

Week 1 RISC V Tape-Out Report on Yosys Tool

By

Dr M A. Raheem,



Department of Electronics and Communication Engineering

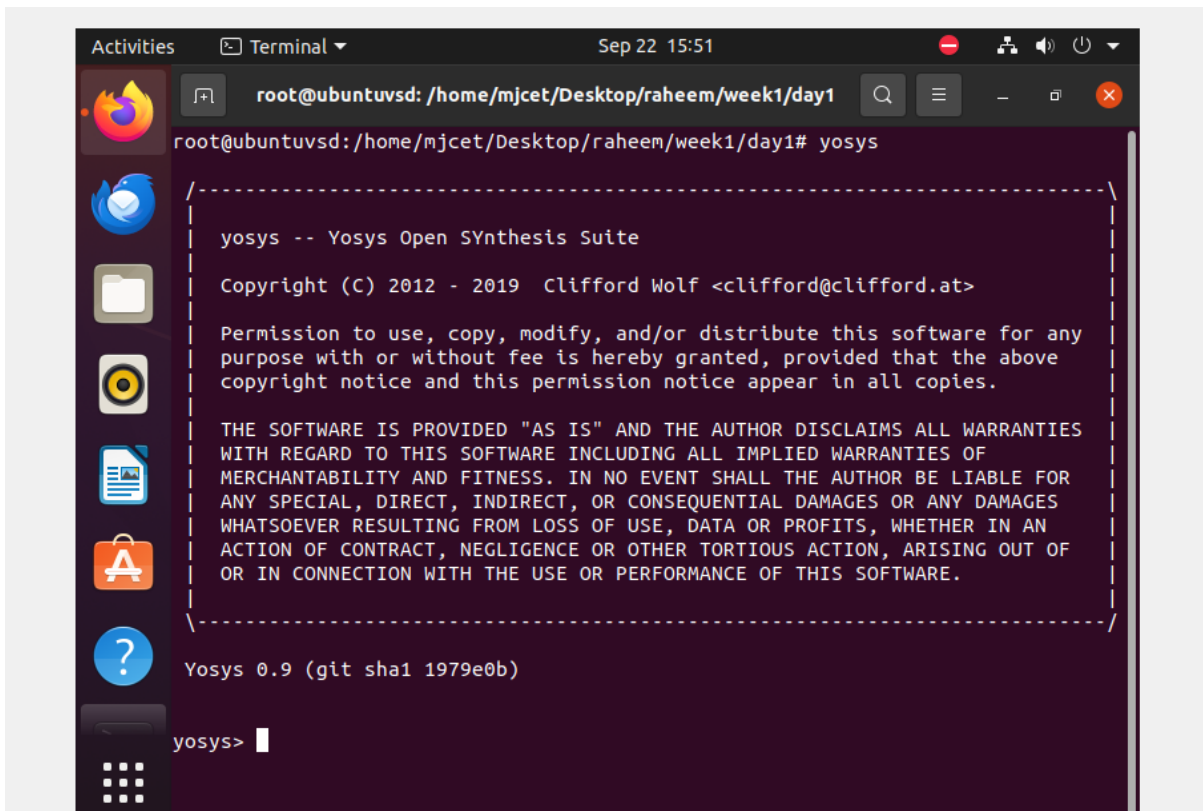
Muffakham Jah College of Engineering and Technology

Banjara Hills, Hyderabad-500 034

Your directory should be something like `.../verilog_files/`, containing both the `lib` and `verilog_files` directories (from a cloned repository). All standard cell libraries reside in `lib`

Open Yosys from the terminal.

```
yosys
```



```
Activities Terminal Sep 22 15:51
root@ubuntuvsd: /home/mjcet/Desktop/raheem/week1/day1
root@ubuntuvsd: /home/mjcet/Desktop/raheem/week1/day1# yosys

/-----/
yosys -- Yosys Open SYnthesis Suite

Copyright (C) 2012 - 2019 Clifford Wolf <clifford@clifford.at>

Permission to use, copy, modify, and/or distribute this software for any
purpose with or without fee is hereby granted, provided that the above
copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
/-----/

Yosys 0.9 (git sha1 1979e0b)

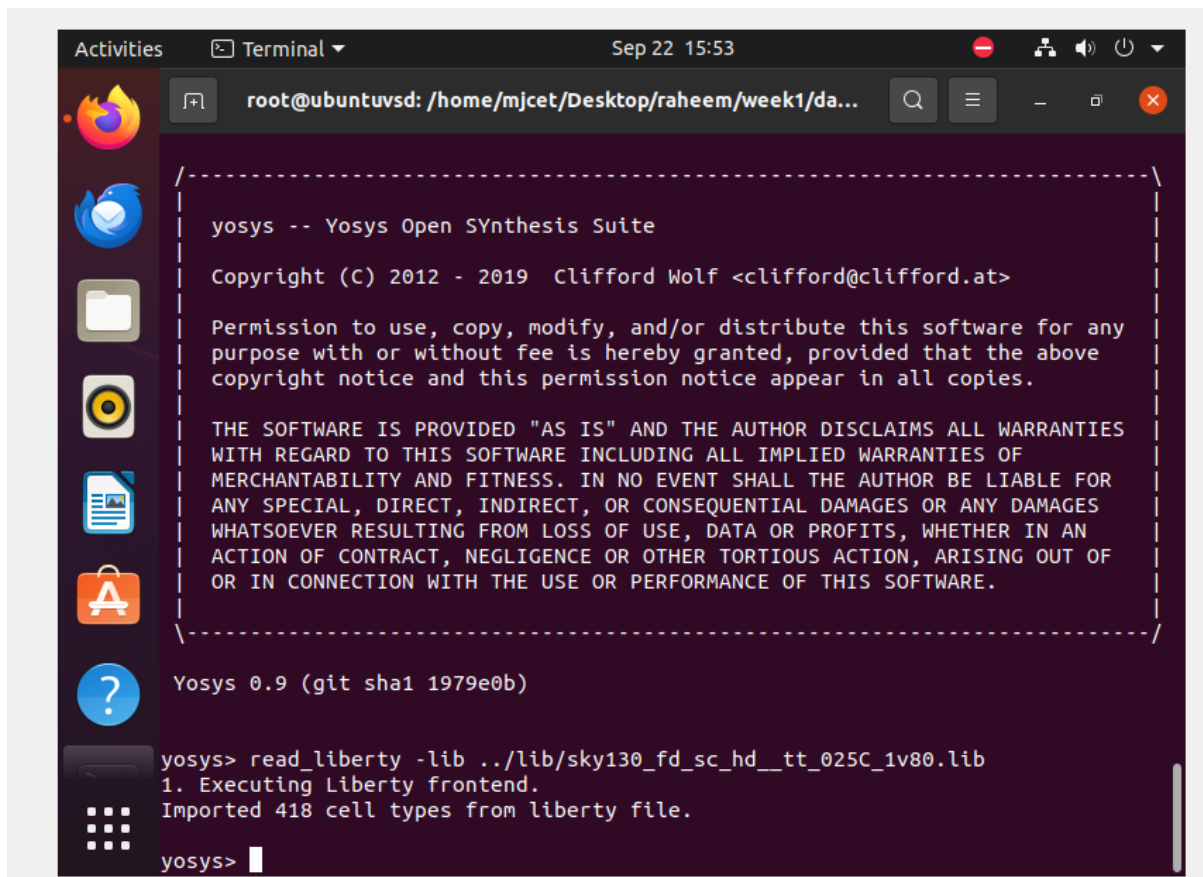
yosys> 
```

Read the standard cell library for synthesis. Paths may be absolute or relative; here we use a relative path:

```
read_liberty -lib ../lib/sky130_fd_sc_hd__tt_025C_1v80.lib
```

Read the standard cell library for synthesis. Paths may be absolute or relative; here we use a relative path:

```
read_liberty -lib ../lib/sky130_fd_sc_hd__tt_025C_1v80.lib
```



```
Activities Terminal Sep 22 15:53
root@ubuntuvsd: /home/mjcet/Desktop/raheem/week1/da...

-----
yosys -- Yosys Open SYnthesis Suite

Copyright (C) 2012 - 2019 Clifford Wolf <clifford@clifford.at>

Permission to use, copy, modify, and/or distribute this software for any
purpose with or without fee is hereby granted, provided that the above
copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
-----

Yosys 0.9 (git sha1 1979e0b)

yosys> read_liberty -lib ../lib/sky130_fd_sc_hd__tt_025C_1v80.lib
1. Executing Liberty frontend.
Imported 418 cell types from liberty file.

yosys>
```

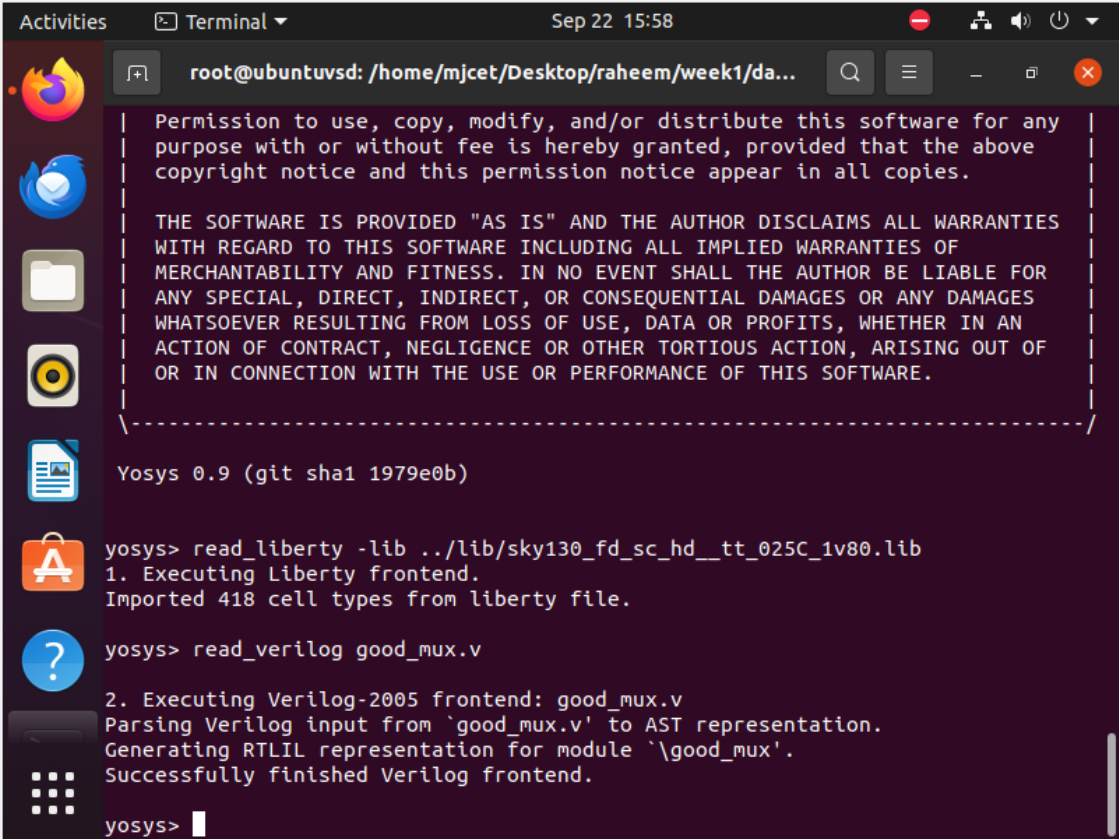
Read your Verilog design (in this case: good_mux.v):

```
read_verilog good_mux.v
```

On success, Yosys should report:

Successfully finished Verilog frontend.

If your design consists of multiple files, repeat the `read_verilog` command as needed for each file.



The screenshot shows a terminal window titled 'Terminal' with the date and time 'Sep 22 15:58'. The user is logged in as 'root' on an 'ubuntu' machine, with the current directory being '/home/mjcet/Desktop/raheem/week1/da...'. The terminal displays the following content:

```
Permission to use, copy, modify, and/or distribute this software for any
purpose with or without fee is hereby granted, provided that the above
copyright notice and this permission notice appear in all copies.

THE SOFTWARE IS PROVIDED "AS IS" AND THE AUTHOR DISCLAIMS ALL WARRANTIES
WITH REGARD TO THIS SOFTWARE INCLUDING ALL IMPLIED WARRANTIES OF
MERCHANTABILITY AND FITNESS. IN NO EVENT SHALL THE AUTHOR BE LIABLE FOR
ANY SPECIAL, DIRECT, INDIRECT, OR CONSEQUENTIAL DAMAGES OR ANY DAMAGES
WHATSOEVER RESULTING FROM LOSS OF USE, DATA OR PROFITS, WHETHER IN AN
ACTION OF CONTRACT, NEGLIGENCE OR OTHER TORTIOUS ACTION, ARISING OUT OF
OR IN CONNECTION WITH THE USE OR PERFORMANCE OF THIS SOFTWARE.
-----
Yosys 0.9 (git sha1 1979e0b)

yosys> read_liberty -lib ../lib/sky130_fd_sc_hd__tt_025C_1v80.lib
1. Executing Liberty frontend.
Imported 418 cell types from liberty file.

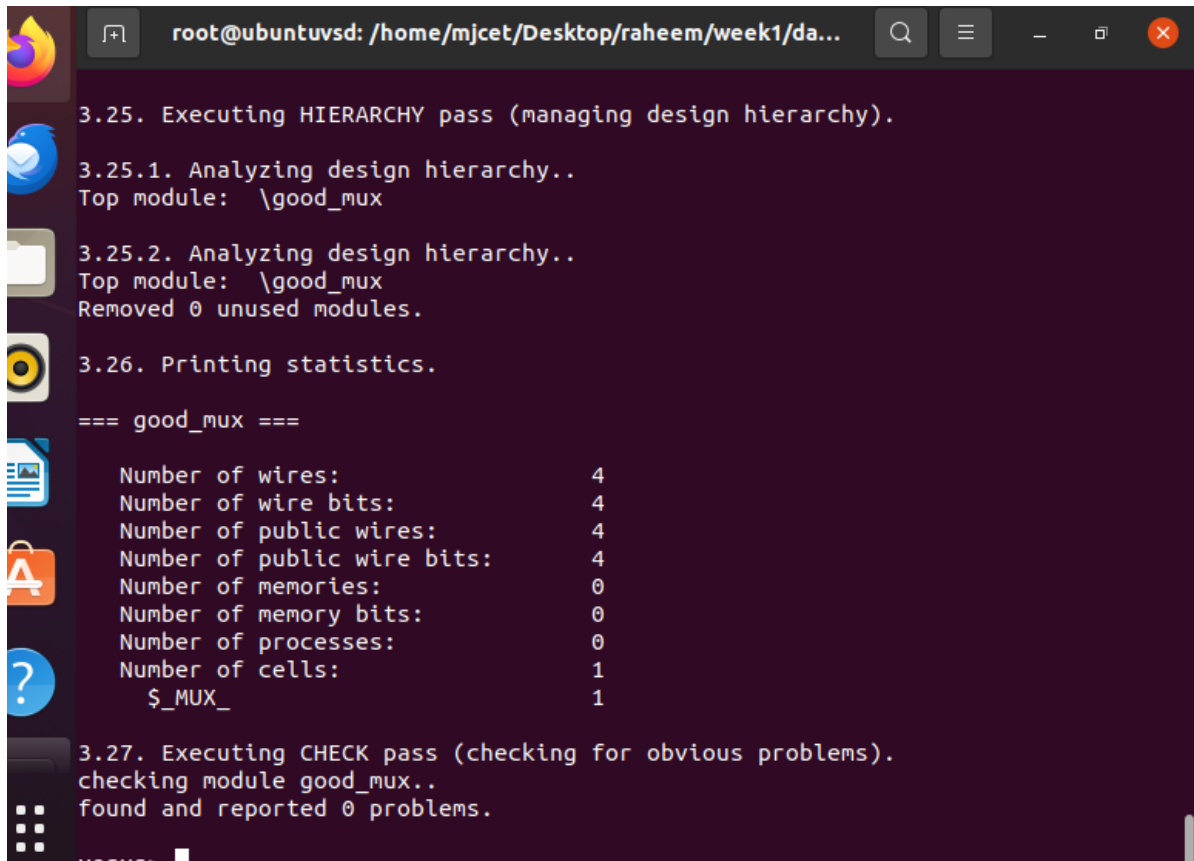
yosys> read_verilog good_mux.v

2. Executing Verilog-2005 frontend: good_mux.v
Parsing Verilog input from `good_mux.v' to AST representation.
Generating RTLIL representation for module `good_mux'.
Successfully finished Verilog frontend.

yosys> 
```

Specify the module to synthesize using the `synth` flow. In our example, the module name is `good_mux`:

```
synth -top good_mux
```

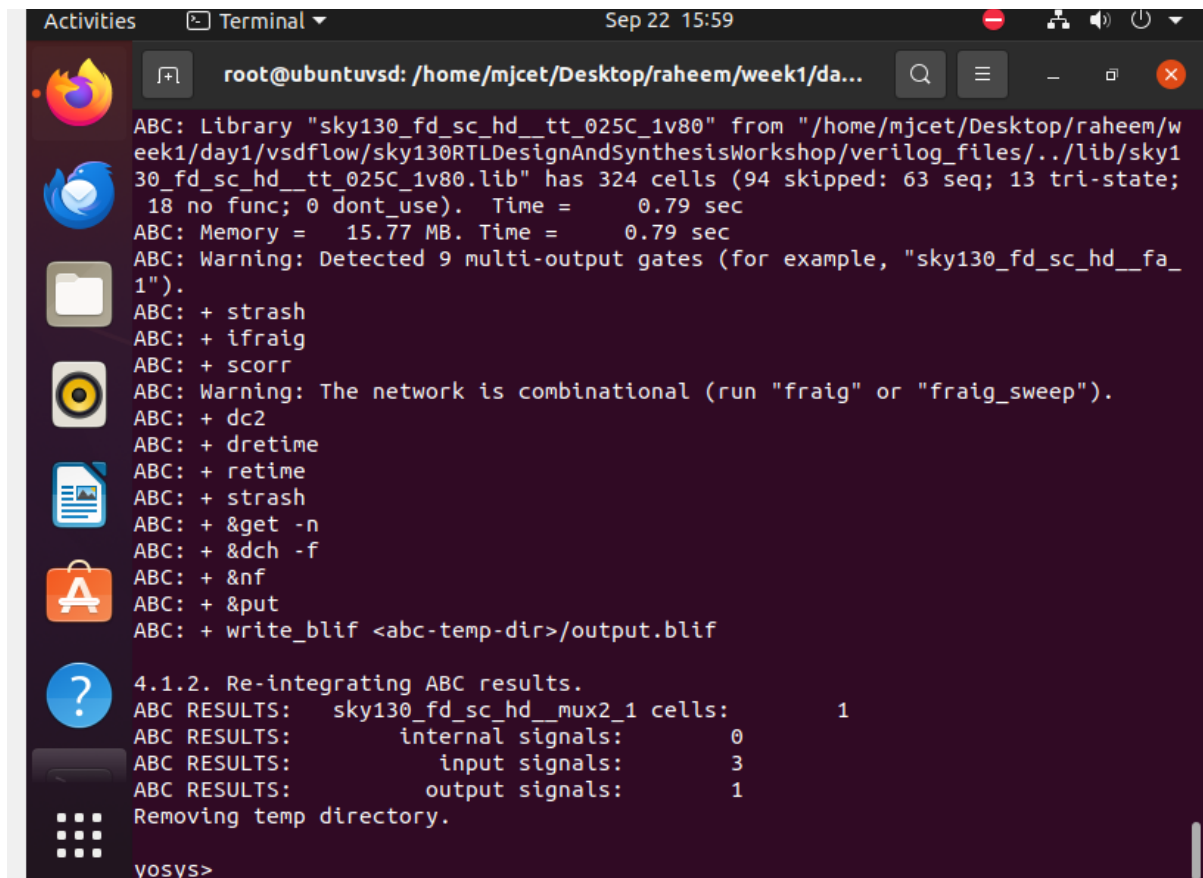


A terminal window titled 'root@ubuntuvsd: /home/mjcet/Desktop/raheem/week1/da...' displays the output of a synthesis process. The output shows the execution of the HIERARCHY pass, including analyzing the design hierarchy and printing statistics for the 'good_mux' module. The statistics table indicates 4 wires, 4 wire bits, 4 public wires, 4 public wire bits, 0 memories, 0 memory bits, 0 processes, 1 cell, and 1 _MUX_.

```
3.25. Executing HIERARCHY pass (managing design hierarchy).
3.25.1. Analyzing design hierarchy..
Top module: \good_mux
3.25.2. Analyzing design hierarchy..
Top module: \good_mux
Removed 0 unused modules.
3.26. Printing statistics.
=== good_mux ===
Number of wires:          4
Number of wire bits:      4
Number of public wires:   4
Number of public wire bits: 4
Number of memories:       0
Number of memory bits:    0
Number of processes:      0
Number of cells:          1
    _MUX_                  1
3.27. Executing CHECK pass (checking for obvious problems).
checking module good_mux..
found and reported 0 problems.
```

Run technology mapping to your chosen standard cell library:

```
abc -liberty ../lib/sky130_fd_sc_hd__tt_025C_1v80.lib
```



```
root@ubuntuvsd: /home/mjcet/Desktop/raheem/week1/day1/vsdf/flow/sky130RTLDesignAndSynthesisWorkshop/verilog_files/..
ABC: Library "sky130_fd_sc_hd__tt_025C_1v80" from "/home/mjcet/Desktop/raheem/week1/day1/vsdf/flow/sky130RTLDesignAndSynthesisWorkshop/verilog_files/..lib/sky130_fd_sc_hd__tt_025C_1v80.lib" has 324 cells (94 skipped: 63 seq; 13 tri-state; 18 no func; 0 dont_use). Time = 0.79 sec
ABC: Memory = 15.77 MB. Time = 0.79 sec
ABC: Warning: Detected 9 multi-output gates (for example, "sky130_fd_sc_hd__fa_1").
ABC: + strash
ABC: + ifraig
ABC: + scorrr
ABC: Warning: The network is combinational (run "fraig" or "fraig_sweep").
ABC: + dc2
ABC: + dretime
ABC: + retime
ABC: + strash
ABC: + &get -n
ABC: + &dch -f
ABC: + &nf
ABC: + &put
ABC: + write_blif <abc-temp-dir>/output.blif

4.1.2. Re-integrating ABC results.
ABC RESULTS: sky130_fd_sc_hd_mux2_1 cells: 1
ABC RESULTS: internal signals: 0
ABC RESULTS: input signals: 3
ABC RESULTS: output signals: 1
Removing temp directory.
yosys>
```

Yosys will print details of inferred cells and IOs, such as:

- Number of input signals: 3
- Number of output signals: 1
- Number of internal signals: 0

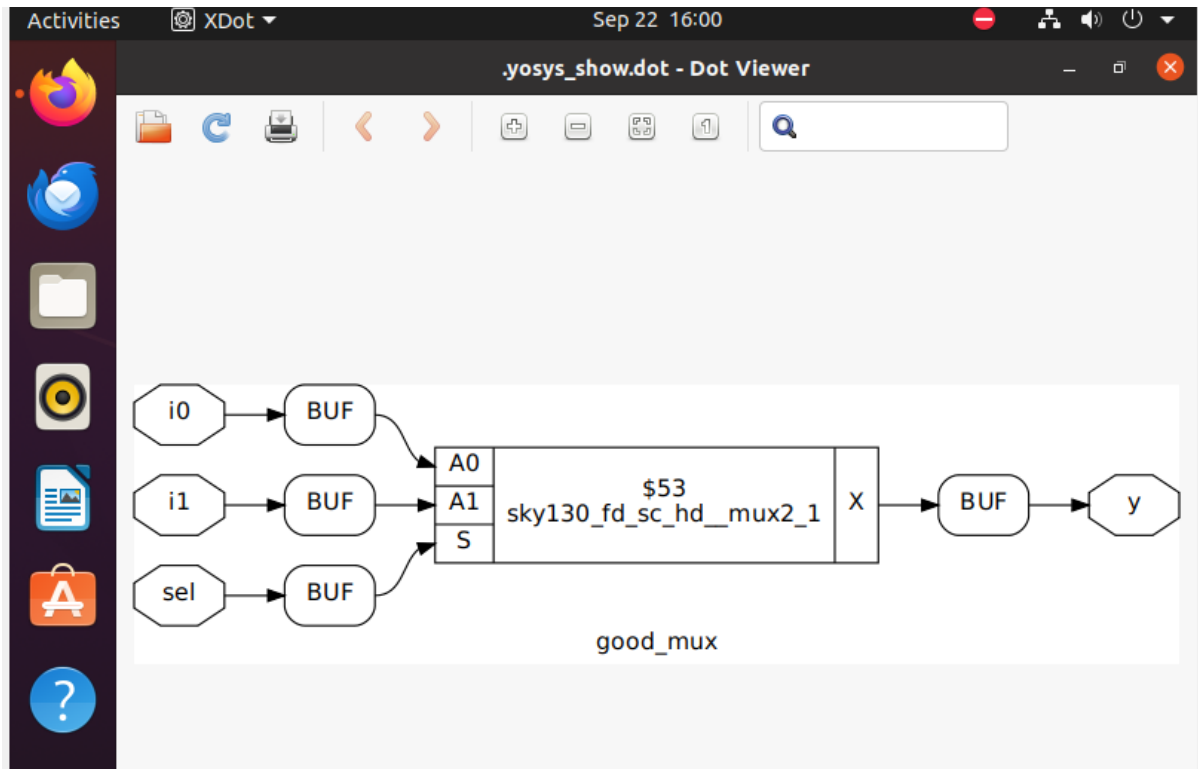
It also lists cell usage, such as:

- 1 clock inverter
- 1 NAND gate
- 1 O2AI gate (OR-AND-Invert) (or)
- 1 mux2_1 (In my case)

Compare these mappings to your original RTL code to ensure expected logic cell usage.

To view the synthesized logic structure:

show

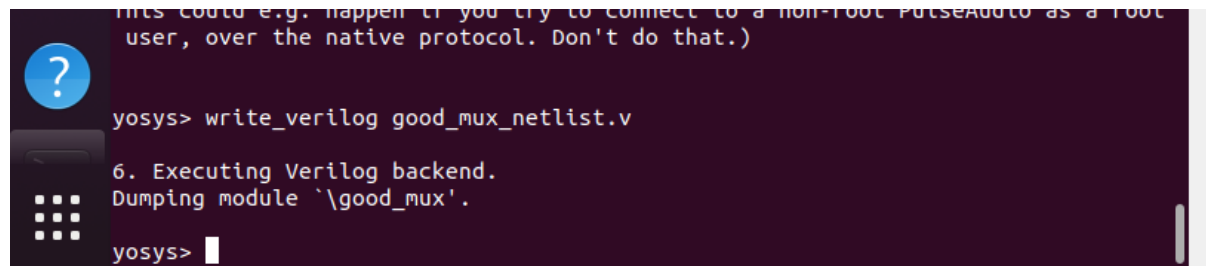


Export the synthesized netlist in Verilog format:

```
write_verilog good_mux_netlist.v
```

This initial output may contain attributes and extra info. For a cleaner, attribute-free netlist, use:

```
write_verilog -noattr good_mux_netlist.v
```



```
3 (* top = 1 *)
4 (* src = "good_mux.v:2" *)
5 module good_mux(i0, i1, sel, y);
6   (* src = "good_mux.v:2" *)
7   wire _0_;
8   (* src = "good_mux.v:2" *)
9   wire _1_;
0   (* src = "good_mux.v:2" *)
1   wire _2_;
2   (* src = "good_mux.v:2" *)
3   wire _3_;
4   (* src = "good_mux.v:2" *)
5   input i0;
6   (* src = "good_mux.v:2" *)
7   input i1;
8   (* src = "good_mux.v:2" *)
9   input sel;
0   (* src = "good_mux.v:2" *)
1   output y;
2   sky130_fd_sc_hd__mux2_1_4_ (
3     .A0(_0_),
4     .A1(_1_),
5     .S(_2_),
6     .X(_3_)
7   );
8   assign y = i0;
```

Verilog ▼ Tab Wid


```
opt_checks.v      up_dn_ctr.v
opt_check4.v      up_dn_ctr_with_load.v
opt_check.v        up_dn_ctr_with_load_with_start_stop.v
mjcet@ubuntuvsd:~/Desktop/raheem/week1/day1/vsdf/flow/sky130RTLDesignAndSynthesis
Workshop/verilog_files$ gedit good_mux_netlist.v
mjcet@ubuntuvsd:~/Desktop/raheem/week1/day1/vsdf/flow/sky130RTLDesignAndSynthesis
Workshop/verilog_files$ gedit good_mux_netlist.v
mjcet@ubuntuvsd:~/Desktop/raheem/week1/day1/vsdf/flow/sky130RTLDesignAndSynthesis
Workshop/verilog_files$
```

```
XDG_RUNTIME_DIR (/run/user/1000) is not owned by us
This could e.g. happen if you try to connect to a no
user, over the native protocol. Don't do that.)

yosys> write_verilog good_mux_netlist.v

6. Executing Verilog backend.
Dumping module `good_mux'.

yosys> write_verilog -noattr good_mux_netlist.v

7. Executing Verilog backend.
Dumping module `good_mux'.

yosys>
```

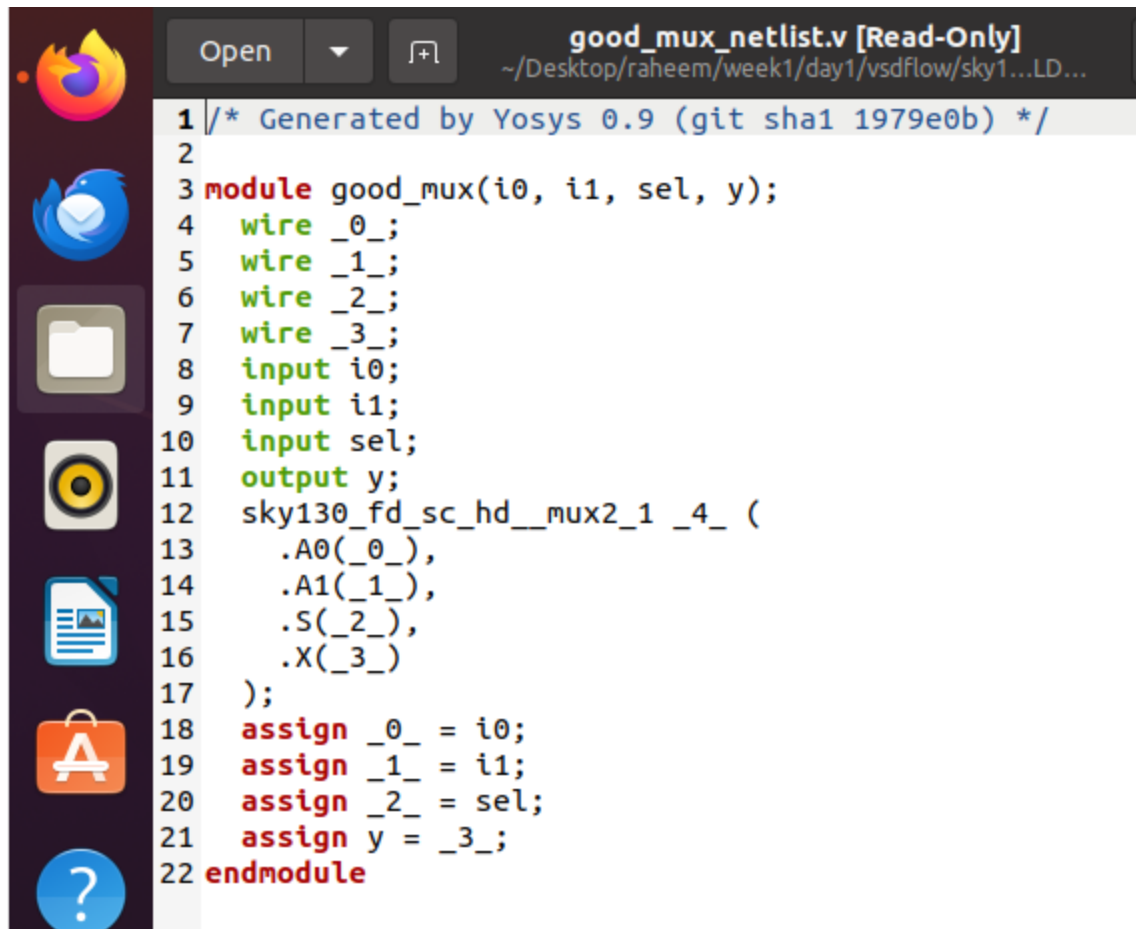
Open and inspect your netlist (example using gvim):

```
Gedit good_mux_netlist.v
```

Netlist Structure

- Module name at top matches what was set in `synth -top`.
- Instantiations for each logic gate (inverter, NAND, O2AI).
- Signals (nets) internally connect gate outputs to subsequent gate inputs.
- Primary inputs and outputs clearly mapped to original Verilog I/O.

Trace through each net in your netlist to verify logic connectivity as seen in the schematic.



The image shows a screenshot of an Ubuntu desktop environment. On the left side, there is a vertical dock with several application icons: Firefox, Thunderbird, a file manager, a CD/DVD player, a document viewer, an application store, and a help icon. The main window is a terminal or code editor titled "good_mux_netlist.v [Read-Only]". The address bar shows the file path: "~/Desktop/raheem/week1/day1/vsdflow/sky1...LD...". The code is Verilog, generated by Yosys 0.9 (git sha1 1979e0b). It defines a module named "good_mux" with inputs "i0", "i1", and "sel", and an output "y". The module contains four wires, three inputs, and one output. It instantiates a component "sky130_fd_sc_hd__mux2_1_4_" with specific port connections. Finally, it assigns the values of the inputs to the wires and the output to the output signal.

```
1 /* Generated by Yosys 0.9 (git sha1 1979e0b) */
2
3 module good_mux(i0, i1, sel, y);
4     wire _0_;
5     wire _1_;
6     wire _2_;
7     wire _3_;
8     input i0;
9     input i1;
10    input sel;
11    output y;
12    sky130_fd_sc_hd__mux2_1_4_ (
13        .A0(_0_),
14        .A1(_1_),
15        .S(_2_),
16        .X(_3_)
17    );
18    assign _0_ = i0;
19    assign _1_ = i1;
20    assign _2_ = sel;
21    assign y = _3_;
22 endmodule
```