PART A

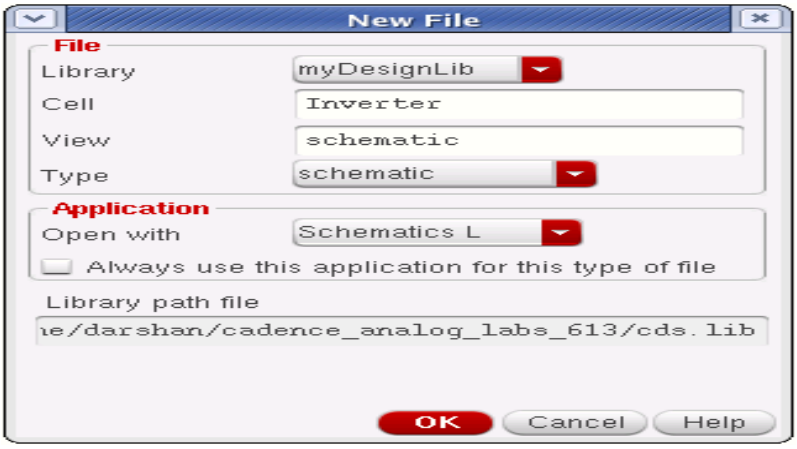
Steps to Execute Analog Design

1. **Right Click on the Desktop and Create a new folder(ex-USN)**
2. **Right click ->Open in Terminal and type the following commands**

   csh -> source /home/cad/cshrc

   (Welcome to cadence tool suite)

   Virtuoso &
3. **Virtuoso 6.1.6 window opens and follow the steps below to create library**
   - **Tools ->Library Manager -> File ->new -> Library**
   - **In the new library window give a new library name (ex: your name) ->ok**
   - **In the Technology file for new library window select "Attach to an existing technology library" ->ok ->select gpdk180->ok**
4. **In Library manager work area window select your created library, File -> new ->cellview. In new File window type cell name (ex:inv) {Make sure that library is your library name} and click ok -> yes**



a) A blank Schematic Editor window opens. In this window schematic has to be drawn using following steps

Adding Components to schematic

a) Create→ Instance or press i from keyboard.

b) Click on browse button. Browse to gpdk180 library and select the required components.(ex: PMOS and NMOS) Modify the specifications as required.

c) After you complete the Add Instance form, move your cursor to schematic window and left click to place a component.

d) After entering the components, click cancel in the Add Instance or press ESC.

Adding PINS to Schematic

a) Create→ pin or press "p"

b) The add pin form appears.

c) Type all the required pins along with input/output and place it on schematic window.

Adding Wires to Schematic

a) Click the Wire(narrow) icon in the schematic window.

b) Connect all the components using the wires. click on check and save icon in the schematic editor window. Check for any errors in CIW window.

Symbol Creation

a) Create→cellview→ From Cellview

b) In Cellviewform , verify that From View Name field is to schematic, and TO View Name field is set to symbol, with the Tool/Data Type set as Schematic Symbol.

c) Click ok in the CellView form and Symbol generation form appears.

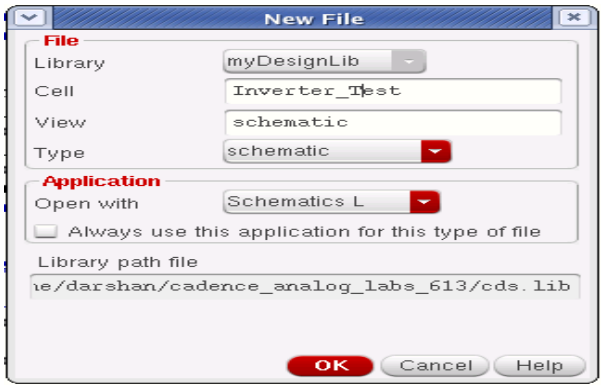
d) Modify the Pin Specifications. For ex: Put pins in appropriate position ie., left/right/top/bottom.

e) Click ok and a new window appears with automatically created symbol.

Building the test design

a) **In CIW, file→new→Cellview**

b) **Set up New file form appears as follows:**



In the above form for library select your library created by you.

c) **Click ok; a blank schematic window for test design appears.**

Building the Test circuit

a) **Draw the design test circuits according to the specification by adding symbol from your library and Vpulse, Vdc , gnd from analog library.**

b) **Once design is completed click check and save icon.**

To simulate the test design

a) **Launch→ADE L**

b) **Virtuoso ADE(Analog Design Environment) simulating window appears**

Choosing Analyses

Transient Analysis:

a) **Select Analyses → choose**

b) **To setup the transient analysis, select tran.**

c) **Click at the moderate button and press apply.**

d) **Enter all the required specification and press ok.**

DC analysis:

a) **Select Analyses→choose**

b) **To setup dc analysis, select dc.**

c) **In the DC analysis window, select save DC operating point.**

d) **Turn on the Component Parameter.**

e) **Click on the select component, which takes you to test schematic window.**

f) **Select input signal ex: vpulse source in inverter_test.**

g) **Enter all the required specification and press ok.**

AC analysis:

a) **Select Analyses→choose**
b) **To setup ac analysis, select ac**
c) **Enter all the required specification and press ok.**

Selecting outputs for plotting

a) **Outputs→To be plotted→Select on Schematic or click on setup output icon.**
b) **In the Select the required inputs and outputs from schematic.**

Running the simulation

a) **Simulation→Netlist and Run or select Netlist and Run icon.**
b) **When simulation finishes, the Transint, DC,and AC plots automatically will be popped up along with netlist.**

Layout Design Steps

1. **Open schematic editor window and draw the schematic circuit of the specified experiment. Save the circuit and select check and save icon.**
2. **If no errors in the circuit then select Launch -> Layout XL -> select create new and Automatic in start up option window. In new file window set cell name as eg: inverter, View as Layout and type should be Layout. Then select ok.**
3. **In Layout editor window select connectivity -> Generate -> All from source ok.**
4. **Extend the PR boundary area by using stretch icon.**
5. **Place NMOS and PMOS transistor inside PR boundary and press shift f.**
6. **Select the NMOS/PMOS, Right click and select properties. In properties window select parameter -> Body type -> Integrated/Detachable as per requirement.**
7. **Select place icon -> pin placement and select input parameters one by one eg (Vss, then Vdd, then Vin etc). Select attribute and select edge as a bottom for VSS and top for Vddand then select apply. Later select HR rails. Whereas for other inputs select only the attributes as left and outputs as right. No HR rails are selected for these parameters.**
8. **Select create -> ViaVia define and select -> M1-poly to connect metal to polysilicon for inputs.**
9. **For Vdd and Vss connection select „p‟ from keyboard and select metal1 from the layer window and connect. (When narrow wire is required use „p‟ form keyboard else select create -> wiring -> wire and then select required layer in layer window.**
10. **Design the layout of specified circuit and save the design.**
11. **For Simulation**
    a) **Assura → Technology→home/cad/Foundary/Analog/180n/Assura_tech.lib**
    b) **Select Assura ->Run DRC -> type file name as abc and select technology as gpdk180->ok.**

c) **If no errors then select Assura->Run LVS->type file name abcd and select technology as gpdk180 -> ok.**

d) **If LVS matches then select Assura -> Run QRC-> select technology as gpdk180 and output as extract view -> Extraction type as RC -> Select reference node as Vss -> ok.**

e) **Select file -> open -> select cell -> ok (eg: cell:inverters, view: AV_extracted ok)**

f) **Open RC extracted file select launch -> Adel -> setup -> stimulus -> Provide all values and select ok**

g) **Analysis -> Choose -> Tran (fill values either 200n or 5m)**

h) **Select inputs and output from schematic and run the circuit to get transfer characteristics.**

1. CMOS INVERTER

Aim: Capture the Schematic of a CMOS Inverter with Load Capacitance of 0.1 pF and set the Widths of Inverter with
(i) WN = WP
(ii) WN = 2 WP
(iii) WN = WP / 2
and Length at selected Technology. Carry out the following:
1. Set the Input Signal to a pulse with Rise Time, Fall Time of 1 ps and Pulse Width of 10 ns, Time Period of 20 ns and plot the input voltage and output voltage of the designed Inverter
2. From the Simulation Results, compute tpHL, tpLH and tPD for all the three geometrical settings of Width
3. Tabulate the results of delay and find the best geometry for minimum delay for CMOS Inverter

(i) WN = WP

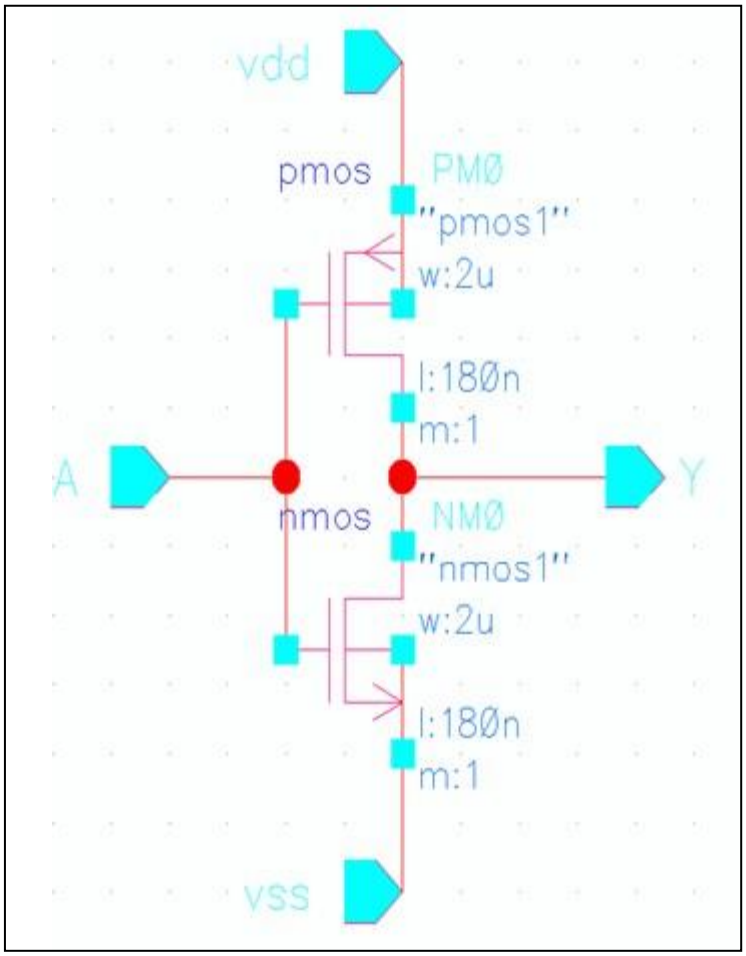


Fig: Schematic Diagram

(ii) Wn=2Wp:

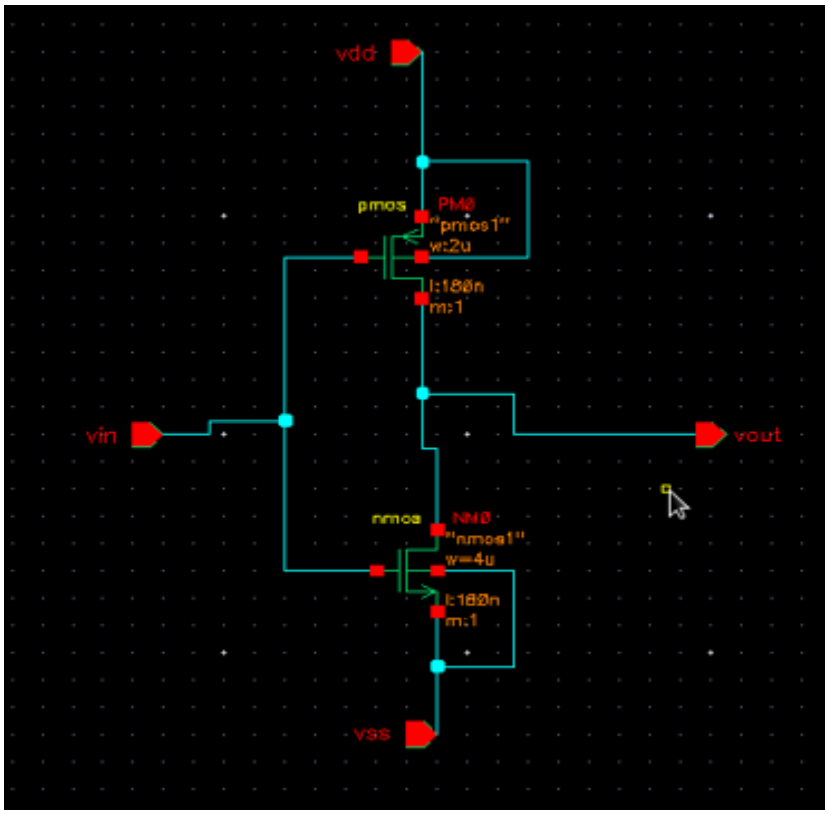


Fig: Schematic Diagram

(ii) Repeat the schematic, symbol, and test circuit for case Wn=Wp/2, by choosing Wp=2um

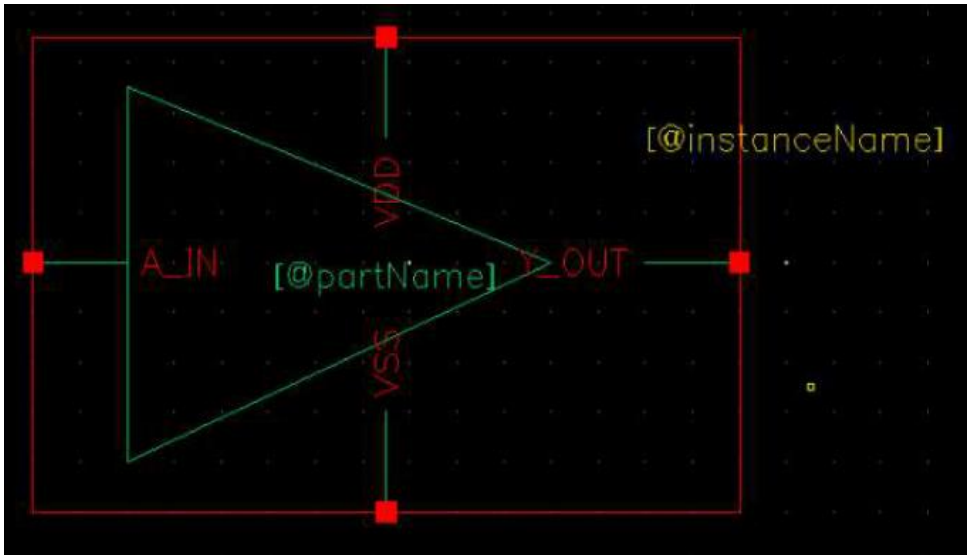


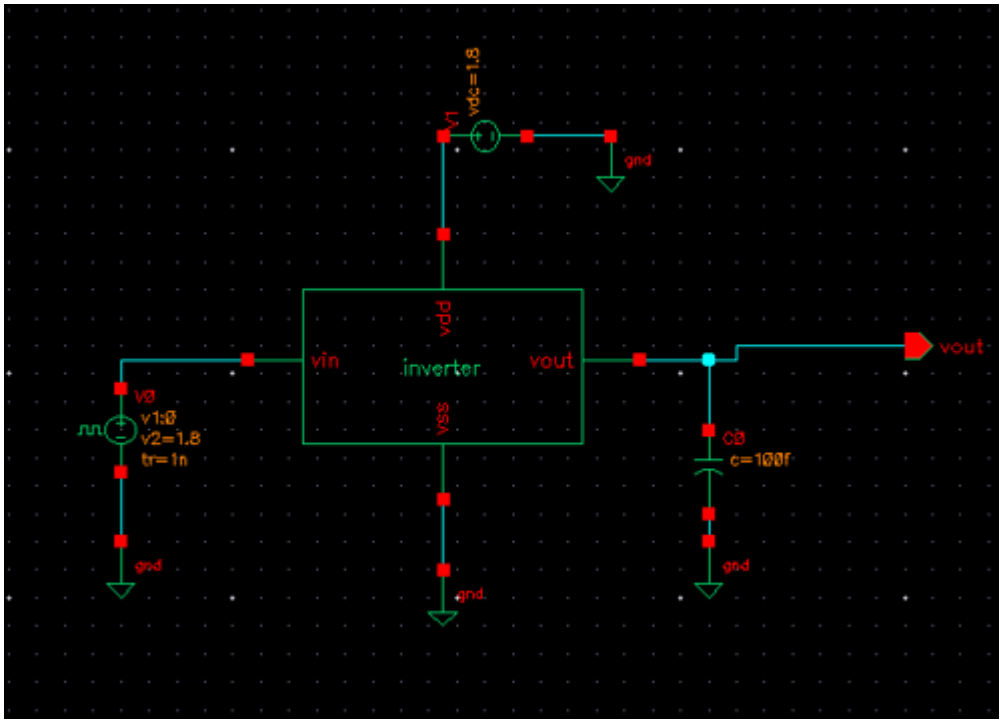
Fig: Inverter Symbol

Specifications: **Vpulse →V1=0**

V2=1.8 V

Delay = 0, Rise time =Fall time = 1n, Pulse width = 10n , Period = 20n

Inverter_test



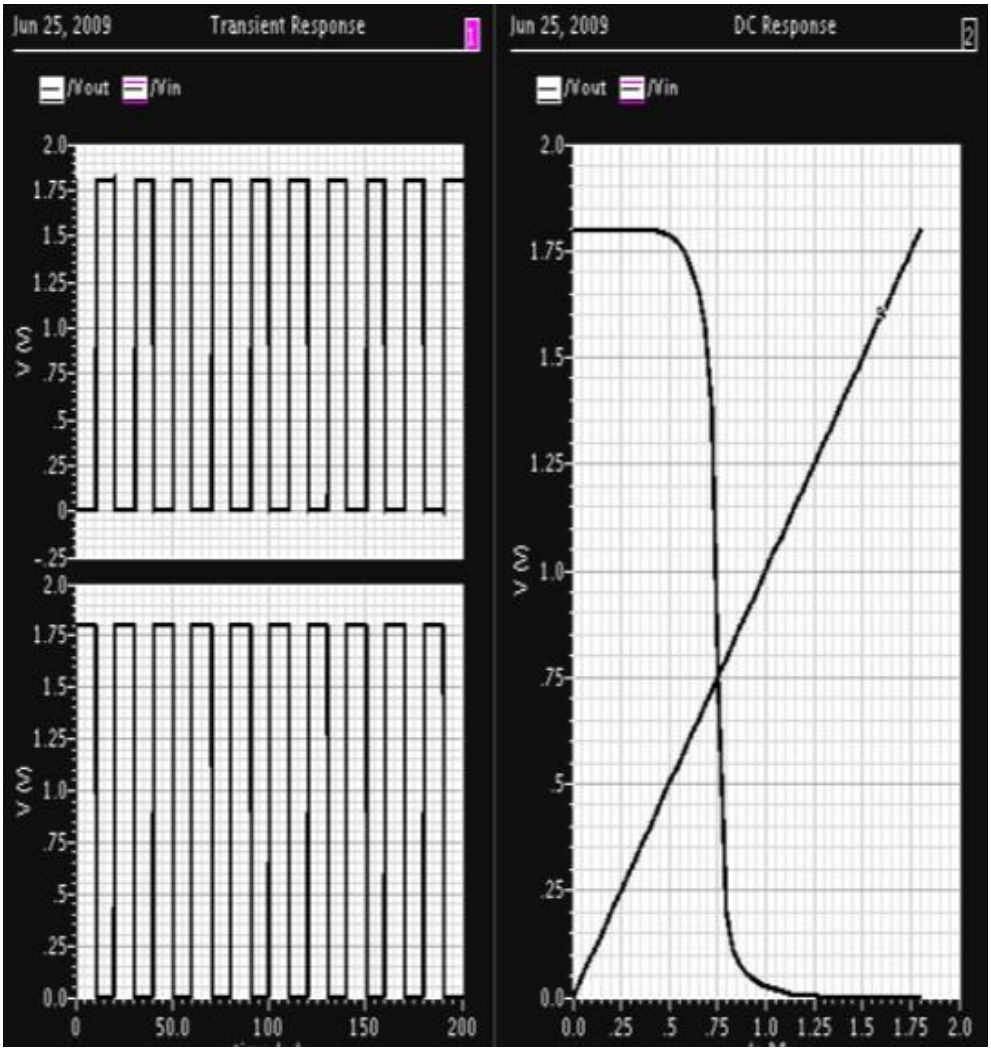
Simulation settings:

set up for transient analysis: Stop time = 200n

Setup for DC analysis:

1. **Component to be selected in schematic is _ for DC analysis**
2. **Start = 0 , stop = 1.8**

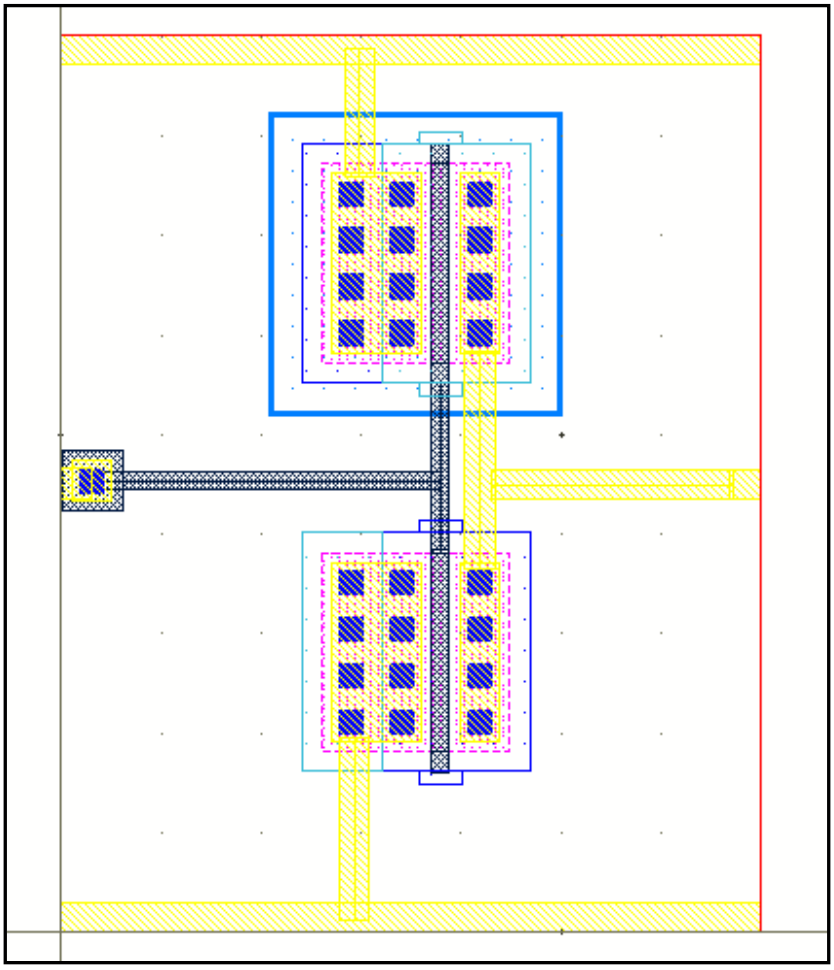
Inverter output Waveforms



Calculation of tphl , tplh and td:

- Click on the calculator in waveform window (go to tools click on calculator)
- In calculator go to file reset GUI
- In the waveform window right click on input signal, send to calculator, in calculator window copy the function of input signal and paste it in function panel using delay function. Repeat the same for output signal.
- Initialize the threshold of (**VDC/2= 0.9**)
- Edge no.1= 1 for signal1 and edge no. 1=2 for signal 2
- Edge type = rising for signal 1 and edge type =falling for signal 2

INVERTER LAYOUT



2.CMOS NAND

Aim: Capture the Schematic of 2-input CMOS NAND gate having similar delay as that of CMOS inverter computed in experiment 1. Verify the functionality of NAND gate and find out the delay td for all four possible combinations of input vectors.

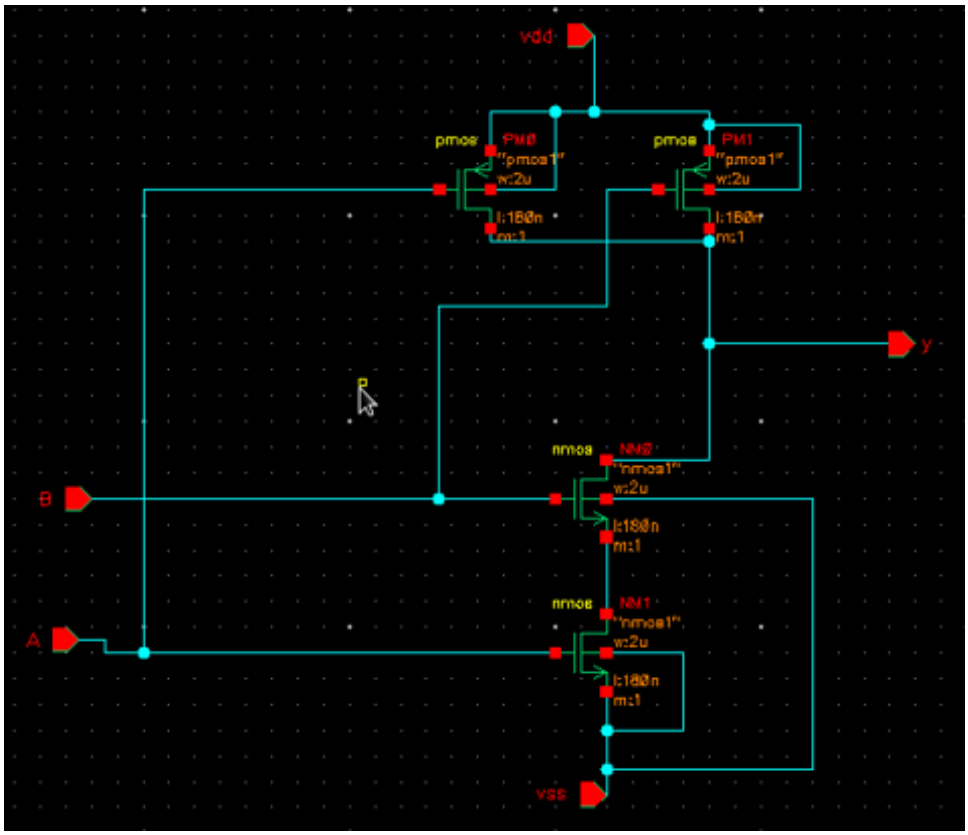


Fig : NAND Schematic

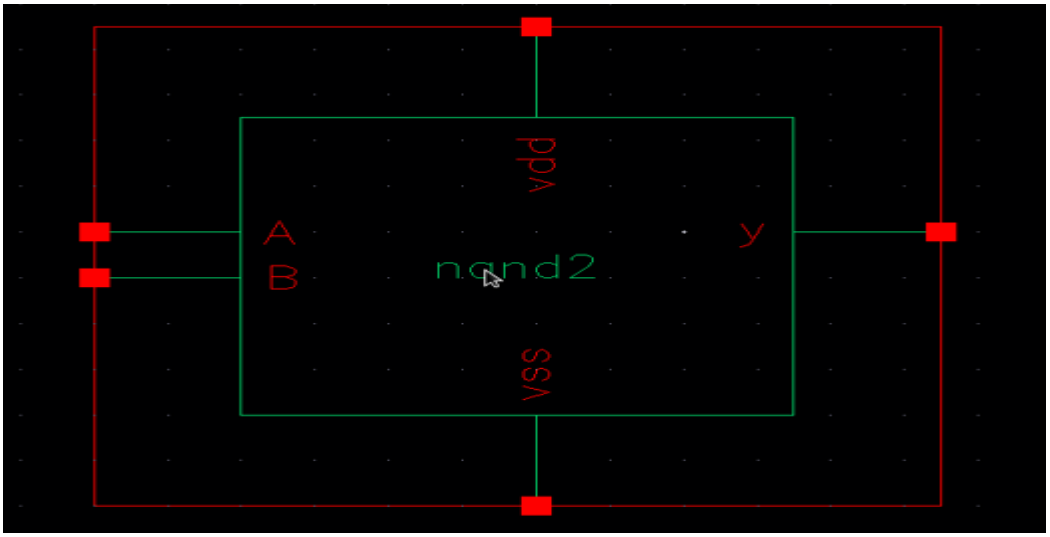


Fig: NAND Symbol

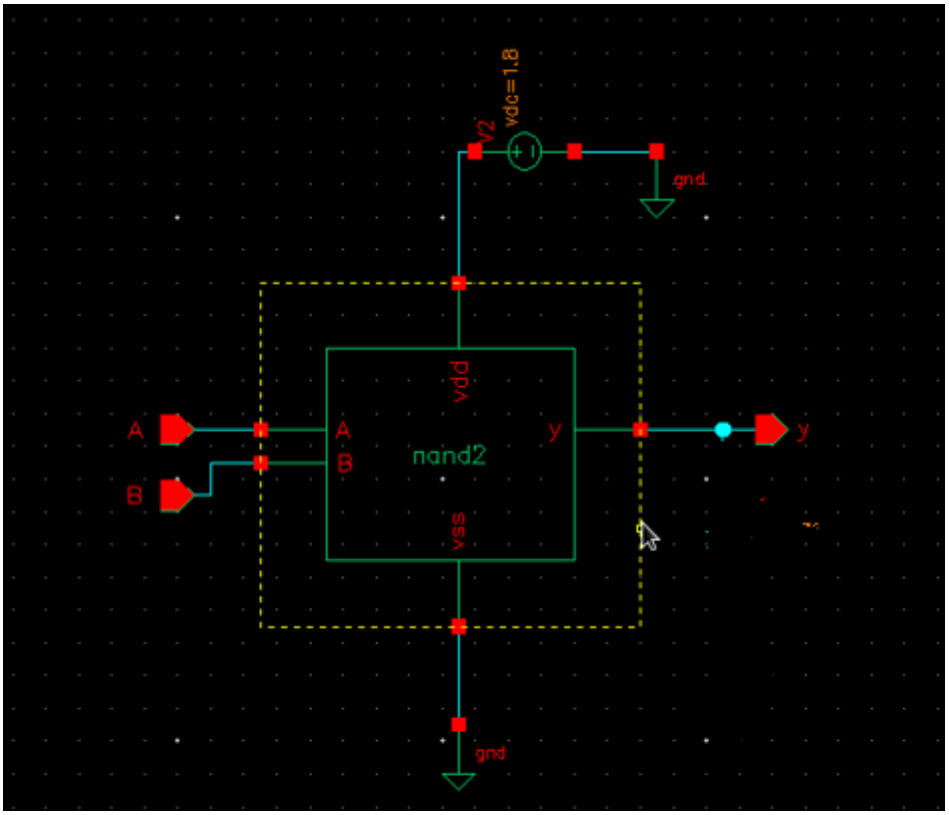
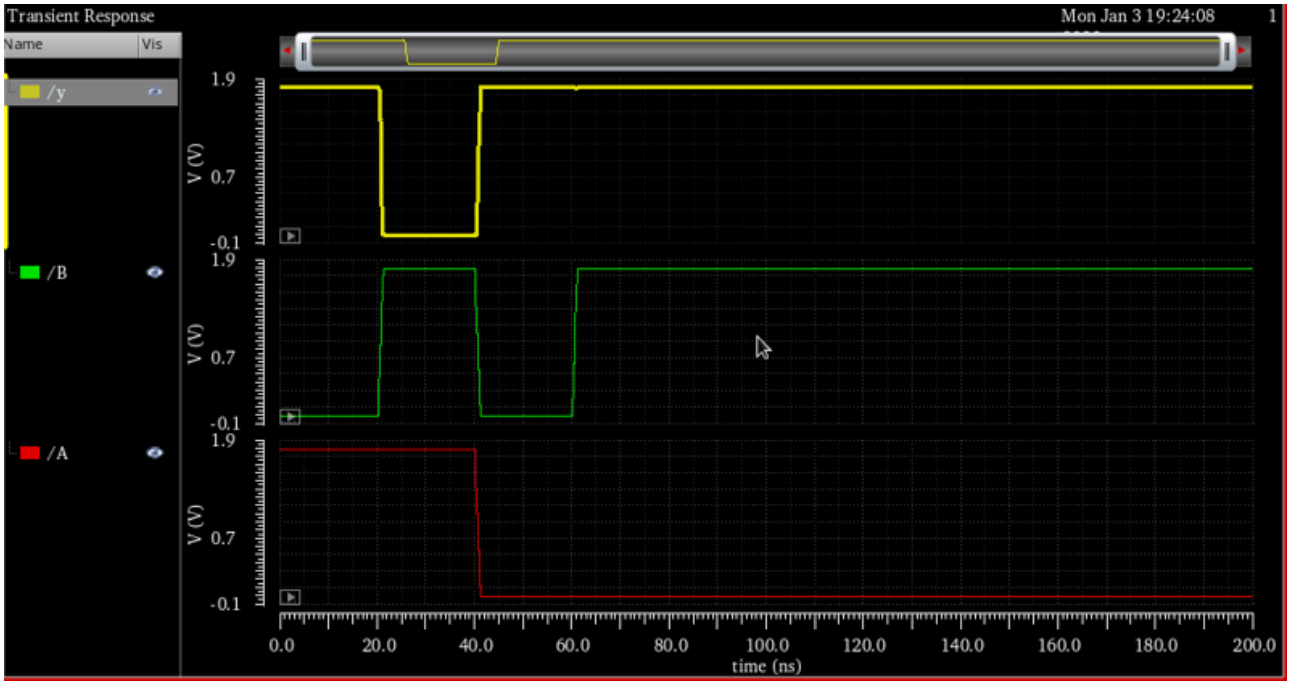


Fig: NAND_test

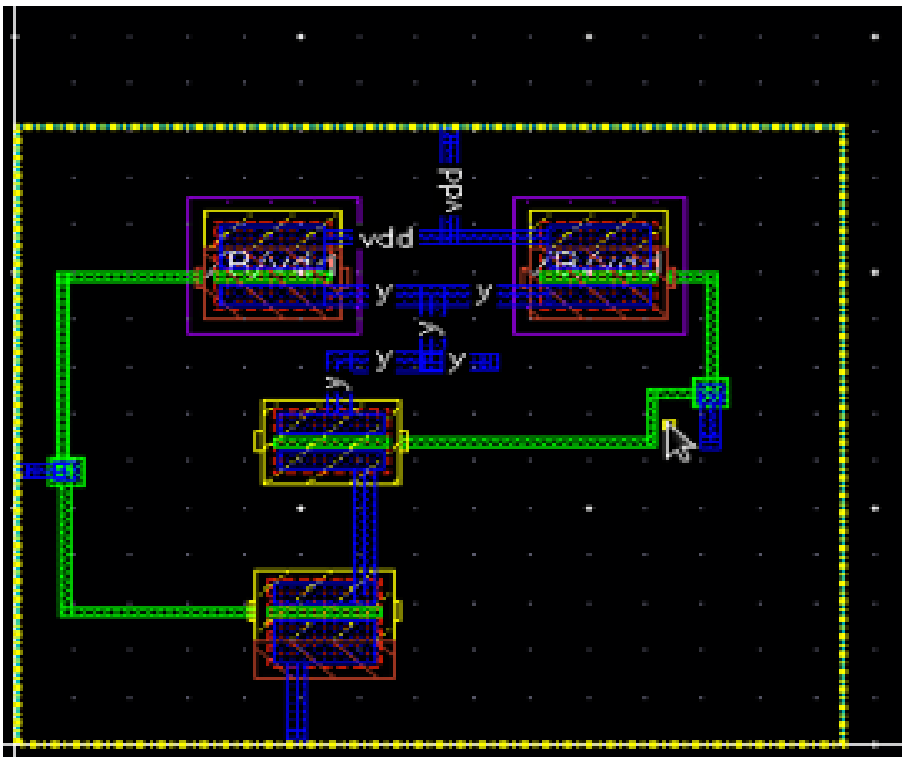
Simulation settings:

set up for transient analysis: Stop time = 200n

Output waveform:



NAND Layout:



STEPS TO EXECUTE DIGITAL DESIGN

(a) Simulation:

1. **Right Click on the Desktop and Create a folder(ex-USN)**
2. **Open the folder and create two files**
   (Right Click -> create document -> Empty file).
3. **Rename the created file by**
   Design_name.v (ex-inv1.v) and Test_name.v(ex-inv2.v)
4. **Copy and paste lib folder and slow.v file inside your folder(USN)**
5. **Open Design_name file, type the verilog program and save it.**
6. **Open Test_name file, type the test bench program and save it and close the folder.**
7. **Right click ->Open in Terminal and type the following commands**
   csh -> source /home/install/cshrc
   (Welcome to cadence tool suite)
   nclaunch –new
   (NC Launch Window opens, Select Multisteps)
   Make sure that
   ->In Design Directory the path should be /root/Desktop/folder name(USN)
   -> In Library Mapping File the path should be /root/Desktop/folder name(USN)/cds.lib
   -> Click on create cds.lib file -> save -> yes -> select don‟ t include any libraries-> ok ->ok
8. **In NC Launch window select both design and test bench file and click on “Launch Verilog Compiler with current selection” icon and check for any syntax errors.**
9. **Expand Worklib, select only test bench and click “launch elaborator with current selection” icon**
10. **Expand Snapshots, select test bench module and click “Launch simulator with current selection” icon (wait for 5 min)**
11. **Sim vision window opens. In Design Browser select test bench-> right click ->send towaveform window -> click run icon. Verify the waveforms.**
12. **Close all simulation windows except terminal window**

(b) Synthesize the design using Constraints and analyze report:

Step 1: Getting Started

- Make sure you close out all the Incisive tool windows first.
- Synthesis requires three files as follows,
- Liberty Files (.lib)
- Verilog/VHDL Files (.v or .vhdl or .vhd)
- SDC (Synopsis Design Constraint) File (.sdc)

Step 2: Creating an SDC File

- In your terminal type "gedit counter_top.sdc" to create an SDC File if you do not have one.
- The SDC File must contain the following commands.

i.create_clock -name clk -period 2 -waveform {0 1} [get_ports "clk"]
ii. set_clock_transition -rise 0.1 [get_clocks "clk"]
iii. set_clock_transition -fall 0.1 [get_clocks "clk"]
iv. set_clock_uncertainty 0.01 [get_ports "clk"]
v. set_input_delay -max 0.8 [get_ports "rst"] -clock [get_clocks "clk"]
vi. set_output_delay -max 0.8 [get_ports "count"] -clock [get_clocks "clk"]
vii. set_input_transition 0.12 [all_inputs]
viii. set_load 0.15 [all_outputs]
ix. set_max_fanout 30.00 [current_design]

i→ Creates a Clock named "clk" with Time Period 2ns and On Time from t=0 to t=1.

ii, iii → Sets Clock Rise and Fall time to 100ps.

iv → Sets Clock Uncertainty to 10ps.

v, vi → Sets the maximum limit for I/O port delay to 1ps.

Step 3: Performing Synthesis:

In terminal window type the following commands

genus

set_db lib_search_path ./lib/90
set_db library slow.lib
set_db hdl search path /
read_hdl design_filename.v (ex:counter.v)
elaborate
read_sdc constraints_filename.sdc (ex constraints_count.sdc) //Reading Top Level SDC
set_db syn_generic_effort medium //Effort level to medium for generic, mapping & Optimization
set_db syn_map_effort medium
set_db syn_opt_effort medium
syn_generic
syn_map
syn_opt //Performing Synthesis Mapping and Optimisation
report_timing > counter_timing.rep
//Generates Timing report for worst datapath and dumps into file
report_area > counter_area.rep //Generates Synthesis Area report and dumps into a file
report_power > counter_power.rep //Generates Power Report [Pre-Layout]
write_hdl > counter_netlist.v //Creates readable Netlist File
write_sdc > counter_sdc.sdc //Creates Block Level SDC

Note :-

1) Report_timing gives you the path with highest failing slack
where
Setup Slack = Required Time – Arrival Time.
2) Worst Setup Slack ==> Highest Arrival time ==> Highest Propagation Delay.
3) Maximum Clock Frequency = 1/ (Max Data Path Delay – Min Clock Path Delay + Tsetup)
All the Information can be gathered from report timing.
4) The Cells given in the netlist can be checked in the .lib files for their properties.

PART B

Progarm1: 4-bit Up/Down Counter

Aim: To write a Verilog code for 4bit up/down asynchronous rest counter and its test-bench for verification.

- Synthesizing the design by setting area and timing constraint and analyze reports.
- Finding the critical path and maximum frequency of operations
- Recording the power, area requirement and properties of each cell in terms of driving strength.

Fig: Block Diagram

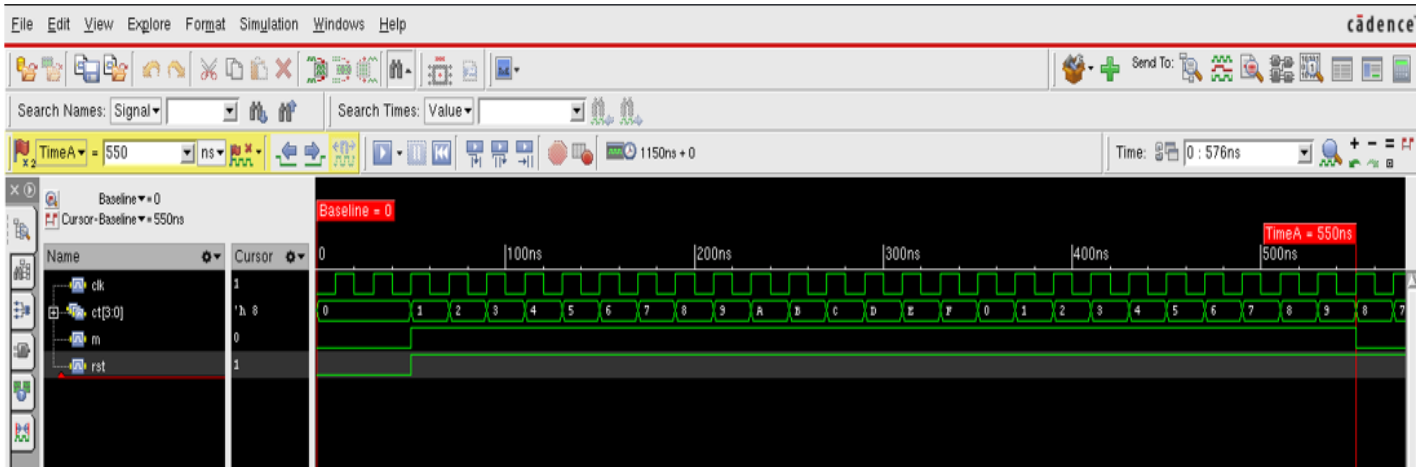
(a) 4-bit up/down counter Design Program

```
`resetall
`timescale 1ns/1ns
module counter(clk,rst,m,count);
        input clk,rst,m;
        output reg [3:0]count;
always@(posedge clk or negedge rst)
        begin
        if(!rst)
        count=4'b0;
        else if(m==0)
        count=count+1;
        else
        count=count-1;
        end
        endmodule
```

4-bit up/down counter testbench Program

```
`resetall
`rimescale 1ns/1ns
module counter_test;
reg clk, rst,m;
wire [3:0] count;
counter c1(clk,rst,m,count);
        always
        #10 clk= ~clk

        initial
        begin
        clk=1'b0; rst=1'b0;

        #200 rst=1'b1; m=1'b0;
        #200 rst=1'b1; m=1'b1;
        #200 rst=1'b1; m=1'b0;
        #1000 $finish;
end
endmodule
```

Simulation waveform:



Constraints File for counter:

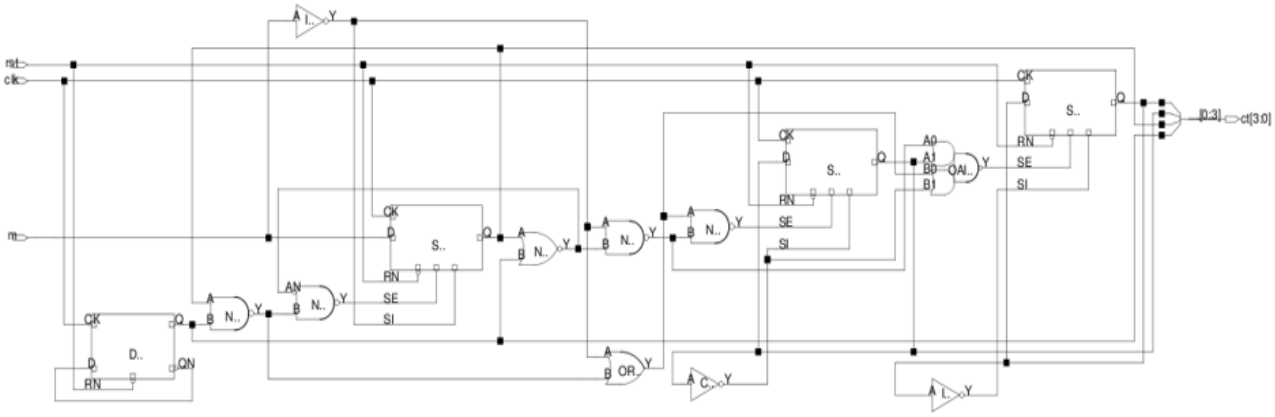
```
create_clock -name clk -period 2 -waveform {0 1} [get_ports "clk"]

set_input_delay -max 0.8 -clock clk [all_inputs]
set_output_delay -max 0.8 -clock clk [all_outputs]

set_input_transition 0.2 [all_inputs]
set_max_capacitance 30 [get_ports]

set_clock_transition -rise 0.1 [get_clocks "clk"]
set_clock_transition -fall 0.1 [get_clocks "clk"]

set_clock_uncertainty 0.01 [get_ports "clk"]

set_input_transition 0.12 [all_inputs]
set_load 0.15 [all_outputs]
set_max_fanout 30.00 [current_design]
```

SYNTHESIS REPORT:

```
@genus:root: 13> report_timing
Warning : Possible timing problems have been detected in this design. [TIM-11]
        : The design is 'counter'.
        : Use 'report timing -lint' for more information.
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 02 2021  08:18:15 pm
  Module:                 counter
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================


Path 1: MET (958 ps) Setup Check with Pin ct_reg[3]/CK->SE
          Group: clk
     Startpoint: (R) ct_reg[0]/CK
          Clock: (R) clk
       Endpoint: (R) ct_reg[3]/SE
          Clock: (R) clk

                    Capture       Launch
        Clock Edge:+    2000            0
       Src Latency:+       0            0
       Net Latency:+       0 (I)        0 (I)
           Arrival:=    2000            0

             Setup:-     243
       Uncertainty:-      50
     Required Time:=    1707
      Launch Clock:-       0
         Data Path:-     749
             Slack:=     958

#-------------------------------------------------------------------------------
--------
```

```
# Timing Point   Flags   Arc   Edge   Cell     Fanout Load Trans Delay Arrival
Instance
#                                                  (fF)   (ps)  (ps)    (ps)
Location
#-------------------------------------------------------------------------------
--------
  ct_reg[0]/CK   -       -     R     (arrival)      4    -   100     -       0
(-,-)
  ct_reg[0]/Q    -       CK->Q F     DFFRX1         3  3.4    59   364     364
(-,-)
  g245__2683/Y   -       B->Y  R     NOR2XL         2  3.4   126   112     476
(-,-)
  g240__3772/Y   -       B->Y  F     NAND2XL        2  4.4   136   126     601
(-,-)
  g238__8780/Y   -       A0->Y R     OAI22X1        1  4.9   154   148     749
(-,-)
  ct_reg[3]/SE   <<<     -     R     SDFFRHQX1      1    -     -     0     749
(-,-)
#-------------------------------------------------------------------------------
--------

@genus:root: 14> report_area
====================================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 02 2021  08:18:20 pm
  Module:                 counter
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================================

Instance Module  Cell Count  Cell Area  Net Area   Total Area  Wireload
-----------------------------------------------------------------------------
counter                  14    130.944     0.000      130.944    <none> (D)

 (D) = wireload is default in technology library
@genus:root: 15> report_power
====================================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 02 2021  08:18:24 pm
  Module:                 counter
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================================

               Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
-------------------------------------------
counter     14   743.621 67654.684 68398.306
```

Synthesis RTL Schematic:



RESULT

(b) 16-bit up/down counter Design Program

```
`resetall
`timescale 1ns/1ns
module counter(clk,rst,m,count);
        input clk,rst,m;
        output reg [15:0]count;
always@(posedge clk or negedge rst)
begin
if(!rst)
count=15'b0;
else if(m==0)
count=count+1;
else
count=count-1;
end
endmodule
```

16-bit up/down counter testbench Program

```
`resetall
`rimescale 1ns/1ns
module counter_test;
reg clk, rst,m;
wire [15:0] count;
counter c1(clk,rst,m,count);
        always
        #10 clk= ~clk

        initial
        begin

        clk=1'b0; rst=1'b0;

        #200 rst=1'b1; m=1'b0;
        #200 rst=1'b1; m=1'b1;
        #200 rst=1'b1; m=1'b0;
        #1000 $finish;
end
endmodule
```

(c) 32-bit up/down counter Design Program

```
`resetall
`timescale 1ns/1ns
module counter(clk,rst,m,count);
        input clk,rst,m;
        output reg [31:0]count;
always@(posedge clk or negedge rst)
begin
if(!rst)
count=32'b0;
else if(m==0)
count=count+1;
else
count=count-1;
end
endmodule
```

32-bit up/down testbench Program

```
`resetall
`rimescale 1ns/1ns
module counter_test;
reg clk, rst,m;
wire [32:0] count;
counter c1(clk,rst,m,count);
        always
        #10 clk= ~clk

        initial
        begin
        clk=1'b0; rst=1'b0;

        #200 rst=1'b1; m=1'b0;
        #200 rst=1'b1; m=1'b1;
        #200 rst=1'b1; m=1'b0;
        #1000 $finish;
end
endmodule
```

Progarm2: Latches and Flip Flops

Aim: Write a Verilog code for Latch and Flip-flops (D, SR, JK), Synthesize the design and compare the synthesis report.

Latches and flip-flops are the basic elements for storing information. One latch or flip-flop can store one bit of information. The main difference between latches and flip-flops is that for latches, their outputs are constantly affected by their inputs if the enable signal is asserted. In other words, when they are enabled, their content changes immediately when their inputs change.
Flip-flops, on the other hand, have their content change only either at the rising or falling edge of the enable signal. This enable signal is usually the controlling clock signal. After the rising or falling edge of the clock, the flip-flop content remains constant even if the input changes. There are basically four main types of latches and flip-flops: SR, D, and JK. The major differences in these flip-flop types are the number of inputs they have and how they change state. For each type, there are also different variations that enhance their operations

(a) D latch Design Program:

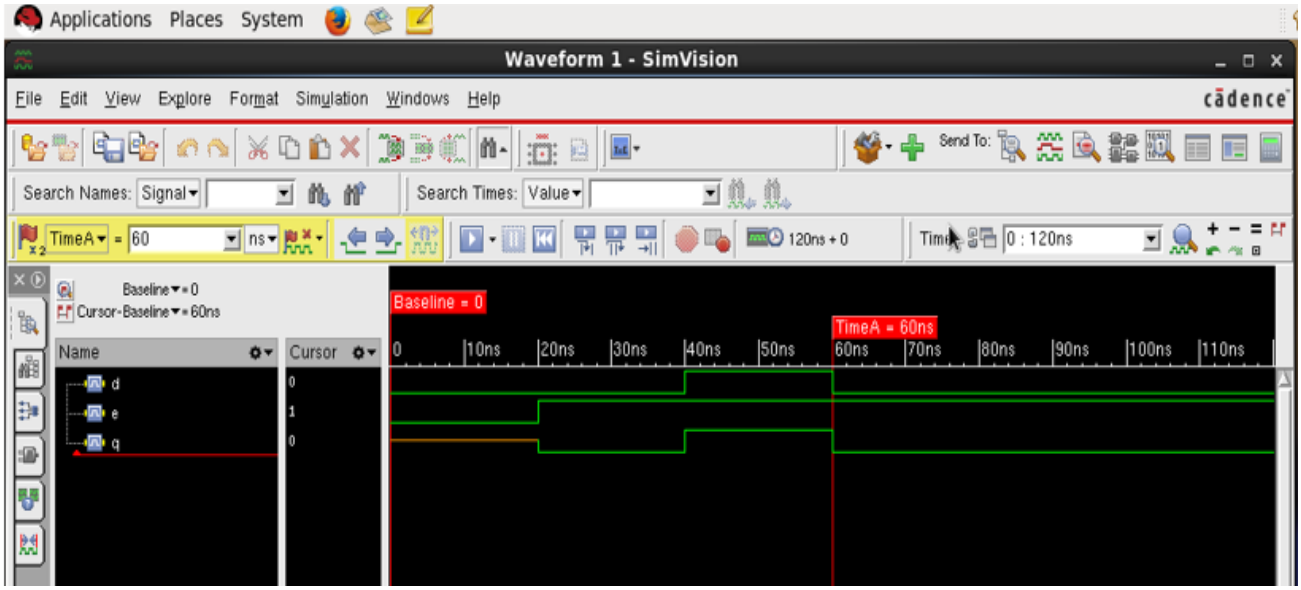
```
    `resetall
    `timescale 1ns/1ns
module dlatch(e,d,q);
input e,d;
output q;
reg q;
always @(e or d)
begin
        if(!e)
q<= 1'bz;
else
q<= d;
end
endmodule
```

(a) D latch test_bench Program:

```
    `resetall
    `timescale 1ns/1ns
module dlatch_test;
        wire q;
        reg e,d;
dlatch d1(e,d,q);
        initial
        begin
d=1'b0;  e=1'b0;
```

#20 e=1'b1; d=1'b0;
#20 e=1'b1; d=1'b1;
#20 e=1'b1; d=1'b0;
#100 $finish;
end
endmodule

Simulation Waveform:

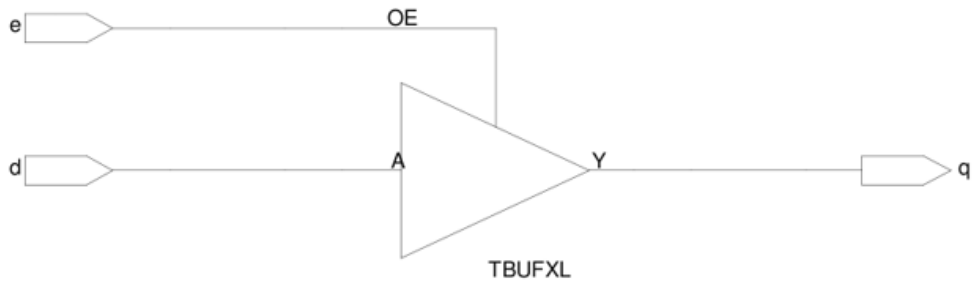


SYNTHESIS REPORT:

```
@genus:root: 13> report_timing
Warning : Possible timing problems have been detected in this design. [TIM-11]
        : The design is 'dlatch1'.
        : Use 'report timing -lint' for more information.
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 09 2021  07:42:27 pm
  Module:                 dlatch1
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

Some unconstrained paths have not been displayed.
Use -unconstrained or set the root attribute 'timing_report_unconstrained' to
'true' to see only these unconstrained paths.

@genus:root: 14> report_area
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 09 2021  07:42:31 pm
```

```
Module:                  dlatch1
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
================================================================

Instance Module  Cell Count  Cell Area  Net Area   Total Area  Wireload
-------------------------------------------------------------------------
dlatch1                  1      9.083     0.000        9.083    <none> (D)

 (D) = wireload is default in technology library
@genus:root: 15> report_power
================================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 09 2021  07:42:35 pm
  Module:                 dlatch1
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
================================================================

                Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
-------------------------------------------
dlatch1      1    41.633   266.335   307.969
```

Synthesis RTL Schematic:



RESULT

(b) D flip flop design program:

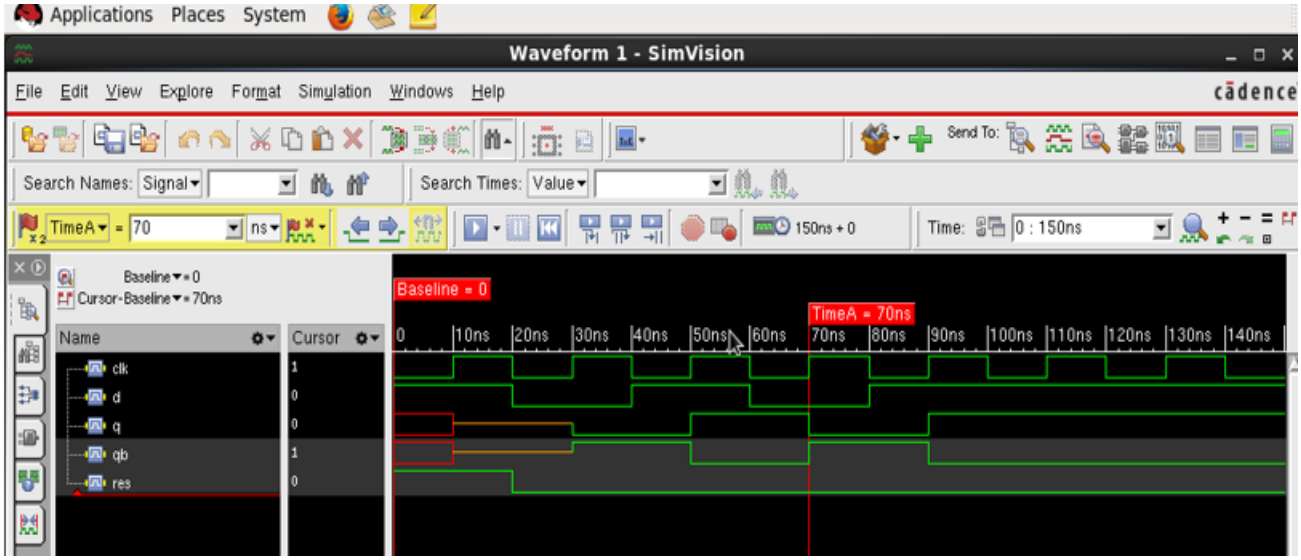
```
`resetall
`timescale 1ns/1ns
module dff1(clk,res,d,q,qb);
input clk,res,d;
output q, qb;
reg q, qb;
        always @(posedge clk)
        begin
                if(res)
                begin q=1'bz;
                        qb=1'bz;
                        end
                         else

                        if(d==1'b1)
                        begin

                                q=d;
                                qb=~q;
                         end
                         else
                         begin
                                q=1'b0;
                                qb=~q;
                end     end
        end
endmodule
```

(b)D flip flop test bench:

```
`resetall
`timescale 1ns/1ns
module dff2;
reg clk,res,d;
wire q,qb;
  dff1 n1(clk,res,d,q,qb);
        initial
        begin
clk=1'b0;
end
always
#10 clk=~clk;
initial
```

```
always
res=1'b1;d=1'b1;
#20 res=1'b0;d=1'b0;
#20 res=1'b0;d=1'b1;
#20 res=1'b0;d=1'b0;
#20 res=1'b0;d=1'b1;
#100 $finish;
```

Simulation Waveform:

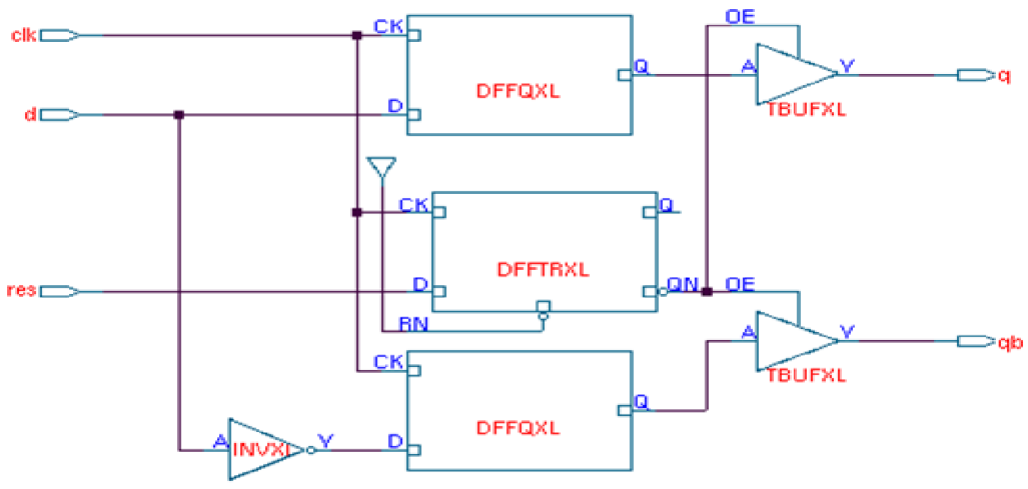


Synthesize Report:

```
@genus:root: 13> report_timing
Warning : Possible timing problems have been detected in this design. [TIM-11]
        : The design is 'dff1'.
        : Use 'report timing -lint' for more information.
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 08 2021  08:49:22 pm
  Module:                 dff1
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================


Path 1: MET (490 ps) Late External Delay Assertion at pin qb
          Group: clk
     Startpoint: (R) q_reg9/CK
          Clock: (R) clk
       Endpoint: (F) qb
          Clock: (R) clk

                    Capture       Launch
       Clock Edge:+    2000            0
      Src Latency:+       0            0
      Net Latency:+       0 (I)        0 (I)
          Arrival:=    2000            0

      Output Delay:-    1000
       Uncertainty:-      10
     Required Time:=     990
      Launch Clock:-       0
         Data Path:-     500
             Slack:=     490

Exceptions/Constraints:
  output_delay             1000            constraints_dff.sdc_line_8

#-------------------------------------------------------------------------------
--------
# Timing Point   Flags   Arc   Edge   Cell     Fanout Load Trans Delay Arrival
Instance
#                                                  (fF)   (ps)  (ps)    (ps)
Location
#-------------------------------------------------------------------------------
--------
  q_reg9/CK      -       -     R     (arrival)      3    -   100     -       0
(-,-)
  q_reg9/Q       -       CK->Q F     DFFQX1         1  2.7    47   330     330
(-,-)
  g13/Y          -       A->Y  R     CLKINVX1       2  6.8    66    66     396
(-,-)
  qb_tri__4296/Y -       OE->Y F     TBUFXL         1  0.8    21   104     500
(-,-)
  qb             <<<     -     F     (port)         -    -     -     0     500
(-,-)
```

```
#-------------------------------------------------------------------------------
--------

@genus:root: 14> report_area
====================================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 08 2021  08:49:27 pm
  Module:                 dff1
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================================

Instance Module  Cell Count  Cell Area  Net Area   Total Area  Wireload
-------------------------------------------------------------------------------
dff1                     7     70.392     0.000       70.392    <none> (D)

 (D) = wireload is default in technology library
@genus:root: 15> report_power
====================================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 08 2021  08:49:31 pm
  Module:                 dff1
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================================

                Leakage   Dynamic    Total
Instance Cells  Power(nW) Power(nW) Power(nW)
-------------------------------------------
dff1         7   401.692 36586.950 36988.643
```

Synthesis RTL Schematic:



RESULT:

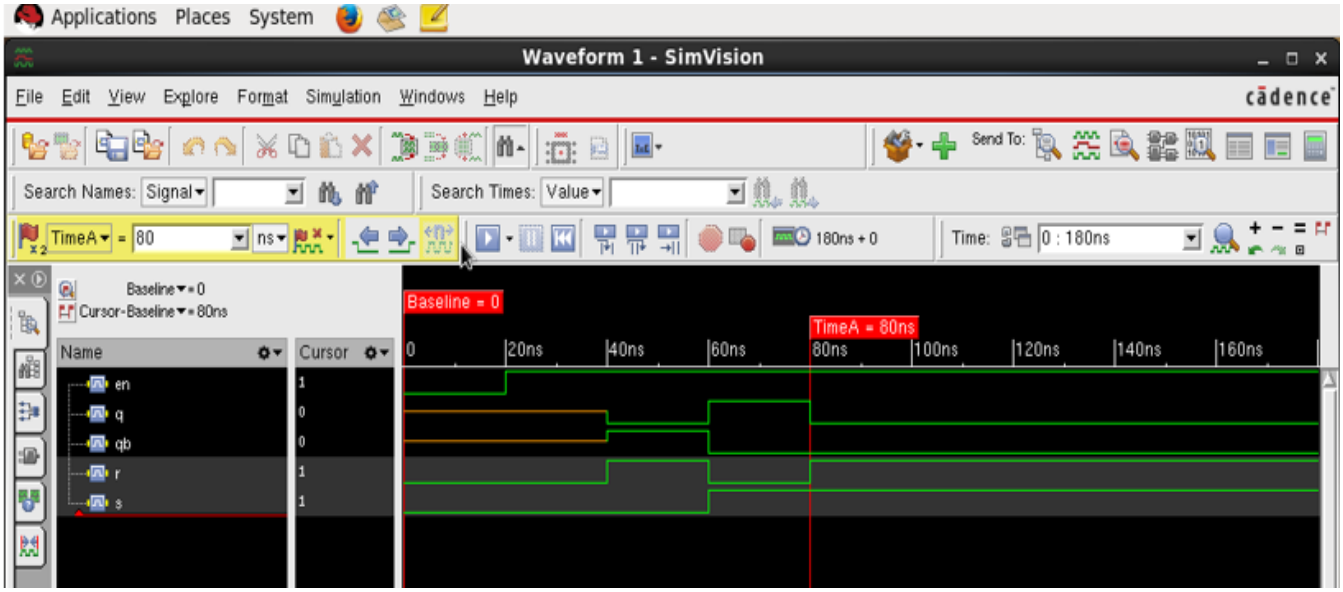
(c)SR Latch Design Program:

```
`resetall
`timescale 1ns/1ns
module srlatch(en,s,r,q,qb);
input s,r,en;
output q, qb;
reg q, qb;
always @(en or s or r)
begin
if(!en)
begin
q=1'bz;
qb=1'bz;
end
else
begin
        if(s==1'b0 && r==1'b0)
begin

        q=q;
        qb=qb;
end
        else if(s==1'b0 && r==1'b1)
begin
        q=1'b0;
        qb=1'b1;
end
        else if(s==1'b1 && r==1'b0)
begin
        q=1'b1;
        qb=1'b0;
end
        else
begin
        q=1'b0;
        qb=1'b0;

end
end
end
endmodule
```

(c)SR Latch Test-bench Program:

```
`resetall
`timescale 1ns/1ns
module srl_tt;
wire q,qb;
reg en,s,r;
srlatch l1(en,s,r,q,qb);
initial
begin
en= 1'b0; s=1'b0; r=1'b1;
#20 en= 1'b1; s=1'b1; r=1'b0;
#20 en= 1'b1; s=1'b0; r=1'b1;
#20 en= 1'b1; s=1'b0; r=1'b0;
#20 en= 1'b1; s=1'b1; r=1'b1;
#100 $finish;
end
endmodule
```

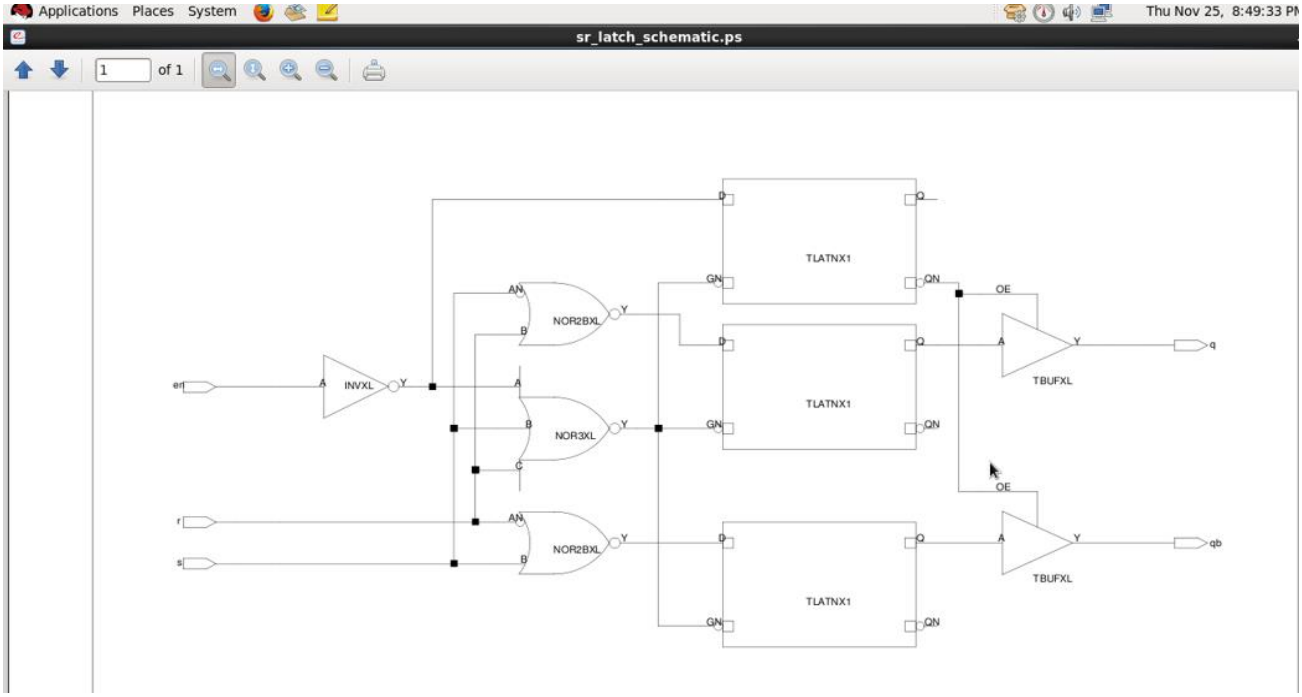
Simulation Waveform:



Synthesis Report:

```
@genus:root: 15> report_timing
Warning : Possible timing problems have been detected in this design. [TIM-11]
        : The design is 'srlatch'.
        : Use 'report timing -lint' for more information.
====================================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 02 2021  07:16:34 pm


  Module:                 srlatch
 Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================================

Some unconstrained paths have not been displayed.
Use -unconstrained or set the root attribute 'timing_report_unconstrained' to
'true' to see only these unconstrained paths.

@genus:root: 16> report_area
====================================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 02 2021  07:16:42 pm
  Module:                 srlatch
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================================

Instance Module  Cell Count  Cell Area  Net Area   Total Area  Wireload
-----------------------------------------------------------------------------
srlatch                  9     77.204     0.000       77.204    <none> (D)

 (D) = wireload is default in technology library

@genus:root: 17> report_power
====================================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 02 2021  07:16:46 pm
  Module:                 srlatch
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================================

               Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
-------------------------------------------
srlatch      9   464.521  2144.658  2609.179
```

Synthesis RTL Schematic:



RESULT:

(d)SR Flip-Flop Design:

```
`resetall
`timescale 1ns/1ns
module srf1(clk,res,s,r,q,qb);
input  clk, res, s, r;
output q,qb;
reg q, qb;
        always @(posedge clk)
        begin
        if(res)

                begin   q=1'bz;
                        qb=1'bz;

        else    end
        begin
```

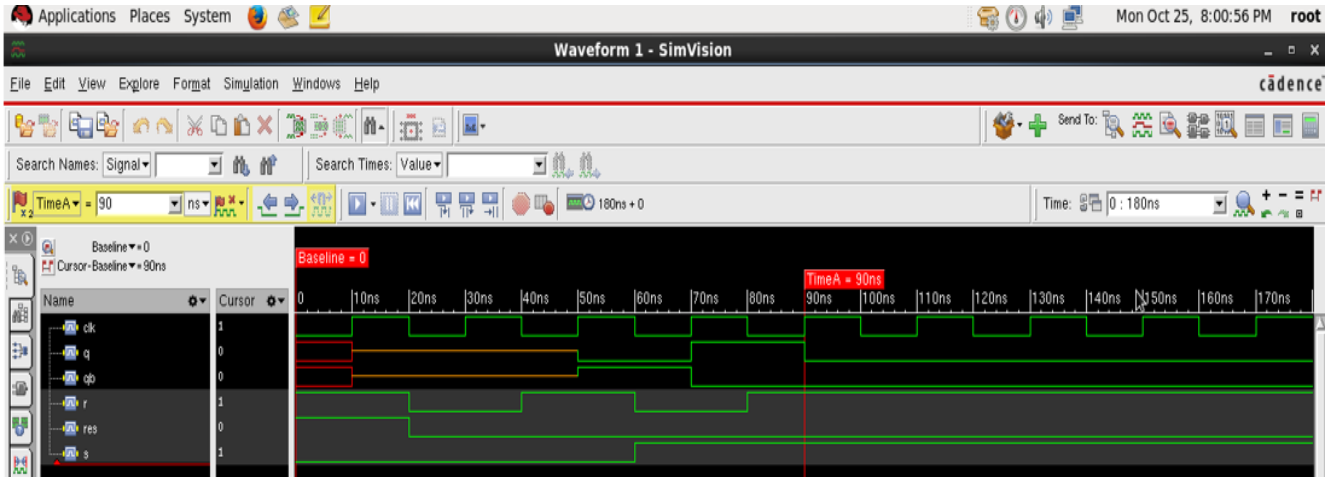
```
                if(s==1'b0 && r==1'b0)
                begin
                        q=q;
                        qb=qb;
                end
                else if(s==1'b0 && r==1'b1)
                begin
                        q=1'b0;
                        qb=~q;
                end
                else if(s==1'b1 && r==1'b0)
                begin

                        q=1'b1;
                        qb=~q;

                end
                else
                begin   q=1'b0;
                        qb=1'b0;

                end


        end
end

   endmodule
```

(d)SR Flip-Flop Testbench:

```
   `resetall
   `timescale 1ns/1ns
   module  srf2;
   reg clk,res,s,r;
   wire  q, qb;
   srf1 n1(clk,res,s,r,q,qb);
        initial
        begin
                clk=1'b0;
        end
        always

        #10 clk=~clk;

        initial
                begin
```

```
                res=1'b1;
                s=1'b0;
                r=1'b1;
                #20 res=1'b0;s=1'b0;r=1'b1;
                #20 res=1'b0;s=1'b0;r=1'b0;
                #20 res=1'b0;s=1'b1;r=1'b0;
                #20 res=1'b0;s=1'b1;r=1'b1;
                #100 $finish;

    end
    endmodule
```

Simulation Result:



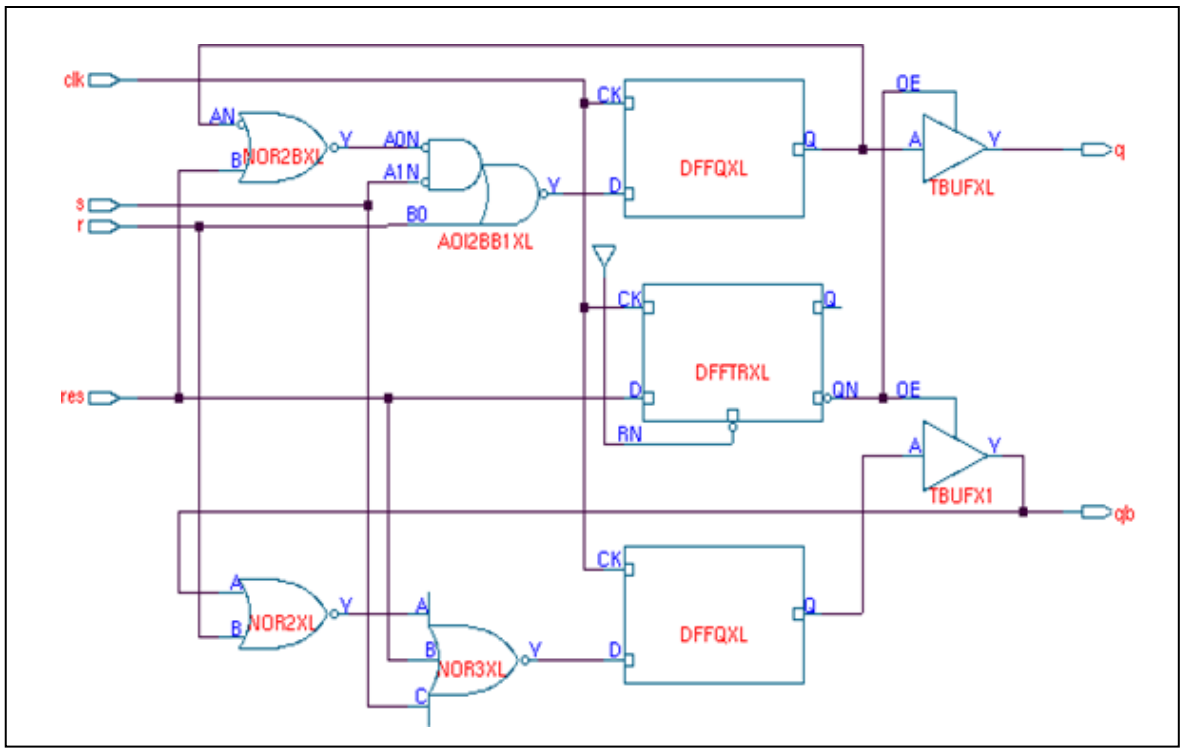
Synthesis Report:

```
@genus:root: 13> report_timing
Warning : Possible timing problems have been detected in this design. [TIM-11]
        : The design is 'srff'.
        : Use 'report timing -lint' for more information.
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Oct 25 2021  08:16:31 pm
  Module:                 srff
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================


Path 1: MET (469 ps) Late External Delay Assertion at pin qb
          Group: clk
     Startpoint: (R) q_reg36/CK
          Clock: (R) clk
       Endpoint: (F) qb
          Clock: (R) clk
```

```
                    Capture       Launch
       Clock Edge:+    2000            0
      Src Latency:+       0            0
      Net Latency:+       0 (I)        0 (I)
          Arrival:=    2000            0

     Output Delay:-    1000
      Uncertainty:-      10
    Required Time:=     990
     Launch Clock:-       0
        Data Path:-     521
            Slack:=     469



Exceptions/Constraints:
  output_delay             1000            constraints_srff.sdc_line_9

#--------------------------------------------------------------------------
--------
# Timing Point   Flags   Arc   Edge   Cell     Fanout Load Trans Delay Arrival
Instance
#                                                 (fF)   (ps)  (ps)    (ps)
Location
#--------------------------------------------------------------------------
--------
  q_reg36/CK     -       -     R     (arrival)      3    -   100     -       0
(-,-)
  q_reg36/Q      -       CK->Q F     DFFQX1         1  2.7    47   330     330
(-,-)
  g185/Y         -       A->Y  R     CLKINVX1       2  6.8    66    66     396
(-,-)
  qb_tri__4547/Y -       OE->Y F     TBUFXL         2  2.4    41   125     521
(-,-)
  qb             <<<     -     F     (port)         -    -     -     0     521
(-,-)
#--------------------------------------------------------------------------
--------

@genus:root: 14> report_area
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Oct 25 2021  08:16:39 pm
  Module:                 srff
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

Instance Module  Cell Count  Cell Area  Net Area   Total Area  Wireload
-----------------------------------------------------------------------------
srff                    11     85.530     0.000       85.530    <none> (D)

 (D) = wireload is default in technology library
@genus:root: 15> report_power
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
```

```
Generated on:            Oct 25 2021  08:16:45 pm
  Module:                  srff
  Operating conditions:    slow (balanced_tree)
  Wireload mode:           enclosed
  Area mode:               timing library
================================================================

                 Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
------------------------------------------------
srff        11   469.157 39045.295 39514.453
```

Synthesize RTL Schematic:



RESULT

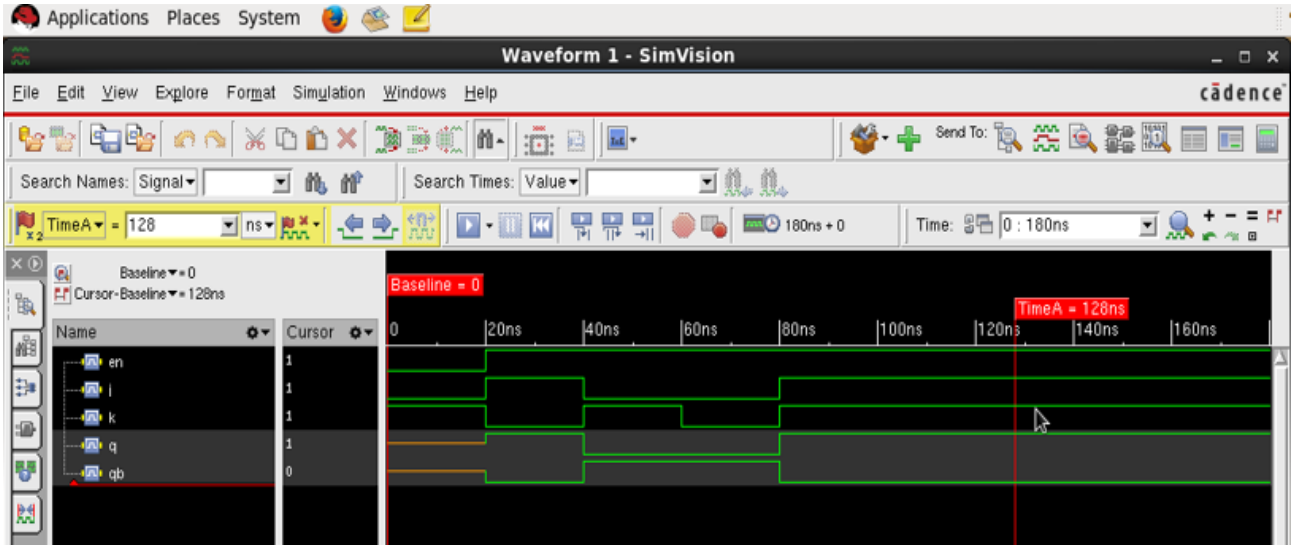
JK Latch Design:

```
`resetall
`timescale 1ns/1ns
module jklatch(en,j,k,q,qb);
input j,k,en;
output q, qb;
reg q, qb;
always @(en or j or k)
begin
if(!en)
begin
q=1'bz;
qb=1'bz;
end
else
begin
        if(j==1'b0 && k==1'b0)
begin
        q=q;
        qb=qb;
end
        else if(j==1'b0 && k==1'b1)
begin
        q=1'b0;
        qb=1'b1;
end
        else if(j==1'b1 && k==1'b0)
begin
        q=1'b1;
        qb=1'b0;
end
        else
begin
        q=~q;
        qb=~qb;

end
end
end
endmodule
```

JK Latch testbench:

```
`resetall
`timescale 1ns/1ns
module jkl_tt;
wire q,qb;
reg en,j,k;
jklatch l1(en,j,k,q,qb);
initial
begin
en= 1'b0; j=1'b0; k=1'b1;
#20 en= 1'b1; j=1'b1; k=1'b0;
#20 en= 1'b1; j=1'b0; k=1'b1;
#20 en= 1'b1; j=1'b0; k=1'b0;
#20 en= 1'b1; j=1'b1; k=1'b1;
#100 $finish;
end
endmodule
```

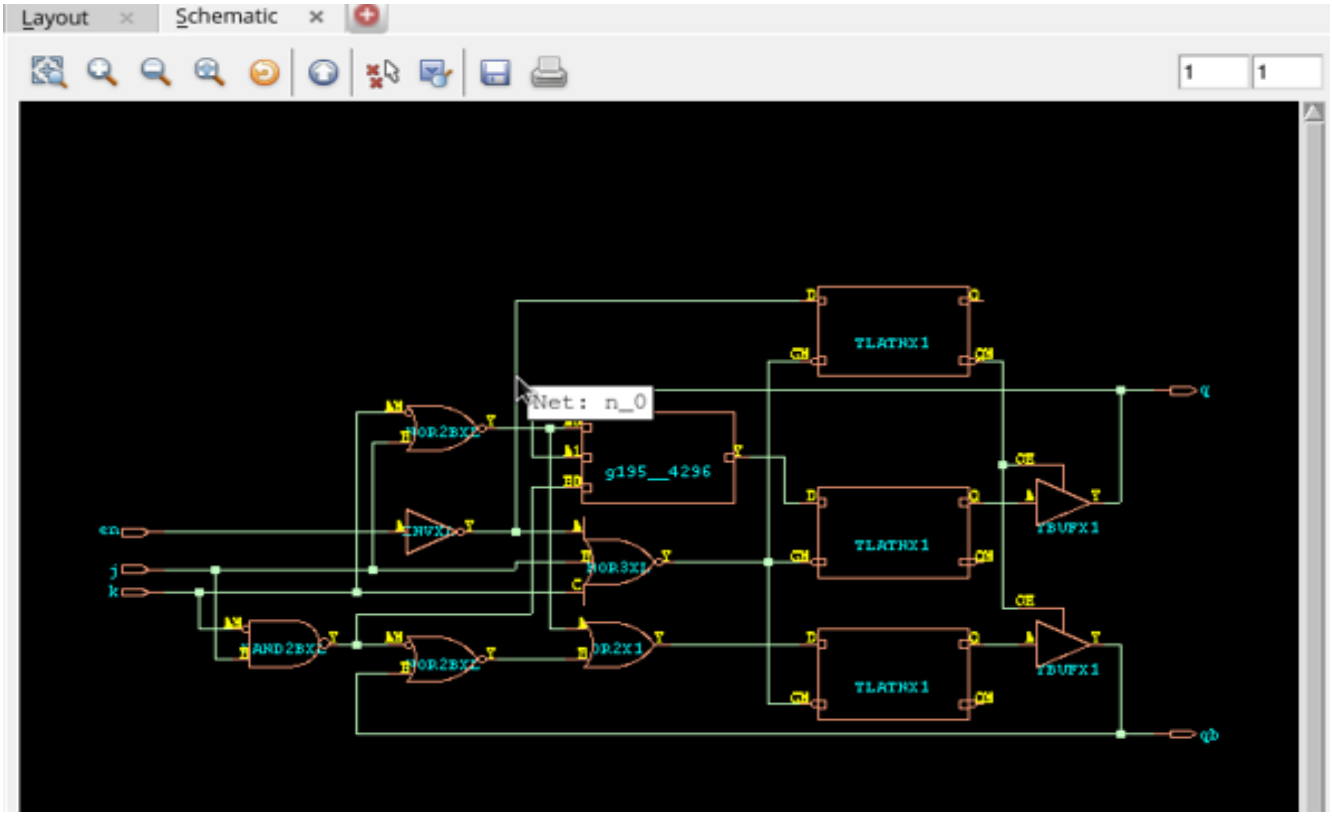
Simulation waveform:



Synthesize Report:

```
@genus:root: 14> reporrt_timing
invalid command name "reporrt_timing"
@genus:root: 15> report_timing
Warning : Possible timing problems have been detected in this design. [TIM-11]
        : The design is 'jklatch'.
        : Use 'report timing -lint' for more information.
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 02 2021  08:11:10 pm
  Module:                 jklatch
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

Some unconstrained paths have not been displayed.
Use -unconstrained or set the root attribute 'timing_report_unconstrained' to
'true' to see only these unconstrained paths.

@genus:root: 16> report_area
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 02 2021  08:11:15 pm
  Module:                 jklatch
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

Instance Module  Cell Count  Cell Area  Net Area   Total Area  Wireload
-----------------------------------------------------------------------------
jklatch                  12     90.828     0.000       90.828    <none> (D)

 (D) = wireload is default in technology library

@genus:root: 17> report_power
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 02 2021  08:11:19 pm
  Module:                 jklatch
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

                Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
-------------------------------------------
jklatch     12   550.613  2816.223  3366.836
```

Synthesize RTL Schematic:



RESULT

JK Flip-Flop Design:

```
`resetall
`timescale 1ns/1ns

module jkf1(clk,res,j,k,q,qb);
input clk, res, j,k;
output q, qb;
reg q, qb;
        always @(posedge clk)
        begin
                if (res)
                begin
```

```
                q=1'bz;
                qb=1'bz;
                end
                else
                begin
                if(j==1'b0 && k==1'b0)
                begin
                        q=q;
                        qb=qb;
                end
                else if(j==1'b0 && k==1'b1)

                begin
                        q=1'b0;
                        qb=~q;
                end
                else if(j==1'b1 && k==1'b0)
                begin
                        q=1'b1;
                        qb=~q;
                end
                else if (j==1'b1 && k==1'b1)

                begin
                            q=~q;
                        qb=~q;

            end
            end

            end

    endmodule
```

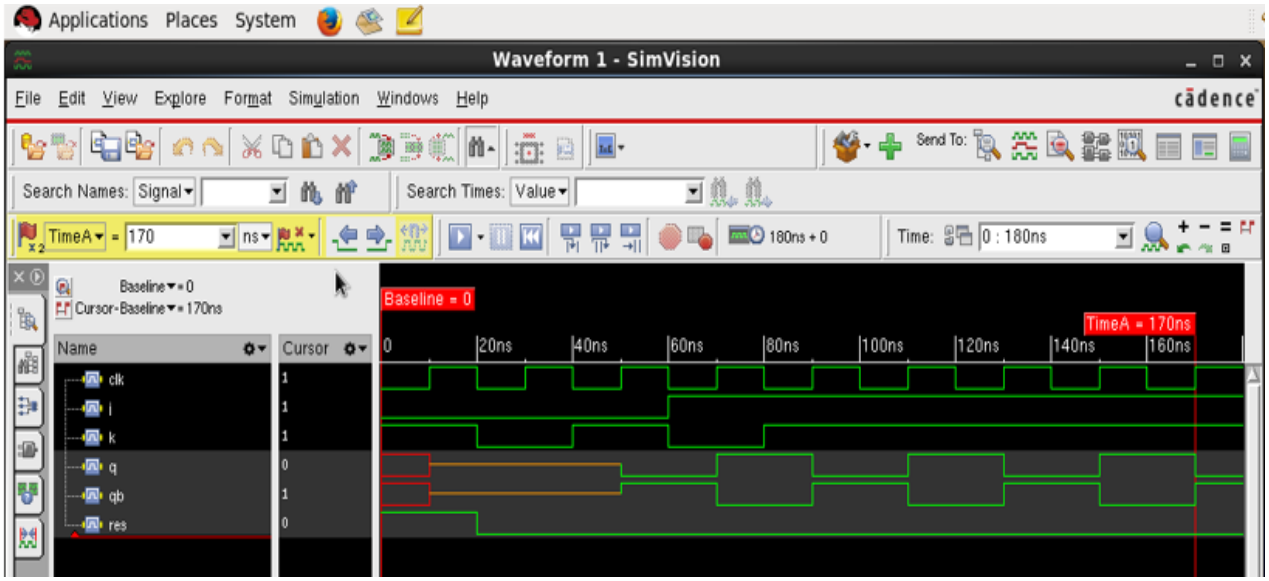
JK flip-flop testbench:

```
`resetall
`timescale 1ns/1ns

module  jkf2;
reg clk, res, j, k;
wire  q, qb;
jkf1 n1 (clk,res,j,k,q,qb);
        initial
        begin
```

```
        clk=1'b0;
        always
        begin
                #10 clk=~clk;
        end
initial
begin
```

```
            res=1'b
            1;
            j=1'b0;
            k=1'b1;
            #40 res=1'b0;j=1'b0;k=1'b0;

            #20 res=1'b0;j=1'b0;k=1'b1;
            #20 res=1'b0;j=1'b1;k=1'b0;
            #20 res=1'b0;j=1'b1;k=1'b1;
            #50 $finish;

      end

      end

      end module
```

Simulation Waveform:



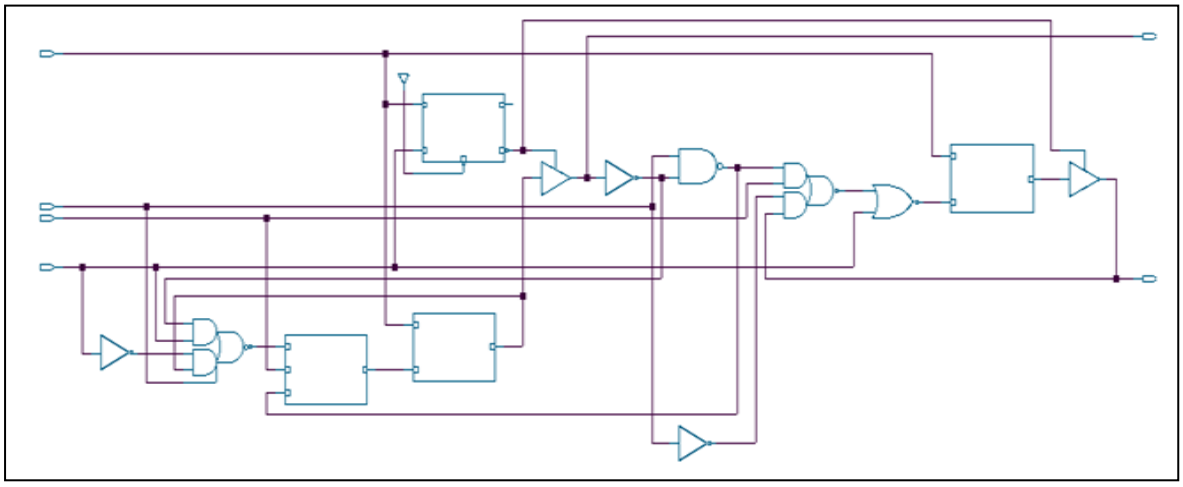
Synthesize Report:

```
@genus:root: 14> report_timing
Warning : Possible timing problems have been detected in this design. [TIM-11]
        : The design is 'jkff'.
```

```
        : Use 'report timing -lint' for more information.
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 02 2021  07:37:25 pm
  Module:                 jkff
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================


Path 1: MET (453 ps) Late External Delay Assertion at pin qb
          Group: clk
     Startpoint: (R) q_reg38/CK
          Clock: (R) clk
       Endpoint: (F) qb
          Clock: (R) clk


Capture       Launch
        Clock Edge:+    2000            0
       Src Latency:+       0            0
       Net Latency:+       0 (I)        0 (I)
           Arrival:=    2000            0

      Output Delay:-    1000
       Uncertainty:-      10


      Required Time:=     990
      Launch Clock:-       0
         Data Path:-     537
             Slack:=     453

Exceptions/Constraints:
  output_delay             1000           constraints_jkff.sdc_line_9

#-------------------------------------------------------------------------------
--------
# Timing Point   Flags   Arc   Edge   Cell     Fanout Load Trans Delay Arrival
Instance
#                                                 (fF)   (ps)   (ps)    (ps)
Location
#-------------------------------------------------------------------------------
--------
  q_reg38/CK     -       -     R     (arrival)      3    -   100     -       0
(-,-)
  q_reg38/Q      -       CK->Q F     DFFQX1         1  2.7    47   330     330
(-,-)
  g218/Y         -       A->Y  R     CLKINVX1       2  6.8    66    66     396
(-,-)
  qb_tri__4547/Y -       OE->Y F     TBUFX1         2  5.1    50   141     537
(-,-)
  qb             <<<     -     F     (port)         -    -     -     0     537
(-,-)
#-------------------------------------------------------------------------------

@genus:root: 15> report_map
```

```
invalid command name "report_map"
@genus:root: 16> report_area
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 02 2021  07:37:38 pm
  Module:                 jkff
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

Instance Module  Cell Count  Cell Area  Net Area   Total Area  Wireload
-----------------------------------------------------------------------------
jkff                    13     97.640     0.000       97.640    <none> (D)

 (D) = wireload is default in technology library
@genus:root: 17> report_power
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 02 2021  07:37:43 pm
  Module:                 jkff
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

                Leakage   Dynamic    Total
Instance Cells Power(nW) Power(nW) Power(nW)
-----------------------------------------------
jkff        13   504.704 44198.794 44703.498
```

Synthesize RTL Schematic:



RESULT

Program 3: 4-Bit Adder

Aim: To write a Verilog code for 4bit adder and verify the functionality using Test bench.

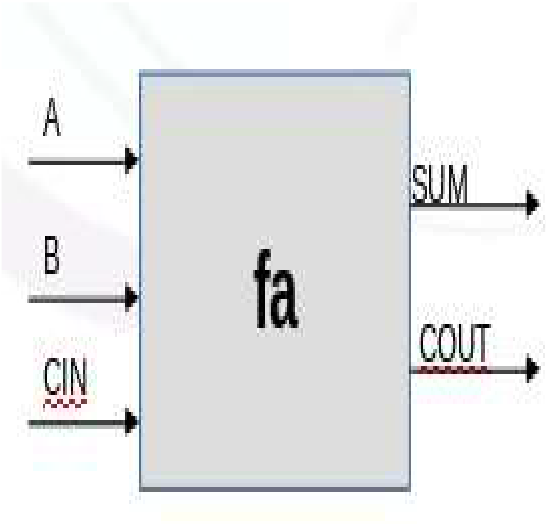
• Synthesize, Analyze Reports and Netlist, Critical Path and Max Operating Frequency.

• From the report generated find the total number of cells, power requirement and total area requirement.

Design Information and Bock Diagram:

A full adder is a combinational circuit that performs the arithmetic sum of three input bits Ai, addend Bi and carry in C in from the previous adder. Its results contain the sum Si and the carryout, C out to the next stage. So to design a 4-bit adder circuit we start by designing the 1 −bit full adder then connecting the four 1-bit full adders to get the 4-bit adder as shown in the diagram below.

For the 1-bit full adder, the design begins by drawing the Truth Table for the three input and the corresponding output SUM and CARRY.



A	B	C	SUM OUT	CARRY OUT
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Fig: Diagram and truth table of full adder

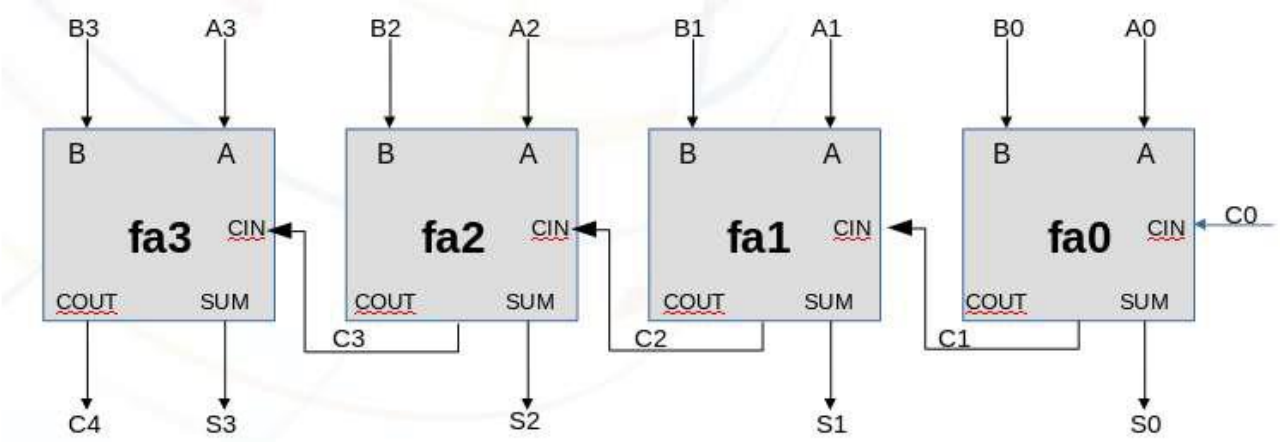


Fig: Diagram of 4 Bit Adder

Adder Design Program:

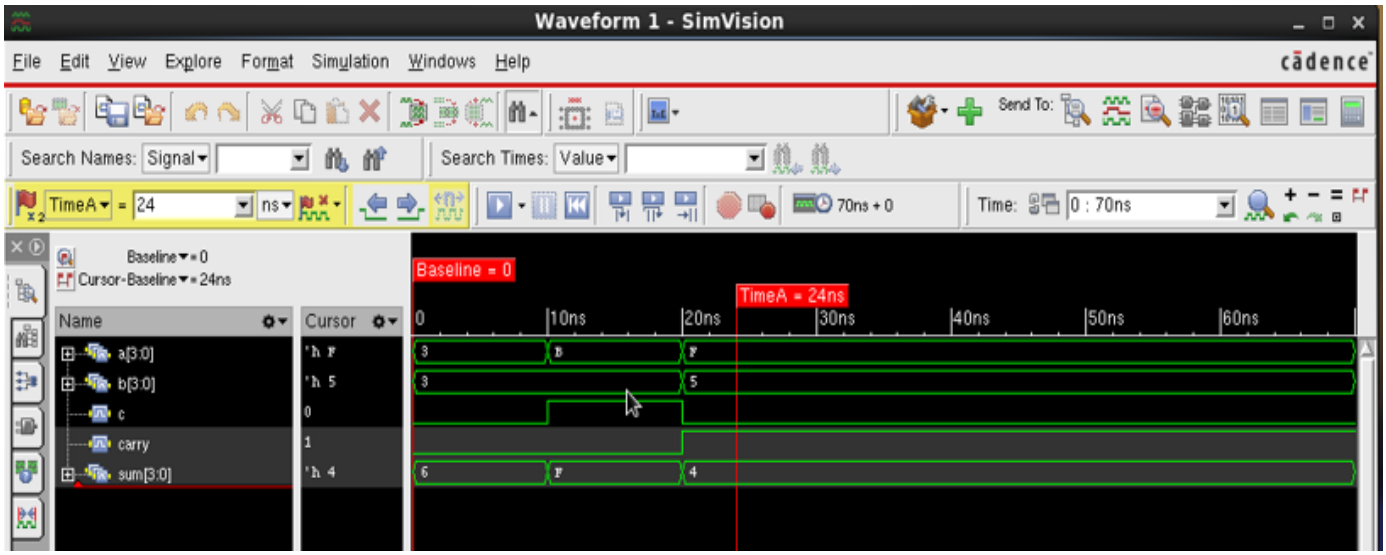
```
`resetall
`timescale 1ns/1ns
module adder_4bit(a,b,c,sum,carry);
output [3:0]sum;
output carry;
input[3:0]a,b;
input c;
wire s1,s2,s3;
full_adder fa0 (a[0],b[0],c,sum[0], s1);
full_adder fa1 (a[1],b[1],s1,sum[1], s2);
full_adder fa2 (a[2],b[2],s2,sum[2], s3);
full_adder fa3 (a[3],b[3],s3,sum[3],carry);
endmodule
module full_adder(a,b,c,sum,carry);
input a,b,c;
output sum,carry;
assign sum = a^b^c;
assign carry = ((a&b)| (b&c)|(c&a));
endmodule
```

Adder-testbench Program:

```
`resetall
`timescale 1ns/1ns
module test_4_bit;
reg [3:0] a;
reg [3:0] b;
reg c;
```

```
wire [3:0] sum;
wire carry;
adder_4bit dut(a,b,c,sum,carry);
initial
begin
a = 4'b0011;b=4'b0011;c = 1'b0;
#10 a = 4'b1011;b=4'b0011;c = 1'b1;
#10 a = 4'b1111;b=4'b0101;c = 1'b0;
#50 $finish;
end
endmodule
```

Simulation Waveform:

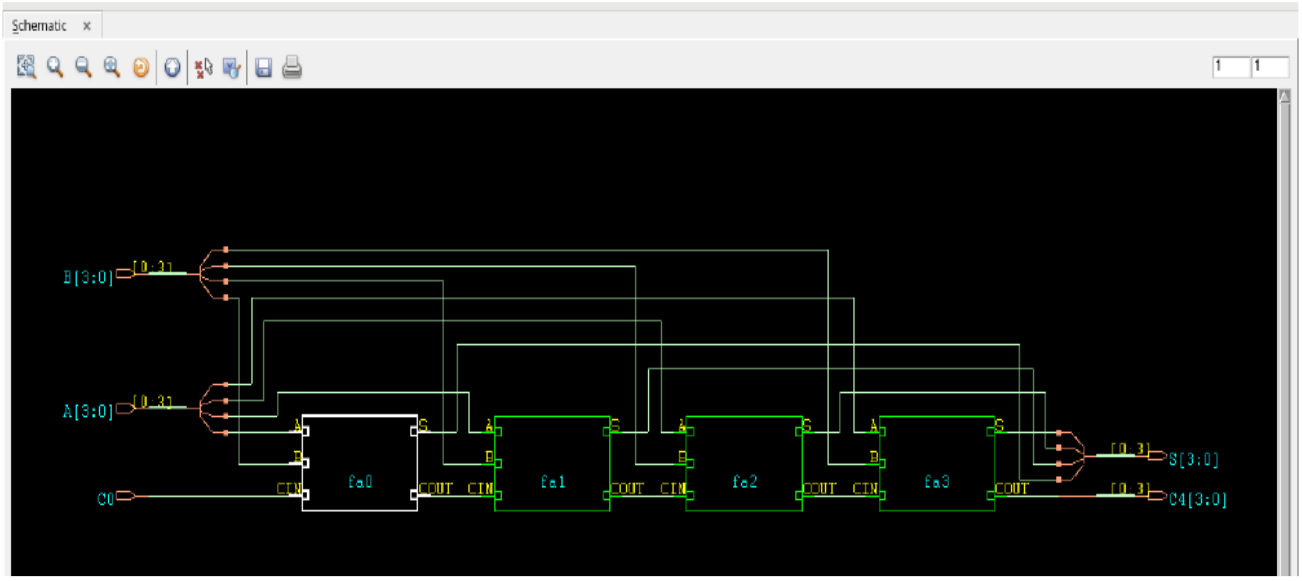


Synthesize Report:

```
@genus:root: 21> report_timing
Warning : Possible timing problems have been detected in this design. [TIM-11]
        : The design is 'adder_4bit'.
        : Use 'report timing -lint' for more information.
====================================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 15 2021  05:43:48 pm
  Module:                 adder_4bit
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================================


Some unconstrained paths have not been displayed.
Use -unconstrained or set the root attribute 'timing_report_unconstrained' to
'true' to see only these unconstrained paths.

@genus:root: 22> report_area
====================================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 15 2021  05:43:55 pm
  Module:                 adder_4bit
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================================

 Instance     Module     Cell Count  Cell Area  Net Area   Total Area  Wireload
-------------------------------------------------------------------------------
----


adder_4bit                        4     78.718     0.000       78.718    <none>
(D)
  fa3      full_adder_3           1     19.679     0.000       19.679    <none>
(D)
  fa2      full_adder_2           1     19.679     0.000       19.679    <none>
(D)
  fa1      full_adder_1           1     19.679     0.000       19.679    <none>
(D)
  fa0      full_adder             1     19.679     0.000       19.679    <none>
(D)

 (D) = wireload is default in technology library
@genus:root: 23> report_power
====================================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 15 2021  05:44:08 pm
  Module:                 adder_4bit
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
====================================================================
```

```
                    Leakage    Dynamic     Total
 Instance   Cells Power(nW)  Power(nW)  Power(nW)
------------------------------------------------
adder_4bit      4   320.470   2652.610   2973.080
  fa0           1    84.476    601.294    685.770
  fa1           1    84.476    630.160    714.636
  fa2           1    84.476    551.755    636.231
  fa3           1    67.041    392.683    459.724
```

Synthesize RTL Schematic:



RESULT

Program 4: 32-bit ALU

Aim: Write a verilog code for 32 bit ALU supporting four logical and four arithmetic operations, use case statement and if statement for ALU behavioral modeling.

- To Verify the Functionality using Test Bench
- Synthesize and compare the results using if and case statements
- Identify Critical Path and constraints

Design Information and Bock Diagram:

The ALU will take in two 32-bit values, and control line. An Arithmetic unit does the following task like addition subtraction, multi-fiction, and logical operations. As the input is given in 32 bit we get 32 bit output. The arithmetic will show only one output at a time, so a selector is necessary to select one of the operator.

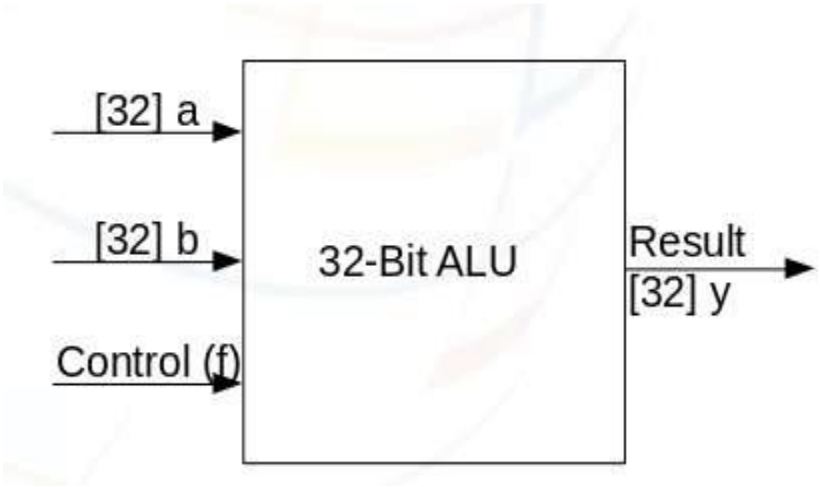


Fig: Block Diagram

32-bit ALU Design:

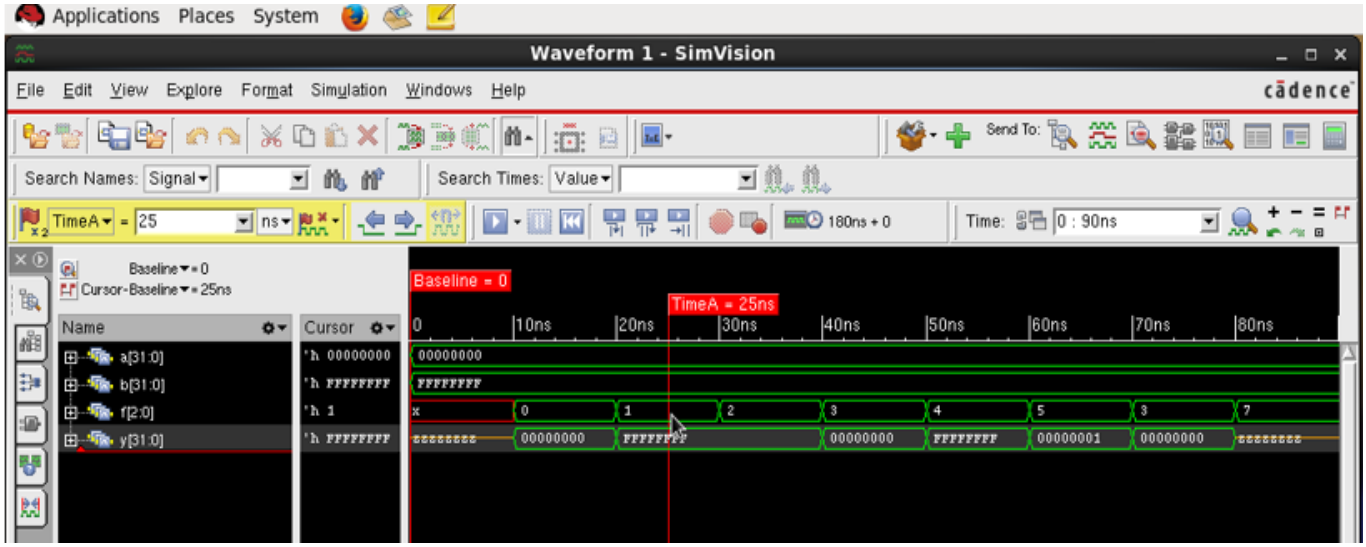
```
`resetall
`timescale 1ns/1ns
module alu_32bit1(y,a,b,f);
input [31:0]a;
input [31:0]b;
input [2:0]f;
output reg [31:0]y;
always@(*)
begin
case(f)
3'b000:y=a&b;                    //AND Operation
3'b001:y=a|b;                    //OR Operation
```

```
3'b010:y=~(a&b);          //NAND Operation
3'b011:y=~(a|b);          //NOR Operation
3'b100:y=a+b;             //Addition
3'b101:y=a-b;             //Subtraction
3'b110:y=~a;              //Not
default:y=32'bz;
endcase
end
endmodule
```

32-bit ALU Testbench:

```
`resetall
`timescale 1ns/1ns
module alu_32bit_tb_case;
reg [31:0]a;
reg [31:0]b;
reg [2:0]f;
wire [31:0]y;
alu_32bit1 test2(.y(y),.a(a),.b(b),.f(f));
initial
begin
a=32'h00000000;
b=32'hFFFFFFFF;
#10 f=3'b000;
#10 f=3'b001;
#10 f=3'b010;
#10 f=3'b011;
#10 f=3'b100;
#10 f=3'b101;
#10 f=3'b110;
#10 f=3'b111;
#100 $finish;
end
endmodule
```

Simulation Waveform:

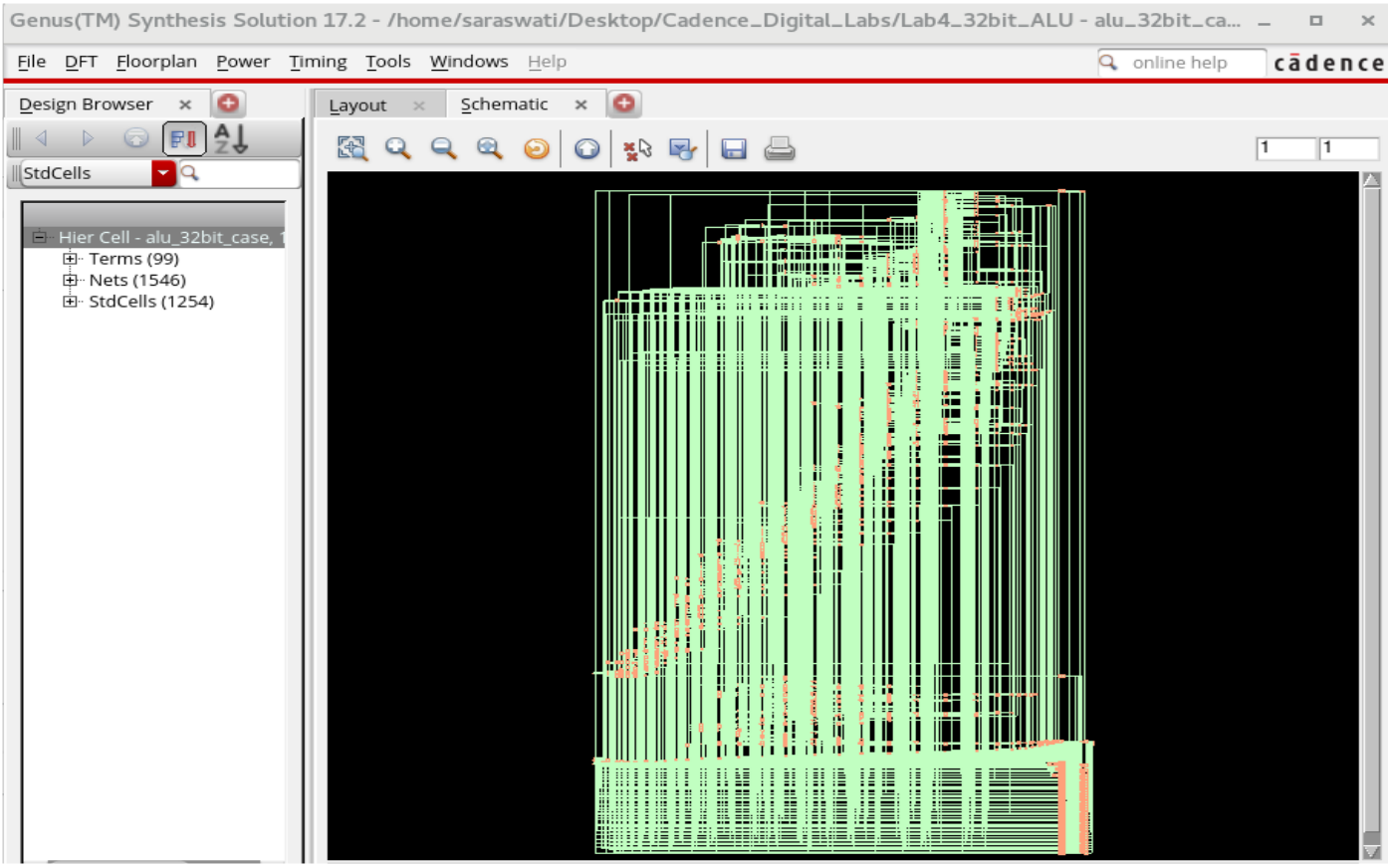


Synthesize Report:

```
@genus:root: 14> report_timing
Warning : Possible timing problems have been detected in this design. [TIM-11]
        : The design is 'alu_32bit1'.
        : Use 'report timing -lint' for more information.
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 20 2021  09:01:39 pm
  Module:                 alu_32bit1
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

Some unconstrained paths have not been displayed.
Use -unconstrained or set the root attribute 'timing_report_unconstrained' to
'true' to see only these unconstrained paths.

@genus:root: 15> report_area
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 20 2021  09:01:44 pm
  Module:                 alu_32bit1
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

 Instance  Module  Cell Count  Cell Area  Net Area   Total Area  Wireload
-------------------------------------------------------------------------------
alu_32bit1              333   2073.906     0.000     2073.906    <none> (D)
```

```
 (D) = wireload is default in technology library

@genus:root: 16> report_power
============================================================
  Generated by:           Genus(TM) Synthesis Solution 17.22-s017_1
  Generated on:           Nov 20 2021  09:01:48 pm
  Module:                 alu_32bit1
  Operating conditions:   slow (balanced_tree)
  Wireload mode:          enclosed
  Area mode:              timing library
============================================================

                  Leakage   Dynamic    Total
 Instance  Cells Power(nW) Power(nW) Power(nW)
------------------------------------------------
alu_32bit1   333  7526.723 77401.565 84928.288
```

Synthesize RTL Schematic:



RESULT

Program 5: UART

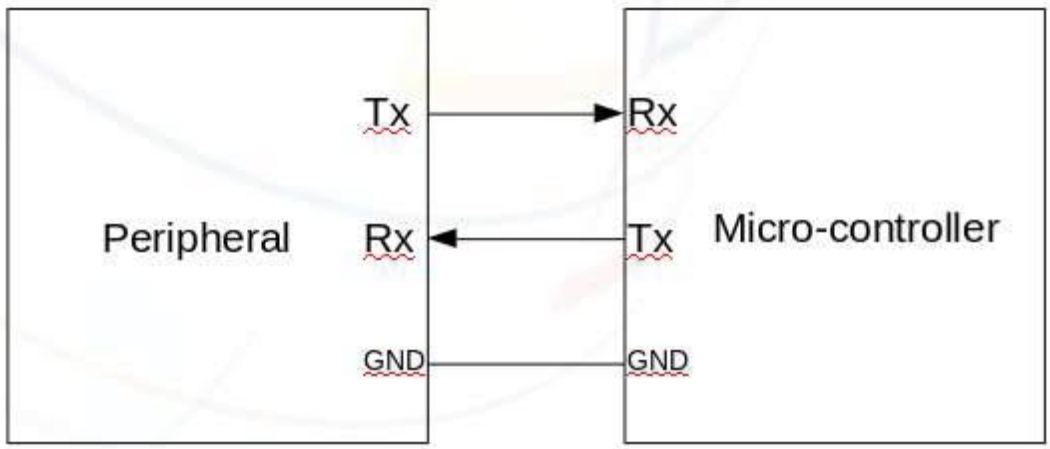
Aim: Write a verilog code for UART and carry out the following:

- To Verify the Functionality using test Bench
- Synthesize Design using constraints
- Tabulate Reports using various Constraints
- Identify Critical Path and calculate Max Operating Frequency

Design Information and Bock Diagram:

The **UART** is "Universal Asynchronous Receiver/Transmitter", and it is an inbuilt IC within a micro-controller but not like a communication protocol (I2C & SPI). The main function of UART is to serial data communication. In UART, the communication between two devices can be done in two ways namely serial data communication and parallel data communication.

The transmitter section includes three blocks namely transmit hold register, shift register and control logic. Likewise, the receiver section includes a receive hold register, shift register, and control logic. These two sections are commonly provided by a baud-rate-generator. This generator is used for generating the speed when the transmitter section & receiver section must transmit or receiver.



UART Communication

UART Design Program:

```
module UART_TX
#(parameter CLKS_PER_BIT = 217)
(
input i_Clock,
input i_TX_DV,
input [7:0] i_TX_Byte,
output o_TX_Active,
output reg o_TX_Serial,
output o_TX_Done
);
parameter IDLE = 3'b000;
parameter TX_START_BIT = 3'b001;

parameter TX_DATA_BITS = 3'b010;
parameter TX_STOP_BIT = 3'b011;
parameter CLEANUP = 3'b100;
reg [2:0] r_SM_Main = 0;
reg [7:0] r_Clock_Count = 0;
reg [2:0] r_Bit_Index = 0;
reg [7:0] r_TX_Data = 0;
reg r_TX_Done = 0;
reg r_TX_Active = 0;
always @(posedge i_Clock)
begin
case (r_SM_Main)
IDLE :
begin
o_TX_Serial <= 1'b1; // Drive Line High for Idle
r_TX_Done <= 1'b0;
r_Clock_Count <= 0;
r_Bit_Index <= 0;
28
if (i_TX_DV == 1'b1)
begin
r_TX_Active <= 1'b1;
r_TX_Data <= i_TX_Byte;
r_SM_Main <= TX_START_BIT;
end
else
r_SM_Main <= IDLE;
end // case: IDLE
// Send out Start Bit. Start bit = 0
TX_START_BIT :
begin
```

```
o_TX_Serial <= 1'b0;

// Wait CLKS_PER_BIT-1 clock cycles for start bit to finish
if (r_Clock_Count < CLKS_PER_BIT-1)
begin
r_Clock_Count <= r_Clock_Count + 1;
r_SM_Main <= TX_START_BIT;
end
else
begin
r_Clock_Count <= 0;
r_SM_Main <= TX_DATA_BITS;
end
end // case: TX_START_BIT
// Wait CLKS_PER_BIT-1 clock cycles for data bits to finish
TX_DATA_BITS :
begin
o_TX_Serial <= r_TX_Data[r_Bit_Index];
if (r_Clock_Count < CLKS_PER_BIT-1)
begin
r_Clock_Count <= r_Clock_Count + 1;
r_SM_Main <= TX_DATA_BITS;
end
else
begin
r_Clock_Count <= 0;
// Check if we have sent out all bits
29
if (r_Bit_Index < 7)
begin
r_Bit_Index <= r_Bit_Index + 1;

r_SM_Main <= TX_DATA_BITS;
end
else
begin
r_Bit_Index <= 0;
r_SM_Main <= TX_STOP_BIT;
end
end
end // case: TX_DATA_BITS
// Send out Stop bit. Stop bit = 1
TX_STOP_BIT :
begin
o_TX_Serial <= 1'b1;
// Wait CLKS_PER_BIT-1 clock cycles for Stop bit to finish
if (r_Clock_Count < CLKS_PER_BIT-1)
begin
```

```
r_Clock_Count <= r_Clock_Count + 1;
r_SM_Main <= TX_STOP_BIT;
end
else
begin
r_TX_Done <= 1'b1;
r_Clock_Count <= 0;
r_SM_Main <= CLEANUP;
r_TX_Active <= 1'b0;
end
end // case: TX_STOP_BIT
// Stay here 1 clock
CLEANUP :
begin
r_TX_Done <= 1'b1;
r_SM_Main <= IDLE;
end
default :
r_SM_Main <= IDLE;
endcase
end
assign o_TX_Active = r_TX_Active;
assign o_TX_Done = r_TX_Done;
endmodule
```

30

Source Code − Receiver :

```
// This file contains the UART Receiver. This receiver is able to
// receive 8 bits of serial data, one start bit, one stop bit,
// and no parity bit. When receive is complete o_rx_dv will be
// driven high for one clock cycle.
//
// Set Parameter CLKS_PER_BIT as follows:
// CLKS_PER_BIT = (Frequency of i_Clock)/(Frequency of UART)
// Example: 25 MHz Clock, 115200 baud UART
// (25000000)/(115200) = 217
module UART_RX
#(parameter CLKS_PER_BIT = 217)
(
input i_Clock,
input i_RX_Serial,
output o_RX_DV,
output [7:0] o_RX_Byte
);
parameter IDLE = 3'b000;
parameter RX_START_BIT = 3'b001;
parameter RX_DATA_BITS = 3'b010;
parameter RX_STOP_BIT = 3'b011;
```

```
parameter CLEANUP = 3'b100;
reg [7:0] r_Clock_Count = 0;
reg [2:0] r_Bit_Index = 0; //8 bits total

reg [7:0] r_RX_Byte = 0;

reg r_RX_DV = 0;
reg [2:0] r_SM_Main = 0;
// Purpose: Control RX state machine
always @(posedge i_Clock)
begin
case (r_SM_Main)
IDLE :
begin
r_RX_DV <= 1'b0;
r_Clock_Count <= 0;
r_Bit_Index <= 0;
31
if (i_RX_Serial == 1'b0) // Start bit detected
r_SM_Main <= RX_START_BIT;
else
r_SM_Main <= IDLE;
end
// Check middle of start bit to make sure it's still low
RX_START_BIT :
begin
if (r_Clock_Count == (CLKS_PER_BIT-1)/2)
begin
if (i_RX_Serial == 1'b0)
begin
r_Clock_Count <= 0; // reset counter, found the middle
r_SM_Main <= RX_DATA_BITS;
end
else
r_SM_Main <= IDLE;
end
else
begin
r_Clock_Count <= r_Clock_Count + 1;
r_SM_Main <= RX_START_BIT;
end
end // case: RX_START_BIT
// Wait CLKS_PER_BIT-1 clock cycles to sample serial data
RX_DATA_BITS :
begin
if (r_Clock_Count < CLKS_PER_BIT-1)
begin
r_Clock_Count <= r_Clock_Count + 1;
```

```
r_SM_Main <= RX_DATA_BITS;
end
else
begin

r_Clock_Count <= 0;
r_RX_Byte[r_Bit_Index] <= i_RX_Serial;

// Check if we have received all bits
if (r_Bit_Index < 7)
begin
r_Bit_Index <= r_Bit_Index + 1;
r_SM_Main <= RX_DATA_BITS;
end
32
else
begin
r_Bit_Index <= 0;
r_SM_Main <= RX_STOP_BIT;
end
end
end // case: RX_DATA_BITS
// Receive Stop bit. Stop bit = 1
RX_STOP_BIT :
begin

// Wait CLKS_PER_BIT-1 clock cycles for Stop bit to finish
if (r_Clock_Count < CLKS_PER_BIT-1)
begin
r_Clock_Count <= r_Clock_Count + 1;
r_SM_Main <= RX_STOP_BIT;
end
else
begin
r_RX_DV <= 1'b1;
r_Clock_Count <= 0;
r_SM_Main <= CLEANUP;
end

end // case: RX_STOP_BIT
// Stay here 1 clock
CLEANUP :
begin
r_SM_Main <= IDLE;
r_RX_DV <= 1'b0;
end
default :
r_SM_Main <= IDLE;
```

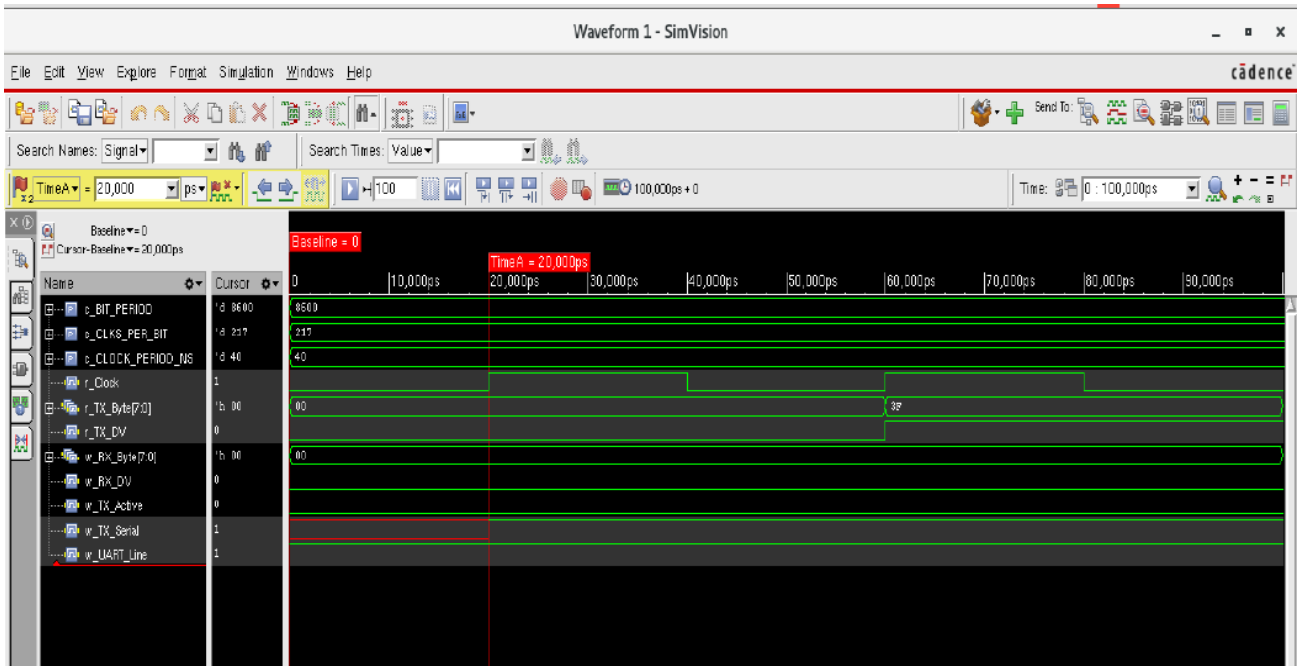
```
endcase
end
assign o_RX_DV = r_RX_DV;
assign o_RX_Byte = r_RX_Byte;
endmodule // UART_RX
```

UART Test bench:

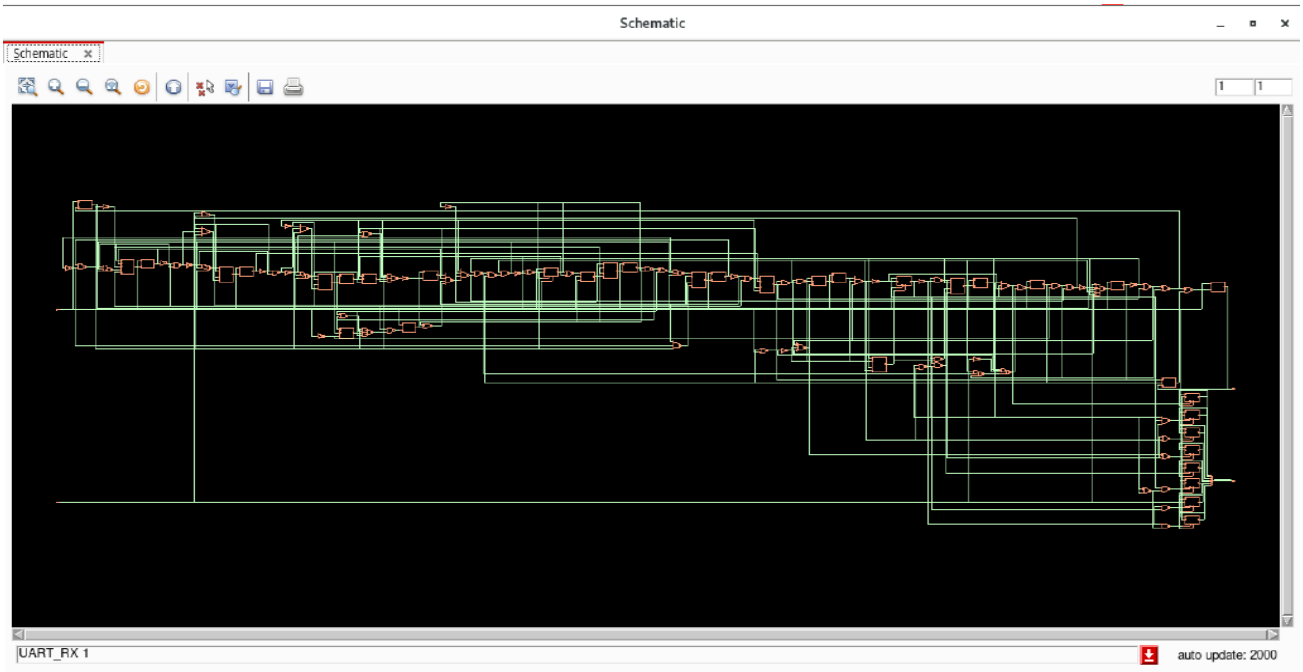
```
// This testbench will exercise the UART RX.
// It sends out byte 0x37, and ensures the RX receives it correctly.
`timescale 1ns/10ps
`include "uart_tx.v"
`include "uart_rx.v"
module UART_TB ();
// Testbench uses a 25 MHz clock
// Want to interface to 115200 baud UART
// 25000000 / 115200 = 217 Clocks Per Bit.
parameter c_CLOCK_PERIOD_NS = 40;
parameter c_CLKS_PER_BIT = 217;
parameter c_BIT_PERIOD = 8600;
reg r_Clock = 0;
reg r_TX_DV = 0;
wire w_TX_Active, w_UART_Line;
wire w_TX_Serial;
reg [7:0] r_TX_Byte = 0;

wire [7:0] w_RX_Byte;
UART_RX #(.CLKS_PER_BIT(c_CLKS_PER_BIT)) UART_RX_Inst
(.i_Clock(r_Clock),
.i_RX_Serial(w_UART_Line),
.o_RX_DV(w_RX_DV),
.o_RX_Byte(w_RX_Byte)
);
UART_TX #(.CLKS_PER_BIT(c_CLKS_PER_BIT)) UART_TX_Inst
(.i_Clock(r_Clock),
.i_TX_DV(r_TX_DV),
.i_TX_Byte(r_TX_Byte),

.o_TX_Active(w_TX_Active),
.o_TX_Serial(w_TX_Serial),
.o_TX_Done()
);
// Keeps the UART Receive input high (default) when
// UART transmitter is not active
assign w_UART_Line = w_TX_Active ? w_TX_Serial : 1'b1;
always
```

```
#(c_CLOCK_PERIOD_NS/2) r_Clock <= !r_Clock;
34
// Main Testing:
initial
begin
// Tell UART to send a command (exercise TX)

@(posedge r_Clock);
@(posedge r_Clock);
r_TX_DV <= 1'b1;
r_TX_Byte <= 8'h3F;
@(posedge r_Clock);
r_TX_DV <= 1'b0;
end
endmodule
```

Simulation Waveform:



Synthesize RTL Schematic:



Result: