4. Executing SYNTH pass.

4.1. Executing HIERARCHY pass (managing design hierarchy).

4.1.1. Analyzing design hierarchy..

Top module: \controller

4.1.2. Analyzing design hierarchy..

Top module: \controller

Removed 0 unused modules.

4.2. Executing PROC pass (convert processes to netlists).

4.2.1. Executing PROC\_CLEAN pass (remove empty switches from decision trees).

Cleaned up 0 empty switches.

4.2.2. Executing PROC\_RMDEAD pass (remove dead branches from decision trees).

Marked 1 switch rules as full\_case in process $proc$SAR\_LOGIC.v:18$1 in module controller.

Removed a total of 0 dead cases.

4.2.3. Executing PROC\_PRUNE pass (remove redundant assignments in processes).

Removed 0 redundant assignments.

Promoted 0 assignments to connections.

4.2.4. Executing PROC\_INIT pass (extract init attributes).

4.2.5. Executing PROC\_ARST pass (detect async resets in processes).

4.2.6. Executing PROC\_ROM pass (convert switches to ROMs).

Converted 0 switches.

<suppressed ~4 debug messages>

4.2.7. Executing PROC\_MUX pass (convert decision trees to multiplexers).

Creating decoders for process `\controller.$proc$SAR\_LOGIC.v:18$1'.

1/3: $0\mask[7:0]

2/3: $0\state[1:0]

3/3: $0\result[7:0]

4.2.8. Executing PROC\_DLATCH pass (convert process syncs to latches).

4.2.9. Executing PROC\_DFF pass (convert process syncs to FFs).

Creating register for signal `\controller.\result' using process `\controller.$proc$SAR\_LOGIC.v:18$1'.

created $dff cell `$procdff$31' with positive edge clock.

Creating register for signal `\controller.\state' using process `\controller.$proc$SAR\_LOGIC.v:18$1'.

created $dff cell `$procdff$32' with positive edge clock.

Creating register for signal `\controller.\mask' using process `\controller.$proc$SAR\_LOGIC.v:18$1'.

created $dff cell `$procdff$33' with positive edge clock.

4.2.10. Executing PROC\_MEMWR pass (convert process memory writes to cells).

4.2.11. Executing PROC\_CLEAN pass (remove empty switches from decision trees).

Found and cleaned up 4 empty switches in `\controller.$proc$SAR\_LOGIC.v:18$1'.

Removing empty process `controller.$proc$SAR\_LOGIC.v:18$1'.

Cleaned up 4 empty switches.

4.2.12. Executing OPT\_EXPR pass (perform const folding).

Optimizing module controller.

<suppressed ~5 debug messages>

4.3. Executing OPT\_EXPR pass (perform const folding).

Optimizing module controller.

4.4. Executing OPT\_CLEAN pass (remove unused cells and wires).

Finding unused cells or wires in module \controller..

Removed 1 unused cells and 13 unused wires.

<suppressed ~2 debug messages>

4.5. Executing CHECK pass (checking for obvious problems).

Checking module controller...

Found and reported 0 problems.

4.6. Executing OPT pass (performing simple optimizations).

4.6.1. Executing OPT\_EXPR pass (perform const folding).

Optimizing module controller.

4.6.2. Executing OPT\_MERGE pass (detect identical cells).

Finding identical cells in module `\controller'.

<suppressed ~18 debug messages>

Removed a total of 6 cells.

4.6.3. Executing OPT\_MUXTREE pass (detect dead branches in mux trees).

Running muxtree optimizer on module \controller..

Creating internal representation of mux trees.

Evaluating internal representation of mux trees.

Analyzing evaluation results.

Removed 0 multiplexer ports.

<suppressed ~3 debug messages>

4.6.4. Executing OPT\_REDUCE pass (consolidate $\*mux and $reduce\_\* inputs).

Optimizing cells in module \controller.

Performed a total of 0 changes.

4.6.5. Executing OPT\_MERGE pass (detect identical cells).

Finding identical cells in module `\controller'.

Removed a total of 0 cells.

4.6.6. Executing OPT\_DFF pass (perform DFF optimizations).

4.6.7. Executing OPT\_CLEAN pass (remove unused cells and wires).

Finding unused cells or wires in module \controller..

Removed 0 unused cells and 6 unused wires.

<suppressed ~1 debug messages>

4.6.8. Executing OPT\_EXPR pass (perform const folding).

Optimizing module controller.

4.6.9. Rerunning OPT passes. (Maybe there is more to do..)

4.6.10. Executing OPT\_MUXTREE pass (detect dead branches in mux trees).

Running muxtree optimizer on module \controller..

Creating internal representation of mux trees.

Evaluating internal representation of mux trees.

Analyzing evaluation results.

Removed 0 multiplexer ports.

<suppressed ~3 debug messages>

4.6.11. Executing OPT\_REDUCE pass (consolidate $\*mux and $reduce\_\* inputs).

Optimizing cells in module \controller.

Performed a total of 0 changes.

4.6.12. Executing OPT\_MERGE pass (detect identical cells).

Finding identical cells in module `\controller'.

Removed a total of 0 cells.

4.6.13. Executing OPT\_DFF pass (perform DFF optimizations).

4.6.14. Executing OPT\_CLEAN pass (remove unused cells and wires).

Finding unused cells or wires in module \controller..

4.6.15. Executing OPT\_EXPR pass (perform const folding).

Optimizing module controller.

4.6.16. Finished OPT passes. (There is nothing left to do.)

4.7. Executing FSM pass (extract and optimize FSM).

4.7.1. Executing FSM\_DETECT pass (finding FSMs in design).

Found FSM state register controller.state.

4.7.2. Executing FSM\_EXTRACT pass (extracting FSM from design).

Extracting FSM `\state' from module `\controller'.

found $dff cell for state register: $procdff$32

root of input selection tree: $0\state[1:0]

found reset state: 2'00 (guessed from mux tree)

found ctrl input: \go

found state code: 2'00

found ctrl input: $procmux$17\_CMP

found ctrl input: \sample

found ctrl input: $procmux$19\_CMP

found ctrl input: \mask [0]

found state code: 2'11

found state code: 2'10

found state code: 2'01

found ctrl output: \sample

found ctrl output: \valid

found ctrl output: $procmux$17\_CMP

found ctrl output: $procmux$19\_CMP

ctrl inputs: { \mask [0] \go }

ctrl outputs: { $procmux$19\_CMP $procmux$17\_CMP $0\state[1:0] \sample \valid }

transition: 2'00 2'-0 -> 2'00 6'100000

transition: 2'00 2'-1 -> 2'01 6'100100

transition: 2'10 2'-0 -> 2'00 6'010000

transition: 2'10 2'01 -> 2'10 6'011000

transition: 2'10 2'11 -> 2'11 6'011100

transition: 2'01 2'-0 -> 2'00 6'000010

transition: 2'01 2'-1 -> 2'10 6'001010

transition: 2'11 2'-0 -> 2'00 6'000001

transition: 2'11 2'-1 -> 2'11 6'001101

4.7.3. Executing FSM\_OPT pass (simple optimizations of FSMs).

Optimizing FSM `$fsm$\state$34' from module `\controller'.

4.7.4. Executing OPT\_CLEAN pass (remove unused cells and wires).

Finding unused cells or wires in module \controller..

Removed 8 unused cells and 8 unused wires.

<suppressed ~9 debug messages>

4.7.5. Executing FSM\_OPT pass (simple optimizations of FSMs).

Optimizing FSM `$fsm$\state$34' from module `\controller'.

Removing unused output signal $0\state[1:0] [0].

Removing unused output signal $0\state[1:0] [1].

Removing unused output signal $procmux$19\_CMP.

4.7.6. Executing FSM\_RECODE pass (re-assigning FSM state encoding).

Recoding FSM `$fsm$\state$34' from module `\controller' using `auto' encoding:

mapping auto encoding to `one-hot` for this FSM.

00 -> ---1

10 -> --1-

01 -> -1--

11 -> 1---

4.7.7. Executing FSM\_INFO pass (dumping all available information on FSM cells).

FSM `$fsm$\state$34' from module `controller':

-------------------------------------

Information on FSM $fsm$\state$34 (\state):

Number of input signals: 2

Number of output signals: 3

Number of state bits: 4

Input signals:

0: \go

1: \mask [0]

Output signals:

0: \valid

1: \sample

2: $procmux$17\_CMP

State encoding:

0: 4'---1 <RESET STATE>

1: 4'--1-

2: 4'-1--

3: 4'1---

Transition Table (state\_in, ctrl\_in, state\_out, ctrl\_out):

0: 0 2'-0 -> 0 3'000

1: 0 2'-1 -> 2 3'000

2: 1 2'-0 -> 0 3'100

3: 1 2'01 -> 1 3'100

4: 1 2'11 -> 3 3'100

5: 2 2'-0 -> 0 3'010

6: 2 2'-1 -> 1 3'010

7: 3 2'-0 -> 0 3'001

8: 3 2'-1 -> 3 3'001

-------------------------------------

4.7.8. Executing FSM\_MAP pass (mapping FSMs to basic logic).

Mapping FSM `$fsm$\state$34' from module `\controller'.

4.8. Executing OPT pass (performing simple optimizations).

4.8.1. Executing OPT\_EXPR pass (perform const folding).

Optimizing module controller.

<suppressed ~4 debug messages>

4.8.2. Executing OPT\_MERGE pass (detect identical cells).

Finding identical cells in module `\controller'.

Removed a total of 0 cells.

4.8.3. Executing OPT\_MUXTREE pass (detect dead branches in mux trees).

Running muxtree optimizer on module \controller..

Creating internal representation of mux trees.

Evaluating internal representation of mux trees.

Analyzing evaluation results.

Removed 0 multiplexer ports.

<suppressed ~2 debug messages>

4.8.4. Executing OPT\_REDUCE pass (consolidate $\*mux and $reduce\_\* inputs).

Optimizing cells in module \controller.

Performed a total of 0 changes.

4.8.5. Executing OPT\_MERGE pass (detect identical cells).

Finding identical cells in module `\controller'.

Removed a total of 0 cells.

4.8.6. Executing OPT\_DFF pass (perform DFF optimizations).

Adding EN signal on $procdff$33 ($dff) from module controller (D = $procmux$8\_Y, Q = \mask).

Adding EN signal on $procdff$31 ($dff) from module controller (D = $procmux$25\_Y, Q = \result).

4.8.7. Executing OPT\_CLEAN pass (remove unused cells and wires).

Finding unused cells or wires in module \controller..

Removed 2 unused cells and 11 unused wires.

<suppressed ~3 debug messages>

4.8.8. Executing OPT\_EXPR pass (perform const folding).

Optimizing module controller.

<suppressed ~2 debug messages>

4.8.9. Rerunning OPT passes. (Maybe there is more to do..)

4.8.10. Executing OPT\_MUXTREE pass (detect dead branches in mux trees).

Running muxtree optimizer on module \controller..

Creating internal representation of mux trees.

Evaluating internal representation of mux trees.

Analyzing evaluation results.

Removed 0 multiplexer ports.

<suppressed ~2 debug messages>

4.8.11. Executing OPT\_REDUCE pass (consolidate $\*mux and $reduce\_\* inputs).

Optimizing cells in module \controller.

Performed a total of 0 changes.

4.8.12. Executing OPT\_MERGE pass (detect identical cells).

Finding identical cells in module `\controller'.

<suppressed ~3 debug messages>

Removed a total of 1 cells.

4.8.13. Executing OPT\_DFF pass (perform DFF optimizations).

4.8.14. Executing OPT\_CLEAN pass (remove unused cells and wires).

Finding unused cells or wires in module \controller..

Removed 0 unused cells and 1 unused wires.

<suppressed ~1 debug messages>

4.8.15. Executing OPT\_EXPR pass (perform const folding).

Optimizing module controller.

4.8.16. Rerunning OPT passes. (Maybe there is more to do..)

4.8.17. Executing OPT\_MUXTREE pass (detect dead branches in mux trees).

Running muxtree optimizer on module \controller..

Creating internal representation of mux trees.

Evaluating internal representation of mux trees.

Analyzing evaluation results.

Removed 0 multiplexer ports.

<suppressed ~2 debug messages>

4.8.18. Executing OPT\_REDUCE pass (consolidate $\*mux and $reduce\_\* inputs).

Optimizing cells in module \controller.

Performed a total of 0 changes.

4.8.19. Executing OPT\_MERGE pass (detect identical cells).

Finding identical cells in module `\controller'.

Removed a total of 0 cells.

4.8.20. Executing OPT\_DFF pass (perform DFF optimizations).

4.8.21. Executing OPT\_CLEAN pass (remove unused cells and wires).

Finding unused cells or wires in module \controller..

4.8.22. Executing OPT\_EXPR pass (perform const folding).

Optimizing module controller.

4.8.23. Finished OPT passes. (There is nothing left to do.)

4.9. Executing WREDUCE pass (reducing word size of cells).

Removed top 1 bits (of 2) from port B of cell controller.$auto$fsm\_map.cc:77:implement\_pattern\_cache$47 ($eq).

4.10. Executing PEEPOPT pass (run peephole optimizers).

4.11. Executing OPT\_CLEAN pass (remove unused cells and wires).

Finding unused cells or wires in module \controller..

4.12. Executing ALUMACC pass (create $alu and $macc cells).

Extracting $alu and $macc cells in module controller:

created 0 $alu and 0 $macc cells.

4.13. Executing SHARE pass (SAT-based resource sharing).

4.14. Executing OPT pass (performing simple optimizations).

4.14.1. Executing OPT\_EXPR pass (perform const folding).

Optimizing module controller.

4.14.2. Executing OPT\_MERGE pass (detect identical cells).

Finding identical cells in module `\controller'.

Removed a total of 0 cells.

4.14.3. Executing OPT\_MUXTREE pass (detect dead branches in mux trees).

Running muxtree optimizer on module \controller..

Creating internal representation of mux trees.

Evaluating internal representation of mux trees.

Analyzing evaluation results.

Removed 0 multiplexer ports.

<suppressed ~2 debug messages>

4.14.4. Executing OPT\_REDUCE pass (consolidate $\*mux and $reduce\_\* inputs).

Optimizing cells in module \controller.

Performed a total of 0 changes.

4.14.5. Executing OPT\_MERGE pass (detect identical cells).

Finding identical cells in module `\controller'.

Removed a total of 0 cells.

4.14.6. Executing OPT\_DFF pass (perform DFF optimizations).

4.14.7. Executing OPT\_CLEAN pass (remove unused cells and wires).

Finding unused cells or wires in module \controller..

4.14.8. Executing OPT\_EXPR pass (perform const folding).

Optimizing module controller.

4.14.9. Finished OPT passes. (There is nothing left to do.)

4.15. Executing MEMORY pass.

4.15.1. Executing OPT\_MEM pass (optimize memories).

Performed a total of 0 transformations.

4.15.2. Executing OPT\_MEM\_PRIORITY pass (removing unnecessary memory write priority relations).

Performed a total of 0 transformations.

4.15.3. Executing OPT\_MEM\_FEEDBACK pass (finding memory read-to-write feedback paths).

4.15.4. Executing MEMORY\_BMUX2ROM pass (converting muxes to ROMs).

4.15.5. Executing MEMORY\_DFF pass (merging $dff cells to $memrd).

4.15.6. Executing OPT\_CLEAN pass (remove unused cells and wires).

Finding unused cells or wires in module \controller..

4.15.7. Executing MEMORY\_SHARE pass (consolidating $memrd/$memwr cells).

4.15.8. Executing OPT\_MEM\_WIDEN pass (optimize memories where all ports are wide).

Performed a total of 0 transformations.

4.15.9. Executing OPT\_CLEAN pass (remove unused cells and wires).

Finding unused cells or wires in module \controller..

4.15.10. Executing MEMORY\_COLLECT pass (generating $mem cells).

4.16. Executing OPT\_CLEAN pass (remove unused cells and wires).

Finding unused cells or wires in module \controller..

4.17. Executing OPT pass (performing simple optimizations).

4.17.1. Executing OPT\_EXPR pass (perform const folding).

Optimizing module controller.

<suppressed ~3 debug messages>

4.17.2. Executing OPT\_MERGE pass (detect identical cells).

Finding identical cells in module `\controller'.

Removed a total of 0 cells.

4.17.3. Executing OPT\_DFF pass (perform DFF optimizations).

Adding SRST signal on $auto$ff.cc:266:slice$68 ($dffe) from module controller (D = \mask [7:1], Q = \mask [6:0], rval = 7'0000000).

Adding SRST signal on $auto$ff.cc:266:slice$73 ($dffe) from module controller (D = $procmux$23\_Y, Q = \result, rval = 8'00000000).

4.17.4. Executing OPT\_CLEAN pass (remove unused cells and wires).

Finding unused cells or wires in module \controller..

Removed 1 unused cells and 2 unused wires.

<suppressed ~2 debug messages>

4.17.5. Rerunning OPT passes. (Removed registers in this run.)

4.17.6. Executing OPT\_EXPR pass (perform const folding).

Optimizing module controller.

4.17.7. Executing OPT\_MERGE pass (detect identical cells).

Finding identical cells in module `\controller'.

Removed a total of 0 cells.

4.17.8. Executing OPT\_DFF pass (perform DFF optimizations).

4.17.9. Executing OPT\_CLEAN pass (remove unused cells and wires).

Finding unused cells or wires in module \controller..

4.17.10. Finished fast OPT passes.

4.18. Executing MEMORY\_MAP pass (converting memories to logic and flip-flops).

4.19. Executing OPT pass (performing simple optimizations).

4.19.1. Executing OPT\_EXPR pass (perform const folding).

Optimizing module controller.

4.19.2. Executing OPT\_MERGE pass (detect identical cells).

Finding identical cells in module `\controller'.

Removed a total of 0 cells.

4.19.3. Executing OPT\_MUXTREE pass (detect dead branches in mux trees).

Running muxtree optimizer on module \controller..

Creating internal representation of mux trees.

Evaluating internal representation of mux trees.

Analyzing evaluation results.

Removed 0 multiplexer ports.

<suppressed ~1 debug messages>

4.19.4. Executing OPT\_REDUCE pass (consolidate $\*mux and $reduce\_\* inputs).

Optimizing cells in module \controller.

Performed a total of 0 changes.

4.19.5. Executing OPT\_MERGE pass (detect identical cells).

Finding identical cells in module `\controller'.

Removed a total of 0 cells.

4.19.6. Executing OPT\_SHARE pass.

4.19.7. Executing OPT\_DFF pass (perform DFF optimizations).

4.19.8. Executing OPT\_CLEAN pass (remove unused cells and wires).

Finding unused cells or wires in module \controller..

4.19.9. Executing OPT\_EXPR pass (perform const folding).

Optimizing module controller.

4.19.10. Finished OPT passes. (There is nothing left to do.)

4.20. Executing TECHMAP pass (map to technology primitives).

4.20.1. Executing Verilog-2005 frontend: C:\Users\Lenovo\DOWNLO~1\OSS-CA~1\bin\../share/yosys/techmap.v

Parsing Verilog input from `C:\Users\Lenovo\DOWNLO~1\OSS-CA~1\bin\../share/yosys/techmap.v' to AST representation.

Generating RTLIL representation for module `\\_90\_simplemap\_bool\_ops'.

Generating RTLIL representation for module `\\_90\_simplemap\_reduce\_ops'.

Generating RTLIL representation for module `\\_90\_simplemap\_logic\_ops'.

Generating RTLIL representation for module `\\_90\_simplemap\_compare\_ops'.

Generating RTLIL representation for module `\\_90\_simplemap\_various'.

Generating RTLIL representation for module `\\_90\_simplemap\_registers'.

Generating RTLIL representation for module `\\_90\_shift\_ops\_shr\_shl\_sshl\_sshr'.

Generating RTLIL representation for module `\\_90\_shift\_shiftx'.

Generating RTLIL representation for module `\\_90\_fa'.

Generating RTLIL representation for module `\\_90\_lcu\_brent\_kung'.

Generating RTLIL representation for module `\\_90\_alu'.

Generating RTLIL representation for module `\\_90\_macc'.

Generating RTLIL representation for module `\\_90\_alumacc'.

Generating RTLIL representation for module `\$\_\_div\_mod\_u'.

Generating RTLIL representation for module `\$\_\_div\_mod\_trunc'.

Generating RTLIL representation for module `\\_90\_div'.

Generating RTLIL representation for module `\\_90\_mod'.

Generating RTLIL representation for module `\$\_\_div\_mod\_floor'.

Generating RTLIL representation for module `\\_90\_divfloor'.

Generating RTLIL representation for module `\\_90\_modfloor'.

Generating RTLIL representation for module `\\_90\_pow'.

Generating RTLIL representation for module `\\_90\_pmux'.

Generating RTLIL representation for module `\\_90\_demux'.

Generating RTLIL representation for module `\\_90\_lut'.

Successfully finished Verilog frontend.

4.20.2. Continuing TECHMAP pass.

Using extmapper simplemap for cells of type $reduce\_and.

Using extmapper simplemap for cells of type $or.

Using extmapper simplemap for cells of type $reduce\_or.

Using extmapper simplemap for cells of type $dffe.

Using extmapper simplemap for cells of type $eq.

Using extmapper simplemap for cells of type $ne.

Using extmapper simplemap for cells of type $and.

Using extmapper simplemap for cells of type $mux.

Using extmapper simplemap for cells of type $not.

Using extmapper simplemap for cells of type $reduce\_bool.

Using extmapper simplemap for cells of type $sdffce.

Using extmapper simplemap for cells of type $dff.

No more expansions possible.

<suppressed ~93 debug messages>

4.21. Executing OPT pass (performing simple optimizations).

4.21.1. Executing OPT\_EXPR pass (perform const folding).

Optimizing module controller.

<suppressed ~6 debug messages>

4.21.2. Executing OPT\_MERGE pass (detect identical cells).

Finding identical cells in module `\controller'.

<suppressed ~9 debug messages>

Removed a total of 3 cells.

4.21.3. Executing OPT\_DFF pass (perform DFF optimizations).

4.21.4. Executing OPT\_CLEAN pass (remove unused cells and wires).

Finding unused cells or wires in module \controller..

Removed 7 unused cells and 14 unused wires.

<suppressed ~8 debug messages>

4.21.5. Finished fast OPT passes.

4.22. Executing ABC pass (technology mapping using ABC).

4.22.1. Extracting gate netlist of module `\controller' to `<abc-temp-dir>/input.blif'..

Extracted 27 gates and 49 wires to a netlist network with 22 inputs and 15 outputs.

4.22.1.1. Executing ABC.

Running ABC command: "<yosys-exe-dir>/yosys-abc" -s -f <abc-temp-dir>/abc.script 2>&1

ABC: ABC command line: "source <abc-temp-dir>/abc.script".

ABC:

ABC: + read\_blif <abc-temp-dir>/input.blif

ABC: + read\_library <abc-temp-dir>/stdcells.genlib

ABC: Entered genlib library with 13 gates from file "<abc-temp-dir>/stdcells.genlib".

ABC: + strash

ABC: + dretime

ABC: + map

ABC: + write\_blif <abc-temp-dir>/output.blif

4.22.1.2. Re-integrating ABC results.

ABC RESULTS: NOT cells: 2

ABC RESULTS: ANDNOT cells: 1

ABC RESULTS: NAND cells: 3

ABC RESULTS: NOR cells: 5

ABC RESULTS: ORNOT cells: 4

ABC RESULTS: OR cells: 8

ABC RESULTS: AND cells: 1

ABC RESULTS: internal signals: 12

ABC RESULTS: input signals: 22

ABC RESULTS: output signals: 15

Removing temp directory.

4.23. Executing OPT pass (performing simple optimizations).

4.23.1. Executing OPT\_EXPR pass (perform const folding).

Optimizing module controller.

4.23.2. Executing OPT\_MERGE pass (detect identical cells).

Finding identical cells in module `\controller'.

Removed a total of 0 cells.

4.23.3. Executing OPT\_DFF pass (perform DFF optimizations).

4.23.4. Executing OPT\_CLEAN pass (remove unused cells and wires).

Finding unused cells or wires in module \controller..

Removed 0 unused cells and 47 unused wires.

<suppressed ~1 debug messages>

4.23.5. Finished fast OPT passes.

4.24. Executing HIERARCHY pass (managing design hierarchy).

4.24.1. Analyzing design hierarchy..

Top module: \controller

4.24.2. Analyzing design hierarchy..

Top module: \controller

Removed 0 unused modules.

4.25. Printing statistics.

=== controller ===

Number of wires: 25

Number of wire bits: 49

Number of public wires: 9

Number of public wire bits: 33

Number of ports: 7

Number of port bits: 21

Number of memories: 0

Number of memory bits: 0

Number of processes: 0

Number of cells: 44

$\_ANDNOT\_ 1

$\_AND\_ 1

$\_DFFE\_PP\_ 1

$\_DFF\_P\_ 4

$\_NAND\_ 3

$\_NOR\_ 5

$\_NOT\_ 2

$\_ORNOT\_ 4

$\_OR\_ 8

$\_SDFFCE\_PN0P\_ 15

4.26. Executing CHECK pass (checking for obvious problems).

Checking module controller...

Found and reported 0 problems.