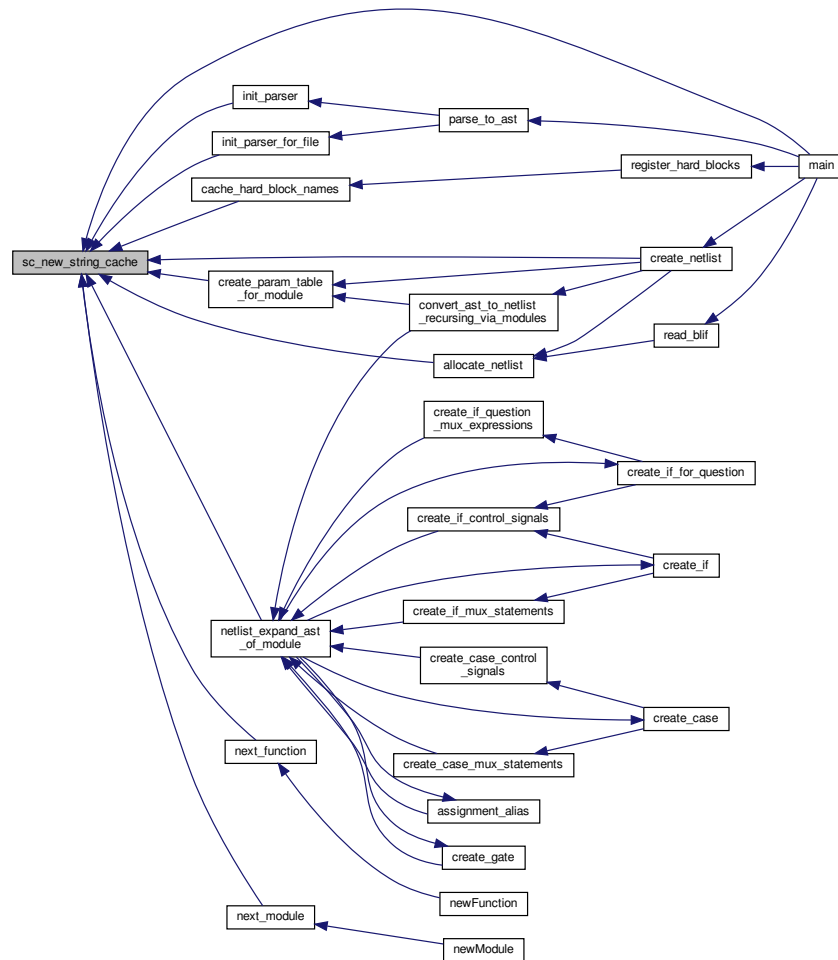


2.59.1.6 sc_new_string_cache()

```
STRING_CACHE* sc_new_string_cache (
    void )
```

Definition at line 54 of file string_cache.cpp.

Here is the caller graph for this function:



2.59.1.7 sc_valid_id()

```

int sc_valid_id (
    STRING_CACHE * sc,
    long string_id )

```

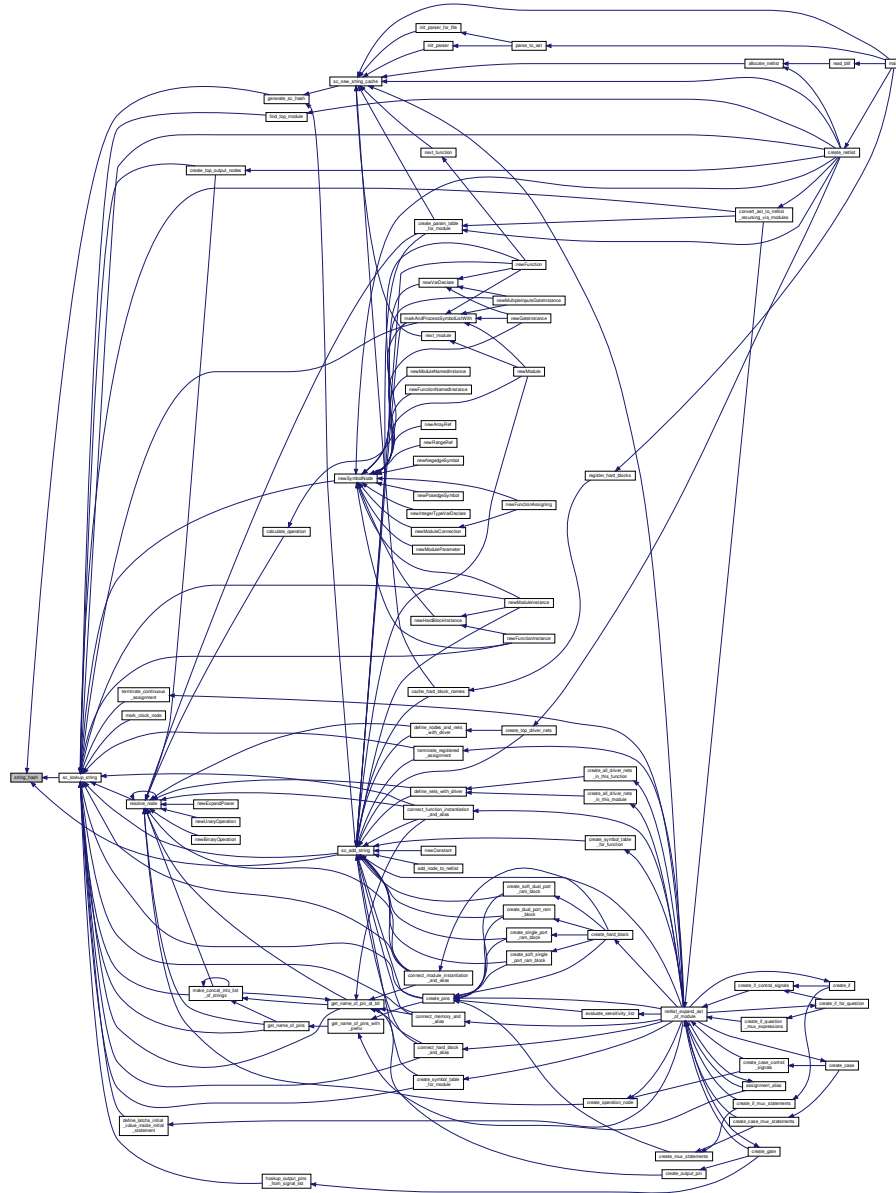
Definition at line 134 of file `string_cache.cpp`.

2.59.1.8 string_hash()

```
unsigned long string_hash (
    STRING_CACHE * sc,
    const char * string )
```

Definition at line 16 of file string_cache.cpp.

Here is the caller graph for this function:



2.60 vtr-verilog-to-routing/ODIN_II/SRC/STRING_HASH_TOOL/string_cache.h File Reference

Data Structures

- struct [STRING_CACHE](#)

Functions

- [STRING_CACHE * sc_new_string_cache](#) (void)
- [long sc_lookup_string](#) (STRING_CACHE *sc, const char *string)
- [long sc_add_string](#) (STRING_CACHE *sc, const char *string)
- [int sc_valid_id](#) (STRING_CACHE *sc, long string_id)
- [void * sc_do_alloc](#) (long, long)
- [STRING_CACHE * sc_free_string_cache](#) (STRING_CACHE *sc)

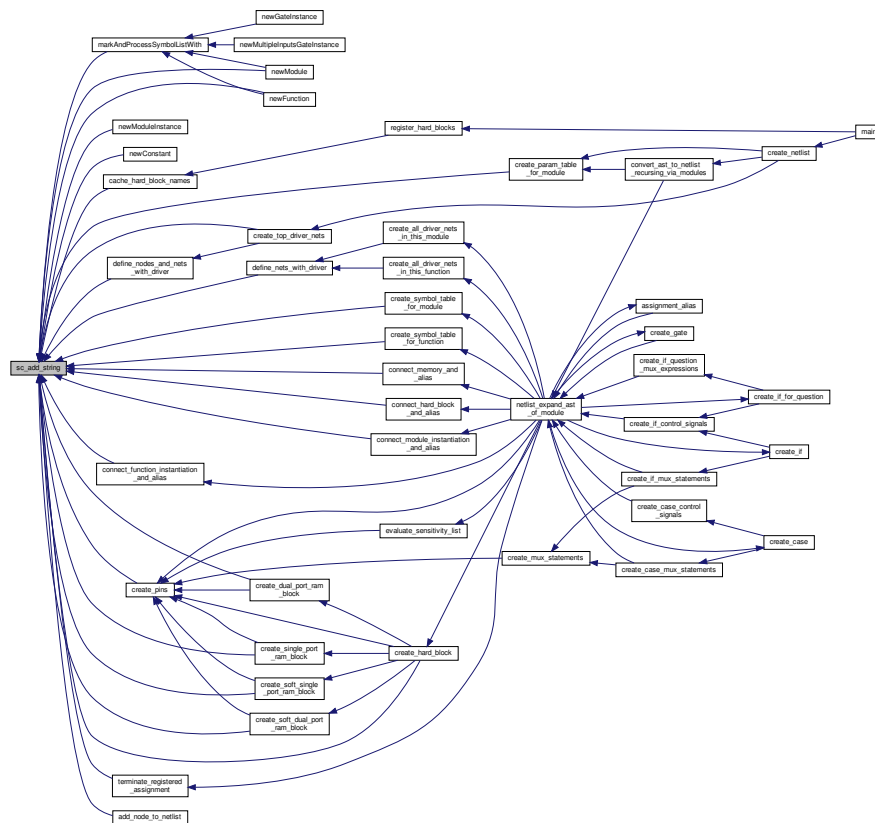
2.60.1 Function Documentation

2.60.1.1 sc_add_string()

```
long sc_add_string (
    STRING_CACHE * sc,
    const char * string )
```

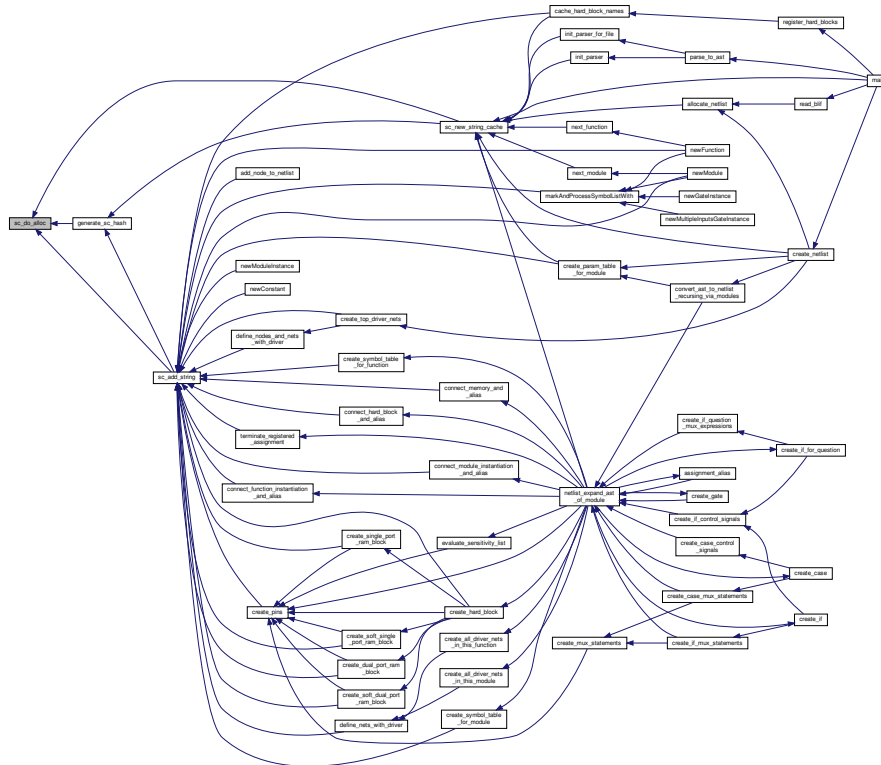
Definition at line 95 of file string_cache.cpp.

Here is the caller graph for this function:




```
void* sc_do_alloc (
                long ,
                long )
```

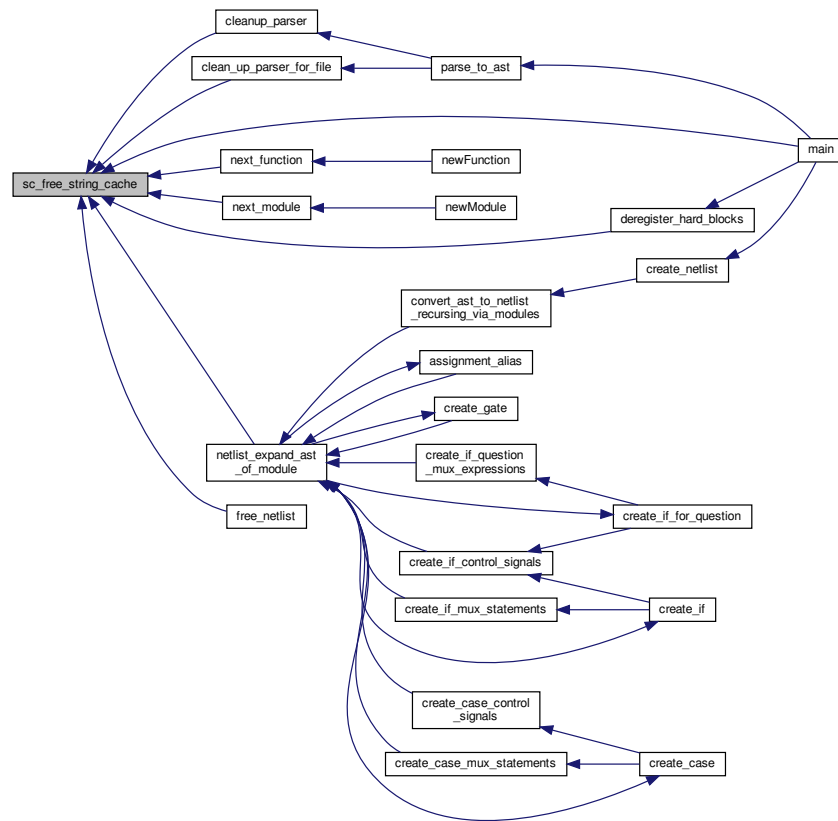
Here is the caller graph for this function:



```
STRING_CACHE* sc_free_string_cache (
    STRING_CACHE * sc )
```

Generated on Thu Aug 10 2017 19:13:39 for Odin by Doxygen

Here is the caller graph for this function:



2.60.1.4 sc_lookup_string()

```

long sc_lookup_string (
    STRING_CACHE * sc,
    const char * string )

```

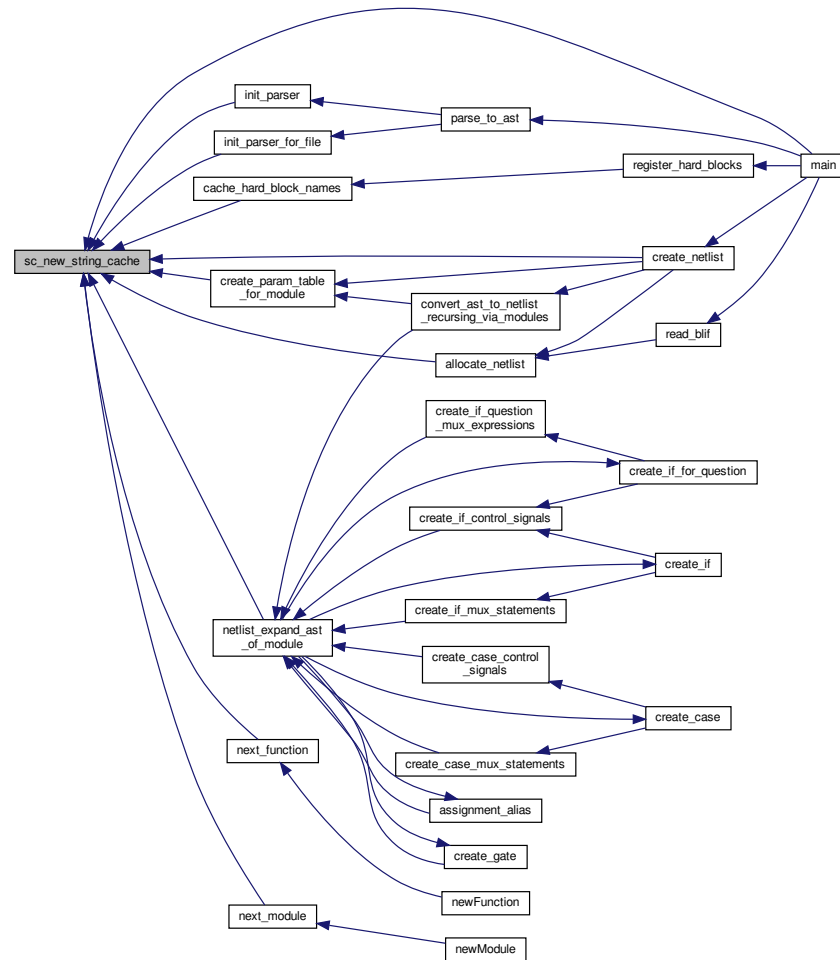
Definition at line 73 of file string_cache.cpp.

[illegible]

```
STRING_CACHE* sc_new_string_cache (
    void )
```

Generated on Thu Aug 10 2017 19:13:39 for Odin by Doxygen

Here is the caller graph for this function:



2.60.1.6 sc_valid_id()

```

int sc_valid_id (
    STRING_CACHE * sc,
    long string_id )

```

Definition at line 134 of file string_cache.cpp.

2.61 vtr-verilog-to-routing/ODIN_II/SRC/types.h File Reference

```

#include "string_cache.h"
#include "odin_util.h"

```

```
#include "read_xml_arch_file.h"  
#include "simulate_blif.h"  
#include "argparse_value.hpp"  
#include <stdlib.h>
```

Index

- ___hashtable_add
 - hashtable.cpp, 609
- ___hashtable_compare_keys
 - hashtable.cpp, 609
- ___hashtable_destroy
 - hashtable.cpp, 610
- ___hashtable_destroy_free_items
 - hashtable.cpp, 610
- ___hashtable_get
 - hashtable.cpp, 611
- ___hashtable_get_all
 - hashtable.cpp, 611
- ___hashtable_hash
 - hashtable.cpp, 612
- ___hashtable_is_empty
 - hashtable.cpp, 612
- ___hashtable_remove
 - hashtable.cpp, 613
- ___queue_add
 - queue.cpp, 528
- ___queue_destroy
 - queue.cpp, 529
- ___queue_is_empty
 - queue.cpp, 529
- ___queue_remove
 - queue.cpp, 530
- ___queue_remove_all
 - queue.cpp, 530
- _visited_backward
 - netlist_cleanup.cpp, 315
- _visited_forward
 - netlist_cleanup.cpp, 315
- _visited_removal
 - netlist_cleanup.cpp, 315
- a
 - adder_signals, 3
- ACE_P0TO1
 - ace.cpp, 517
- ACE_P1TO0
 - ace.cpp, 517
- ADD_string
 - node_creation_library.cpp, 433
- ADDER_FUNC_string
 - node_creation_library.cpp, 433
- ALIAS_INPUTS
 - netlist_create_from_ast.cpp, 321
- ALLOC
 - ace.cpp, 517
- ace.cpp
 - ACE_P0TO1, 517
- ACE_P1TO0, 517
- ALLOC, 517
- ace_bdd_count_paths, 521
- ace_cube_dup, 521
- ace_cube_new_dc, 521
- alloc_and_init_ace_info_object, 521
- alloc_and_init_ace_structs, 522
- BPI, 517
- build_bdd_for_node, 522
- calc_cube_switch_prob, 522
- calc_cube_switch_prob_recur, 523
- calc_node_depth, 523
- calc_switch_prob_recur, 524
- calculate_activity, 524
- compute_switching_activities, 525
- DASH, 517
- fail, 517
- GETINPUT, 518
- LOGBPI, 518
- LOOPINIT, 518
- MAX, 518
- node_error, 525
- node_get_literal, 518
- ONE, 519
- output_ace_info, 525
- output_ace_info_node, 526
- prob_epsilon_fix, 526
- pset, 520
- set_clear, 526
- set_insert, 519
- set_new, 519
- set_remove, 519
- set_size, 519
- TWO, 520
- WHICH_BIT, 520
- WHICH_WORD, 520
- ZERO, 520
- ace.h
 - alloc_and_init_ace_structs, 527
 - calculate_activity, 527
- ace_bdd_count_paths
 - ace.cpp, 521
- ace_cube_dup
 - ace.cpp, 521
- ace_cube_new_dc
 - ace.cpp, 521
- ace_cube_t, 2
 - cube, 2
 - num_literals, 2
 - static_prob, 2
- add

- hashtable_t_t, 14
- queue_t_t, 24
- add_additional_items_to_lines
 - simulate_blif.cpp, 536
 - simulate_blif.h, 574
- add_arrays
 - simulate_blif.cpp, 536
 - simulate_blif.h, 574
- add_child_at_the_beginning_of_the_node
 - ast_util.cpp, 101
 - ast_util.h, 117
- add_child_to_node
 - ast_util.cpp, 101
 - ast_util.h, 117
- add_driver_pin_to_net
 - netlist_utils.cpp, 359
 - netlist_utils.h, 391
- add_dummy_input_port_to_implicit_memory
 - implicit_memory.cpp, 230
- add_dummy_output_port_to_implicit_memory
 - implicit_memory.cpp, 231
- add_fanout_pin_to_net
 - netlist_utils.cpp, 360
 - netlist_utils.h, 392
- add_hard_block_model
 - read_blif.cpp, 184
- add_input_pin_to_node
 - netlist_utils.cpp, 361
 - netlist_utils.h, 393
- add_input_port_information
 - netlist_utils.cpp, 363
 - netlist_utils.h, 395
- add_input_port_to_implicit_memory
 - implicit_memory.cpp, 231
 - implicit_memory.h, 238
- add_input_port_to_memory
 - memories.cpp, 245
 - memories.h, 265
- add_list
 - adders.cpp, 205
 - adders.h, 217
- add_node_to_netlist
 - netlist_utils.cpp, 363
 - netlist_utils.h, 395
- add_output_pin_to_node
 - netlist_utils.cpp, 363
 - netlist_utils.h, 395
- add_output_port_information
 - netlist_utils.cpp, 364
 - netlist_utils.h, 396
- add_output_port_to_implicit_memory
 - implicit_memory.cpp, 232
 - implicit_memory.h, 238
- add_output_port_to_memory
 - memories.cpp, 245
 - memories.h, 265
- add_pin_to_signal_list
 - netlist_utils.cpp, 365
 - netlist_utils.h, 397
- add_tag_data
 - ast_util.cpp, 101
 - ast_util.h, 118
- add_test_vector_to_lines
 - simulate_blif.cpp, 536
 - simulate_blif.h, 575
- add_the_blackbox_for_adds
 - adders.cpp, 196
 - adders.h, 209
- add_the_blackbox_for_mults
 - multipliers.cpp, 280
 - multipliers.h, 289
- add_top_input_nodes
 - read_blif.cpp, 184
- add_veri_define
 - verilog_preprocessor.cpp, 501
 - verilog_preprocessor.h, 510
- add_veri_include
 - verilog_preprocessor.cpp, 502
 - verilog_preprocessor.h, 510
- adder
 - adders.cpp, 205
- adder_chain_count
 - adders.cpp, 206
 - netlist_cleanup.cpp, 315
- adder_signals, 3
 - a, 3
 - b, 3
 - cin, 3
 - cout, 3
 - sumout, 4
- adders.cpp
 - add_list, 205
 - add_the_blackbox_for_adds, 196
 - adder, 205
 - adder_chain_count, 206
 - chain_list, 206
 - clean_adders, 197
 - declare_hard_adder, 197
 - define_add_function, 197
 - find_hard_adders, 198
 - free_op_nodes, 198
 - geomean_addsub_length, 206
 - hard_adders, 206
 - init_add_distribution, 199
 - init_split_adder, 199
 - instantiate_hard_adder, 200
 - iterate_adders, 200
 - longest_adder_chain, 206

- match_node, [200](#)
- match_pins, [201](#)
- match_ports, [201](#)
- merge_nodes, [201](#)
- min_add, [206](#)
- min_threshold_adder, [207](#)
- processed_adder_list, [207](#)
- reallocate_pins, [202](#)
- record_add_distribution, [202](#)
- reduce_operations, [202](#)
- remove_fanout_pins, [203](#)
- remove_list_node, [203](#)
- report_add_distribution, [203](#)
- split_adder, [204](#)
- sum_of_addsub_logs, [207](#)
- the_netlist, [207](#)
- total, [207](#)
- total_adders, [207](#)
- total_addsub_chain_count, [208](#)
- traverse_list, [204](#)
- traverse_operation_node, [205](#)
- adders.h
 - add_list, [217](#)
 - add_the_blackbox_for_adds, [209](#)
 - chain_list, [217](#)
 - clean_adders, [209](#)
 - declare_hard_adder, [210](#)
 - define_add_function, [210](#)
 - find_hard_adders, [211](#)
 - free_op_nodes, [211](#)
 - hard_adders, [217](#)
 - init_add_distribution, [211](#)
 - instantiate_hard_adder, [212](#)
 - instantiate_simple_soft_adder, [212](#)
 - iterate_adders, [212](#)
 - match_node, [213](#)
 - match_pins, [213](#)
 - match_ports, [213](#)
 - merge_nodes, [214](#)
 - min_add, [218](#)
 - min_threshold_adder, [218](#)
 - processed_adder_list, [218](#)
 - reallocate_pins, [214](#)
 - reduce_operations, [214](#)
 - remove_fanout_pins, [215](#)
 - remove_list_node, [215](#)
 - report_add_distribution, [215](#)
 - split_adder, [216](#)
 - t_adder, [209](#)
 - total, [218](#)
 - traverse_list, [216](#)
 - traverse_operation_node, [217](#)
- addr
 - sp_ram_signals, [31](#)
- addr1
 - dp_ram_signals, [4](#)
- addr2
 - dp_ram_signals, [5](#)
- addr_width
 - implicit_memory, [16](#)
- addsub_list_next
 - netlist_cleanup.cpp, [316](#)
- addsub_nodes
 - netlist_cleanup.cpp, [316](#)
- adjoin_constant
 - ast_elaborate.cpp, [61](#)
 - ast_elaborate.h, [81](#)
- alias_output_assign_pins_to_inputs
 - netlist_create_from_ast.cpp, [321](#)
- all_file_items_list
 - parse_making_ast.cpp, [153](#)
- alloc_and_init_ace_info_object
 - ace.cpp, [521](#)
- alloc_and_init_ace_structs
 - ace.cpp, [522](#)
 - ace.h, [527](#)
- allocate_chain_info
 - netlist_utils.cpp, [366](#)
 - netlist_utils.h, [398](#)
- allocate_children_to_node
 - ast_util.cpp, [102](#)
 - ast_util.h, [118](#)
- allocate_more_input_pins
 - netlist_utils.cpp, [367](#)
 - netlist_utils.h, [399](#)
- allocate_more_output_pins
 - netlist_utils.cpp, [367](#)
 - netlist_utils.h, [399](#)
- allocate_netlist
 - netlist_utils.cpp, [368](#)
 - netlist_utils.h, [400](#)
- allocate_nnet
 - netlist_utils.cpp, [369](#)
 - netlist_utils.h, [401](#)
- allocate_nnode
 - netlist_utils.cpp, [370](#)
 - netlist_utils.h, [402](#)
- allocate_npin
 - netlist_utils.cpp, [371](#)
 - netlist_utils.h, [403](#)
- append_string
 - odin_util.cpp, [463](#)
 - odin_util.h, [481](#)
- Arch
 - globals.h, [53](#)
 - odin_ii.cpp, [461](#)
- assign_memory_from_mif_file
 - simulate_blif.cpp, [537](#)

- simulate_blif.h, 575
- assign_node_to_line
 - simulate_blif.cpp, 537
 - simulate_blif.h, 576
- assign_node_type_from_node_name
 - read_blif.cpp, 184
- assignment_alias
 - netlist_create_from_ast.cpp, 322
- associate_names
 - read_blif.cpp, 184
- ast_elaborate.cpp
 - adjoin_constant, 61
 - calculation, 61
 - change_exp_list, 62
 - change_para_node, 62
 - check_binary_operation, 62
 - check_exp_list, 63
 - check_mult_bracket, 63
 - check_node_number, 63
 - check_operation, 64
 - check_tree_operation, 64
 - combine_constant, 64
 - construct_new_tree, 65
 - copy_enode, 65
 - copy_enode_list, 65
 - count, 79
 - count_assign, 79
 - count_id, 79
 - count_write, 79
 - create_ast_node, 66
 - create_enode, 66
 - create_op_node, 66
 - deal_with_bracket, 67
 - delete_bracket, 67
 - delete_bracket_body, 67
 - delete_bracket_head, 68
 - delete_bracket_tail, 68
 - delete_continuous_multiply, 68
 - e_data, 60
 - e_operation, 60
 - e_variable, 60
 - find_assign_node, 69
 - find_leaf_node, 69
 - find_most_unique_count, 69
 - find_parameter, 70
 - find_tail, 70
 - free_exp_list, 70
 - get_copy_tree, 71
 - has_intermediate_variable, 71
 - head, 79
 - keep_all_branch, 71
 - mark_node_read, 72
 - mark_node_write, 72
 - Max_size, 60

- N, 61
- optimize_for_tree, 72
- p, 80
- read_node, 61
- reallocate_node, 73
- record_expression, 73
- record_tree_info, 73
- recursive_tree, 74
- reduce_assignment_expression, 74
- reduce_enode_list, 74
- reduce_parameter, 75
- remove_intermediate_variable, 75
- remove_para_node, 75
- replace_enode, 76
- search_certain_operation, 76
- search_for_node, 76
- search_marked_node, 77
- shift_operation, 77
- simplify_ast, 77
- simplify_expression, 78
- store_exp_list, 78
- translate_expression, 78
- write_node, 61
- ast_elaborate.h
 - adjoin_constant, 81
 - calculation, 82
 - change_exp_list, 82
 - change_para_node, 82
 - check_binary_operation, 83
 - check_exp_list, 83
 - check_mult_bracket, 83
 - check_node_number, 84
 - check_operation, 84
 - check_tree_operation, 84
 - combine_constant, 85
 - construct_new_tree, 85
 - copy_enode, 85
 - copy_enode_list, 86
 - create_ast_node, 86
 - create_enode, 86
 - create_op_node, 87
 - deal_with_bracket, 87
 - delete_bracket, 87
 - delete_bracket_body, 88
 - delete_bracket_head, 88
 - delete_bracket_tail, 88
 - delete_continuous_multiply, 89
 - enode, 81
 - find_assign_node, 89
 - find_leaf_node, 89
 - find_most_unique_count, 90
 - find_parameter, 90
 - find_tail, 90
 - find_top_module, 91

- free_exp_list, 91
- get_copy_tree, 91
- has_intermediate_variable, 92
- keep_all_branch, 92
- mark_node_read, 92
- mark_node_write, 93
- optimize_for_tree, 93
- reallocate_node, 93
- record_expression, 94
- record_tree_info, 94
- recursive_tree, 94
- reduce_assignment_expression, 95
- reduce_enode_list, 95
- reduce_parameter, 95
- remove_intermediate_variable, 96
- remove_para_node, 96
- replace_enode, 96
- search_certain_operation, 97
- search_for_node, 97
- search_marked_node, 97
- shift_operation, 98
- simplify_ast, 98
- simplify_expression, 98
- store_exp_list, 99
- translate_expression, 99
- ast_functions
 - parse_making_ast.cpp, 153
- ast_modules
 - globals.h, 53
 - parse_making_ast.cpp, 153
- ast_node_deep_copy
 - ast_util.cpp, 103
 - ast_util.h, 120
- ast_util.cpp
 - add_child_at_the_beginning_of_the_node, 101
 - add_child_to_node, 101
 - add_tag_data, 101
 - allocate_children_to_node, 102
 - ast_node_deep_copy, 103
 - change_to_number_node, 104
 - create_node_w_type, 104
 - create_tree_node_id, 105
 - create_tree_node_long_long_number, 106
 - create_tree_node_number, 107
 - fold_binary, 107
 - fold_unary, 108
 - free_assignment_of_node_keep_tree, 108
 - free_single_node, 109
 - free_whole_tree, 109
 - get_name_of_pin_at_bit, 109
 - get_name_of_pins, 110
 - get_name_of_pins_number, 110
 - get_name_of_pins_with_prefix, 111
 - get_name_of_var_declare_at_bit, 111
 - get_range, 112
 - initial_node, 112
 - make_concat_into_list_of_strings, 113
 - make_module_param_name, 113
 - move_ast_node, 114
 - node_is_constant, 114
 - resolve_node, 115
 - update_tree_tag, 116
- ast_util.h
 - add_child_at_the_beginning_of_the_node, 117
 - add_child_to_node, 117
 - add_tag_data, 118
 - allocate_children_to_node, 118
 - ast_node_deep_copy, 120
 - change_to_number_node, 121
 - create_node_w_type, 121
 - create_tree_node_id, 122
 - create_tree_node_long_long_number, 123
 - create_tree_node_number, 124
 - fold_binary, 124
 - fold_unary, 125
 - free_assignment_of_node_keep_tree, 125
 - free_single_node, 126
 - free_whole_tree, 126
 - get_name_of_pin_at_bit, 126
 - get_name_of_pins, 127
 - get_name_of_pins_with_prefix, 127
 - get_name_of_var_declare_at_bit, 128
 - get_range, 128
 - initial_node, 129
 - make_concat_into_list_of_strings, 129
 - make_module_param_name, 130
 - move_ast_node, 130
 - node_is_constant, 131
 - resolve_node, 131
- b
 - adder_signals, 3
 - BITWISE_AND_string
 - node_creation_library.cpp, 433
 - BITWISE_NAND_string
 - node_creation_library.cpp, 434
 - BITWISE_NOR_string
 - node_creation_library.cpp, 434
 - BITWISE_NOT_string
 - node_creation_library.cpp, 434
 - BITWISE_OR_string
 - node_creation_library.cpp, 434
 - BITWISE_XNOR_string
 - node_creation_library.cpp, 434
 - BITWISE_XOR_string
 - node_creation_library.cpp, 434
 - BLIF_ONE_STRING
 - read_blif.cpp, 194

- BLIF_PAD_STRING
 - read_blif.cpp, 194
- BLIF_ZERO_STRING
 - read_blif.cpp, 194
- BPI
 - ace.cpp, 517
- BUF_NODE_string
 - node_creation_library.cpp, 435
- BUFFER_MAX_SIZE
 - simulate_blif.h, 573
- backward_traversal_net_graph_display
 - netlist_visualizer.c, 43
 - netlist_visualizer.cpp, 46
- blif_netlist
 - globals.h, 53
- block_instantiations_instance
 - parse_making_ast.cpp, 153
- block_tag
 - odin_ii.cpp, 461
- build_bdd_for_node
 - ace.cpp, 522
- CARRY_FUNC_string
 - node_creation_library.cpp, 435
- CASE_EQUAL_string
 - node_creation_library.cpp, 435
- CASE_NOT_EQUAL_string
 - node_creation_library.cpp, 435
- CLOCK_NODE_string
 - node_creation_library.cpp, 435
- COMBINATIONAL
 - netlist_create_from_ast.cpp, 321
- cache_hard_block_names
 - hard_blocks.cpp, 219
- calc_cube_switch_prob
 - ace.cpp, 522
- calc_cube_switch_prob_recur
 - ace.cpp, 523
- calc_node_depth
 - ace.cpp, 523
- calc_switch_prob_recur
 - ace.cpp, 524
- calculate_activity
 - ace.cpp, 524
 - ace.h, 527
- calculate_addsub_statistics
 - netlist_cleanup.cpp, 312
- calculate_operation
 - parse_making_ast.cpp, 135
 - parse_making_ast.h, 159
- calculation
 - ast_elaborate.cpp, 61
 - ast_elaborate.h, 82
- chain_list
 - adders.cpp, 206
 - adders.h, 217
- change_exp_list
 - ast_elaborate.cpp, 62
 - ast_elaborate.h, 82
- change_para_node
 - ast_elaborate.cpp, 62
 - ast_elaborate.h, 82
- change_to_number_node
 - ast_util.cpp, 104
 - ast_util.h, 121
- check_binary_operation
 - ast_elaborate.cpp, 62
 - ast_elaborate.h, 83
- check_exp_list
 - ast_elaborate.cpp, 63
 - ast_elaborate.h, 83
- check_for_initial_reg_value
 - netlist_create_from_ast.cpp, 323
- check_memories_and_report_distribution
 - memories.cpp, 246
 - memories.h, 266
- check_mult_bracket
 - ast_elaborate.cpp, 63
 - ast_elaborate.h, 83
- check_netlist
 - netlist_check.cpp, 304
 - netlist_check.h, 310
- check_node_number
 - ast_elaborate.cpp, 63
 - ast_elaborate.h, 84
- check_operation
 - ast_elaborate.cpp, 64
 - ast_elaborate.h, 84
- check_tree_operation
 - ast_elaborate.cpp, 64
 - ast_elaborate.h, 84
- cin
 - adder_signals, 3
- clean_adders
 - adders.cpp, 197
 - adders.h, 209
- clean_adders_for_sub
 - subtractions.cpp, 295
 - subtractions.h, 300
- clean_multipliers
 - multipliers.cpp, 280
 - multipliers.h, 290
- clean_up_parser_for_file
 - parse_making_ast.cpp, 135
 - parse_making_ast.h, 159
- clean_veri_define
 - verilog_preprocessor.cpp, 502
 - verilog_preprocessor.h, 511

- clean_veri_include
 - verilog_preprocessor.cpp, 503
 - verilog_preprocessor.h, 511
- cleanup_hard_blocks
 - parse_making_ast.cpp, 135
 - parse_making_ast.h, 160
- cleanup_parser
 - parse_making_ast.cpp, 136
 - parse_making_ast.h, 160
- cleanup_veri_preproc
 - verilog_preprocessor.cpp, 503
 - verilog_preprocessor.h, 511
- clk
 - dp_ram_signals, 5
 - sp_ram_signals, 31
- clock_added
 - implicit_memory, 17
- collapse_implicit_memory_to_single_port_ram
 - implicit_memory.cpp, 232
- combine_constant
 - ast_elaborate.cpp, 64
 - ast_elaborate.h, 85
- combine_lists
 - netlist_utils.cpp, 372
 - netlist_utils.h, 404
- combine_lists_without_freeing_originals
 - netlist_utils.cpp, 373
 - netlist_utils.h, 405
- combine_nets
 - netlist_utils.cpp, 374
 - netlist_utils.h, 406
- compare_test_vectors
 - simulate_blif.cpp, 538
 - simulate_blif.h, 576
- compute_add_node
 - simulate_blif.cpp, 538
 - simulate_blif.h, 577
- compute_and_store_value
 - simulate_blif.cpp, 539
 - simulate_blif.h, 577
- compute_dual_port_memory
 - simulate_blif.cpp, 539
 - simulate_blif.h, 578
- compute_flipflop_node
 - simulate_blif.cpp, 540
 - simulate_blif.h, 578
- compute_generic_node
 - simulate_blif.cpp, 540
 - simulate_blif.h, 578
- compute_hard_ip_node
 - simulate_blif.cpp, 540
 - simulate_blif.h, 579
- compute_memory_address
 - simulate_blif.cpp, 541
- simulate_blif.h, 579
- compute_memory_node
 - simulate_blif.cpp, 541
 - simulate_blif.h, 579
- compute_multiply_node
 - simulate_blif.cpp, 541
 - simulate_blif.h, 580
- compute_mux_2_node
 - simulate_blif.cpp, 542
 - simulate_blif.h, 580
- compute_single_port_memory
 - simulate_blif.cpp, 542
 - simulate_blif.h, 580
- compute_switching_activities
 - ace.cpp, 525
- compute_unary_sub_node
 - simulate_blif.cpp, 542
 - simulate_blif.h, 581
- concatenate_signal_lists
 - netlist_create_from_ast.cpp, 323
- configuration
 - globals.h, 53
 - read_xml_config_file.cpp, 499
- connect_function_instantiation_and_alias
 - netlist_create_from_ast.cpp, 324
- connect_hard_block_and_alias
 - netlist_create_from_ast.cpp, 325
- connect_memory_and_alias
 - netlist_create_from_ast.cpp, 326
 - netlist_create_from_ast.h, 356
- connect_module_instantiation_and_alias
 - netlist_create_from_ast.cpp, 327
- connect_nodes
 - netlist_utils.cpp, 374
 - netlist_utils.h, 406
- construct_new_tree
 - ast_elaborate.cpp, 65
 - ast_elaborate.h, 85
- convert_ast_to_netlist_recurring_via_modules
 - netlist_create_from_ast.cpp, 328
- convert_binary_string_of_size_to_bit_string
 - odin_util.cpp, 463
 - odin_util.h, 481
- convert_dec_string_of_size_to_long_long
 - odin_util.cpp, 464
 - odin_util.h, 481
- convert_hex_string_of_size_to_bit_string
 - odin_util.cpp, 464
 - odin_util.h, 482
- convert_long_long_to_bit_string
 - odin_util.cpp, 464
 - odin_util.h, 482
- convert_oct_string_of_size_to_bit_string
 - odin_util.cpp, 465

- odin_util.h, [483](#)
- convert_string_of_radix_to_bit_string
 - odin_util.cpp, [465](#)
 - odin_util.h, [483](#)
- convert_string_of_radix_to_long_long
 - odin_util.cpp, [466](#)
 - odin_util.h, [484](#)
- copy_enode
 - ast_elaborate.cpp, [65](#)
 - ast_elaborate.h, [85](#)
- copy_enode_list
 - ast_elaborate.cpp, [65](#)
 - ast_elaborate.h, [86](#)
- copy_input_npin
 - netlist_utils.cpp, [375](#)
 - netlist_utils.h, [407](#)
- copy_input_port_to_memory
 - memories.cpp, [246](#)
- copy_input_signals
 - netlist_utils.cpp, [376](#)
 - netlist_utils.h, [408](#)
- copy_output_npin
 - netlist_utils.cpp, [376](#)
 - netlist_utils.h, [408](#)
- copy_output_port_to_memory
 - memories.cpp, [247](#)
- copy_output_signals
 - netlist_utils.cpp, [377](#)
 - netlist_utils.h, [409](#)
- count
 - ast_elaborate.cpp, [79](#)
 - hard_block_models, [10](#)
 - hard_block_pins, [11](#)
 - hard_block_ports, [12](#)
 - hashtable_t_t, [14](#)
 - lines_t, [19](#)
 - pin_names, [22](#)
 - queue_t_t, [24](#)
 - stages_t, [32](#)
 - test_vector, [36](#)
- count_assign
 - ast_elaborate.cpp, [79](#)
- count_blif_lines
 - read_blif.cpp, [185](#)
- count_id
 - ast_elaborate.cpp, [79](#)
- count_nodes_in_netlist
 - netlist_utils.cpp, [377](#)
 - netlist_utils.h, [409](#)
- count_test_vectors
 - simulate_blif.cpp, [543](#)
 - simulate_blif.h, [581](#)
- count_write
 - ast_elaborate.cpp, [79](#)

- counts
 - stages_t, [32](#)
 - test_vector, [37](#)
- cout
 - adder_signals, [3](#)
- create_all_driver_nets_in_this_function
 - netlist_create_from_ast.cpp, [328](#)
- create_all_driver_nets_in_this_module
 - netlist_create_from_ast.cpp, [329](#)
- create_ast_node
 - ast_elaborate.cpp, [66](#)
 - ast_elaborate.h, [86](#)
- create_case
 - netlist_create_from_ast.cpp, [330](#)
- create_case_control_signals
 - netlist_create_from_ast.cpp, [331](#)
- create_case_mux_statements
 - netlist_create_from_ast.cpp, [332](#)
- create_decoder
 - memories.cpp, [247](#)
 - memories.h, [266](#)
- create_dual_port_ram_block
 - netlist_create_from_ast.cpp, [332](#)
- create_enode
 - ast_elaborate.cpp, [66](#)
 - ast_elaborate.h, [86](#)
- create_gate
 - netlist_create_from_ast.cpp, [333](#)
- create_hard_block
 - netlist_create_from_ast.cpp, [333](#)
- create_hard_block_models
 - read_blif.cpp, [185](#)
- create_hard_block_nodes
 - read_blif.cpp, [186](#)
- create_hashtable
 - hashtable.cpp, [613](#)
 - hashtable.h, [615](#)
- create_if
 - netlist_create_from_ast.cpp, [334](#)
- create_if_control_signals
 - netlist_create_from_ast.cpp, [335](#)
- create_if_for_question
 - netlist_create_from_ast.cpp, [336](#)
- create_if_mux_statements
 - netlist_create_from_ast.cpp, [336](#)
- create_if_question_mux_expressions
 - netlist_create_from_ast.cpp, [337](#)
- create_implicit_memory_block
 - implicit_memory.cpp, [233](#)
 - implicit_memory.h, [239](#)
- create_internal_node_and_driver
 - read_blif.cpp, [186](#)
- create_latch_node_and_driver
 - read_blif.cpp, [186](#)

- create_line
 - simulate_blif.cpp, 543
 - simulate_blif.h, 581
- create_lines
 - simulate_blif.cpp, 544
 - simulate_blif.h, 582
- create_mux_expressions
 - netlist_create_from_ast.cpp, 337
- create_mux_statements
 - netlist_create_from_ast.cpp, 338
- create_netlist
 - netlist_create_from_ast.cpp, 338
 - netlist_create_from_ast.h, 357
- create_node_w_type
 - ast_util.cpp, 104
 - ast_util.h, 121
- create_op_node
 - ast_elaborate.cpp, 66
 - ast_elaborate.h, 87
- create_operation_node
 - netlist_create_from_ast.cpp, 339
- create_output_pin
 - netlist_create_from_ast.cpp, 339
- create_param_table_for_module
 - netlist_create_from_ast.cpp, 340
- create_pins
 - netlist_create_from_ast.cpp, 340
- create_queue
 - queue.cpp, 531
 - queue.h, 532
- create_single_port_ram_block
 - netlist_create_from_ast.cpp, 341
- create_soft_dual_port_ram_block
 - netlist_create_from_ast.cpp, 342
- create_soft_single_port_ram_block
 - netlist_create_from_ast.cpp, 342
- create_symbol_table_for_function
 - netlist_create_from_ast.cpp, 343
- create_symbol_table_for_module
 - netlist_create_from_ast.cpp, 344
- create_top_driver_nets
 - netlist_create_from_ast.cpp, 345
- create_top_output_nodes
 - netlist_create_from_ast.cpp, 345
- create_tree_node_id
 - ast_util.cpp, 105
 - ast_util.h, 122
- create_tree_node_long_long_number
 - ast_util.cpp, 106
 - ast_util.h, 123
- create_tree_node_number
 - ast_util.cpp, 107
 - ast_util.h, 124
- cube
 - ace_cube_t, 2
- current_index
 - veri_Defines, 39
 - veri_Includes, 42
- current_parse_file
 - globals.h, 53
 - odin_ii.cpp, 461
- current_size
 - veri_Defines, 39
 - veri_Includes, 42
- DASH
 - ace.cpp, 517
- DEFAULT_CLOCK_NAME
 - read_blif.cpp, 194
- DIVIDE_string
 - node_creation_library.cpp, 435
- DUAL_PORT_MEMORY_NAME
 - simulate_blif.h, 573
- data
 - exp_node, 7
 - STRING_CACHE, 34
 - sp_ram_signals, 31
- data1
 - dp_ram_signals, 5
- data2
 - dp_ram_signals, 5
- data_width
 - implicit_memory, 17
- deal_with_bracket
 - ast_elaborate.cpp, 67
 - ast_elaborate.h, 87
- declare_hard_adder
 - adders.cpp, 197
 - adders.h, 210
- declare_hard_adder_for_sub
 - subtractions.cpp, 296
 - subtractions.h, 300
- declare_hard_multiplier
 - multipliers.cpp, 281
 - multipliers.h, 290
- default_choices
 - ParseInitRegState, 21
- DefaultSize
 - verilog_preprocessor.h, 509
- define_add_function
 - adders.cpp, 197
 - adders.h, 210
- define_decoded_mux
 - output_blif.cpp, 177
- define_ff
 - output_blif.cpp, 177
- define_hard_block
 - hard_blocks.cpp, 219

- hard_blocks.h, 225
- define_latches_initial_value_inside_initial_statement
 - netlist_create_from_ast.cpp, 346
- define_logical_function
 - output_blif.cpp, 177
- define_mult_function
 - multipliers.cpp, 281
 - multipliers.h, 290
- define_nets_with_driver
 - netlist_create_from_ast.cpp, 346
- define_nodes_and_nets_with_driver
 - netlist_create_from_ast.cpp, 347
- define_set_input_logical_function
 - output_blif.cpp, 178
- defined_constants
 - veri_Defines, 39
- defined_in
 - veri_define, 37
- defines_for_file_sc
 - parse_making_ast.cpp, 154
- defines_for_function_sc
 - parse_making_ast.cpp, 154
- defines_for_module_sc
 - parse_making_ast.cpp, 154
- delete_bracket
 - ast_elaborate.cpp, 67
 - ast_elaborate.h, 87
- delete_bracket_body
 - ast_elaborate.cpp, 67
 - ast_elaborate.h, 88
- delete_bracket_head
 - ast_elaborate.cpp, 68
 - ast_elaborate.h, 88
- delete_bracket_tail
 - ast_elaborate.cpp, 68
 - ast_elaborate.h, 88
- delete_continuous_multiply
 - ast_elaborate.cpp, 68
 - ast_elaborate.h, 89
- depth_first_traversal_check_if_forward_levelled
 - netlist_check.cpp, 305
- depth_first_traversal_graph_display
 - netlist_visualizer.c, 43
 - netlist_visualizer.cpp, 47
- depth_first_traversal_to_output
 - output_blif.cpp, 178
- depth_first_traversal_to_partial_map
 - partial_map.cpp, 450
- depth_first_traverse_check_if_forward_levelled
 - netlist_check.cpp, 305
- depth_first_traverse_parital_map
 - partial_map.cpp, 450
- depth_first_traverse_until_next_ff_or_output
 - netlist_check.cpp, 306
- depth_first_traverse_visualize
 - netlist_visualizer.c, 44
 - netlist_visualizer.cpp, 47
- depth_traverse_check_combinational_loop
 - netlist_check.cpp, 306
- depth_traverse_count
 - netlist_utils.cpp, 377
- depth_traverse_output_blif
 - output_blif.cpp, 179
- deregister_hard_blocks
 - hard_blocks.cpp, 220
 - hard_blocks.h, 225
- destroy
 - hashtable_t_t, 15
 - queue_t_t, 24
- destroy_free_items
 - hashtable_t_t, 15
- dp_memory_list
 - memories.cpp, 262
 - memories.h, 278
- dp_ram_signals, 4
 - addr1, 4
 - addr2, 5
 - clk, 5
 - data1, 5
 - data2, 5
 - out1, 5
 - out2, 5
 - we1, 6
 - we2, 6
- dual_port_rams
 - memories.cpp, 262
 - memories.h, 278
- e_data
 - ast_elaborate.cpp, 60
- e_operation
 - ast_elaborate.cpp, 60
- e_variable
 - ast_elaborate.cpp, 60
- empty_string
 - read_xml_config_file.cpp, 499
- enode
 - ast_elaborate.h, 81
- enqueue_node_if_ready
 - simulate_blif.cpp, 544
 - simulate_blif.h, 582
- error_message
 - odin_util.cpp, 466
- evaluate_sensitivity_list
 - netlist_create_from_ast.cpp, 347
- exp_node, 6
 - data, 7
 - flag, 7

- id, [7](#)
- next, [7](#)
- operation, [7](#)
- pre, [7](#)
- priority, [7](#)
- type, [8](#)
- variable, [8](#)
- FF_NODE_string
 - node_creation_library.cpp, [436](#)
- fail
 - ace.cpp, [517](#)
- file_line_number
 - globals.h, [54](#)
 - read_blif.cpp, [194](#)
- filter_memories_by_soft_logic_cutoff
 - memories.cpp, [247](#)
- finalize_implicit_memory
 - implicit_memory.cpp, [233](#)
- find_assign_node
 - ast_elaborate.cpp, [69](#)
 - ast_elaborate.h, [89](#)
- find_hard_adders
 - adders.cpp, [198](#)
 - adders.h, [211](#)
- find_hard_block
 - hard_blocks.cpp, [220](#)
 - hard_blocks.h, [226](#)
- find_hard_multipliers
 - multipliers.cpp, [282](#)
 - multipliers.h, [291](#)
- find_leaf_node
 - ast_elaborate.cpp, [69](#)
 - ast_elaborate.h, [89](#)
- find_most_unique_count
 - ast_elaborate.cpp, [69](#)
 - ast_elaborate.h, [90](#)
- find_node_at_top_of_combo_loop
 - netlist_check.cpp, [306](#)
- find_parameter
 - ast_elaborate.cpp, [70](#)
 - ast_elaborate.h, [90](#)
- find_portname_in_lines
 - simulate_blif.cpp, [544](#)
 - simulate_blif.h, [583](#)
- find_smallest_non_numerical
 - netlist_create_from_ast.cpp, [348](#)
- find_substring
 - odin_util.cpp, [467](#)
 - odin_util.h, [484](#)
- find_tail
 - ast_elaborate.cpp, [70](#)
 - ast_elaborate.h, [90](#)
- find_top_module
 - ast_elaborate.h, [91](#)
 - netlist_create_from_ast.cpp, [349](#)
- flag
 - exp_node, [7](#)
 - veri_flag_node, [40](#)
- flag_undriven_input_pins
 - simulate_blif.cpp, [545](#)
 - simulate_blif.h, [583](#)
- fold_binary
 - ast_util.cpp, [107](#)
 - ast_util.h, [124](#)
- fold_unary
 - ast_util.cpp, [108](#)
 - ast_util.h, [125](#)
- format_verilog_file
 - verilog_preprocessor.cpp, [503](#)
- format_verilog_variable
 - verilog_preprocessor.cpp, [504](#)
- forward_traversal_net_graph_display
 - netlist_visualizer.c, [44](#)
 - netlist_visualizer.cpp, [48](#)
- free
 - STRING_CACHE, [35](#)
- free_assignment_of_node_keep_tree
 - ast_util.cpp, [108](#)
 - ast_util.h, [125](#)
- free_dp_ram_signals
 - memories.cpp, [248](#)
 - memories.h, [267](#)
- free_exp_list
 - ast_elaborate.cpp, [70](#)
 - ast_elaborate.h, [91](#)
- free_hard_block_model
 - read_blif.cpp, [186](#)
- free_hard_block_models
 - read_blif.cpp, [187](#)
- free_hard_block_pins
 - read_blif.cpp, [187](#)
- free_hard_block_ports
 - read_blif.cpp, [187](#)
- free_implicit_memory_index_and_finalize_memories
 - implicit_memory.cpp, [234](#)
 - implicit_memory.h, [240](#)
- free_lines
 - simulate_blif.cpp, [545](#)
 - simulate_blif.h, [583](#)
- free_memory_lists
 - memories.cpp, [248](#)
 - memories.h, [267](#)
- free_netlist
 - netlist_utils.cpp, [378](#)
 - netlist_utils.h, [409](#)
- free_nnode
 - netlist_utils.cpp, [378](#)

- netlist_utils.h, 410
- free_npin
 - netlist_utils.h, 410
- free_op_nodes
 - adders.cpp, 198
 - adders.h, 211
- free_pin_name_list
 - simulate_blif.cpp, 545
 - simulate_blif.h, 584
- free_signal_list
 - netlist_utils.cpp, 379
 - netlist_utils.h, 410
- free_single_node
 - ast_util.cpp, 109
 - ast_util.h, 126
- free_sp_ram_signals
 - memories.cpp, 249
 - memories.h, 268
- free_stages
 - simulate_blif.cpp, 546
 - simulate_blif.h, 584
- free_test_vector
 - simulate_blif.cpp, 546
 - simulate_blif.h, 585
- free_whole_tree
 - ast_util.cpp, 109
 - ast_util.h, 126
- from_str
 - ParseInitRegState, 21
- function_instantiations_instance
 - parse_making_ast.cpp, 154
- function_instantiations_instance_by_module
 - parse_making_ast.cpp, 154
- function_local_symbol_table
 - netlist_create_from_ast.cpp, 353
- function_local_symbol_table_sc
 - globals.h, 54
 - netlist_create_from_ast.cpp, 353
- function_num_local_symbol_table
 - netlist_create_from_ast.cpp, 353
- function_to_print_node_and_its_pin
 - print_netlist.c, 50
 - print_netlist.cpp, 51
- functions_inputs_sc
 - parse_making_ast.cpp, 154
- functions_outputs_sc
 - parse_making_ast.cpp, 155
- GETINPUT
 - ace.cpp, 518
- GND_NAME
 - read_blif.cpp, 183
- GND_NODE_string
 - node_creation_library.cpp, 436
- GT_string
 - node_creation_library.cpp, 436
- GTE_string
 - node_creation_library.cpp, 436
- generate_hard_block_ports_signature
 - read_blif.cpp, 188
- generate_random_test_vector
 - simulate_blif.cpp, 547
 - simulate_blif.h, 585
- generate_sc_hash
 - string_cache.cpp, 616
- generate_vector_header
 - simulate_blif.cpp, 547
 - simulate_blif.h, 586
- geomean_addsub_length
 - adders.cpp, 206
 - netlist_cleanup.cpp, 316
- get
 - hashtable_t_t, 15
- get_all
 - hashtable_t_t, 15
- get_bit
 - odin_util.cpp, 468
 - odin_util.h, 484
- get_children_of
 - simulate_blif.cpp, 548
 - simulate_blif.h, 586
- get_children_of_nodepin
 - simulate_blif.cpp, 548
 - simulate_blif.h, 586
- get_children_pinnumber_of
 - simulate_blif.cpp, 548
 - simulate_blif.h, 587
- get_circuit_filename
 - simulate_blif.cpp, 549
 - simulate_blif.h, 587
- get_clock_ratio
 - simulate_blif.cpp, 549
 - simulate_blif.h, 587
- get_copy_tree
 - ast_elaborate.cpp, 71
 - ast_elaborate.h, 91
- get_dp_ram_depth
 - memories.cpp, 249
 - memories.h, 268
- get_dp_ram_signals
 - memories.cpp, 250
 - memories.h, 269
- get_dp_ram_split_depth
 - memories.cpp, 250
 - memories.h, 269
- get_dp_ram_split_width
 - memories.cpp, 251
- get_dp_ram_width

- memories.cpp, 251
- memories.h, 270
- get_hard_block_model
 - read_blif.cpp, 188
- get_hard_block_pin_number
 - read_blif.cpp, 188
- get_hard_block_port_name
 - read_blif.cpp, 189
- get_hard_block_ports
 - read_blif.cpp, 189
- get_input_pin_index_from_mapping
 - netlist_utils.cpp, 380
 - netlist_utils.h, 411
- get_input_port_index_from_mapping
 - netlist_utils.cpp, 380
 - netlist_utils.h, 412
- get_line_pin_value
 - simulate_blif.cpp, 549
 - simulate_blif.h, 587
- get_memory_port_size
 - memories.cpp, 252
 - memories.h, 270
- get_mif_filename
 - simulate_blif.cpp, 549
 - simulate_blif.h, 588
- get_model_port
 - hard_blocks.cpp, 221
 - hard_blocks.h, 226
- get_name_of_pin_at_bit
 - ast_util.cpp, 109
 - ast_util.h, 126
- get_name_of_pins
 - ast_util.cpp, 110
 - ast_util.h, 127
- get_name_of_pins_number
 - ast_util.cpp, 110
- get_name_of_pins_with_prefix
 - ast_util.cpp, 111
 - ast_util.h, 127
- get_name_of_var_declare_at_bit
 - ast_util.cpp, 111
 - ast_util.h, 128
- get_next_vector
 - simulate_blif.cpp, 550
 - simulate_blif.h, 588
- get_num_covered_nodes
 - simulate_blif.cpp, 550
 - simulate_blif.h, 589
- get_one_pin
 - netlist_utils.h, 412
 - node_creation_library.cpp, 424
 - node_creation_library.h, 441
- get_options
 - odin_ii.cpp, 459
- get_output_pin_index_from_mapping
 - netlist_utils.cpp, 381
 - netlist_utils.h, 413
- get_output_port_index_from_mapping
 - netlist_utils.cpp, 381
 - netlist_utils.h, 413
- get_pad_pin
 - netlist_utils.h, 414
 - node_creation_library.cpp, 424
- get_pin_cycle
 - simulate_blif.cpp, 551
 - simulate_blif.h, 589
- get_pin_name
 - odin_util.cpp, 468
 - odin_util.h, 485
- get_pin_number
 - odin_util.cpp, 469
 - odin_util.h, 486
- get_pin_value
 - simulate_blif.cpp, 551
 - simulate_blif.h, 589
- get_port_name
 - odin_util.cpp, 469
 - odin_util.h, 486
- get_range
 - ast_util.cpp, 112
 - ast_util.h, 128
- get_sp_ram_depth
 - memories.cpp, 252
 - memories.h, 271
- get_sp_ram_signals
 - memories.cpp, 253
 - memories.h, 271
- get_sp_ram_split_depth
 - memories.cpp, 253
 - memories.h, 272
- get_sp_ram_split_width
 - memories.cpp, 253
- get_sp_ram_width
 - memories.cpp, 254
 - memories.h, 272
- get_values_offset
 - simulate_blif.cpp, 552
 - simulate_blif.h, 590
- get_zero_pin
 - netlist_utils.h, 414
 - node_creation_library.cpp, 425
 - node_creation_library.h, 441
- global_args
 - globals.h, 54
 - netlist_utils.cpp, 389
 - odin_ii.cpp, 461
- global_args_read_blif
 - globals.h, 54

- global_param_table_sc
 - globals.h, [54](#)
 - netlist_create_from_ast.cpp, [353](#)
- globals.h
 - Arch, [53](#)
 - ast_modules, [53](#)
 - blif_netlist, [53](#)
 - configuration, [53](#)
 - current_parse_file, [53](#)
 - file_line_number, [54](#)
 - function_local_symbol_table_sc, [54](#)
 - global_args, [54](#)
 - global_args_read_blif, [54](#)
 - global_param_table_sc, [54](#)
 - gnd_node, [54](#)
 - input_nets_sc, [54](#)
 - local_symbol_table_sc, [55](#)
 - module_names_to_idx, [55](#)
 - num_modules, [55](#)
 - num_top_input_nodes, [55](#)
 - num_top_output_nodes, [55](#)
 - one_net, [55](#)
 - one_string, [55](#)
 - output_nets_sc, [56](#)
 - pad_net, [56](#)
 - pad_node, [56](#)
 - pad_string, [56](#)
 - read_blif_netlist, [56](#)
 - to_view_parse, [56](#)
 - top_input_nodes, [56](#)
 - top_module, [57](#)
 - top_output_nodes, [57](#)
 - type_descriptors, [57](#)
 - vcc_node, [57](#)
 - verilog_netlist, [57](#)
 - yylineno, [57](#)
 - zero_net, [57](#)
 - zero_string, [58](#)
- gnd_node
 - globals.h, [54](#)
- graphVizOutputAst
 - parse_making_ast.cpp, [136](#)
 - parse_making_ast.h, [160](#)
- graphVizOutputAst_traverse_node
 - parse_making_ast.cpp, [137](#)
 - parse_making_ast.h, [161](#)
- graphVizOutputCombinationalNet
 - netlist_visualizer.c, [45](#)
 - netlist_visualizer.cpp, [48](#)
 - netlist_visualizer.h, [49](#)
- graphVizOutputNetlist
 - netlist_visualizer.c, [45](#)
 - netlist_visualizer.cpp, [48](#)
 - netlist_visualizer.h, [49](#)
- graphVizOutputPreproc
 - parse_making_ast.cpp, [137](#)
- HARD_IP_string
 - node_creation_library.cpp, [436](#)
- HBPAD_NAME
 - read_blif.cpp, [183](#)
- hard_adders
 - adders.cpp, [206](#)
 - adders.h, [217](#)
- hard_block_model, [8](#)
 - input_ports, [8](#)
 - inputs, [9](#)
 - name, [9](#)
 - output_ports, [9](#)
 - outputs, [9](#)
- hard_block_models, [9](#)
 - count, [10](#)
 - index, [10](#)
 - models, [10](#)
- hard_block_names
 - hard_blocks.cpp, [224](#)
 - hard_blocks.h, [229](#)
- hard_block_pins, [10](#)
 - count, [11](#)
 - index, [11](#)
 - names, [11](#)
- hard_block_port_direction
 - hard_blocks.cpp, [222](#)
 - hard_blocks.h, [227](#)
- hard_block_port_size
 - hard_blocks.cpp, [222](#)
 - hard_blocks.h, [228](#)
- hard_block_ports, [11](#)
 - count, [12](#)
 - index, [12](#)
 - names, [12](#)
 - signature, [12](#)
 - sizes, [12](#)
- hard_blocks.cpp
 - cache_hard_block_names, [219](#)
 - define_hard_block, [219](#)
 - deregister_hard_blocks, [220](#)
 - find_hard_block, [220](#)
 - get_model_port, [221](#)
 - hard_block_names, [224](#)
 - hard_block_port_direction, [222](#)
 - hard_block_port_size, [222](#)
 - instantiate_hard_block, [223](#)
 - output_hard_blocks, [223](#)
 - register_hard_blocks, [224](#)
- hard_blocks.h
 - define_hard_block, [225](#)
 - deregister_hard_blocks, [225](#)

- find_hard_block, 226
- get_model_port, 226
- hard_block_names, 229
- hard_block_port_direction, 227
- hard_block_port_size, 228
- instantiate_hard_block, 228
- output_hard_blocks, 228
- register_hard_blocks, 229
- hard_multipliers
 - multipliers.cpp, 288
 - multipliers.h, 294
- hard_node_name
 - node_creation_library.cpp, 426
 - node_creation_library.h, 441
- has_intermediate_variable
 - ast_elaborate.cpp, 71
 - ast_elaborate.h, 92
- hashtable.cpp
 - __hashtable_add, 609
 - __hashtable_compare_keys, 609
 - __hashtable_destroy, 610
 - __hashtable_destroy_free_items, 610
 - __hashtable_get, 611
 - __hashtable_get_all, 611
 - __hashtable_hash, 612
 - __hashtable_is_empty, 612
 - __hashtable_remove, 613
 - create_hashtable, 613
- hashtable.h
 - create_hashtable, 615
 - hashtable_node_t, 614
 - hashtable_t, 615
- hashtable_node_t
 - hashtable.h, 614
- hashtable_node_t_t, 13
 - item, 13
 - key, 13
 - key_length, 13
 - next, 13
- hashtable_t
 - hashtable.h, 615
- hashtable_t_t, 14
 - add, 14
 - count, 14
 - destroy, 15
 - destroy_free_items, 15
 - get, 15
 - get_all, 15
 - is_empty, 15
 - remove, 15
 - store, 16
 - store_size, 16
- head
 - ast_elaborate.cpp, 79
- queue_t_t, 24
- hook_up_nets
 - read_blif.cpp, 189
- hook_up_node
 - read_blif.cpp, 189
- hookup_hb_input_pins_from_signal_list
 - netlist_utils.cpp, 382
 - netlist_utils.h, 415
- hookup_input_pins_from_signal_list
 - netlist_utils.cpp, 382
 - netlist_utils.h, 416
- hookup_output_pins_from_signal_list
 - netlist_utils.cpp, 383
 - netlist_utils.h, 416
- INPUT_NODE_string
 - node_creation_library.cpp, 436
- INPUT_VECTOR_FILE_NAME
 - simulate_blif.h, 573
- INstantiate_DRIVERS
 - netlist_create_from_ast.cpp, 321
- id
 - exp_node, 7
- identify_unused_nodes
 - netlist_cleanup.cpp, 312
- implicit_memories
 - implicit_memory.cpp, 237
 - implicit_memory.h, 243
- implicit_memory, 16
 - addr_width, 16
 - clock_added, 17
 - data_width, 17
 - name, 17
 - node, 17
 - output_added, 17
- implicit_memory.cpp
 - add_dummy_input_port_to_implicit_memory, 230
 - add_dummy_output_port_to_implicit_memory, 231
 - add_input_port_to_implicit_memory, 231
 - add_output_port_to_implicit_memory, 232
 - collapse_implicit_memory_to_single_port_ram, 232
 - create_implicit_memory_block, 233
 - finalize_implicit_memory, 233
 - free_implicit_memory_index_and_finalize_memories, 234
 - implicit_memories, 237
 - implicit_memory_inputs, 237
 - init_implicit_memory_index, 234
 - is_valid_implicit_memory_reference_ast, 234
 - lookup_implicit_memory, 235
 - lookup_implicit_memory_input, 235
 - lookup_implicit_memory_reference_ast, 236
 - register_implicit_memory_input, 236
- implicit_memory.h

- add_input_port_to_implicit_memory, 238
- add_output_port_to_implicit_memory, 238
- create_implicit_memory_block, 239
- free_implicit_memory_index_and_finalize_memories, 240
- implicit_memories, 243
- implicit_memory_inputs, 243
- init_implicit_memory_index, 240
- is_valid_implicit_memory_reference_ast, 241
- lookup_implicit_memory_input, 241
- lookup_implicit_memory_reference_ast, 242
- register_implicit_memory_input, 242
- implicit_memory_inputs
 - implicit_memory.cpp, 237
 - implicit_memory.h, 243
- included_files
 - veri_includes, 42
- included_from
 - veri_include, 41
- index
 - hard_block_models, 10
 - hard_block_pins, 11
 - hard_block_ports, 12
- index_names
 - read_blif.cpp, 190
- index_pin_name_list
 - simulate_blif.cpp, 552
 - simulate_blif.h, 590
- init_add_distribution
 - adders.cpp, 199
 - adders.h, 211
- init_cascade_adder
 - multipliers.cpp, 282
- init_implicit_memory_index
 - implicit_memory.cpp, 234
 - implicit_memory.h, 240
- init_memory_distribution
 - memories.h, 273
- init_mult_distribution
 - multipliers.cpp, 283
 - multipliers.h, 291
- init_parser
 - parse_making_ast.cpp, 137
 - parse_making_ast.h, 161
- init_parser_for_file
 - parse_making_ast.cpp, 138
 - parse_making_ast.h, 162
- init_signal_list
 - netlist_utils.cpp, 384
 - netlist_utils.h, 417
- init_split_adder
 - adders.cpp, 199
- init_split_adder_for_sub
 - subtractions.cpp, 296
- init_split_multiplier
 - multipliers.cpp, 283
- init_sub_distribution
 - subtractions.h, 301
- init_veri_preproc
 - verilog_preprocessor.cpp, 504
 - verilog_preprocessor.h, 512
- initial_node
 - ast_util.cpp, 112
 - ast_util.h, 129
- initialize_pin
 - simulate_blif.cpp, 553
 - simulate_blif.h, 591
- input_nets_sc
 - globals.h, 54
 - netlist_create_from_ast.cpp, 354
- input_ports
 - hard_block_model, 8
- inputs
 - hard_block_model, 9
- insert_node_list
 - netlist_cleanup.cpp, 313
- insert_pin_into_line
 - simulate_blif.cpp, 553
 - simulate_blif.h, 591
- instantiate_EQUAL
 - partial_map.cpp, 452
- instantiate_GE
 - partial_map.cpp, 453
- instantiate_GT
 - partial_map.cpp, 453
- instantiate_add_w_carry
 - partial_map.cpp, 451
 - partial_map.h, 457
- instantiate_bitwise_logic
 - partial_map.cpp, 451
- instantiate_bitwise_reduction
 - partial_map.cpp, 452
- instantiate_buffer
 - partial_map.cpp, 452
- instantiate_hard_adder
 - adders.cpp, 200
 - adders.h, 212
- instantiate_hard_adder_subtraction
 - subtractions.cpp, 297
 - subtractions.h, 301
- instantiate_hard_block
 - hard_blocks.cpp, 223
 - hard_blocks.h, 228
- instantiate_hard_multiplier
 - multipliers.cpp, 284
 - multipliers.h, 292
- instantiate_logical_logic
 - partial_map.cpp, 454

- instantiate_memory
 - simulate_blif.cpp, 554
 - simulate_blif.h, 592
- instantiate_multi_port_mux
 - partial_map.cpp, 454
 - partial_map.h, 457
- instantiate_not_logic
 - partial_map.cpp, 454
- instantiate_shift_left_or_right
 - partial_map.cpp, 455
- instantiate_simple_soft_adder
 - adders.h, 212
- instantiate_simple_soft_multiplier
 - multipliers.cpp, 284
 - multipliers.h, 292
- instantiate_soft_dual_port_ram
 - memories.cpp, 254
 - memories.h, 273
- instantiate_soft_logic_ram
 - partial_map.cpp, 455
- instantiate_soft_single_port_ram
 - memories.cpp, 254
 - memories.h, 273
- instantiate_sub_w_carry
 - partial_map.cpp, 455
- instantiate_unary_sub
 - partial_map.cpp, 456
- is_ast_dp_ram
 - memories.cpp, 255
 - memories.h, 274
- is_ast_sp_ram
 - memories.cpp, 255
 - memories.h, 274
- is_binary_string
 - odin_util.cpp, 470
 - odin_util.h, 487
- is_clock_node
 - simulate_blif.cpp, 554
 - simulate_blif.h, 592
- is_decimal_string
 - odin_util.cpp, 470
 - odin_util.h, 487
- is_dont_care_string
 - odin_util.cpp, 470
 - odin_util.h, 487
- is_dp_ram
 - memories.cpp, 256
 - memories.h, 275
- is_empty
 - hashtable_t_t, 15
 - queue_t_t, 25
- is_even_cycle
 - simulate_blif.cpp, 555
 - simulate_blif.h, 593
- is_hex_string
 - odin_util.cpp, 471
 - odin_util.h, 488
- is_node_complete
 - simulate_blif.cpp, 555
 - simulate_blif.h, 593
- is_node_ready
 - simulate_blif.cpp, 555
 - simulate_blif.h, 593
- is_octal_string
 - odin_util.cpp, 471
 - odin_util.h, 488
- is_posedge
 - simulate_blif.cpp, 556
 - simulate_blif.h, 594
- is_sp_ram
 - memories.cpp, 256
 - memories.h, 275
- is_string_of_radix
 - odin_util.cpp, 471
 - odin_util.h, 488
- is_valid_implicit_memory_reference_ast
 - implicit_memory.cpp, 234
 - implicit_memory.h, 241
- is_vector
 - simulate_blif.cpp, 556
 - simulate_blif.h, 594
- item
 - hashtable_node_t_t, 13
 - queue_node_t_t, 23
- iterate_adders
 - adders.cpp, 200
 - adders.h, 212
- iterate_adders_for_sub
 - subtractions.cpp, 297
 - subtractions.h, 301
- iterate_memories
 - memories.cpp, 257
 - memories.h, 276
- iterate_multipliers
 - multipliers.cpp, 284
 - multipliers.h, 292
- join_nets
 - netlist_utils.cpp, 385
 - netlist_utils.h, 418
- keep_all_branch
 - ast_elaborate.cpp, 71
 - ast_elaborate.h, 92
- key
 - hashtable_node_t_t, 13
- key_length
 - hashtable_node_t_t, 13

- LOGBPI
 - ace.cpp, 518
- LOGICAL_AND_string
 - node_creation_library.cpp, 437
- LOGICAL_EQUAL_string
 - node_creation_library.cpp, 437
- LOGICAL_NAND_string
 - node_creation_library.cpp, 437
- LOGICAL_NOR_string
 - node_creation_library.cpp, 437
- LOGICAL_NOT_string
 - node_creation_library.cpp, 437
- LOGICAL_OR_string
 - node_creation_library.cpp, 437
- LOGICAL_XNOR_string
 - node_creation_library.cpp, 438
- LOGICAL_XOR_string
 - node_creation_library.cpp, 438
- LOOPINIT
 - ace.cpp, 518
- LT_string
 - node_creation_library.cpp, 438
- LTE_string
 - node_creation_library.cpp, 438
- levelize_and_check_for_combinational_loop_and_↔
 - liveness
 - netlist_check.cpp, 307
 - netlist_check.h, 310
- levelize_backwards
 - netlist_check.cpp, 307
- levelize_backwards_clean_checking_for_liveness
 - netlist_check.cpp, 307
- levelize_forwards
 - netlist_check.cpp, 308
- levelize_forwards_clean_checking_for_combo_loop_↔
 - and_liveness
 - netlist_check.cpp, 308
- line
 - veri_define, 38
 - veri_include, 41
- line_has_unknown_pin
 - simulate_blif.cpp, 557
 - simulate_blif.h, 595
- line_t, 18
 - max_number_of_pins, 18
 - name, 18
 - number_of_pins, 18
 - pin_numbers, 18
 - pins, 19
 - type, 19
- lines
 - lines_t, 20
- lines_t, 19
 - count, 19
 - lines, 20
 - pin_numbers, 20
- local_clock_found
 - netlist_create_from_ast.cpp, 354
- local_clock_idx
 - netlist_create_from_ast.cpp, 354
- local_clock_list
 - netlist_create_from_ast.cpp, 354
- local_symbol_table
 - netlist_create_from_ast.cpp, 354
- local_symbol_table_sc
 - globals.h, 55
 - netlist_create_from_ast.cpp, 354
- longest_adder_chain
 - adders.cpp, 206
 - netlist_cleanup.cpp, 316
- longest_subtractor_chain
 - netlist_cleanup.cpp, 316
 - subtractions.cpp, 298
- look_for_clocks
 - netlist_create_from_ast.cpp, 349
- lookup_implicit_memory
 - implicit_memory.cpp, 235
- lookup_implicit_memory_input
 - implicit_memory.cpp, 235
 - implicit_memory.h, 241
- lookup_implicit_memory_reference_ast
 - implicit_memory.cpp, 236
 - implicit_memory.h, 242
- MAX_BUF
 - odin_util.h, 480
- MAX
 - ace.cpp, 518
- MEMORY_DEPTH_LIMIT
 - memories.h, 264
- MEMORY_string
 - node_creation_library.cpp, 438
- MINUS_string
 - node_creation_library.cpp, 438
- MODULO_string
 - node_creation_library.cpp, 439
- MULTI_PORT_MUX_string
 - node_creation_library.cpp, 439
- MULTIPLY_string
 - node_creation_library.cpp, 439
- MUX_2_string
 - node_creation_library.cpp, 439
- main
 - odin_ii.cpp, 460
- make_1port_gate
 - node_creation_library.cpp, 426
 - node_creation_library.h, 442
- make_1port_logic_gate

- node_creation_library.cpp, 427
- node_creation_library.h, 443
- make_1port_logic_gate_with_inputs
 - node_creation_library.cpp, 428
 - node_creation_library.h, 443
- make_2port_gate
 - node_creation_library.cpp, 428
 - node_creation_library.h, 444
- make_2port_logic_gates_with_inputs
 - node_creation_library.h, 444
- make_3port_gate
 - node_creation_library.cpp, 429
 - node_creation_library.h, 445
- make_concat_into_list_of_strings
 - ast_util.cpp, 113
 - ast_util.h, 129
- make_full_ref_name
 - odin_util.cpp, 472
 - odin_util.h, 489
- make_module_param_name
 - ast_util.cpp, 113
 - ast_util.h, 130
- make_mult_block
 - node_creation_library.cpp, 429
 - node_creation_library.h, 445
- make_not_gate
 - node_creation_library.cpp, 429
 - node_creation_library.h, 445
- make_not_gate_with_input
 - node_creation_library.cpp, 430
 - node_creation_library.h, 446
- make_nport_gate
 - node_creation_library.cpp, 430
 - node_creation_library.h, 447
- make_output_pins_for_existing_node
 - netlist_utils.cpp, 386
 - netlist_utils.h, 419
- make_signal_name
 - odin_util.cpp, 473
 - odin_util.h, 490
- make_simple_name
 - odin_util.cpp, 474
 - odin_util.h, 491
- make_string_based_on_id
 - odin_util.cpp, 474
 - odin_util.h, 491
- mark_clock_node
 - netlist_utils.cpp, 386
 - netlist_utils.h, 419
- mark_node_read
 - ast_elaborate.cpp, 72
 - ast_elaborate.h, 92
- mark_node_write
 - ast_elaborate.cpp, 72
- ast_elaborate.h, 93
- mark_output_dependencies
 - netlist_cleanup.cpp, 313
- markAndProcessSymbolListWith
 - parse_making_ast.cpp, 138
 - parse_making_ast.h, 162
- match_node
 - adders.cpp, 200
 - adders.h, 213
- match_pins
 - adders.cpp, 201
 - adders.h, 213
- match_ports
 - adders.cpp, 201
 - adders.h, 213
- max
 - simulate_blif.cpp, 535
- max_number_of_pins
 - line_t, 18
- Max_size
 - ast_elaborate.cpp, 60
- MaxLine
 - verilog_preprocessor.h, 509
- memories.cpp
 - add_input_port_to_memory, 245
 - add_output_port_to_memory, 245
 - check_memories_and_report_distribution, 246
 - copy_input_port_to_memory, 246
 - copy_output_port_to_memory, 247
 - create_decoder, 247
 - dp_memory_list, 262
 - dual_port_rams, 262
 - filter_memories_by_soft_logic_cutoff, 247
 - free_dp_ram_signals, 248
 - free_memory_lists, 248
 - free_sp_ram_signals, 249
 - get_dp_ram_depth, 249
 - get_dp_ram_signals, 250
 - get_dp_ram_split_depth, 250
 - get_dp_ram_split_width, 251
 - get_dp_ram_width, 251
 - get_memory_port_size, 252
 - get_sp_ram_depth, 252
 - get_sp_ram_signals, 253
 - get_sp_ram_split_depth, 253
 - get_sp_ram_split_width, 253
 - get_sp_ram_width, 254
 - instantiate_soft_dual_port_ram, 254
 - instantiate_soft_single_port_ram, 254
 - is_ast_dp_ram, 255
 - is_ast_sp_ram, 255
 - is_dp_ram, 256
 - is_sp_ram, 256
 - iterate_memories, 257

- memory_instances, 262
- memory_port_size_list, 262
- pad_dp_memory_width, 257
- pad_memory_input_port, 258
- pad_memory_output_port, 258
- pad_sp_memory_width, 259
- remap_input_port_to_memory, 259
- remap_output_port_to_memory, 260
- single_port_rams, 263
- sp_memory_list, 263
- split_dp_memory_depth, 260
- split_dp_memory_width, 260
- split_list, 263
- split_sp_memory_depth, 261
- split_sp_memory_width, 261
- memories.h
 - add_input_port_to_memory, 265
 - add_output_port_to_memory, 265
 - check_memories_and_report_distribution, 266
 - create_decoder, 266
 - dp_memory_list, 278
 - dual_port_rams, 278
 - free_dp_ram_signals, 267
 - free_memory_lists, 267
 - free_sp_ram_signals, 268
 - get_dp_ram_depth, 268
 - get_dp_ram_signals, 269
 - get_dp_ram_split_depth, 269
 - get_dp_ram_width, 270
 - get_memory_port_size, 270
 - get_sp_ram_depth, 271
 - get_sp_ram_signals, 271
 - get_sp_ram_split_depth, 272
 - get_sp_ram_width, 272
 - init_memory_distribution, 273
 - instantiate_soft_dual_port_ram, 273
 - instantiate_soft_single_port_ram, 273
 - is_ast_dp_ram, 274
 - is_ast_sp_ram, 274
 - is_dp_ram, 275
 - is_sp_ram, 275
 - iterate_memories, 276
 - MEMORY_DEPTH_LIMIT, 264
 - memory_instances, 279
 - memory_port_size_list, 279
 - single_port_rams, 279
 - sp_memory_list, 279
 - split_dp_memory_depth, 276
 - split_dp_memory_width, 277
 - split_sp_memory_depth, 277
 - split_sp_memory_width, 278
 - t_memory, 265
 - t_memory_port_sizes, 265
- memory_instances
 - memories.cpp, 262
 - memories.h, 279
- memory_port_size_list
 - memories.cpp, 262
 - memories.h, 279
- merge_nodes
 - adders.cpp, 201
 - adders.h, 214
- min
 - simulate_blif.cpp, 535
- min_add
 - adders.cpp, 206
 - adders.h, 218
- min_mult
 - multipliers.cpp, 288
 - multipliers.h, 294
- min_threshold_adder
 - adders.cpp, 207
 - adders.h, 218
- mod
 - STRING_CACHE, 35
- models
 - hard_block_models, 10
- module_instantiations_instance
 - parse_making_ast.cpp, 155
- module_names_to_idx
 - globals.h, 55
 - parse_making_ast.cpp, 155
- module_variables_not_defined
 - parse_making_ast.cpp, 155
- modules_inputs_sc
 - parse_making_ast.cpp, 155
- modules_outputs_sc
 - parse_making_ast.cpp, 155
- move_ast_node
 - ast_util.cpp, 114
 - ast_util.h, 130
- move_input_pin
 - netlist_utils.cpp, 387
 - netlist_utils.h, 420
- move_output_pin
 - netlist_utils.cpp, 387
 - netlist_utils.h, 420
- mul
 - STRING_CACHE, 35
- mult_list
 - multipliers.cpp, 288
 - multipliers.h, 294
- multipliers.cpp
 - add_the_blackbox_for_mults, 280
 - clean_multipliers, 280
 - declare_hard_multiplier, 281
 - define_mult_function, 281
 - find_hard_multipliers, 282

- hard_multipliers, 288
- init_cascade_adder, 282
- init_mult_distribution, 283
- init_split_multiplier, 283
- instantiate_hard_multiplier, 284
- instantiate_simple_soft_multiplier, 284
- iterate_multipliers, 284
- min_mult, 288
- mult_list, 288
- mults, 288
- pad_multiplier, 285
- record_mult_distribution, 285
- report_mult_distribution, 286
- split_multiplier, 286
- split_multiplier_a, 287
- split_multiplier_b, 287
- multipliers.h
 - add_the_blackbox_for_mults, 289
 - clean_multipliers, 290
 - declare_hard_multiplier, 290
 - define_mult_function, 290
 - find_hard_multipliers, 291
 - hard_multipliers, 294
 - init_mult_distribution, 291
 - instantiate_hard_multiplier, 292
 - instantiate_simple_soft_multiplier, 292
 - iterate_multipliers, 292
 - min_mult, 294
 - mult_list, 294
 - report_mult_distribution, 293
 - split_multiplier, 293
 - t_multiplier, 289
- multiply_arrays
 - simulate_blif.cpp, 557
 - simulate_blif.h, 595
- mults
 - multipliers.cpp, 288
- my_malloc_struct
 - odin_util.cpp, 474
 - odin_util.h, 491
- my_power
 - odin_util.cpp, 476
 - odin_util.h, 493
- N
 - ast_elaborate.cpp, 61
- NOT_EQUAL_string
 - node_creation_library.cpp, 439
- name
 - hard_block_model, 9
 - implicit_memory, 17
 - line_t, 18
 - s_memory_port_sizes, 29
- names
 - hard_block_pins, 11
 - hard_block_ports, 12
- netlist_check.cpp
 - check_netlist, 304
 - depth_first_traversal_check_if_forward_leveled, 305
 - depth_first_traverse_check_if_forward_leveled, 305
 - depth_first_traverse_until_next_ff_or_output, 306
 - depth_traverse_check_combinational_loop, 306
 - find_node_at_top_of_combo_loop, 306
 - levelize_and_check_for_combinational_loop_and_↔
liveness, 307
 - levelize_backwards, 307
 - levelize_backwards_clean_checking_for_liveness,
307
 - levelize_forwards, 308
 - levelize_forwards_clean_checking_for_combo_↔
loop_and_liveness, 308
 - sequential_levelized_dfs, 309
- netlist_check.h
 - check_netlist, 310
 - levelize_and_check_for_combinational_loop_and_↔
liveness, 310
- netlist_cleanup.cpp
 - _visited_backward, 315
 - _visited_forward, 315
 - _visited_removal, 315
 - adder_chain_count, 315
 - addsub_list_next, 316
 - addsub_nodes, 316
 - calculate_addsub_statistics, 312
 - geomean_addsub_length, 316
 - identify_unused_nodes, 312
 - insert_node_list, 313
 - longest_adder_chain, 316
 - longest_subtractor_chain, 316
 - mark_output_dependencies, 313
 - node_list_t, 312
 - removal_list_next, 316
 - remove_unused_logic, 313
 - remove_unused_nodes, 314
 - subtractor_chain_count, 317
 - sum_of_addsub_logs, 317
 - total_adders, 317
 - total_addsub_chain_count, 317
 - total_subtractors, 317
 - traverse_backward, 314
 - traverse_forward, 314
 - useless_nodes, 317
 - VISITED_BACKWARD, 311
 - VISITED_FORWARD, 312
 - VISITED_REMOVAL, 312
- netlist_cleanup.h
 - remove_unused_logic, 318
- netlist_create_from_ast.cpp

ALIAS_INPUTS, 321
 alias_output_assign_pins_to_inputs, 321
 assignment_alias, 322
 COMBINATIONAL, 321
 check_for_initial_reg_value, 323
 concatenate_signal_lists, 323
 connect_function_instantiation_and_alias, 324
 connect_hard_block_and_alias, 325
 connect_memory_and_alias, 326
 connect_module_instantiation_and_alias, 327
 convert_ast_to_netlist_recurring_via_modules, 328
 create_all_driver_nets_in_this_function, 328
 create_all_driver_nets_in_this_module, 329
 create_case, 330
 create_case_control_signals, 331
 create_case_mux_statements, 332
 create_dual_port_ram_block, 332
 create_gate, 333
 create_hard_block, 333
 create_if, 334
 create_if_control_signals, 335
 create_if_for_question, 336
 create_if_mux_statements, 336
 create_if_question_mux_expressions, 337
 create_mux_expressions, 337
 create_mux_statements, 338
 create_netlist, 338
 create_operation_node, 339
 create_output_pin, 339
 create_param_table_for_module, 340
 create_pins, 340
 create_single_port_ram_block, 341
 create_soft_dual_port_ram_block, 342
 create_soft_single_port_ram_block, 342
 create_symbol_table_for_function, 343
 create_symbol_table_for_module, 344
 create_top_driver_nets, 345
 create_top_output_nodes, 345
 define_latches_initial_value_inside_initial_statement, 346
 define_nets_with_driver, 346
 define_nodes_and_nets_with_driver, 347
 evaluate_sensitivity_list, 347
 find_smallest_non_numerical, 348
 find_top_module, 349
 function_local_symbol_table, 353
 function_local_symbol_table_sc, 353
 function_num_local_symbol_table, 353
 global_param_table_sc, 353
 INSTANTIATE_DRIVERS, 321
 input_nets_sc, 354
 local_clock_found, 354
 local_clock_idx, 354
 local_clock_list, 354
 local_symbol_table, 354
 local_symbol_table_sc, 354
 look_for_clocks, 349
 netlist_create_line_number, 355
 netlist_expand_ast_of_module, 350
 num_local_symbol_table, 355
 one_string, 355
 output_nets_sc, 355
 pad_string, 355
 pad_with_zeros, 350
 SEQUENTIAL, 321
 terminate_continuous_assignment, 351
 terminate_registered_assignment, 352
 top_module, 355
 type_of_circuit, 356
 verilog_netlist, 356
 zero_string, 356
 netlist_create_from_ast.h
 connect_memory_and_alias, 356
 create_netlist, 357
 netlist_create_line_number
 netlist_create_from_ast.cpp, 355
 netlist_expand_ast_of_module
 netlist_create_from_ast.cpp, 350
 netlist_utils.cpp
 add_driver_pin_to_net, 359
 add_fanout_pin_to_net, 360
 add_input_pin_to_node, 361
 add_input_port_information, 363
 add_node_to_netlist, 363
 add_output_pin_to_node, 363
 add_output_port_information, 364
 add_pin_to_signal_list, 365
 allocate_chain_info, 366
 allocate_more_input_pins, 367
 allocate_more_output_pins, 367
 allocate_netlist, 368
 allocate_nnet, 369
 allocate_nnode, 370
 allocate_npin, 371
 combine_lists, 372
 combine_lists_without_freeing_originals, 373
 combine_nets, 374
 connect_nodes, 374
 copy_input_npin, 375
 copy_input_signals, 376
 copy_output_npin, 376
 copy_output_signals, 377
 count_nodes_in_netlist, 377
 depth_traverse_count, 377
 free_netlist, 378
 free_nnode, 378
 free_signal_list, 379
 get_input_pin_index_from_mapping, 380

- get_input_port_index_from_mapping, 380
- get_output_pin_index_from_mapping, 381
- get_output_port_index_from_mapping, 381
- global_args, 389
- hookup_hb_input_pins_from_signal_list, 382
- hookup_input_pins_from_signal_list, 382
- hookup_output_pins_from_signal_list, 383
- init_signal_list, 384
- join_nets, 385
- make_output_pins_for_existing_node, 386
- mark_clock_node, 386
- move_input_pin, 387
- move_output_pin, 387
- remap_pin_to_new_net, 387
- remap_pin_to_new_node, 388
- remove_fanout_pins_from_net, 388
- sort_signal_list_alphabetically, 389
- netlist_utils.h
 - add_driver_pin_to_net, 391
 - add_fanout_pin_to_net, 392
 - add_input_pin_to_node, 393
 - add_input_port_information, 395
 - add_node_to_netlist, 395
 - add_output_pin_to_node, 395
 - add_output_port_information, 396
 - add_pin_to_signal_list, 397
 - allocate_chain_info, 398
 - allocate_more_input_pins, 399
 - allocate_more_output_pins, 399
 - allocate_netlist, 400
 - allocate_nnet, 401
 - allocate_nnode, 402
 - allocate_npin, 403
 - combine_lists, 404
 - combine_lists_without_freeing_originals, 405
 - combine_nets, 406
 - connect_nodes, 406
 - copy_input_npin, 407
 - copy_input_signals, 408
 - copy_output_npin, 408
 - copy_output_signals, 409
 - count_nodes_in_netlist, 409
 - free_netlist, 409
 - free_nnode, 410
 - free_npin, 410
 - free_signal_list, 410
 - get_input_pin_index_from_mapping, 411
 - get_input_port_index_from_mapping, 412
 - get_one_pin, 412
 - get_output_pin_index_from_mapping, 413
 - get_output_port_index_from_mapping, 413
 - get_pad_pin, 414
 - get_zero_pin, 414
 - hookup_hb_input_pins_from_signal_list, 415
 - hookup_input_pins_from_signal_list, 416
 - hookup_output_pins_from_signal_list, 416
 - init_signal_list, 417
 - join_nets, 418
 - make_output_pins_for_existing_node, 419
 - mark_clock_node, 419
 - move_input_pin, 420
 - move_output_pin, 420
 - remap_pin_to_new_net, 420
 - remap_pin_to_new_node, 421
 - remove_fanout_pins_from_net, 421
 - sort_signal_list_alphabetically, 422
- netlist_visualizer.c
 - backward_traversal_net_graph_display, 43
 - depth_first_traversal_graph_display, 43
 - depth_first_traverse_visualize, 44
 - forward_traversal_net_graph_display, 44
 - graphVizOutputCombinationalNet, 45
 - graphVizOutputNetlist, 45
- netlist_visualizer.cpp
 - backward_traversal_net_graph_display, 46
 - depth_first_traversal_graph_display, 47
 - depth_first_traverse_visualize, 47
 - forward_traversal_net_graph_display, 48
 - graphVizOutputCombinationalNet, 48
 - graphVizOutputNetlist, 48
- netlist_visualizer.h
 - graphVizOutputCombinationalNet, 49
 - graphVizOutputNetlist, 49
- newAlways
 - parse_making_ast.cpp, 139
 - parse_making_ast.h, 163
- newArrayRef
 - parse_making_ast.cpp, 139
 - parse_making_ast.h, 163
- newAssign
 - parse_making_ast.h, 163
- newBinaryOperation
 - parse_making_ast.cpp, 139
 - parse_making_ast.h, 163
- newBlocking
 - parse_making_ast.cpp, 139
 - parse_making_ast.h, 164
- newCase
 - parse_making_ast.cpp, 140
 - parse_making_ast.h, 164
- newCaseItem
 - parse_making_ast.cpp, 140
 - parse_making_ast.h, 164
- newConstant
 - parse_making_ast.cpp, 140
 - parse_making_ast.h, 164
- newDefaultCase
 - parse_making_ast.cpp, 140

parse_making_ast.h, 164
 newDefparam
 parse_making_ast.cpp, 140
 parse_making_ast.h, 165
 newExpandPower
 parse_making_ast.cpp, 141
 parse_making_ast.h, 165
 newFor
 parse_making_ast.cpp, 141
 parse_making_ast.h, 165
 newFunction
 parse_making_ast.cpp, 141
 parse_making_ast.h, 165
 newFunctionAssigning
 parse_making_ast.cpp, 141
 newFunctionInstance
 parse_making_ast.cpp, 142
 parse_making_ast.h, 165
 newFunctionNamedInstance
 parse_making_ast.cpp, 142
 parse_making_ast.h, 166
 newGate
 parse_making_ast.cpp, 142
 parse_making_ast.h, 166
 newGateInstance
 parse_making_ast.cpp, 142
 parse_making_ast.h, 166
 newHardBlockInstance
 parse_making_ast.cpp, 143
 newIf
 parse_making_ast.cpp, 143
 parse_making_ast.h, 166
 newIfQuestion
 parse_making_ast.cpp, 143
 parse_making_ast.h, 167
 newInitial
 parse_making_ast.cpp, 144
 parse_making_ast.h, 167
 newIntegerTypeVarDeclare
 parse_making_ast.cpp, 144
 parse_making_ast.h, 167
 newList
 parse_making_ast.cpp, 144
 parse_making_ast.h, 167
 newList_entry
 parse_making_ast.cpp, 145
 parse_making_ast.h, 168
 newListReplicate
 parse_making_ast.cpp, 145
 parse_making_ast.h, 168
 newModule
 parse_making_ast.cpp, 145
 parse_making_ast.h, 169
 newModuleConnection
 parse_making_ast.cpp, 146
 parse_making_ast.h, 169
 newModuleInstance
 parse_making_ast.cpp, 146
 parse_making_ast.h, 169
 newModuleNamedInstance
 parse_making_ast.cpp, 146
 parse_making_ast.h, 169
 newModuleParameter
 parse_making_ast.cpp, 147
 parse_making_ast.h, 170
 newMultipleInputsGateInstance
 parse_making_ast.cpp, 147
 parse_making_ast.h, 170
 newNegedgeSymbol
 parse_making_ast.cpp, 147
 parse_making_ast.h, 170
 newNonBlocking
 parse_making_ast.cpp, 147
 parse_making_ast.h, 170
 newNumberNode
 parse_making_ast.cpp, 147
 parse_making_ast.h, 170
 newParallelConnection
 parse_making_ast.cpp, 148
 parse_making_ast.h, 171
 newPosedgeSymbol
 parse_making_ast.cpp, 148
 parse_making_ast.h, 171
 newRangeRef
 parse_making_ast.cpp, 148
 parse_making_ast.h, 171
 newSymbolNode
 parse_making_ast.cpp, 149
 parse_making_ast.h, 172
 newUnaryOperation
 parse_making_ast.cpp, 150
 parse_making_ast.h, 173
 newVarDeclare
 parse_making_ast.cpp, 151
 parse_making_ast.h, 174
 newWhile
 parse_making_ast.cpp, 151
 parse_making_ast.h, 174
 newfunctionList
 parse_making_ast.cpp, 142
 parse_making_ast.h, 166
 next
 exp_node, 7
 hashtable_node_t_t, 13
 node_list_t_t, 20
 queue_node_t_t, 23
 s_adder, 26
 s_memory, 27

- s_multiplier, 30
 - veri_flag_node, 40
- next_function
 - parse_making_ast.cpp, 151
 - parse_making_ast.h, 174
- next_module
 - parse_making_ast.cpp, 152
 - parse_making_ast.h, 175
- next_parsed_verilog_file
 - parse_making_ast.cpp, 152
 - parse_making_ast.h, 175
- next_string
 - STRING_CACHE, 35
- node
 - implicit_memory, 17
 - node_list_t_t, 20
- node_creation_library.cpp
 - ADD_string, 433
 - ADDER_FUNC_string, 433
 - BITWISE_AND_string, 433
 - BITWISE_NAND_string, 434
 - BITWISE_NOR_string, 434
 - BITWISE_NOT_string, 434
 - BITWISE_OR_string, 434
 - BITWISE_XNOR_string, 434
 - BITWISE_XOR_string, 434
 - BUF_NODE_string, 435
 - CARRY_FUNC_string, 435
 - CASE_EQUAL_string, 435
 - CASE_NOT_EQUAL_string, 435
 - CLOCK_NODE_string, 435
 - DIVIDE_string, 435
 - FF_NODE_string, 436
 - GND_NODE_string, 436
 - GT_string, 436
 - GTE_string, 436
 - get_one_pin, 424
 - get_pad_pin, 424
 - get_zero_pin, 425
 - HARD_IP_string, 436
 - hard_node_name, 426
 - INPUT_NODE_string, 436
 - LOGICAL_AND_string, 437
 - LOGICAL_EQUAL_string, 437
 - LOGICAL_NAND_string, 437
 - LOGICAL_NOR_string, 437
 - LOGICAL_NOT_string, 437
 - LOGICAL_OR_string, 437
 - LOGICAL_XNOR_string, 438
 - LOGICAL_XOR_string, 438
 - LT_string, 438
 - LTE_string, 438
 - MEMORY_string, 438
 - MINUS_string, 438
 - MODULO_string, 439
 - MULTI_PORT_MUX_string, 439
 - MULTIPLY_string, 439
 - MUX_2_string, 439
 - make_1port_gate, 426
 - make_1port_logic_gate, 427
 - make_1port_logic_gate_with_inputs, 428
 - make_2port_gate, 428
 - make_3port_gate, 429
 - make_mult_block, 429
 - make_not_gate, 429
 - make_not_gate_with_input, 430
 - make_nport_gate, 430
 - NOT_EQUAL_string, 439
 - node_name, 431
 - node_name_based_on_op, 432
 - OUTPUT_NODE_string, 439
 - SL_string, 440
 - SR_string, 440
 - unique_node_name_id, 440
 - VCC_NODE_string, 440
- node_creation_library.h
 - get_one_pin, 441
 - get_zero_pin, 441
 - hard_node_name, 441
 - make_1port_gate, 442
 - make_1port_logic_gate, 443
 - make_1port_logic_gate_with_inputs, 443
 - make_2port_gate, 444
 - make_2port_logic_gates_with_inputs, 444
 - make_3port_gate, 445
 - make_mult_block, 445
 - make_not_gate, 445
 - make_not_gate_with_input, 446
 - make_nport_gate, 447
 - node_name, 447
 - node_name_based_on_op, 448
- node_error
 - ace.cpp, 525
- node_get_literal
 - ace.cpp, 518
- node_is_constant
 - ast_util.cpp, 114
 - ast_util.h, 131
- node_list_t
 - netlist_cleanup.cpp, 312
- node_list_t_t, 20
 - next, 20
 - node, 20
- node_name
 - node_creation_library.cpp, 431
 - node_creation_library.h, 447
- node_name_based_on_op
 - node_creation_library.cpp, 432

- node_creation_library.h, 448
- num_children
 - stages_t, 33
- num_connections
 - stages_t, 33
- num_functions
 - parse_making_ast.cpp, 156
- num_literals
 - ace_cube_t, 2
- num_local_symbol_table
 - netlist_create_from_ast.cpp, 355
- num_modules
 - globals.h, 55
 - parse_making_ast.cpp, 156
- num_nodes
 - stages_t, 33
- num_parallel_nodes
 - stages_t, 33
- num_top_input_nodes
 - globals.h, 55
- num_top_output_nodes
 - globals.h, 55
- number_of_pins
 - line_t, 18
- ONE
 - ace.cpp, 519
- OUTPUT_ACTIVITY_FILE_NAME
 - simulate_blif.h, 573
- OUTPUT_NODE_string
 - node_creation_library.cpp, 439
- OUTPUT_VECTOR_FILE_NAME
 - simulate_blif.h, 573
- odin_ii.cpp
 - Arch, 461
 - block_tag, 461
 - current_parse_file, 461
 - get_options, 459
 - global_args, 461
 - main, 460
 - print_usage, 460
 - set_default_config, 460
 - type_descriptors, 461
- odin_util.cpp
 - append_string, 463
 - convert_binary_string_of_size_to_bit_string, 463
 - convert_dec_string_of_size_to_long_long, 464
 - convert_hex_string_of_size_to_bit_string, 464
 - convert_long_long_to_bit_string, 464
 - convert_oct_string_of_size_to_bit_string, 465
 - convert_string_of_radix_to_bit_string, 465
 - convert_string_of_radix_to_long_long, 466
 - error_message, 466
 - find_substring, 467
 - get_bit, 468
 - get_pin_name, 468
 - get_pin_number, 469
 - get_port_name, 469
 - is_binary_string, 470
 - is_decimal_string, 470
 - is_dont_care_string, 470
 - is_hex_string, 471
 - is_octal_string, 471
 - is_string_of_radix, 471
 - make_full_ref_name, 472
 - make_signal_name, 473
 - make_simple_name, 474
 - make_string_based_on_id, 474
 - my_malloc_struct, 474
 - my_power, 476
 - pow2, 476
 - reverse_string, 476
 - search_replace, 476
 - string_to_lower, 477
 - string_to_upper, 477
 - twos_complement, 477
 - validate_string_regex, 478
 - warning_message, 478
- odin_util.h
 - append_string, 481
 - convert_binary_string_of_size_to_bit_string, 481
 - convert_dec_string_of_size_to_long_long, 481
 - convert_hex_string_of_size_to_bit_string, 482
 - convert_long_long_to_bit_string, 482
 - convert_oct_string_of_size_to_bit_string, 483
 - convert_string_of_radix_to_bit_string, 483
 - convert_string_of_radix_to_long_long, 484
 - find_substring, 484
 - get_bit, 484
 - get_pin_name, 485
 - get_pin_number, 486
 - get_port_name, 486
 - is_binary_string, 487
 - is_decimal_string, 487
 - is_dont_care_string, 487
 - is_hex_string, 488
 - is_octal_string, 488
 - is_string_of_radix, 488
 - MAX_BUF, 480
 - make_full_ref_name, 489
 - make_signal_name, 490
 - make_simple_name, 491
 - make_string_based_on_id, 491
 - my_malloc_struct, 491
 - my_power, 493
 - pow2, 493
 - reverse_string, 493
 - search_replace, 493

- string_to_lower, [494](#)
- string_to_upper, [494](#)
- twos_complement, [494](#)
- validate_string_regex, [495](#)
- one_net
 - globals.h, [55](#)
- one_string
 - globals.h, [55](#)
 - netlist_create_from_ast.cpp, [355](#)
- open_source_file
 - verilog_preprocessor.cpp, [504](#)
- operation
 - exp_node, [7](#)
- optimize_for_tree
 - ast_elaborate.cpp, [72](#)
 - ast_elaborate.h, [93](#)
- out
 - sp_ram_signals, [31](#)
- out1
 - dp_ram_signals, [5](#)
- out2
 - dp_ram_signals, [5](#)
- output_ace_info
 - ace.cpp, [525](#)
- output_ace_info_node
 - ace.cpp, [526](#)
- output_added
 - implicit_memory, [17](#)
- output_blif
 - output_blif.cpp, [179](#)
 - output_blif.h, [181](#)
- output_blif.cpp
 - define_decoded_mux, [177](#)
 - define_ff, [177](#)
 - define_logical_function, [177](#)
 - define_set_input_logical_function, [178](#)
 - depth_first_traversal_to_output, [178](#)
 - depth_traverse_output_blif, [179](#)
 - output_blif, [179](#)
 - output_blif_pin_connect, [180](#)
 - output_node, [180](#)
- output_blif.h
 - output_blif, [181](#)
- output_blif_pin_connect
 - output_blif.cpp, [180](#)
- output_hard_blocks
 - hard_blocks.cpp, [223](#)
 - hard_blocks.h, [228](#)
- output_nets_sc
 - globals.h, [56](#)
 - netlist_create_from_ast.cpp, [355](#)
- output_node
 - output_blif.cpp, [180](#)
- output_ports
 - hard_block_model, [9](#)
- outputs
 - hard_block_model, [9](#)
- p
 - ast_elaborate.cpp, [80](#)
- pad_dp_memory_width
 - memories.cpp, [257](#)
- pad_memory_input_port
 - memories.cpp, [258](#)
- pad_memory_output_port
 - memories.cpp, [258](#)
- pad_multiplier
 - multipliers.cpp, [285](#)
- pad_net
 - globals.h, [56](#)
- pad_node
 - globals.h, [56](#)
- pad_sp_memory_width
 - memories.cpp, [259](#)
- pad_string
 - globals.h, [56](#)
 - netlist_create_from_ast.cpp, [355](#)
- pad_with_zeros
 - netlist_create_from_ast.cpp, [350](#)
- parallel_times
 - stages_t, [33](#)
- parse_making_ast.cpp
 - all_file_items_list, [153](#)
 - ast_functions, [153](#)
 - ast_modules, [153](#)
 - block_instantiations_instance, [153](#)
 - calculate_operation, [135](#)
 - clean_up_parser_for_file, [135](#)
 - cleanup_hard_blocks, [135](#)
 - cleanup_parser, [136](#)
 - defines_for_file_sc, [154](#)
 - defines_for_function_sc, [154](#)
 - defines_for_module_sc, [154](#)
 - function_instantiations_instance, [154](#)
 - function_instantiations_instance_by_module, [154](#)
 - functions_inputs_sc, [154](#)
 - functions_outputs_sc, [155](#)
 - graphVizOutputAst, [136](#)
 - graphVizOutputAst_traverse_node, [137](#)
 - graphVizOutputPreproc, [137](#)
 - init_parser, [137](#)
 - init_parser_for_file, [138](#)
 - markAndProcessSymbolListWith, [138](#)
 - module_instantiations_instance, [155](#)
 - module_names_to_idx, [155](#)
 - module_variables_not_defined, [155](#)
 - modules_inputs_sc, [155](#)
 - modules_outputs_sc, [155](#)

- [newAlways](#), 139
- [newArrayRef](#), 139
- [newBinaryOperation](#), 139
- [newBlocking](#), 139
- [newCase](#), 140
- [newCaseItem](#), 140
- [newConstant](#), 140
- [newDefaultCase](#), 140
- [newDefparam](#), 140
- [newExpandPower](#), 141
- [newFor](#), 141
- [newFunction](#), 141
- [newFunctionAssigning](#), 141
- [newFunctionInstance](#), 142
- [newFunctionNamedInstance](#), 142
- [newGate](#), 142
- [newGateInstance](#), 142
- [newHardBlockInstance](#), 143
- [newIf](#), 143
- [newIfQuestion](#), 143
- [newInitial](#), 144
- [newIntegerTypeVarDeclare](#), 144
- [newList](#), 144
- [newList_entry](#), 145
- [newListReplicate](#), 145
- [newModule](#), 145
- [newModuleConnection](#), 146
- [newModuleInstance](#), 146
- [newModuleNamedInstance](#), 146
- [newModuleParameter](#), 147
- [newMultipleInputsGateInstance](#), 147
- [newNegedgeSymbol](#), 147
- [newNonBlocking](#), 147
- [newNumberNode](#), 147
- [newParallelConnection](#), 148
- [newPosedgeSymbol](#), 148
- [newRangeRef](#), 148
- [newSymbolNode](#), 149
- [newUnaryOperation](#), 150
- [newVarDeclare](#), 151
- [newWhile](#), 151
- [newfunctionList](#), 142
- [next_function](#), 151
- [next_module](#), 152
- [next_parsed_verilog_file](#), 152
- [num_functions](#), 156
- [num_modules](#), 156
- [parse_to_ast](#), 152
- [size_all_file_items_list](#), 156
- [size_block_instantiations](#), 156
- [size_function_instantiations](#), 156
- [size_function_instantiations_by_module](#), 156
- [size_module_instantiations](#), 157
- [size_module_variables_not_defined](#), 157
- [to_view_parse](#), 157
- [unique_label_count](#), 157
- [yylineno](#), 157
- [parse_making_ast.h](#)
 - [calculate_operation](#), 159
 - [clean_up_parser_for_file](#), 159
 - [cleanup_hard_blocks](#), 160
 - [cleanup_parser](#), 160
 - [graphVizOutputAst](#), 160
 - [graphVizOutputAst_traverse_node](#), 161
 - [init_parser](#), 161
 - [init_parser_for_file](#), 162
 - [markAndProcessSymbolListWith](#), 162
 - [newAlways](#), 163
 - [newArrayRef](#), 163
 - [newAssign](#), 163
 - [newBinaryOperation](#), 163
 - [newBlocking](#), 164
 - [newCase](#), 164
 - [newCaseItem](#), 164
 - [newConstant](#), 164
 - [newDefaultCase](#), 164
 - [newDefparam](#), 165
 - [newExpandPower](#), 165
 - [newFor](#), 165
 - [newFunction](#), 165
 - [newFunctionInstance](#), 165
 - [newFunctionNamedInstance](#), 166
 - [newGate](#), 166
 - [newGateInstance](#), 166
 - [newIf](#), 166
 - [newIfQuestion](#), 167
 - [newInitial](#), 167
 - [newIntegerTypeVarDeclare](#), 167
 - [newList](#), 167
 - [newList_entry](#), 168
 - [newListReplicate](#), 168
 - [newModule](#), 169
 - [newModuleConnection](#), 169
 - [newModuleInstance](#), 169
 - [newModuleNamedInstance](#), 169
 - [newModuleParameter](#), 170
 - [newMultipleInputsGateInstance](#), 170
 - [newNegedgeSymbol](#), 170
 - [newNonBlocking](#), 170
 - [newNumberNode](#), 170
 - [newParallelConnection](#), 171
 - [newPosedgeSymbol](#), 171
 - [newRangeRef](#), 171
 - [newSymbolNode](#), 172
 - [newUnaryOperation](#), 173
 - [newVarDeclare](#), 174
 - [newWhile](#), 174
 - [newfunctionList](#), 166

- next_function, 174
- next_module, 175
- next_parsed_verilog_file, 175
- parse_to_ast, 175
- parse_mif_radix
 - simulate_blif.cpp, 557
 - simulate_blif.h, 595
- parse_pin_name_list
 - simulate_blif.cpp, 558
 - simulate_blif.h, 596
- parse_test_vector
 - simulate_blif.cpp, 558
 - simulate_blif.h, 596
- parse_to_ast
 - parse_making_ast.cpp, 152
 - parse_making_ast.h, 175
- ParseInitRegState, 21
 - default_choices, 21
 - from_str, 21
 - to_str, 22
- partial_map.cpp
 - depth_first_traversal_to_partial_map, 450
 - depth_first_traverse_partial_map, 450
 - instantiate_EQUAL, 452
 - instantiate_GE, 453
 - instantiate_GT, 453
 - instantiate_add_w_carry, 451
 - instantiate_bitwise_logic, 451
 - instantiate_bitwise_reduction, 452
 - instantiate_buffer, 452
 - instantiate_logical_logic, 454
 - instantiate_multi_port_mux, 454
 - instantiate_not_logic, 454
 - instantiate_shift_left_or_right, 455
 - instantiate_soft_logic_ram, 455
 - instantiate_sub_w_carry, 455
 - instantiate_unary_sub, 456
 - partial_map_node, 456
 - partial_map_top, 456
- partial_map.h
 - instantiate_add_w_carry, 457
 - instantiate_multi_port_mux, 457
 - partial_map_top, 458
- partial_map_node
 - partial_map.cpp, 456
- partial_map_top
 - partial_map.cpp, 456
 - partial_map.h, 458
- path
 - veri_include, 41
- pin_names, 22
 - count, 22
 - pins, 22
- pin_numbers
 - line_t, 18
 - lines_t, 20
- pins
 - line_t, 19
 - pin_names, 22
- pop
 - verilog_preprocessor.cpp, 505
 - verilog_preprocessor.h, 512
- pow2
 - odin_util.cpp, 476
 - odin_util.h, 493
- pre
 - exp_node, 7
- preprocess_mif_file
 - simulate_blif.cpp, 558
 - simulate_blif.h, 596
- print_ancestry
 - simulate_blif.cpp, 559
 - simulate_blif.h, 597
- print_netlist.c
 - function_to_print_node_and_its_pin, 50
 - print_netlist_for_checking, 50
- print_netlist.cpp
 - function_to_print_node_and_its_pin, 51
 - print_netlist_for_checking, 51
- print_netlist.h
 - print_netlist_for_checking, 52
- print_netlist_for_checking
 - print_netlist.c, 50
 - print_netlist.cpp, 51
 - print_netlist.h, 52
- print_netlist_stats
 - simulate_blif.cpp, 559
 - simulate_blif.h, 597
- print_progress_bar
 - simulate_blif.cpp, 559
 - simulate_blif.h, 597
- print_simulation_stats
 - simulate_blif.cpp, 560
 - simulate_blif.h, 598
- print_time
 - simulate_blif.cpp, 560
 - simulate_blif.h, 598
- print_update_trace
 - simulate_blif.cpp, 561
 - simulate_blif.h, 599
- print_usage
 - odin_ii.cpp, 460
- priority
 - exp_node, 7
- prob_epsilon_fix
 - ace.cpp, 526
- processed_adder_list
 - adders.cpp, 207

- adders.h, 218
- pset
 - ace.cpp, 520
- push
 - verilog_preprocessor.cpp, 505
 - verilog_preprocessor.h, 512
- queue.cpp
 - __queue_add, 528
 - __queue_destroy, 529
 - __queue_is_empty, 529
 - __queue_remove, 530
 - __queue_remove_all, 530
 - create_queue, 531
- queue.h
 - create_queue, 532
 - queue_node_t, 532
 - queue_t, 532
- queue_node_t
 - queue.h, 532
- queue_node_t_t, 23
 - item, 23
 - next, 23
- queue_t
 - queue.h, 532
- queue_t_t, 24
 - add, 24
 - count, 24
 - destroy, 24
 - head, 24
 - is_empty, 25
 - remove, 25
 - remove_all, 25
 - tail, 25
- READ_BLIF_BUFFER
 - read_blif.cpp, 183
- rb_create_top_driver_nets
 - read_blif.cpp, 190
- rb_create_top_output_nodes
 - read_blif.cpp, 191
- rb_look_for_clocks
 - read_blif.cpp, 191
- read_bit_map_find_unknown_gate
 - read_blif.cpp, 191
- read_blif
 - read_blif.cpp, 192
 - read_blif.h, 195
- read_blif.cpp
 - add_hard_block_model, 184
 - add_top_input_nodes, 184
 - assign_node_type_from_node_name, 184
 - associate_names, 184
 - BLIF_ONE_STRING, 194
 - BLIF_PAD_STRING, 194
 - BLIF_ZERO_STRING, 194
 - count_blif_lines, 185
 - create_hard_block_models, 185
 - create_hard_block_nodes, 186
 - create_internal_node_and_driver, 186
 - create_latch_node_and_driver, 186
 - DEFAULT_CLOCK_NAME, 194
 - file_line_number, 194
 - free_hard_block_model, 186
 - free_hard_block_models, 187
 - free_hard_block_pins, 187
 - free_hard_block_ports, 187
 - GND_NAME, 183
 - generate_hard_block_ports_signature, 188
 - get_hard_block_model, 188
 - get_hard_block_pin_number, 188
 - get_hard_block_port_name, 189
 - get_hard_block_ports, 189
 - HBPAD_NAME, 183
 - hook_up_nets, 189
 - hook_up_node, 189
 - index_names, 190
 - READ_BLIF_BUFFER, 183
 - rb_create_top_driver_nets, 190
 - rb_create_top_output_nodes, 191
 - rb_look_for_clocks, 191
 - read_bit_map_find_unknown_gate, 191
 - read_blif, 192
 - read_hard_block_model, 192
 - read_tokens, 192
 - search_clock_name, 193
 - TOKENS, 183
 - VCC_NAME, 184
 - verify_hard_block_ports_against_model, 193
- read_blif.h
 - read_blif, 195
- read_blif_netlist
 - globals.h, 56
- read_config_file
 - read_xml_config_file.cpp, 496
 - read_xml_config_file.h, 500
- read_debug_switches
 - read_xml_config_file.cpp, 496
- read_hard_block_model
 - read_blif.cpp, 192
- read_node
 - ast_elaborate.cpp, 61
- read_optimizations
 - read_xml_config_file.cpp, 497
- read_outputs
 - read_xml_config_file.cpp, 497
- read_tokens
 - read_blif.cpp, 192
- read_verilog_files

- read_xml_config_file.cpp, 498
- read_xml_config_file.cpp
 - configuration, 499
 - empty_string, 499
 - read_config_file, 496
 - read_debug_switches, 496
 - read_optimizations, 497
 - read_outputs, 497
 - read_verilog_files, 498
 - set_default_optimization_settings, 498
- read_xml_config_file.h
 - read_config_file, 500
- reallocate_node
 - ast_elaborate.cpp, 73
 - ast_elaborate.h, 93
- reallocate_pins
 - adders.cpp, 202
 - adders.h, 214
- record_add_distribution
 - adders.cpp, 202
- record_expression
 - ast_elaborate.cpp, 73
 - ast_elaborate.h, 94
- record_mult_distribution
 - multipliers.cpp, 285
- record_tree_info
 - ast_elaborate.cpp, 73
 - ast_elaborate.h, 94
- recursive_tree
 - ast_elaborate.cpp, 74
 - ast_elaborate.h, 94
- reduce_assignment_expression
 - ast_elaborate.cpp, 74
 - ast_elaborate.h, 95
- reduce_enode_list
 - ast_elaborate.cpp, 74
 - ast_elaborate.h, 95
- reduce_operations
 - adders.cpp, 202
 - adders.h, 214
- reduce_parameter
 - ast_elaborate.cpp, 75
 - ast_elaborate.h, 95
- register_hard_blocks
 - hard_blocks.cpp, 224
 - hard_blocks.h, 229
- register_implicit_memory_input
 - implicit_memory.cpp, 236
 - implicit_memory.h, 242
- remap_input_port_to_memory
 - memories.cpp, 259
- remap_output_port_to_memory
 - memories.cpp, 260
- remap_pin_to_new_net
 - netlist_utils.cpp, 387
 - netlist_utils.h, 420
- remap_pin_to_new_node
 - netlist_utils.cpp, 388
 - netlist_utils.h, 421
- removal_list_next
 - netlist_cleanup.cpp, 316
- remove
 - hashtable_t_t, 15
 - queue_t_t, 25
- remove_all
 - queue_t_t, 25
- remove_comments
 - verilog_preprocessor.cpp, 505
- remove_fanout_pins
 - adders.cpp, 203
 - adders.h, 215
- remove_fanout_pins_from_net
 - netlist_utils.cpp, 388
 - netlist_utils.h, 421
- remove_intermediate_variable
 - ast_elaborate.cpp, 75
 - ast_elaborate.h, 96
- remove_list_node
 - adders.cpp, 203
 - adders.h, 215
- remove_para_node
 - ast_elaborate.cpp, 75
 - ast_elaborate.h, 96
- remove_unused_logic
 - netlist_cleanup.cpp, 313
 - netlist_cleanup.h, 318
- remove_unused_nodes
 - netlist_cleanup.cpp, 314
- replace_enode
 - ast_elaborate.cpp, 76
 - ast_elaborate.h, 96
- report_add_distribution
 - adders.cpp, 203
 - adders.h, 215
- report_mult_distribution
 - multipliers.cpp, 286
 - multipliers.h, 293
- report_sub_distribution
 - subtractions.cpp, 297
 - subtractions.h, 302
- resolve_node
 - ast_util.cpp, 115
 - ast_util.h, 131
- ret_veri_definedval
 - verilog_preprocessor.cpp, 506
 - verilog_preprocessor.h, 513
- reverse_string
 - odin_util.cpp, 476

- odin_util.h, [493](#)
- s_adder, [25](#)
 - next, [26](#)
 - size_a, [26](#)
 - size_b, [26](#)
 - size_cin, [26](#)
 - size_cout, [26](#)
 - size_sumout, [27](#)
- s_memory, [27](#)
 - next, [27](#)
 - size_addr1, [27](#)
 - size_addr2, [28](#)
 - size_d1, [28](#)
 - size_d2, [28](#)
 - size_out1, [28](#)
 - size_out2, [28](#)
- s_memory_port_sizes, [29](#)
 - name, [29](#)
 - size, [29](#)
- s_multiplier, [29](#)
 - next, [30](#)
 - size_a, [30](#)
 - size_b, [30](#)
 - size_out, [30](#)
- SEQUENTIAL
 - netlist_create_from_ast.cpp, [321](#)
- SIM_WAVE_LENGTH
 - simulate_blif.h, [573](#)
- SINGLE_PORT_MEMORY_NAME
 - simulate_blif.h, [574](#)
- SL_string
 - node_creation_library.cpp, [440](#)
- SR_string
 - node_creation_library.cpp, [440](#)
- STRING_CACHE, [34](#)
 - data, [34](#)
 - free, [35](#)
 - mod, [35](#)
 - mul, [35](#)
 - next_string, [35](#)
 - size, [35](#)
 - string, [35](#)
 - string_hash, [36](#)
 - string_hash_size, [36](#)
- sc_add_string
 - string_cache.cpp, [616](#)
 - string_cache.h, [623](#)
- sc_do_alloc
 - string_cache.cpp, [617](#)
 - string_cache.h, [623](#)
- sc_free_string_cache
 - string_cache.cpp, [618](#)
 - string_cache.h, [624](#)
- sc_lookup_string
 - string_cache.cpp, [619](#)
 - string_cache.h, [625](#)
- sc_new_string_cache
 - string_cache.cpp, [620](#)
 - string_cache.h, [626](#)
- sc_valid_id
 - string_cache.cpp, [621](#)
 - string_cache.h, [627](#)
- search_certain_operation
 - ast_elaborate.cpp, [76](#)
 - ast_elaborate.h, [97](#)
- search_clock_name
 - read_blif.cpp, [193](#)
- search_for_node
 - ast_elaborate.cpp, [76](#)
 - ast_elaborate.h, [97](#)
- search_marked_node
 - ast_elaborate.cpp, [77](#)
 - ast_elaborate.h, [97](#)
- search_replace
 - odin_util.cpp, [476](#)
 - odin_util.h, [493](#)
- sequential_levelized_dfs
 - netlist_check.cpp, [309](#)
- sequential_times
 - stages_t, [33](#)
- set_clear
 - ace.cpp, [526](#)
- set_clock_ratio
 - simulate_blif.cpp, [561](#)
 - simulate_blif.h, [599](#)
- set_default_config
 - odin_ii.cpp, [460](#)
- set_default_optimization_settings
 - read_xml_config_file.cpp, [498](#)
- set_insert
 - ace.cpp, [519](#)
- set_new
 - ace.cpp, [519](#)
- set_pin_cycle
 - simulate_blif.cpp, [561](#)
 - simulate_blif.h, [599](#)
- set_remove
 - ace.cpp, [519](#)
- set_size
 - ace.cpp, [519](#)
- shift_operation
 - ast_elaborate.cpp, [77](#)
 - ast_elaborate.h, [98](#)
- signature
 - hard_block_ports, [12](#)
- sim_block.h
 - simulate_block_cycle, [533](#)

simplify_ast
 ast_elaborate.cpp, 77
 ast_elaborate.h, 98
simplify_expression
 ast_elaborate.cpp, 78
 ast_elaborate.h, 98
simulate_blif.cpp
 add_additional_items_to_lines, 536
 add_arrays, 536
 add_test_vector_to_lines, 536
 assign_memory_from_mif_file, 537
 assign_node_to_line, 537
 compare_test_vectors, 538
 compute_add_node, 538
 compute_and_store_value, 539
 compute_dual_port_memory, 539
 compute_flipflop_node, 540
 compute_generic_node, 540
 compute_hard_ip_node, 540
 compute_memory_address, 541
 compute_memory_node, 541
 compute_multiply_node, 541
 compute_mux_2_node, 542
 compute_single_port_memory, 542
 compute_unary_sub_node, 542
 count_test_vectors, 543
 create_line, 543
 create_lines, 544
 enqueue_node_if_ready, 544
 find_portname_in_lines, 544
 flag_undriven_input_pins, 545
 free_lines, 545
 free_pin_name_list, 545
 free_stages, 546
 free_test_vector, 546
 generate_random_test_vector, 547
 generate_vector_header, 547
 get_children_of, 548
 get_children_of_nodepin, 548
 get_children_pinnumber_of, 548
 get_circuit_filename, 549
 get_clock_ratio, 549
 get_line_pin_value, 549
 get_mif_filename, 549
 get_next_vector, 550
 get_num_covered_nodes, 550
 get_pin_cycle, 551
 get_pin_value, 551
 get_values_offset, 552
 index_pin_name_list, 552
 initialize_pin, 553
 insert_pin_into_line, 553
 instantiate_memory, 554
 is_clock_node, 554
 is_even_cycle, 555
 is_node_complete, 555
 is_node_ready, 555
 is_posedge, 556
 is_vector, 556
 line_has_unknown_pin, 557
 max, 535
 min, 535
 multiply_arrays, 557
 parse_mif_radix, 557
 parse_pin_name_list, 558
 parse_test_vector, 558
 preprocess_mif_file, 558
 print_ancestry, 559
 print_netlist_stats, 559
 print_progress_bar, 559
 print_simulation_stats, 560
 print_time, 560
 print_update_trace, 561
 set_clock_ratio, 561
 set_pin_cycle, 561
 simulate_cycle, 562
 simulate_first_cycle, 562
 simulate_netlist, 563
 stage_ordered_nodes, 563
 trim_string, 564
 unary_sub_arrays, 564
 update_pin_value, 565
 update_undriven_input_pins, 565
 vector_value_to_hex, 566
 verify_lines, 566
 verify_output_vectors, 566
 verify_test_vector_headers, 567
 wall_time, 567
 write_vector_headers, 568
 write_vector_to_file, 568
 write_vector_to_modelsim_file, 569
 write_wave_to_file, 569
 write_wave_to_modelsim_file, 570
simulate_blif.h
 add_additional_items_to_lines, 574
 add_arrays, 574
 add_test_vector_to_lines, 575
 assign_memory_from_mif_file, 575
 assign_node_to_line, 576
 BUFFER_MAX_SIZE, 573
 compare_test_vectors, 576
 compute_add_node, 577
 compute_and_store_value, 577
 compute_dual_port_memory, 578
 compute_flipflop_node, 578
 compute_generic_node, 578
 compute_hard_ip_node, 579
 compute_memory_address, 579

compute_memory_node, 579
 compute_multiply_node, 580
 compute_mux_2_node, 580
 compute_single_port_memory, 580
 compute_unary_sub_node, 581
 count_test_vectors, 581
 create_line, 581
 create_lines, 582
 DUAL_PORT_MEMORY_NAME, 573
 enqueue_node_if_ready, 582
 find_portname_in_lines, 583
 flag_undriven_input_pins, 583
 free_lines, 583
 free_pin_name_list, 584
 free_stages, 584
 free_test_vector, 585
 generate_random_test_vector, 585
 generate_vector_header, 586
 get_children_of, 586
 get_children_of_nodepin, 586
 get_children_pinnumber_of, 587
 get_circuit_filename, 587
 get_clock_ratio, 587
 get_line_pin_value, 587
 get_mif_filename, 588
 get_next_vector, 588
 get_num_covered_nodes, 589
 get_pin_cycle, 589
 get_pin_value, 589
 get_values_offset, 590
 INPUT_VECTOR_FILE_NAME, 573
 index_pin_name_list, 590
 initialize_pin, 591
 insert_pin_into_line, 591
 instantiate_memory, 592
 is_clock_node, 592
 is_even_cycle, 593
 is_node_complete, 593
 is_node_ready, 593
 is_posedge, 594
 is_vector, 594
 line_has_unknown_pin, 595
 multiply_arrays, 595
 OUTPUT_ACTIVITY_FILE_NAME, 573
 OUTPUT_VECTOR_FILE_NAME, 573
 parse_mif_radix, 595
 parse_pin_name_list, 596
 parse_test_vector, 596
 preprocess_mif_file, 596
 print_ancestry, 597
 print_netlist_stats, 597
 print_progress_bar, 597
 print_simulation_stats, 598
 print_time, 598
 print_update_trace, 599
 SIM_WAVE_LENGTH, 573
 SINGLE_PORT_MEMORY_NAME, 574
 set_clock_ratio, 599
 set_pin_cycle, 599
 simulate_cycle, 600
 simulate_first_cycle, 600
 simulate_netlist, 601
 stage_ordered_nodes, 601
 trim_string, 602
 unary_sub_arrays, 602
 update_pin_value, 603
 update_undriven_input_pins, 603
 vector_value_to_hex, 604
 verify_lines, 604
 verify_output_vectors, 604
 verify_test_vector_headers, 605
 wall_time, 605
 write_vector_headers, 606
 write_vector_to_file, 606
 write_vector_to_modelsim_file, 607
 write_wave_to_file, 607
 write_wave_to_modelsim_file, 608
 simulate_block_cycle
 sim_block.h, 533
 simulate_cycle
 simulate_blif.cpp, 562
 simulate_blif.h, 600
 simulate_first_cycle
 simulate_blif.cpp, 562
 simulate_blif.h, 600
 simulate_netlist
 simulate_blif.cpp, 563
 simulate_blif.h, 601
 single_port_rams
 memories.cpp, 263
 memories.h, 279
 size
 s_memory_port_sizes, 29
 STRING_CACHE, 35
 size_a
 s_adder, 26
 s_multiplier, 30
 size_addr1
 s_memory, 27
 size_addr2
 s_memory, 28
 size_all_file_items_list
 parse_making_ast.cpp, 156
 size_b
 s_adder, 26
 s_multiplier, 30
 size_block_instantiations
 parse_making_ast.cpp, 156

- size_cin
 - s_adder, [26](#)
- size_cout
 - s_adder, [26](#)
- size_d1
 - s_memory, [28](#)
- size_d2
 - s_memory, [28](#)
- size_function_instantiations
 - parse_making_ast.cpp, [156](#)
- size_function_instantiations_by_module
 - parse_making_ast.cpp, [156](#)
- size_module_instantiations
 - parse_making_ast.cpp, [157](#)
- size_module_variables_not_defined
 - parse_making_ast.cpp, [157](#)
- size_out
 - s_multiplier, [30](#)
- size_out1
 - s_memory, [28](#)
- size_out2
 - s_memory, [28](#)
- size_sumout
 - s_adder, [27](#)
- sizes
 - hard_block_ports, [12](#)
- sort_signal_list_alphabetically
 - netlist_utils.cpp, [389](#)
 - netlist_utils.h, [422](#)
- sp_memory_list
 - memories.cpp, [263](#)
 - memories.h, [279](#)
- sp_ram_signals, [30](#)
 - addr, [31](#)
 - clk, [31](#)
 - data, [31](#)
 - out, [31](#)
 - we, [31](#)
- split_adder
 - adders.cpp, [204](#)
 - adders.h, [216](#)
- split_adder_for_sub
 - subtractions.cpp, [298](#)
 - subtractions.h, [302](#)
- split_dp_memory_depth
 - memories.cpp, [260](#)
 - memories.h, [276](#)
- split_dp_memory_width
 - memories.cpp, [260](#)
 - memories.h, [277](#)
- split_list
 - memories.cpp, [263](#)
- split_multiplier
 - multipliers.cpp, [286](#)
 - multipliers.h, [293](#)
- split_multiplier_a
 - multipliers.cpp, [287](#)
- split_multiplier_b
 - multipliers.cpp, [287](#)
- split_sp_memory_depth
 - memories.cpp, [261](#)
 - memories.h, [277](#)
- split_sp_memory_width
 - memories.cpp, [261](#)
 - memories.h, [278](#)
- stage_ordered_nodes
 - simulate_blif.cpp, [563](#)
 - simulate_blif.h, [601](#)
- stages
 - stages_t, [34](#)
- stages_t, [32](#)
 - count, [32](#)
 - counts, [32](#)
 - num_children, [33](#)
 - num_connections, [33](#)
 - num_nodes, [33](#)
 - num_parallel_nodes, [33](#)
 - parallel_times, [33](#)
 - sequential_times, [33](#)
 - stages, [34](#)
- static_prob
 - ace_cube_t, [2](#)
- store
 - hashtable_t_t, [16](#)
- store_exp_list
 - ast_elaborate.cpp, [78](#)
 - ast_elaborate.h, [99](#)
- store_size
 - hashtable_t_t, [16](#)
- string
 - STRING_CACHE, [35](#)
- string_cache.cpp
 - generate_sc_hash, [616](#)
 - sc_add_string, [616](#)
 - sc_do_alloc, [617](#)
 - sc_free_string_cache, [618](#)
 - sc_lookup_string, [619](#)
 - sc_new_string_cache, [620](#)
 - sc_valid_id, [621](#)
 - string_hash, [621](#)
- string_cache.h
 - sc_add_string, [623](#)
 - sc_do_alloc, [623](#)
 - sc_free_string_cache, [624](#)
 - sc_lookup_string, [625](#)
 - sc_new_string_cache, [626](#)
 - sc_valid_id, [627](#)
- string_hash

- STRING_CACHE, 36
- string_cache.cpp, 621
- string_hash_size
 - STRING_CACHE, 36
- string_to_lower
 - odin_util.cpp, 477
 - odin_util.h, 494
- string_to_upper
 - odin_util.cpp, 477
 - odin_util.h, 494
- sub
 - subtractions.cpp, 299
- sub_chain_list
 - subtractions.cpp, 299
 - subtractions.h, 303
- sub_list
 - subtractions.cpp, 299
 - subtractions.h, 303
- subchaintotal
 - subtractions.cpp, 299
- subtractions.cpp
 - clean_adders_for_sub, 295
 - declare_hard_adder_for_sub, 296
 - init_split_adder_for_sub, 296
 - instantiate_hard_adder_subtraction, 297
 - iterate_adders_for_sub, 297
 - longest_subtractor_chain, 298
 - report_sub_distribution, 297
 - split_adder_for_sub, 298
 - sub, 299
 - sub_chain_list, 299
 - sub_list, 299
 - subchaintotal, 299
 - subtractor_chain_count, 299
 - total_subtractors, 299
- subtractions.h
 - clean_adders_for_sub, 300
 - declare_hard_adder_for_sub, 300
 - init_sub_distribution, 301
 - instantiate_hard_adder_subtraction, 301
 - iterate_adders_for_sub, 301
 - report_sub_distribution, 302
 - split_adder_for_sub, 302
 - sub_chain_list, 303
 - sub_list, 303
- subtractor_chain_count
 - netlist_cleanup.cpp, 317
 - subtractions.cpp, 299
- sum_of_addsub_logs
 - adders.cpp, 207
 - netlist_cleanup.cpp, 317
- sumout
 - adder_signals, 4
- symbol
 - veri_define, 38
- t_adder
 - adders.h, 209
- t_memory
 - memories.h, 265
- t_memory_port_sizes
 - memories.h, 265
- t_multiplier
 - multipliers.h, 289
- TOKENS
 - read_blif.cpp, 183
- TWO
 - ace.cpp, 520
- tail
 - queue_t_t, 25
- terminate_continuous_assignment
 - netlist_create_from_ast.cpp, 351
- terminate_registered_assignment
 - netlist_create_from_ast.cpp, 352
- test_vector, 36
 - count, 36
 - counts, 37
 - values, 37
- the_netlist
 - adders.cpp, 207
- to_str
 - ParseInitRegState, 22
- to_view_parse
 - globals.h, 56
 - parse_making_ast.cpp, 157
- top
 - veri_flag_stack, 40
 - verilog_preprocessor.cpp, 506
 - verilog_preprocessor.h, 513
- top_input_nodes
 - globals.h, 56
- top_module
 - globals.h, 57
 - netlist_create_from_ast.cpp, 355
- top_output_nodes
 - globals.h, 57
- total
 - adders.cpp, 207
 - adders.h, 218
- total_adders
 - adders.cpp, 207
 - netlist_cleanup.cpp, 317
- total_addsub_chain_count
 - adders.cpp, 208
 - netlist_cleanup.cpp, 317
- total_subtractors
 - netlist_cleanup.cpp, 317
 - subtractions.cpp, 299

- translate_expression
 - ast_elaborate.cpp, 78
 - ast_elaborate.h, 99
- traverse_backward
 - netlist_cleanup.cpp, 314
- traverse_forward
 - netlist_cleanup.cpp, 314
- traverse_list
 - adders.cpp, 204
 - adders.h, 216
- traverse_operation_node
 - adders.cpp, 205
 - adders.h, 217
- trim
 - verilog_preprocessor.cpp, 506
 - verilog_preprocessor.h, 513
- trim_string
 - simulate_blif.cpp, 564
 - simulate_blif.h, 602
- twos_complement
 - odin_util.cpp, 477
 - odin_util.h, 494
- type
 - exp_node, 8
 - line_t, 19
- type_descriptors
 - globals.h, 57
 - odin_ii.cpp, 461
- type_of_circuit
 - netlist_create_from_ast.cpp, 356
- unary_sub_arrays
 - simulate_blif.cpp, 564
 - simulate_blif.h, 602
- unique_label_count
 - parse_making_ast.cpp, 157
- unique_node_name_id
 - node_creation_library.cpp, 440
- update_pin_value
 - simulate_blif.cpp, 565
 - simulate_blif.h, 603
- update_tree_tag
 - ast_util.cpp, 116
- update_undriven_input_pins
 - simulate_blif.cpp, 565
 - simulate_blif.h, 603
- useless_nodes
 - netlist_cleanup.cpp, 317
- VCC_NAME
 - read_blif.cpp, 184
- VCC_NODE_string
 - node_creation_library.cpp, 440
- VISITED_BACKWARD
 - netlist_cleanup.cpp, 311
- VISITED_FORWARD
 - netlist_cleanup.cpp, 312
- VISITED_REMOVAL
 - netlist_cleanup.cpp, 312
- validate_string_regex
 - odin_util.cpp, 478
 - odin_util.h, 495
- value
 - veri_define, 38
- values
 - test_vector, 37
- variable
 - exp_node, 8
- vcc_node
 - globals.h, 57
- vector_value_to_hex
 - simulate_blif.cpp, 566
 - simulate_blif.h, 604
- veri_Defines, 38
 - current_index, 39
 - current_size, 39
 - defined_constants, 39
- veri_Includes, 42
 - current_index, 42
 - current_size, 42
 - included_files, 42
- veri_define, 37
 - defined_in, 37
 - line, 38
 - symbol, 38
 - value, 38
 - verilog_preprocessor.h, 510
- veri_defines
 - verilog_preprocessor.cpp, 508
 - verilog_preprocessor.h, 515
- veri_flag_node, 39
 - flag, 40
 - next, 40
 - verilog_preprocessor.h, 510
- veri_flag_stack, 40
 - top, 40
- veri_include, 41
 - included_from, 41
 - line, 41
 - path, 41
 - verilog_preprocessor.h, 510
- veri_includes
 - verilog_preprocessor.cpp, 508
 - verilog_preprocessor.h, 515
- veri_is_defined
 - verilog_preprocessor.cpp, 507
 - verilog_preprocessor.h, 514
- veri_preproc
 - verilog_preprocessor.cpp, 507

- verilog_preprocessor.h, 514
- veri_preproc_bootstrapped
 - verilog_preprocessor.cpp, 507
 - verilog_preprocessor.h, 514
- verify_hard_block_ports_against_model
 - read_blif.cpp, 193
- verify_lines
 - simulate_blif.cpp, 566
 - simulate_blif.h, 604
- verify_output_vectors
 - simulate_blif.cpp, 566
 - simulate_blif.h, 604
- verify_test_vector_headers
 - simulate_blif.cpp, 567
 - simulate_blif.h, 605
- verilog_bison_user_defined.h
 - yyparse, 500
- verilog_netlist
 - globals.h, 57
 - netlist_create_from_ast.cpp, 356
- verilog_preprocessor.cpp
 - add_veri_define, 501
 - add_veri_include, 502
 - clean_veri_define, 502
 - clean_veri_include, 503
 - cleanup_veri_preproc, 503
 - format_verilog_file, 503
 - format_verilog_variable, 504
 - init_veri_preproc, 504
 - open_source_file, 504
 - pop, 505
 - push, 505
 - remove_comments, 505
 - ret_veri_definedval, 506
 - top, 506
 - trim, 506
 - veri_defines, 508
 - veri_includes, 508
 - veri_is_defined, 507
 - veri_preproc, 507
 - veri_preproc_bootstrapped, 507
- verilog_preprocessor.h
 - add_veri_define, 510
 - add_veri_include, 510
 - clean_veri_define, 511
 - clean_veri_include, 511
 - cleanup_veri_preproc, 511
 - DefaultSize, 509
 - init_veri_preproc, 512
 - MaxLine, 509
 - pop, 512
 - push, 512
 - ret_veri_definedval, 513
 - top, 513
 - trim, 513
 - veri_define, 510
 - veri_defines, 515
 - veri_flag_node, 510
 - veri_include, 510
 - veri_includes, 515
 - veri_is_defined, 514
 - veri_preproc, 514
 - veri_preproc_bootstrapped, 514
- vtr-verilog-to-routing/ODIN_II/SRC/DEBUG_TOOL↔
 - S/netlist_visualizer.c, 43
- vtr-verilog-to-routing/ODIN_II/SRC/DEBUG_TOOL↔
 - S/netlist_visualizer.cpp, 46
- vtr-verilog-to-routing/ODIN_II/SRC/DEBUG_TOOL↔
 - S/netlist_visualizer.h, 49
- vtr-verilog-to-routing/ODIN_II/SRC/DEBUG_TOOL↔
 - S/print_netlist.c, 50
- vtr-verilog-to-routing/ODIN_II/SRC/DEBUG_TOOL↔
 - S/print_netlist.cpp, 51
- vtr-verilog-to-routing/ODIN_II/SRC/DEBUG_TOOL↔
 - S/print_netlist.h, 52
- vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL/read↔
 - _xml_config_file.cpp, 495
- vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL/read↔
 - _xml_config_file.h, 499
- vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL↔
 - L/verilog_bison_user_defined.h, 500
- vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL↔
 - L/verilog_preprocessor.cpp, 501
- vtr-verilog-to-routing/ODIN_II/SRC/PARSE_TOOL↔
 - L/verilog_preprocessor.h, 508
- vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/ace.cpp, 515
- vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/ace.h, 527
- vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/queue.↔
 - cpp, 528
- vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/queue.h, 531
- vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/sim_↔
 - block.h, 532
- vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/simulate↔
 - _blif.cpp, 533
- vtr-verilog-to-routing/ODIN_II/SRC/SIM_TOOLS/simulate↔
 - _blif.h, 570
- vtr-verilog-to-routing/ODIN_II/SRC/STRING_HASH_TO↔
 - OL/hashtable.cpp, 608
- vtr-verilog-to-routing/ODIN_II/SRC/STRING_HASH_TO↔
 - OL/hashtable.h, 614
- vtr-verilog-to-routing/ODIN_II/SRC/STRING_HASH_TO↔
 - OL/string_cache.cpp, 615
- vtr-verilog-to-routing/ODIN_II/SRC/STRING_HASH_TO↔
 - OL/string_cache.h, 622
- vtr-verilog-to-routing/ODIN_II/SRC/globals.h, 52

- vtr-verilog-to-routing/ODIN_II/SRC/lib_ast/ast_elaborate.↵
cpp, 58
- vtr-verilog-to-routing/ODIN_II/SRC/lib_ast/ast_elaborate.↵
h, 80
- vtr-verilog-to-routing/ODIN_II/SRC/lib_ast/ast_util.cpp,
100
- vtr-verilog-to-routing/ODIN_II/SRC/lib_ast/ast_util.h, 116
- vtr-verilog-to-routing/ODIN_II/SRC/lib_ast/parse_↵
making_ast.cpp, 132
- vtr-verilog-to-routing/ODIN_II/SRC/lib_ast/parse_↵
making_ast.h, 157
- vtr-verilog-to-routing/ODIN_II/SRC/lib_blif/output_blif.cpp,
176
- vtr-verilog-to-routing/ODIN_II/SRC/lib_blif/output_blif.h,
181
- vtr-verilog-to-routing/ODIN_II/SRC/lib_blif/read_blif.cpp,
182
- vtr-verilog-to-routing/ODIN_II/SRC/lib_blif/read_blif.h, 194
- vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/adders.cpp,
195
- vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/adders.h,
208
- vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/hard_↵
blocks.cpp, 218
- vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/hard_↵
blocks.h, 225
- vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/implicit_↵
memory.cpp, 230
- vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/implicit_↵
memory.h, 237
- vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/memories.↵
cpp, 244
- vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/memories.↵
h, 263
- vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/multipliers.↵
cpp, 279
- vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/multipliers.↵
h, 288
- vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/subtractions.↵
cpp, 295
- vtr-verilog-to-routing/ODIN_II/SRC/lib_blocks/subtractions.↵
h, 300
- vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_↵
check.cpp, 304
- vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_↵
check.h, 309
- vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_↵
cleanup.cpp, 310
- vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_↵
cleanup.h, 318
- vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_↵
create_from_ast.cpp, 318
- vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_↵
create_from_ast.h, 356
- vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_↵
utils.cpp, 358
- vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/netlist_↵
utils.h, 390
- vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/node_↵
creation_library.cpp, 422
- vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/node_↵
creation_library.h, 440
- vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/partial_↵
map.cpp, 449
- vtr-verilog-to-routing/ODIN_II/SRC/lib_netlist/partial_↵
map.h, 457
- vtr-verilog-to-routing/ODIN_II/SRC/odin_ii.cpp, 458
- vtr-verilog-to-routing/ODIN_II/SRC/odin_util.cpp, 462
- vtr-verilog-to-routing/ODIN_II/SRC/odin_util.h, 480
- vtr-verilog-to-routing/ODIN_II/SRC/types.h, 627
- WHICH_BIT
ace.cpp, 520
- WHICH_WORD
ace.cpp, 520
- wall_time
simulate_blif.cpp, 567
simulate_blif.h, 605
- warning_message
odin_util.cpp, 478
- we
sp_ram_signals, 31
- we1
dp_ram_signals, 6
- we2
dp_ram_signals, 6
- write_node
ast_elaborate.cpp, 61
- write_vector_headers
simulate_blif.cpp, 568
simulate_blif.h, 606
- write_vector_to_file
simulate_blif.cpp, 568
simulate_blif.h, 606
- write_vector_to_modelsim_file
simulate_blif.cpp, 569
simulate_blif.h, 607
- write_wave_to_file
simulate_blif.cpp, 569
simulate_blif.h, 607
- write_wave_to_modelsim_file
simulate_blif.cpp, 570
simulate_blif.h, 608
- yylineno
globals.h, 57
parse_making_ast.cpp, 157
- yyparse
verilog_bison_user_defined.h, 500

ZERO

- ace.cpp, [520](#)

zero_net

- globals.h, [57](#)

zero_string

- globals.h, [58](#)

- netlist_create_from_ast.cpp, [356](#)