REPORT

ON

Design of a 3 bit asynchronous counter using D flip-flops

AND

Implementation of ‘1110’ overlapping sequence detector circuit using Verilog

ANALOG AND DIGITAL VLSI DESIGN

DIGITAL ASSIGNMENT

Problem number- 74

PREPARED BY:

KSHITIJ ARORA – 2017A3PS0197P

BHAGYAM GUPTA - 2017A8PS0525P

NAGESH SAMANE - 2017A8PS0612P

Birla Institute of Technology and Science, Pilani, Pilani Campus

October 2019

PROBLEM STATEMENT

Problem 74

(a) Design a 3-bit asynchronous counter using D flip flops.

(b) Design a sequential circuit which produces an output ‘1’ every time a sequence “1110” is detected, otherwise the output is ‘0’. Overlapping sequence detection has to be implemented.

ABSTRACT

The objective of the assignment was to get familiarize with the digital circuit design, layout, and simulation techniques used widely in the industry. In the first part of it we had to design the schematic of a 3 bit asynchronous counter using D flip-flops and had to figure out the optimal logic along with the sizes of the transistors for the optimum delay, also we had to design the layout of this schematic with proper design rules to actuate the design with minimum parasitic possible. In the second part of it we had to write and simulate a Verilog code for the Finite State Machine (FSM) which act as a sequence detector. While working on this assignment we got to know how our digital circuits/units function at the circuit level by the coordination between voltages and currents hence we got a better insight into the Digital VLSI Design.

TABLE OF CONTENTS

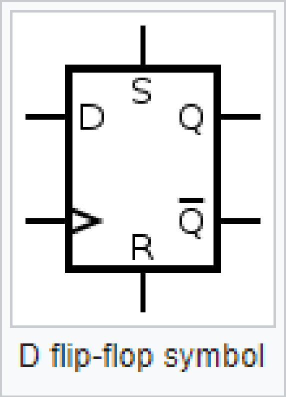
1. Introduction
2. Schematic of a D Flip-flop
3. Reason for choosing the Logic Family
4. W/L Calculations
5. Schematic of 3 bit Asynchronous Counter
6. Transient Analysis of Counter using Schematic
7. Layout of Flip - Flop and Counter
8. Layout analysis
    a. DRC
    b. LVS
    c. PEX
9. Sequence Detector
10. Conclusion
11. References

INTRODUCTION:

A flip-flop is a circuit that has two stable states and can be used to store state information.
The D flip-flop is a widely used flip-flop. It is also known as a "data" or "delay" flip-flop. The D flip-flop captures the value of the D-input at a definite portion of the clock cycle (such as the rising/falling edge of the clock). That captured value becomes the Q output. At other times, the output Q does not change

TRUTH TABLE:

INPUT(D)	CLOCK(CLK)	OUTPUT(Q)	Q’
0	LOW TO HIGH	NO CHANGE	NO CHANGE
1	LOW TO HIGH	NO CHANGE	NO CHANGE
0	HIGH TO LOW	0	1
1	HIGH TO LOW	1	0



D flip-flop symbol

Figure 1. Symbol of D flip-flop

SCHEMATIC OF D FLIP-FLOP:

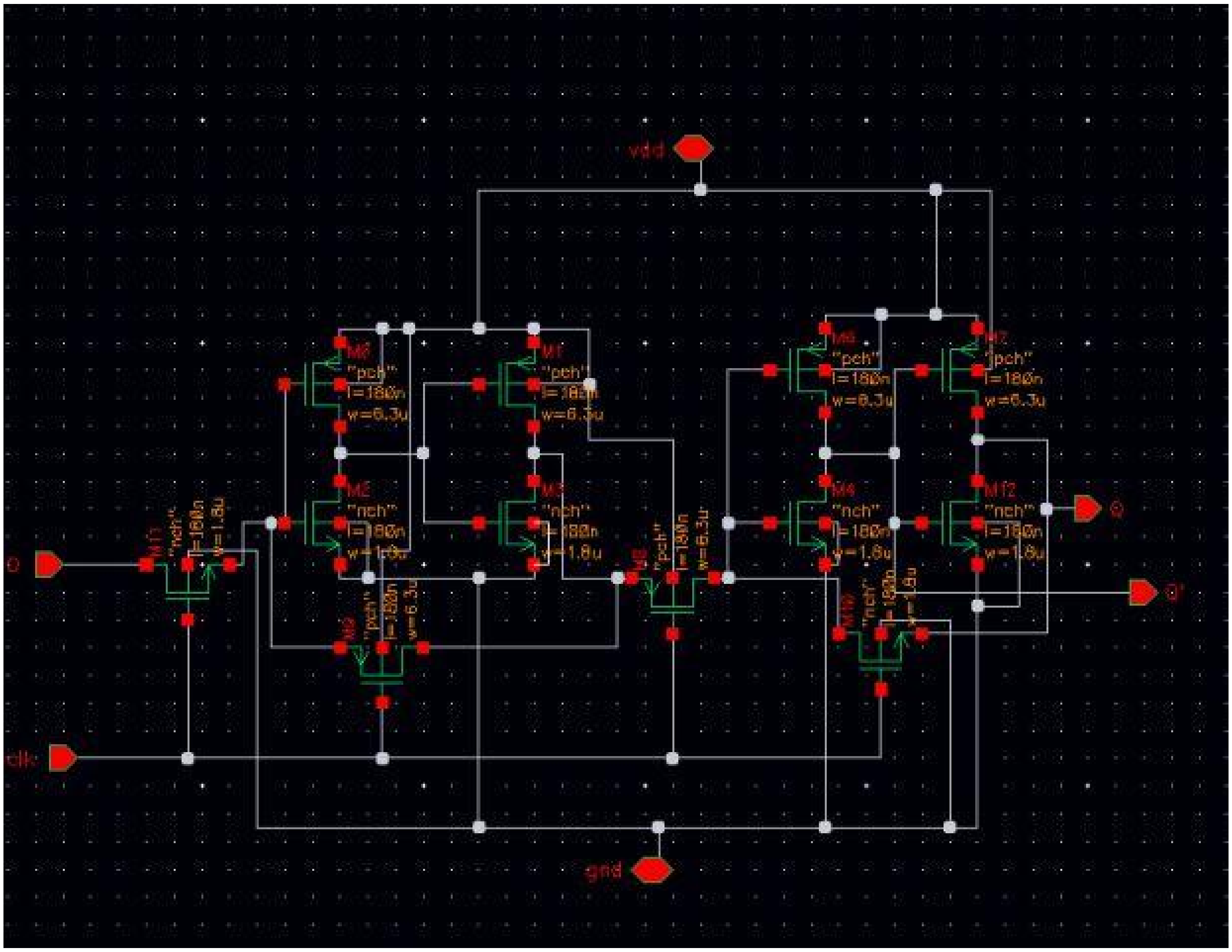


Figure 2. Schematic of D flip-flop designed in Cadence Virtuoso

This schematic of a flip-flop is a Pass Transistor family implementation of a flip-flop. This schematic was used to generate a symbol of D Flip-flop, in our working Cadence Library, which is then further used to design the 3-bit counter.

REASON FOR CHOOSING THE LOGIC FAMILY:

The standard approach of a D Flip-flop uses Master-Slave Configuration, which requires multiple NAND gates to implement. A standard D flip-flop would require 8 NAND gates, as well as 1 NOT gate.

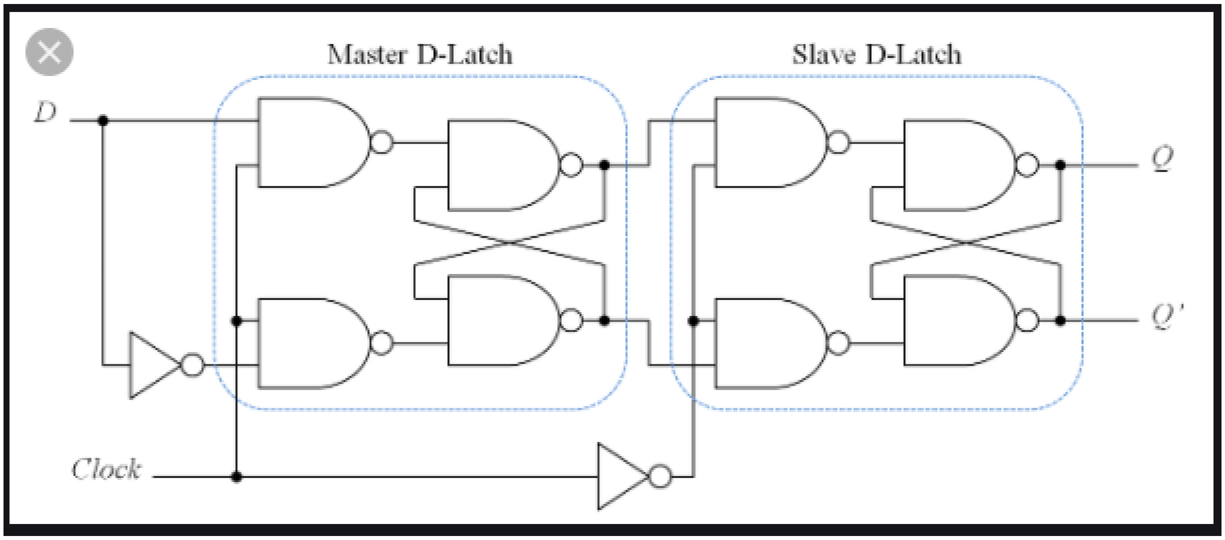


Figure 3. CMOS NAND implementation of master slave D flip-flop

Implementing this circuit in any logic family (CMOS, PTL, TG) would result in very high delay values, as the critical path length is very high. Hence, we had to look for other logic styles which would result in a D flip-flop using lesser number of gates, hence a smaller critical path.

The design that we chose consisted of Pass Transistors as well as CMOS inverters for the master-slave latches. The two chained inverters are in memory state when the PMOS loop transistor is on, that is when clk = 0. Other two chain inverters on the right hand side acts in the opposite way. This results in a flip-flop which changes its state during the falling edge of the clock.

This Pass Transistor series consists of only 4 series connected gates, which results in a tolerable delay. The major issue faced while using Pass Transistors is the case of degraded ‘1’ and ‘0’. These problems are resolved by the use of Buffers (made up of 2 CMOS Inverters), which helps us in providing a proper ‘HIGH’ or ‘LOW’ at the input of the next stage, helping in maintaining a proper output.

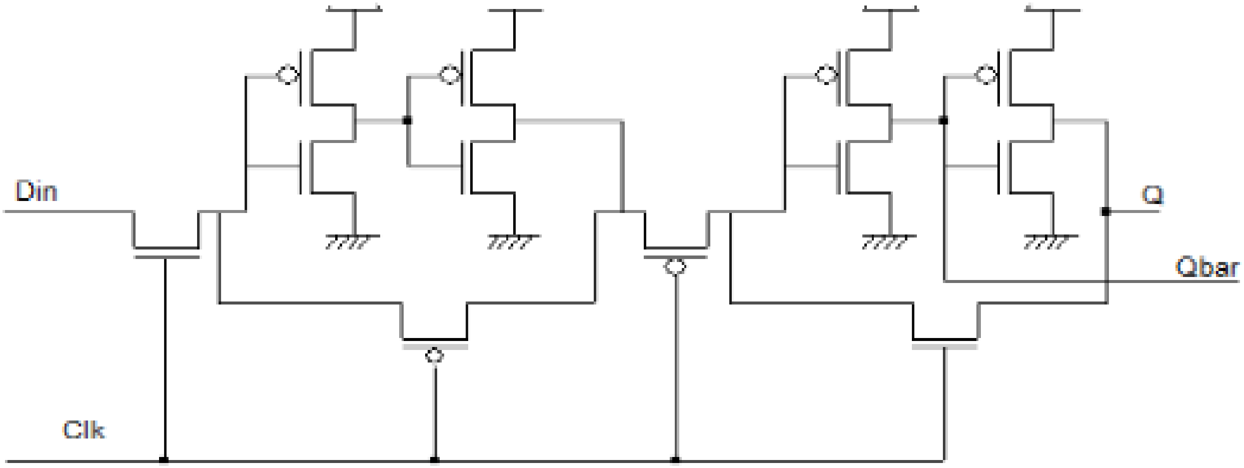


Figure 4. Implementation of D flip-flop using Pass Transistor Logic and CMOS buffers

Now, the next step of consideration was delay minimization. For this purpose, the values of μn:μp had to be calculated. For this purpose, the βeff values were calculated for a saturated nMOS and pMOS.

Hence, **μn:μp = (βeff)n : (βeff)p = 3.24**

Hence, the value of 3.5 was chosen for the ratio of μn:μp.

W/L Calculations:

Our task was to find the W/L values of the MOSFETs, which would result in a minimum delay. Delay of a counter can be minimized if the delay of respective Flip-flop is minimized.

Hence, a parametric sweep was conducted under the transient mode of analysis by varying Wn = w, and Wp = 3.5*w.

For these operations, a clock of period = 1.6 ns was chosen.
Hence, Clock Frequency = 1/Clock Period
= **625 MHz**

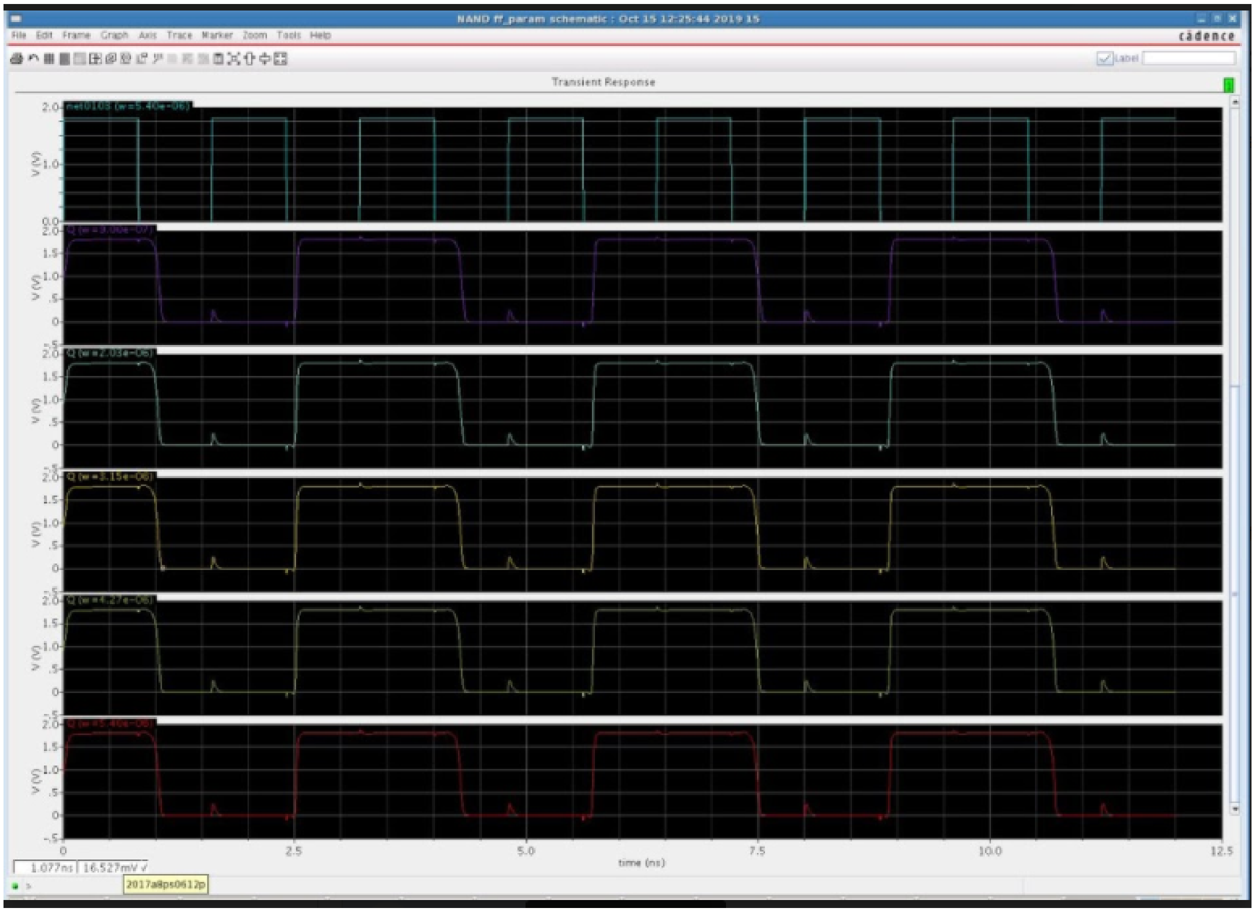


Figure 5. Transient analysis of D flip-flop output with sweeping W from 450 nm to 2250 nm.

W values	Delay(H->L)	Delay(L->H)
450 nm	0.411 ns	0.255 ns
900 nm	0.402 ns	0.253 ns

1350 nm	0.373 ns	0.251 ns
1800 nm	0.371 ns	0.251 ns
2250 nm	0.371 ns	0.251 ns

It was seen that after a value of w = 1800n, there was no significant change in the delay. Hence, this minimum size of w was found optimum by us.

Therefore,

Wn = 1.8 µm

Wp = 3.5*1.8 = 6.3 µm

Now, the value of **(W/L)p = (6.3 µm)/(180 nm) = 35** >20.
So, the pMOS was made in the layout using Multi-Fingering, by the usage of 2 fingers, so as to reduce the parasitics.

SCHEMATIC OF 3 BIT ASYNCHRONOUS COUNTER:

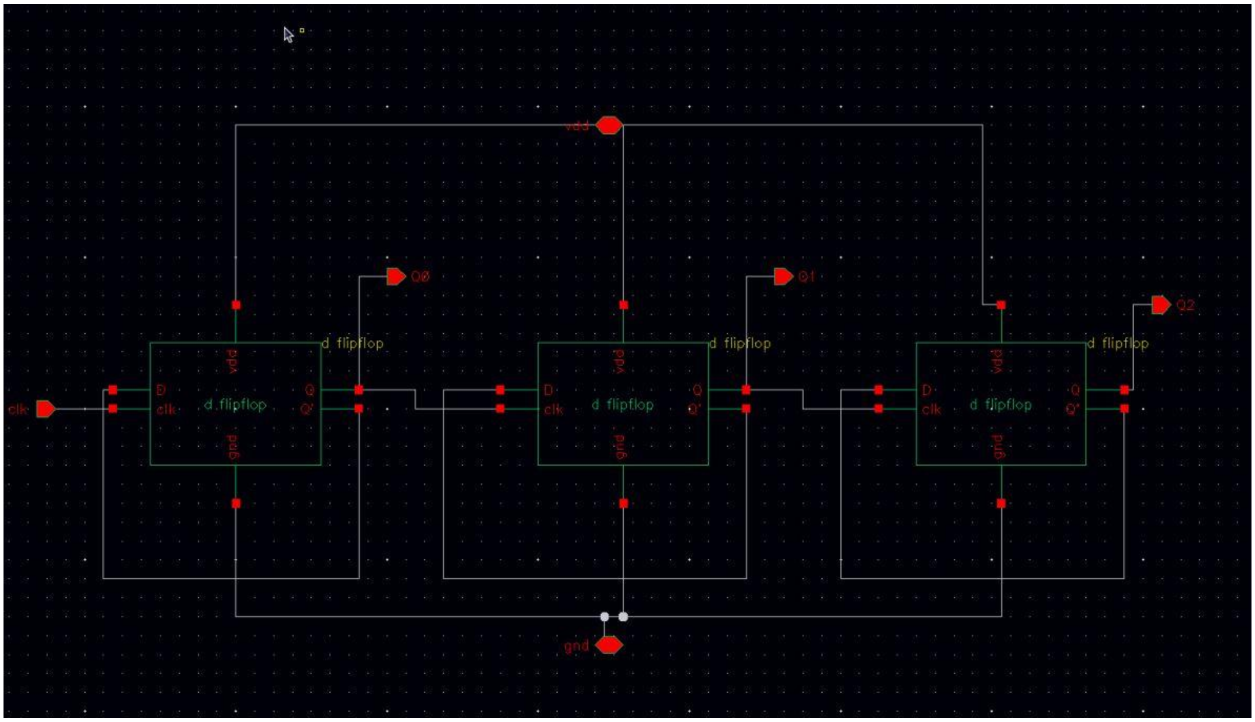
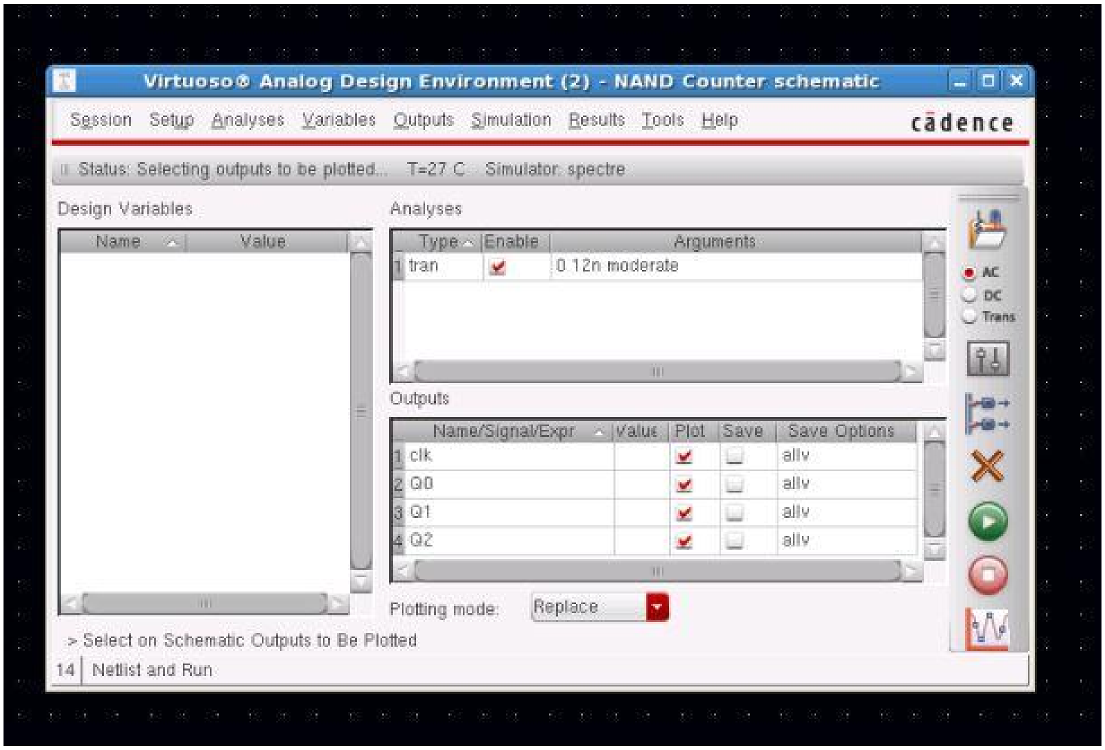


Figure 6. Schematic of 3 bit asynchronous counter

TRANSIENT ANALYSIS OF COUNTER FROM SCHEMATIC:

Transient analysis was performed on the designed schematic of 3 bit asynchronous counter from ADE L tool in Cadence. The output of counter Q0 Q1 Q2 was plotted along with CLOCK input of 625 MHz. The frequency divide by 2 operation was observed and a **3 bit up counter** was realized with **(MSB) Q2 Q1 Q0 (LSB).**



Virtuoso® Analog Design Environment (2) - NAND Counter schematic

Session Setup Analyses Variables Outputs Simulation Results Tools Help
cādence

Status: Selecting outputs to be plotted... T=27 C Simulator: spectre

Design Variables

Name	Value

Analyses

	Type	Enable	Arguments
1	tran	✓	0 12n moderate

Outputs

	Name/Signal/Expr	Value	Plot	Save	Save Options
1	clk		✓	□	allv
2	Q0		✓	□	allv
3	Q1		✓	□	allv
4	Q2		✓	□	allv

Plotting mode: Replace

> Select on Schematic Outputs to Be Plotted

14 Netlist and Run

Figure 7. Running transient analysis for designed counter schematic

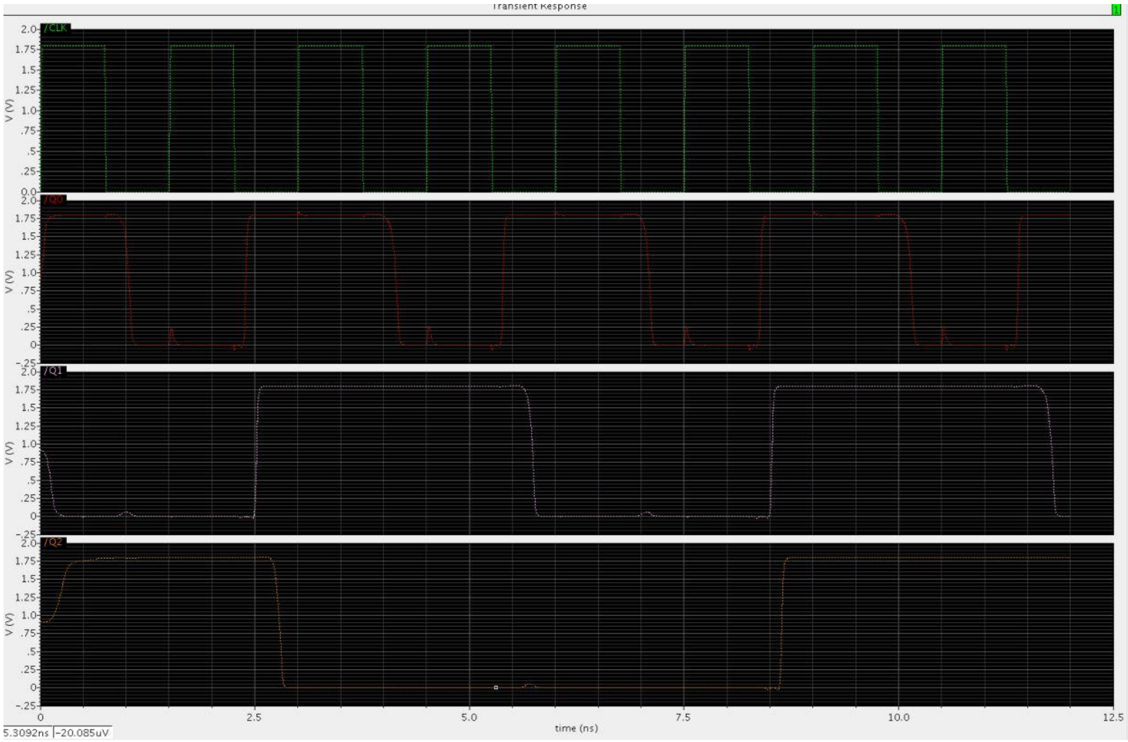


Figure 8. Results of transient analysis of counter schematic

LAYOUT OF D FLIPFLOP:

From the schematic of counter, we designed full custom layout following all the design rules. As mentioned earlier, the pMOS were designed using 2 Fingers, so as to minimize the parasitic capacitance (as the sidewall as well as area Junction Capacitances are reduced).

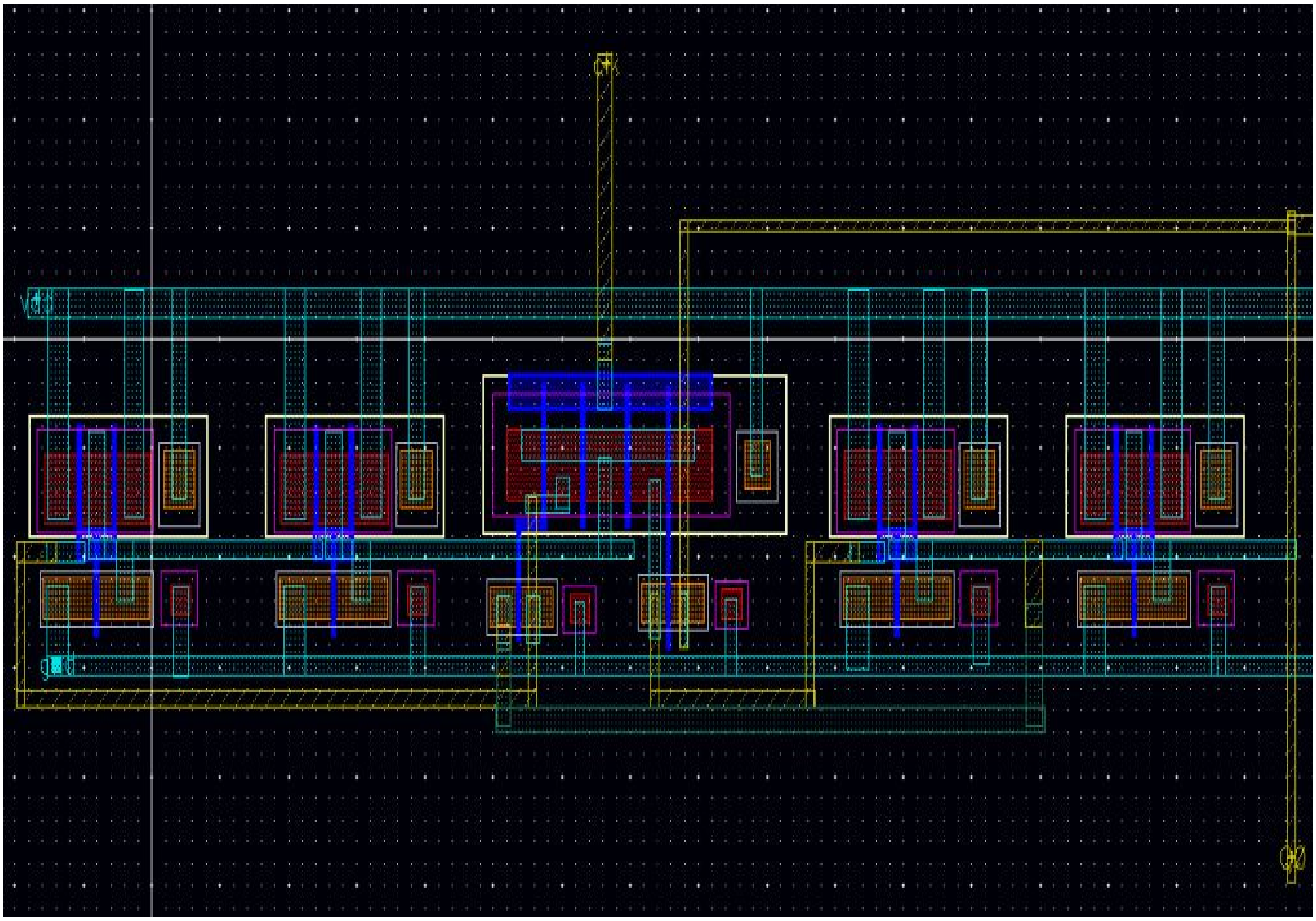


Figure 9. Layout of D flip-flop

LAYOUT OF 3 BIT ASYNCHRONOUS COUNTER:

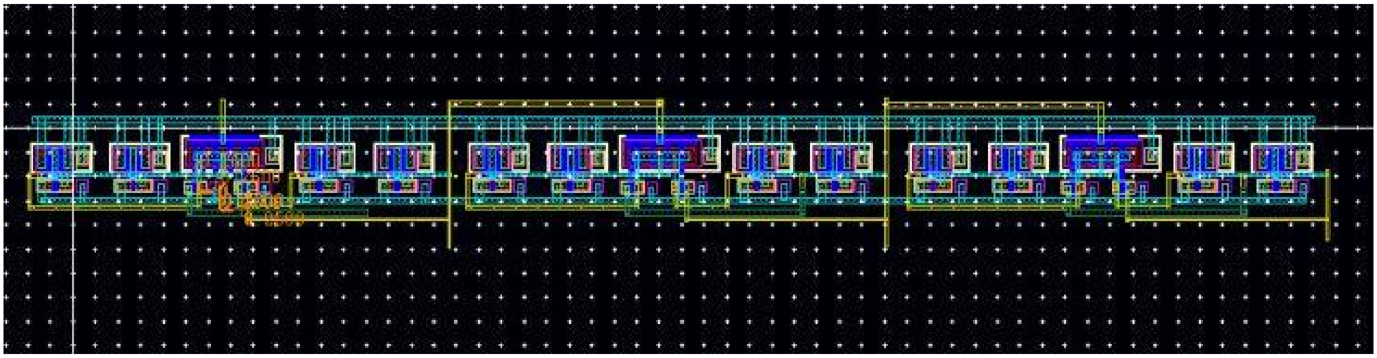
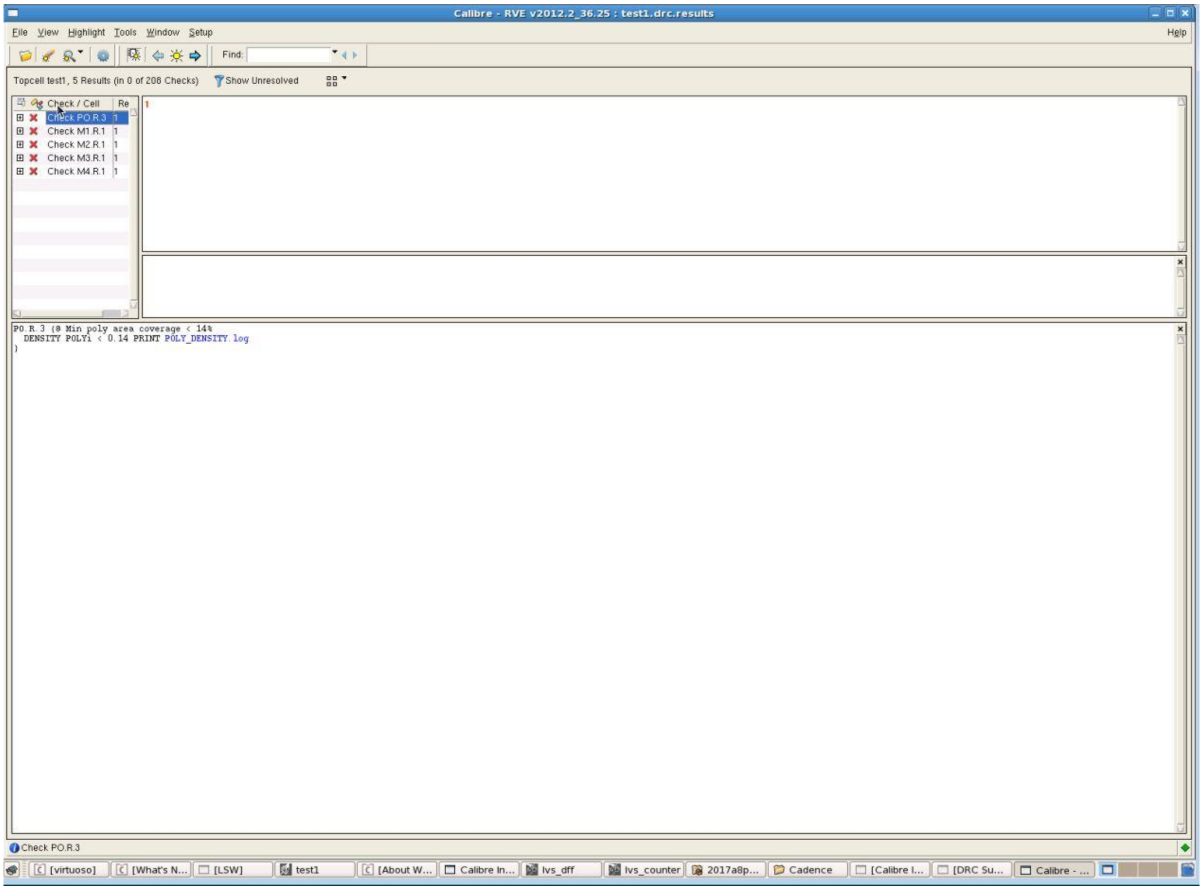


Figure 9. Layout of 3 bit counter

The layout passed DRC (Design Rule Check) and LVS (Layout Versus Schematic) results of which are shown below.



Calibre - RVE v2012.2_36.25 : test1.drc.results
File View Highlight Tools Window Setup Help
Find:
Show Unresolved

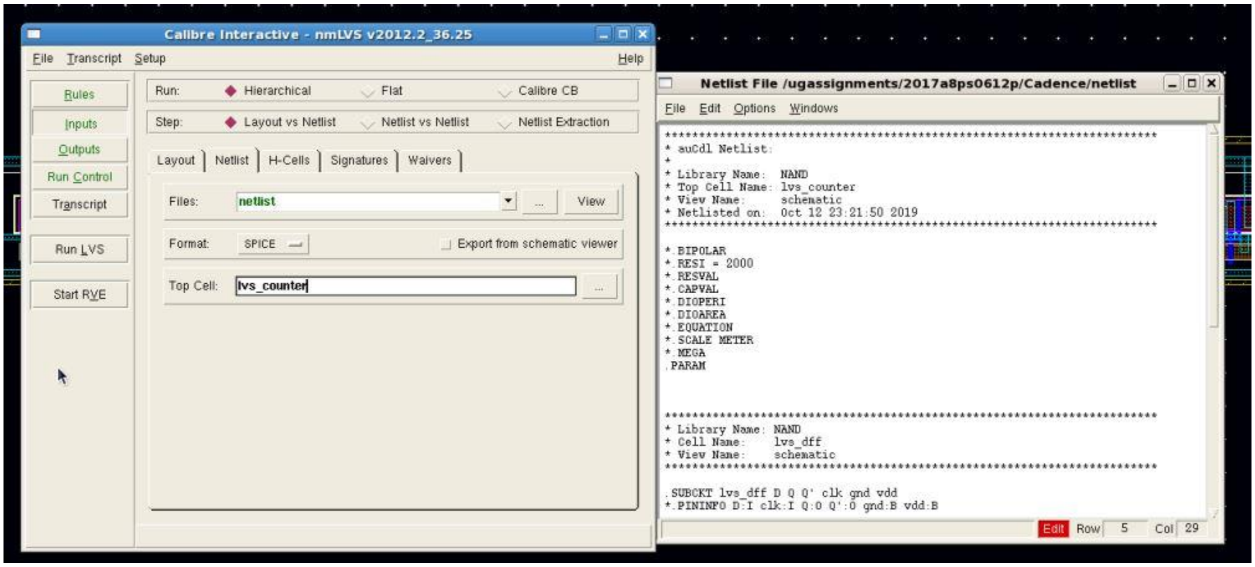
Topcell test1, 5 Results (in 0 of 208 Checks)

Check / Cell	Re
Check PO.R.3	1
Check M1.R.1	1
Check M2.R.1	1
Check M3.R.1	1
Check M4.R.1	1

```
PO.R.3 {@ Min poly area coverage < 14%
  DENSITY POLYi < 0.14 PRINT POLY_DENSITY.log
}
```

Check PO.R.3

Figure 10. Results of DRC (showing only unresolved density errors)



Calibre Interactive - nmLVS v2012.2_36.25
File Transcript Setup Help
Rules
Inputs
Outputs
Run Control
Transcript
Run LVS
Start RVE
Layout
Netlist
H-Cells
Signatures
Waivers
View
Export from schematic viewer
Netlist File /ugassignments/2017a8ps0612p/Cadence/netlist
File Edit Options Windows

Run: ◆ Hierarchical ◇ Flat ◇ Calibre CB

Step: ◆ Layout vs Netlist ◇ Netlist vs Netlist ◇ Netlist Extraction

Files: netlist

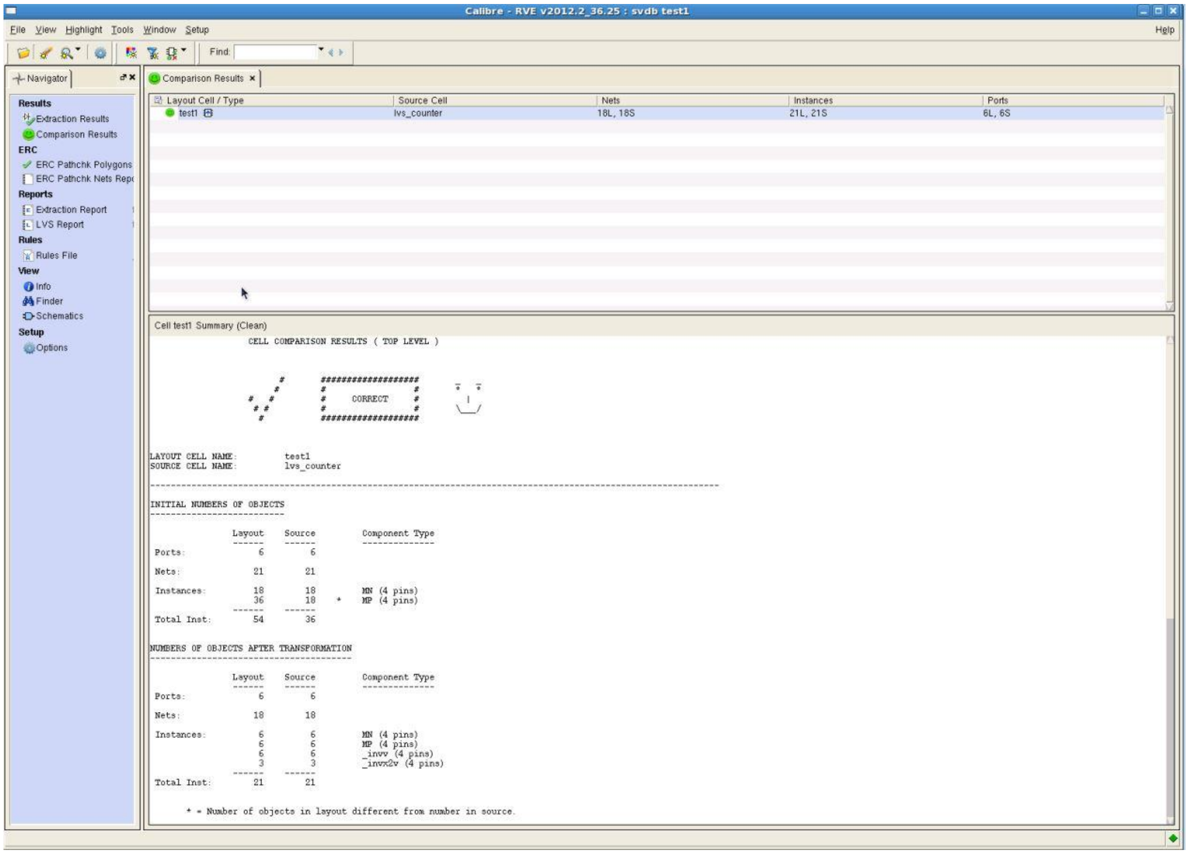
Format: SPICE

Top Cell: lvs_counter

```
************************************************************************
* auCdl Netlist:
*
* Library Name:  NAND
* Top Cell Name: lvs_counter
* View Name:     schematic
* Netlisted on:  Oct 12 23:21:50 2019
************************************************************************

*.BIPOLAR
*.RESI = 2000
*.RESVAL
*.CAPVAL
*.DIOPERI
*.DIOAREA
*.EQUATION
*.SCALE METER
*.MEGA
.PARAM



************************************************************************
* Library Name: NAND
* Cell Name:    lvs_dff
* View Name:    schematic
************************************************************************

.SUBCKT lvs_dff D Q Q' clk gnd vdd
*.PININFO D:I clk:I Q:O Q':O gnd:B vdd:B
```

Edit Row 5 Col 29

Figure 11. Running LVS



Layout Cell / Type	Source Cell	Nets	Instances	Ports
● test1	lvs_counter	18L, 18S	21L, 21S	6L, 6S

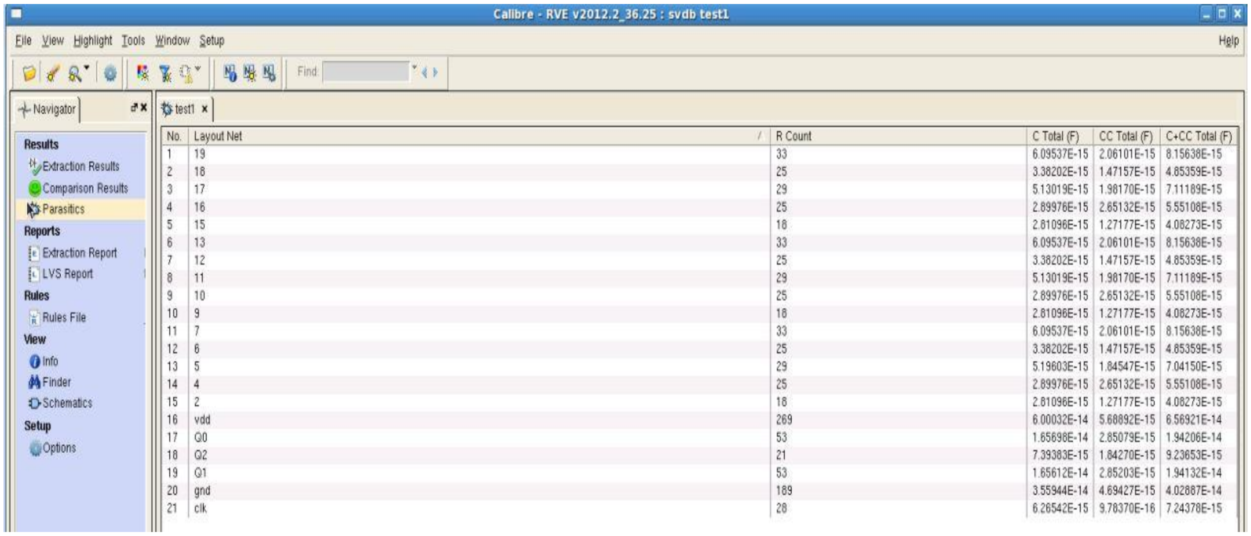
Cell test1 Summary (Clean)

```
                  CELL COMPARISON RESULTS ( TOP LEVEL )


                     #       ###################      _   _
                    #        #                 #       *   *
                #  #         #     CORRECT     #         |
                 # #         #                 #       \___/
                  #          ###################


LAYOUT CELL NAME:        test1
SOURCE CELL NAME:        lvs_counter

--------------------------------------------------------------------------------

INITIAL NUMBERS OF OBJECTS
--------------------------

                Layout    Source         Component Type
                ------    ------         --------------
 Ports:              6         6

 Nets:              21        21

 Instances:         18        18         MN (4 pins)
                    36        18    *    MP (4 pins)
                ------    ------
 Total Inst:        54        36

NUMBERS OF OBJECTS AFTER TRANSFORMATION
---------------------------------------

                Layout    Source         Component Type
                ------    ------         --------------
 Ports:              6         6

 Nets:              18        18

 Instances:          6         6         MN (4 pins)
                     6         6         MP (4 pins)
                     6         6         _invv (4 pins)
                     3         3         _invx2v (4 pins)
                ------    ------
 Total Inst:        21        21

      * = Number of objects in layout different from number in source.
```

Figure 12. Passed LVS

PEX ANALYSIS:

PEX (Parasitic Extraction) was performed on designed layout to get the parasitic capacitance value at each layout net. Following figure shows the result of PEX analysis. All the capacitance are in the range of **4 fF to 10 fF**.



No.	Layout Net	R Count	C Total (F)	CC Total (F)	C+CC Total (F)
1	19	33	6.09537E-15	2.06101E-15	8.15638E-15
2	18	25	3.38202E-15	1.47157E-15	4.85359E-15
3	17	29	5.13019E-15	1.98170E-15	7.11189E-15
4	16	25	2.89976E-15	2.65132E-15	5.55108E-15
5	15	18	2.81096E-15	1.27177E-15	4.08273E-15
6	13	33	6.09537E-15	2.06101E-15	8.15638E-15
7	12	25	3.38202E-15	1.47157E-15	4.85359E-15
8	11	29	5.13019E-15	1.98170E-15	7.11189E-15
9	10	25	2.89976E-15	2.65132E-15	5.55108E-15
10	9	18	2.81096E-15	1.27177E-15	4.08273E-15
11	7	33	6.09537E-15	2.06101E-15	8.15638E-15
12	6	25	3.38202E-15	1.47157E-15	4.85359E-15
13	5	29	5.19603E-15	1.84547E-15	7.04150E-15
14	4	25	2.89976E-15	2.65132E-15	5.55108E-15
15	2	18	2.81096E-15	1.27177E-15	4.08273E-15
16	vdd	269	6.00032E-14	5.68892E-15	6.56921E-14
17	Q0	53	1.65698E-14	2.85079E-15	1.94206E-14
18	Q2	21	7.39383E-15	1.84270E-15	9.23653E-15
19	Q1	53	1.65612E-14	2.85203E-15	1.94132E-14
20	gnd	189	3.55944E-14	4.69427E-15	4.02887E-14
21	clk	28	6.26542E-15	9.78370E-16	7.24378E-15

Figure 13. Results of PEX analysis

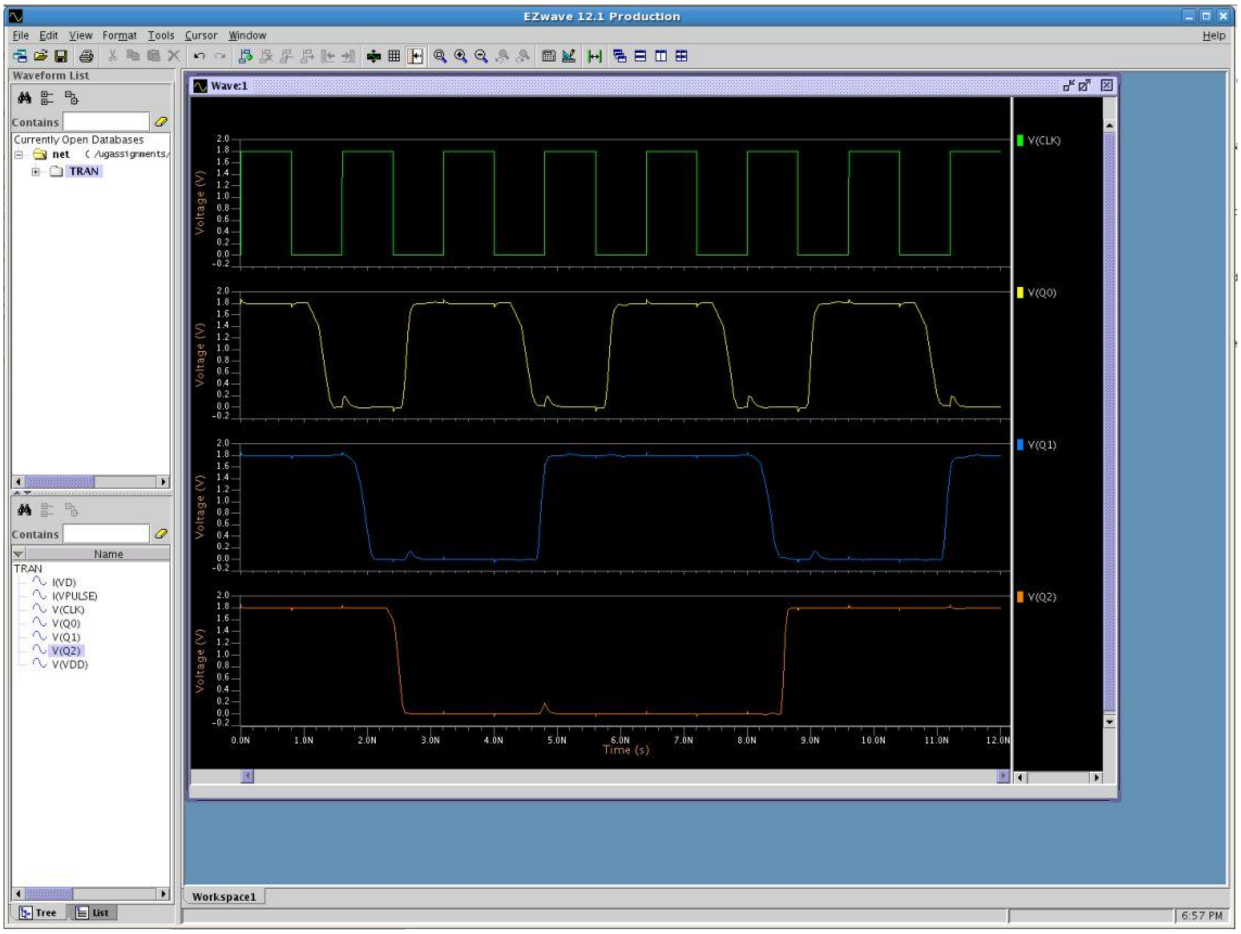


Figure 14. Simulating layout on Eldo tool

To find the delay introduced by parasitics in the layout, a netlist (.cir) was extracted and additional lines were added in the .cir file to stimulate the circuit with clock input of 625 MHz frequency or 1.6 ns time period. This analysis was performed using **Eldo tool.**

Above figure shows the timing diagram of clock input along with counter output Q0 Q1 Q2. The Q0 output is delayed **0.4 ns** with respect to clock, Q1 is delayed **0.6 ns** with respect to Q0 and Q2 is delayed **0.6 ns** with respect to Q1.

SEQUENCE DETECTOR

A mealy machine is designed to detect the sequence ‘1110’, we provide the input sequence and output ‘1’ is generated by the machine every time the above sequence is detected else the output is ‘0’. It detects the overlapping sequence as well, and output is kept synchronised with the clock.

STATE DIAGRAM

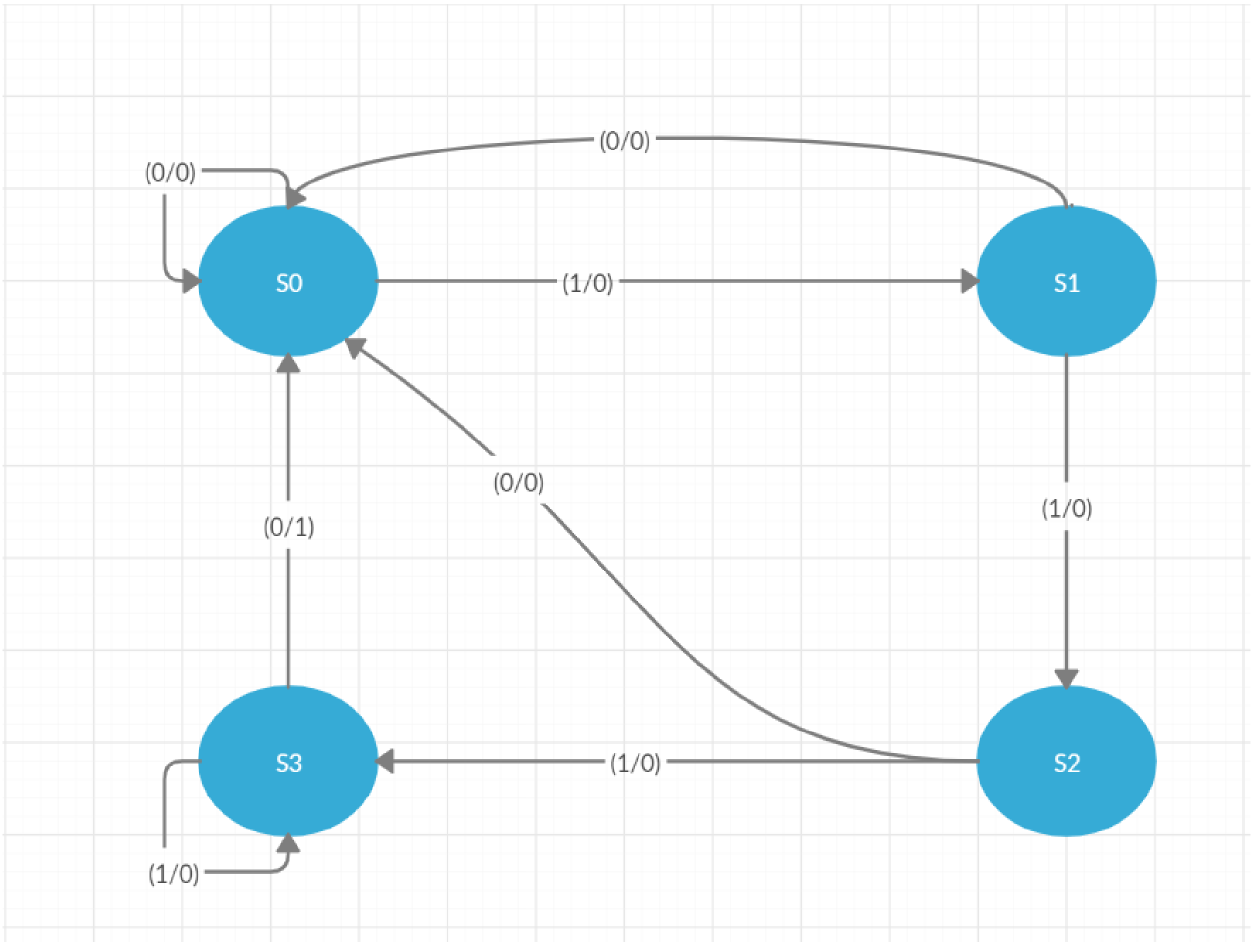


Figure 15. State diagram of sequence detector

VERILOG CODE

```
module seq_det (in, rst, clk, out);

input in, clk, rst;
output reg out;
reg[1:0] current_state, next_state;
reg z;

parameter s0 = 2'b00,
          s1 = 2'b01,
          s2 = 2'b10,
          s3 = 2'b11;

always @(current_state or in)
    begin
    case(current_state)
        s0: if(in == 1'b1)
                begin
                    next_state = s1;
                    z = 1'b0;
                end
            else
                begin
                    next_state = s0;
                    z = 1'b0;
                end

        s1: if(in == 1'b1)
                begin
                    next_state = s2;
                    z = 1'b0;
                end
            else
                begin
                    next_state = s0;
                    z = 1'b0;
                end

        s2: if(in == 1'b1)
                begin
                    next_state = s3;
                    z = 1'b0;
                end
            else
                begin
                    next_state = s0;
                    z = 1'b0;
                end

        s3: if(in == 1'b1)
                begin
                    next_state = s3;
                    z = 1'b0;
                end
            else
                begin
                    next_state = s0;
                    z = 1'b1;
                end

//      default: s0;
    endcase
    end
```

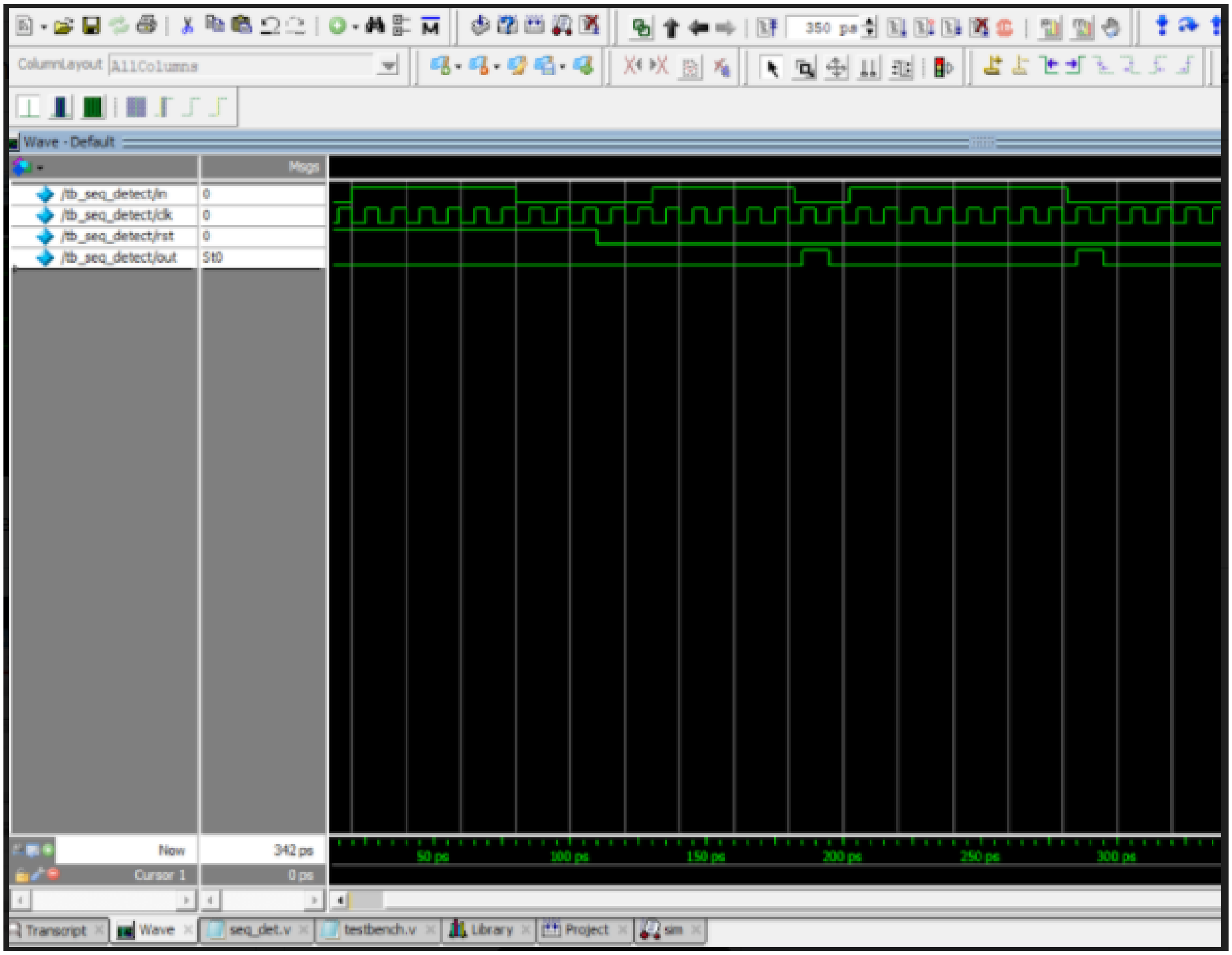
```
always @(posedge clk)
        begin
                if(rst)
                        begin
                        current_state = s0;
                        out = 0;
                        end
                else
                        begin
                        current_state <= next_state;
                        out <= z;
                        end
        end
endmodule
```

TEST BENCH

```
module tb_seq_detect;

reg in,clk,rst;
wire out;

seq_det u1(

.in(in),
.rst(rst),
.clk(clk),
.out(out)
 );

 always #5 clk = ~clk;
```

```
initial begin
in = 0;
clk = 0;
rst = 1;

#20 in = 1;
#60 in = 0;
#20 in = 0;
#10 rst = 0;
#20 in = 1;
#52 in = 0;
#20 in = 1;
#80 in = 0;

#60

$finish ;
end

endmodule
```

OUTPUT WAVEFORMS



	Msgs
/tb_seq_detect/in	0
/tb_seq_detect/clk	0
/tb_seq_detect/rst	0
/tb_seq_detect/out	St0
Now	342 ps
Cursor 1	0 ps

Figure 16. Timing diagram of sequence detector circuit simulated on ModelSim (1)

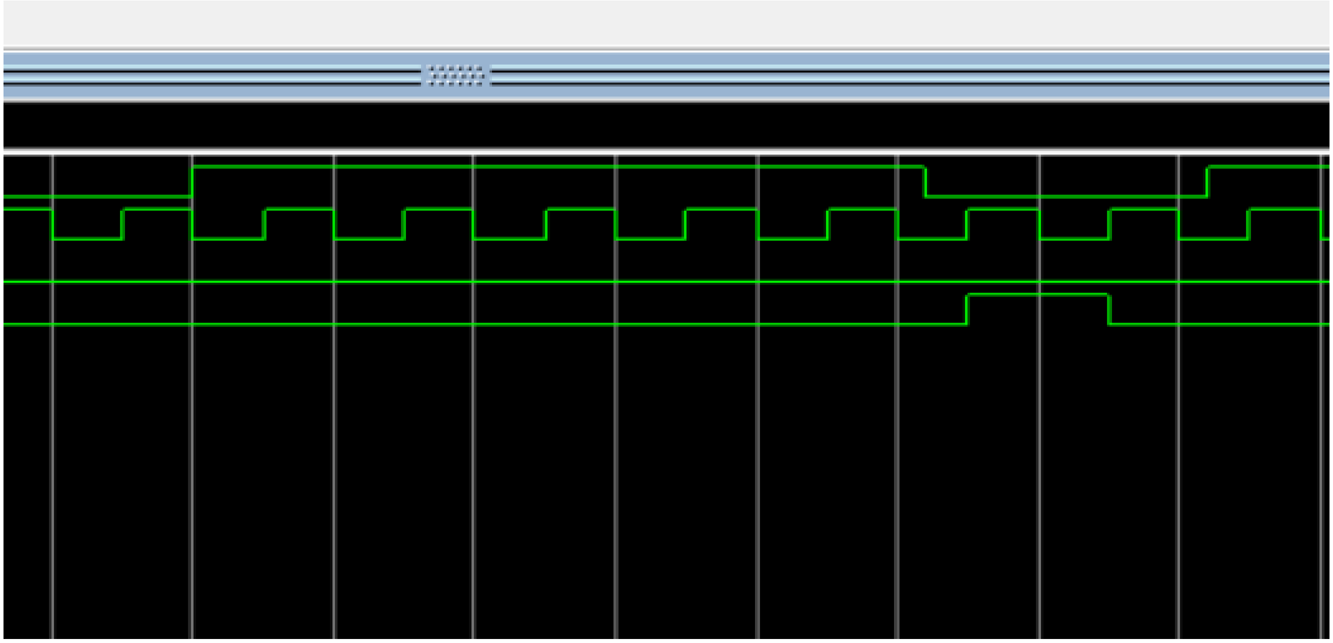


Figure 17. Timing diagram of sequence detector circuit simulated on ModelSim (2)

CONCLUSIONS:

Through this assignment, we were able to:

- Get a better understanding of digital design using a Hardware Description Language like Verilog
- Learn the art of simulating Test Benches
- Learn the usage of Cadence Virtuoso and Eldo tool
- Understand the advantages and disadvantages of various logic families

REFERENCES:

- “Digital Integrated Circuits” by Jan M. Rabaey
- M. Janaki Rani, S. Malarkann - “Leakage Power Reduction and Analysis of CMOS Sequential Circuits.”
- “Verilog HDL” by Samir Palnitkar