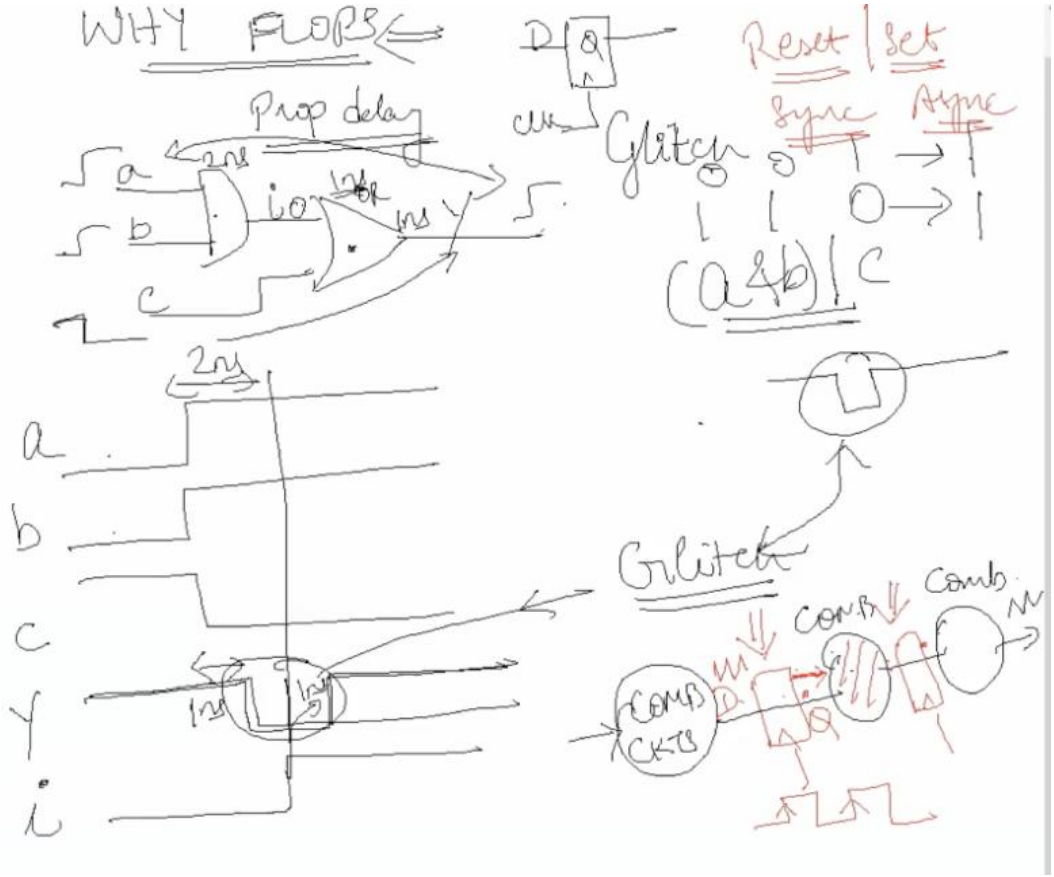
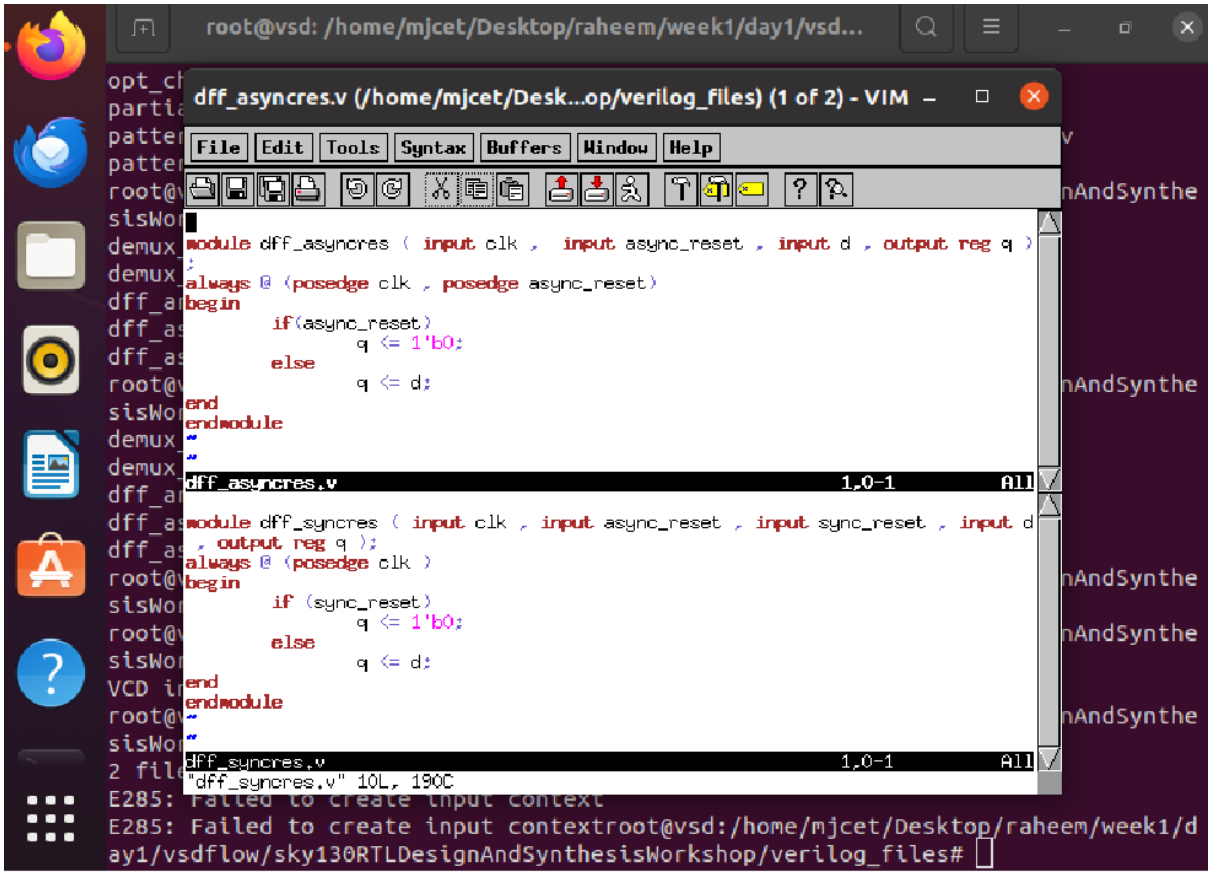
Various Flop Coding Styles and Optimization

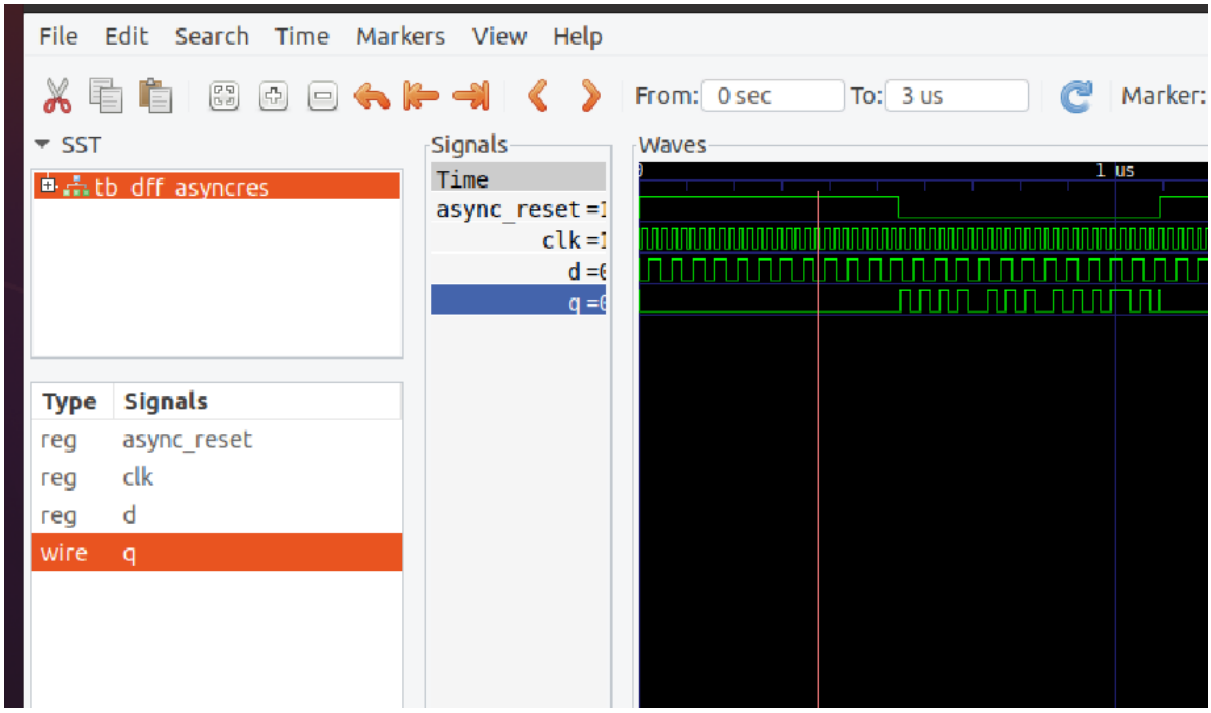
Why do we need flops and how do they prevent glitches in the circuit?

Glitches can occur in digital circuits due to various reasons such as signal delays, noise, or timing issues. Flops prevent glitches during the operation in the following ways:

- Synchronization: Flops are edge-triggered devices, meaning they respond only to transitions of the input signal (e.g., rising edge, falling edge). This synchronization ensures that the output changes only at specific points, reducing the likelihood of glitches caused by transient signal variations.
- Timing Control: Flops are typically controlled by a clock signal, ensuring that all circuit operations occur synchronously. This eliminates timing issues that could lead to glitches due to data arriving at different times.
- 




```
root@vsd:/home/mjcet/Desktop/raheem/week1/day1/vsdflow/sky130RTLDesignAndSynthesisWorkshop/verilog_files# iverilog d
demux_case.v            dff_asyncres.v          dff_const4.v
demux_generate.v        dff_async_set.v         dff_const5.v
dff_ares.net.v          dff_const1.v            dff_net.v
dff_asyncres_net.v      dff_const2.v            dff_syncres.v
dff_asyncres_syncres.v  dff_const3.v
root@vsd:/home/mjcet/Desktop/raheem/week1/day1/vsdflow/sky130RTLDesignAndSynthesisWorkshop/verilog_files# iverilog dff_asyncres.v tb_dff_asyncres.v
root@vsd:/home/mjcet/Desktop/raheem/week1/day1/vsdflow/sky130RTLDesignAndSynthesisWorkshop/verilog_files# ./a.out
VCD info: dumpfile tb_dff_asyncres.vcd opened for output.
root@vsd:/home/mjcet/Desktop/raheem/week1/day1/vsdflow/sky130RTLDesignAndSynthesisWorkshop/verilog_files# gvim dff_asyncres.v -o dff_syncres.v
2 files to edit
E285: Failed to create input context
E285: Failed to create input contextroot@vsd:/home/mjcet/Desktop/raheem/week1/day1/vsdflow/sky130RTLDesignAndSynthesisWorkshop/verilog_files#
```



read_liberty -lib ../lib/sky130_fd_sc_hd__tt_025C_1v80.lib

read_verilog dff_asyncres.v

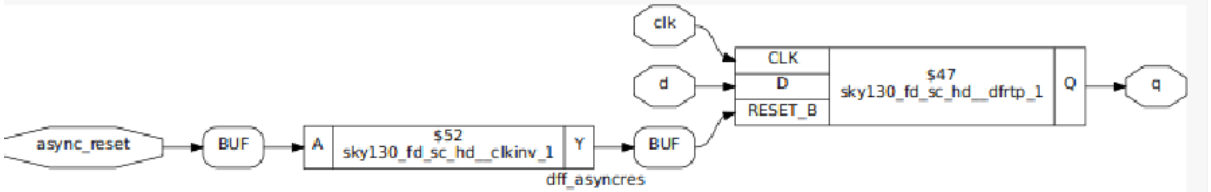
synth -top dff_asyncres

dfflibmap -liberty ../lib/sky130_fd_sc_hd__tt_025C_1v80.lib

abc -liberty ../lib/sky130_fd_sc_hd__tt_025C_1v80.lib

show

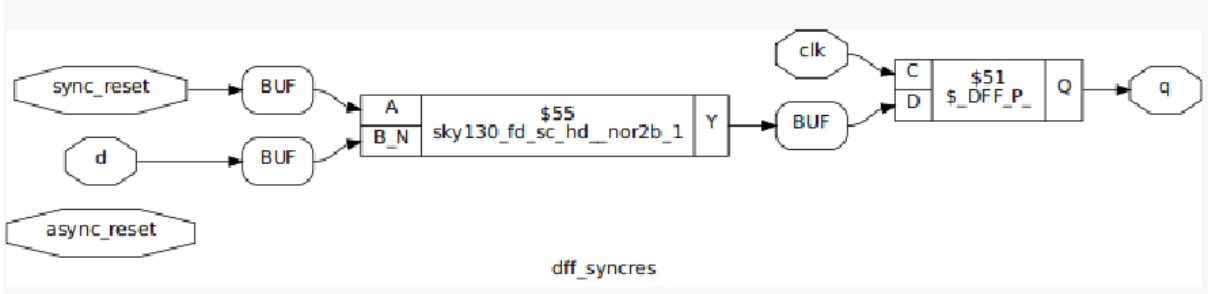
```
ABC: + retime
ABC: + strash
ABC: + &get -n
ABC: + &dch -f
ABC: + &nf
ABC: + &put
ABC: + write_blif <abc-temp-dir>/output.blif

5.1.2. Re-integrating ABC results.
ABC RESULTS:   sky130_fd_sc_hd__clkinv_1 cells:        1
ABC RESULTS:        internal signals:        0
ABC RESULTS:           input signals:        1
ABC RESULTS:          output signals:        1
Removing temp directory.
```




```
=== dff_syncres ===

   Number of wires:                  6
   Number of wire bits:              6
   Number of public wires:           5
   Number of public wire bits:       5
   Number of memories:               0
   Number of memory bits:            0
   Number of processes:              0
   Number of cells:                  2
     $_ANDNOT_                       1
     $_DFF_P_                        1

3.27. Executing CHECK pass (checking for obvious problems).
checking module dff_syncres..
found and reported 0 problems.
```

LibreOffice Writer

```
ABC: + &put
ABC: + write_blif <abc-temp-dir>/output.blif

4.1.2. Re-integrating ABC results.
ABC RESULTS:   sky130_fd_sc_hd__nor2b_1 cells:        1
ABC RESULTS:        internal signals:        0
ABC RESULTS:           input signals:        2
ABC RESULTS:          output signals:        1
Removing temp directory.
```

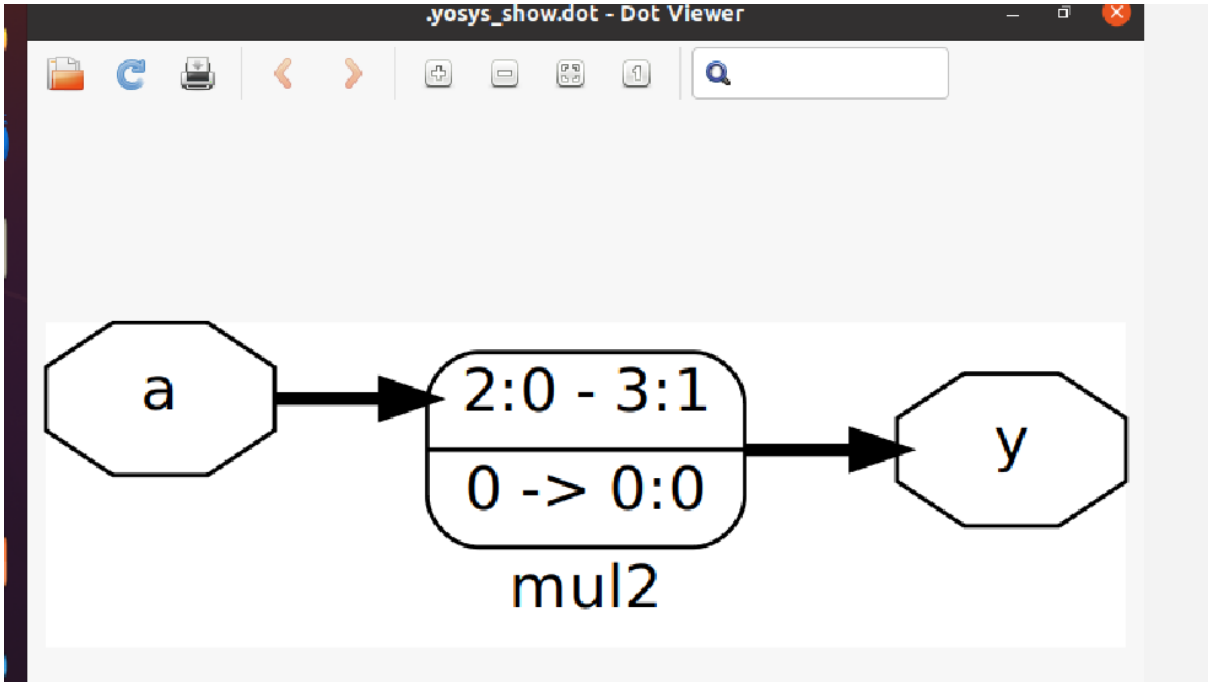


Mult_2

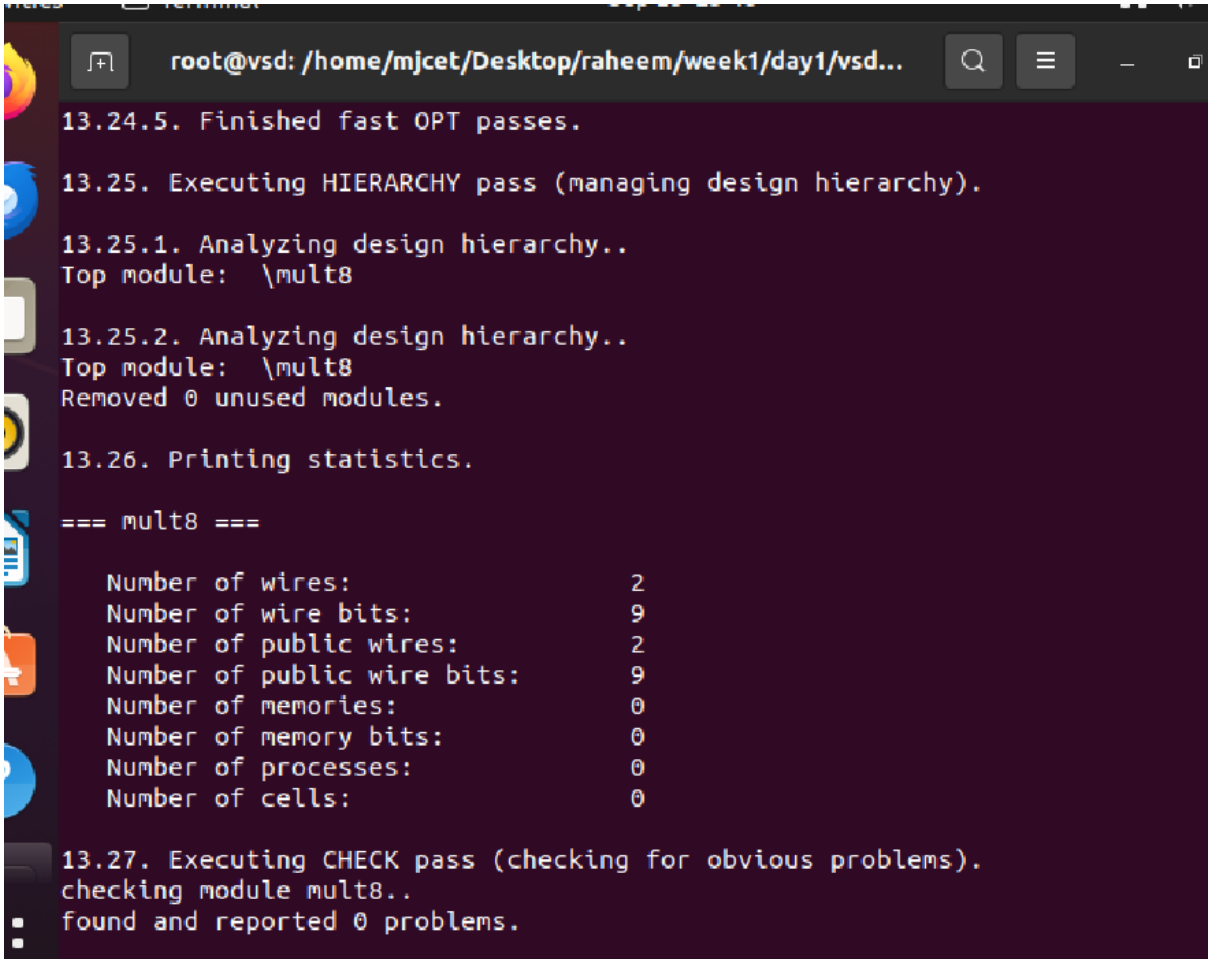
```
yosys> read_verilog mu
mul2_net.v                multiple_module_net.v     multiple_modules_hier.v
mult_2.v                  multiple_module_opt.v     mux_generate.net.v
mult_8.v                  multiple_module_opt2.v    mux_generate.v
multiple_module_herar.v   multiple_modules.v        mux_spice.v
multiple_module_hier.v    multiple_modules_flat.v

yosys> read_verilog mu
mul2_net.v                multiple_module_net.v     multiple_modules_hier.v
mult_2.v                  multiple_module_opt.v     mux_generate.net.v
mult_8.v                  multiple_module_opt2.v    mux_generate.v
multiple_module_herar.v   multiple_modules.v        mux_spice.v
multiple_module_hier.v    multiple_modules_flat.v

yosys> read_verilog mult_2.v
```

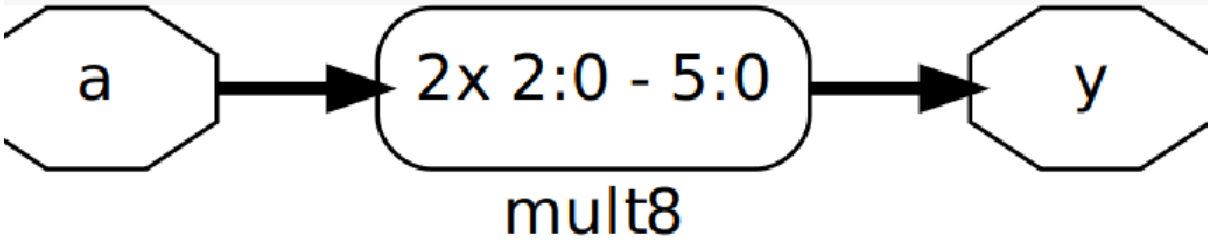
```
=== mul2 ===

   Number of wires:                 2
   Number of wire bits:             7
   Number of public wires:          2
   Number of public wire bits:      7
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                 0

8.27. Executing CHECK pass (checking for obvious problems).
checking module mul2..
found and reported 0 problems.
```

```
=== mul2 ===

   Number of wires:                 2
   Number of wire bits:             7
   Number of public wires:          2
   Number of public wire bits:      7
   Number of memories:              0
   Number of memory bits:           0
   Number of processes:             0
   Number of cells:                 0

8.27. Executing CHECK pass (checking for obvious problems).
checking module mul2..
found and reported 0 problems.

yosys> abc -liberty ../lib/sky130_fd_sc_hd__tt_025C_1v80.lib

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

9.1. Extracting gate netlist of module `\mul2' to `<abc-temp-dir>/input.blif'..
Extracted 0 gates and 0 wires to a netlist network with 0 inputs and 0 outputs.
Don't call ABC as there is nothing to map.
Removing temp directory.
```

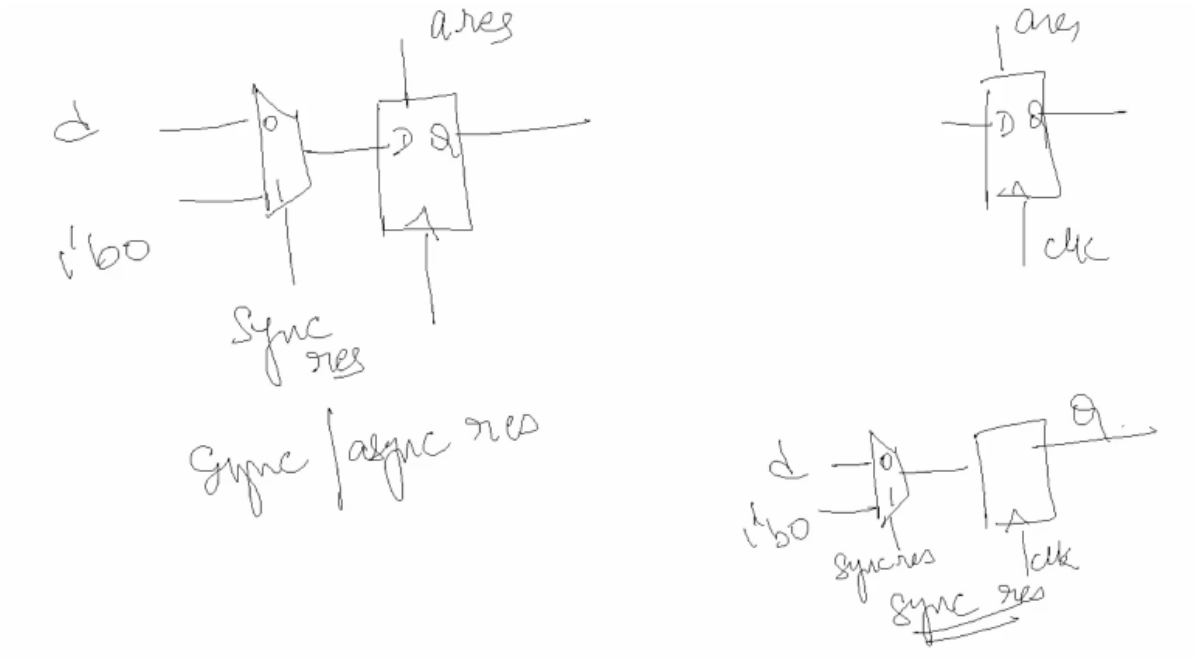


Mult8


root@vsd: /home/mjcet/Desktop/raheem/week1/day1/vsd...
13.24.5. Finished fast OPT passes.
13.25. Executing HIERARCHY pass (managing design hierarchy).
13.25.1. Analyzing design hierarchy..
Top module: \mult8
13.25.2. Analyzing design hierarchy..
Top module: \mult8
Removed 0 unused modules.
13.26. Printing statistics.
=== mult8 ===
Number of wires: 2
Number of wire bits: 9
Number of public wires: 2
Number of public wire bits: 9
Number of memories: 0
Number of memory bits: 0
Number of processes: 0
Number of cells: 0
13.27. Executing CHECK pass (checking for obvious problems).
checking module mult8..
found and reported 0 problems.


a
2x 2:0 - 5:0
y
mult8

Different types of flops



The screenshot below shows DFF with asynchronous reset HDL simulation in Iverilog and waveform display in GTKwave. Irrespective of the clock and d, as soon as async_reset=1, q=0.

Synthesizing flops

The command to synthesize ***DFF with asynchronous reset*** as an example

read_liberty -lib ../lib/sky130_fd_sc_hd__tt_025C_1v80.lib

read_verilog dff_asyncres.v

synth -top dff_asyncres

dfflibmap -liberty ../lib/sky130_fd_sc_hd__tt_025C_1v80.lib

abc -liberty ../lib/sky130_fd_sc_hd__tt_025C_1v80.lib

show

On synthesizing ***DFF with synchronous reset*** we get NOR gate with inverted d as shown in the screenshot below. However,on evaluating the boolean expression, we reached the same logic realization.

Using the stat command, all the cells used for logic synthesis are visible even though it is not evident from the statistics of doing synthesis.

Synthesizing mult2 (multiply by 2)

To implement y[3:0] = 2*a[2:0], we append a 1'b0 to the a[2:0] i.e, y[3:0] = {a[2:0],0}. This is also equal to left shift the input bits by 1. This can be realized by just wiring. So we expect no hardware which is also seen in the screenshot below, analysis after synthesis and show. The command 'abc' is not required for mapping when there are no cells.

Synthesizing mult9 (multiply by 9 or 8+1)

y=9*a can be considered 8*a+1*a To implement y[5:0] = 9*a[2:0], we append 000 to a[2:0] and then add a i.e, y[5:0] = {a[2:0],000} + a[2:0]. This can be realized just by wiring. So we expect no hardware which is also seen in the screenshot below, analysis after synthesis and show. The command 'abc' is not required for mapping when there are no cells.