WEEK:1
RTL Simulation using Iverilog and GTKWave

Week 1 Overview – RISC-V SoC Tapeout Program
Week 1 lays the foundation for RTL design and simulation, focusing on Verilog HDL, standard cell libraries, and open-source EDA workflows. By the end of the week, you'll have a solid grasp of design entry, testbench creation, synthesis basics, and waveform-based debugging.

Week 1 Schedule

- **Day 1 (Done):** RTL design basics & simulation environment setup
- **Day 2 (Next):** Logic synthesis & technology mapping
- **Day 3:** Combinational and sequential optimizations
- **Day 4:** Gate-level simulation & handling synthesis–simulation mismatches
- **Day 5:** Design-for-testability and `if`/`case` constructs

RTL Design & Verification Concepts

Before coding, it's important to understand the building blocks of RTL verification:

Testbench Architecture

A **testbench** validates your design by applying inputs and checking outputs. Unlike the DUT (Design Under Test), the testbench itself has no "primary inputs" or "outputs" — it's a self-contained verification environment.

- **Stimulus Generator**
  - Drives input patterns and control signals to the DUT
  - Ensures functional coverage with varied test cases
  - Manages timing and sequencing
- **Design Under Test (DUT)**
  - The RTL module being verified
  - Receives stimuli and produces outputs
- **Output Monitor/Checker**
  - Observes DUT outputs
  - Compares them against expected results
  - Flags mismatches and reports verification status

This separation keeps design and verification independent and easier to manage.

Icarus Verilog Simulation Flow

Open-source tools like **Icarus Verilog (iverilog)** and **GTKWave** provide a professional-grade workflow for RTL verification:

1. **Design & Testbench Input**
    - RTL code (`.v`) and testbench files are provided to the compiler.
2. **Compilation with `iverilog`**
    - Syntax and semantic checks
    - Generates a simulation executable
3. **Simulation & VCD Generation**
    - Executable runs the testbench
    - Signal changes are logged into a **VCD (Value Change Dump)** file
4. **Waveform Analysis in GTKWave**
    - Graphical view of signal activity
    - Timing inspection and debugging
    - Supports zooming, measurement, and interactive exploration

---

Why This Flow Matters

- **Thorough Debugging:** Full visibility of all signals
- **Timing Accuracy:** Matches RTL behavior closely
- **Tool Interoperability:** Uses industry-standard VCD format
- **Intuitive Analysis:** Visual waveforms simplify understanding

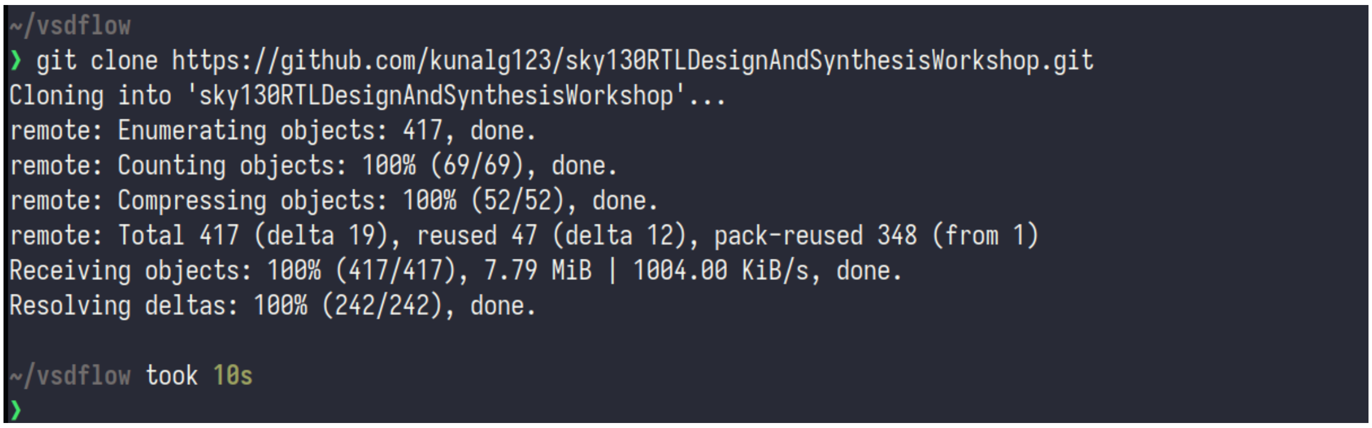
---

Key Takeaways

- RTL design must follow structured coding practices.
- A well-designed testbench ensures comprehensive coverage.
- Simulation and waveform analysis validate correctness before synthesis.
- This disciplined approach shortens debug time and increases confidence in results.

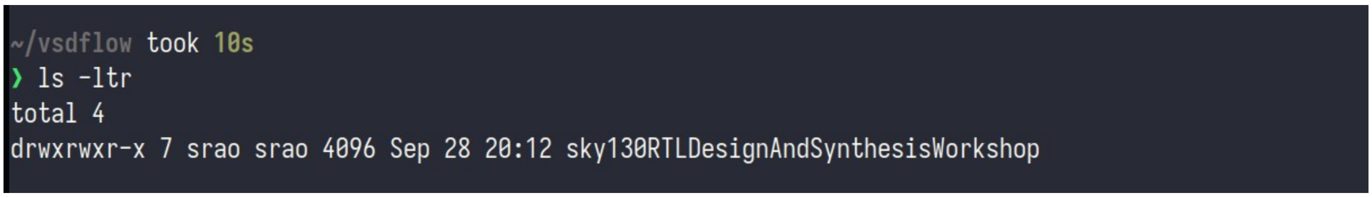
PROOF OF WORK:

#1: CLONE REPO INTO FOLDER: vsdflow

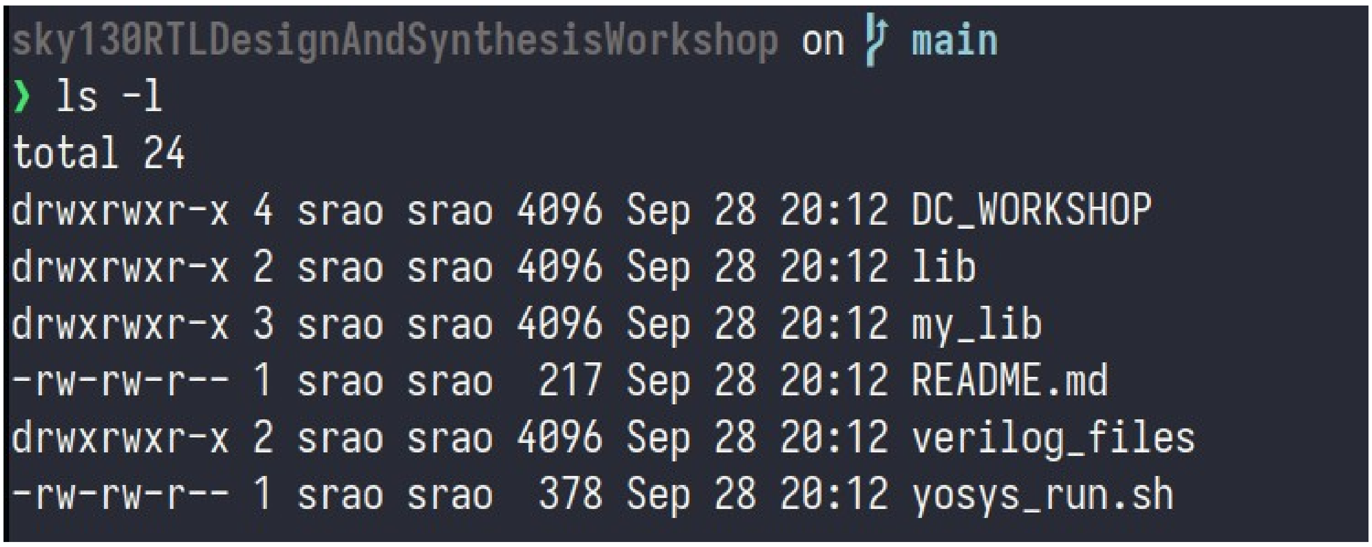


```
~/vsdflow
❯ git clone https://github.com/kunalg123/sky130RTLDesignAndSynthesisWorkshop.git
Cloning into 'sky130RTLDesignAndSynthesisWorkshop'...
remote: Enumerating objects: 417, done.
remote: Counting objects: 100% (69/69), done.
remote: Compressing objects: 100% (52/52), done.
remote: Total 417 (delta 19), reused 47 (delta 12), pack-reused 348 (from 1)
Receiving objects: 100% (417/417), 7.79 MiB | 1004.00 KiB/s, done.
Resolving deltas: 100% (242/242), done.

~/vsdflow took 10s
❯
```

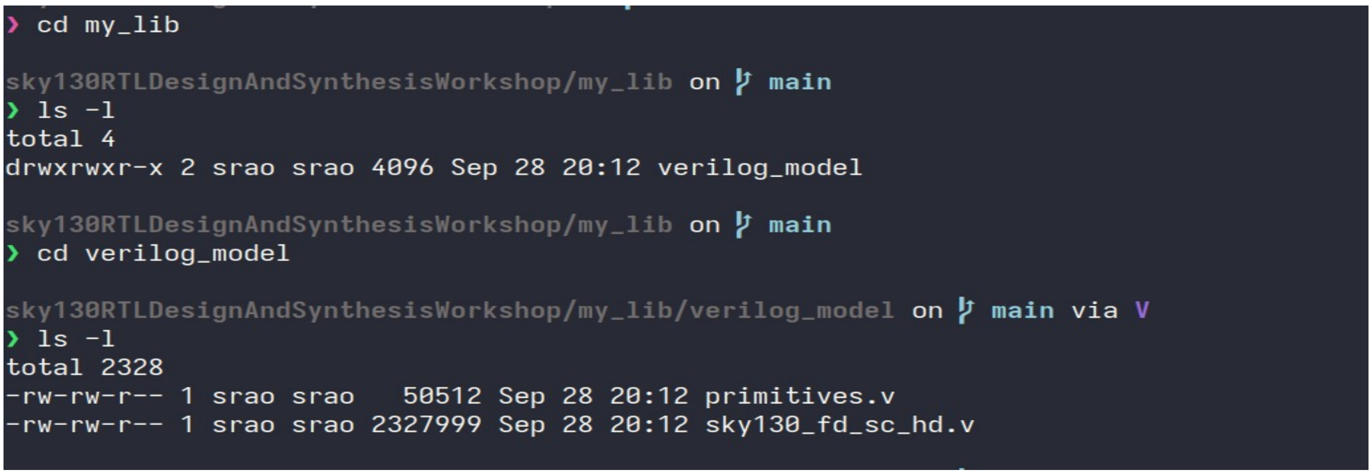
#ANALYSE THE FOLDER



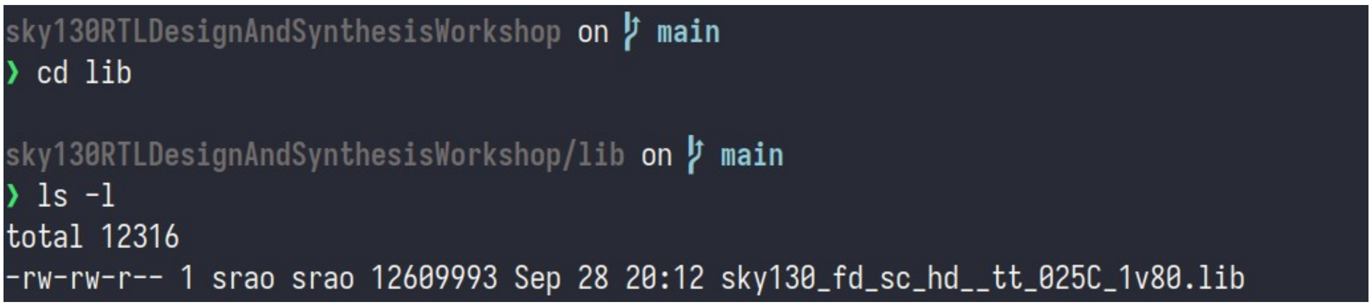
```
~/vsdflow took 10s
❯ ls -ltr
total 4
drwxrwxr-x 7 srao srao 4096 Sep 28 20:12 sky130RTLDesignAndSynthesisWorkshop
```



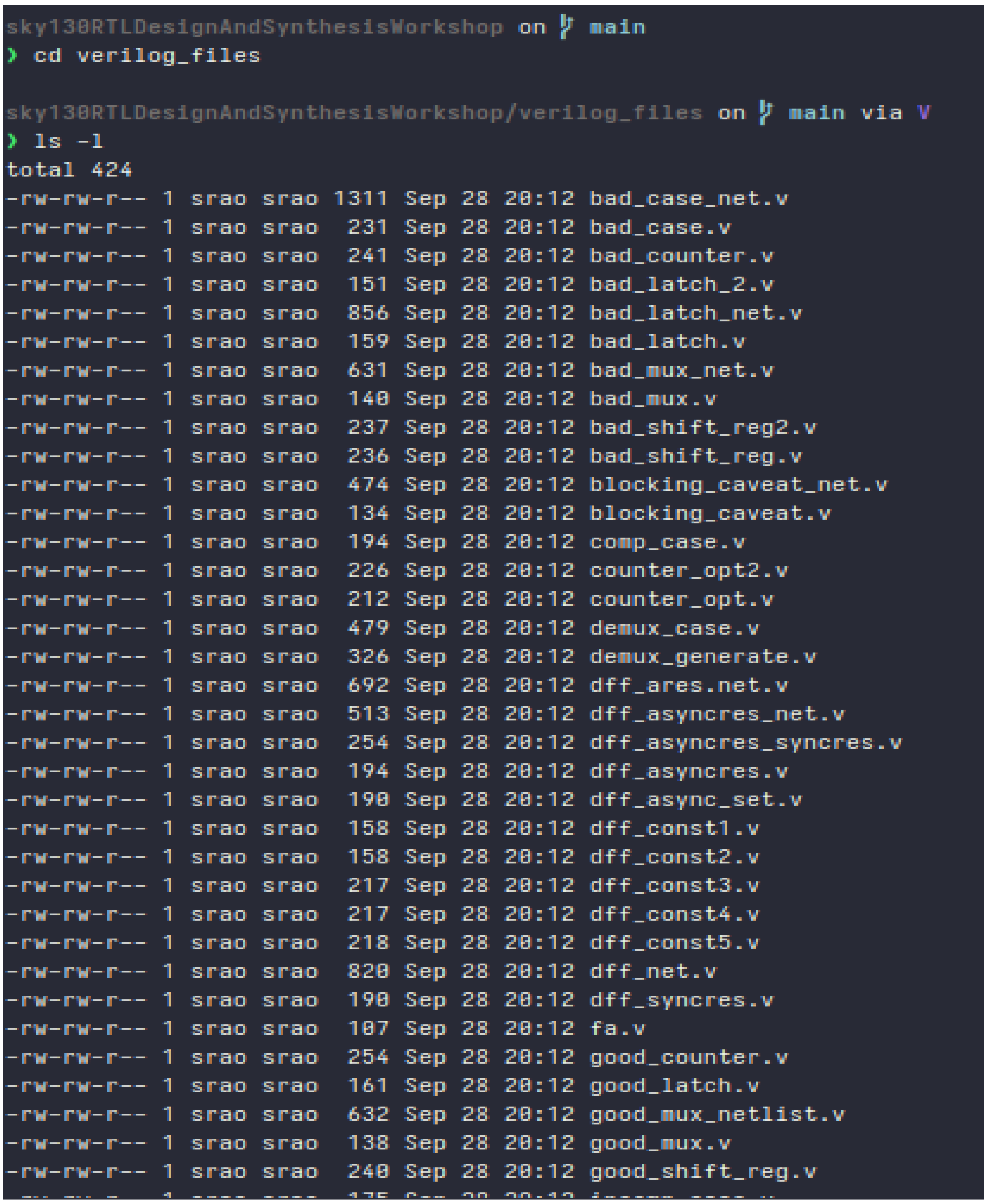
```
sky130RTLDesignAndSynthesisWorkshop on  main
❯ ls -l
total 24
drwxrwxr-x 4 srao srao 4096 Sep 28 20:12 DC_WORKSHOP
drwxrwxr-x 2 srao srao 4096 Sep 28 20:12 lib
drwxrwxr-x 3 srao srao 4096 Sep 28 20:12 my_lib
-rw-rw-r-- 1 srao srao  217 Sep 28 20:12 README.md
drwxrwxr-x 2 srao srao 4096 Sep 28 20:12 verilog_files
-rw-rw-r-- 1 srao srao  378 Sep 28 20:12 yosys_run.sh
```



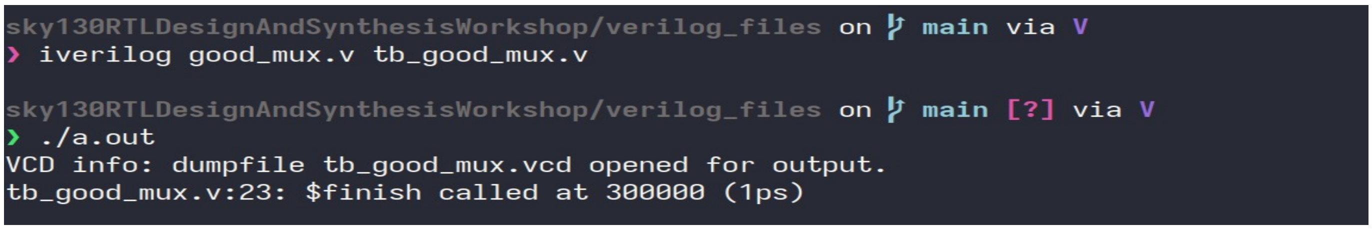
```
❯ cd my_lib

sky130RTLDesignAndSynthesisWorkshop/my_lib on  main
❯ ls -l
total 4
drwxrwxr-x 2 srao srao 4096 Sep 28 20:12 verilog_model

sky130RTLDesignAndSynthesisWorkshop/my_lib on  main
❯ cd verilog_model

sky130RTLDesignAndSynthesisWorkshop/my_lib/verilog_model on  main via V
❯ ls -l
total 2328
-rw-rw-r-- 1 srao srao   50512 Sep 28 20:12 primitives.v
-rw-rw-r-- 1 srao srao 2327999 Sep 28 20:12 sky130_fd_sc_hd.v
```



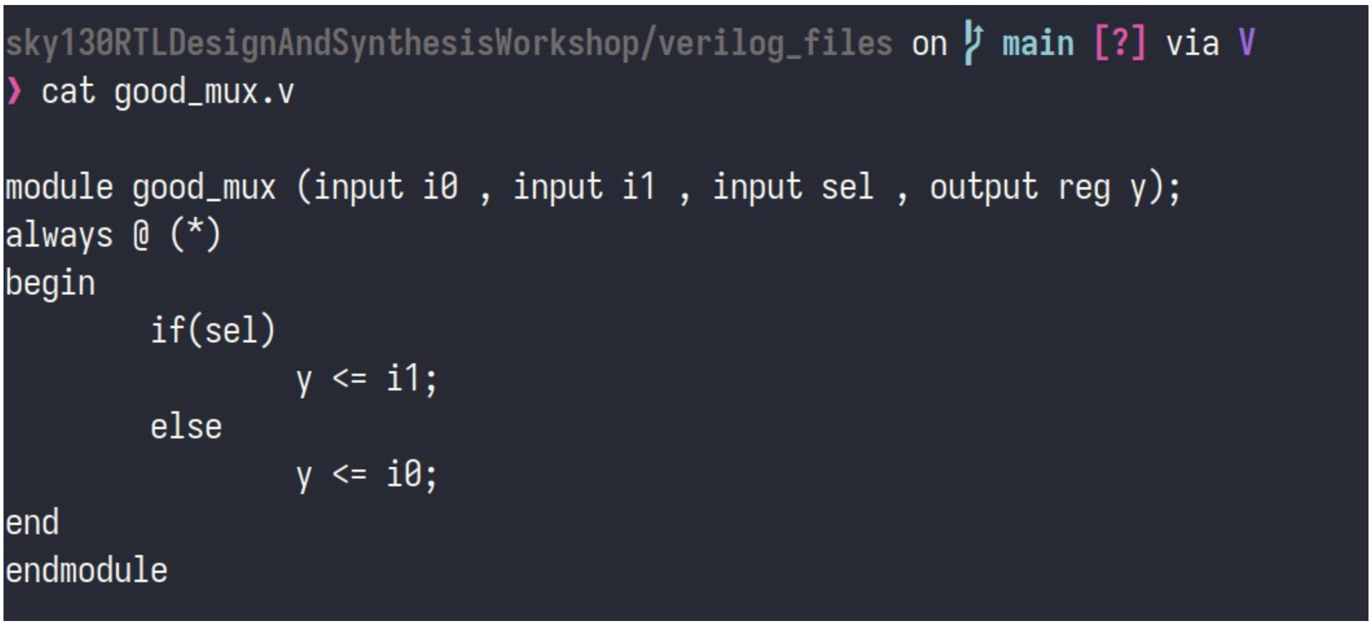
```
sky130RTLDesignAndSynthesisWorkshop on  main
❯ cd lib

sky130RTLDesignAndSynthesisWorkshop/lib on  main
❯ ls -l
total 12316
-rw-rw-r-- 1 srao srao 12609993 Sep 28 20:12 sky130_fd_sc_hd__tt_025C_1v80.lib
```



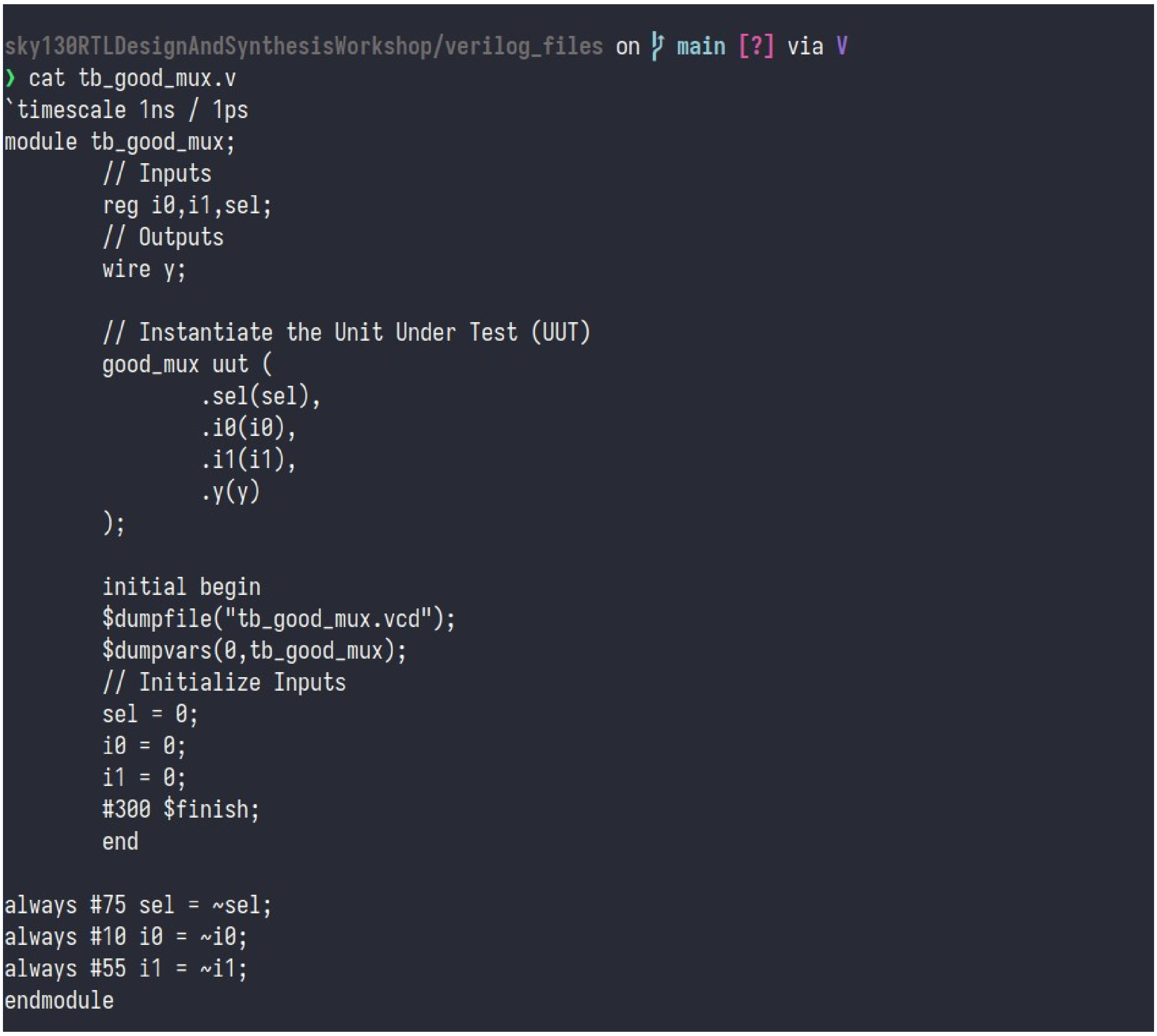
```
sky130RTLDesignAndSynthesisWorkshop on  main
❯ cd verilog_files

sky130RTLDesignAndSynthesisWorkshop/verilog_files on  main via V
❯ ls -l
total 424
-rw-rw-r-- 1 srao srao 1311 Sep 28 20:12 bad_case_net.v
-rw-rw-r-- 1 srao srao  231 Sep 28 20:12 bad_case.v
-rw-rw-r-- 1 srao srao  241 Sep 28 20:12 bad_counter.v
-rw-rw-r-- 1 srao srao  151 Sep 28 20:12 bad_latch_2.v
-rw-rw-r-- 1 srao srao  856 Sep 28 20:12 bad_latch_net.v
-rw-rw-r-- 1 srao srao  159 Sep 28 20:12 bad_latch.v
-rw-rw-r-- 1 srao srao  631 Sep 28 20:12 bad_mux_net.v
-rw-rw-r-- 1 srao srao  140 Sep 28 20:12 bad_mux.v
-rw-rw-r-- 1 srao srao  237 Sep 28 20:12 bad_shift_reg2.v
-rw-rw-r-- 1 srao srao  236 Sep 28 20:12 bad_shift_reg.v
-rw-rw-r-- 1 srao srao  474 Sep 28 20:12 blocking_caveat_net.v
-rw-rw-r-- 1 srao srao  134 Sep 28 20:12 blocking_caveat.v
-rw-rw-r-- 1 srao srao  194 Sep 28 20:12 comp_case.v
-rw-rw-r-- 1 srao srao  226 Sep 28 20:12 counter_opt2.v
-rw-rw-r-- 1 srao srao  212 Sep 28 20:12 counter_opt.v
-rw-rw-r-- 1 srao srao  479 Sep 28 20:12 demux_case.v
-rw-rw-r-- 1 srao srao  326 Sep 28 20:12 demux_generate.v
-rw-rw-r-- 1 srao srao  692 Sep 28 20:12 dff_ares.net.v
-rw-rw-r-- 1 srao srao  513 Sep 28 20:12 dff_asyncres_net.v
-rw-rw-r-- 1 srao srao  254 Sep 28 20:12 dff_asyncres_syncres.v
-rw-rw-r-- 1 srao srao  194 Sep 28 20:12 dff_asyncres.v
-rw-rw-r-- 1 srao srao  190 Sep 28 20:12 dff_async_set.v
-rw-rw-r-- 1 srao srao  158 Sep 28 20:12 dff_const1.v
-rw-rw-r-- 1 srao srao  158 Sep 28 20:12 dff_const2.v
-rw-rw-r-- 1 srao srao  217 Sep 28 20:12 dff_const3.v
-rw-rw-r-- 1 srao srao  217 Sep 28 20:12 dff_const4.v
-rw-rw-r-- 1 srao srao  218 Sep 28 20:12 dff_const5.v
-rw-rw-r-- 1 srao srao  820 Sep 28 20:12 dff_net.v
-rw-rw-r-- 1 srao srao  190 Sep 28 20:12 dff_syncres.v
-rw-rw-r-- 1 srao srao  107 Sep 28 20:12 fa.v
-rw-rw-r-- 1 srao srao  254 Sep 28 20:12 good_counter.v
-rw-rw-r-- 1 srao srao  161 Sep 28 20:12 good_latch.v
-rw-rw-r-- 1 srao srao  632 Sep 28 20:12 good_mux_netlist.v
-rw-rw-r-- 1 srao srao  138 Sep 28 20:12 good_mux.v
-rw-rw-r-- 1 srao srao  240 Sep 28 20:12 good_shift_reg.v
```



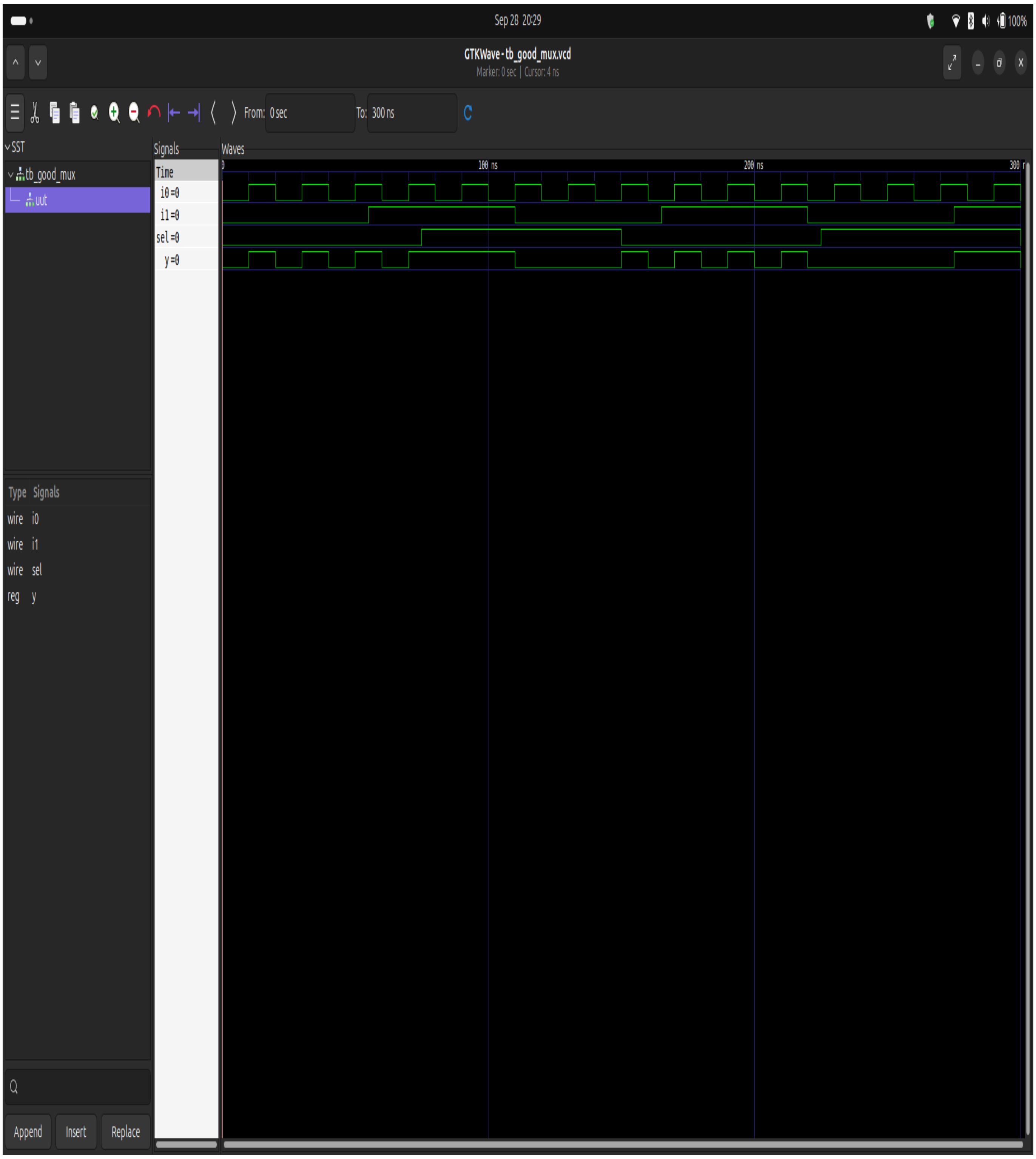
```
sky130RTLDesignAndSynthesisWorkshop/verilog_files on  main via V
❯ iverilog good_mux.v tb_good_mux.v

sky130RTLDesignAndSynthesisWorkshop/verilog_files on  main [?] via V
❯ ./a.out
VCD info: dumpfile tb_good_mux.vcd opened for output.
tb_good_mux.v:23: $finish called at 300000 (1ps)
```



```
sky130RTLDesignAndSynthesisWorkshop/verilog_files on  main [?] via V
❯ cat good_mux.v

module good_mux (input i0 , input i1 , input sel , output reg y);
always @ (*)
begin
        if(sel)
                y <= i1;
        else
                y <= i0;
end
endmodule
```



```
sky130RTLDesignAndSynthesisWorkshop/verilog_files on  main [?] via V
❯ cat tb_good_mux.v
`timescale 1ns / 1ps
module tb_good_mux;
        // Inputs
        reg i0,i1,sel;
        // Outputs
        wire y;

        // Instantiate the Unit Under Test (UUT)
        good_mux uut (
                .sel(sel),
                .i0(i0),
                .i1(i1),
                .y(y)
        );

        initial begin
        $dumpfile("tb_good_mux.vcd");
        $dumpvars(0,tb_good_mux);
        // Initialize Inputs
        sel = 0;
        i0 = 0;
        i1 = 0;
        #300 $finish;
        end

always #75 sel = ~sel;
always #10 i0 = ~i0;
always #55 i1 = ~i1;
endmodule
```


Sep 28 20:29
100%
GTKWave - tb_good_mux.vcd
Marker: 0 sec | Cursor: 4 ns
From: 0 sec
To: 300 ns
SST
tb_good_mux
uut
Type Signals
wire i0
wire i1
wire sel
reg y
Append
Insert
Replace
Signals
Time
i0=0
i1=0
sel=0
y=0
Waves
100 ns
200 ns
300 ns