Day-3

Introduction to Logic Optimization

It is mainly done to reduce the logic to the most optimized design. Most optimized design will be in terms of Area and Power.

Techniques of Optimization for Combinational Logic

1.Constant Propagation-Direct Optimization technique

2.Boolean Logic Optimization-Techniques such as K-Map and Quine Mckluskey

Techniques of Optimization for Sequential Logic

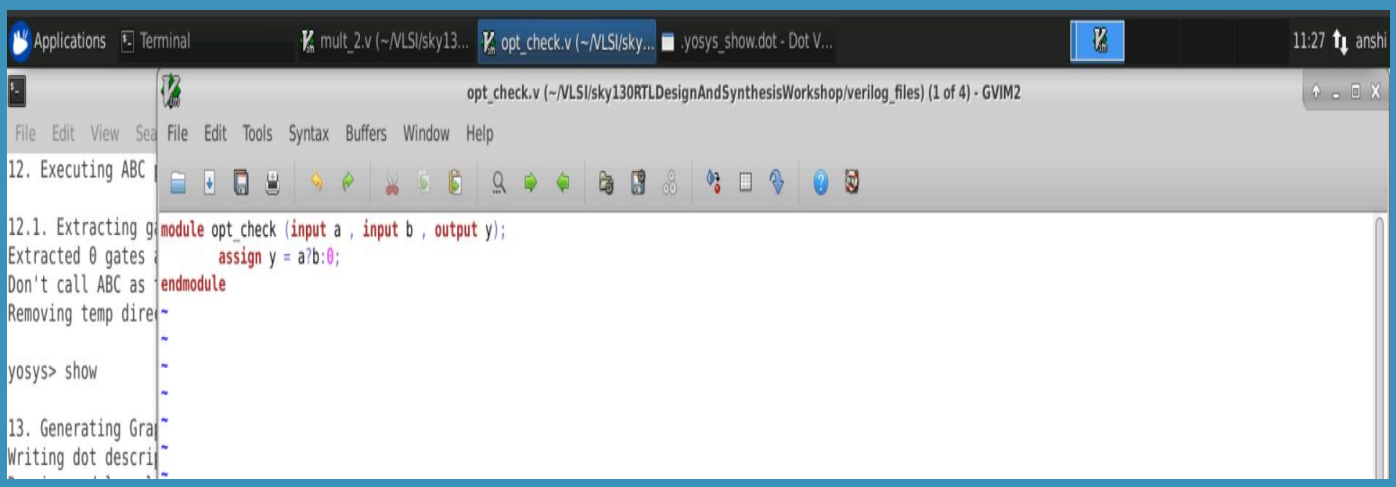
1. **Basic**
   - Sequential Constant Propagation
     Whenever the Flip-Flop output is constant irrespective of clock and reset/set signals, then it is in the state of Sequential Constant.
     This state of the Flip-Flop can be further optimized.
     Otherwise, if it is taking values both 1 and 0, then it is not in the state of Sequential Constant.

2. **Advanced**
   - State Optimization
     Optimization of Unused States
   - Retiming-It is done by slicing or Logic Partitioning to balance the timing across the circuit in order to get better performance with increased frequency.
   - Sequential Logic Cloning-It causes multiple instances of input flip flop with Large Positive Slack.

Lab Examples of Optimization

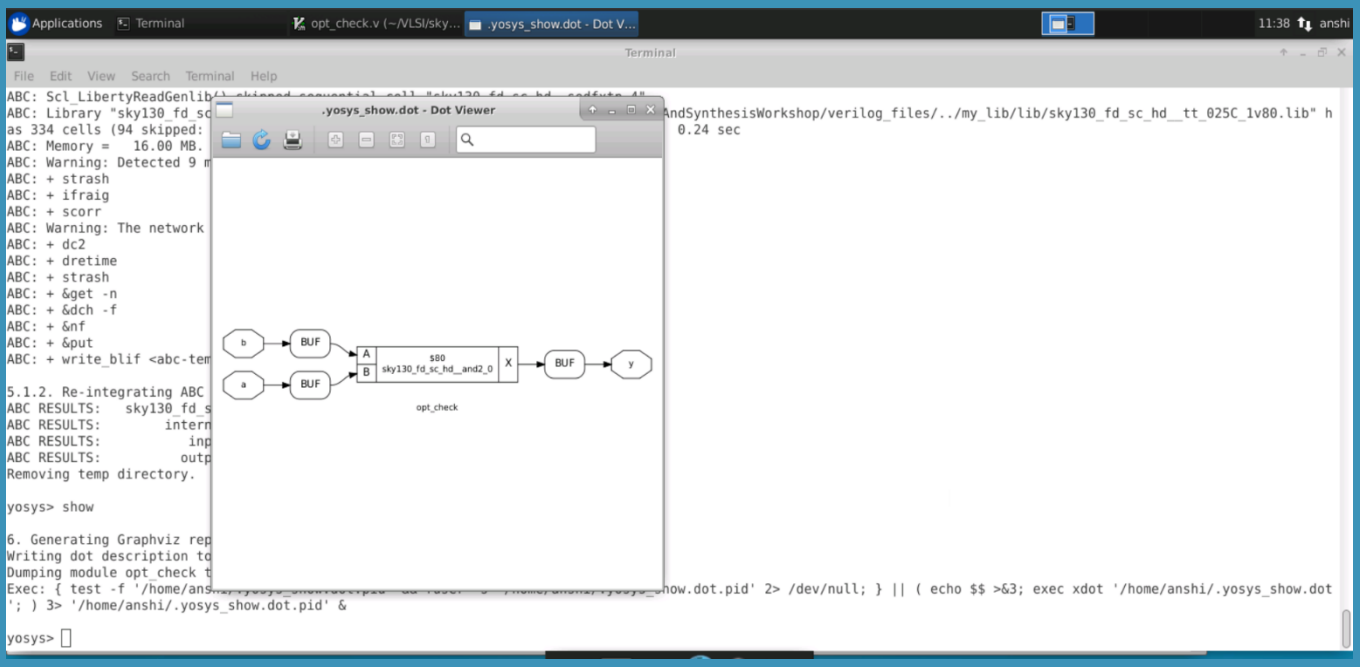
1. Logic Optimized to AND Gate

This is an example of Boolean Logic Optimization.



Command to do Constant Propagation and Optimization is:

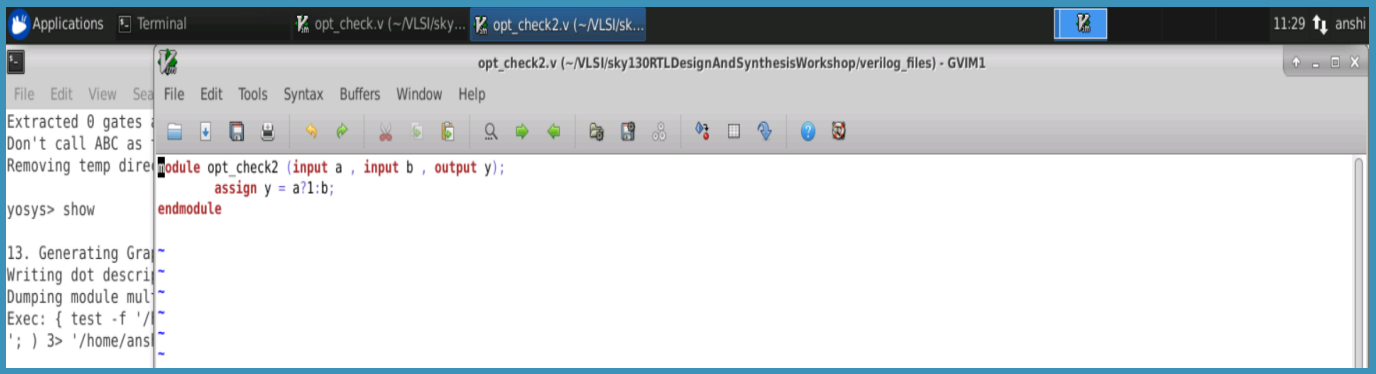
opt_clean -purge

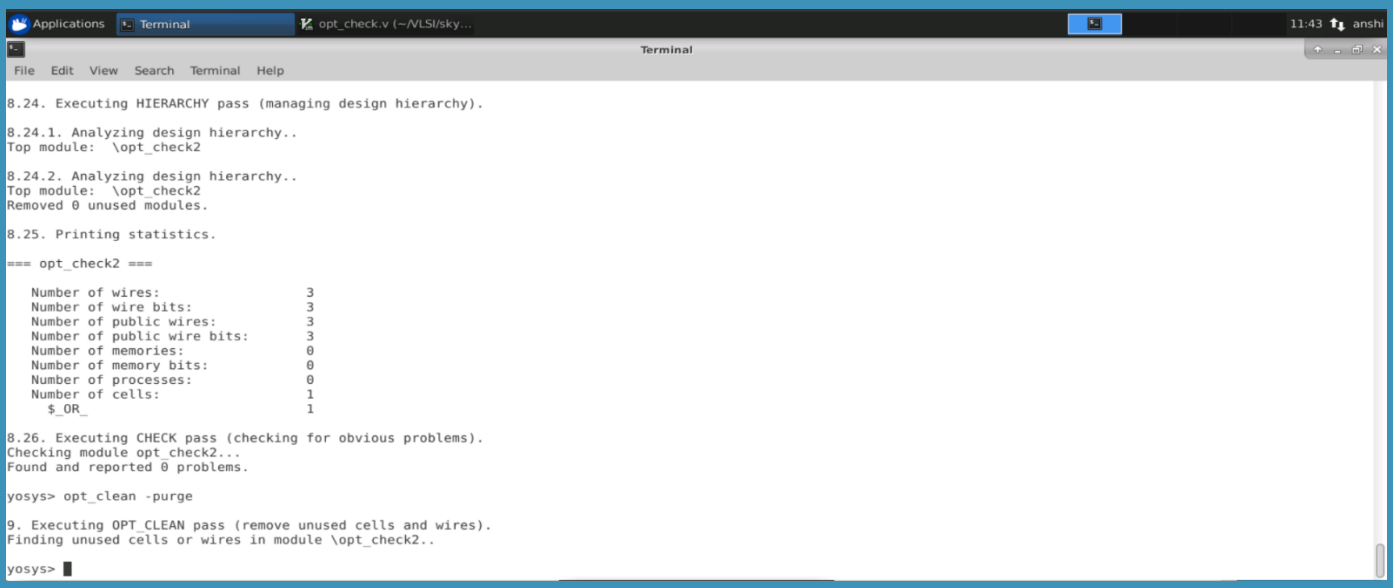


A 2-input AND gate is synthesized after Logic Optimization

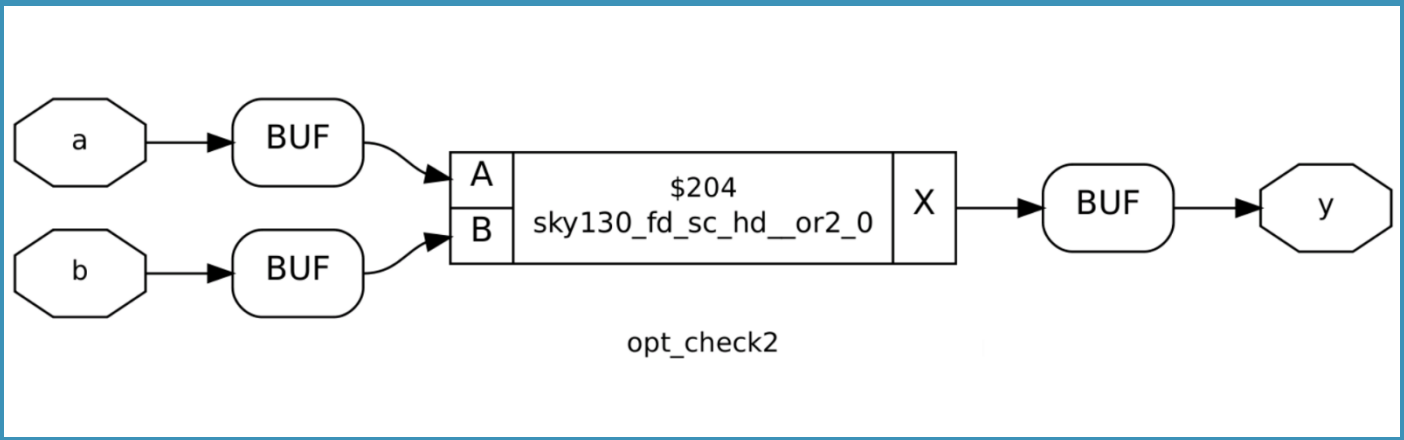
2.Optimization to OR Gate

The below logic gets optimized to 2-input OR Gate



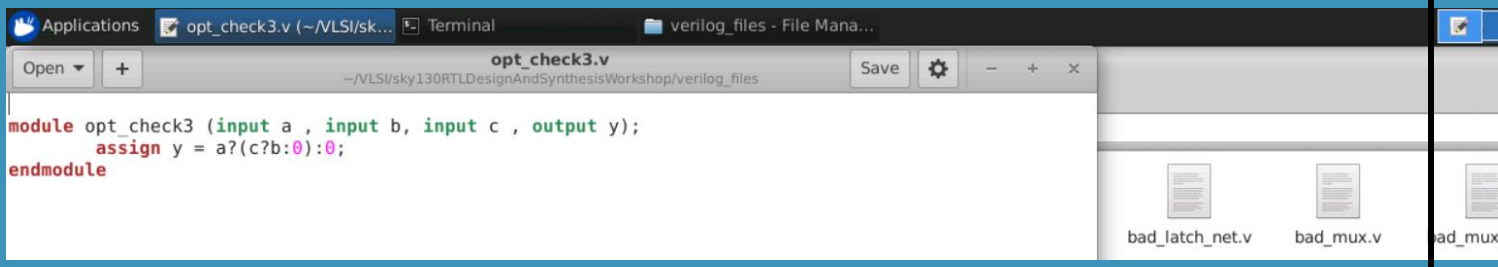


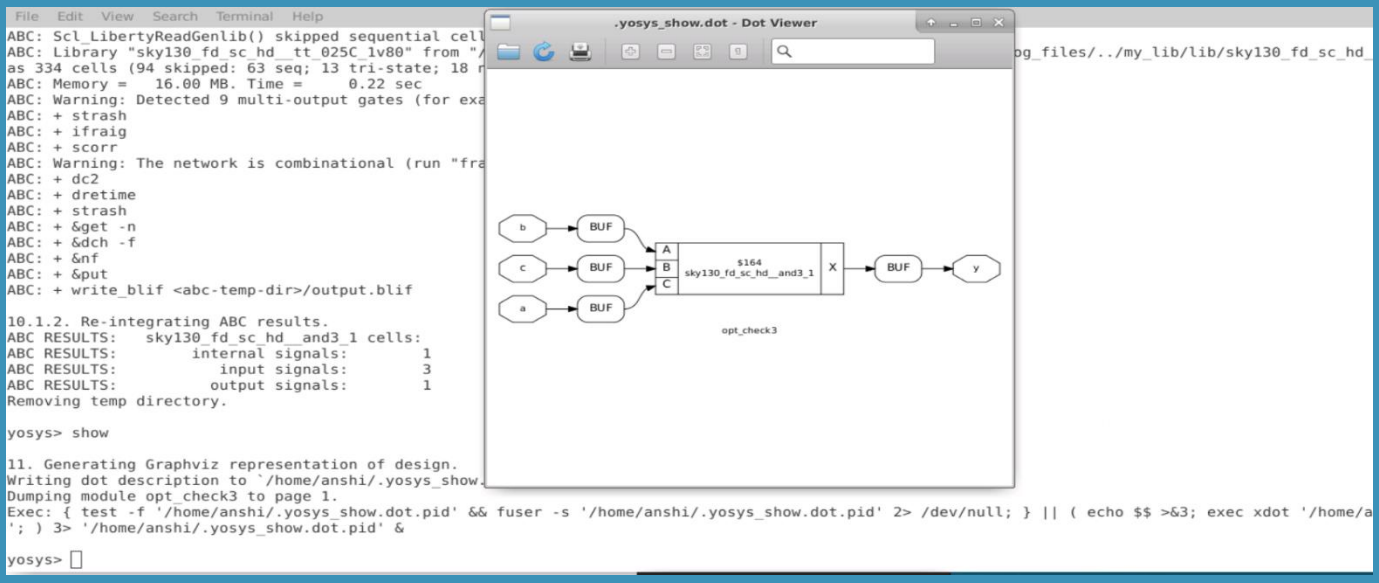
OR Gate Design



3.Optimization to 3-input AND Gate

The given logic gets optimized to 3-input And Gate

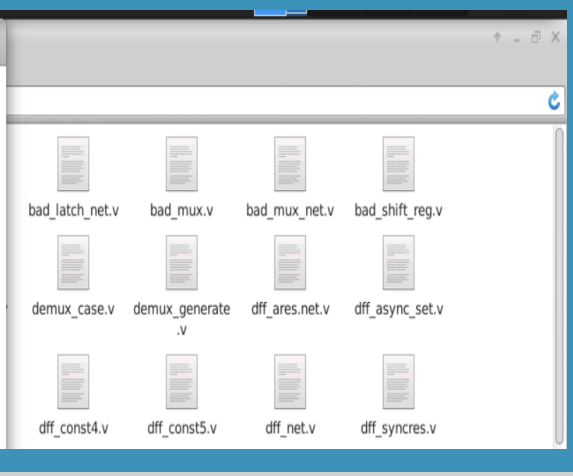




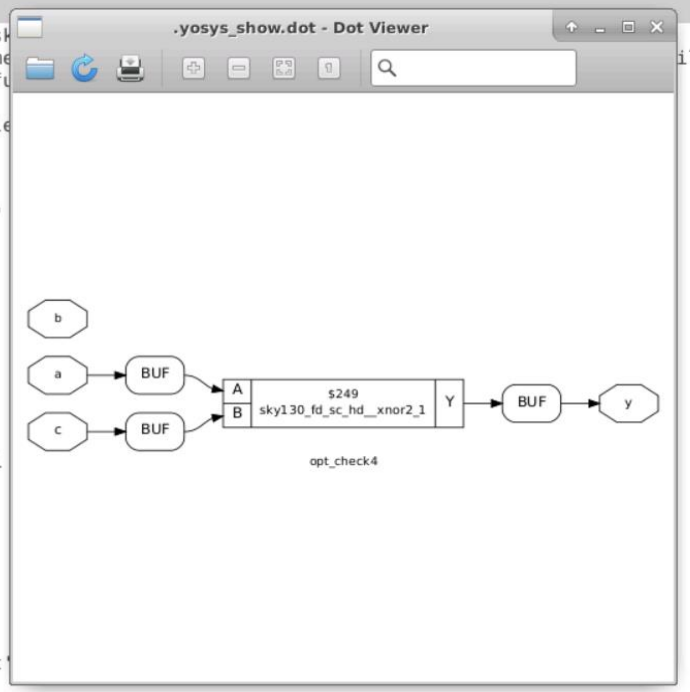
4.Optimization to 2-Input X-OR Gate

opt_check4.v

```
module opt_check4 (input a , input b , input c , output y);
 assign y = a?(b?(a & c ):c):(!c);
 endmodule
```




```
BC: Scl_LibertyReadGenlib() skipped sequential cell "sk
BC: Library "sky130_fd_sc_hd__tt_025C_1v80" from "/home
s 334 cells (94 skipped: 63 seq; 13 tri-state; 18 no fu
BC: Memory =   16.00 MB. Time =     0.23 sec
BC: Warning: Detected 9 multi-output gates (for example
BC: + strash
BC: + ifraig
BC: + scorr
BC: Warning: The network is combinational (run "fraig"
BC: + dc2
BC: + dretime
BC: + strash
BC: + &get -n
BC: + &dch -f
BC: + &nf
BC: + &put
BC: + write_blif <abc-temp-dir>/output.blif

5.1.2. Re-integrating ABC results.
BC RESULTS:   sky130_fd_sc_hd__xnor2_1 cells:        1
BC RESULTS:        internal signals:        0
BC RESULTS:           input signals:        2
BC RESULTS:          output signals:        1
emoving temp directory.

osys> show

6. Generating Graphviz representation of design.
riting dot description to `/home/anshi/.yosys_show.dot'
umping module opt_check4 to page 1.
xec: { test -f '/home/anshi/.yosys_show.dot.pid' && fuser -s '/home/anshi/.yosys_show.dot.pid' 2> /dev/null; } || ( echo
; ) 3> '/home/anshi/.yosys_show.dot.pid' &
```



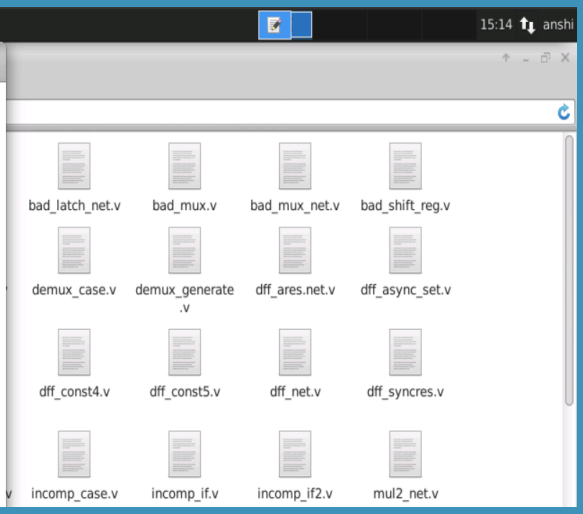
Instead of realising 2 Multiplexer, one single XNOR gate is realised.

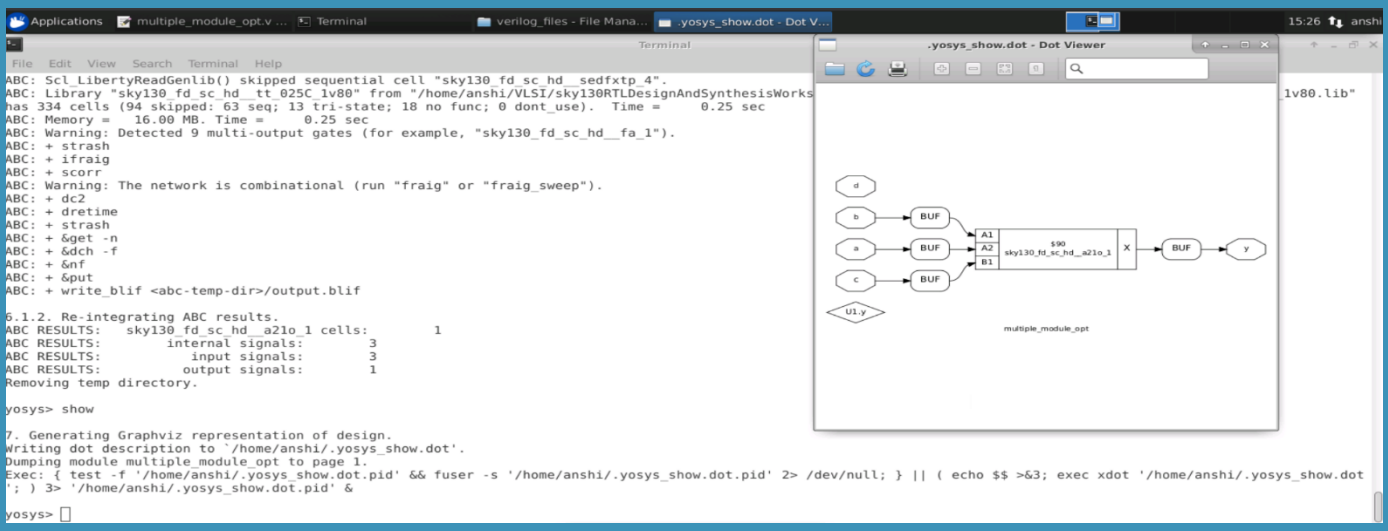
5.Multiple_Module_opt

Instead of realising 2 gates, one single gate is realised

multiple_module_opt.v

```
module sub_module1(input a , input b , output y);
 assign y = a & b;
endmodule


module sub_module2(input a , input b , output y);
 assign y = a^b;
endmodule


module multiple_module_opt(input a , input b , input c , input d , output y);
wire n1,n2,n3;

sub_module1 U1 (.a(a) , .b(1'b1) , .y(n1));
sub_module2 U2 (.a(n1), .b(1'b0) , .y(n2));
sub_module2 U3 (.a(b), .b(d) , .y(n3));

assign y = c | (b & n1);


endmodule
```

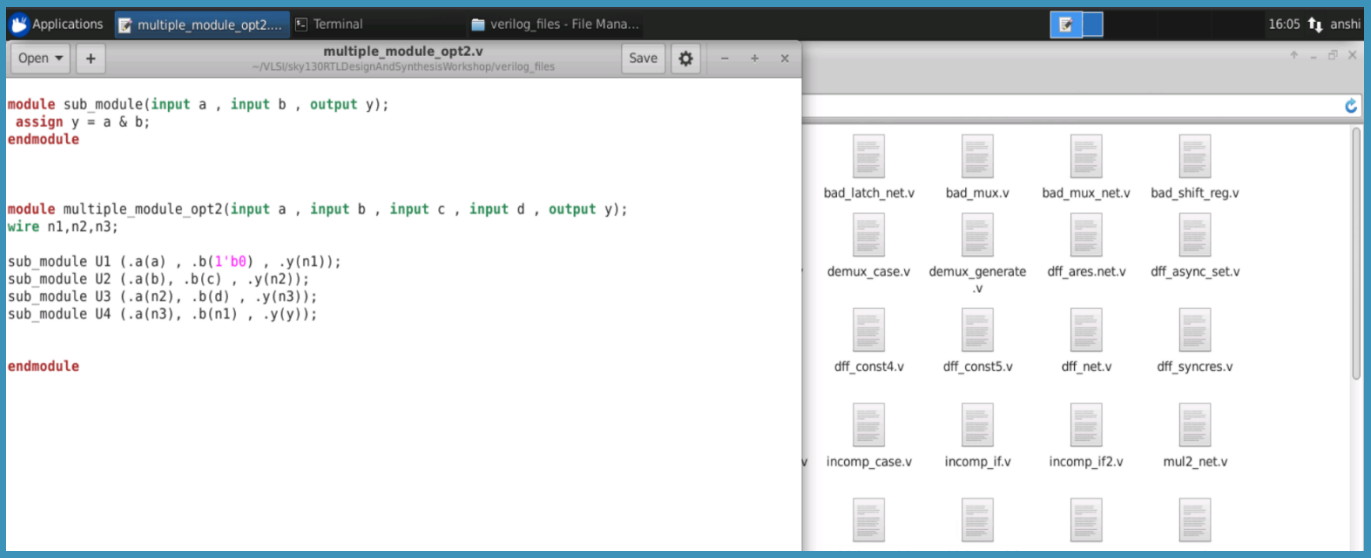


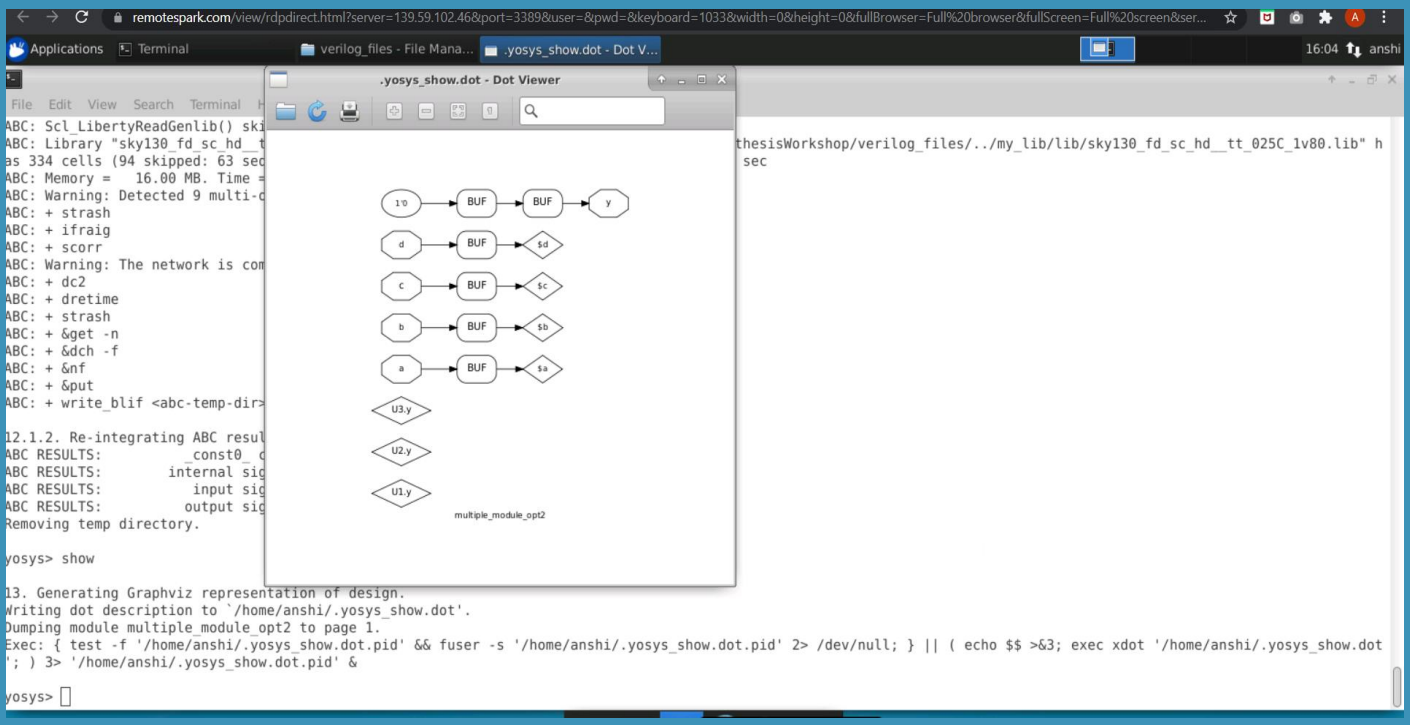


Ab+c is realised and got optimised as shown above.

6. Multiple_Module_opt_2

Y is optimized to 1 without using any Gate

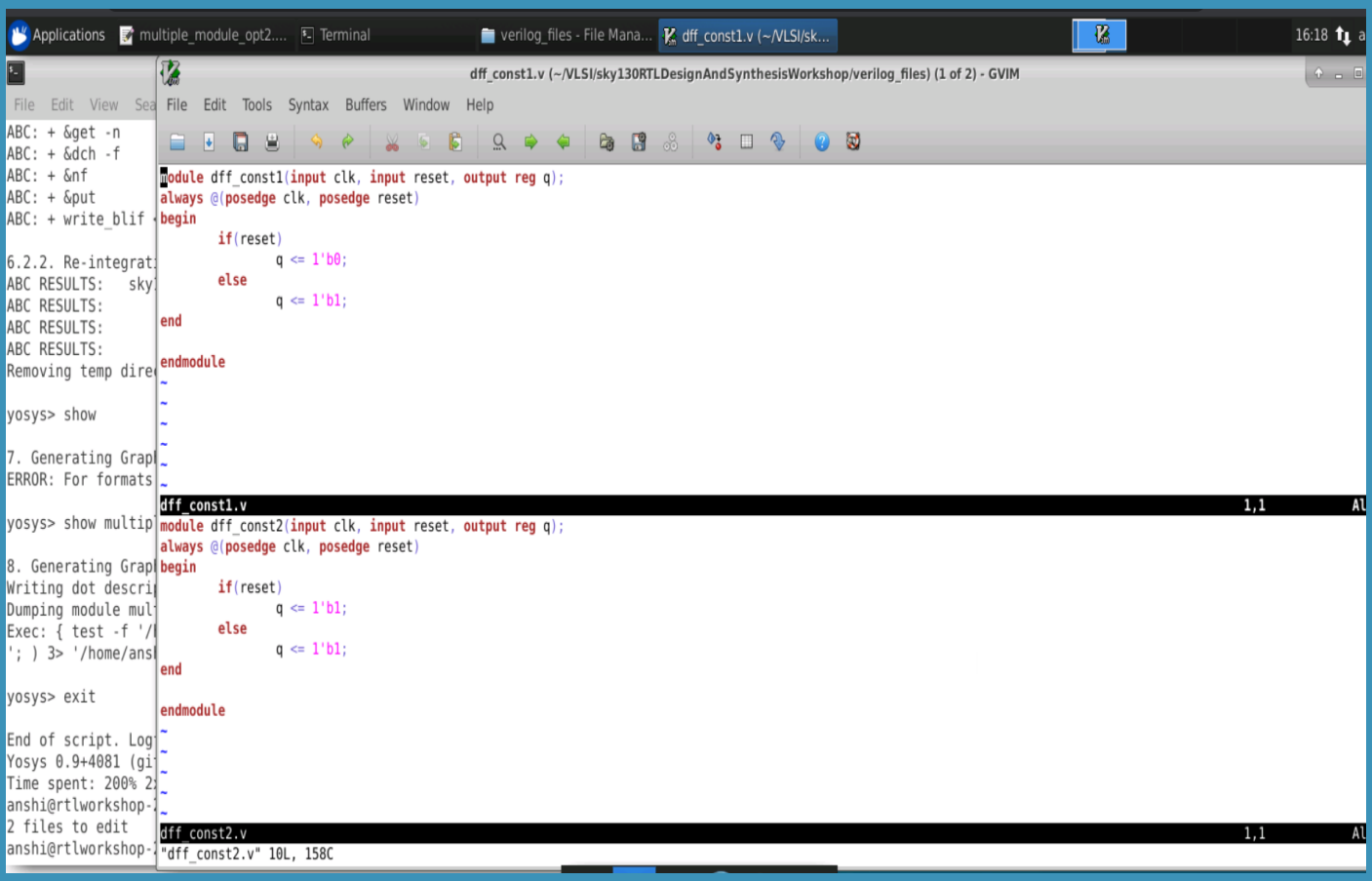




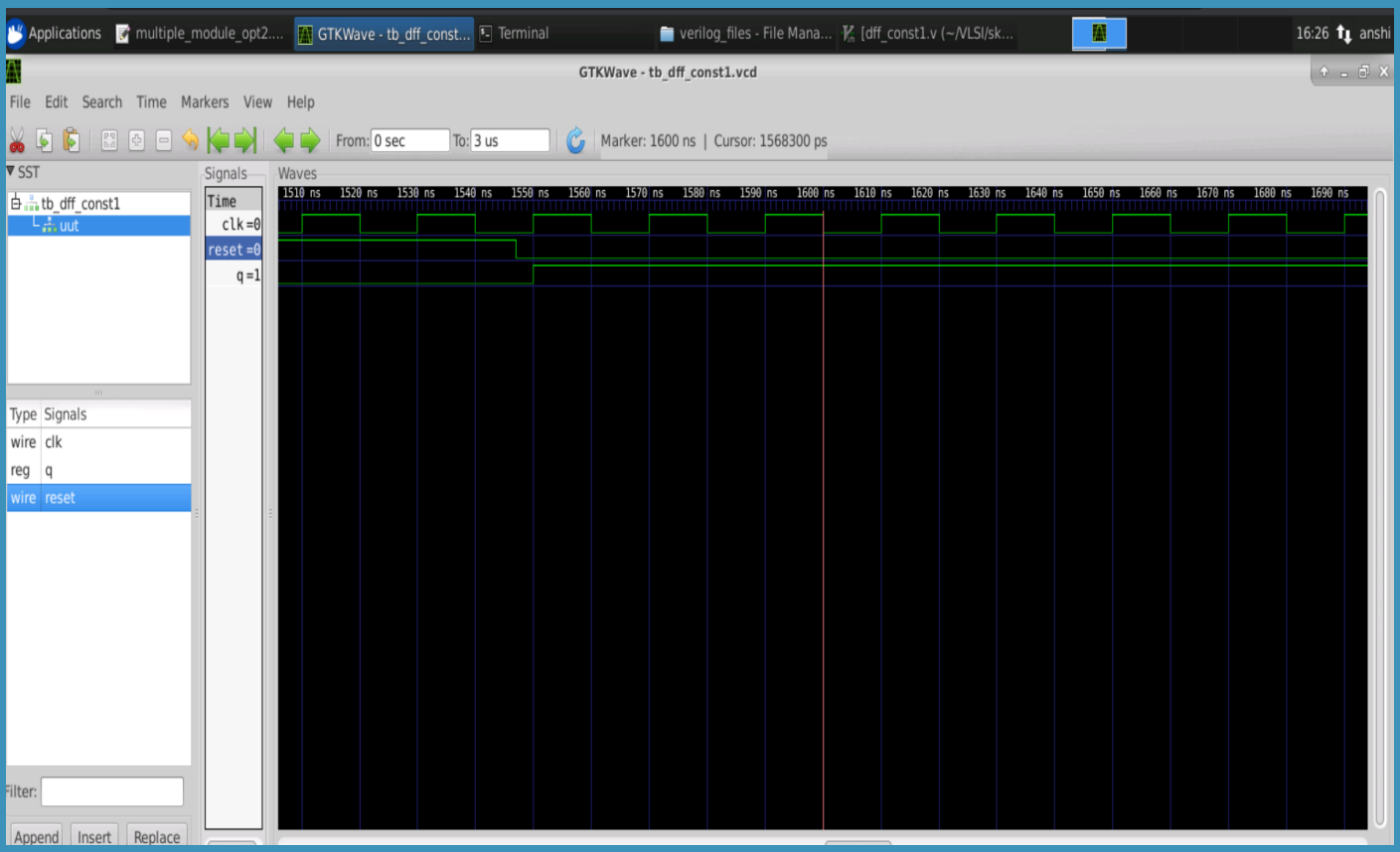
Sequential Optimization- Sequential Constant Propagation

Lab Examples of DFF Flip-Flops

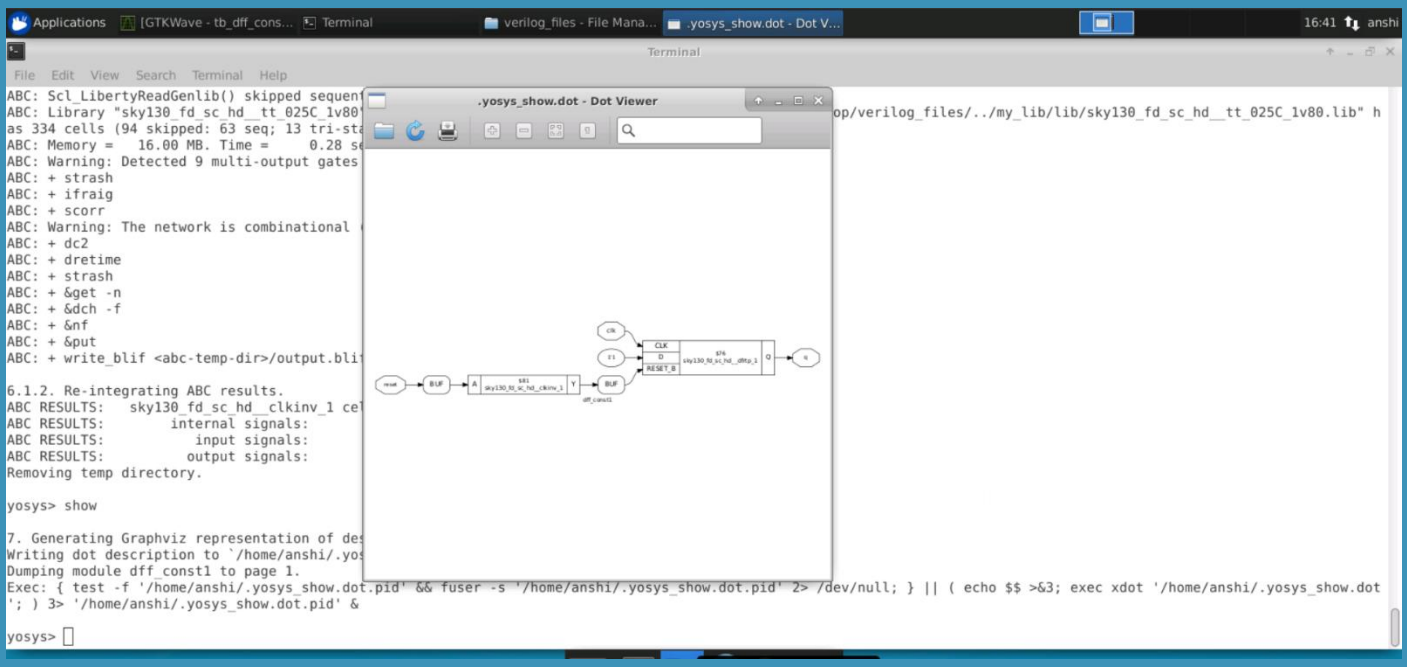
1. **Dff_const1**



In dff_const_1 design Dff is set to 0 at reset else high

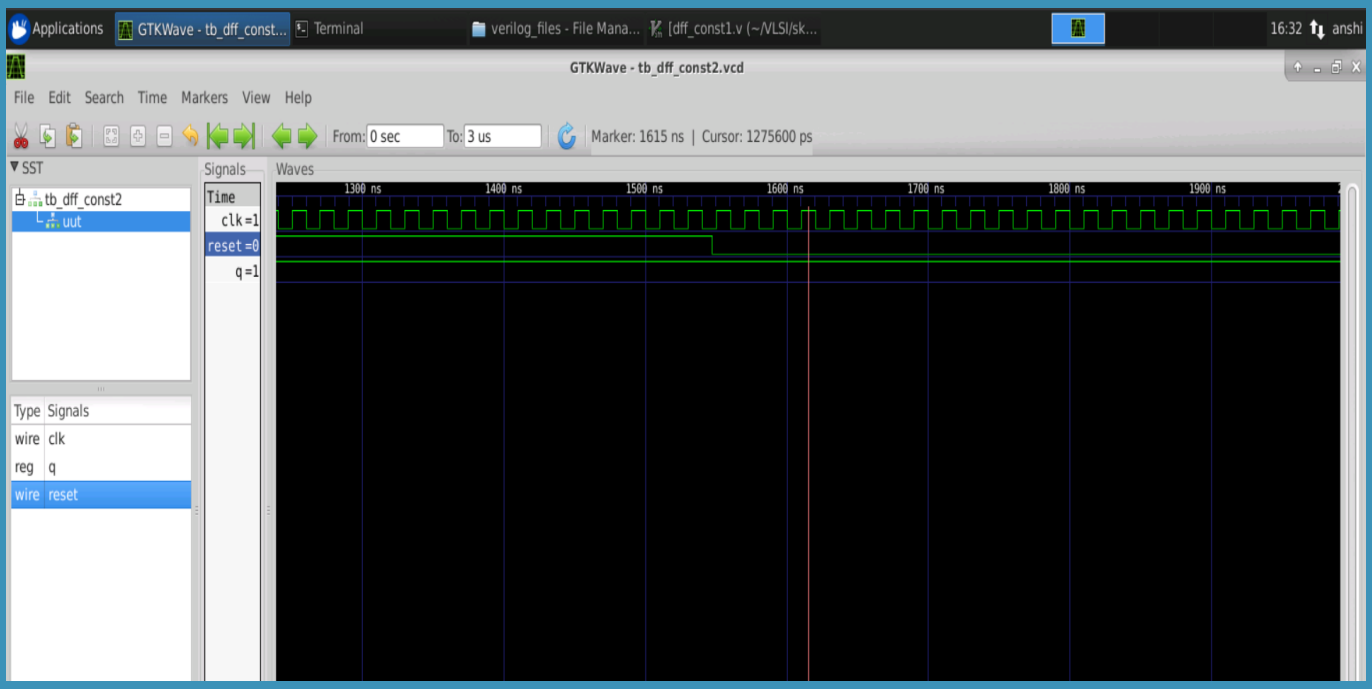


Design: A dff is realised



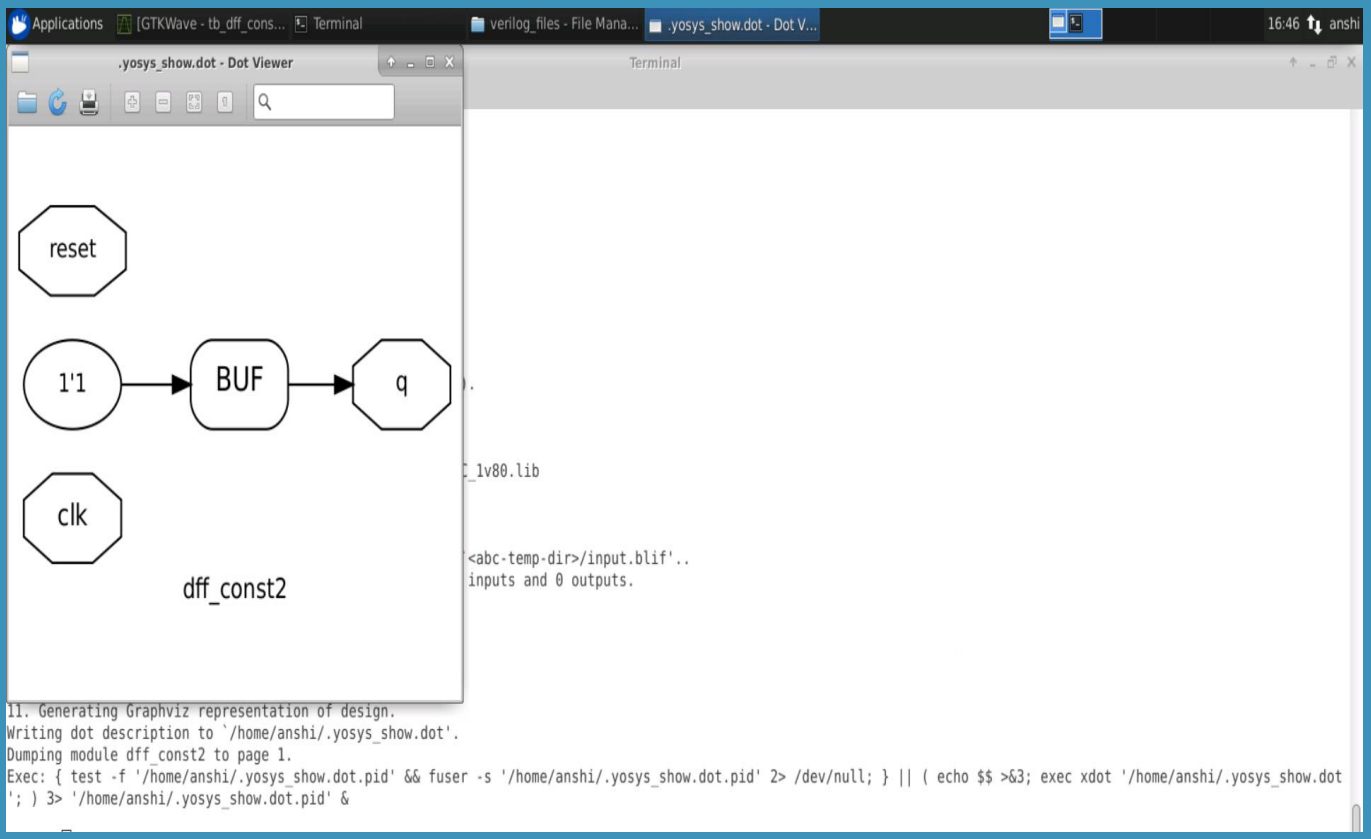
2.Dff_const2

The output of the D-FF is always set 1 irrespective of reset or clock

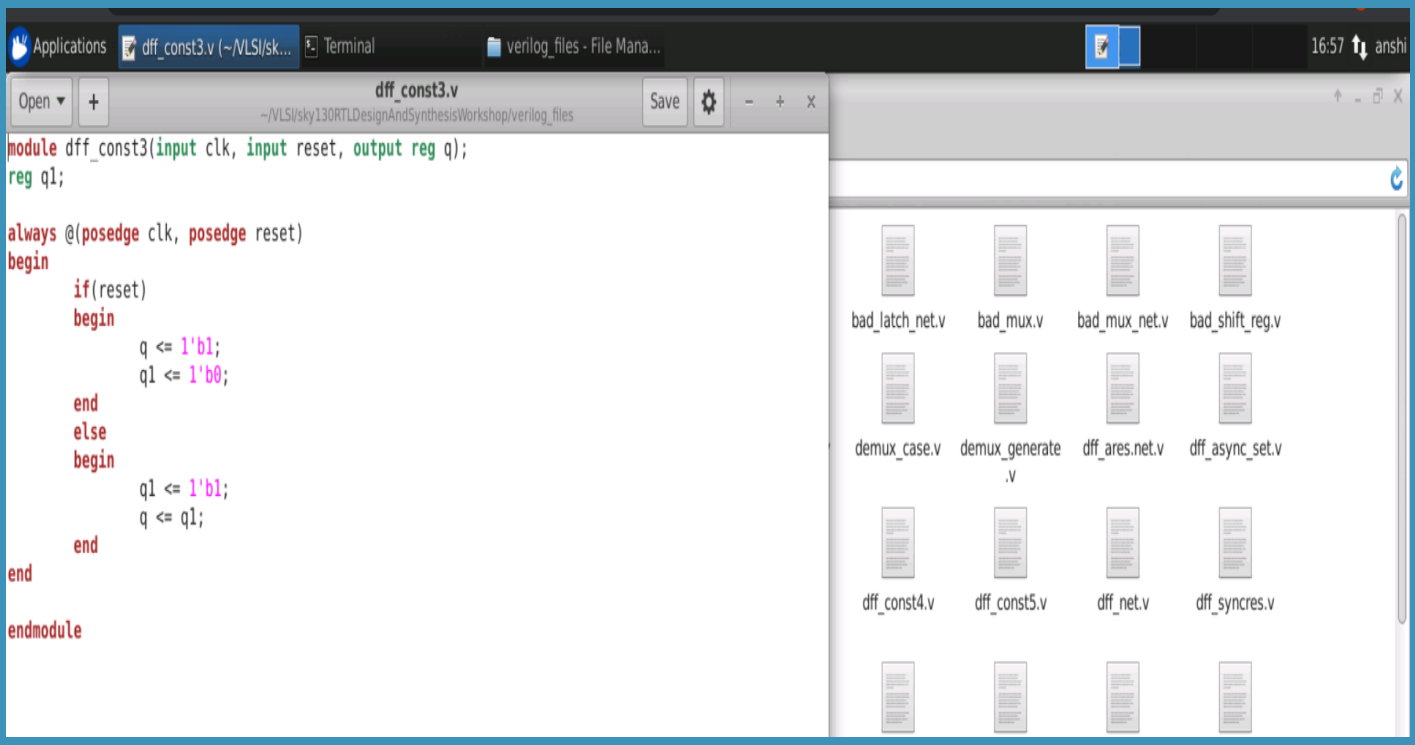


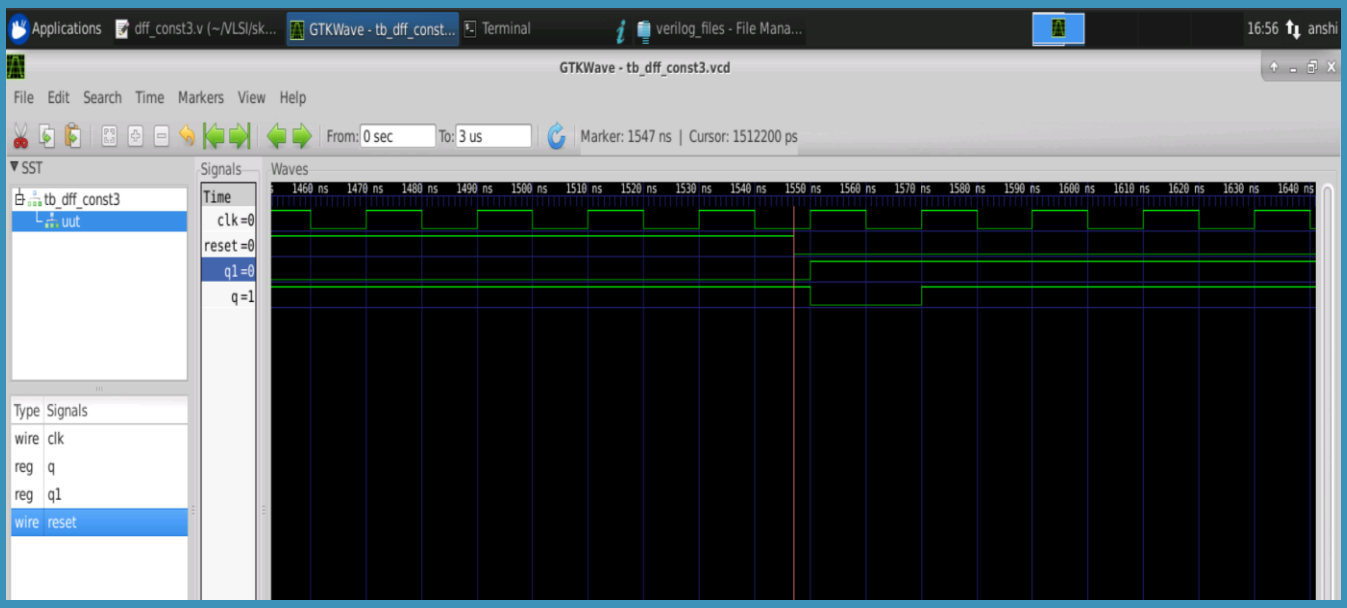
After Synthesizing

Because, library is having active low reset so it is inferring an inverter in order to have reset as active high. No flip-flop is realised q is assigned the value of 1

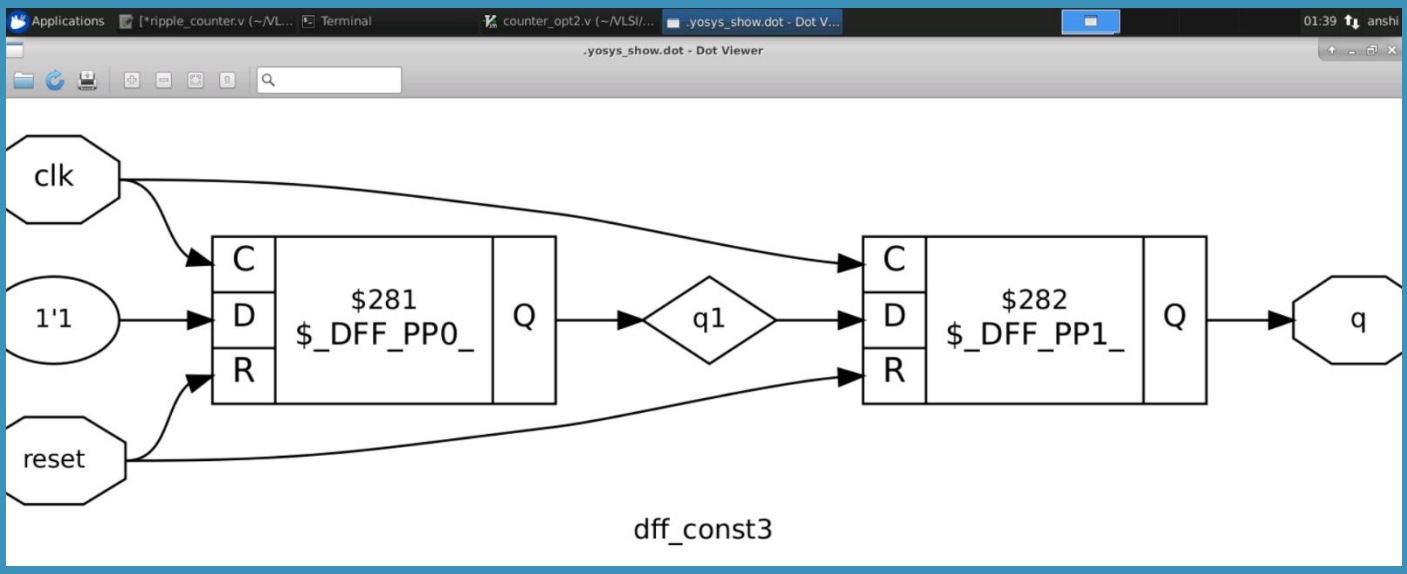

Applications
[GTKWave - tb_dff_cons...
Terminal
verilog_files - File Mana...
.yosys_show.dot - Dot V...
16:46
anshi
.yosys_show.dot - Dot Viewer
Terminal
reset
1'1
BUF
q
clk
dff_const2
_1v80.lib
<abc-temp-dir>/input.blif'..
inputs and 0 outputs.
11. Generating Graphviz representation of design.
Writing dot description to `/home/anshi/.yosys_show.dot'.
Dumping module dff_const2 to page 1.
Exec: { test -f '/home/anshi/.yosys_show.dot.pid' && fuser -s '/home/anshi/.yosys_show.dot.pid' 2> /dev/null; } || (echo $$ >&3; exec xdot '/home/anshi/.yosys_show.dot
';) 3> '/home/anshi/.yosys_show.dot.pid' &

3.Dff_const3

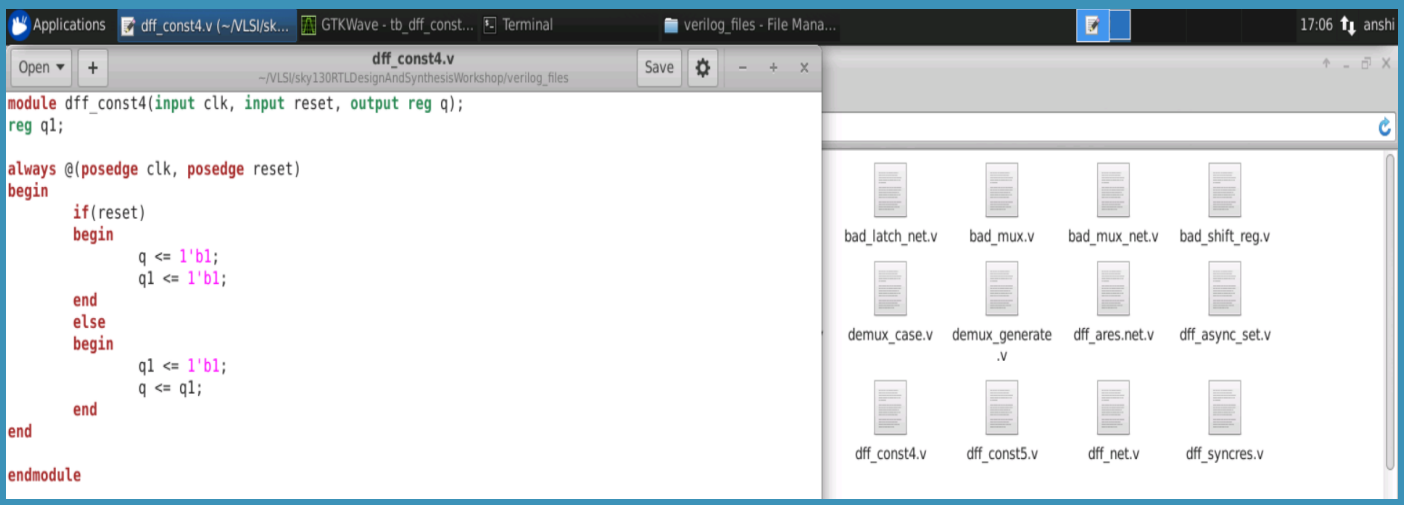

Applications
dff_const3.v (~/VLSI/sk...
Terminal
verilog_files - File Mana...
16:57
anshi
Open
dff_const3.v
~/VLSI/sky130RTLDesignAndSynthesisWorkshop/verilog_files
Save
module dff_const3(input clk, input reset, output reg q);
reg q1;
always @(posedge clk, posedge reset)
begin
if(reset)
begin
q <= 1'b1;
q1 <= 1'b0;
end
else
begin
q1 <= 1'b1;
q <= q1;
end
end
endmodule
bad_latch_net.v
bad_mux.v
bad_mux_net.v
bad_shift_reg.v
demux_case.v
demux_generate.v
dff_ares.net.v
dff_async_set.v
dff_const4.v
dff_const5.v
dff_net.v
dff_syncres.v

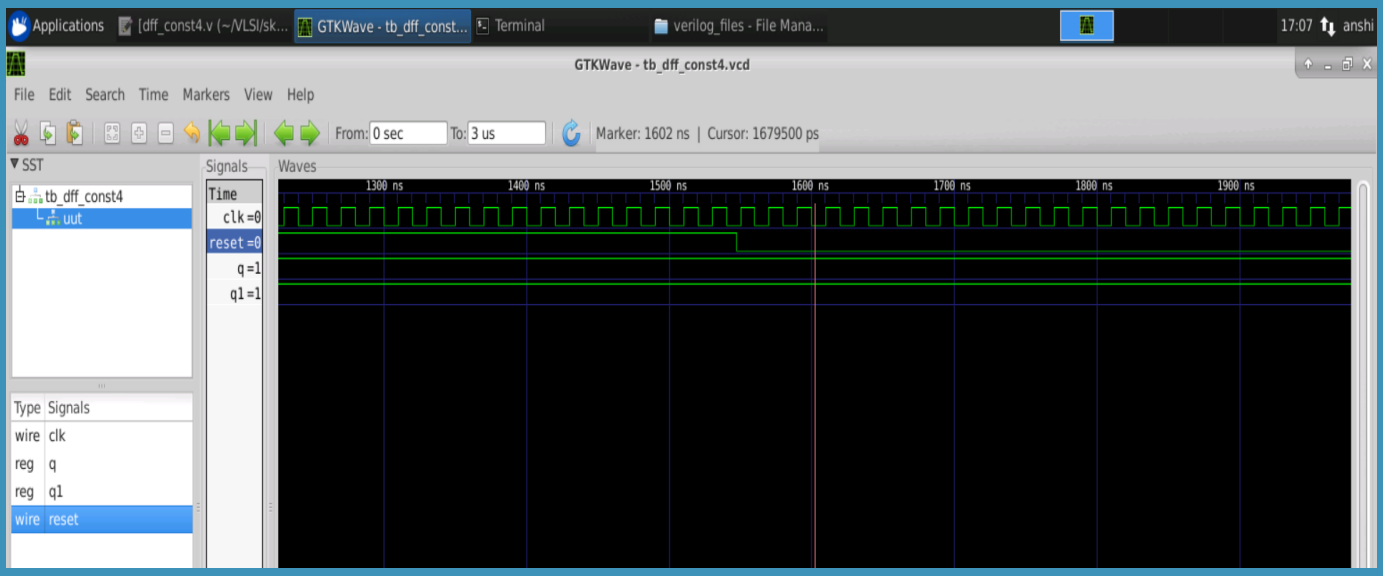


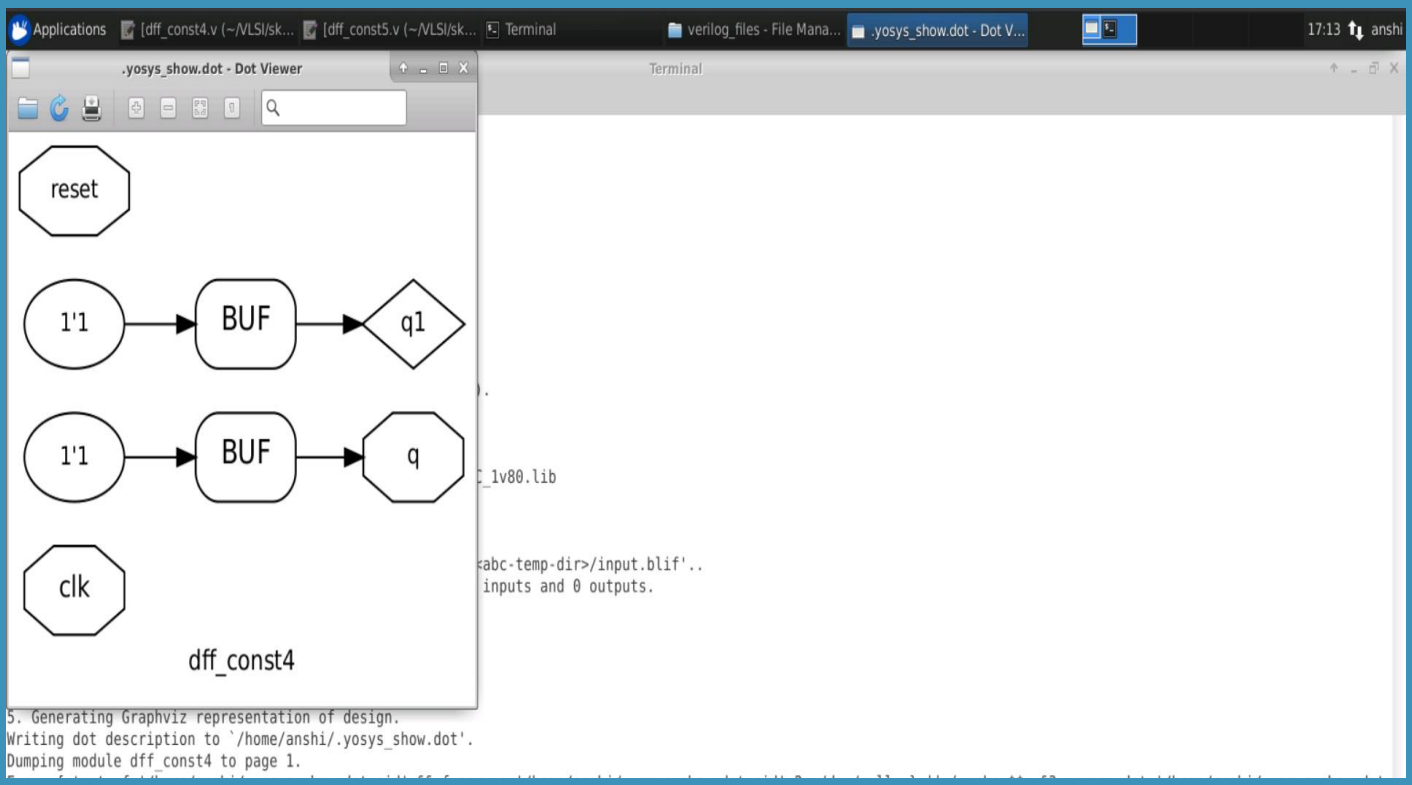
2 Flip-flops are realised with q1 given as input 1 and q is assigned q1



3. Dff_const4

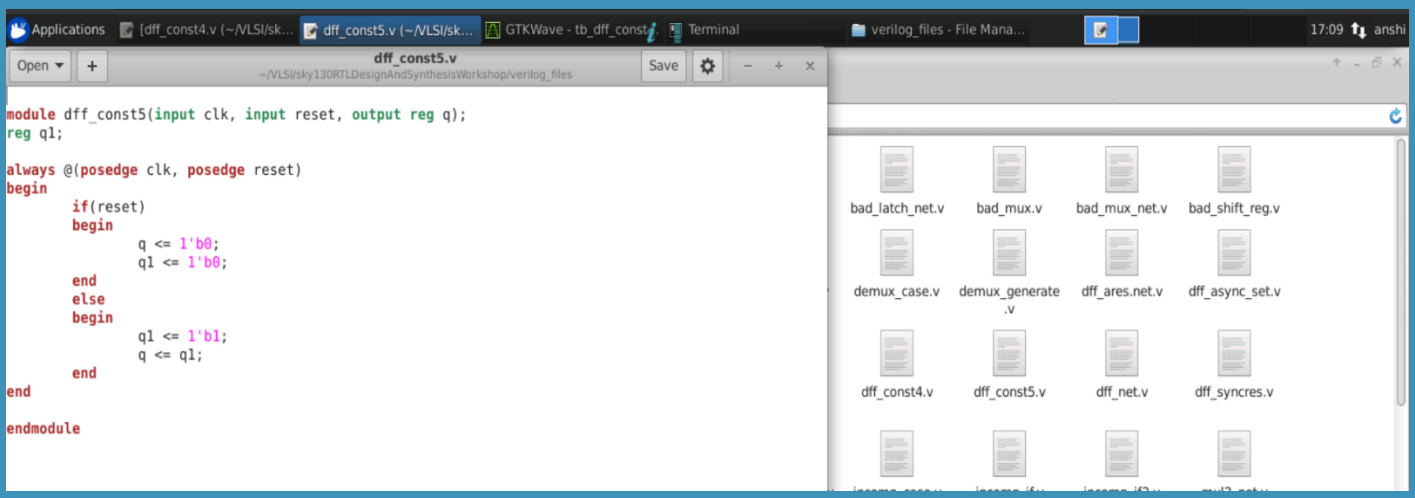


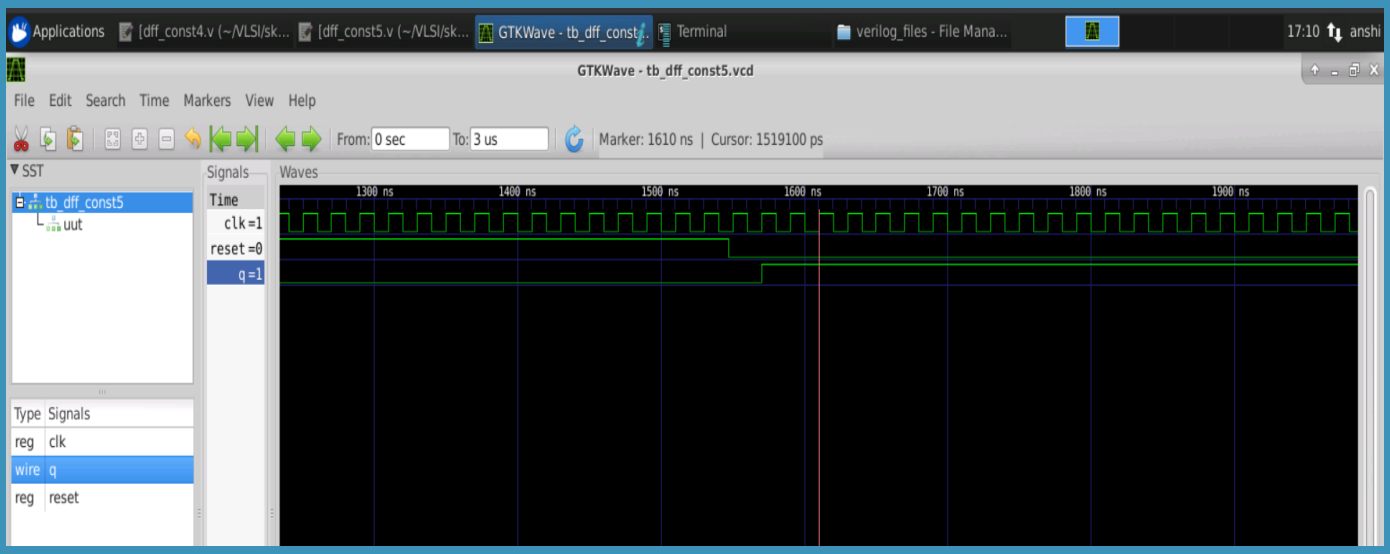


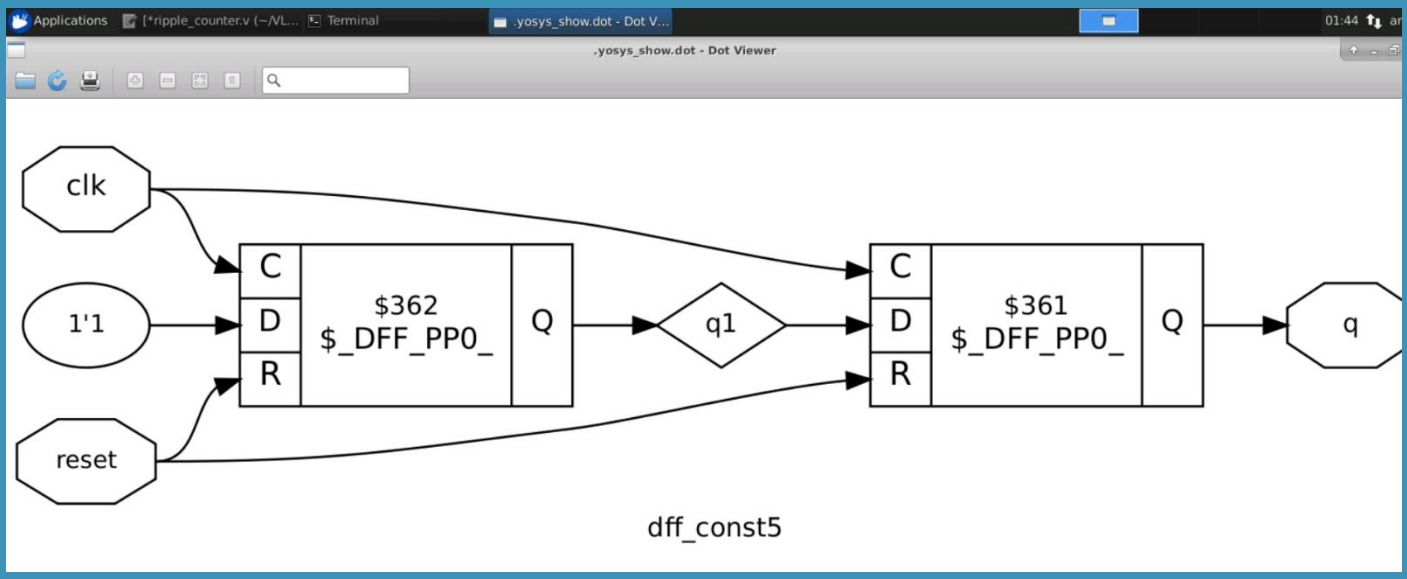


When reset q and q1 are1 else q1 is 1 and q is assigned q1. So, both the flip-flops are always one

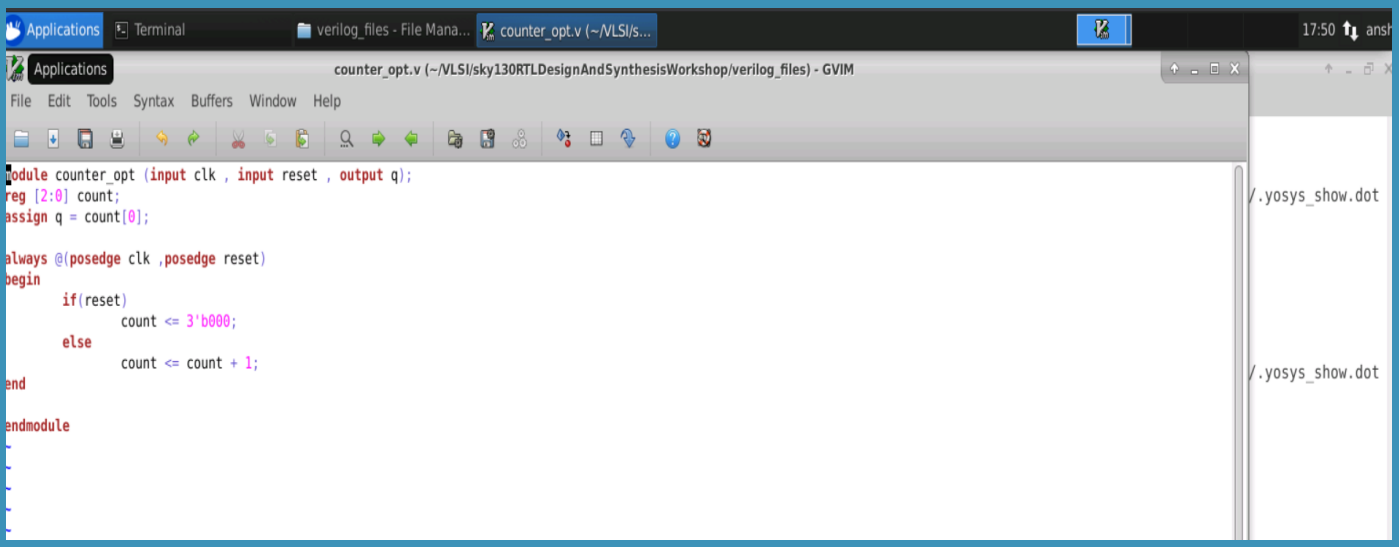
5.Dff_const5.v

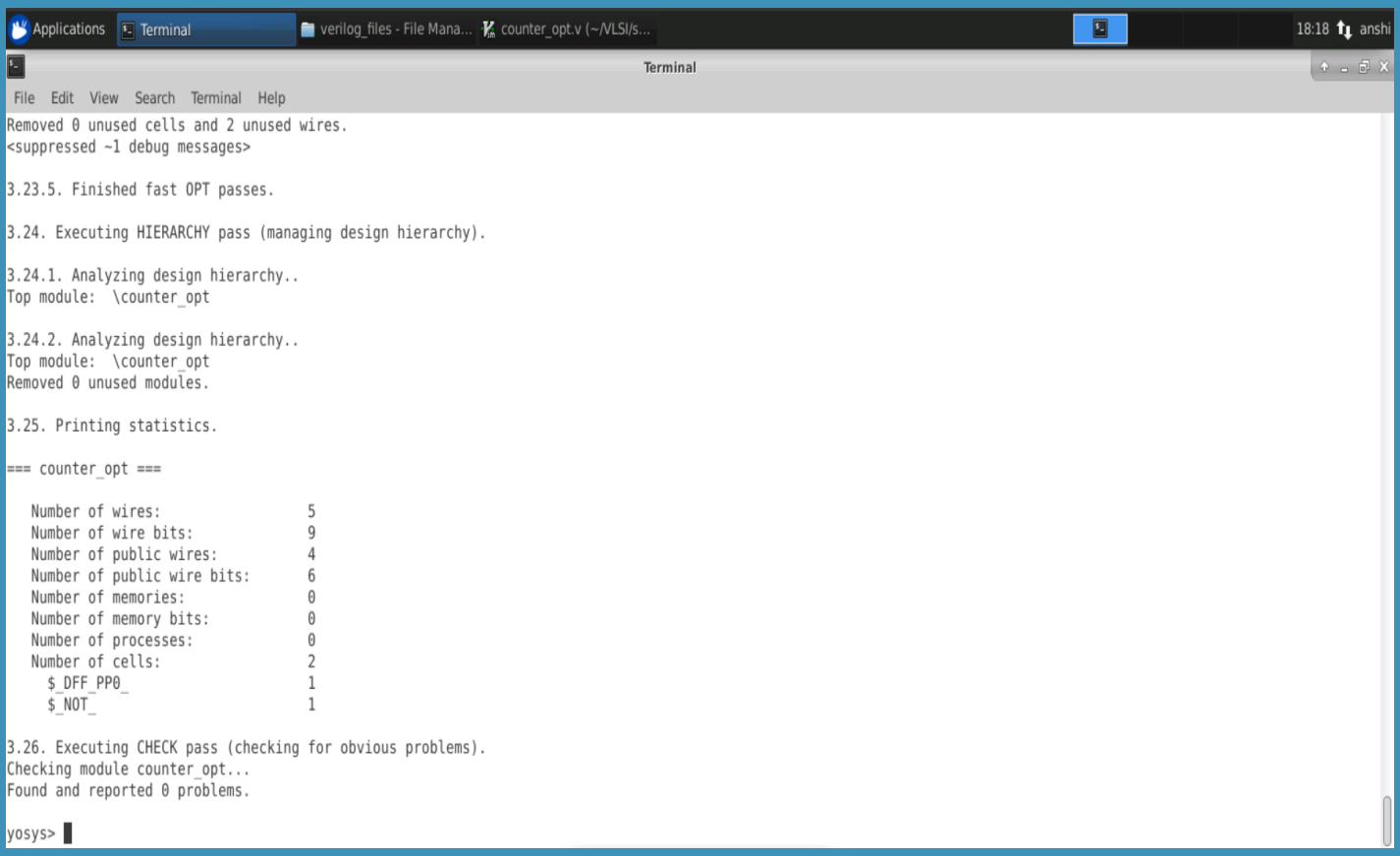


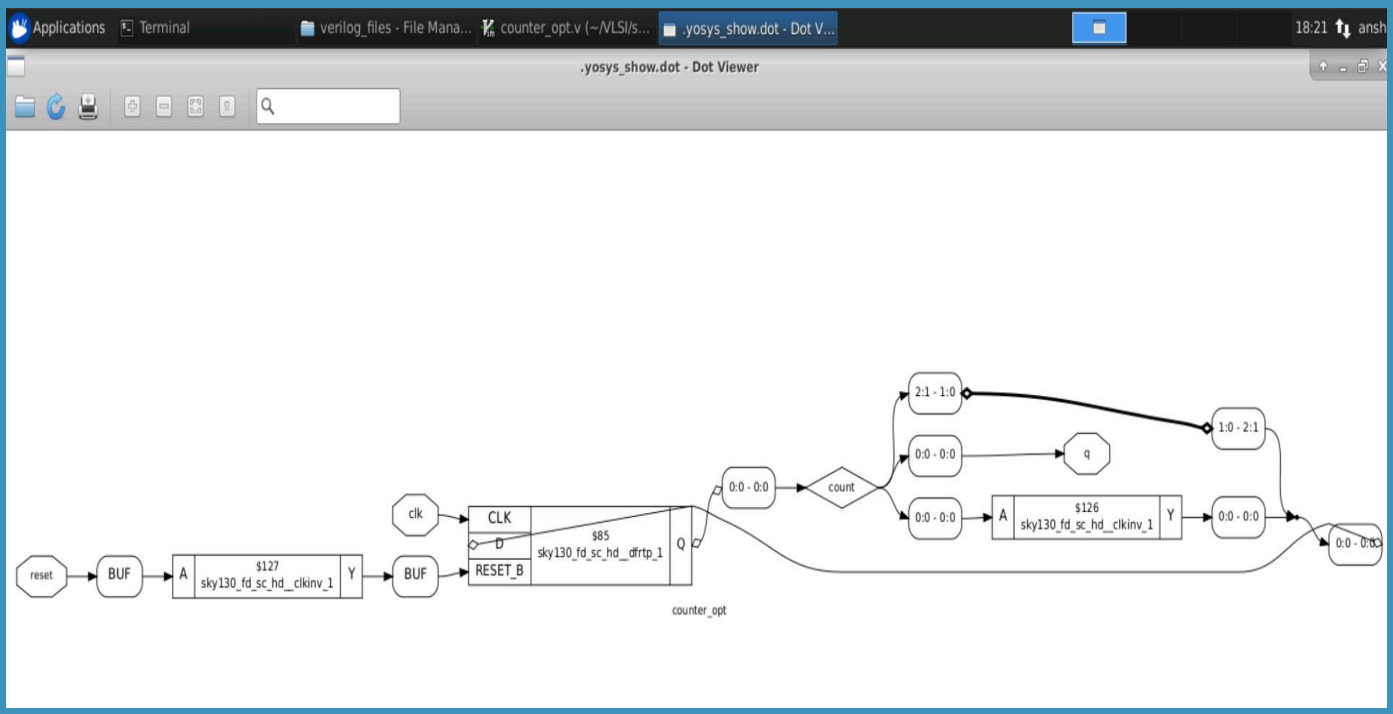




Unused Output Optimization







Toggle Flip-Flop is realised as output is just last bit of the count which keeps toggling on every clock cycle

Counter_opt2