

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 Mccluskey

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.

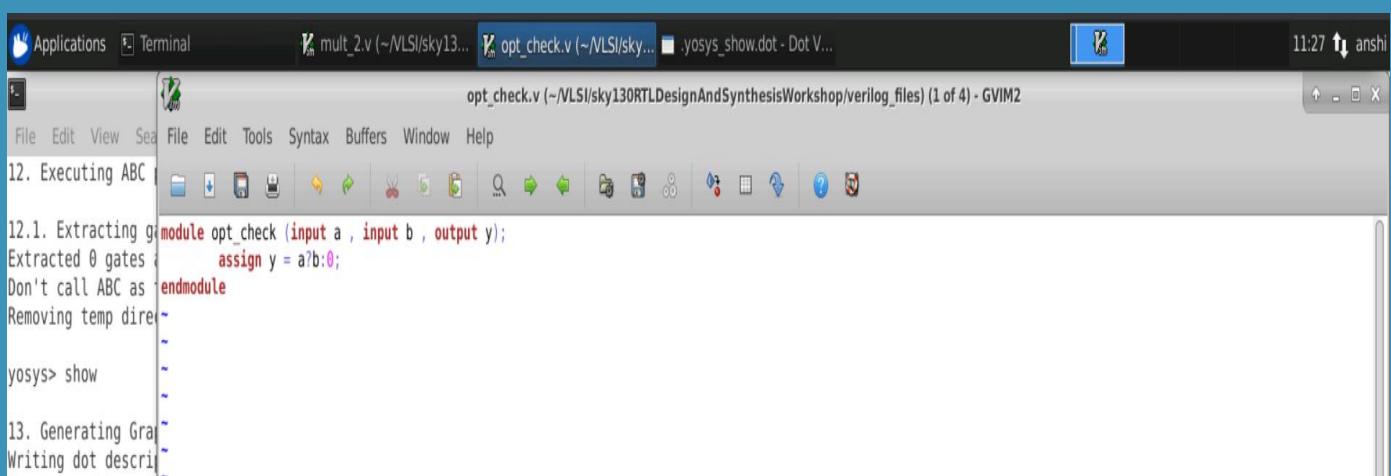
2. Advanced

- 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.

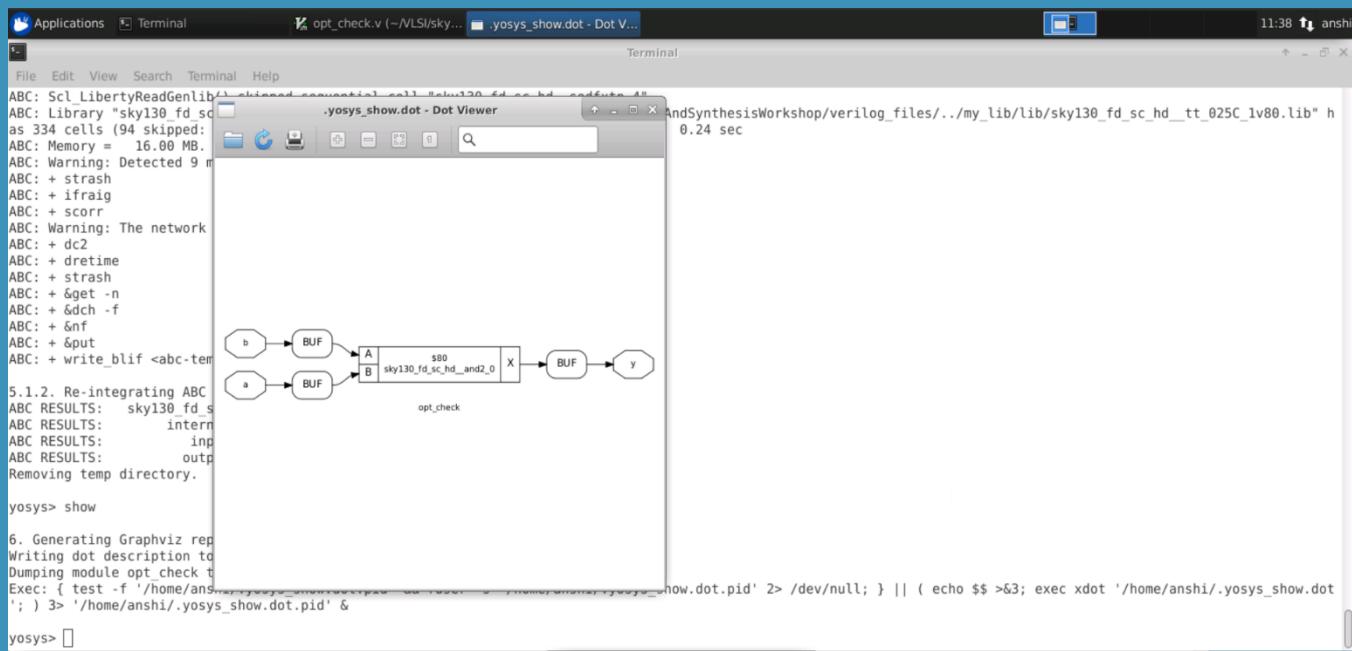


The screenshot shows a GVIM window with several tabs open at the top: 'mult_2.v (~VLSI/sky130...', 'opt_check.v (~VLSI/sky130...', and '.yosys_show.dot - Dot V...'. The main editor area displays Verilog code for a module named 'opt_check' with an assignment `assign y = a?b:0;`. Below the code, the terminal output shows steps 12 and 13 of a process:

```
12. Executing ABC
12.1. Extracting gates
Extracted 0 gates
Don't call ABC as
Removing temp directory
yosys> show
13. Generating Graph
Writing dot description
```

Command to do Constant Propagation and Optimization is:

```
opt_clean -purge
```



```
ABC: Scl_LibertyReadGenlib
ABC: Library "sky130_fd_sc_hd_and2_0"
ABC: as 334 cells (94 skipped)
ABC: Memory = 16.00 MB.
ABC: Warning: Detected 9 m
ABC: + strash
ABC: + ifraig
ABC: + scor
ABC: Warning: The network
ABC: + dc2
ABC: + dftime
ABC: + strash
ABC: + &get -n
ABC: + &dch -f
ABC: + &nf
ABC: + &put
ABC: + write_bif <abc-temp.bif>
5.1.2. Re-integrating ABC
ABC RESULTS: sky130_fd_sc_hd_and2_0
ABC RESULTS: internal
ABC RESULTS: input
ABC RESULTS: output
Removing temp directory.

yosys> show
6. Generating Graphviz representation
Writing dot description to opt_check.t
Dumping module opt_check
Exec: { test -f '/home/anshi/.yosys_show.dot.pid' &
'; } 3> '/home/anshi/.yosys_show.dot.pid' &

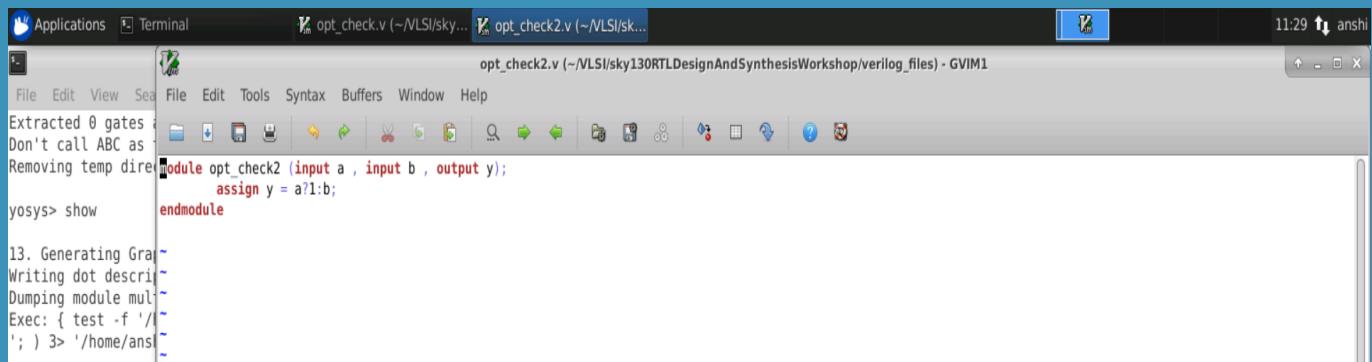
yosys>
```

The dot viewer window displays a logic diagram for an AND gate. It shows two inputs, 'a' and 'b', each passing through a BUF (buffer) gate. The outputs of these two BUF gates are labeled 'A' and 'B'. These signals are then combined at a logic AND gate, labeled '\$80 sky130_fd_sc_hd_and2_0 X'. The output of this AND gate is labeled 'X', which then passes through another BUF gate. Finally, the output of the second BUF gate is labeled 'y'.

A 2-input AND gate is synthesized after Logic Optimization

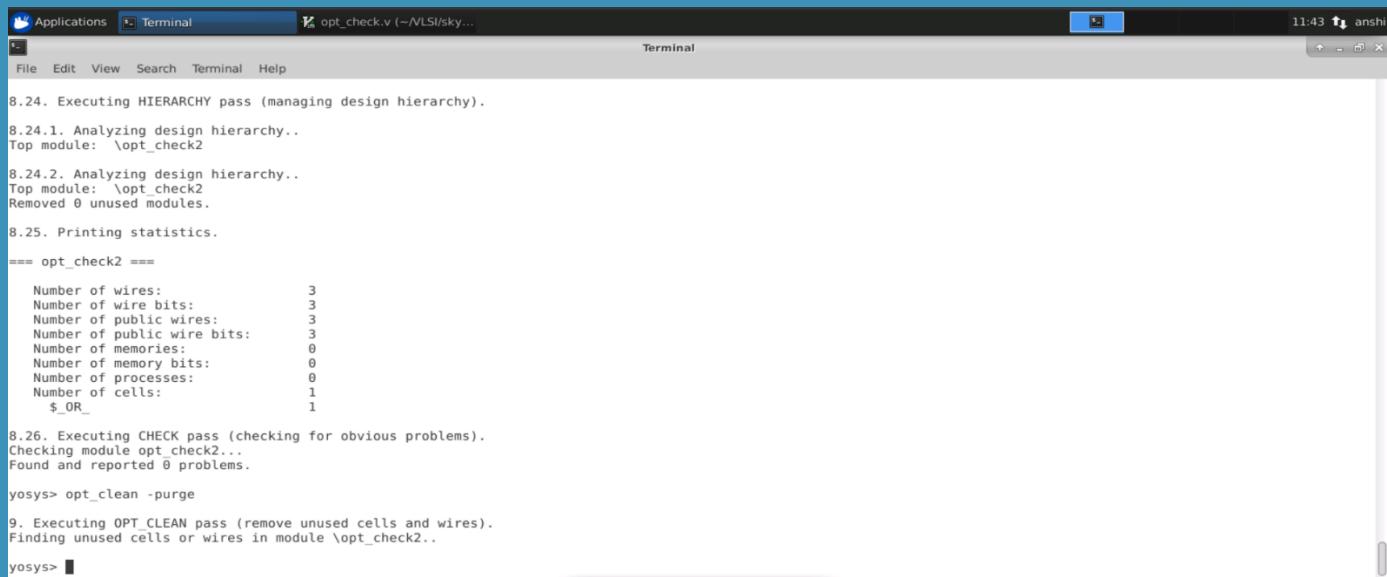
2.Optimization to OR Gate

The below logic gets optimized to 2-input OR Gate



```
Extracted 0 gates
Don't call ABC as
Removing temp directory.
module opt_check2 (input a, input b, output y);
    assign y = a?1:b;
endmodule

yosys> show
13. Generating Graphviz representation
Writing dot description to opt_check2.t
Dumping module opt_check2
Exec: { test -f '/home/anshi/.yosys_show.dot.pid' &
'; } 3> '/home/anshi/.yosys_show.dot.pid' &
```



```
8.24. Executing HIERARCHY pass (managing design hierarchy).
8.24.1. Analyzing design hierarchy..
Top module: \opt_check2
8.24.2. Analyzing design hierarchy..
Top module: \opt_check2
Removed 0 unused modules.

8.25. Printing statistics.

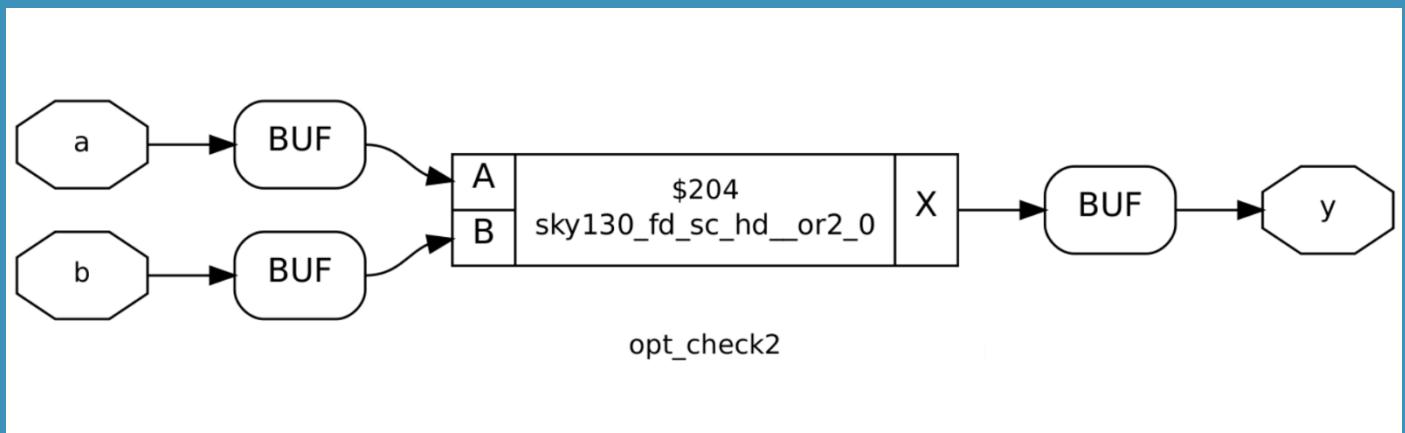
==== opt_check2 ====
Number of wires: 3
Number of wire bits: 3
Number of public wires: 3
Number of public wire bits: 3
Number of memories: 0
Number of memory bits: 0
Number of processes: 0
Number of cells: 1
$_OR_ 1

8.26. Executing CHECK pass (checking for obvious problems).
Checking module opt_check2...
Found and reported 0 problems.

yosys> opt_clean -purge
9. Executing OPT_CLEAN pass (remove unused cells and wires).
Finding unused cells or wires in module \opt_check2.

yosys>
```

OR Gate Design



3.Optimization to 3-input AND Gate

The given logic gets optimized to 3-input And Gate

The screenshot shows a software interface with a terminal window and a graphical viewer window. The terminal window displays Verilog code for an OR gate optimization:

```
module opt_check3 (input a , input b , input c , output y);
    assign y = a?(c?b:0):0;
endmodule
```

The graphical viewer window shows the optimized logic. The Verilog code is processed by ABC (ABC: + write_blib <abc-temp-dir>/output.blif) to generate a logic diagram. The diagram consists of three inputs (a, b, c) each passing through a buffer (BUF). Their outputs enter a logic block labeled '\$164 sky130_fd_sc_hd_and3_1'. This block has three ports: 'A', 'B', and 'C'. The 'A' and 'B' ports receive the inputs from the buffers. The output of this block is connected to the input of another buffer, which then produces the final output 'y'.

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

```

bad_latch_net.v bad_mux.v bad_mux_net.v bad_shift_reg.v
 demux_case.v demux_generate.v diff_ares.net.v diff_async_set.v
 dff_const4.v dff_const5.v dff_net.v dff_syncres.v

File Edit View Search Terminal Help

BC: Scl_LibertyReadGenlib() skipped sequential cell "sh" BC: Library "sky130_fd_sc_hd_tt_025C_lv80" 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: + dftime BC: + strash BC: + &get -n BC: + &dch -f BC: + &nf BC: + &put BC: + write_blf <abc-temp-dir>/output.blif

5.1.2. Re-integrating ABC results.

BC RESULTS: sky130_fd_sc_hd_xnor2_1 cells: 0
 BC RESULTS: internal signals: 0
 BC RESULTS: input signals: 2
 BC RESULTS: output signals: 1

removing temp directory.

osys> show

6. Generating Graphviz representation of design.
 writing dot description to `/home/anshi/.yosys_show.dot'
 dumping module opt_check4 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> '/home/anshi/.yosys_show.dot.pid' &

.yosys_show.dot - Dot Viewer

```

graph LR
    a((a)) --> BUF1((BUF))
    b((b)) --> BUF2((BUF))
    c((c)) --> BUF3((BUF))
    BUF1 --> AND[sky130_fd_sc_hd_xnor2_1]
    BUF2 --> AND
    AND --> BUF4((BUF))
    BUF4 --> Y((Y))

```

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

```

bad_latch_net.v bad_mux.v bad_mux_net.v bad_shift_reg.v
 demux_case.v demux_generate.v diff_ares.net.v diff_async_set.v
 dff_const4.v dff_const5.v dff_net.v dff_syncres.v
 incomp_case.v incomp_if.v incomp_if2.v mul2_net.v

```

Applications multiple_module_opt.v... Terminal verilog_files - File Manager .yosys_show.dot - Dot Viewer
File Edit View Search Terminal Help Terminal .yosys_show.dot - Dot Viewer
ABC: Scl.LibertyReadGenlib() skipped sequential cell "sky130_fd_sc_hd_sedfxtp_4".
ABC: Library "sky130_fd_sc_hd_tt_025C_lv80" from "/home/anshi/VLSI/sky130RTLDesignAndSynthesisWorkshop"
has 334 cells (94 skipped; 63 seq; 13 tri-state; 18 no func; 0 dont_use). Time = 0.25 sec
ABC: Memory = 16.00 MB. Time = 0.25 sec
ABC: Warning: Detected 9 multi-output gates (for example, "sky130_fd_sc_hd_fa_1").
ABC: + strash
ABC: + ifraig
ABC: + scorr
ABC: Warning: The network is combinational (run "fraig" or "fraig_sweep").
ABC: + dc2
ABC: + dretime
ABC: + strash
ABC: + $get -n
ABC: + $dch -f
ABC: + $nf
ABC: + $put
ABC: + write_bif <abc-temp-dir>/output.blif

6.1.2. Re-integrating ABC results.
ABC RESULTS: sky130_fd_sc_hd_a2lo_1 cells: 1
ABC RESULTS: internal signals: 3
ABC RESULTS: input signals: 3
ABC RESULTS: output signals: 1
Removing temp directory.

yosys> show
7. Generating Graphviz representation of design.
Writing dot description to '/home/anshi/.yosys_show.dot'.
Dumping module multiple_module_opt 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' &
yosys> 

```

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

6. Multiple_Module_opt_2

Y is optimized to 1 without using any Gate

```

Applications multiple_module_opt2... Terminal verilog_files - File Manager .yosys_show.dot - Dot Viewer
File Open + Save - + x 16:05 anshi
multiple_module_opt2.v
~/VLSI/sky130RTLDesignAndSynthesisWorkshop/verilog_files
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
incomp_case.v incomp_if.v incomp_if2.v mul2_net.v

```

```

Applications Terminal verilog_files - File Manager .yosys_show.dot - Dot Viewer
File Edit View Search Terminal Help Terminal .yosys_show.dot - Dot Viewer
ABC: Scl.LibertyReadGenlib() skipped sequential cell "sky130_fd_sc_hd_sedfxtp_4".
ABC: Library "sky130_fd_sc_hd_tt_025C_lv80" from "/home/anshi/VLSI/sky130RTLDesignAndSynthesisWorkshop"
has 334 cells (94 skipped; 63 seq; 13 tri-state; 18 no func; 0 dont_use). Time = 0.25 sec
ABC: Memory = 16.00 MB. Time = 0.25 sec
ABC: Warning: Detected 9 multi-output gates (for example, "sky130_fd_sc_hd_fa_1").
ABC: + strash
ABC: + ifraig
ABC: + scorr
ABC: Warning: The network is combinational (run "fraig" or "fraig_sweep").
ABC: + dc2
ABC: + dretime
ABC: + strash
ABC: + $get -n
ABC: + $dch -f
ABC: + $nf
ABC: + $put
ABC: + write_bif <abc-temp-dir>/output.blif

12.1.2. Re-integrating ABC results.
ABC RESULTS: const0_0
ABC RESULTS: internal signals: 0
ABC RESULTS: input signals: 0
ABC RESULTS: output signals: 0
Removing temp directory.

yosys> show
13. Generating Graphviz representation of design.
Writing dot description to '/home/anshi/.yosys_show.dot'.
Dumping module multiple_module_opt2 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' &
yosys> 

```

Sequential Optimization- Sequential Constant Propagation

Lab Examples of DFF Flip-Flops

1. Dff_const1

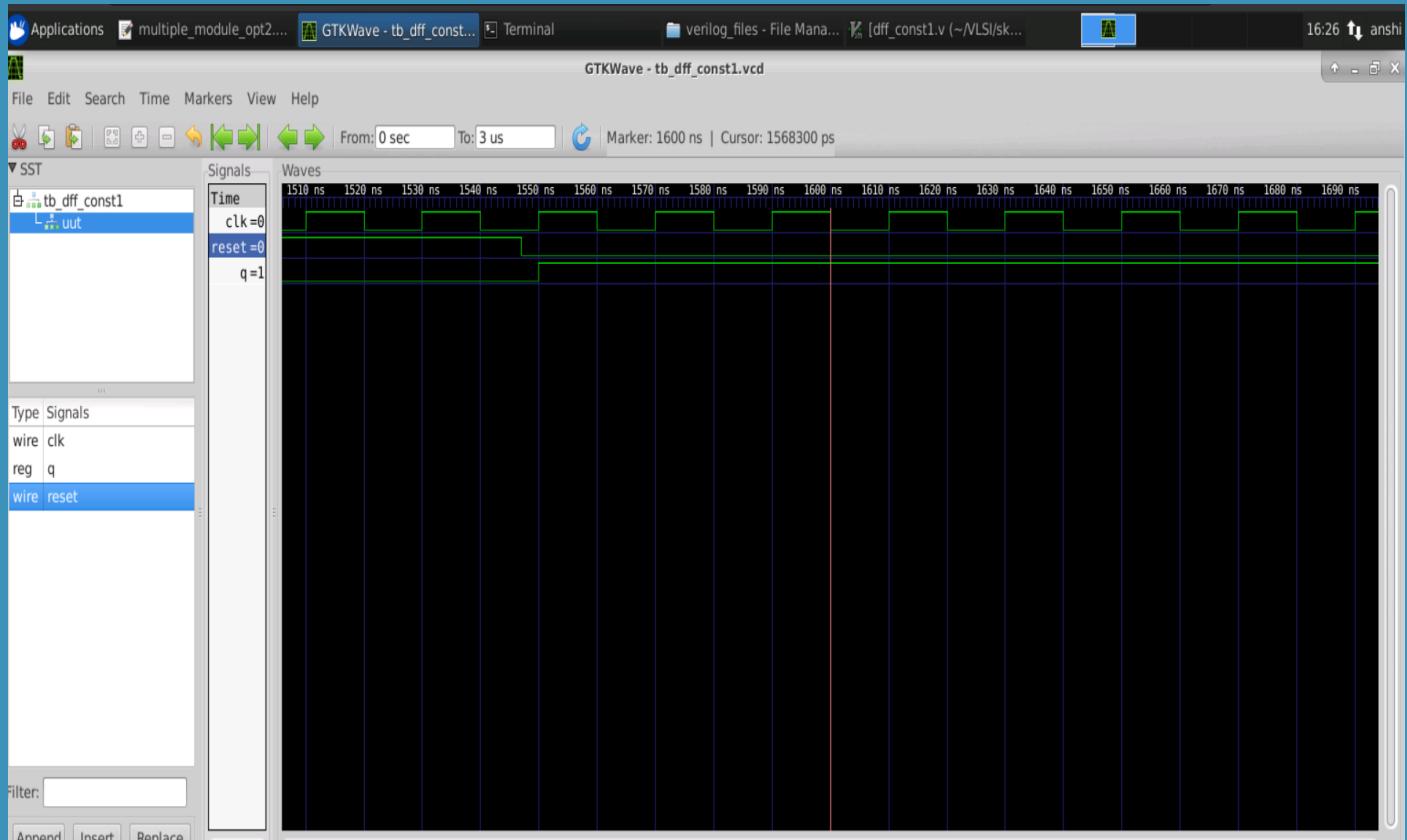
```
ABC: + &get -n
ABC: + &dch -f
ABC: + &nf
ABC: + &put
ABC: + write_blf
begin
    if(reset)
        q <= 1'b0;
    else
        q <= 1'b1;
end
endmodule

yosys> show
yosys> show multiple
module dff_const1(input clk, input reset, output reg q);
    always @(posedge clk, posedge reset)
begin
    if(reset)
        q <= 1'b0;
    else
        q <= 1'b1;
end
endmodule

7. Generating Graphs
ERROR: For formats
yosys> show multiple
module dff_const2(input clk, input reset, output reg q);
    always @(posedge clk, posedge reset)
begin
    if(reset)
        q <= 1'b1;
    else
        q <= 1'b0;
end
endmodule

8. Generating Graphs
Writing dot description
Dumping module multi...
Exec: { test -f '/home/anshi/rtlworkshop/dff_const2.v' > /dev/null
'; } 3> '/home/anshi/rtlworkshop/dff_const2.v'
yosys> exit
End of script. Logfile: /home/anshi/rtlworkshop/dff_const2.v
Yosys 0.9+4081 (git 0.9+4081)
Time spent: 200% 2ns
anshi@rtlworkshop:~/rtlworkshop$
```

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



Design: A dff is realised

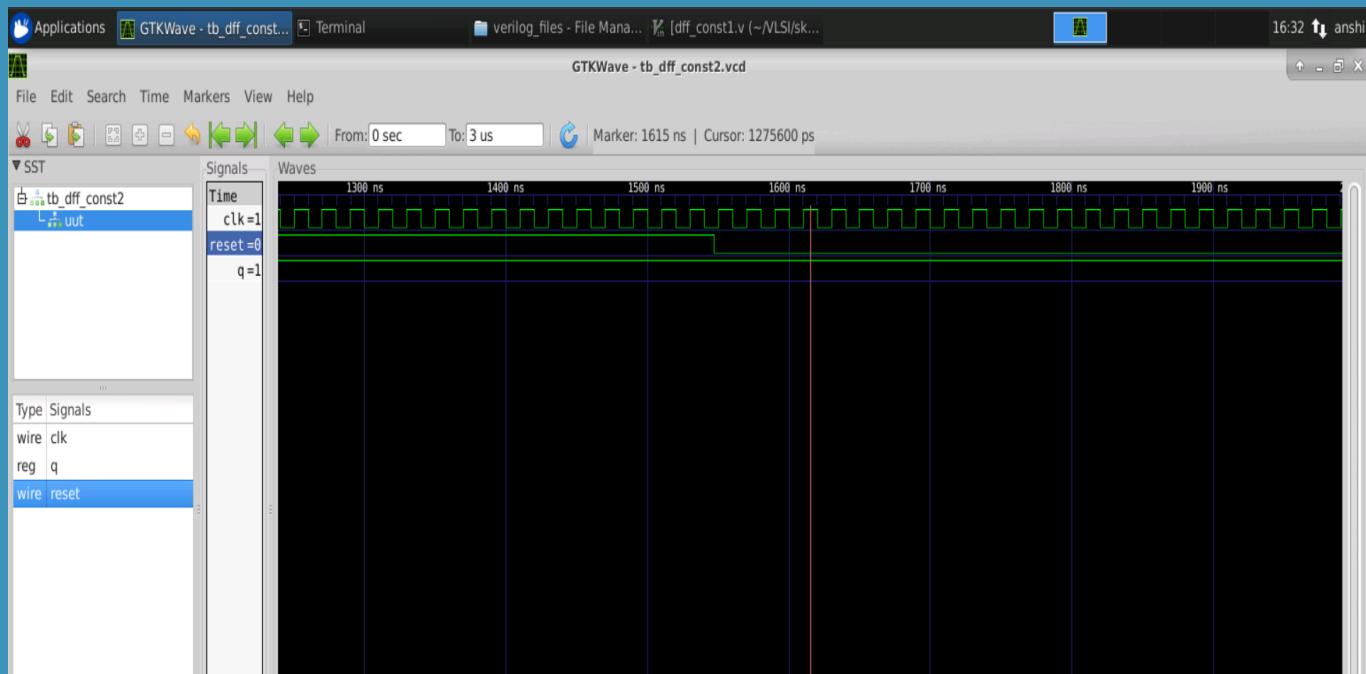
```
ABC: Scl_LibertyReadGenlib() skipped sequence
ABC: Library "sky130_fd_sc_hd_tt_025C_1v80" has 334 cells (94 skipped: 63 seq; 13 tri-state)
ABC: Memory = 16.00 MB. Time = 0.28 s
ABC: Warning: Detected 9 multi-output gates
ABC: + strash
ABC: + ifraig
ABC: + scorr
ABC: Warning: The network is combinational
ABC: + dc2
ABC: + dftime
ABC: + strash
ABC: + &get -n
ABC: + &dch -f
ABC: + &nf
ABC: + &put
ABC: + write_bif <abc-temp-dir>/output.bif
6.1.2. Re-integrating ABC results.
ABC RESULTS: sky130_fd_sc_hd_clkinv_1 cell
ABC RESULTS: internal signals:
ABC RESULTS: input signals:
ABC RESULTS: output signals:
Removing temp directory.

yosys> show
7. Generating Graphviz representation of design
Writing dot description to '/home/anshi/.yosys/dot'
Dumping module dff_const1 to page 1.
Exec: { test -f '/home/anshi/.yosys/dot.pid' && fuser -s '/home/anshi/.yosys/dot.pid' 2> /dev/null; } || ( echo $$ >&3; exec xdot '/home/anshi/.yosys/dot.pid' &
'; ) 3> '/home/anshi/.yosys/dot.pid' &

yosys>
```

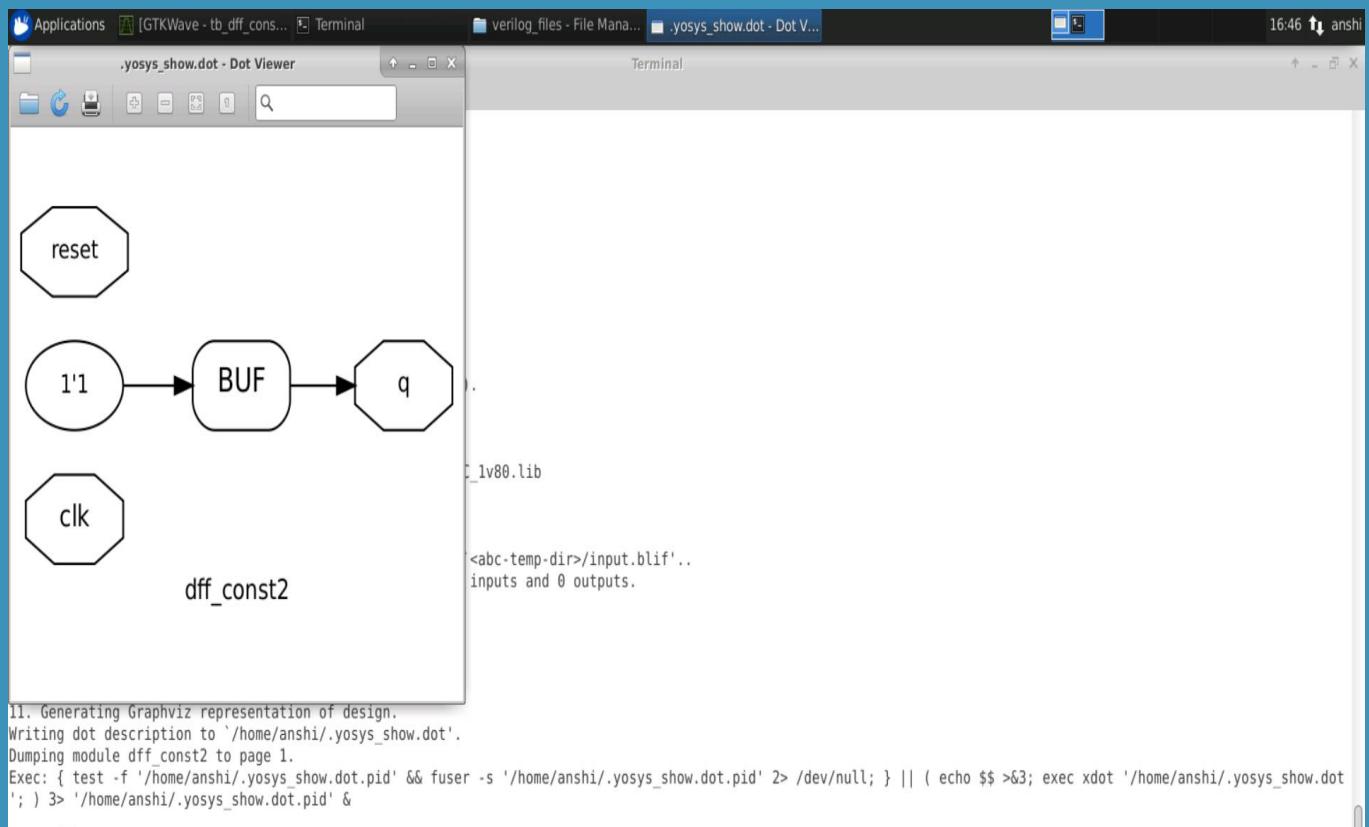
2.Dff_const2

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



3.Dff_const3

Applications dff_const3.v (~VLSI/sk... Terminal verilog_files - File Mana... 16:57 anshi

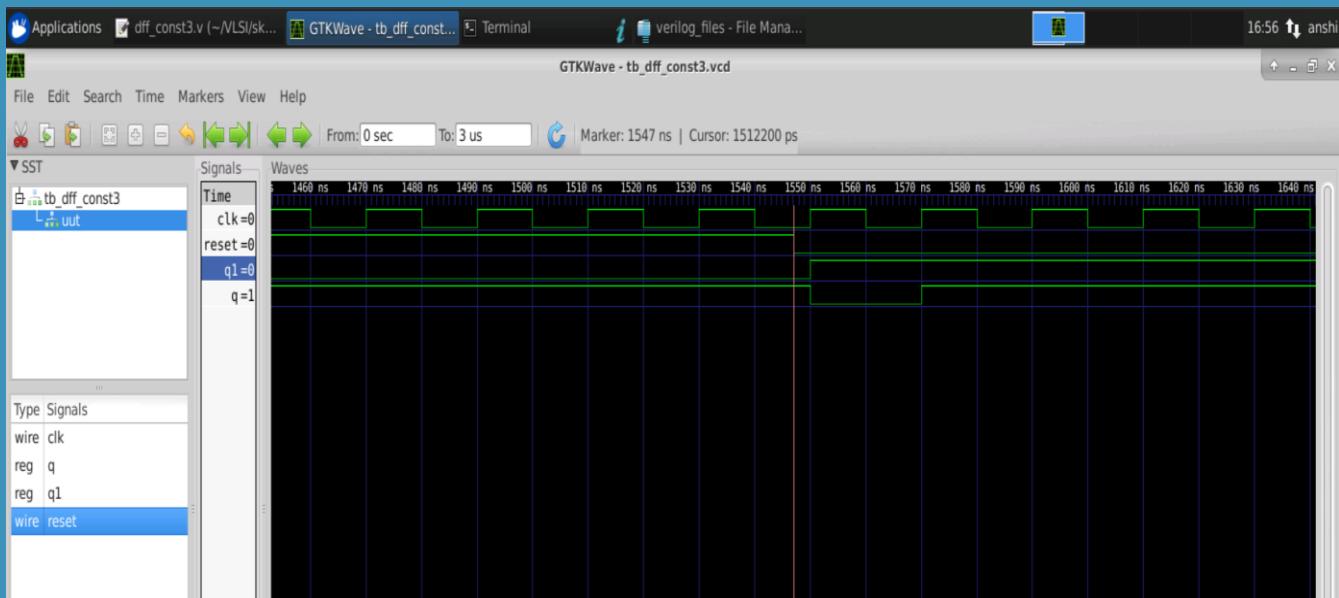
```

module dff_const3(input clk, input reset, output reg q);
reg q1;

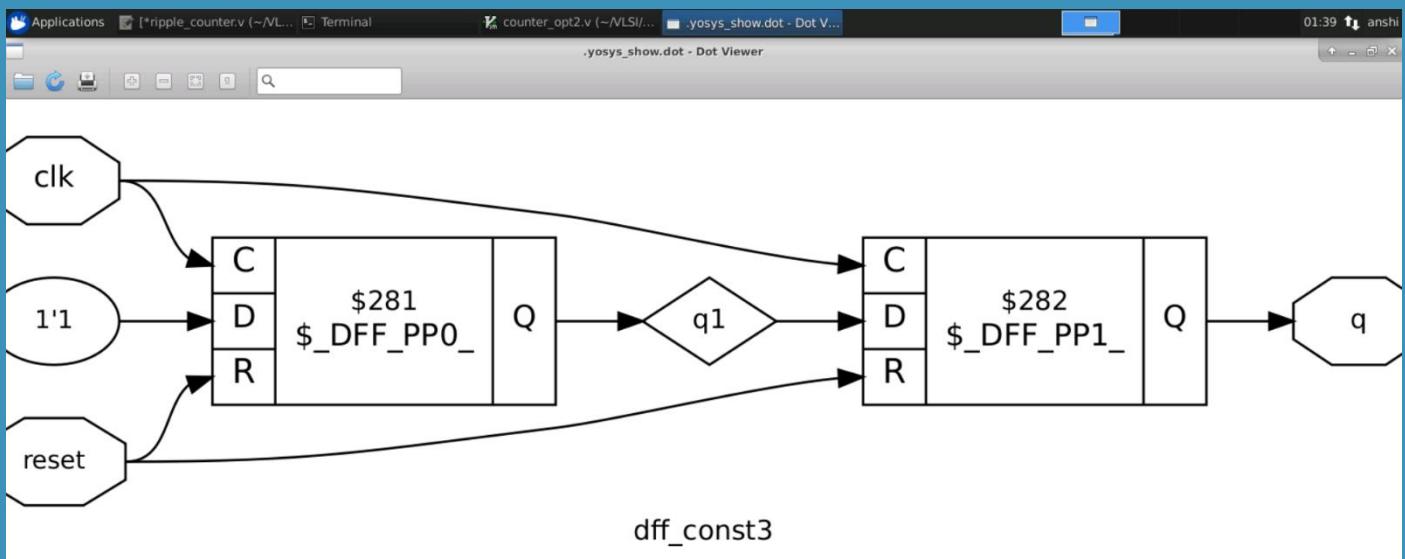
always @(posedge clk, posedge reset)
begin
    if(reset)
        begin
            q <= 1'b1;
            q1 <= 1'b0;
        end
    else
        begin
            q1 <= 1'b1;
            q <= q1;
        end
end
endmodule

```

bad_latch_net.v bad_mux.v bad_mux_net.v bad_shift_reg.v
demux_case.v demux_generate.vdff_ares.net.vdff_async_set.v
dff_const4.vdff_const5.vdff_net.vdff_syncres.v



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



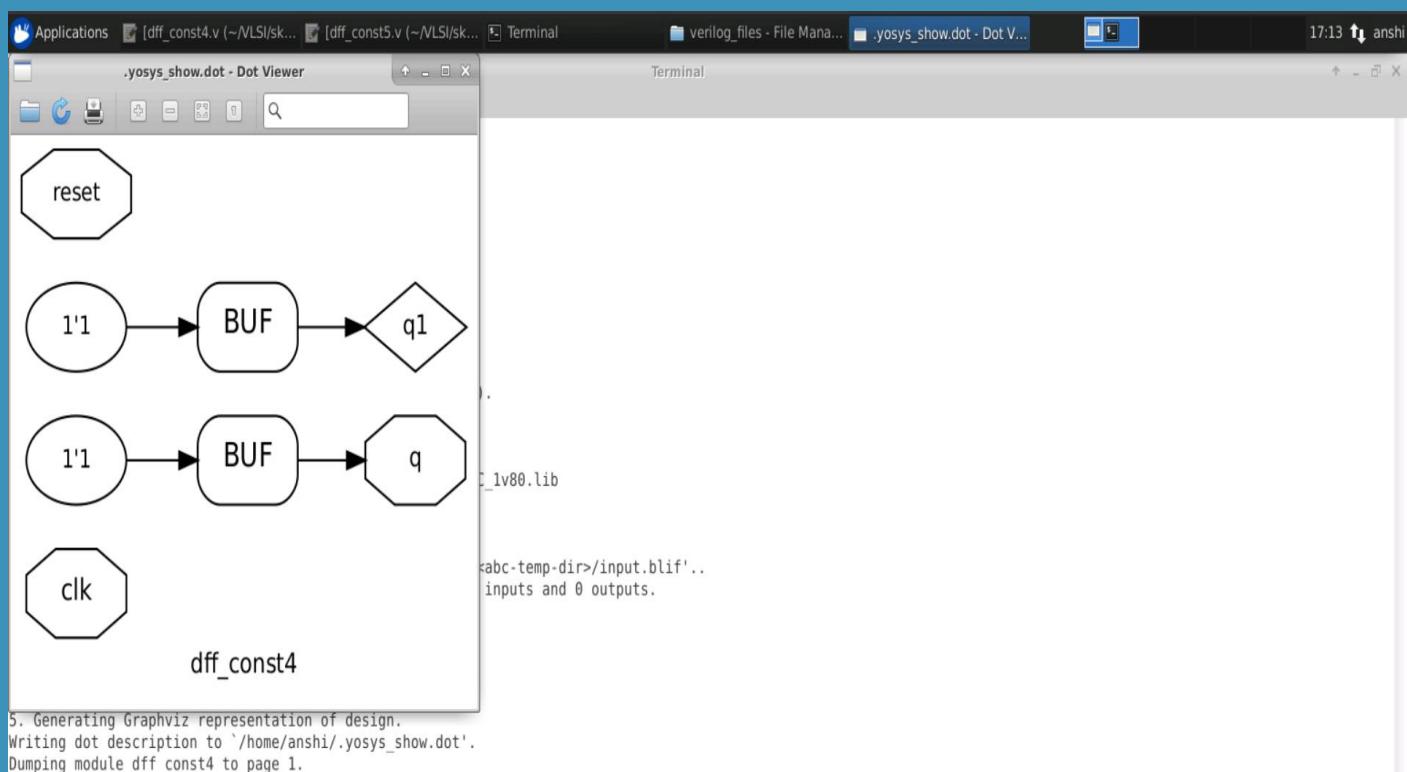
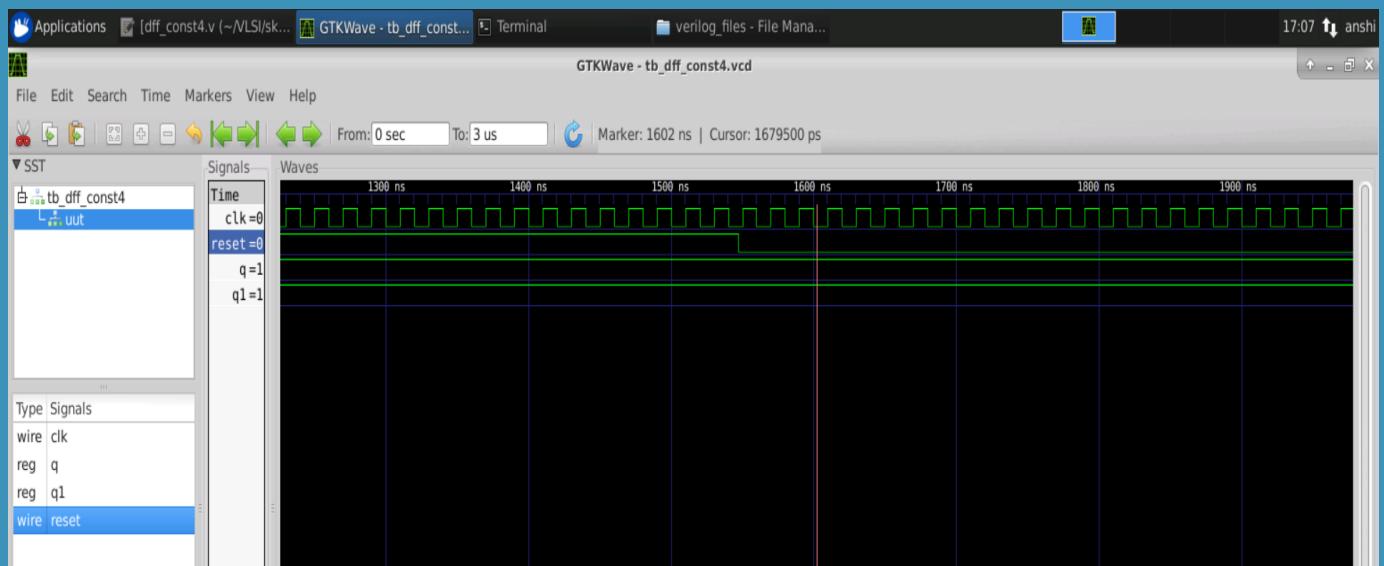
3. Dff_const4

```

module dff_const4(input clk, input reset, output reg q);
reg q1;

always @ (posedge clk, posedge reset)
begin
    if(reset)
        begin
            q <= 1'b1;
            q1 <= 1'b1;
        end
    else
        begin
            q1 <= 1'b1;
            q <= q1;
        end
end
endmodule

```



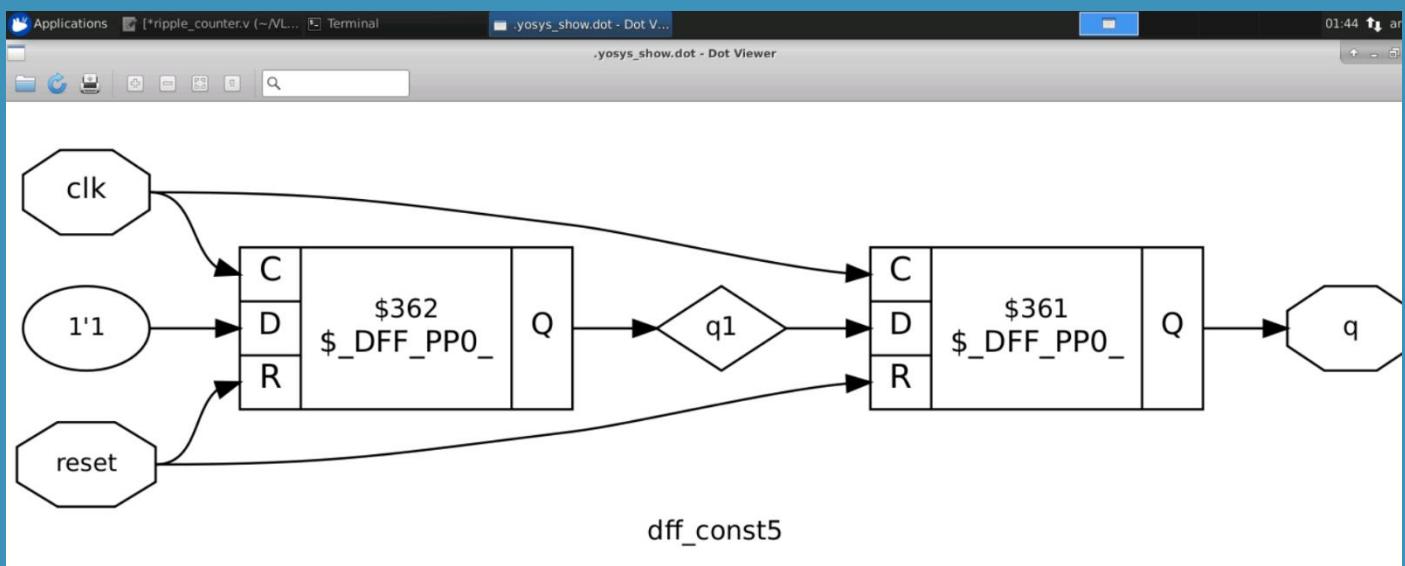
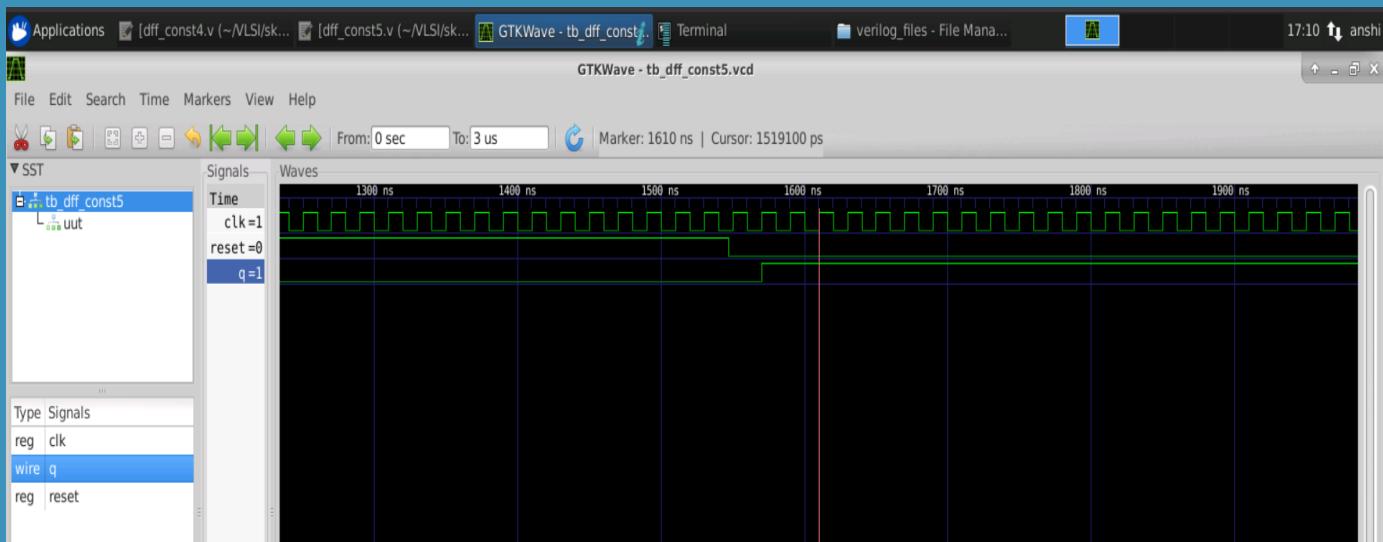
When reset q and q1 are 1 else q1 is 1 and q is assigned q1. So, both the flip-flops are always one

5.Dff_const5.v

```

module dff_const5(input clk, input reset, output reg q);
  parameter q1;
  begin
    if(reset)
      begin
        q <= 1'b0;
        q1 <= 1'b0;
      end
    else
      begin
        q1 <= 1'b1;
        q <= q1;
      end
  end
endmodule

```



Unused Output Optimization

Applications Applications Terminal verilog_files - File Mana... counter_opt.v (~/VLSI/sky130RTLDesignAndSynthesisWorkshop/verilog_files) - GVIM 17:50

File Edit Tools Syntax Buffers Window Help

```
module counter_opt (input clk , input reset , output q);
reg [2:0] count;
assign q = count[0];

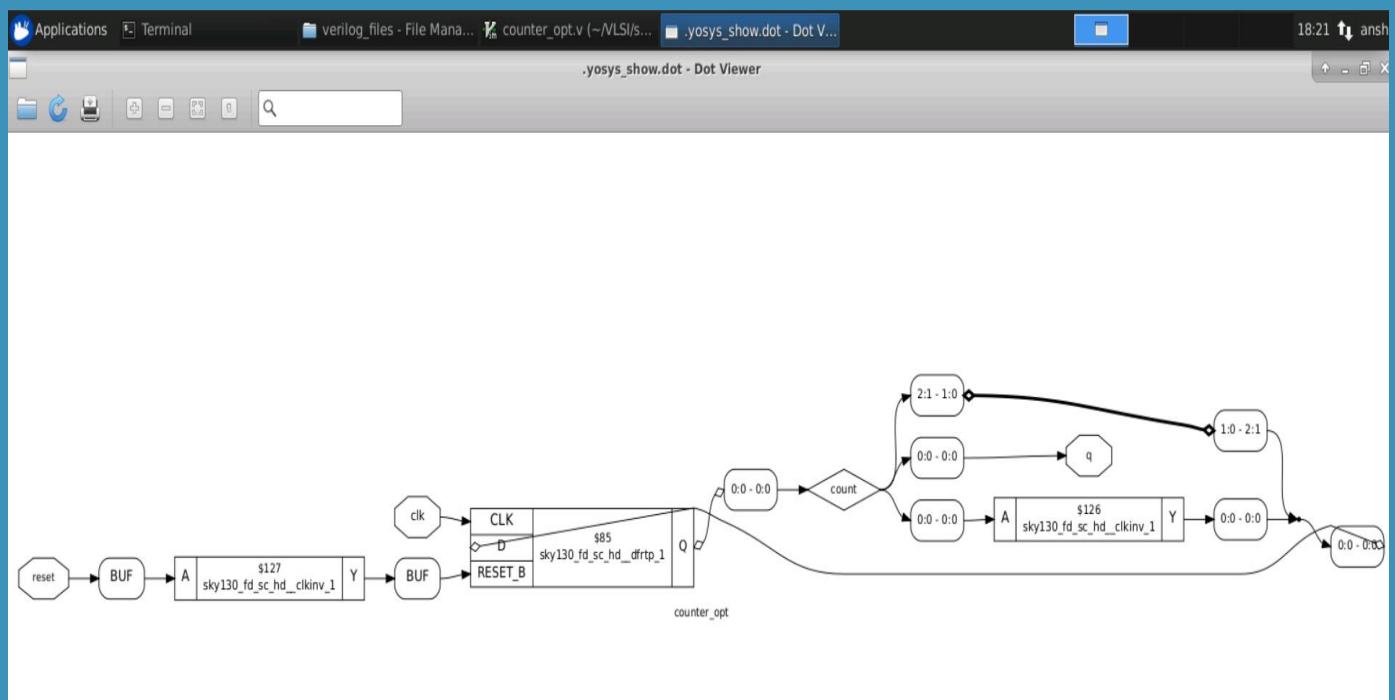
always @(posedge clk ,posedge reset)
begin
    if(reset)
        count <= 3'b000;
    else
        count <= count + 1;
end

endmodule
```

./yosys_show.dot
./yosys_show.dot

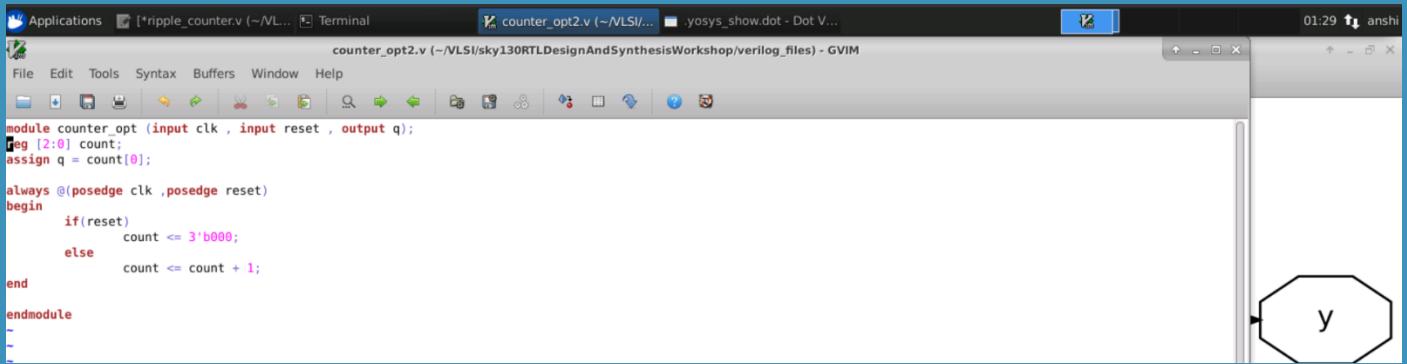
Applications Terminal verilog_files - File Mana... counter_opt.v (~VLSI/s... 18:18 anshi
 Terminal
 File Edit View Search Terminal Help
 Removed 0 unused cells and 2 unused wires.
 <suppressed ~1 debug messages>
 3.23.5. Finished fast OPT passes.
 3.24. Executing HIERARCHY pass (managing design hierarchy).
 3.24.1. Analyzing design hierarchy..
 Top module: \counter_opt
 3.24.2. Analyzing design hierarchy..
 Top module: \counter_opt
 Removed 0 unused modules.
 3.25. Printing statistics.
 === counter_opt ===
 Number of wires: 5
 Number of wire bits: 9
 Number of public wires: 4
 Number of public wire bits: 6
 Number of memories: 0
 Number of memory bits: 0
 Number of processes: 0
 Number of cells: 2
 \$_DFF_PP0_ 1
 \$_NOT_ 1
 3.26. Executing CHECK pass (checking for obvious problems).
 Checking module counter_opt...
 Found and reported 0 problems.

yosys>



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

Counter_opt2



File Edit Tools Syntax Buffers Window Help

```
module counter_opt (input clk , input reset , output q);
reg [2:0] count;
assign q = count[0];

always @(posedge clk ,posedge reset)
begin
    if(reset)
        count <= 3'b000;
    else
        count <= count + 1;
end
endmodule
```

