BANGLADESH UNIVERSITY OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING

EEE 468 (Jan 2024)

VLSI Design Laboratory

Final Project Report

G2 (Project Group 6)

16 BIT SERIAL IN SERIAL OUT FIFO SHIFT REGISTER

Course Instructors:

Nafis Sadik, Lecturer

Rafid Hassan Palash, Part-Time Lecturer

Signature of Instructor: _____

Academic Honesty Statement:

IMPORTANT! Please carefully read and sign the Academic Honesty Statement, below. Type the student ID and name, and put your signature. *You will not receive credit for this project experiment unless this statement is signed in the presence of your lab instructor.*

"In signing this statement, We hereby certify that the work on this project is our own and that we have not copied the work of any other students (past or present), and cited all relevant sources while completing this project. We understand that if we fail to honor this agreement, We will each receive a score of ZERO for this project and be subject to failure of this course."

Signature: _____ **Full Name: Md. Ali Ahmed** **Student ID: 1906143**	**Signature:** _____ **Full Name: Md. Nayem Hasan** **Student ID: 1906175**
Signature: _____ **Full Name: Tanvir Rahman** **Student ID: 1906176**	**Signature:** _____ **Full Name: Samiul Hossain** **Student ID: 1906179**

Introduction

The 16-bit Serial-In Serial-Out (SISO) FIFO Shift Register is a fundamental component in digital systems, widely used in communication systems, data processing units, and memory management. This project implements a versatile shift register that allows both parallel and serial data handling, enabling efficient data storage and transfer operations.

In this project, the shift register can perform three key operations:

1. Parallel Load: The entire 16-bit data can be loaded simultaneously when the ‘Load’ signal is asserted.

2. Left Shift: When the ‘Left’ control signal is high, the data shifts left by one position, with new bits entering from the least significant bit (LSB).

3. Right Shift: When the ‘Left’ control signal is low, the data shifts right by one position, with new bits entering from the most significant bit (MSB).

The clock (‘Clk’) ensures synchronous operation, and the serial input (‘Din’) provides flexibility in serial data transfer. The serial output (‘Dout’) is used to transmit the last shifted bit, making the design suitable for FIFO (First-In-First-Out) operations.

16-bit SISO FIFO Block Diagram

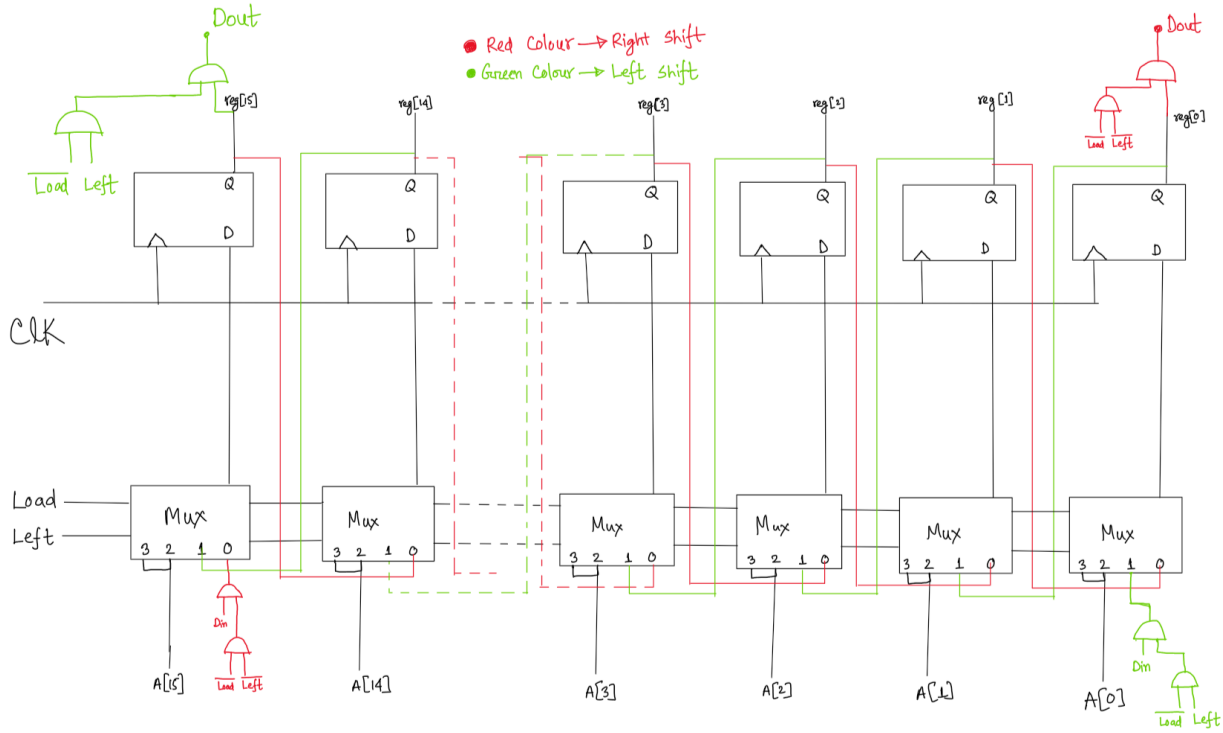


Figure: 16-bit SISO FIFO Shift Register

Selection lines of muxes are as below:

{Load, Left}	Perform
00	Right Shift
01	Left Shift
1X	Parallel Load

Figure: MUX's selection lines and their operation

Working Principle

The 16-bit SISO FIFO Shift Register operates synchronously based on the clock signal ('Clk') and is controlled by three primary inputs: 'Load', 'Left', and 'Din'. Its operation can be categorized into the following modes:

1. Parallel Load Mode

When the 'Load' signal is high ('Load = 1'), the shift register enters the parallel load mode. Here, the 16-bit data from the input 'A[15:0]' is loaded directly into the internal register in one clock cycle. This mode is particularly useful for initializing the register with a specific data pattern or for quickly updating its content without serial shifting.

2. Left Shift Mode

When the 'Load' signal is low ('Load = 0') and the 'Left' signal is high ('Left = 1'), the shift register operates in left shift mode. Here, The internal data shifts one position to the left. The new bit enters from the 'Din' input at the least significant bit (LSB) position. The most significant bit (MSB) is shifted out through the serial output 'Dout'.

3. Right Shift Mode

When the 'Load' signal is low ('Load = 0') and the 'Left' signal is low ('Left = 0'), the shift register operates in right shift mode. In this mode, the internal data shifts one position to the right. The new bit enters from the 'Din' input at the most significant bit (MSB) position. The least significant bit (LSB) is shifted out through the serial output 'Dout'. This mode complements the left shift operation, providing flexibility in data manipulation.

Data Storage and Transfer

The internal register acts as a temporary storage medium, holding up to 16 bits of data at any given time. As a FIFO device, the data stored in the register is shifted in and out in a sequential manner, maintaining the order of bits.

Clock Synchronization

The entire operation is synchronized with the clock signal (‘Clk’), ensuring that changes in the register occur only at the rising edge of the clock. This synchronous behavior eliminates race conditions and provides predictable timing for data shifts and loads.

Output Behavior

The serial output (‘Dout’) continuously provides the bit being shifted out, either from the MSB (in right shift mode) or LSB (in left shift mode). This output can be monitored for data extraction or interfaced with other digital components in larger systems.

The 16-bit SISO FIFO Shift Register is an essential digital module capable of parallel loading and bidirectional shifting. Its synchronous design and versatile operation make it suitable for numerous real-world applications, including data buffering, serialization, and communication. By understanding its working principle and leveraging its features, engineers and designers can integrate this module into larger, more complex systems efficiently.

RTL Code-

Design Code: (design.sv)

```
module SISO_FIFO_ShiftRegister (
    input         Clk,
    input         Load,
    input         Left,
    input         Din,
    input  [15:0] A,
    output    reg    Dout,
   output reg[15:0] register
);


    always @(posedge Clk)
      begin
        if (Load)
          begin
            register <= A;
          end

        else
          begin
            if (Left)
              begin
                Dout <= (register[15]);
```

```
                register <= {register[14:0], Din};
              end

            else
              begin
                Dout <= register[0];
                register <= {Din, register[15:1]};
              end
          end
       end

endmodule
```

Directed Testbench:

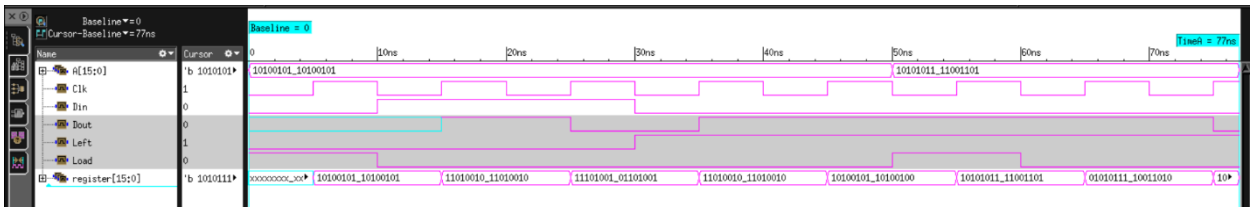
```
module tb_SISO_FIFO_ShiftRegister;

    // Testbench signals
    reg Clk;
    reg Load;
    reg Left;
    reg Din;
    reg [15:0] A;
    wire Dout;
    wire [15:0] register;


    SISO_FIFO_ShiftRegister DUT (
        .Clk(Clk),
        .Load(Load),
        .Left(Left),
        .Din(Din),
        .A(A),
       .Dout(Dout),
       .register(register)
    );

    initial begin
        Clk = 0;
        forever #5 Clk = ~Clk;
    end
initial begin

      $monitor("Time: %t | Clk: %b | A: %b | Load: %d | Left: %d | Din: %d | Dout: %d
| Register: %b",$time,Clk, A, Load, Left, Din, Dout, register);

        Load = 0; Left = 0; Din = 0; A = 16'h0000;

        Load = 1; A = 16'hA5A5;
        #10;
```

```
        Load = 0;

        // Right shift test
        Left = 0; Din = 1;

      repeat (2) begin
            #10;
        end

        // Left shift test

        Left = 1; Din = 0;


      repeat (2) begin
            #10;

        end
       Load = 1; A = 16'hABCD;
        #10;

        Load = 0;

       // Again Left shift test

        Left = 1; Din = 0;
      repeat (2) begin
            #10;
        end
    end

    initial begin
        $dumpfile("dump.vcd");
        $dumpvars;
        #77;

      $display("All tests completed.");
        $finish;
    end

endmodule
```

Output from Cadence

```
[vlsi30@CadenceServer3 exp_1]$ ncsim tb_SISO_FIFO_ShiftRegister
ncsim(64): 15.10-s015: (c) Copyright 1995-2016 Cadence Design Systems, Inc.
ncsim> run
ncsim: *W,DVEXACC2: some objects excluded from $dumpvars due to -access -R.
           File: ./tb_SISO_FIFO_ShiftRegister.v, line = 74, pos = 16
         Scope: tb_SISO_FIFO_ShiftRegister
          Time: 0 FS + 0

Time:                    0 | Clk: 0 | A: 1010010110100101 | Load: 1 | Left: 0 | Din: 0 | Dout: x | Register: xxxxxxxxxxxxxxxx
Time:                    5 | Clk: 1 | A: 1010010110100101 | Load: 1 | Left: 0 | Din: 0 | Dout: x | Register: 1010010110100101
Time:                   10 | Clk: 0 | A: 1010010110100101 | Load: 0 | Left: 0 | Din: 1 | Dout: x | Register: 1010010110100101
Time:                   15 | Clk: 1 | A: 1010010110100101 | Load: 0 | Left: 0 | Din: 1 | Dout: 1 | Register: 1101001011010010
Time:                   20 | Clk: 0 | A: 1010010110100101 | Load: 0 | Left: 0 | Din: 1 | Dout: 1 | Register: 1101001011010010
Time:                   25 | Clk: 1 | A: 1010010110100101 | Load: 0 | Left: 0 | Din: 1 | Dout: 0 | Register: 1110100101101001
Time:                   30 | Clk: 0 | A: 1010010110100101 | Load: 0 | Left: 1 | Din: 0 | Dout: 0 | Register: 1110100101101001
Time:                   35 | Clk: 1 | A: 1010010110100101 | Load: 0 | Left: 1 | Din: 0 | Dout: 1 | Register: 1101001011010010
Time:                   40 | Clk: 0 | A: 1010010110100101 | Load: 0 | Left: 1 | Din: 0 | Dout: 1 | Register: 1101001011010010
Time:                   45 | Clk: 1 | A: 1010010110100101 | Load: 0 | Left: 1 | Din: 0 | Dout: 1 | Register: 1010010110100100
Time:                   50 | Clk: 0 | A: 1010101111001101 | Load: 1 | Left: 1 | Din: 0 | Dout: 1 | Register: 1010010110100100
Time:                   55 | Clk: 1 | A: 1010101111001101 | Load: 1 | Left: 1 | Din: 0 | Dout: 1 | Register: 1010101111001101
Time:                   60 | Clk: 0 | A: 1010101111001101 | Load: 0 | Left: 1 | Din: 0 | Dout: 1 | Register: 1010101111001101
Time:                   65 | Clk: 1 | A: 1010101111001101 | Load: 0 | Left: 1 | Din: 0 | Dout: 1 | Register: 0101011110011010
Time:                   70 | Clk: 0 | A: 1010101111001101 | Load: 0 | Left: 1 | Din: 0 | Dout: 1 | Register: 0101011110011010
Time:                   75 | Clk: 1 | A: 1010101111001101 | Load: 0 | Left: 1 | Din: 0 | Dout: 0 | Register: 1010111100110100
All tests completed.
Simulation complete via $finish(1) at time 77 NS + 0
./tb_SISO_FIFO_ShiftRegister.v:77        $finish;
ncsim> exit
[vlsi30@CadenceServer3 exp_1]$
```

Waveform:



Self-checking Testbench:

```
module tb_SISO_FIFO_ShiftRegister;

    reg Clk;
    reg Load;
    reg Left;
    reg Din;
    reg [15:0] A;
    wire Dout;
    wire [15:0] register;
    reg [15:0] expected_register;
    reg expected_Dout;
    SISO_FIFO_ShiftRegister DUT (
        .Clk(Clk),
        .Load(Load),
        .Left(Left),
        .Din(Din),
        .A(A),
       .Dout(Dout),
       .register(register)
    );
```

```
initial begin
        Clk = 0;
        forever #5 Clk = ~Clk;
    end
    initial begin
        Load = 0; Left = 0; Din = 0; A = 16'h0000;
        expected_register = 16'h0000; expected_Dout = 0;
             //Parallel Load test
        Load = 1; A = 16'hA5A5;
        expected_register = 16'hA5A5;
        #10;
               Load = 0;
      $display("At time %0t | Load: %b, Left: %b, Din: %b",$time, DUT.Load, DUT.Left,
DUT.Din);
        if (DUT.register !== expected_register)
            $display("ERROR: Register mismatch at time %0t | Expected: %b, Got: %b",
$time, expected_register, DUT.register);
        else
            $display("CORRECT: Register match at time %0t | Expected: %b, Got: %b",
$time, expected_register, DUT.register);
        // Right shift test
        Left = 0; Din = 1;
            repeat (3) begin
            #10;
            expected_Dout = expected_register[0];
            expected_register = {Din, expected_register[15:1]};
             $display("At time %0t | Load: %b, Left: %b, Din: %b",$time, DUT.Load,
DUT.Left, DUT.Din);
            if (DUT.register !== expected_register)
                $display("ERROR: Register mismatch at time %0t | Expected: %b, Got:
%b", $time, expected_register, DUT.register);
            else
                $display("CORRECT: Register match at time %0t | Expected: %b, Got:
%b", $time, expected_register, DUT.register);
            if (Dout !== expected_Dout)
                $display("ERROR: Dout mismatch at time %0t | Expected: %b, Got: %b",
$time, expected_Dout, Dout);
            else
                $display("CORRECT: Dout match at time %0t | Expected: %b, Got: %b",
$time, expected_Dout, Dout);
        end
        // Left shift test

        Left = 1; Din = 0;
        repeat (3) begin
            #10;
            expected_Dout = expected_register[15];
            expected_register = {expected_register[14:0], Din};
          $display("At time %0t | Load: %b, Left: %b, Din: %b",$time, DUT.Load,
DUT.Left, DUT.Din);
```

```
            if (DUT.register !== expected_register)
                $display("ERROR: Register mismatch at time %0t | Expected: %b, Got:
%b", $time, expected_register, DUT.register);
            else
                $display("CORRECT: Register match at time %0t | Expected: %b, Got:
%b", $time, expected_register, DUT.register);
            if (Dout !== expected_Dout)
                $display("ERROR: Dout mismatch at time %0t | Expected: %b, Got: %b",
$time, expected_Dout, Dout);
            else
                $display("CORRECT: Dout match at time %0t | Expected: %b, Got: %b",
$time, expected_Dout, Dout);
            end
        Load = 1; A = 16'hABCD;
         expected_register = 16'hABCD;
         #10;
                Load = 0;
       $display("At time %0t | Load: %b, Left: %b, Din: %b",$time, DUT.Load, DUT.Left,
DUT.Din);
         if (DUT.register !== expected_register)
             $display("ERROR: Register mismatch at time %0t | Expected: %b, Got: %b",
$time, expected_register, DUT.register);
         else
             $display("CORRECT: Register match at time %0t | Expected: %b, Got: %b",
$time, expected_register, DUT.register);

        // Again Left shift test
         Left = 1; Din = 0;
         repeat (3) begin
             #10;
             expected_Dout = expected_register[15];
             expected_register = {expected_register[14:0], Din};
           $display("At time %0t | Load: %b, Left: %b, Din: %b",$time, DUT.Load,
DUT.Left, DUT.Din);

            if (DUT.register !== expected_register)
                $display("ERROR: Register mismatch at time %0t | Expected: %b, Got:
%b", $time, expected_register, DUT.register);
            else
                $display("CORRECT: Register match at time %0t | Expected: %b, Got:
%b", $time, expected_register, DUT.register);

            if (Dout !== expected_Dout)
                $display("ERROR: Dout mismatch at time %0t | Expected: %b, Got: %b",
$time, expected_Dout, Dout);
            else
                $display("CORRECT: Dout match at time %0t | Expected: %b, Got: %b",
$time, expected_Dout, Dout);
         end
         $display("All tests completed.");
    end
```

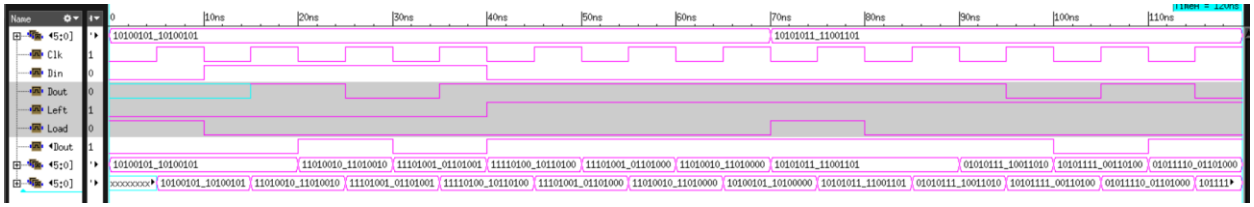
```
    initial begin
        $dumpfile("dump.vcd");
        $dumpvars;
        #120;
        $finish;
    end
endmodule
```

Output from Cadence

```
[vlsi30@CadenceServer3 exp_1]$ ncsim tb_SISO_FIFO_ShiftRegister
ncsim(64): 15.10-s015: (c) Copyright 1995-2016 Cadence Design Systems, Inc.
ncsim> run
ncsim: *W,DVEXACC2: some objects excluded from $dumpvars due to -access -R.
            File: ./tb_SISO_FIFO_ShiftRegister.v, line = 123, pos = 16
           Scope: tb_SISO_FIFO_ShiftRegister
            Time: 0 FS + 0

At time 10 | Load: 1, Left: 0, Din: 0
CORRECT: Register match at time 10 | Expected: 1010010110100101, Got: 1010010110100101
At time 20 | Load: 0, Left: 0, Din: 1
CORRECT: Register match at time 20 | Expected: 1101001011010010, Got: 1101001011010010
CORRECT: Dout match at time 20 | Expected: 1, Got: 1
At time 30 | Load: 0, Left: 0, Din: 1
CORRECT: Register match at time 30 | Expected: 1110100101101001, Got: 1110100101101001
CORRECT: Dout match at time 30 | Expected: 0, Got: 0
At time 40 | Load: 0, Left: 0, Din: 1
CORRECT: Register match at time 40 | Expected: 1111010010110100, Got: 1111010010110100
CORRECT: Dout match at time 40 | Expected: 1, Got: 1
At time 50 | Load: 0, Left: 1, Din: 0
CORRECT: Register match at time 50 | Expected: 1110100101101000, Got: 1110100101101000
CORRECT: Dout match at time 50 | Expected: 1, Got: 1
At time 60 | Load: 0, Left: 1, Din: 0
CORRECT: Register match at time 60 | Expected: 1101001011010000, Got: 1101001011010000
CORRECT: Dout match at time 60 | Expected: 1, Got: 1
At time 70 | Load: 0, Left: 1, Din: 0
CORRECT: Register match at time 70 | Expected: 1010010110100000, Got: 1010010110100000
CORRECT: Dout match at time 70 | Expected: 1, Got: 1
At time 80 | Load: 1, Left: 1, Din: 0
CORRECT: Register match at time 80 | Expected: 1010101111001101, Got: 1010101111001101
At time 90 | Load: 0, Left: 1, Din: 0
CORRECT: Register match at time 90 | Expected: 0101011110011010, Got: 0101011110011010
CORRECT: Dout match at time 90 | Expected: 1, Got: 1
At time 100 | Load: 0, Left: 1, Din: 0
CORRECT: Register match at time 100 | Expected: 1010111100110100, Got: 1010111100110100
CORRECT: Dout match at time 100 | Expected: 0, Got: 0
At time 110 | Load: 0, Left: 1, Din: 0
CORRECT: Register match at time 110 | Expected: 0101111001101000, Got: 0101111001101000
CORRECT: Dout match at time 110 | Expected: 1, Got: 1
All tests completed.
Simulation complete via $finish(1) at time 120 NS + 0
./tb_SISO_FIFO_ShiftRegister.v:125         $finish;
ncsim> exit
[vlsi30@CadenceServer3 exp_1]$
```

Figure: Output from terminal in cadence.

Waveform:



Here, last waveform is the output register & second last is the expected register.

Layered Verification of the Digital Sub-system

testbench.sv

```
‘include "testcase01.sv"
‘include "interface.sv"

module testbench;
  bit Clk;
  initial begin
    Clk = 0;
    forever #5 Clk =~Clk;
  end

  int count=30;

  SISO_FIFO_if SF(Clk);

  test test01(count,SF);

  initial begin
    $dumpfile("dump.vcd");
    $dumpvars;
  end

  SISO_FIFO_ShiftRegister DUT (
    .Clk(SF.Clk),
    .Load(SF.Load),
    .Left(SF.Left),
    .Din(SF.Din),
    .A(SF.A),
    .Dout(SF.Dout),
    .register(SF.register)
  );

endmodule
```

testcase01.sv

```
‘include "environment.sv"

program test(input int count, SISO_FIFO_if SF);
  environment env;

  class testcase01 extends transaction;
    constraint c_s {
       A inside {[0:65535]};

      Load inside {[0:1]};
      Left inside {[0:1]};
      Din inside {[0:1]};
    }
  endclass:testcase01

  initial begin
    testcase01 testcase01handle;
    testcase01handle=new();
    env=new(SF);

    env.gen.g_trans=testcase01handle;

    env.main(count);
  end

endprogram:test
```

driver.sv

```
class driver;
  mailbox gen2driv, driv2sb;
  virtual SISO_FIFO_if.DRIVER SISO_FIFO_if;
  transaction d_trans;
  event driven;

  function new(mailbox gen2driv,mailbox driv2sb , virtual SISO_FIFO_if.DRIVER
SISO_FIFO_if, event driven);
    this.gen2driv=gen2driv;
    this.SISO_FIFO_if=SISO_FIFO_if;
    this.driven=driven;
    this.driv2sb=driv2sb;
  endfunction

  task main(input int count);
    repeat(count) begin
      d_trans=new();
      gen2driv.get(d_trans);
```

```
      @(SISO_FIFO_if.driver_cb);

      SISO_FIFO_if.driver_cb.Load <= d_trans.Load;
      SISO_FIFO_if.driver_cb.Left <= d_trans.Left;
      SISO_FIFO_if.driver_cb.Din <= d_trans.Din;
       SISO_FIFO_if.driver_cb.A <= d_trans.A;

      driv2sb.put(d_trans);
      -> driven;
    end

  endtask:main

endclass:driver
```

environment.sv

```
‘include "generator.sv"
‘include "driver.sv"
‘include "monitor.sv"
‘include "scoreboard.sv"

class environment;
  mailbox gen2driv;
  mailbox driv2sb;
  mailbox mon2sb;

  generator gen;
  driver drv;
  monitor mon;
  scoreboard scb;

  event driven;

  virtual SISO_FIFO_if SF;

  function new(virtual SISO_FIFO_if SF);

    this.SF=SF;
    gen2driv=new();
    driv2sb=new();
    mon2sb=new();
    gen=new(gen2driv);
    drv=new(gen2driv,driv2sb,SF.DRIVER,driven);
    mon=new(mon2sb,SF.MONITOR,driven);
    scb=new(driv2sb,mon2sb);
  endfunction

  task main(input int count);
    fork  gen.main(count);
          drv.main(count);
```

```
          mon.main(count);
          scb.main(count);
    join
    $finish;
  endtask:main

endclass:environment
```

generator.sv

```
‘include "transaction.sv"

class generator;
  mailbox gen2driv;
  transaction g_trans;

  function new(mailbox gen2driv);
    this.gen2driv=gen2driv;
  endfunction

  task main(input int count);
    repeat(count) begin
      g_trans=new();
      g_trans=new g_trans;
      assert(g_trans.randomize());
      gen2driv.put(g_trans);
    end
  endtask:main

endclass:generator
```

interface.sv

```
interface SISO_FIFO_if(input Clk);
  logic Load;
  logic Left;
  logic Din;
  logic [15:0] A;
  logic Dout;
  logic [15:0] register;

  clocking driver_cb @(negedge Clk);
    default input #1 output #1;
    output Load,Left,Din,A;
  endclocking

  clocking mon_cb @(negedge Clk);
    default input #1 output #1;
    input Load,Left,Din,A;
    input Dout,register;
```

```
  endclocking

  modport DRIVER (clocking driver_cb, input Clk);
    modport MONITOR (clocking mon_cb, input Clk);
endinterface
```

monitor.sv

```
class monitor;
  mailbox mon2sb;
  virtual SISO_FIFO_if.MONITOR SF;
  transaction m_trans;
  event driven;

  function new(mailbox mon2sb, virtual SISO_FIFO_if.MONITOR SF, event driven);
    this.mon2sb=mon2sb;
    this.SF=SF;
    this.driven=driven;
  endfunction

  task main(input int count);
    @(driven);

    repeat(count) begin
       @(SF.mon_cb);
      m_trans=new();
       @(posedge SF.Clk);
      m_trans.Load=SF.mon_cb.Load;
      m_trans.Left=SF.mon_cb.Left;
      m_trans.Din=SF.mon_cb.Din;
      m_trans.A=SF.mon_cb.A;
      m_trans.Dout=SF.mon_cb.Dout;
       m_trans.register=SF.mon_cb.register;
      mon2sb.put(m_trans);
    end
  endtask:main

endclass:monitor
```

scoreboard.sv

```
class scoreboard;
  mailbox driv2sb;
  mailbox mon2sb;

  transaction d_trans;
  transaction m_trans;
   bit [15:0] temp;

  real in_type [8]='{default: 64'b0};
```

```
  real Pass [8]='{default: 64'b0};
  real Fail [8]='{default: 64'b0};
  real in_per [8],fail_per [8], pass_per [8];
  logic [2:0] in_concat;


  function new(mailbox driv2sb, mon2sb);
    this.driv2sb=driv2sb;
    this.mon2sb=mon2sb;
  endfunction

  task main(input int count);
    $display("------------------Scoreboard Test Starts--------------------");
    repeat(count) begin
      m_trans=new();
      mon2sb.get(m_trans);
      report();

      in_concat = {d_trans.Load,d_trans.Left,d_trans.Din};
      if (m_trans.register != d_trans.register)
        begin
          $display("time=%d  A = %b   Failed : Load=%d Left=%d  Din=%d Expected_Dout=%d
Resulted_Dout=%d                                                    Expected_register=%b
Resulted_register=%b",$time,d_trans.A,d_trans.Load,d_trans.Left,d_trans.Din,d_trans.D
out,m_trans.Dout,d_trans.register,m_trans.register);

        if(in_concat==3'b000)
            Fail[0]=Fail[0]+1;
          else if(in_concat==3'b001)
            Fail[1]=Fail[1]+1;
          else if(in_concat==3'b010)
            Fail[2]=Fail[2]+1;
          else if(in_concat==3'b011)
            Fail[3]=Fail[3]+1;
          else if(in_concat==3'b100)
            Fail[4]=Fail[4]+1;
          else if(in_concat==3'b101)
            Fail[5]=Fail[5]+1;
          else if(in_concat==3'b110)
            Fail[6]=Fail[6]+1;
          else if(in_concat==3'b111)
            Fail[7]=Fail[7]+1;


        end
        else
          begin
            $display("time=%d    A  =  %b      Passed  :  Load=%d  Left=%d    Din=%d
Expected_Dout=%x                      Resulted_Dout=%d           Expected_register=%b
Resulted_register=%b",$time,d_trans.A,d_trans.Load,d_trans.Left,d_trans.Din,d_trans.D
out,m_trans.Dout,d_trans.register,m_trans.register);
```

```
      if(in_concat==3'b000)
        Pass[0]=Pass[0]+1;
      else if(in_concat==3'b001)
        Pass[1]=Pass[1]+1;
      else if(in_concat==3'b010)
        Pass[2]=Pass[2]+1;
      else if(in_concat==3'b011)
        Pass[3]=Pass[3]+1;
      else if(in_concat==3'b100)
        Pass[4]=Pass[4]+1;
      else if(in_concat==3'b101)
        Pass[5]=Pass[5]+1;
      else if(in_concat==3'b110)
        Pass[6]=Pass[6]+1;
      else if(in_concat==3'b111)
        Pass[7]=Pass[7]+1;


      end

  end

  in_per[0]=(in_type[0]*100)/count;
  in_per[1]=(in_type[1]*100)/count;
  in_per[2]=(in_type[2]*100)/count;
  in_per[3]=(in_type[3]*100)/count;
  in_per[4]=(in_type[4]*100)/count;
  in_per[5]=(in_type[5]*100)/count;
  in_per[6]=(in_type[6]*100)/count;
  in_per[7]=(in_type[7]*100)/count;

  pass_per[0]=(Pass[0]*100)/in_type[0];
  pass_per[1]=(Pass[1]*100)/in_type[1];
  pass_per[2]=(Pass[2]*100)/in_type[2];
  pass_per[3]=(Pass[3]*100)/in_type[3];
  pass_per[4]=(Pass[4]*100)/in_type[4];
  pass_per[5]=(Pass[5]*100)/in_type[5];
  pass_per[6]=(Pass[6]*100)/in_type[6];
  pass_per[7]=(Pass[7]*100)/in_type[7];

  fail_per[0]=(Fail[0]*100)/in_type[0];
  fail_per[1]=(Fail[1]*100)/in_type[1];
  fail_per[2]=(Fail[2]*100)/in_type[2];
  fail_per[3]=(Fail[3]*100)/in_type[3];
  fail_per[4]=(Fail[4]*100)/in_type[4];
  fail_per[5]=(Fail[5]*100)/in_type[5];
  fail_per[6]=(Fail[6]*100)/in_type[6];
  fail_per[7]=(Fail[7]*100)/in_type[7];
```

```
     $display("\n\n------------------Displaying  Covergae  Results--------------------
\n\n");

    $display("Case-Type 1: {Load,Left,Din}='000'=%0.0f cases with percentage=%f.Pass
rate=%f,Fail rate=%f",in_type[0],in_per[0],pass_per[0],fail_per[0]);
    $display("Case-Type 2: {Load,Left,Din}='001'=%0.0f cases with percentage=%f.Pass
rate=%f,Fail rate=%f",in_type[1],in_per[1],pass_per[1],fail_per[1]);
    $display("Case-Type 3: {Load,Left,Din}='010'=%0.0f cases with percentage=%f.Pass
rate=%f,Fail rate=%f",in_type[2],in_per[2],pass_per[2],fail_per[2]);
    $display("Case-Type 4: {Load,Left,Din}='011'=%0.0f cases with percentage=%f.Pass
rate=%f,Fail rate=%f",in_type[3],in_per[3],pass_per[3],fail_per[3]);
    $display("Case-Type 5: {Load,Left,Din}='100'=%0.0f cases with percentage=%f.Pass
rate=%f,Fail rate=%f",in_type[4],in_per[4],pass_per[4],fail_per[4]);
    $display("Case-Type 6: {Load,Left,Din}='101'=%0.0f cases with percentage=%f.Pass
rate=%f,Fail rate=%f",in_type[5],in_per[5],pass_per[5],fail_per[5]);
    $display("Case-Type 7: {Load,Left,Din}='110'=%0.0f cases with percentage=%f.Pass
rate=%f,Fail rate=%f",in_type[6],in_per[6],pass_per[6],fail_per[6]);
    $display("Case-Type 8: {Load,Left,Din}='111'=%0.0f cases with percentage=%f.Pass
rate=%f,Fail rate=%f",in_type[7],in_per[7],pass_per[7],fail_per[7]);



    $display("\n\n------------------Scoreboard Test Ends--------------------\n");
  endtask:main

  task report();

      d_trans=new();
    driv2sb.get(d_trans);

      if (d_trans.Load)
          begin
            d_trans.register = d_trans.A;
            temp = d_trans.register;
           end
        else
          begin

            if (d_trans.Left)
              begin
                d_trans.Dout = temp[15];
                d_trans.register = {temp[14:0], d_trans.Din};

               temp = d_trans.register;
              end
            else
              begin

                d_trans.Dout = temp[0];
                d_trans.register = {d_trans.Din,temp[15:1]};
               temp = d_trans.register;
```

```
            end
          end
    in_concat={d_trans.Load,d_trans.Left,d_trans.Din};

    if(in_concat==3'b000)
      in_type[0]=in_type[0]+1;
    else if(in_concat==3'b001)
      in_type[1]=in_type[1]+1;
    else if(in_concat==3'b010)
      in_type[2]=in_type[2]+1;
    else if(in_concat==3'b011)
      in_type[3]=in_type[3]+1;
    else if(in_concat==3'b100)
      in_type[4]=in_type[4]+1;
    else if(in_concat==3'b101)
      in_type[5]=in_type[5]+1;
    else if(in_concat==3'b110)
      in_type[6]=in_type[6]+1;
    else if(in_concat==3'b111)
      in_type[7]=in_type[7]+1;

  endtask:report
endclass:scoreboard
```

transaction.sv

```
class transaction;
  rand bit [15:0] A;
  rand bit Load;
  rand bit Left;
  rand bit Din;
  bit [15:0] register;
  bit Dout;
endclass:transaction
```

run.sh

irun +access+r -disable_sem2009 -svseed random -incdir ../design/ -incdir ../testbench/ -incdir ../testcase/ -sysv ../design/SISO_FIFO_ShiftRegister.v ../testbench/testbench.sv

After Run command ($source run.sh)

When Count = 30 (‘Count’ is the number of testcase)

```
ncsim> run
------------------Scoreboard Test Starts--------------------
time=                  25  A = 1001011111001001   Passed : Load=1 Left=0  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=1001011111001001  Resulted_register=1001011111001001
time=                  35  A = 1011101101111001   Passed : Load=0 Left=0  Din=1 Expected_Dout=1  Resulted_Dout=1 Expected_register=1100101111100100  Resulted_register=1100101111100100
time=                  45  A = 1001100111001110   Passed : Load=1 Left=0  Din=1 Expected_Dout=0  Resulted_Dout=1 Expected_register=1001100111001110  Resulted_register=1001100111001110
time=                  55  A = 0010011000111011   Passed : Load=0 Left=0  Din=0 Expected_Dout=0  Resulted_Dout=0 Expected_register=0100110011100111  Resulted_register=0100110011100111
time=                  65  A = 1111010001010100   Passed : Load=0 Left=0  Din=1 Expected_Dout=1  Resulted_Dout=1 Expected_register=1010011001110011  Resulted_register=1010011001110011
time=                  75  A = 1011010000101000   Passed : Load=0 Left=1  Din=0 Expected_Dout=1  Resulted_Dout=1 Expected_register=0100110011100110  Resulted_register=0100110011100110
time=                  85  A = 0100001001101110   Passed : Load=0 Left=0  Din=0 Expected_Dout=0  Resulted_Dout=0 Expected_register=0010011001110011  Resulted_register=0010011001110011
time=                  95  A = 0000000011111101   Passed : Load=0 Left=1  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=0100110011100111  Resulted_register=0100110011100111
time=                 105  A = 1011111010100101   Passed : Load=0 Left=1  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=1001100111001111  Resulted_register=1001100111001111
time=                 115  A = 1000010101111110   Passed : Load=0 Left=0  Din=1 Expected_Dout=1  Resulted_Dout=1 Expected_register=1100110011100111  Resulted_register=1100110011100111
time=                 125  A = 0010100000000110   Passed : Load=1 Left=0  Din=1 Expected_Dout=0  Resulted_Dout=1 Expected_register=0010100000000110  Resulted_register=0010100000000110
time=                 135  A = 1101011011110001   Passed : Load=0 Left=1  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=0101000000001101  Resulted_register=0101000000001101
time=                 145  A = 0000100011010110   Passed : Load=1 Left=0  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=0000100011010110  Resulted_register=0000100011010110
time=                 155  A = 1001100111111000   Passed : Load=0 Left=0  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=1000010001101011  Resulted_register=1000010001101011
time=                 165  A = 1010111001011111   Passed : Load=0 Left=0  Din=1 Expected_Dout=1  Resulted_Dout=1 Expected_register=1100001000110101  Resulted_register=1100001000110101
time=                 175  A = 1111011101101010   Passed : Load=0 Left=0  Din=1 Expected_Dout=1  Resulted_Dout=1 Expected_register=1110000100011010  Resulted_register=1110000100011010
time=                 185  A = 1010111100101010   Passed : Load=0 Left=1  Din=0 Expected_Dout=1  Resulted_Dout=1 Expected_register=1100001000110100  Resulted_register=1100001000110100
time=                 195  A = 1011001011001111   Passed : Load=0 Left=1  Din=1 Expected_Dout=1  Resulted_Dout=1 Expected_register=1000010001101001  Resulted_register=1000010001101001
time=                 205  A = 0000100100001010   Passed : Load=1 Left=0  Din=0 Expected_Dout=0  Resulted_Dout=0 Expected_register=0000100100001010  Resulted_register=0000100100001010
time=                 215  A = 0010000110101101   Passed : Load=0 Left=0  Din=0 Expected_Dout=0  Resulted_Dout=0 Expected_register=0000010010000101  Resulted_register=0000010010000101
time=                 225  A = 0111100101110110   Passed : Load=0 Left=1  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=0000100100001011  Resulted_register=0000100100001011
time=                 235  A = 0100100001100011   Passed : Load=1 Left=0  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=0100100001100011  Resulted_register=0100100001100011
time=                 245  A = 1110101000111001   Passed : Load=1 Left=0  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=1110101000111001  Resulted_register=1110101000111001
time=                 255  A = 1101100100110000   Passed : Load=1 Left=1  Din=0 Expected_Dout=0  Resulted_Dout=0 Expected_register=1101100100110000  Resulted_register=1101100100110000
time=                 265  A = 1001001100001111   Passed : Load=0 Left=0  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=1110110010011000  Resulted_register=1110110010011000
time=                 275  A = 0000000010001111   Passed : Load=1 Left=0  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=0000000010001111  Resulted_register=0000000010001111
time=                 285  A = 1111101000101101   Passed : Load=0 Left=0  Din=1 Expected_Dout=1  Resulted_Dout=1 Expected_register=1000000001000111  Resulted_register=1000000001000111
time=                 295  A = 0111000110100011   Passed : Load=0 Left=0  Din=0 Expected_Dout=1  Resulted_Dout=1 Expected_register=0100000000100011  Resulted_register=0100000000100011
time=                 305  A = 0011111011100001   Passed : Load=0 Left=1  Din=0 Expected_Dout=0  Resulted_Dout=0 Expected_register=1000000001000110  Resulted_register=1000000001000110
time=                 315  A = 0001011010110100   Passed : Load=1 Left=0  Din=0 Expected_Dout=0  Resulted_Dout=0 Expected_register=0001011010110100  Resulted_register=0001011010110100

------------------Displaying Covergae Results--------------------

Case-Type 1: {Load,Left,Din}='000'=4 cases with percentage=13.333333.Pass rate=100.000000,Fail rate=0.000000
Case-Type 2: {Load,Left,Din}='001'=8 cases with percentage=26.666667.Pass rate=100.000000,Fail rate=0.000000
Case-Type 3: {Load,Left,Din}='010'=3 cases with percentage=10.000000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 4: {Load,Left,Din}='011'=5 cases with percentage=16.666667.Pass rate=100.000000,Fail rate=0.000000
Case-Type 5: {Load,Left,Din}='100'=2 cases with percentage=6.666667.Pass rate=100.000000,Fail rate=0.000000
Case-Type 6: {Load,Left,Din}='101'=7 cases with percentage=23.333333.Pass rate=100.000000,Fail rate=0.000000
Case-Type 7: {Load,Left,Din}='110'=1 cases with percentage=3.333333.Pass rate=100.000000,Fail rate=0.000000
Case-Type 8: {Load,Left,Din}='111'=0 cases with percentage=0.000000.Pass rate=-nan,Fail rate=-nan

------------------Scoreboard Test Ends--------------------
```

Here, there is no test case generated for Load = 1, Left = 1, Din = 1.

When Count = 100

```
time=                 895  A = 1000100111011000   Passed : Load=1 Left=1  Din=0 Expected_Dout=0  Resulted_Dout=1 Expected_register=1000100111011000  Resulted_register=1000100111011000
time=                 905  A = 1011010110011001   Passed : Load=1 Left=0  Din=1 Expected_Dout=0  Resulted_Dout=1 Expected_register=1011010110011001  Resulted_register=1011010110011001
time=                 915  A = 0000010100011111   Passed : Load=1 Left=1  Din=0 Expected_Dout=0  Resulted_Dout=1 Expected_register=0000010100011111  Resulted_register=0000010100011111
time=                 925  A = 0101010011101011   Passed : Load=1 Left=0  Din=0 Expected_Dout=0  Resulted_Dout=1 Expected_register=0101010011101011  Resulted_register=0101010011101011
time=                 935  A = 0111000101010001   Passed : Load=0 Left=1  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=1010100111010111  Resulted_register=1010100111010111
time=                 945  A = 0110111011011001   Passed : Load=1 Left=0  Din=0 Expected_Dout=0  Resulted_Dout=0 Expected_register=0110111011011001  Resulted_register=0110111011011001
time=                 955  A = 1110111100100100   Passed : Load=1 Left=0  Din=0 Expected_Dout=0  Resulted_Dout=0 Expected_register=1110111100100100  Resulted_register=1110111100100100
time=                 965  A = 0100010010011110   Passed : Load=1 Left=0  Din=0 Expected_Dout=0  Resulted_Dout=0 Expected_register=0100010010011110  Resulted_register=0100010010011110
time=                 975  A = 0000101110011100   Passed : Load=1 Left=1  Din=0 Expected_Dout=0  Resulted_Dout=0 Expected_register=0000101110011100  Resulted_register=0000101110011100
time=                 985  A = 0101100001110100   Passed : Load=0 Left=1  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=0001011100111001  Resulted_register=0001011100111001
time=                 995  A = 1100110000011000   Passed : Load=0 Left=1  Din=0 Expected_Dout=0  Resulted_Dout=0 Expected_register=0010111001110010  Resulted_register=0010111001110010
time=                1005  A = 0101000110000011   Passed : Load=0 Left=0  Din=0 Expected_Dout=0  Resulted_Dout=0 Expected_register=0001011100111001  Resulted_register=0001011100111001
time=                1015  A = 1111101111011111   Passed : Load=1 Left=0  Din=0 Expected_Dout=0  Resulted_Dout=0 Expected_register=1111101111011111  Resulted_register=1111101111011111

------------------Displaying Covergae Results--------------------

Case-Type 1: {Load,Left,Din}='000'=14 cases with percentage=14.000000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 2: {Load,Left,Din}='001'=9 cases with percentage=9.000000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 3: {Load,Left,Din}='010'=13 cases with percentage=13.000000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 4: {Load,Left,Din}='011'=12 cases with percentage=12.000000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 5: {Load,Left,Din}='100'=14 cases with percentage=14.000000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 6: {Load,Left,Din}='101'=17 cases with percentage=17.000000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 7: {Load,Left,Din}='110'=14 cases with percentage=14.000000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 8: {Load,Left,Din}='111'=7 cases with percentage=7.000000.Pass rate=100.000000,Fail rate=0.000000

------------------Scoreboard Test Ends--------------------
```

When Count = 500

```
time=              4935  A = 1001100001111011   Passed : Load=1 Left=1  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=1001100001111011  Resulted_register=1001100001111011
time=              4945  A = 1101010011010010   Passed : Load=1 Left=0  Din=0 Expected_Dout=0  Resulted_Dout=0 Expected_register=1101010011010010  Resulted_register=1101010011010010
time=              4955  A = 0000101001000111   Passed : Load=1 Left=1  Din=0 Expected_Dout=0  Resulted_Dout=0 Expected_register=0000101001000111  Resulted_register=0000101001000111
time=              4965  A = 0011010001101000   Passed : Load=1 Left=1  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=0011010001101000  Resulted_register=0011010001101000
time=              4975  A = 1000110111011011   Passed : Load=0 Left=1  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=0110100011010001  Resulted_register=0110100011010001
time=              4985  A = 1111010101100111   Passed : Load=1 Left=1  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=1111010101100111  Resulted_register=1111010101100111
time=              4995  A = 0001100100010101   Passed : Load=1 Left=0  Din=0 Expected_Dout=0  Resulted_Dout=0 Expected_register=0001100100010101  Resulted_register=0001100100010101
time=              5005  A = 1010100011011100   Passed : Load=0 Left=1  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=0011001000101011  Resulted_register=0011001000101011
time=              5015  A = 1110011110001100   Passed : Load=0 Left=0  Din=1 Expected_Dout=1  Resulted_Dout=1 Expected_register=1001100100010101  Resulted_register=1001100100010101


------------------Displaying Covergae Results--------------------


Case-Type 1: {Load,Left,Din}='000'=60 cases with percentage=12.000000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 2: {Load,Left,Din}='001'=47 cases with percentage=9.400000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 3: {Load,Left,Din}='010'=70 cases with percentage=14.000000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 4: {Load,Left,Din}='011'=59 cases with percentage=11.800000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 5: {Load,Left,Din}='100'=59 cases with percentage=11.800000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 6: {Load,Left,Din}='101'=67 cases with percentage=13.400000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 7: {Load,Left,Din}='110'=72 cases with percentage=14.400000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 8: {Load,Left,Din}='111'=66 cases with percentage=13.200000.Pass rate=100.000000,Fail rate=0.000000


------------------Scoreboard Test Ends--------------------
```

When Count = 1500

```
time=             14945  A = 0110000110100011   Passed : Load=0 Left=0  Din=0 Expected_Dout=0  Resulted_Dout=0 Expected_register=0101110001110010  Resulted_register=0101110001110010
time=             14955  A = 1100011110110110   Passed : Load=1 Left=1  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=1100011110110110  Resulted_register=1100011110110110
time=             14965  A = 0101111001100011   Passed : Load=0 Left=1  Din=1 Expected_Dout=1  Resulted_Dout=1 Expected_register=1000111101101101  Resulted_register=1000111101101101
time=             14975  A = 1000110001011101   Passed : Load=1 Left=1  Din=0 Expected_Dout=0  Resulted_Dout=1 Expected_register=1000110001011101  Resulted_register=1000110001011101
time=             14985  A = 0100111011110000   Passed : Load=0 Left=1  Din=1 Expected_Dout=1  Resulted_Dout=1 Expected_register=0001100010111011  Resulted_register=0001100010111011
time=             14995  A = 1100011001000001   Passed : Load=0 Left=0  Din=0 Expected_Dout=1  Resulted_Dout=1 Expected_register=0000110001011101  Resulted_register=0000110001011101
time=             15005  A = 1010111110111110   Passed : Load=0 Left=0  Din=0 Expected_Dout=1  Resulted_Dout=1 Expected_register=0000011000101110  Resulted_register=0000011000101110
time=             15015  A = 0110110000101111   Passed : Load=1 Left=1  Din=1 Expected_Dout=0  Resulted_Dout=1 Expected_register=0110110000101111  Resulted_register=0110110000101111


------------------Displaying Covergae Results--------------------


Case-Type 1: {Load,Left,Din}='000'=201 cases with percentage=13.400000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 2: {Load,Left,Din}='001'=199 cases with percentage=13.266667.Pass rate=100.000000,Fail rate=0.000000
Case-Type 3: {Load,Left,Din}='010'=222 cases with percentage=14.800000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 4: {Load,Left,Din}='011'=206 cases with percentage=13.733333.Pass rate=100.000000,Fail rate=0.000000
Case-Type 5: {Load,Left,Din}='100'=175 cases with percentage=11.666667.Pass rate=100.000000,Fail rate=0.000000
Case-Type 6: {Load,Left,Din}='101'=177 cases with percentage=11.800000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 7: {Load,Left,Din}='110'=157 cases with percentage=10.466667.Pass rate=100.000000,Fail rate=0.000000
Case-Type 8: {Load,Left,Din}='111'=163 cases with percentage=10.866667.Pass rate=100.000000,Fail rate=0.000000


------------------Scoreboard Test Ends--------------------
```

When Count = 4000

```
time=             39965  A = 0000001100010010   Passed : Load=0 Left=1  Din=0 Expected_Dout=1  Resulted_Dout=1 Expected_register=1101111001111110  Resulted_register=1101111001111110
time=             39975  A = 0000010101100010   Passed : Load=0 Left=0  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=1110111100111111  Resulted_register=1110111100111111
time=             39985  A = 0100001000010100   Passed : Load=0 Left=0  Din=0 Expected_Dout=1  Resulted_Dout=1 Expected_register=0111011110011111  Resulted_register=0111011110011111
time=             39995  A = 0010100110110000   Passed : Load=0 Left=1  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=1110111100111111  Resulted_register=1110111100111111
time=             40005  A = 0011110100011010   Passed : Load=0 Left=1  Din=1 Expected_Dout=1  Resulted_Dout=1 Expected_register=1101111001111111  Resulted_register=1101111001111111
time=             40015  A = 0110100100100110   Passed : Load=1 Left=1  Din=1 Expected_Dout=0  Resulted_Dout=1 Expected_register=0110100100100110  Resulted_register=0110100100100110


------------------Displaying Covergae Results--------------------


Case-Type 1: {Load,Left,Din}='000'=501 cases with percentage=12.525000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 2: {Load,Left,Din}='001'=503 cases with percentage=12.575000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 3: {Load,Left,Din}='010'=500 cases with percentage=12.500000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 4: {Load,Left,Din}='011'=535 cases with percentage=13.375000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 5: {Load,Left,Din}='100'=493 cases with percentage=12.325000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 6: {Load,Left,Din}='101'=503 cases with percentage=12.575000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 7: {Load,Left,Din}='110'=489 cases with percentage=12.225000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 8: {Load,Left,Din}='111'=476 cases with percentage=11.900000.Pass rate=100.000000,Fail rate=0.000000


------------------Scoreboard Test Ends--------------------
```

When Count = 10000

```
time=             99985  A = 1000001000010101   Passed : Load=0 Left=0  Din=0 Expected_Dout=1  Resulted_Dout=1 Expected_register=0110110110111001  Resulted_register=0110110110111001
time=             99995  A = 0111011010001111   Passed : Load=1 Left=1  Din=1 Expected_Dout=0  Resulted_Dout=1 Expected_register=0111011010001111  Resulted_register=0111011010001111
time=            100005  A = 0110100101101000   Passed : Load=0 Left=1  Din=0 Expected_Dout=0  Resulted_Dout=0 Expected_register=1110110100011110  Resulted_register=1110110100011110
time=            100015  A = 0110100100001111   Passed : Load=0 Left=1  Din=1 Expected_Dout=1  Resulted_Dout=1 Expected_register=1101101000111101  Resulted_register=1101101000111101


------------------Displaying Covergae Results--------------------


Case-Type 1: {Load,Left,Din}='000'=1314 cases with percentage=13.140000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 2: {Load,Left,Din}='001'=1213 cases with percentage=12.130000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 3: {Load,Left,Din}='010'=1250 cases with percentage=12.500000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 4: {Load,Left,Din}='011'=1222 cases with percentage=12.220000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 5: {Load,Left,Din}='100'=1238 cases with percentage=12.380000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 6: {Load,Left,Din}='101'=1230 cases with percentage=12.300000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 7: {Load,Left,Din}='110'=1275 cases with percentage=12.750000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 8: {Load,Left,Din}='111'=1258 cases with percentage=12.580000.Pass rate=100.000000,Fail rate=0.000000


------------------Scoreboard Test Ends--------------------
```

When Count = 50000

```
time=            499975  A = 1001110000111101   Passed : Load=0 Left=1  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=0111001010011011  Resulted_register=0111001010011011
time=            499985  A = 0011100100010110   Passed : Load=0 Left=1  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=1110010100110111  Resulted_register=1110010100110111
time=            499995  A = 1100101001110110   Passed : Load=1 Left=0  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=1100101001110110  Resulted_register=1100101001110110
time=            500005  A = 1000101101010010   Passed : Load=1 Left=0  Din=1 Expected_Dout=0  Resulted_Dout=0 Expected_register=1000101101010010  Resulted_register=1000101101010010
time=            500015  A = 1110110101101100   Passed : Load=0 Left=1  Din=1 Expected_Dout=1  Resulted_Dout=1 Expected_register=0001011010100101  Resulted_register=0001011010100101

------------------Displaying Covergae Results--------------------

Case-Type 1: {Load,Left,Din}='000'=6327 cases with percentage=12.654000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 2: {Load,Left,Din}='001'=6163 cases with percentage=12.326000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 3: {Load,Left,Din}='010'=6340 cases with percentage=12.680000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 4: {Load,Left,Din}='011'=6182 cases with percentage=12.364000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 5: {Load,Left,Din}='100'=6202 cases with percentage=12.404000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 6: {Load,Left,Din}='101'=6153 cases with percentage=12.306000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 7: {Load,Left,Din}='110'=6305 cases with percentage=12.610000.Pass rate=100.000000,Fail rate=0.000000
Case-Type 8: {Load,Left,Din}='111'=6328 cases with percentage=12.656000.Pass rate=100.000000,Fail rate=0.000000

------------------Scoreboard Test Ends--------------------
```

Comparison of Coverage by Varying the Number of Testcase (Count):

Testcase (Load, Left, Din)	Percentage						
	Count=30	Count=100	Count=500	Count=1500	Count=4000	Count=10000	Count=50000
000	13.3333	14.0000	12.0000	13.4000	12.5250	13.1400	12.6540
001	26.6667	9.0000	9.4000	13.2667	12.5750	12.1300	12.3260
010	10.0000	13.0000	14.0000	14.8000	12.5000	12.5000	12.6800
011	16.6667	12.0000	11.8000	13.7333	13.3750	12.2200	12.3640
100	6.6667	14.0000	11.8000	11.6667	12.3250	12.3800	12.4040
101	23.3333	17.0000	13.4000	11.8000	12.5750	12.3000	12.3060
110	3.3333	14.0000	14.4000	10.4667	12.2250	12.7500	12.6100
111	0.000	7.0000	13.2000	10.8667	11.9000	12.5800	12.6560

Table 1: Coverage with respect to number of testcase

Synthesis with Cadence Genus (TM) Synthesis Solution

At this stage we have behavioral code of SISO-FIFO-Shift Register (SISO_FIFO_ShiftRegister.v). For synthesis purpose, we also need the SISO_FIFO_ShiftRegister.tcl which is a script to convert behavioral code to synthesized netlist (structural code) and SISO_FIFO_ShiftRegister.sdc which is a synopsys design constraints file. So, we have edited the ALU codes correspondingly.

First of all, for power, performance and area optimization, we have varied parameters like input delay, output delay , clock frequency ,etc. Since ,there are no reset in our project, setup and hold time variation was not needed. Thus we have found the optimum clock freq, then input and output delay. We have mainly analyzed that when the power and area are minimum, that point should be considered as optimum point.

We have simulated the tcl script for clock frequency, input delay and output delay. sdc was being generated by tcl automatically.

Clock_frequency.tcl

```
# Run Genus in Legacy UI if Genus is invoked with Common UI
::legacy::set_attribute common_ui false /
if {[file exists /proc/cpuinfo]} {
    sh grep "model name" /proc/cpuinfo
    sh grep "cpu MHz" /proc/cpuinfo
}
puts "Hostname : [info hostname]"

##############################################################################
### Preset global variables and attributes
##############################################################################

# Set setup times to iterate over
set clock_periods {5 10 20 50 100}
set DESIGN SISO_FIFO_ShiftRegister
set SYN_EFF low
set MAP_EFF low
set OPT_EFF low

# Directory of PDK
set pdk_dir /home/cad/VLSI2Lab/Digital/library/
set_attribute init_lib_search_path $pdk_dir

# Set synthesizing effort for each synthesis stage
set_attribute syn_generic_effort $SYN_EFF
set_attribute syn_map_effort $MAP_EFF
set_attribute syn_opt_effort $OPT_EFF
set_attribute library "slow_vdd1v0_basicCells.lib"
set_dont_use [get_lib_cells CLK*]
set_dont_use [get_lib_cells SDFF*]
set_dont_use [get_lib_cells DLY*]
set_dont_use [get_lib_cells HOLD*]

# Load Design
read_hdl "${DESIGN}.v"
elaborate $DESIGN
#check_design -unresolved

# Iterate over each setup time
foreach clock_period $clock_periods {
    # Create a new SDC file for each setup time
    set sdc_file "${DESIGN}_clock_${clock_period}_low.sdc"

    # Write the new SDC file
    set output [open $sdc_file "w"]
    #puts $output "set setup_time 3"
    puts $output "create_clock -period ${clock_period} -waveform {0 6} -name func_clk
[get_ports Clk]"
    puts $output "set_input_delay 0.4 -clock [get_clocks func_clk] {Load Left Din A}"
```

```
    puts $output "set_output_delay 0.6 -clock [get_clocks func_clk] {Dout register}"
    #puts $output "set_multicycle_path -setup 3 -from [get_ports rst_n]"
    #puts $output "set_multicycle_path -hold 2 -from [get_ports rst_n]"
    close $output

    # Read the newly created SDC file
    read_sdc $sdc_file

    # Run synthesis
    syn_generic
    puts "Runtime & Memory after 'syn_generic' for clock period ${clock_period} ns"
    time_info GENERIC

    # Report power and generate output files
    report_power -verbose -detail >>
reports/power_report_clock_${clock_period}_low.txt
     report_area  >> reports/area__report_clock_${clock_period}_low.txt
    synthesize -to_mapped
    write -mapped > SISO_FIFO_ShiftRegister_synth_clock_${clock_period}_low.v
}

# Write final script
write_script > script
```

Then we have got power, area versus clock frequency

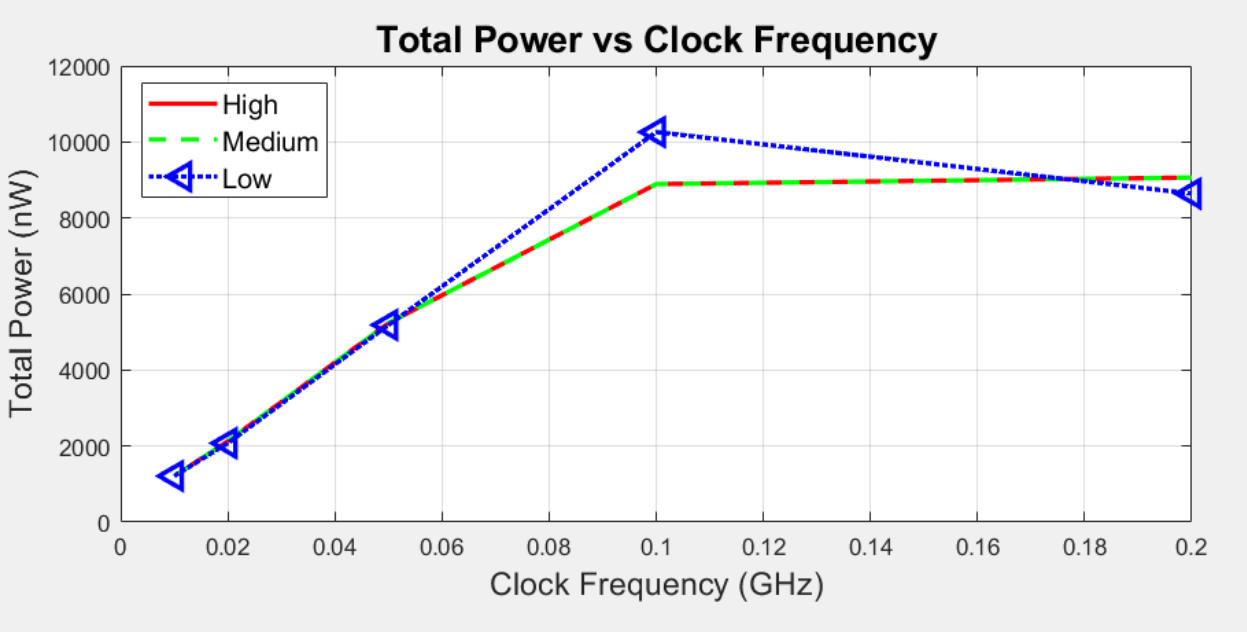


Figure: Total power VS clock frequency for high, medium and low effort

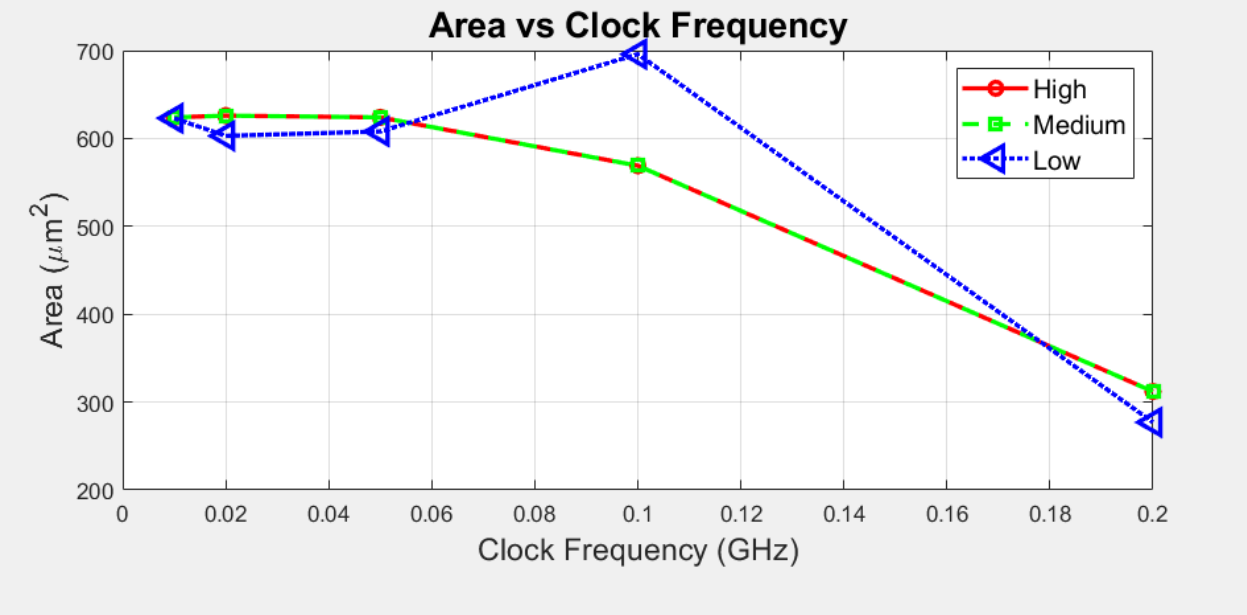


Figure: Total area VS clock frequency for high, medium and low effort

From these, we have decided to set the clock frequency at 0.1 GHz since, if frequency gets increased then area is going to be minimized but power increases. Similarly, when frequency is lower, then power is lessened but area increases. So, there are trade-offs between area and power. Hence, we have come to the mid frequency point to get the optimum values. For now, the effort is high or medium, as for low effort, area and power are more.

Now we will look for optimum delays. We have collected data from cadence text file and then plotted them by Origin.

Outputdelay.tcl

```
# Run Genus in Legacy UI if Genus is invoked with Common UI
::legacy::set_attribute common_ui false /
if {[file exists /proc/cpuinfo]} {
    sh grep "model name" /proc/cpuinfo
    sh grep "cpu MHz" /proc/cpuinfo
}
puts "Hostname : [info hostname]"

##########################################################################
### Preset global variables and attributes
##########################################################################

# Set output delays to iterate over
set out_delays {0.1 0.2 0.4 0.6 0.8 1 1.3 1.5}
set DESIGN SISO_FIFO_ShiftRegister
set SYN_EFF low
set MAP_EFF low
set OPT_EFF low
```

```
# Directory of PDK
set pdk_dir /home/cad/VLSI2Lab/Digital/library/
set_attribute init_lib_search_path $pdk_dir

# Set synthesizing effort for each synthesis stage
set_attribute syn_generic_effort $SYN_EFF
set_attribute syn_map_effort $MAP_EFF
set_attribute syn_opt_effort $OPT_EFF
set_attribute library "slow_vdd1v0_basicCells.lib"

# Exclude certain library cells
set_dont_use [get_lib_cells CLK*]
set_dont_use [get_lib_cells SDFF*]
set_dont_use [get_lib_cells DLY*]
set_dont_use [get_lib_cells HOLD*]

# Load Design
read_hdl "${DESIGN}.v"
elaborate $DESIGN

# Iterate over each output delay
foreach out_delay $out_delays {
    # Create a new SDC file for each output delay
    set sdc_file "${DESIGN}_outDelay_${out_delay}_low.sdc"

    # Write the new SDC file
    set output [open $sdc_file "w"]
    puts $output "create_clock -period 10 -waveform {0 6} -name func_clk [get_ports
Clk]"
    puts $output "set_input_delay 0.4 -clock [get_clocks func_clk] {Load Left Din A}"
    puts $output "set_output_delay ${out_delay} -clock [get_clocks func_clk] {Dout
register}"
    close $output

    # Read the newly created SDC file
    read_sdc $sdc_file

    # Run synthesis
    syn_generic
    puts "Runtime & Memory after 'syn_generic' for output delay ${out_delay} ns"
    time_info GENERIC

    # Report power and generate output files
    report_power -verbose -detail >>
reports/power_report_outDelay_${out_delay}_low.txt
    report_area >> reports/area_report_outDelay_${out_delay}_low.txt
    synthesize -to_mapped
    write -mapped > SISO_FIFO_ShiftRegister_synth_outDelay_${out_delay}_low.v
}
# Write final script
write_script > script
```

Thus, we have got leakage, internal, net and dynamic power plot with respect to output delay variation.

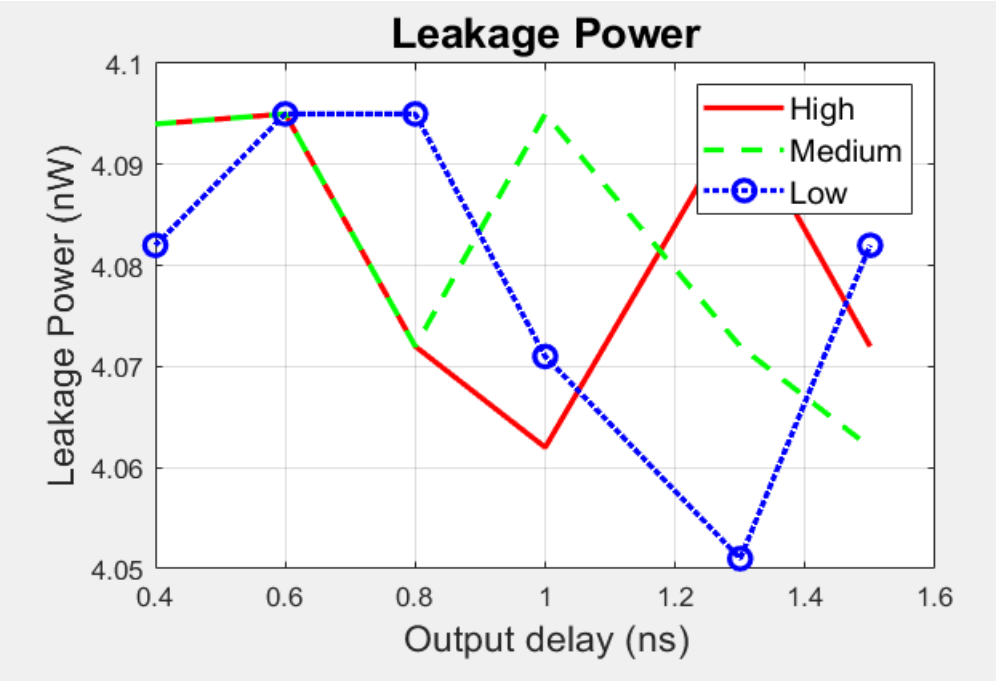


Figure: Leakage power VS output delay

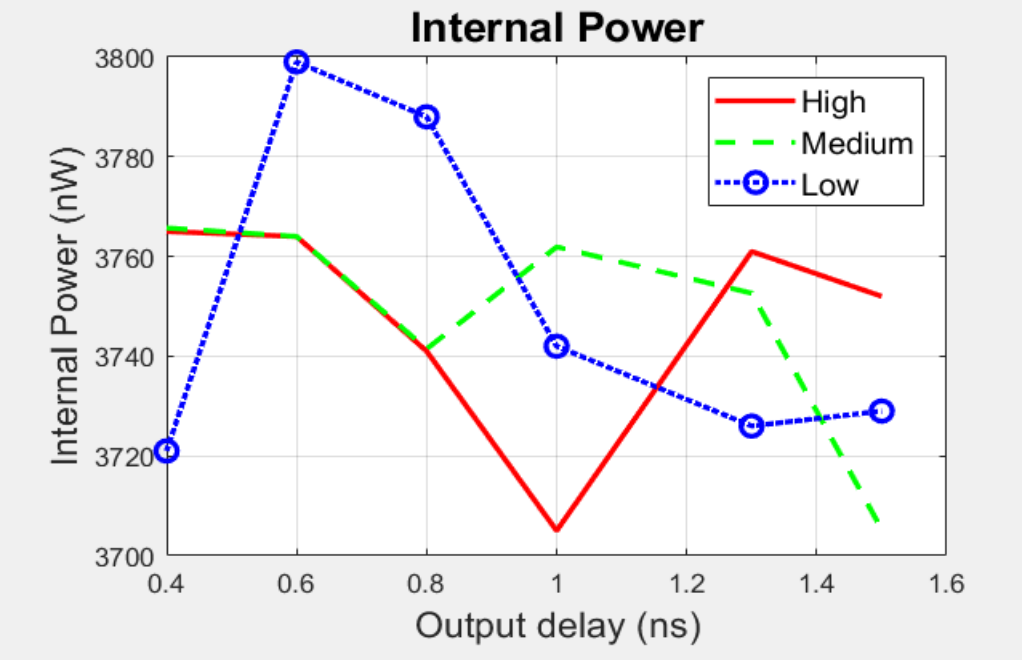


Figure: Internal power VS output delay

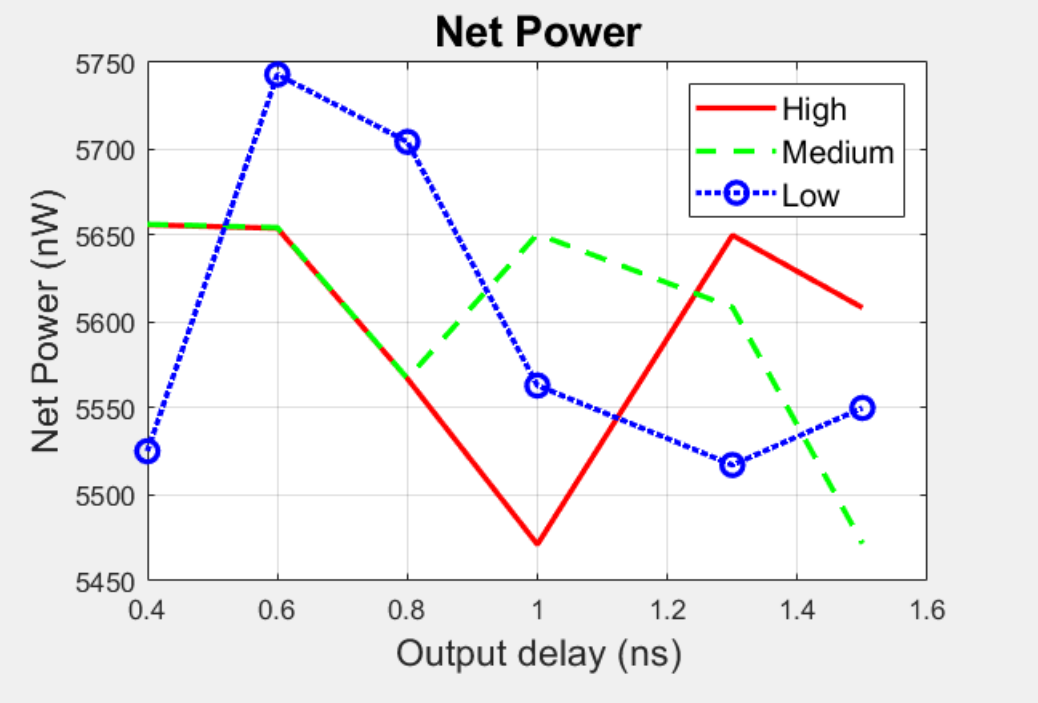


Figure: Net power VS output delay

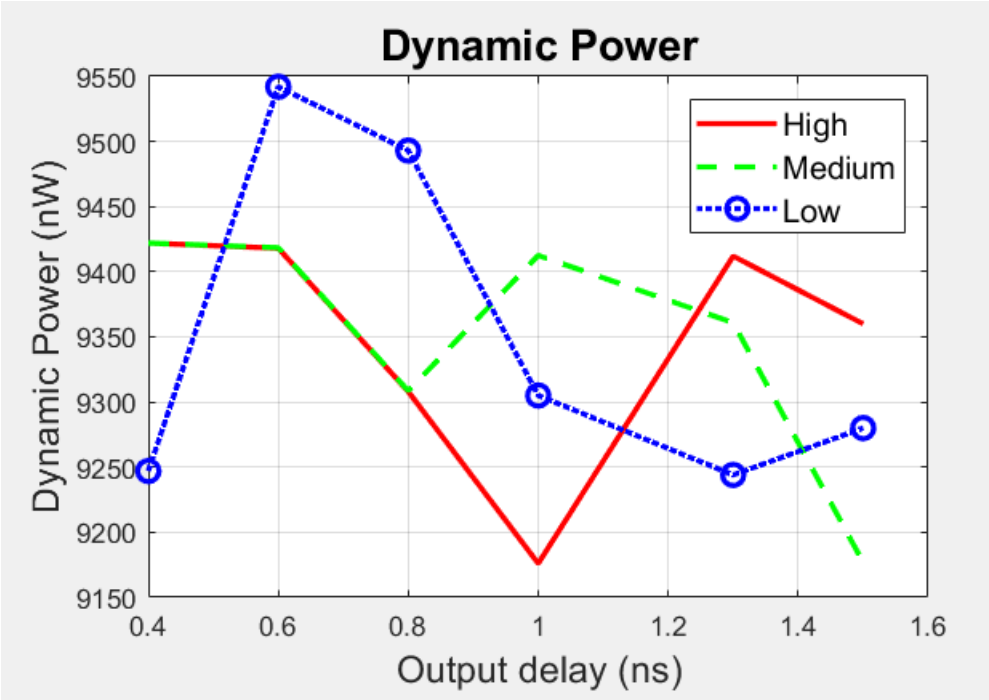


Figure: Dynamic power VS output delay

Inputdelay.tcl

```
# Run Genus in Legacy UI if Genus is invoked with Common UI
::legacy::set_attribute common_ui false /
if {[file exists /proc/cpuinfo]} {
    sh grep "model name" /proc/cpuinfo
    sh grep "cpu MHz" /proc/cpuinfo
}
puts "Hostname : [info hostname]"

##############################################################################
### Preset global variables and attributes
##############################################################################

# Set input delays to iterate over
set inp_delays {0.1 0.2 0.4 0.6 0.8 1 1.3 1.5}
set DESIGN SISO_FIFO_ShiftRegister
set SYN_EFF high
set MAP_EFF high
set OPT_EFF high

# Directory of PDK
set pdk_dir /home/cad/VLSI2Lab/Digital/library/
set_attribute init_lib_search_path $pdk_dir

# Set synthesizing effort for each synthesis stage
set_attribute syn_generic_effort $SYN_EFF
set_attribute syn_map_effort $MAP_EFF
set_attribute syn_opt_effort $OPT_EFF
set_attribute library "slow_vdd1v0_basicCells.lib"

# Exclude certain library cells
set_dont_use [get_lib_cells CLK*]
set_dont_use [get_lib_cells SDFF*]
set_dont_use [get_lib_cells DLY*]
set_dont_use [get_lib_cells HOLD*]

# Load Design
read_hdl "${DESIGN}.v"
elaborate $DESIGN

# Iterate over each input delay
foreach inp_delay $inp_delays {
    # Create a new SDC file for each input delay
    set sdc_file "${DESIGN}_inpDelay_${inp_delay}_high.sdc"

    # Write the new SDC file
    set output [open $sdc_file "w"]
    puts $output "create_clock -period 10 -waveform {0 6} -name func_clk [get_ports
Clk]"
```

```
    puts $output "set_input_delay ${inp_delay} -clock [get_clocks func_clk] {Load
Left Din A}"
    puts $output "set_output_delay 0.6 -clock [get_clocks func_clk] {Dout register}"
    close $output

    # Read the newly created SDC file
    read_sdc $sdc_file

    # Run synthesis
    syn_generic
    puts "Runtime & Memory after 'syn_generic' for input delay ${inp_delay} ns"
    time_info GENERIC

    # Report power and generate output files
    report_power -verbose -detail >>
reports/power_report_inpDelay_${inp_delay}_high.txt
    report_area >> reports/area_report_inpDelay_${inp_delay}_high.txt
    synthesize -to_mapped
    write -mapped > SISO_FIFO_ShiftRegister_synth_inpDelay_${inp_delay}_high.v
}

# Write final script
write_script > script
```

Thus, we have got leakage, internal, net and dynamic power plot with respect to input delay variation.

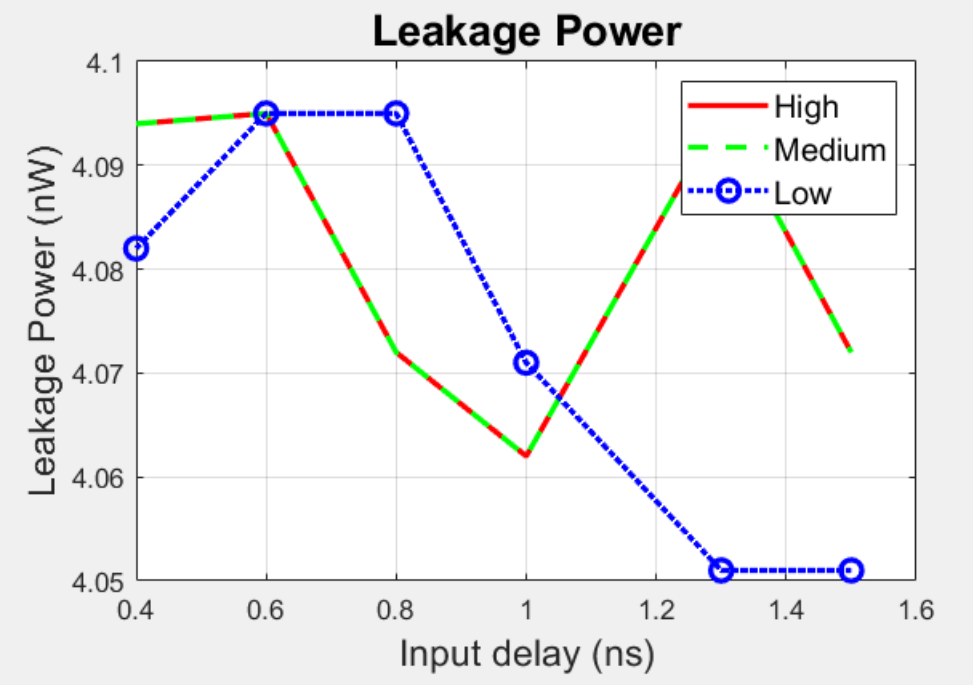


Figure: Leakage power VS input delay

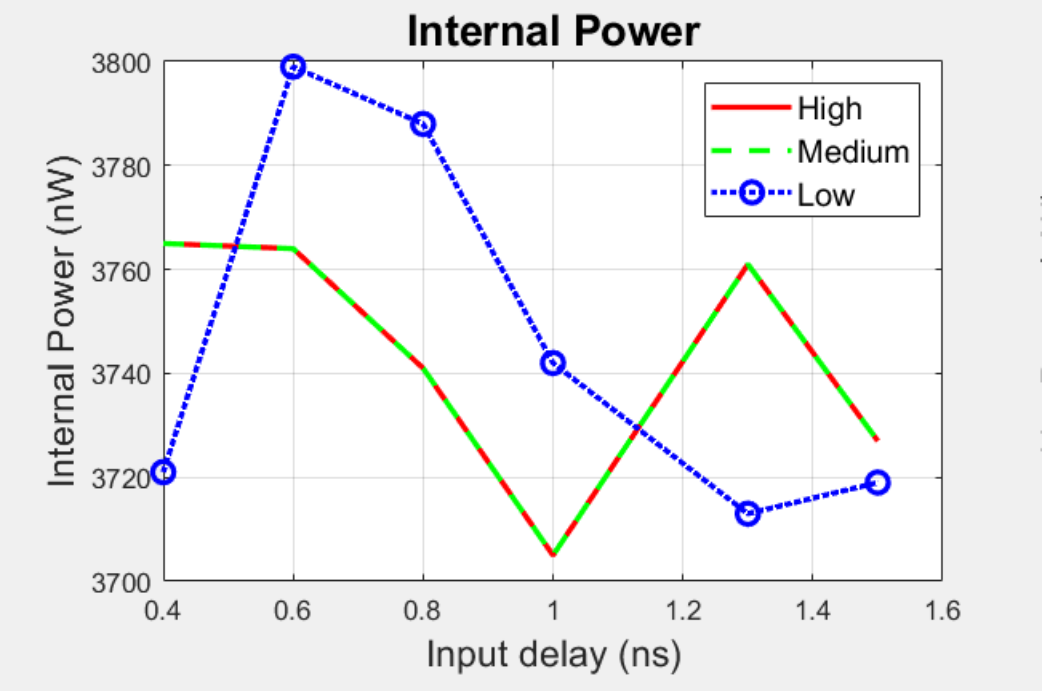


Figure: Internal power VS input delay

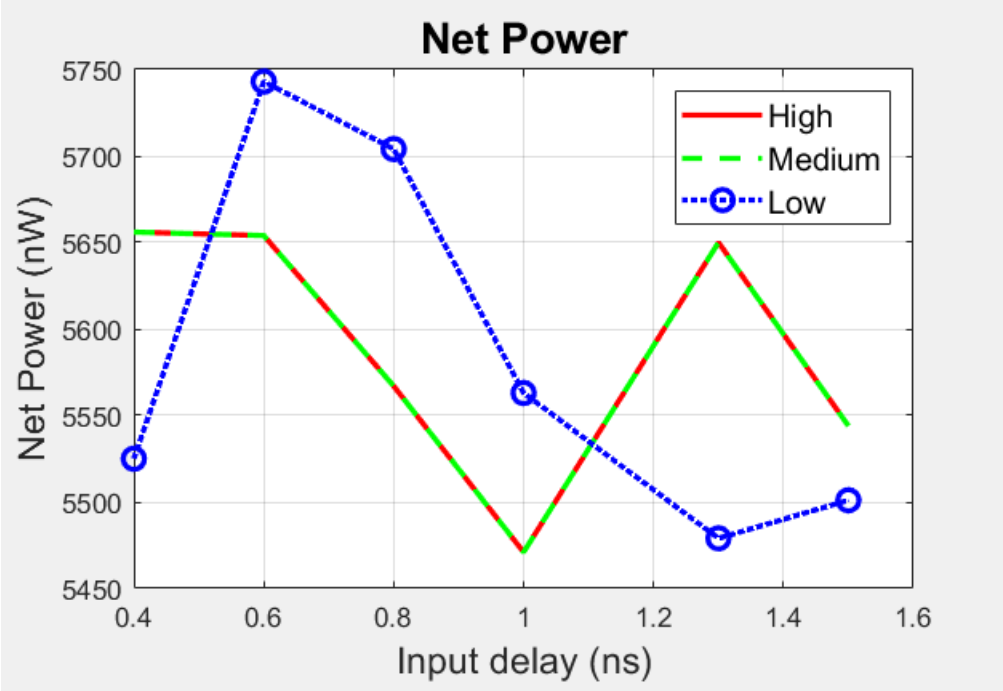


Figure: Net power VS input delay

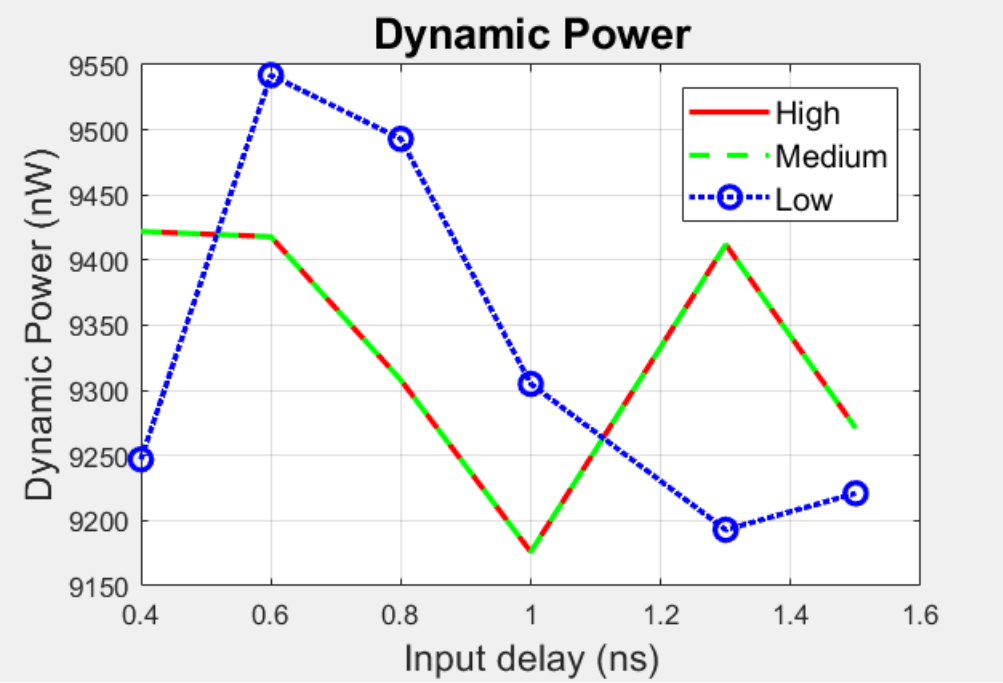


Figure: Dynamic power VS input delay

Optimization:

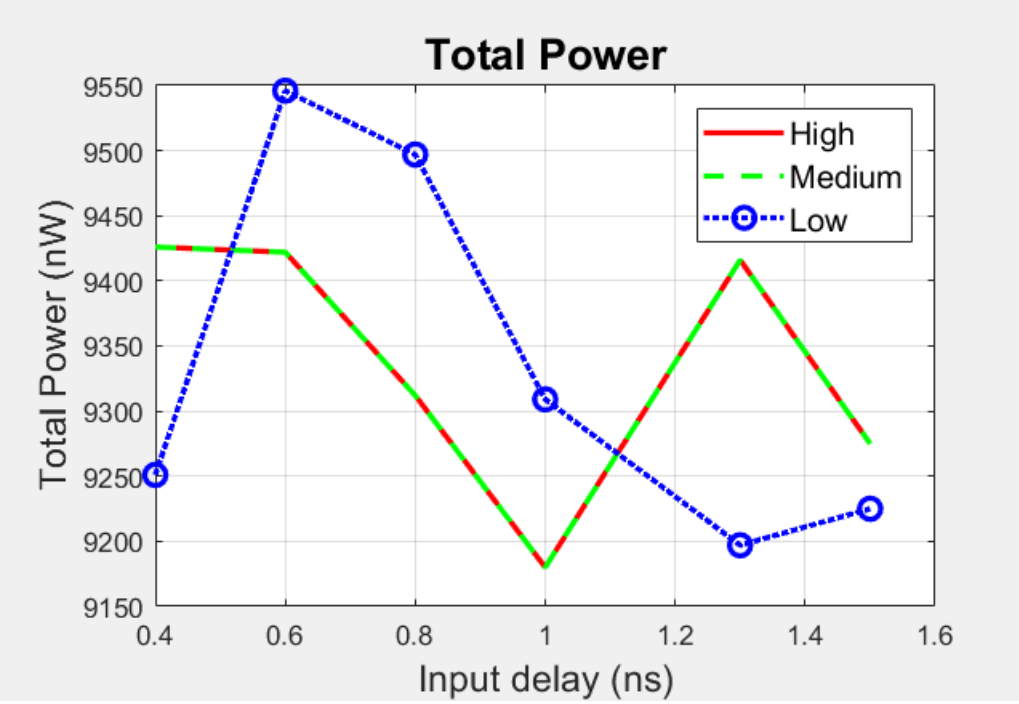


Figure: Total power VS input delay

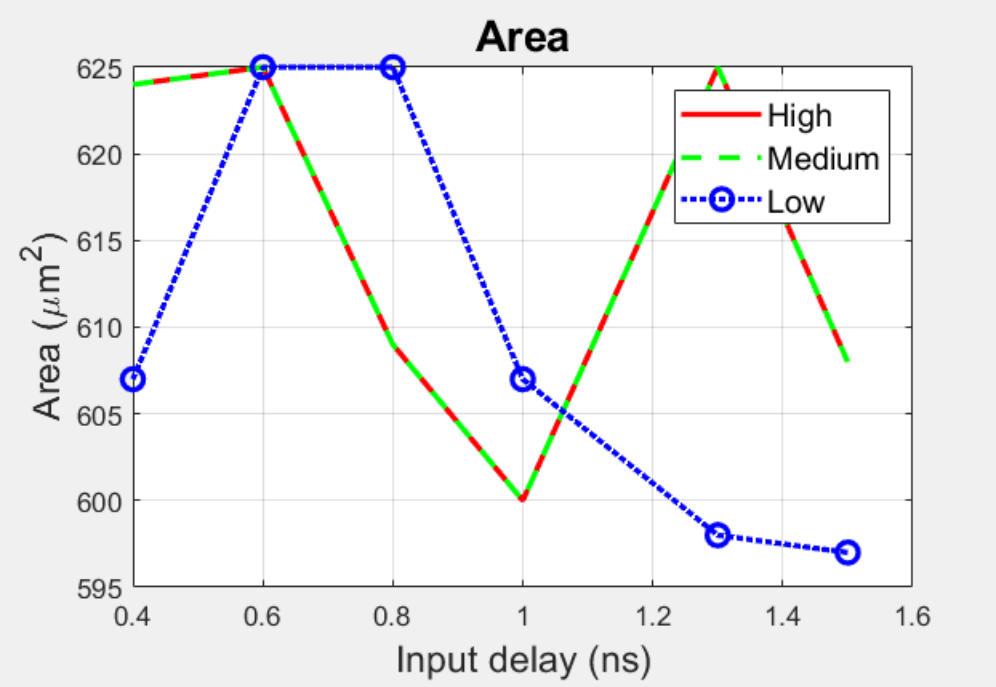


Figure: Total area VS input delay

Here we can see that when input delay is 1ns, then area and power will be minimum.

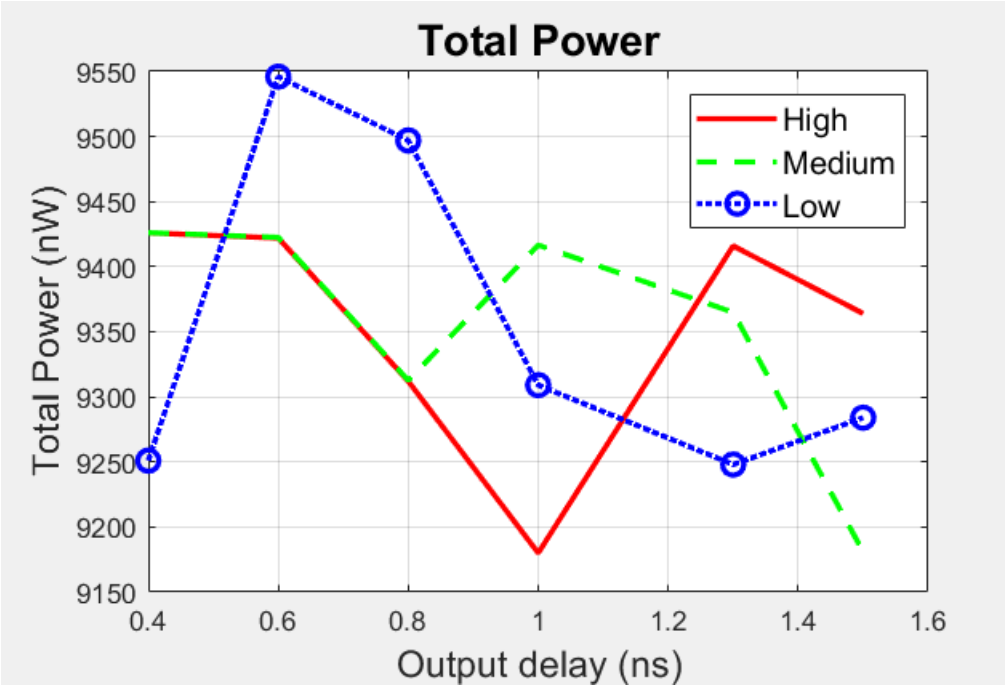


Figure: Total power VS output delay

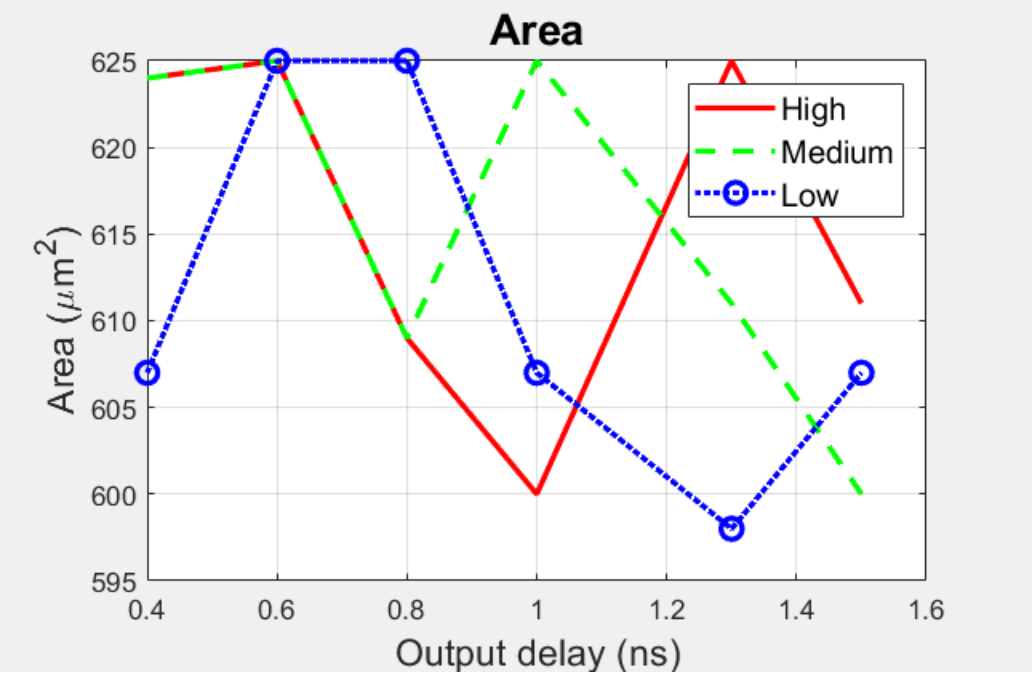


Figure: Total area VS output delay

Similarly, when output delay is 1ns, then area and power will be minimum.

Thus, we have got all the optimum points like below:

Clock frequency 100 MHz
Input delay 1 ns
Output delay 1 ns
SYN_EFF, MAP_EFF, OPT_EFF are high

Now we are going to generate synthesis file using these values.

SISO_FIFO_ShiftRegister.sdc

```
# setting up time units

set_units -time 1ns -capacitance pF

# setting the clock period 10ns, as period = 1/freq, here, freq = 100MHz
set clock_period 10;

set top_module "SISO_FIFO_ShiftRegister"

set clock_port {Clk};

#set reset_port {rst_n};
```

```
# setting the input ports in a list to a variable
set input_ports {Load, Left, Din, A} ;

# setting the output ports in a list to a variable
set output_ports {Dout,register} ;

# define the clocks
create_clock -period ${clock_period} -waveform {0 6} -name func_clk
[get_ports ${clock_port}]

# setting up constraints for the reset signal
# set_multicycle_path -setup 3 -from [get_ports ${reset_port}]
# set_multicycle_path -hold 2 -from [get_ports ${reset_port}]

# Define input delays
set_input_delay 1 -clock [get_clocks {func_clk}] ${input_ports}

# Define output delays
set_output_delay 1 -clock [get_clocks {func_clk}] ${output_ports}
```

SISO_FIFO_ShiftRegister.tcl

```
#run Genus in Legacy UI if Genus is invoked with Common UI
::legacy::set_attribute common_ui false / ;
if {[file exists /proc/cpuinfo]} {
sh grep "model name" /proc/cpuinfo
sh grep "cpu MHz" /proc/cpuinfo
}
puts "Hostname : [info hostname]"
##############################################################################
### Preset global variables and attributes
##############################################################################
set DESIGN SISO_FIFO_ShiftRegister
set SYN_EFF high
set MAP_EFF high
set OPT_EFF high
# Directory of PDK
#set pdk_dir /home/wsadiq/buet_flow/GPDK045
set pdk_dir /home/cad/VLSI2Lab/Digital/library/
#set_attribute init_lib_search_path $pdk_dir/gsclib045/timing
#set_attribute init_hdl_search_path /home/wsadiq/buet_flow/rtl
set_attribute init_lib_search_path $pdk_dir

#set_attribute init_hdl_search_path ../../rtl
##Set synthesizing effort for each synthesis stage
set_attribute syn_generic_effort $SYN_EFF
set_attribute syn_map_effort $MAP_EFF
set_attribute syn_opt_effort $OPT_EFF
#set_attribute library "\
slow_vdd1v0_basicCells_hvt.lib \
```

```
slow_vdd1v0_basicCells.lib \
slow_vdd1v0_basicCells_lvt.lib"
set_attribute library "\
slow_vdd1v0_basicCells.lib"
set_dont_use [get_lib_cells CLK*]
set_dont_use [get_lib_cells SDFF*]
set_dont_use [get_lib_cells DLY*]
set_dont_use [get_lib_cells HOLD*]
# If you dont want to use LVT uncomment this line
#set_dont_use [get_lib_cells *LVT*]

################################################################
### Load Design
################################################################
###source verilog_files.tcl
read_hdl "\
${DESIGN}.v"
elaborate $DESIGN
puts "Runtime & Memory after 'read_hdl'"
time_info Elaboration
check_design -unresolved
################################################################
### Constraints Setup
################################################################
read_sdc SISO_FIFO_ShiftRegister.sdc

report timing -encounter >> reports/${DESIGN}_pretim.rpt

################################################################################################
###############
### Synthesizing to generic
################################################################################################
###############
syn_generic
puts "Runtime & Memory after 'syn_generic'"
time_info GENERIC
report datapath > reports/${DESIGN}_datapath_generic.rpt
generate_reports -outdir reports -tag generic
write_db -to_file ${DESIGN}_generic.db
report timing -encounter >> reports/${DESIGN}_generic.rpt

##This synthesizes your code
synthesize -to_mapped

## This writes all your files
write -mapped > SISO_FIFO_ShiftRegister_synth.v

## THESE FILES ARE NOT REQUIRED, THE SDC FILE IS A TIMING FILE
write_script > script
```

Synthesized logic circuit-

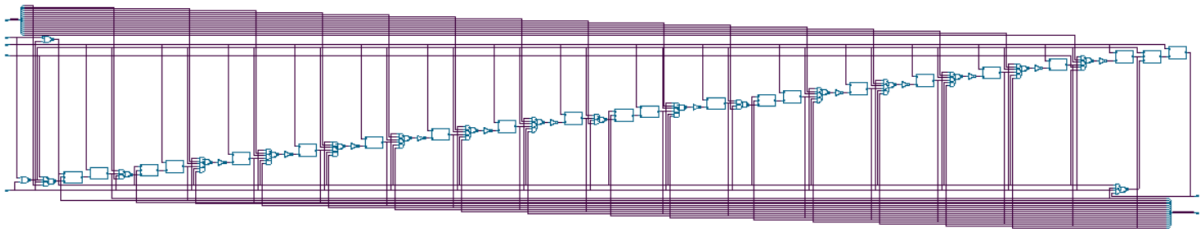
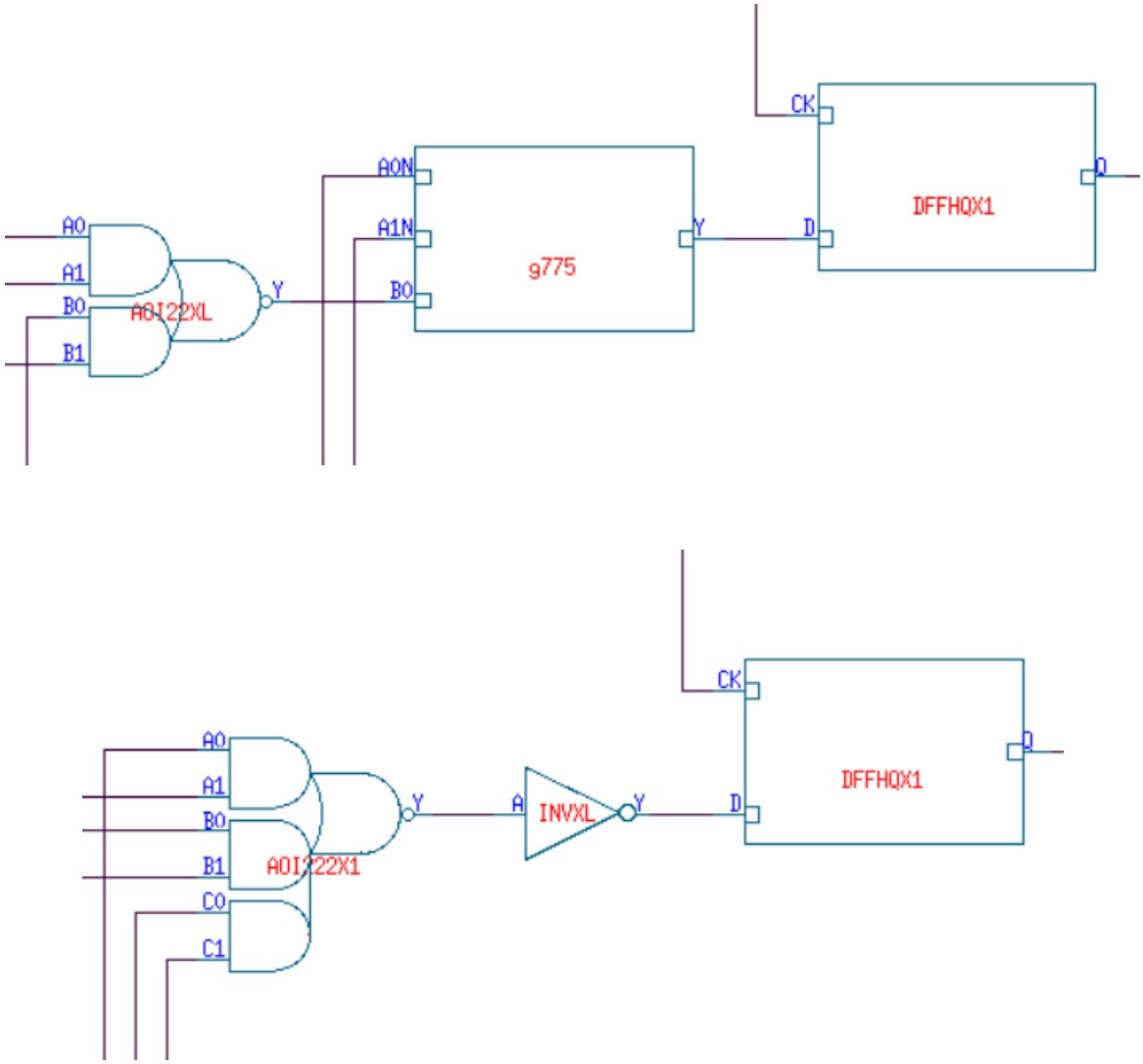
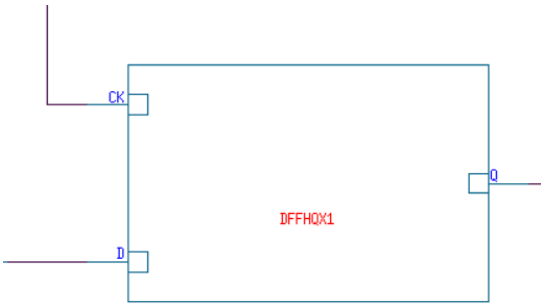


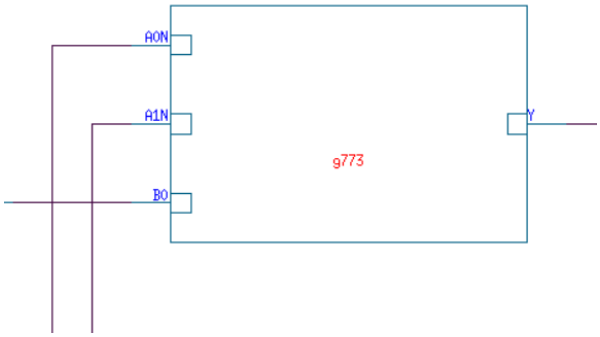
Figure: Total area VS output delay

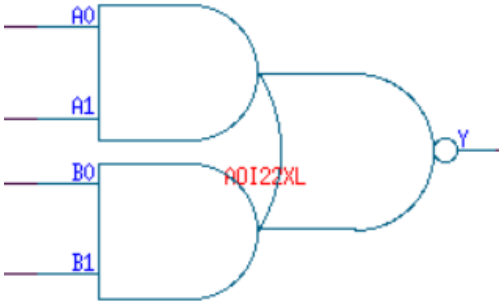
There are different two stages as below:

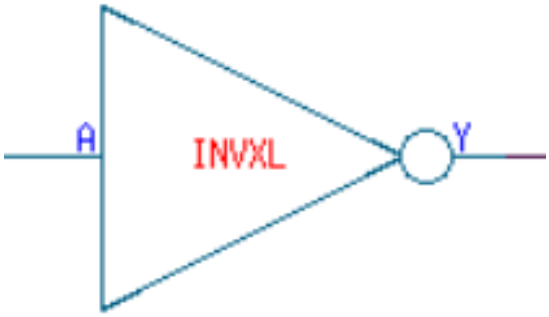


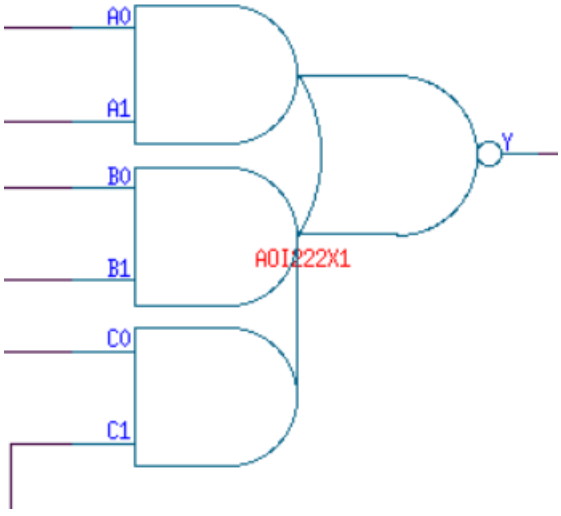
Zoomed version of those blocks-











Different Types of Report:

Report Area

Generated by: Genus(TM) Synthesis Solution 16.13-s036_1 (Dec 20 2016)
Generated on: Dec 06 2024 14:38:54
Module: SISO_FIFO_ShiftRegister
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed

Instance	Cells	Cell Area	Net Area	Total Area	Wireload	WL Flag
SISO_FIFO_ShiftRegister	53	159.37	0.00	159.37	<none>	(D)

Close Help

Figure: Area report

Report Statistics

Generated by: Genus(TM) Synthesis Solution 16.13-s036_1 (Dec 20 2016)
Generated on: Dec 06 2024 14:40:51
Module: SISO_FIFO_ShiftRegister
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed

Type	Instances	Area	Area %
sequential	17	93.02	58.40
inverter	12	8.21	5.20
logic	24	58.14	36.50
physical_cells	0	0.00	0.00
TOTAL	53	159.37	100.10

Close Help

Figure: Individual block area report

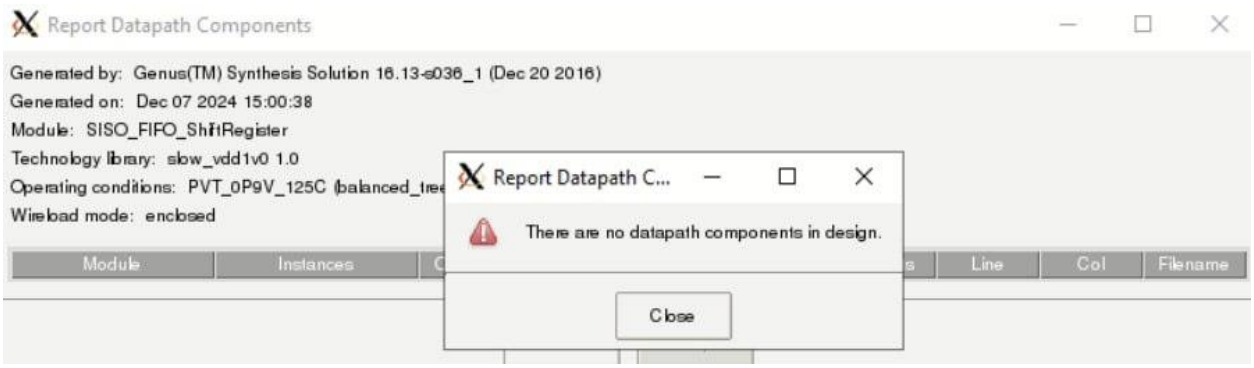


Figure: Datapath components

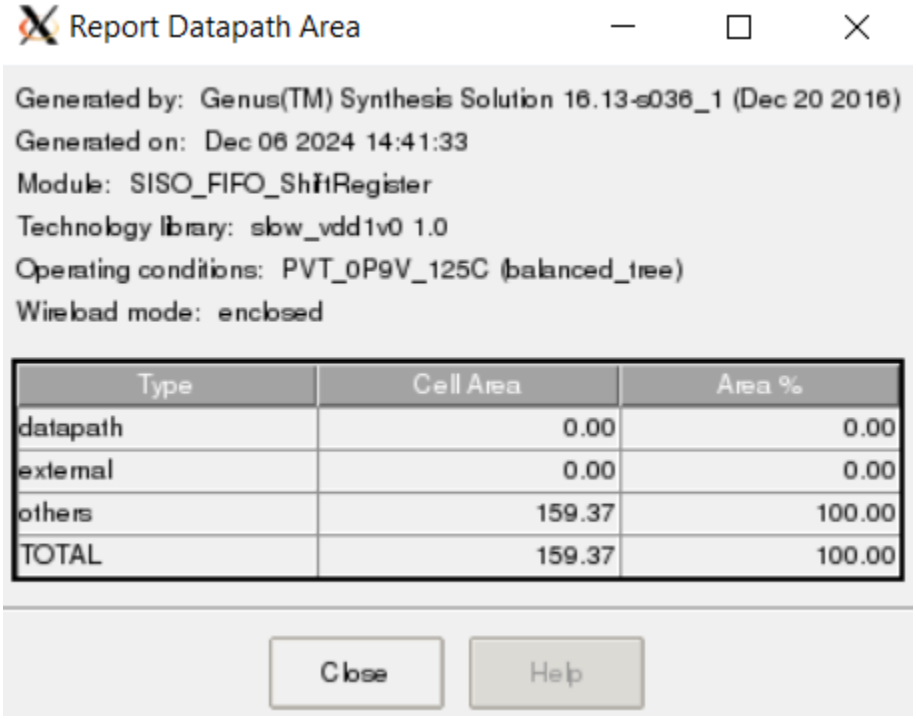


Figure: Datapath area report

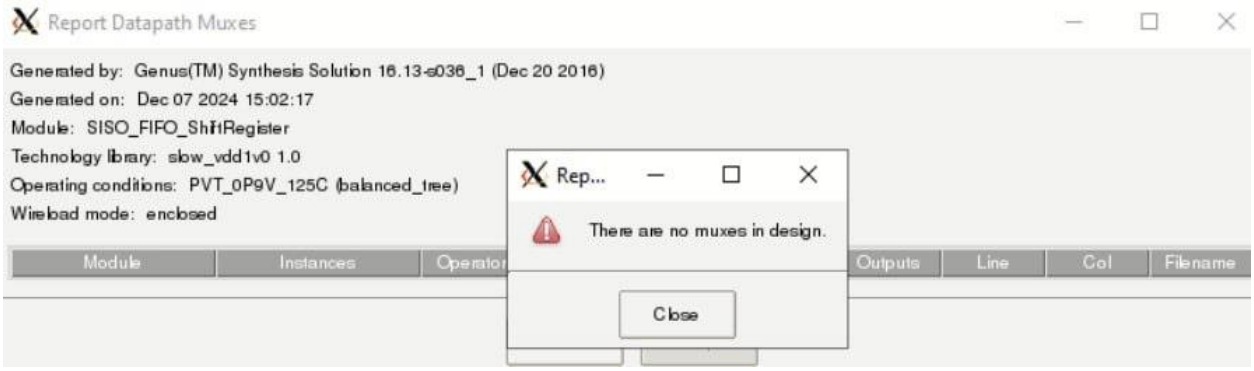


Figure: MUX components

Report Mapped Gates

Generated by: Genus(TM) Synthesis Solution 16.13-s036_1 (Dec 20 2016)
Generated on: Dec 07 2024 14:55:25
Module: SISO_FIFO_ShiftRegister
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed

Gate	Instances	Area	Library
AOI222X1	12	36.94	slow_vdd1v0
AOI22XL	5	10.26	slow_vdd1v0
DFFHQX1	17	93.02	slow_vdd1v0
INVXL	12	8.21	slow_vdd1v0
NOR2BX1	1	1.37	slow_vdd1v0
NOR2X1	1	1.03	slow_vdd1v0
OAI2BB1X1	5	8.55	slow_vdd1v0
TOTAL	53	159.38	

Close Help

Figure: Mapped gates in our design

Report Power

Generated by: Genus(TM) Synthesis Solution 16.13-s036_1 (Dec 20 2016)
Generated on: Dec 06 2024 14:42:32
Module: SISO_FIFO_ShiftRegister
Technology library: slow_vdd1v0 1.0
Operating conditions: PVT_0P9V_125C (balanced_tree)
Wireload mode: enclosed

Instance	Cells	Leakage (nW)	Internal (nW)	Net (nW)	Switching (nW)
SISO_FIFO_ShiftRegister	53	3.23	2451.43	522.28	2973.71

Close Help

Figure: Total power report

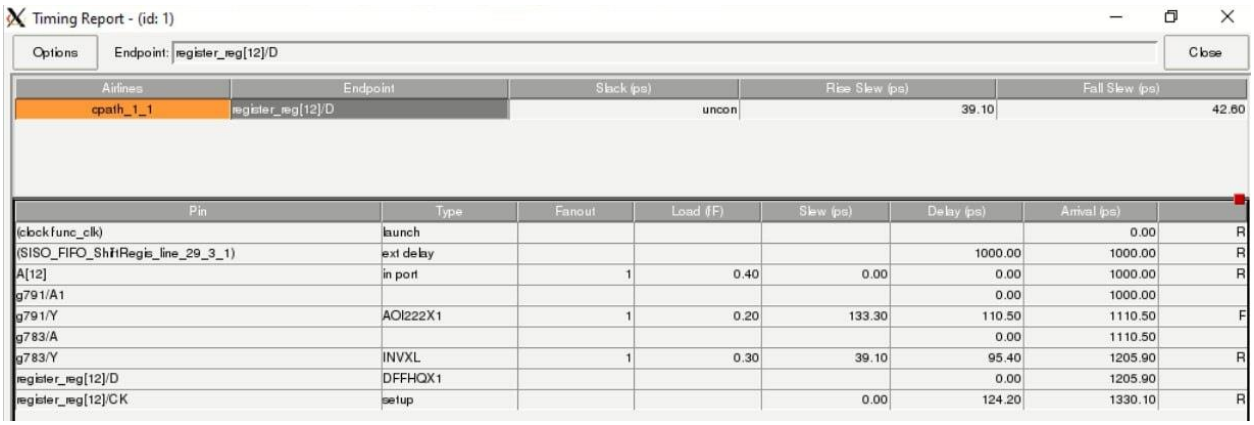


Figure: Timing report

SISO_FIFO_ShiftRegister_synth.v

```
// Generated by Cadence Genus(TM) Synthesis Solution 16.13-s036_1
// Generated on: Dec  6 2024 14:29:00 BDT (Dec  6 2024 08:29:00 UTC)

// Verification Directory fv/SISO_FIFO_ShiftRegister

module SISO_FIFO_ShiftRegister(Clk, Load, Left, Din, A, Dout, register);
  input Clk, Load, Left, Din;
  input [15:0] A;
  output Dout;
  output [15:0] register;
  wire Clk, Load, Left, Din;
  wire [15:0] A;
  wire Dout;
  wire [15:0] register;
  wire n_0, n_1, n_2, n_3, n_4, n_5, n_6, n_7;
  wire n_8, n_9, n_10, n_11, n_12, n_13, n_14, n_15;
  wire n_16, n_17, n_18, n_19, n_20, n_21, n_22, n_23;
  wire n_24, n_25, n_26, n_27, n_28, n_29, n_30, n_31;
  wire n_32, n_33, n_34, n_35;
  DFFHQX1 Dout_reg(.CK (Clk), .D (n_35), .Q (Dout));
```

```
DFFHQX1 \register_reg[14] (.CK (Clk), .D (n_34), .Q (register[14]));
OAI2BB1X1 g747(.A0N (n_32), .A1N (register[0]), .B0 (n_31), .Y
     (n_35));
OAI2BB1X1 g748(.A0N (n_30), .A1N (register[13]), .B0 (n_33), .Y
     (n_34));
AOI22XL g749(.A0 (n_32), .A1 (register[15]), .B0 (Load), .B1 (A[14]),
     .Y (n_33));
AOI22XL g750(.A0 (n_30), .A1 (register[15]), .B0 (Load), .B1 (Dout),
     .Y (n_31));
DFFHQX1 \register_reg[13] (.CK (Clk), .D (n_17), .Q (register[13]));
DFFHQX1 \register_reg[6] (.CK (Clk), .D (n_21), .Q (register[6]));
DFFHQX1 \register_reg[2] (.CK (Clk), .D (n_28), .Q (register[2]));
DFFHQX1 \register_reg[3] (.CK (Clk), .D (n_20), .Q (register[3]));
DFFHQX1 \register_reg[4] (.CK (Clk), .D (n_18), .Q (register[4]));
DFFHQX1 \register_reg[5] (.CK (Clk), .D (n_22), .Q (register[5]));
DFFHQX1 \register_reg[10] (.CK (Clk), .D (n_25), .Q (register[10]));
DFFHQX1 \register_reg[7] (.CK (Clk), .D (n_16), .Q (register[7]));
DFFHQX1 \register_reg[0] (.CK (Clk), .D (n_24), .Q (register[0]));
DFFHQX1 \register_reg[8] (.CK (Clk), .D (n_23), .Q (register[8]));
DFFHQX1 \register_reg[9] (.CK (Clk), .D (n_26), .Q (register[9]));
DFFHQX1 \register_reg[1] (.CK (Clk), .D (n_27), .Q (register[1]));
DFFHQX1 \register_reg[11] (.CK (Clk), .D (n_29), .Q (register[11]));
DFFHQX1 \register_reg[12] (.CK (Clk), .D (n_15), .Q (register[12]));
DFFHQX1 \register_reg[15] (.CK (Clk), .D (n_19), .Q (register[15]));
INVXL g782(.A (n_8), .Y (n_29));
INVXL g777(.A (n_6), .Y (n_28));
INVXL g778(.A (n_14), .Y (n_27));
INVXL g779(.A (n_12), .Y (n_26));
INVXL g780(.A (n_11), .Y (n_25));
INVXL g766(.A (n_13), .Y (n_24));
INVXL g781(.A (n_10), .Y (n_23));
```

```
OAI2BB1X1 g773(.A0N (n_30), .A1N (register[4]), .B0 (n_1), .Y (n_22));
INVXL g768(.A (n_4), .Y (n_21));
INVXL g769(.A (n_3), .Y (n_20));
OAI2BB1X1 g771(.A0N (n_30), .A1N (register[14]), .B0 (n_2), .Y
     (n_19));
INVXL g767(.A (n_9), .Y (n_18));
INVXL g784(.A (n_5), .Y (n_17));
OAI2BB1X1 g775(.A0N (n_30), .A1N (register[6]), .B0 (n_0), .Y (n_16));
INVXL g783(.A (n_7), .Y (n_15));
AOI222X1 g786(.A0 (Load), .A1 (A[1]), .B0 (n_32), .B1 (register[2]),
     .C0 (n_30), .C1 (register[0]), .Y (n_14));
AOI222X1 g770(.A0 (Load), .A1 (A[0]), .B0 (n_32), .B1 (register[1]),
     .C0 (Din), .C1 (n_30), .Y (n_13));
AOI222X1 g787(.A0 (Load), .A1 (A[9]), .B0 (n_32), .B1 (register[10]),
     .C0 (n_30), .C1 (register[8]), .Y (n_12));
AOI222X1 g788(.A0 (Load), .A1 (A[10]), .B0 (n_32), .B1
     (register[11]), .C0 (n_30), .C1 (register[9]), .Y (n_11));
AOI222X1 g789(.A0 (Load), .A1 (A[8]), .B0 (n_32), .B1 (register[9]),
     .C0 (n_30), .C1 (register[7]), .Y (n_10));
AOI222X1 g772(.A0 (Load), .A1 (A[4]), .B0 (n_32), .B1 (register[5]),
     .C0 (n_30), .C1 (register[3]), .Y (n_9));
AOI222X1 g790(.A0 (Load), .A1 (A[11]), .B0 (n_32), .B1
     (register[12]), .C0 (n_30), .C1 (register[10]), .Y (n_8));
AOI222X1 g791(.A0 (Load), .A1 (A[12]), .B0 (n_32), .B1
     (register[13]), .C0 (n_30), .C1 (register[11]), .Y (n_7));
AOI222X1 g785(.A0 (Load), .A1 (A[2]), .B0 (n_32), .B1 (register[3]),
     .C0 (n_30), .C1 (register[1]), .Y (n_6));
AOI222X1 g792(.A0 (Load), .A1 (A[13]), .B0 (n_32), .B1
     (register[14]), .C0 (n_30), .C1 (register[12]), .Y (n_5));
AOI222X1 g774(.A0 (Load), .A1 (A[6]), .B0 (n_32), .B1 (register[7]),
     .C0 (n_30), .C1 (register[5]), .Y (n_4));
```

```
  AOI222X1 g776(.A0 (Load), .A1 (A[3]), .B0 (n_32), .B1 (register[4]),
       .C0 (n_30), .C1 (register[2]), .Y (n_3));
  AOI22XL g793(.A0 (Din), .A1 (n_32), .B0 (Load), .B1 (A[15]), .Y
       (n_2));
  AOI22XL g794(.A0 (n_32), .A1 (register[6]), .B0 (Load), .B1 (A[5]),
       .Y (n_1));
  AOI22XL g795(.A0 (n_32), .A1 (register[8]), .B0 (Load), .B1 (A[7]),
       .Y (n_0));
  NOR2X1 g796(.A (Left), .B (Load), .Y (n_32));
  NOR2BX1 g797(.AN (Left), .B (Load), .Y (n_30));
Endmodule
```

Physical Design

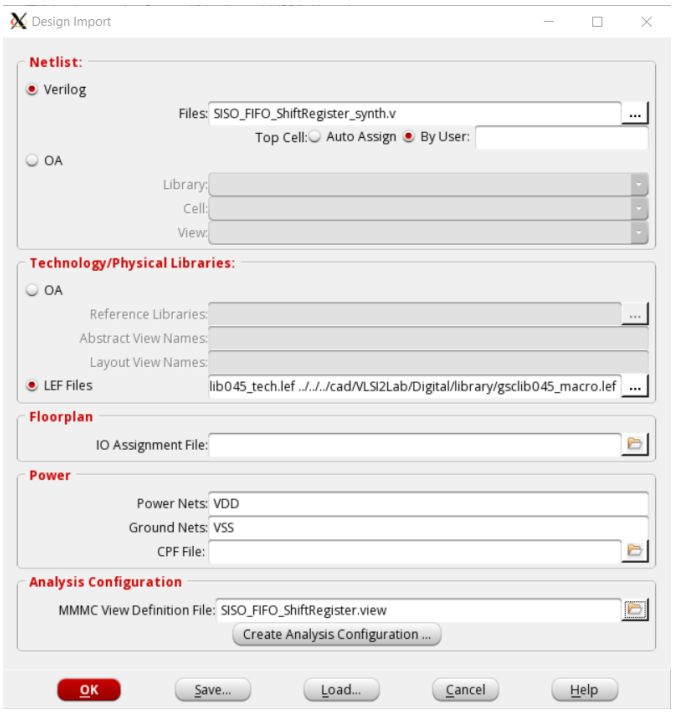
At this stage, we will move on with the physical design (back-end) of our synthesized SISO-FIFO-Shift Register netlist. For this, the Cadence Innovus Implementation System will be employed.

The following Cadence CAD tools will be used for this purpose:

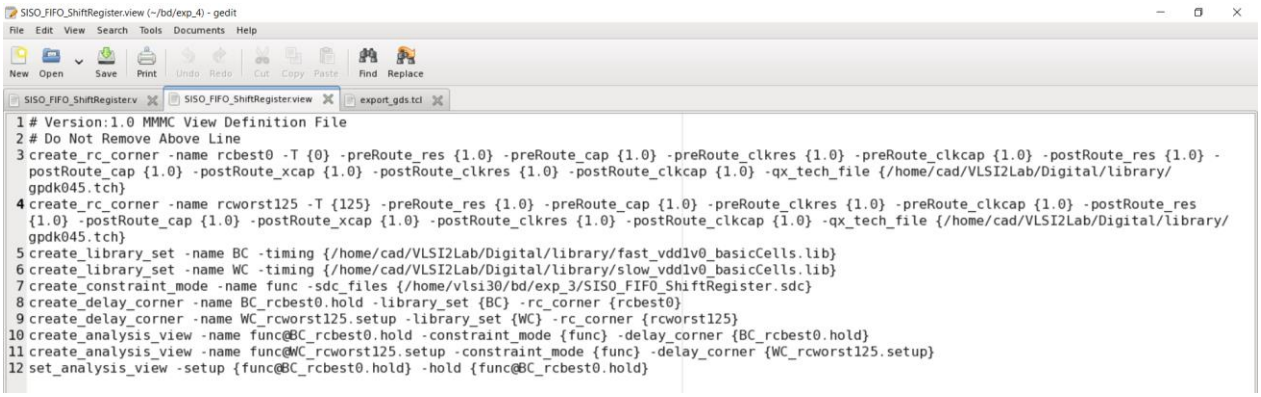
- Cadence Genus Synthesis Solution.
- Innovus Implementation System.

Different Steps of the Procedure: (For SISO-FIFO-Shift Register synthesis)

First of all, we have to import design by going File → Import Design in Innovus Window. Here is the ‘Design Import’ window-

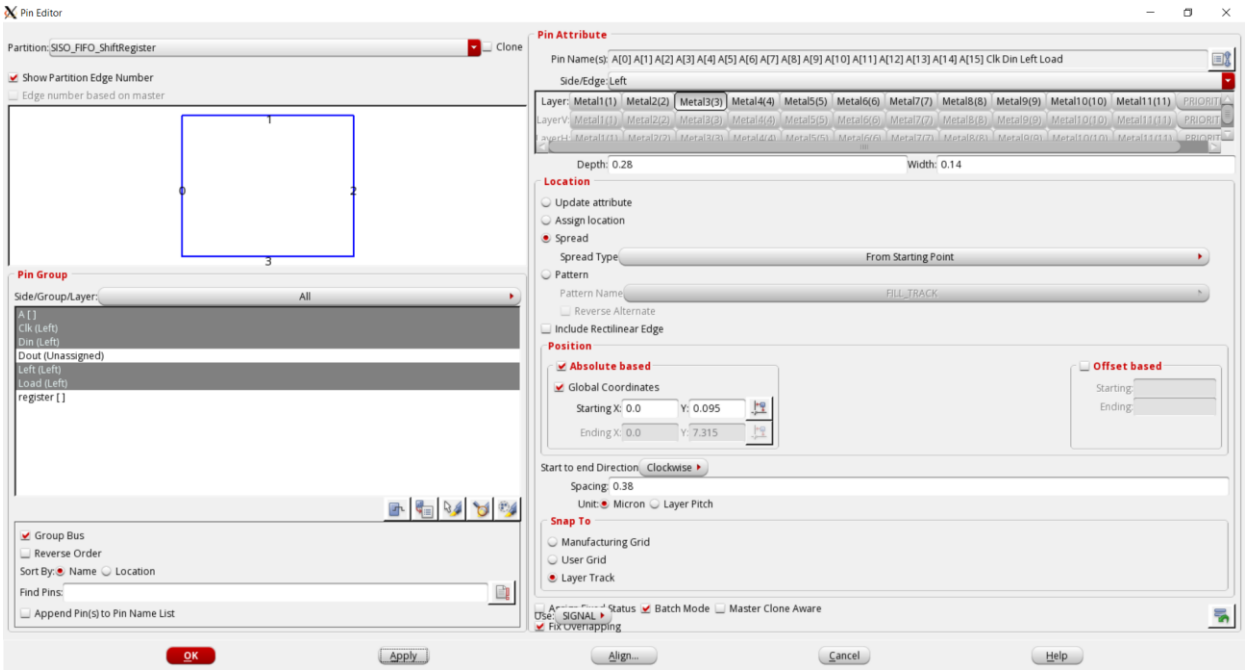


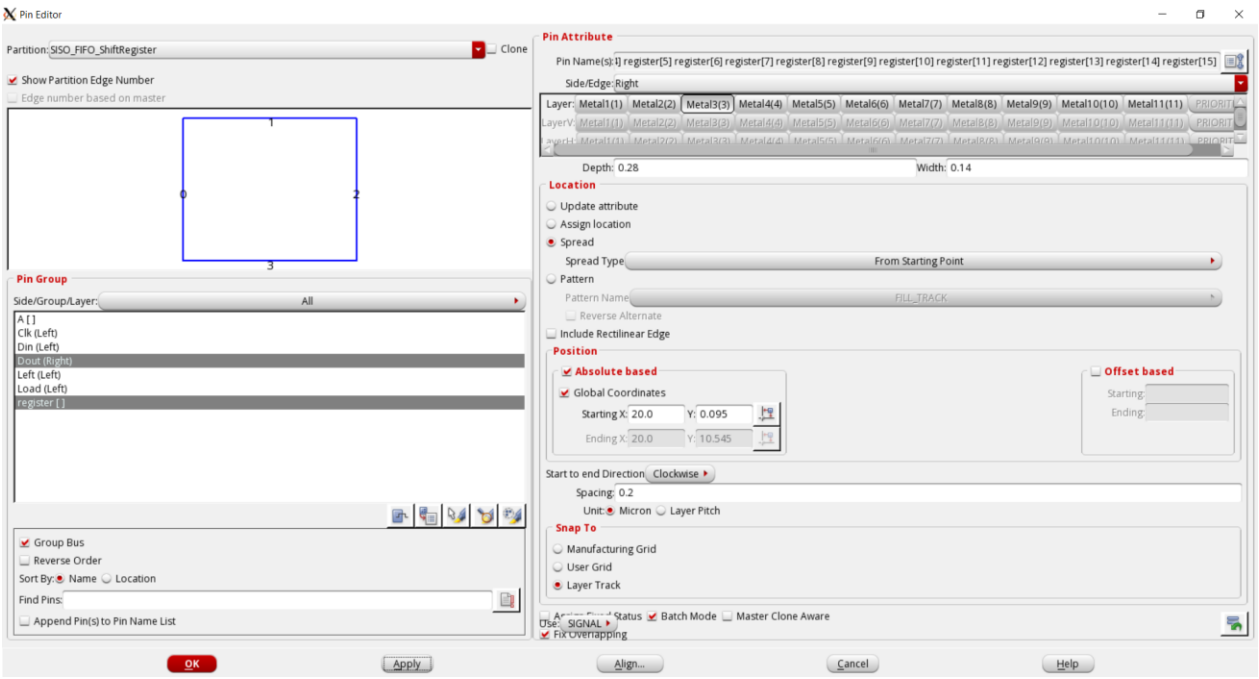
"SISO_FIFO_ShiftRegister.view" file-



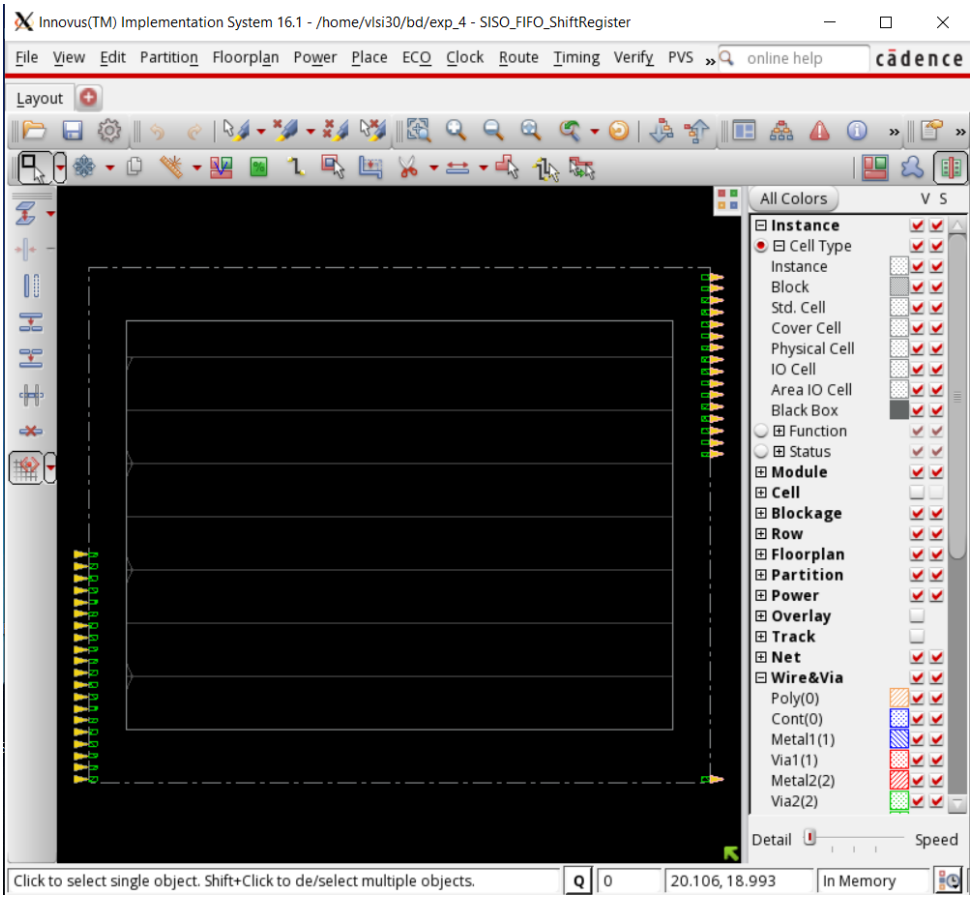
Then, we have to specify the floorplan. Here is the 'Specify Floorplan' window-

After specifying the floorplan, we have to add IO ports in the design. As we don't have any .tcl file for IO pins, we have to set it from 'Pin Editor' window. In Innovus window, we have to go Edit → Pin Editor. We have set the input and output ports on the left and right correspondingly.



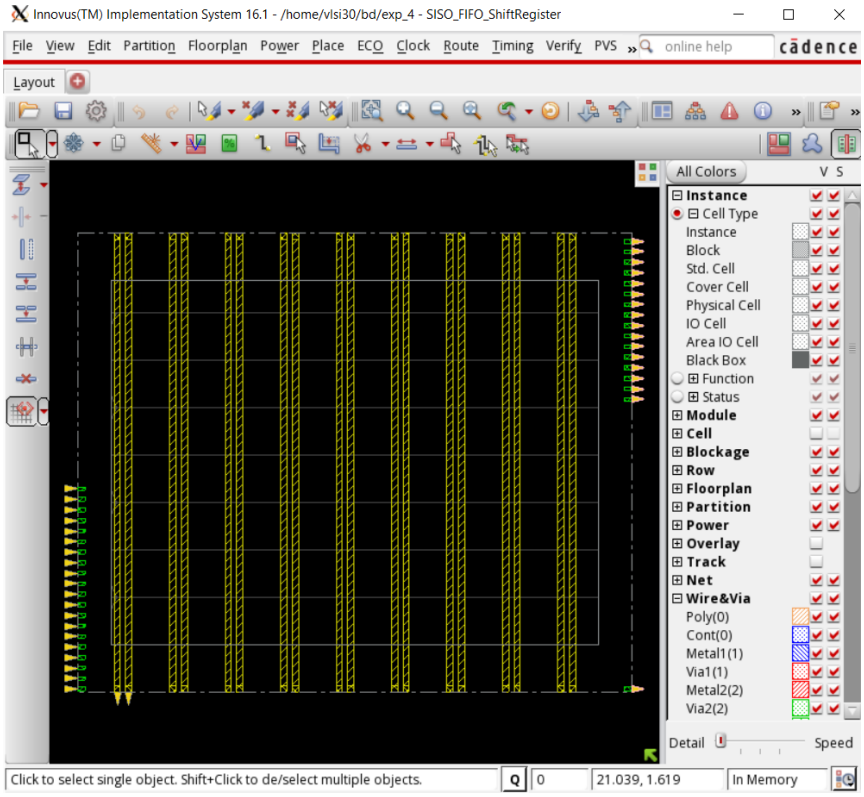


Then, the Innovus window-

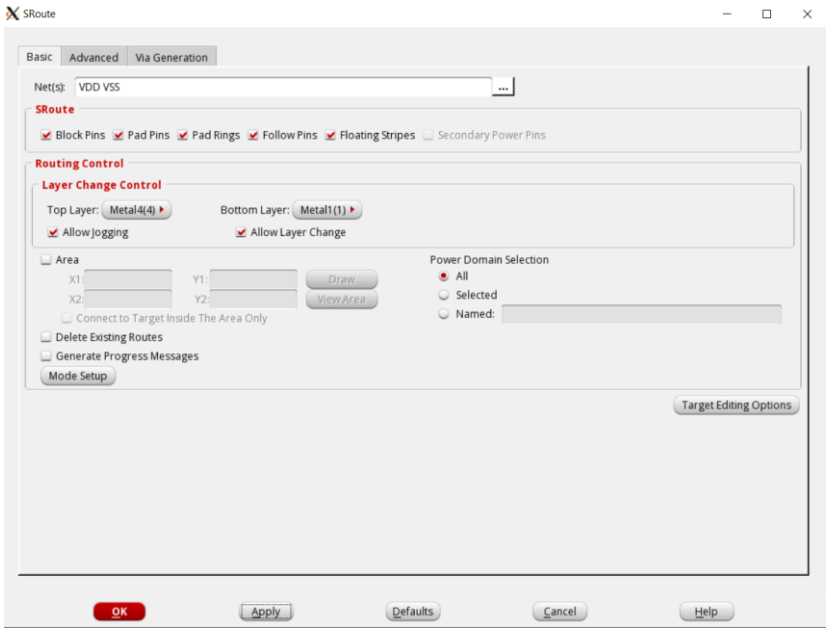


Now, we created the Power Mesh. For that, Power → Power Planning → Add Stripe and the 'Add Stripes' window- (Width and Spacing were default. Only set- x from left)

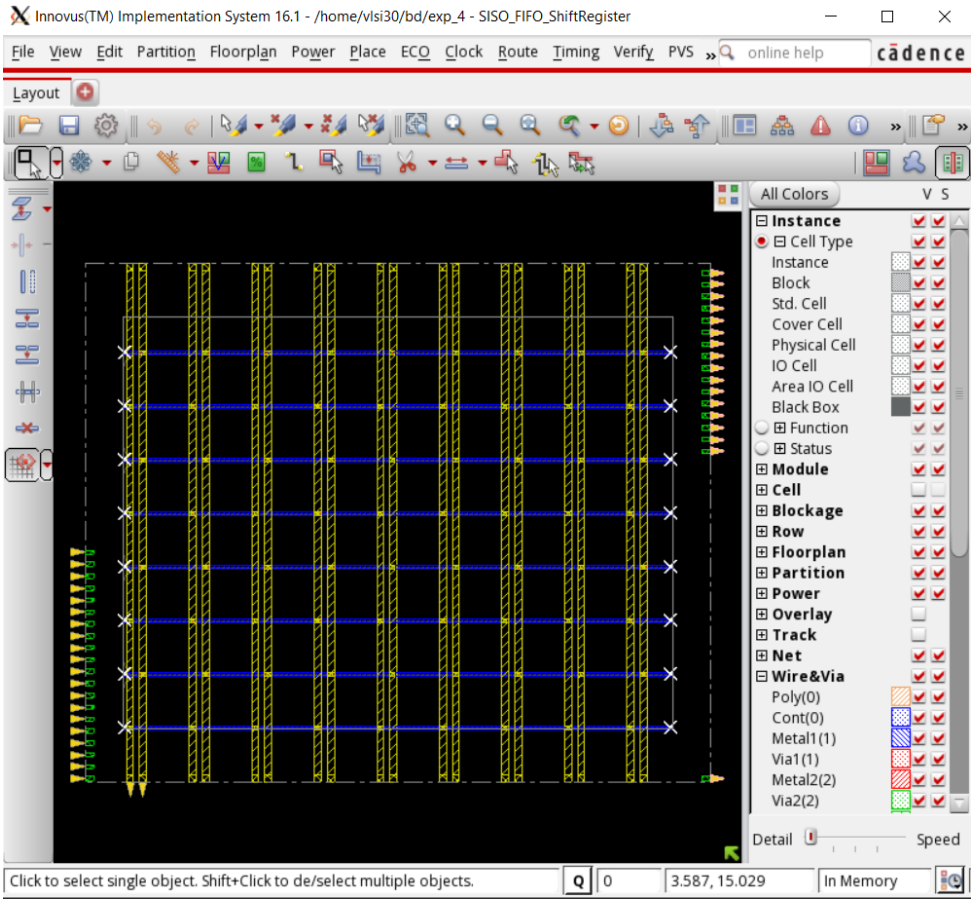
Then the Innovus window was like this-



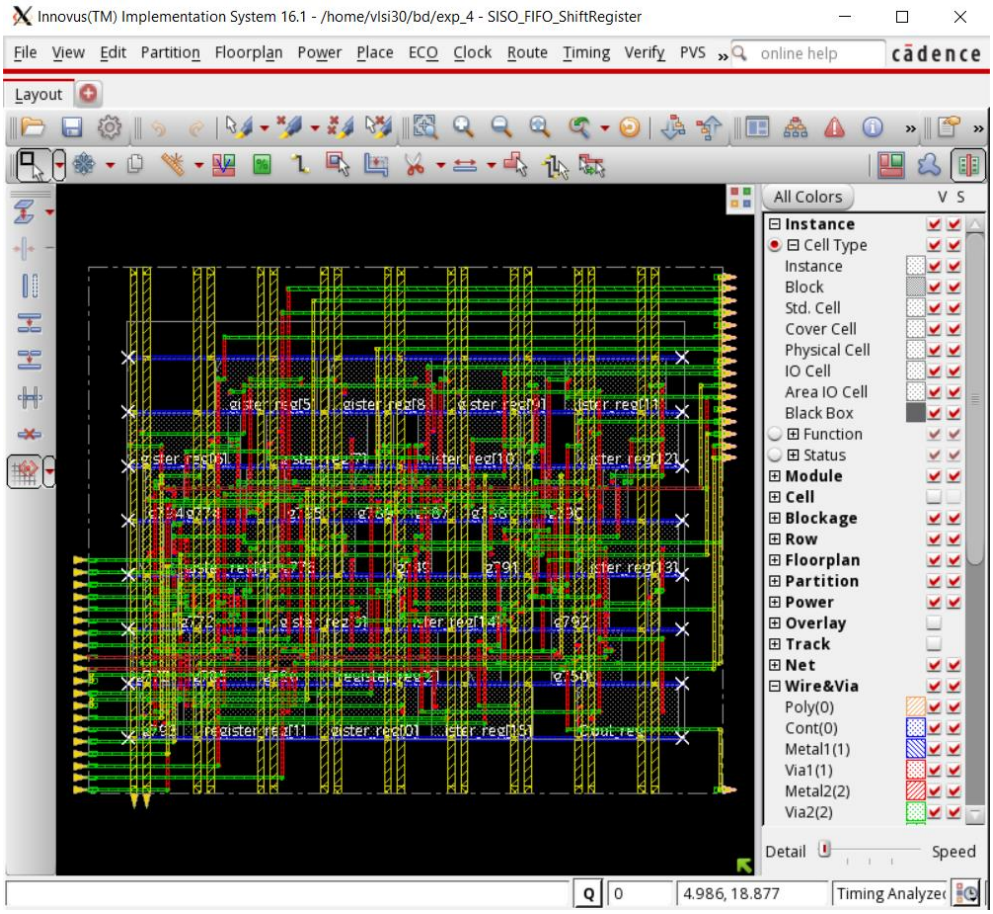
Then we added standard cell M1 layer rail in the design. For this, Route → Special Route and the ‘SRoute’ window-



Then the Innovus window-



To place standard cells in the design and optimize placement, we used ‘place_opt_design’ command and then the Innovus window-

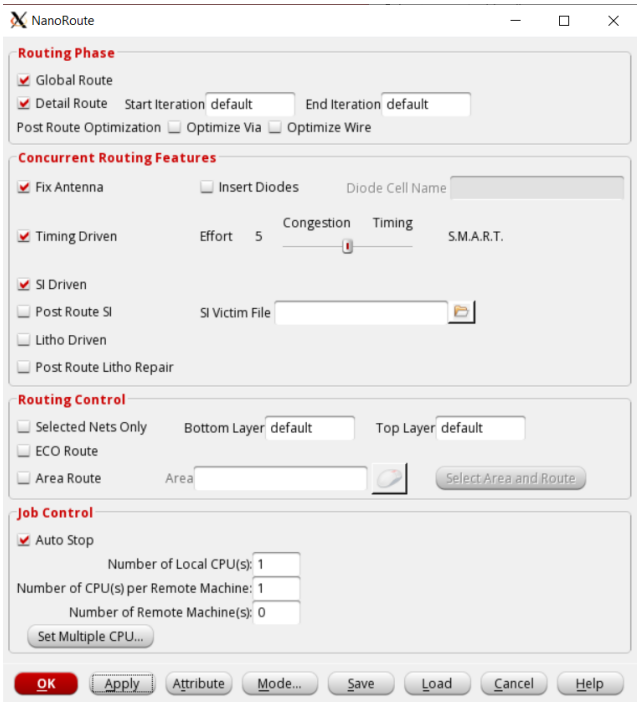


DRVs	Real		Total
	Nr nets(terms)	Worst Vio	Nr nets(terms)
max_cap	0 (0)	0.000	0 (0)
max_tran	0 (0)	0.000	0 (0)
max_fanout	0 (0)	0	0 (0)
max_length	0 (0)	0	0 (0)

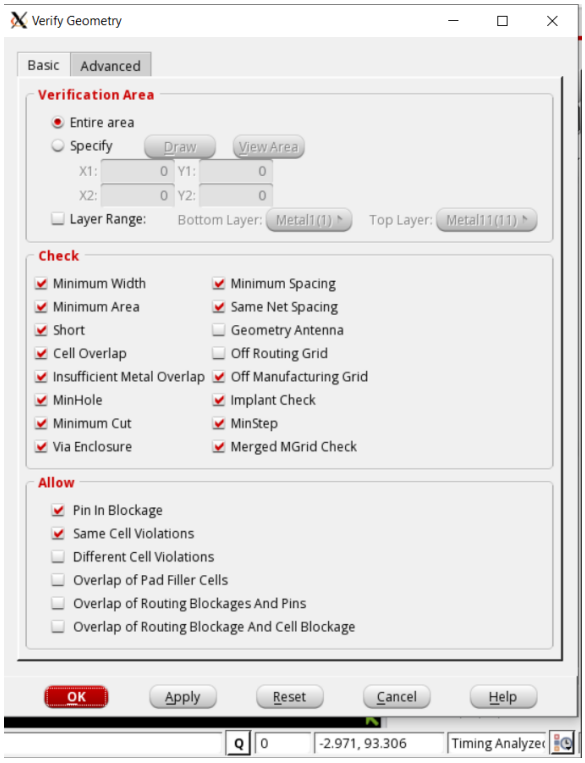
Density: 75.649%

Here, the circuit density of our designed layout is 75.649 %.

Then we did ‘Clock Tree Synthesis’ by lab sheets’ command. To do ‘Running NanoRoute’ step: Route → NanoRoute → Route and the ‘NanoRoute’ window was like-

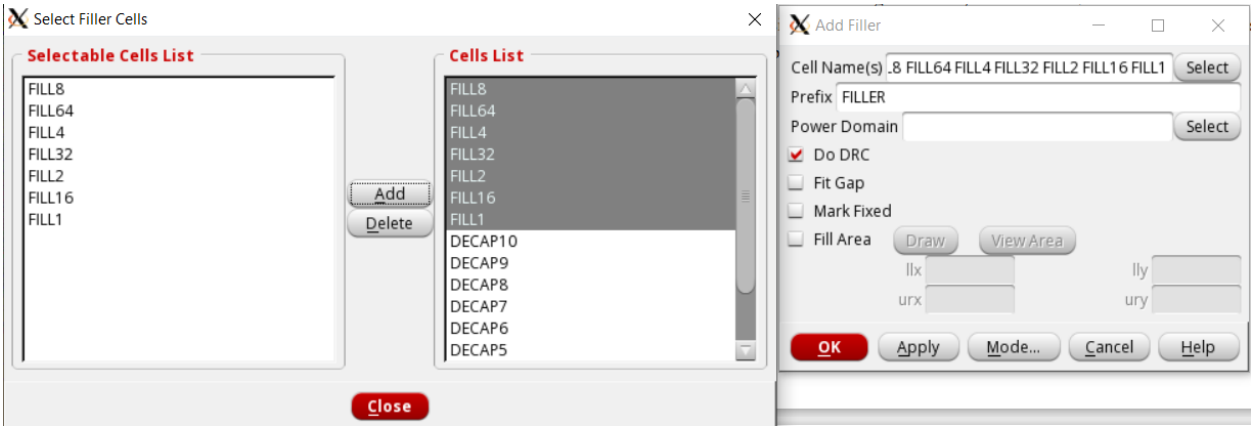


To do ‘Post-Route timing and SI Optimization’ step we followed lab sheet. Then, to verify design for DRC: Verify → Verify Geometry → OK. The ‘Verify Geometry’ window was-

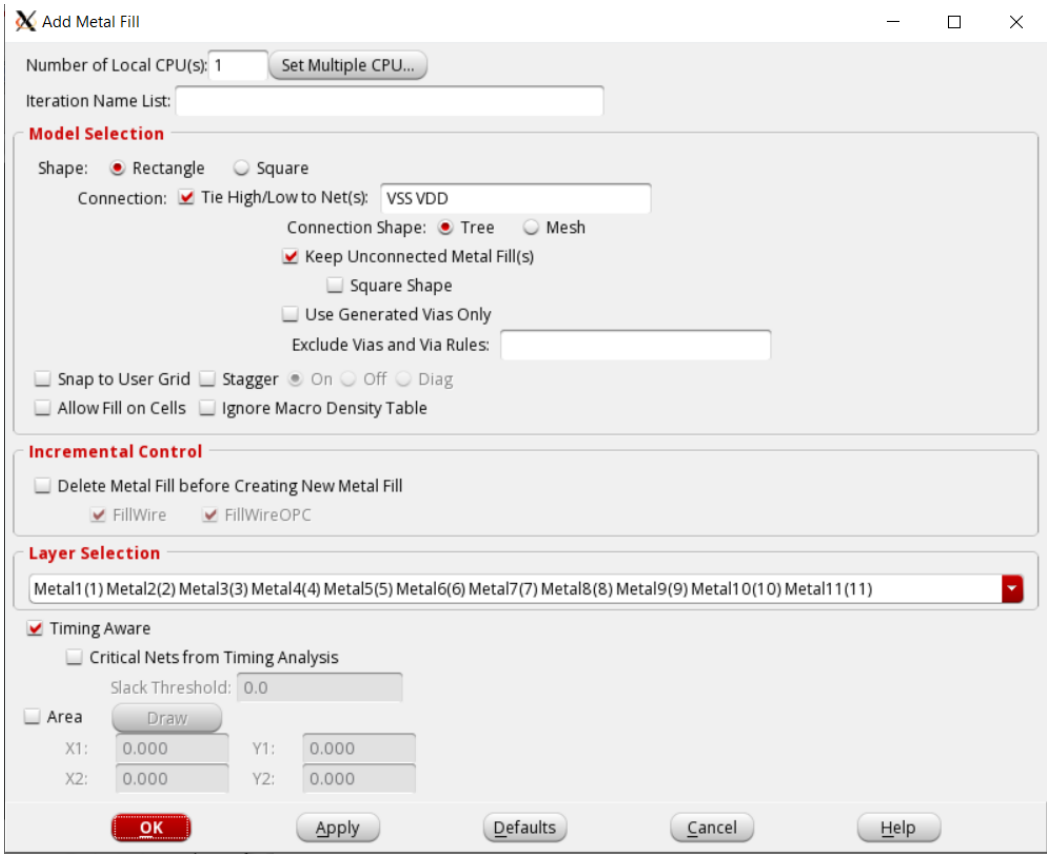


For ‘Adding filler cell and metal filling’ step: Place → Physical cell → Add Filler...

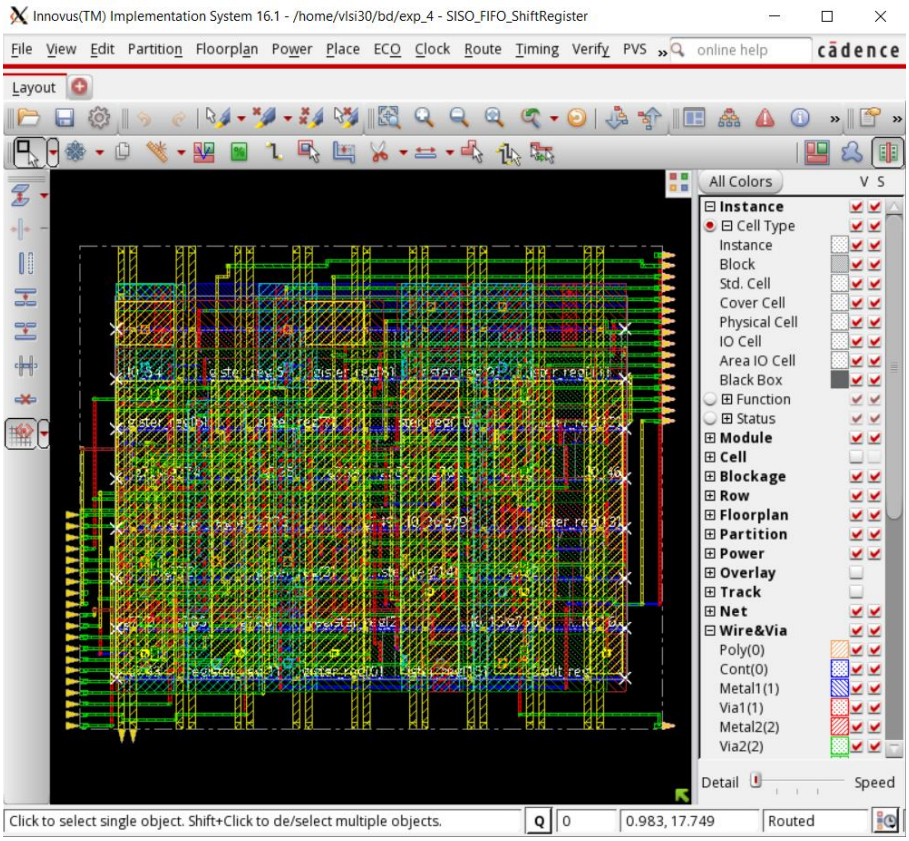
And the ‘Add Filler’ cell was-



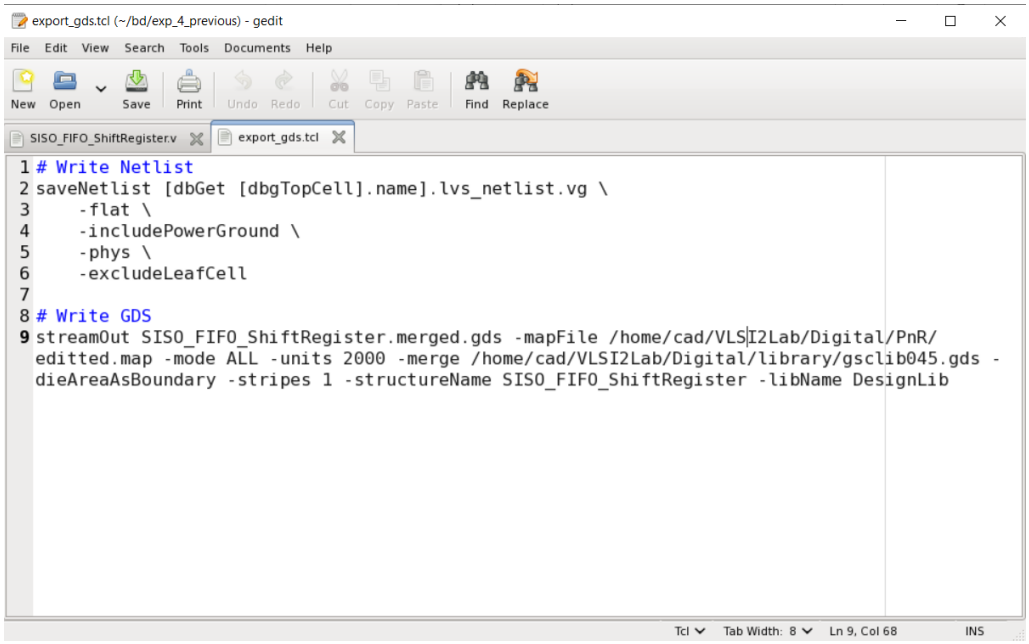
Then by Route → Metal fill → Add, the ‘Add Metal Fill’ window-



Then, the Innovus window-



In the next step, to export GDS & netlist file first we copied the export_gds.tcl file from the '/home/cad/VLSI2Lab/Digital/PnR/export_gds.tcl' location to our run directory. Then we changed the export_gds.tcl as below-



And then used this command in Innovus command shell-

```
innovus 30> source /home/vlsi30/bd/exp_4/export_gds.tcl
```

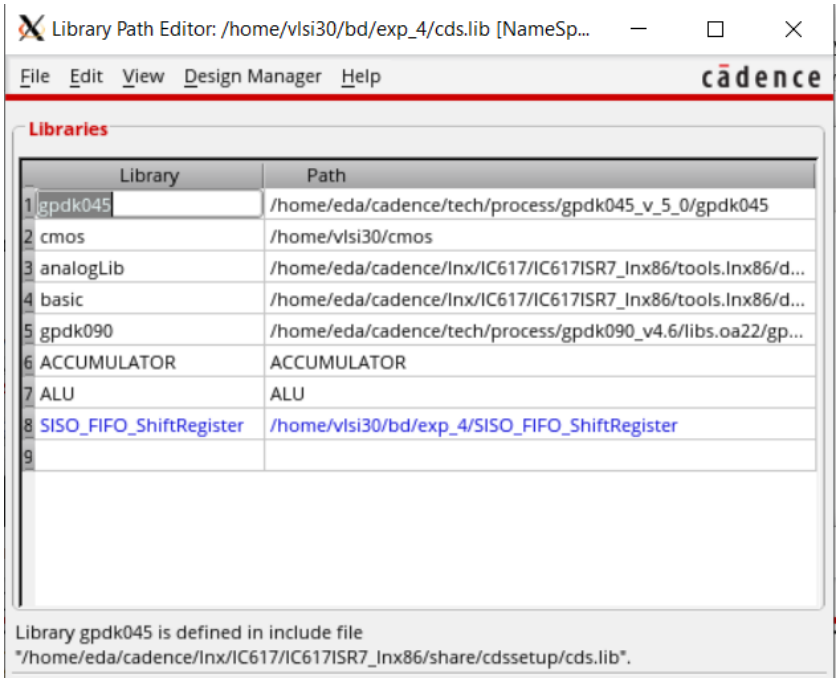
This generated two files ‘alu.lvs_netlist.vg’ and ‘alu.merged.gds’ in the run directory.

vlsi30@CadenceServer3:exp_4

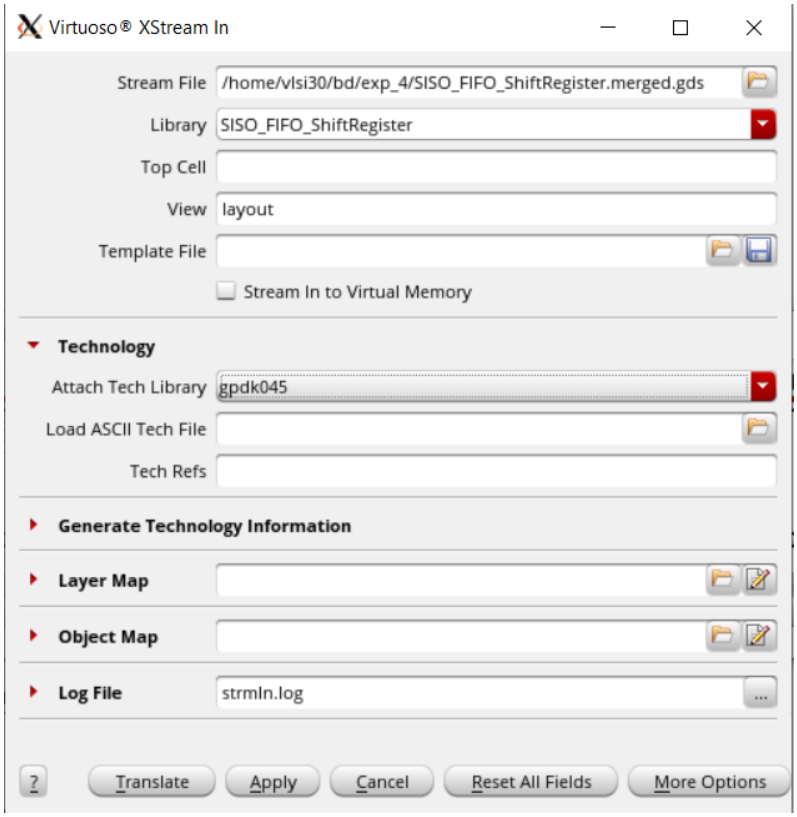
File Edit View Search Terminal Help

```
   31                              Metal4


Stream Out Information Processed for GDS version 5:
Units: 2000 DBU

Object                             Count
----------------------------------------
Instances                            115

Ports/Pins                            71
    metal layer Metal3                36
    metal layer Metal4                35

Nets                                 605
    metal layer Metal1                48
    metal layer Metal2               306
    metal layer Metal3               203
    metal layer Metal4                48

    Via Instances                    405

Special Nets                          24
    metal layer Metal1                 7
    metal layer Metal4                17

    Via Instances                    216

Metal Fills                            7
    metal layer Metal1                 2
    metal layer Metal2                 5

    Via Instances                     68

Metal FillOPCs                         0

    Via Instances                      0

Text                                 144
    metal layer Metal1                12
    metal layer Metal2                37
    metal layer Metal3                57
    metal layer Metal4                38


Blockages                              0


Custom Text                            0


Custom Box                             0

Trim Metal                             0
```

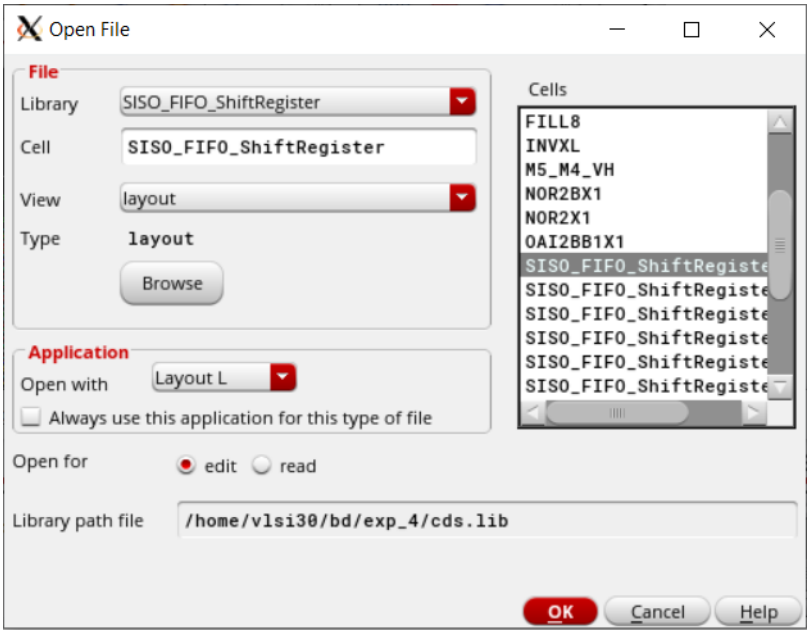
At this stage, we invoked virtuoso and added the library-



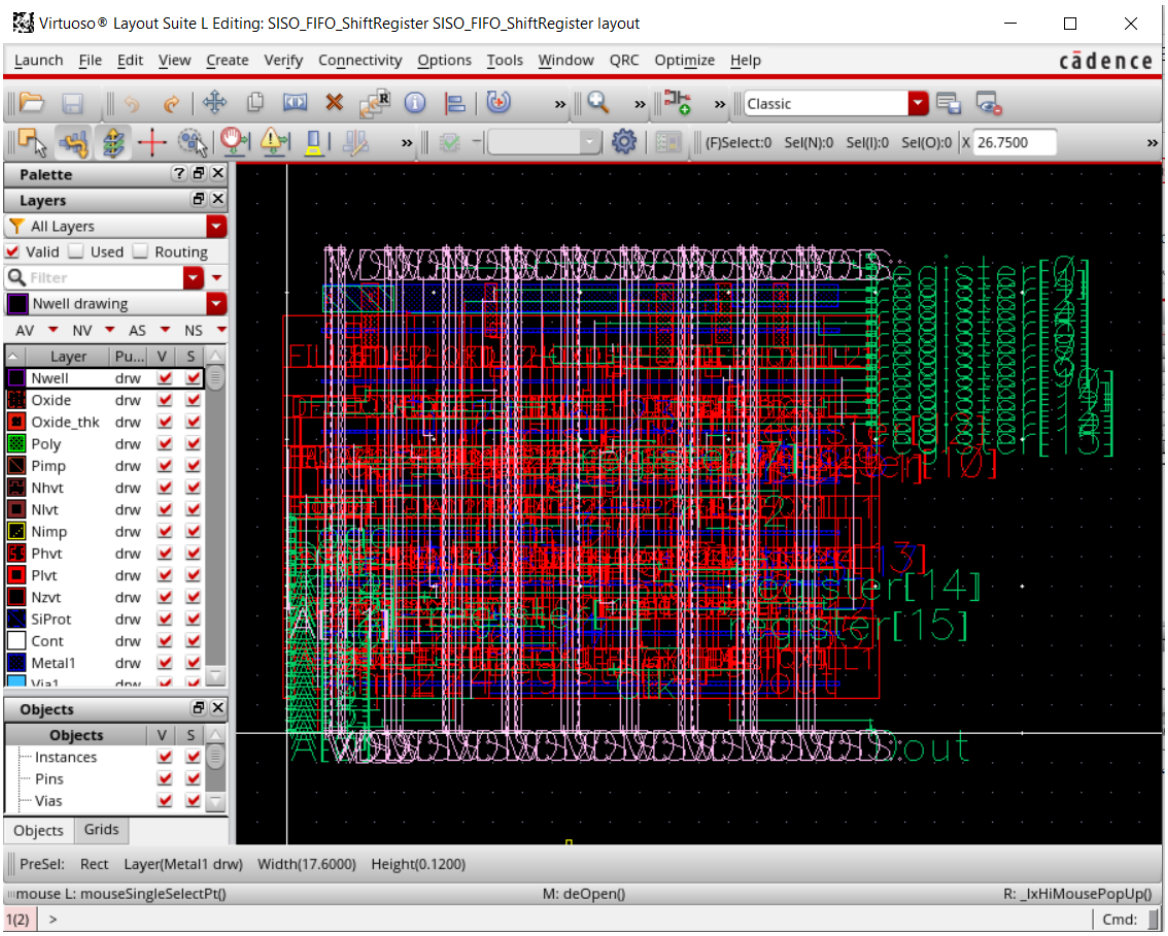
In the final step 'Run DRC', from File → Import → Stream, filled the 'Virtuoso® XStream In' window as like and selected 'Translate'-



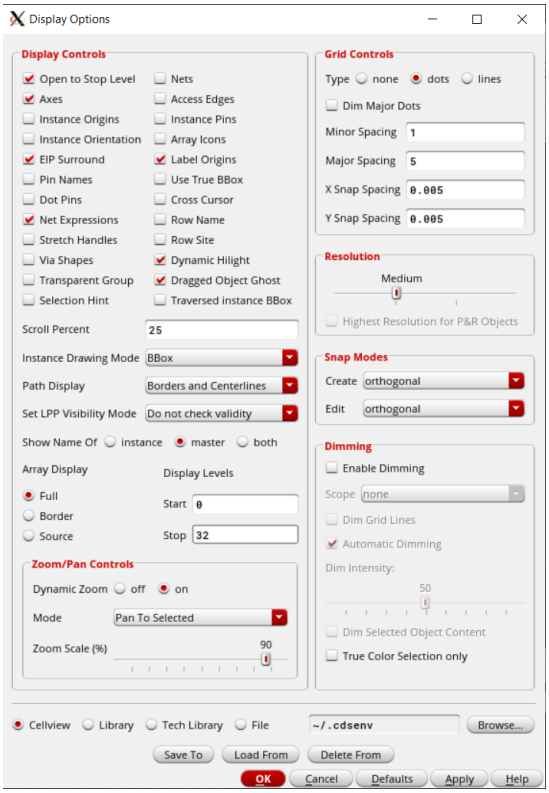
Then, from File → Open, filled the ‘Open File’ window-



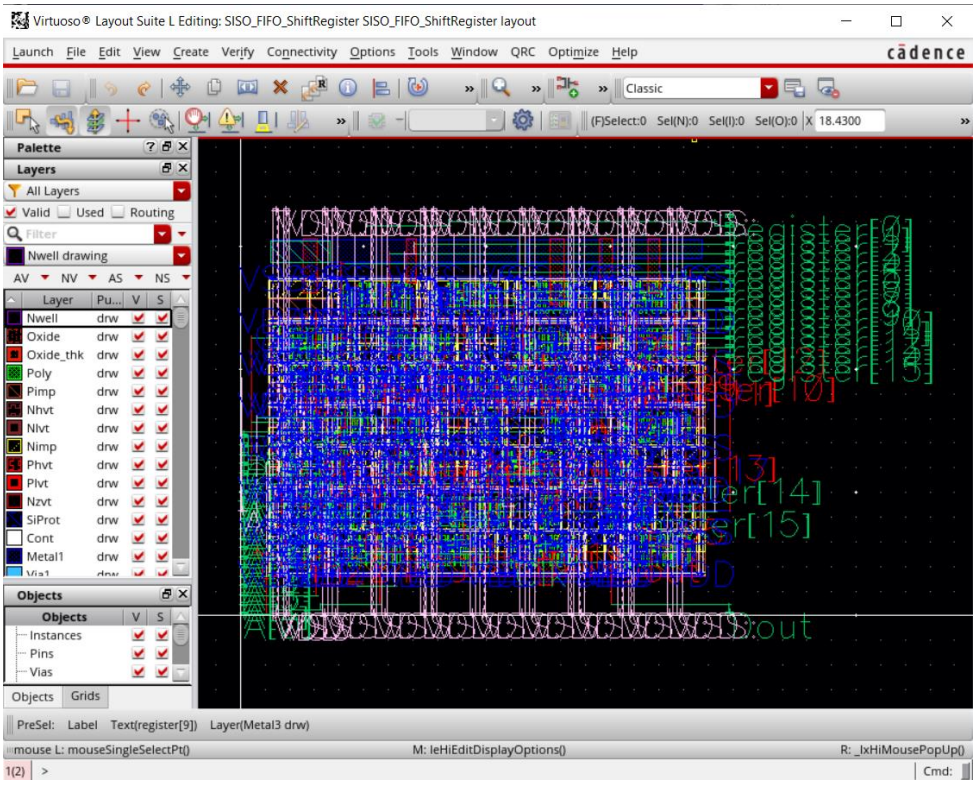
Then the ‘Virtuoso Layout Suit L’ window was like-



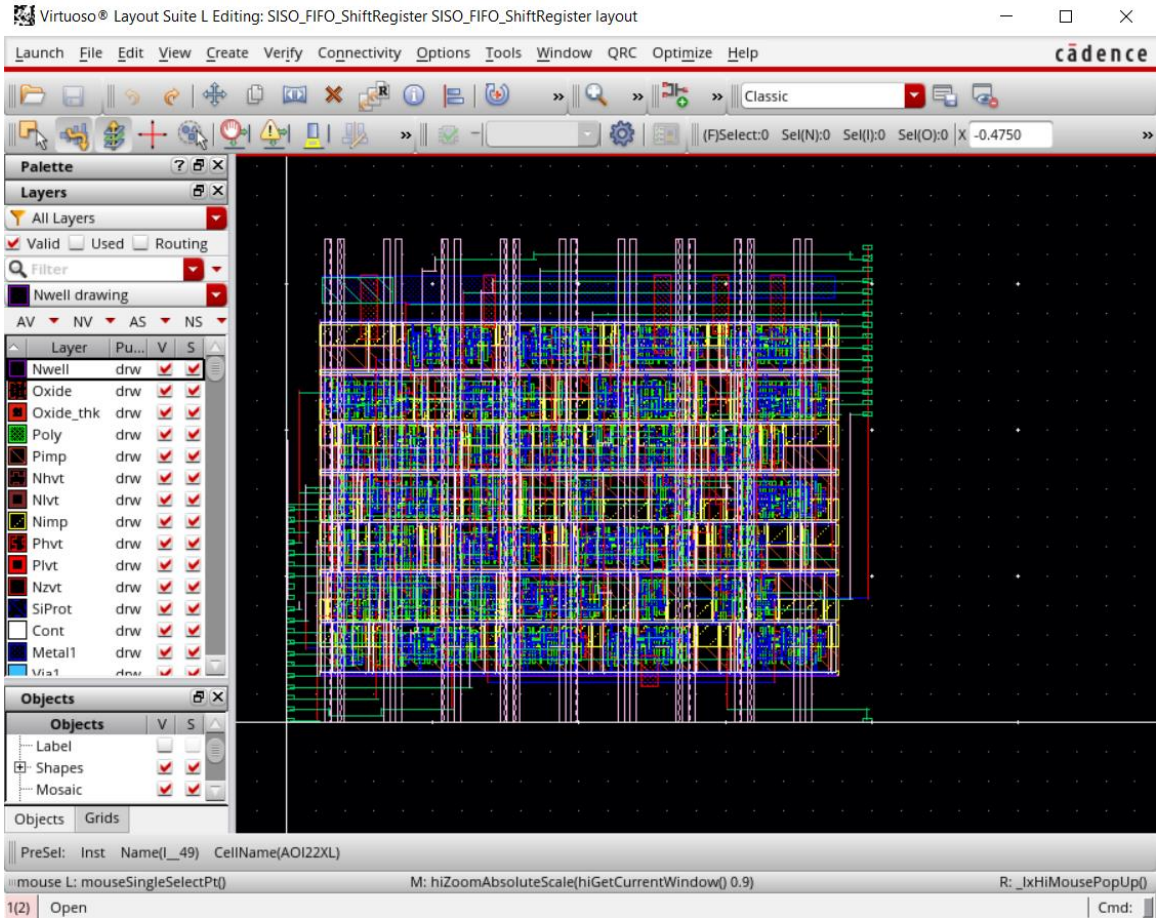
Then by pressing ‘E’, filled the ‘Display Options’ window as- (Only set Stop as 32)



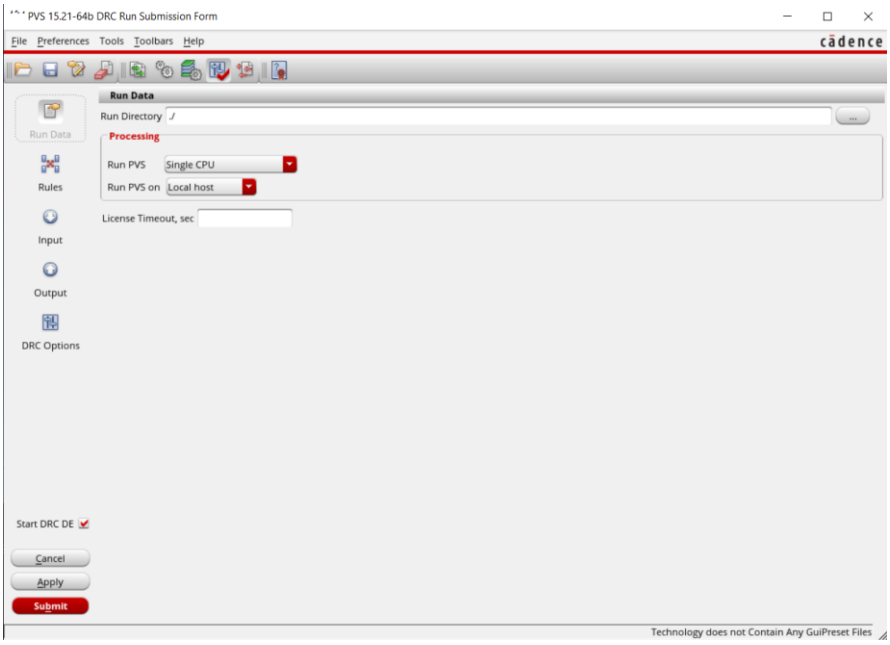
Then the ‘Virtuoso Layout Suit L’ window-

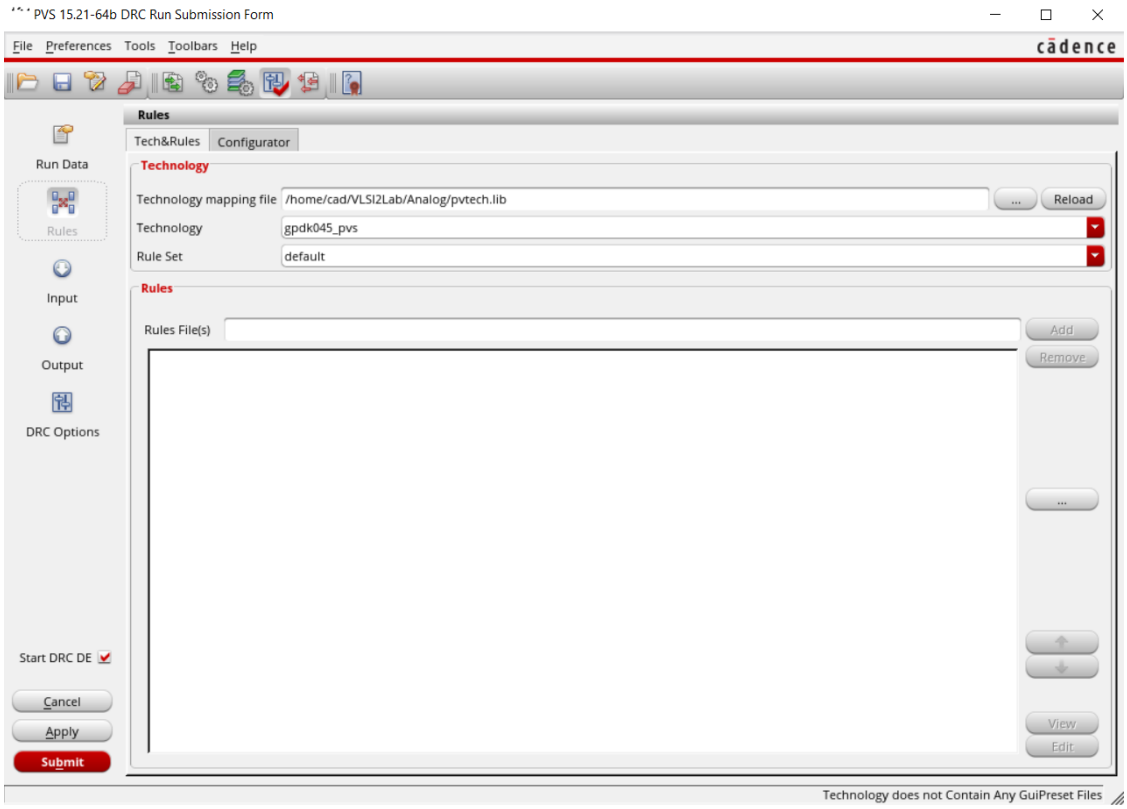


After unticking the ‘Label’ under ‘Objects’ from left side panel, the window-

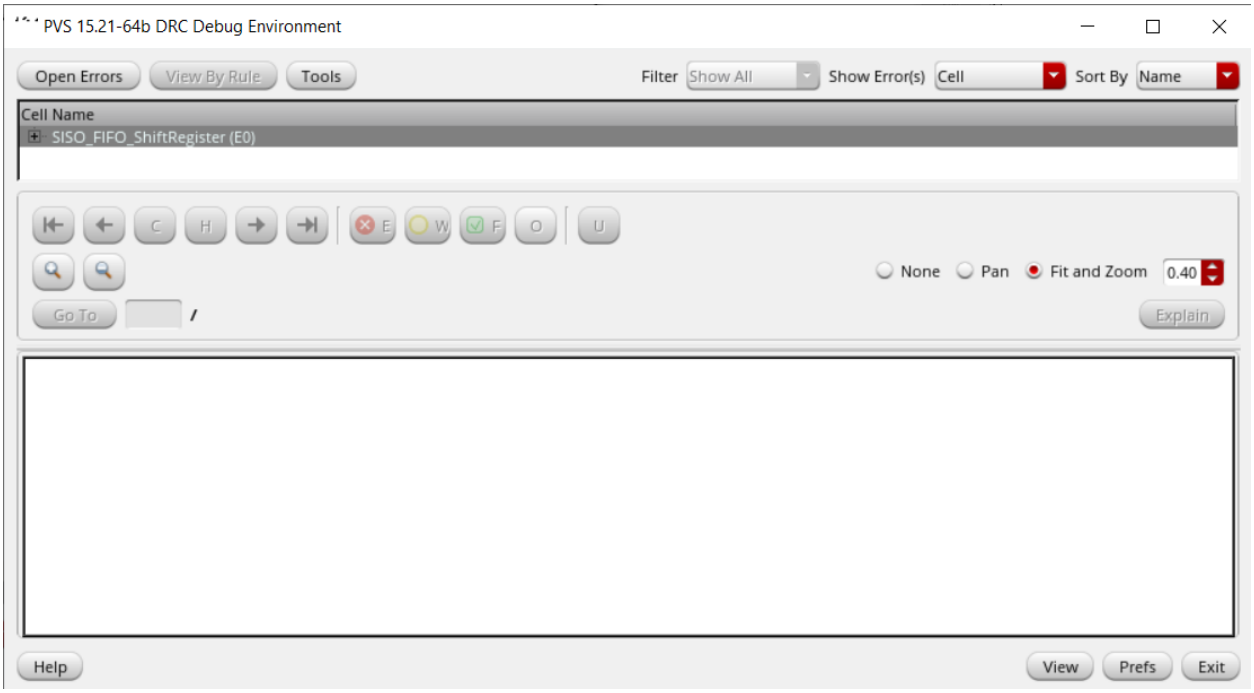


To run DRC the PVS window was set as like-

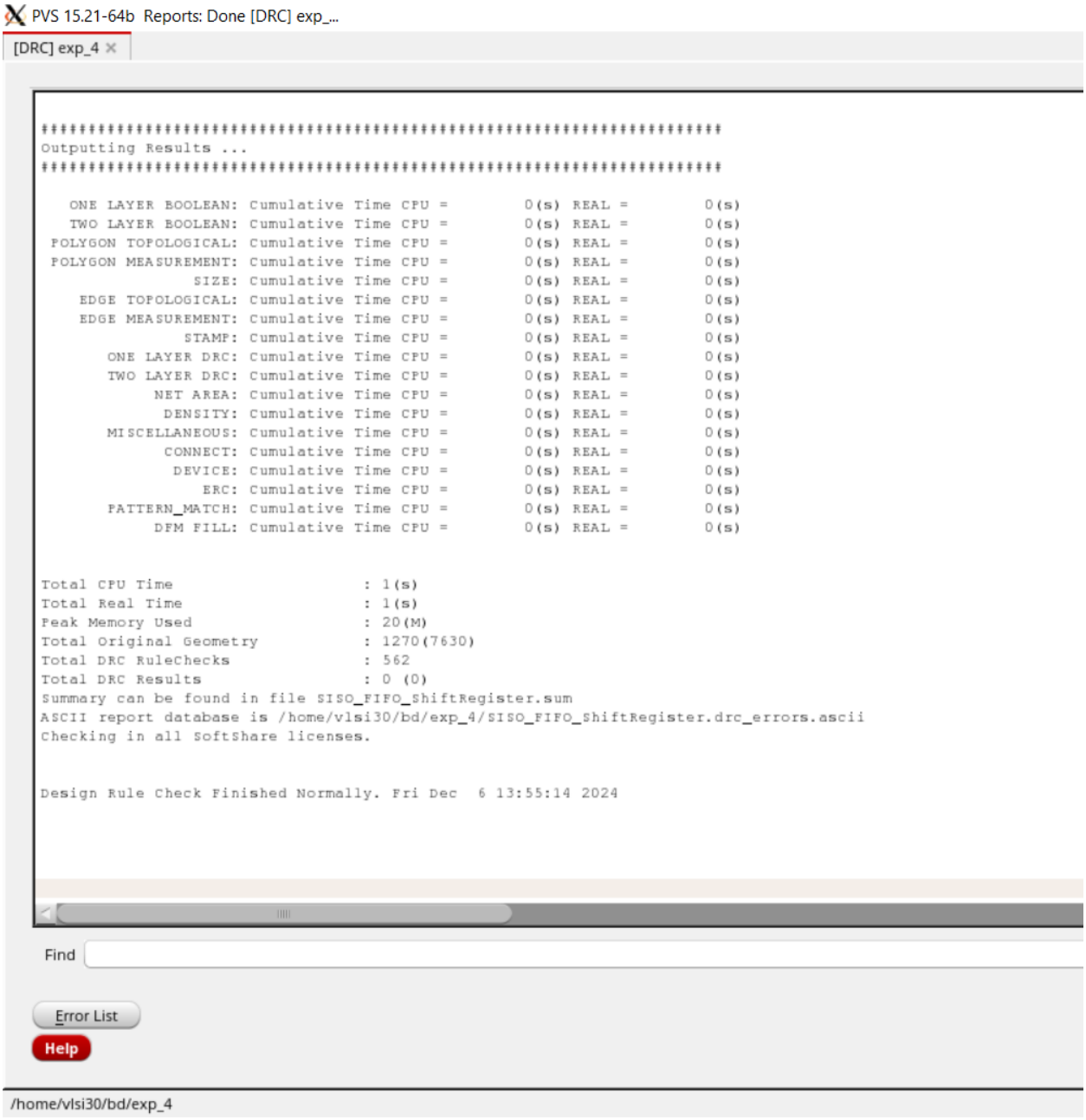




After selecting ‘Submit’ button- (PVS Debug Environment)



(PVS Reports)



As we can see, there is no DRC error in our design.

We finished the parts- physical design, timing, and verification. We started with the synthesized netlist (structural code). This gds can be sent in the semiconductor industry to be manufactured.

Conclusion:

The **16-bit Serial-In Serial-Out (SISO) FIFO Shift Register** is a robust and versatile digital component essential for modern digital systems. Through this project, we successfully designed, implemented, and tested a shift register capable of parallel data loading and bidirectional shifting. The synchronous design ensures predictable and reliable data handling, suitable for applications in serialization, data buffering, and communication systems.

The detailed verification process, including self-checking and layered verification methodologies, confirmed the design's correctness and robustness. Additionally, the front end, the synthesized netlist and back-end design using Cadence tools validated the practical implementation and manufacturability of the shift register.

The project demonstrates a comprehensive understanding of RTL design, synthesis, and physical implementation, paving the way for integration into larger systems or as a standalone module in digital signal processing and memory management.

References:

- Weste N., Harris D. - CMOS VLSI Design_ A Circuits and Systems Perspective-Pearson Education (2004)
- Baker_CMOS_Circuit_Design__Layout__and_Simulation__Second_Edition
- Rabaye_Digital_Integrated_Circuits__A_Design_Perspective