VLSI Design Flow

Problem statement:

If C=00, Take A as single-bit input serially and output into registers parallely
C=01, Take B[0:3] input parallely and output into a register serially
C=10, Take B[0:3] input parallely and output into registers parallely
C=11, NOP

Assumptions and given:

- A is a single bit and B is 4-bit wide.
- The design has a synchronous clock and synchronous reset.
- The inputs and outputs should be registered.
- For C=00, we are considering 4 bit output so after 4 clock cycles 4 single bits from input will be parallely displayed in the output.

Specifications of the design:

- The design should have multiple functionalities *serial in-parallel out*, *parallel in-serial out*, *parallel in-parallel out*.
- These functionalities should be selected by a control signal named '*C*' which is of 2 bits. The last operation/final operation is NOP(nothing happens when C=11).
- If C = 00, serial in-parallel out is selected in which A is taken as single bit input serially and X[0:3] is labeled and taken as output so it takes 4 clock cycles to get a 4 bit parallel output.
- C = 01, parallel in-serial out is selected
- Finally if C =10, parallel in-parallel out is selected which has a parallel input of B[0:3] and output is labeled and given by Z[0:3].
- The design should have a positive edge triggered clock and synchronous reset. So when the reset == 1 when the the selected design's output should be 0.
- So in the above design A, B, C are inputs and X, Y, Z are outputs. These should be registered as in they should be passed through the registers. Apart the normal inputs there is 'loadinputs' which basically loads the inputs to reg so this should be set 1 in the test bench.
- The design used *90nm technology*.
- Constraints are taken in three different variations which is given in synthesis.

Simulation and Coverage analysis:

Verilog code:

```
module SerialParallel(clk, rst,loadinputs,C, A, B, X, Y, Z);
input clk,rst;
input [1:0] C;
input  A;
input loadinputs; //To load inputs to registers
input [3:0] B;
```

```
reg [1:0] Creg; //Registers to which inputs are loaded
reg Areg;       //Registers to which inputs are loaded
reg [3:0] Breg; //Registers to which inputs are loaded
output reg [3:0] X;
output reg [3:0] Z;
output reg Y;
//Syncronous clk and reset
always @(posedge clk)
begin
        if(rst) //checking if reset == 1
        begin
                X <= 4'b0000; //setting all the outputs to zeros
                Z <= 4'b0000;
                Y <= 1'b0;
        end
        if(loadinputs) //Loading inputs to registers.
        begin
                Breg [3:0] <= B [3:0];
                Creg[1:0] <= C[1:0];
                Areg <= A;
        end
        case(Creg)
                2'b00:
                                begin
                                        X[0] <= Areg;
                                        X[1] <= X[0];
                                        X[2] <= X[1];
                                        X[3] <= X[2];
                                end
                2'b01:
                                begin
                                        Y <= Breg[3];
                                        Breg[3] <= Breg[2];
                                        Breg[2] <= Breg[1];
                                      Breg[1] <= Breg[0];
                                end
                2'b10:
                                begin
                                        Z[3] <= Breg[3];
                                        Z[2] <= Breg[2];
                                        Z[1] <= Breg[1];
```

```
                                        Z[0] <= Breg[0];
                                end
                        2'b11: $display("No operation");
                endcase
        end
        endmodule
```

Test benches 1:

```
module test_SP;
// Inputs
reg clk;
reg rst;
reg loadinputs;
reg [1:0] C;
reg A;
reg [3:0] B;

// Outputs
wire [3:0] X;
wire Y;
wire [3:0] Z;

// Instantiate the Design Under Test
SerialParallel uut (
        .clk(clk),
        .rst(rst),
        .loadinputs(loadinputs),
        .C(C),
        .A(A),
        .B(B),
        .X(X),
        .Y(Y),
        .Z(Z)
);

initial begin
        // Initialize Inputs
        clk = 0;
        rst = 0;
        loadinputs = 1;
        C = 0;
        A = 0;
        B = 0;
        #100;
```

```
        end
        initial
        begin
                C = 00;
                #80 C = 01;
                B = 4'b1100;
                #80 C = 10;
                B = 4'b1010;
                #50 C = 11;
                #50 rst = 1;
        end
        always #20 A = ~A;
        always #10 clk = ~clk;
        initial
        #1000 $finish;
endmodule
```

Test benches 2:

```
        module test_SP;

        // Inputs
        reg clk;
        reg rst;
        reg loadinputs;
        reg [1:0] C;
        reg A;
        reg [3:0] B;

        // Outputs
        wire [3:0] X;
        wire Y;
        wire [3:0] Z;

        // Instantiate the Unit Under Test (UUT)
        SerialParallel uut (
                .clk(clk),
                .rst(rst),
                .loadinputs(loadinputs),
                .C(C),
                .A(A),
                .B(B),
```

```
		.X(X),
		.Y(Y),
		.Z(Z)
	);

	initial begin
		// Initialize Inputs
		clk = 0;
		rst = 0;
		loadinputs = 1;
		C = 0;
		A = 1;
		B = 0;

		// Wait 100 ns for global reset to finish
		#100;

		// Add stimulus here

	end
	initial
	begin
		C = 00;
		#80 C = 01;
		B = 4'b1001;
		#80 C = 10;
		B = 4'b1110;
		#50 rst = 1;
	end
	always #20 A = ~A;
	always #10 clk = ~clk;
	initial
	#1000 $finish;
endmodule
```

Test benches 3:

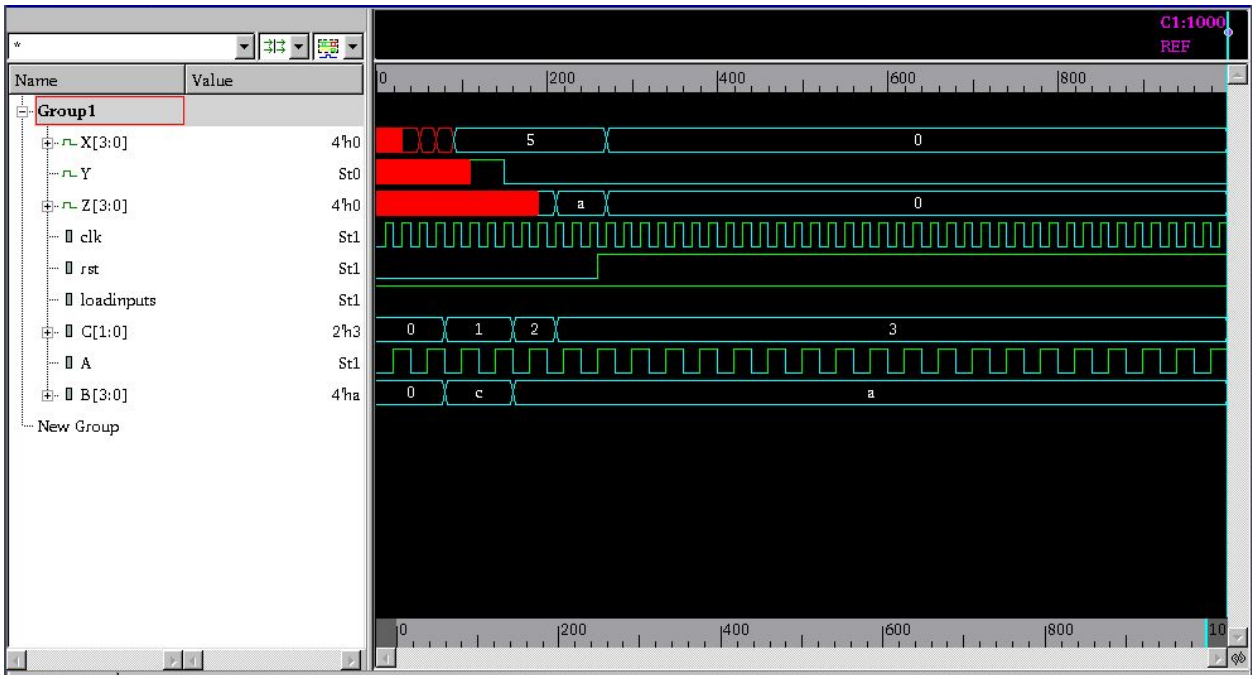
```
	module test_SP;
	// Inputs
	reg clk;
	reg rst;
	reg loadinputs;
	reg [1:0] C;
	reg A;
	reg [3:0] B;

	// Outputs
	wire [3:0] X;
```

```
	wire Y;
	wire [3:0] Z;

	// Instantiate the Design Under Test
	SerialParallel uut (
		.clk(clk),
		.rst(rst),
		.loadinputs(loadinputs),
		.C(C),
		.A(A),
		.B(B),
		.X(X),
		.Y(Y),
		.Z(Z)
	);

	initial begin
		// Initialize Inputs
		clk = 0;
		rst = 0;
		loadinputs = 1;
		C = 0;
		A = 0;
		B = 0;
		#100;

	end
	initial
	begin
		C = 00;
		#80 C = 01;
		B = 4'b0111;
		#80 C = 10;
		B = 4'b1011;
	end
	always #20 A = ~A;
	always #10 clk = ~clk;
	initial
	#1000 $finish;
endmodule
```

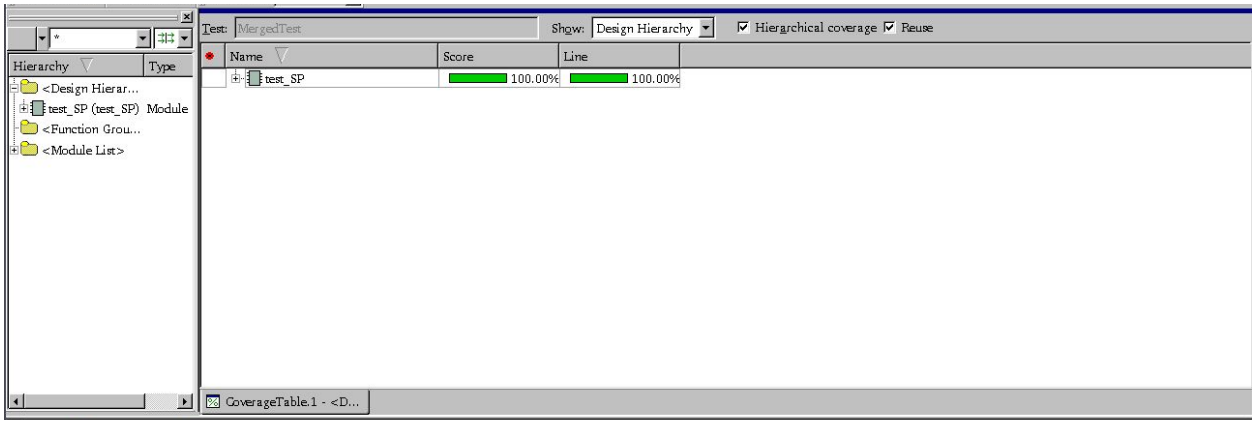
Simulation Results and analysis: I used VCS for simulation of my design which is a Synopsis tools for the above three test benches. The results of the above test benches are as follows:

Test bench 1: In this test bench, we had A toggling from 0 to 1 at a frequency of 20ns and a clock which is operating at 10ns. Initially, C =00 for 80ns i.e 4 clock cycles(to transfer 4 bits to output). Hence after 4 cycles 0101 should be in the output and later on after rst ==1 it should be 0. We can see 5 i.e 0101 and 0 when rst == 1 in the A waveform. Similarly for next 80ns C = 01 for parallel input to serial output which takes 4 clock cycles. For this interval B = 4'1100 hence the Y waveform is 1100 and later on after rst Y = 0. C = 10 takes only one clock cycles to transfer the bits. Since B = 4'b1010 it got directly reflected into Z waveform and it is 0 when rst ==1. The last case does nothing. And this test bench has 100% code coverage since it covers all the case statements 00 to 11 with rst == 0 and 1 at different intervals of clock(In a way it covers all the RTL statements in the code).

Waveforms:

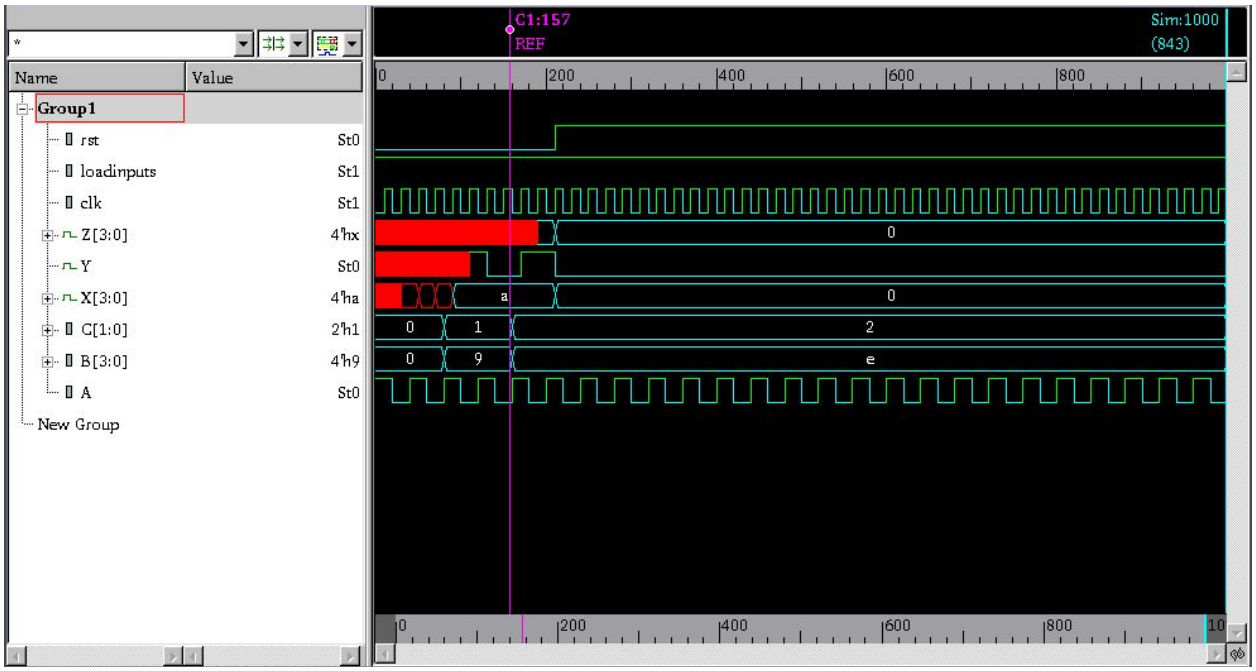


Coverage:

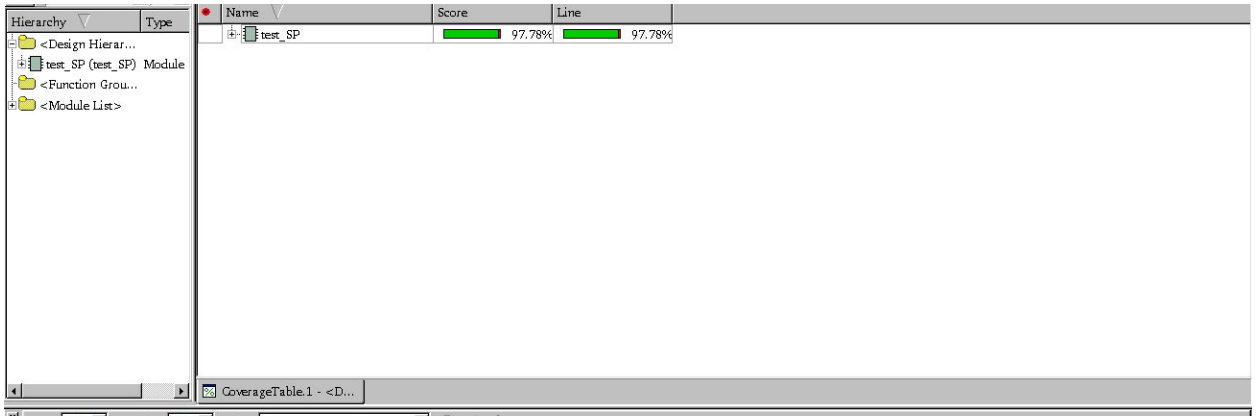


Test bench 2: This test bench has few changes. The data the is being transferred it different in this case. For case C=01 and 10, values of B are 1001, 1110 respectively. And the values of A in initialize as 1 so 1010 is reflected in X rather than 0101. It doesn't cover the last case so it got only 97.7 coverage.

Waveforms:

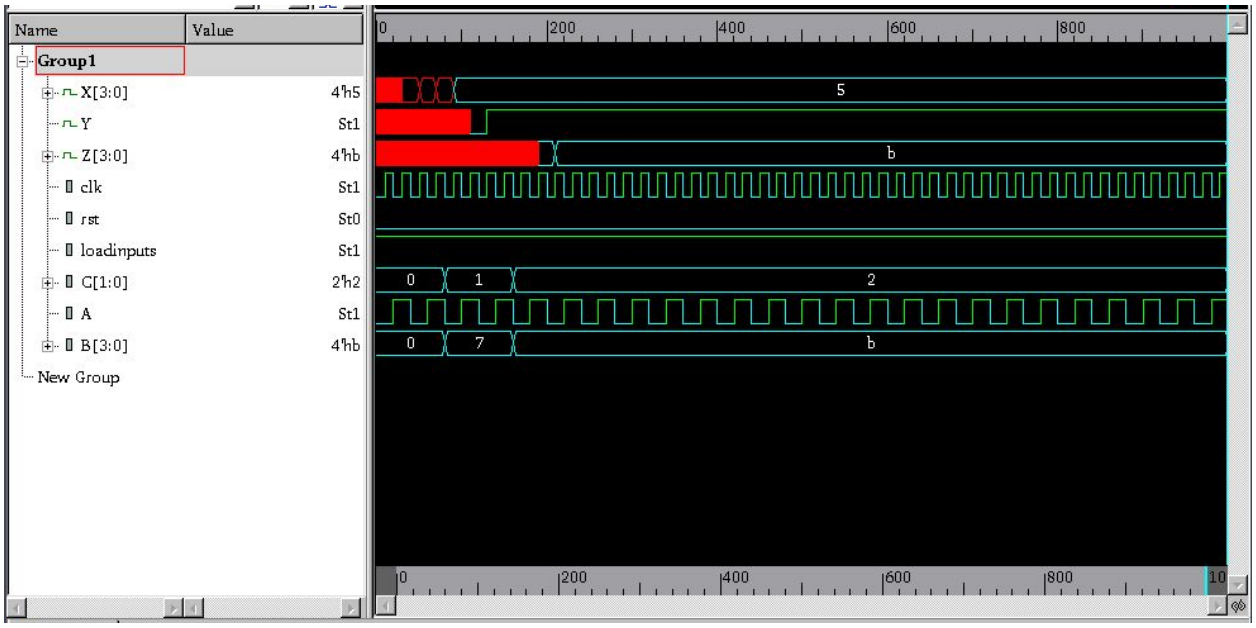


Coverage:



Test Bench 3: Similar to test bench 2, there are minor changes like different inputs for C = 01 and 10. The rst is not applied here. It has a code coverage of 90.7%

Waveforms:



Coverage:

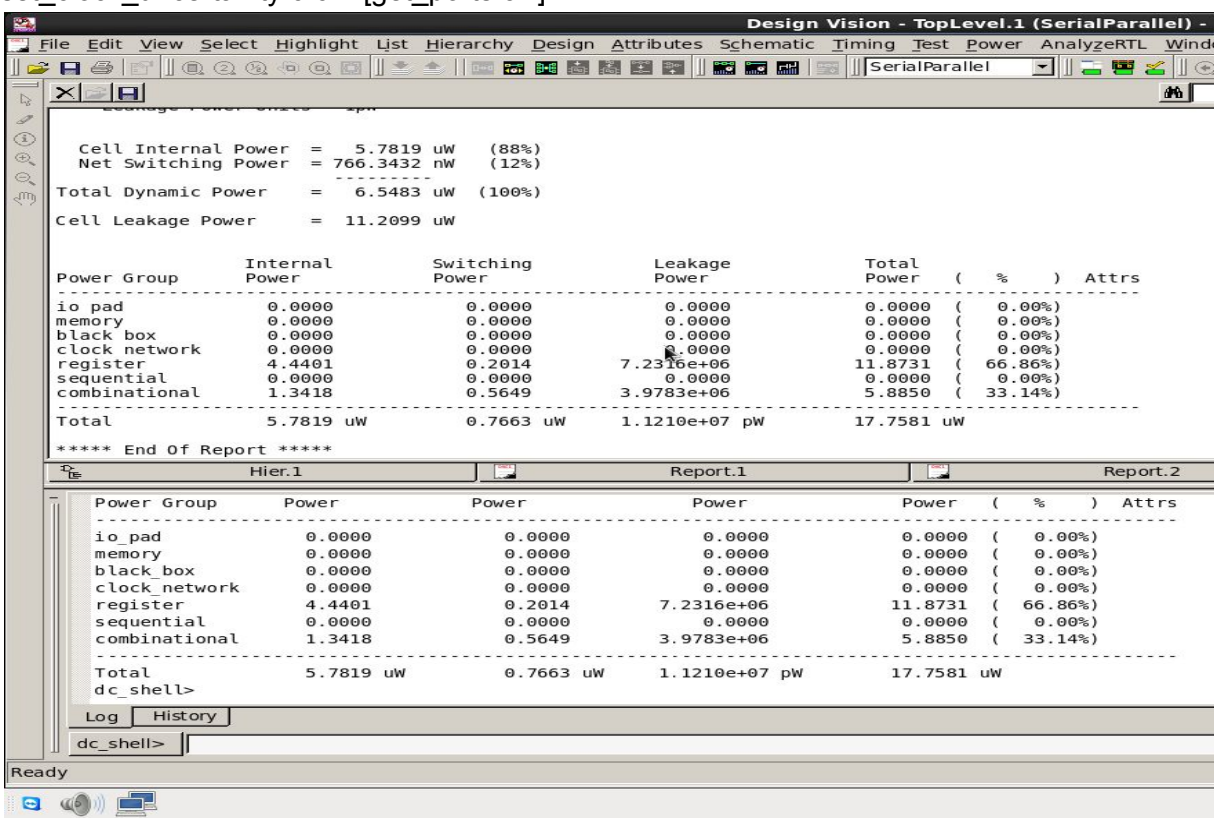
Commands used for VCS:

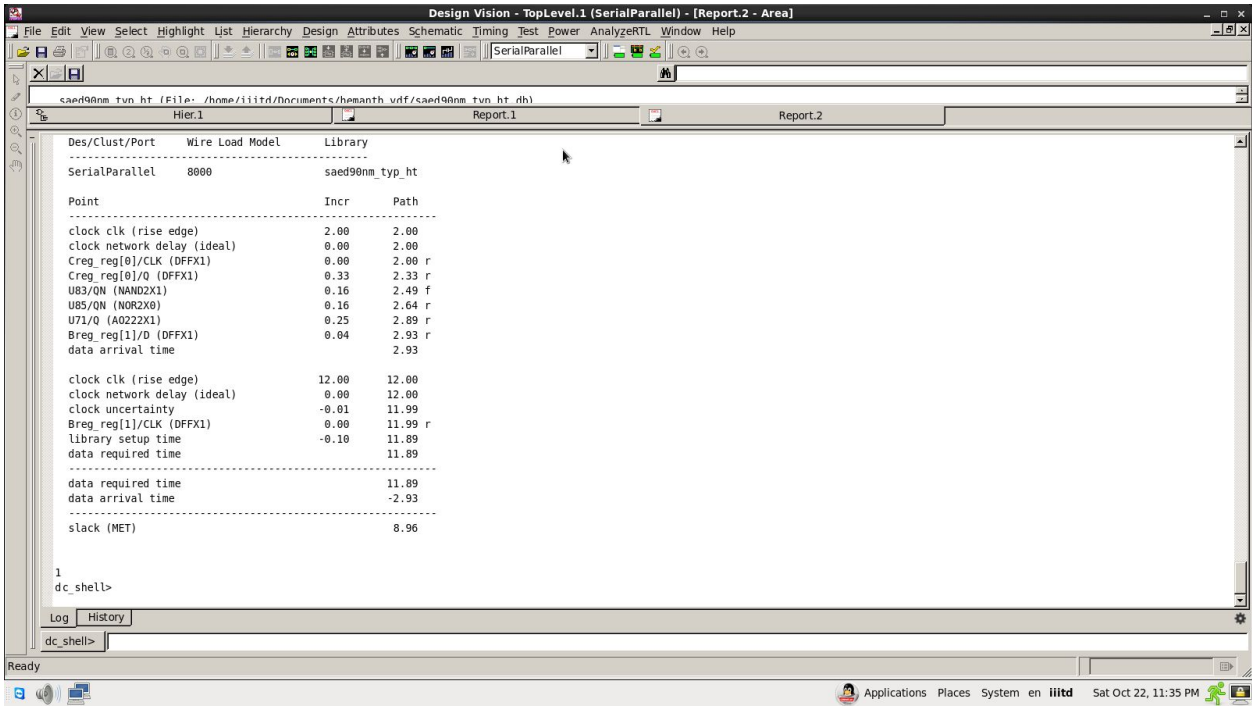
For simulation: vcs -v -R test_SP.v SerialParallel.v -full64 -debug_all
./simv -gui

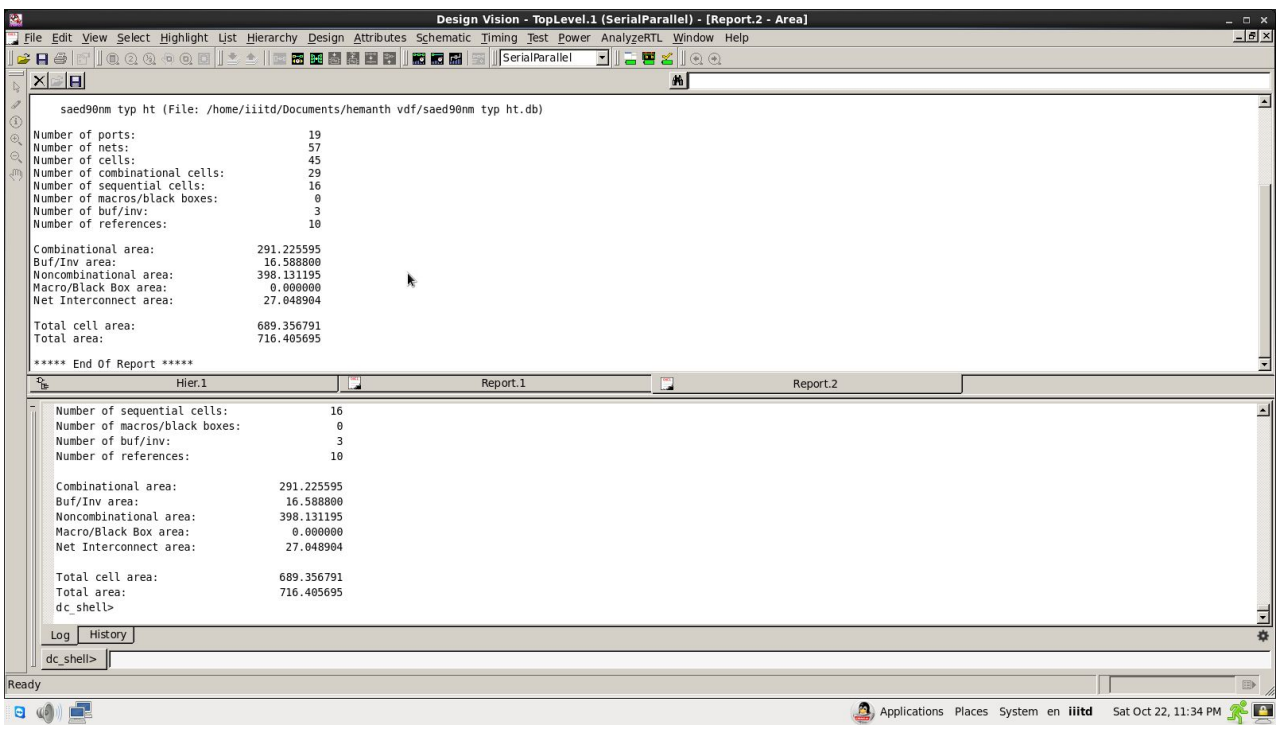
For coverage: vcs -v -R test_SP.v SerialParallel.v -full64 -cm line
./simv -cm line
dve -covdir simv.vdb/ -full64

Synthesis: I used Design compiler for Synthesis of the design. DC is a Synopsis tool. It uses 90nm technology as it's set up. First, area is minimized making timing constraints relaxed. All the results are compared between the area, slack and the power.

Constraints 1:
Minimum Area
```
##########################################################################

# Created by write_sdc on Sat Oct 21 23:47:06 2016

##########################################################################
set sdc_version 2.0

set_units -time ns -resistance MOhm -capacitance fF -voltage V -current uA
set_max_area 717
set_max_fanout 10 [current_design]
create_clock [get_ports clk]  -period 10  -waveform {2 3}
set_clock_transition -max -rise 0.1 [get_clocks clk]
set_clock_transition -max -fall 0.1 [get_clocks clk]
set_clock_transition -min -rise 0.1 [get_clocks clk]
set_clock_transition -min -fall 0.1 [get_clocks clk]
set_input_delay -clock clk  -max 0.01  [get_ports rst]
set_output_delay -clock clk  -max 0.01  [get_ports Y]
set_clock_uncertainty 0.01  [get_ports clk]
```







The above results are when the timing constraints are relaxed so the total area that is coming out to be is approx 716. As the timing constraints are relaxed, kept the values normal. The slack is positive and around 9. And the power is 17.5uW. The netlist is saved as ‘netlist1.v’.

Constraints 2:

Best timing

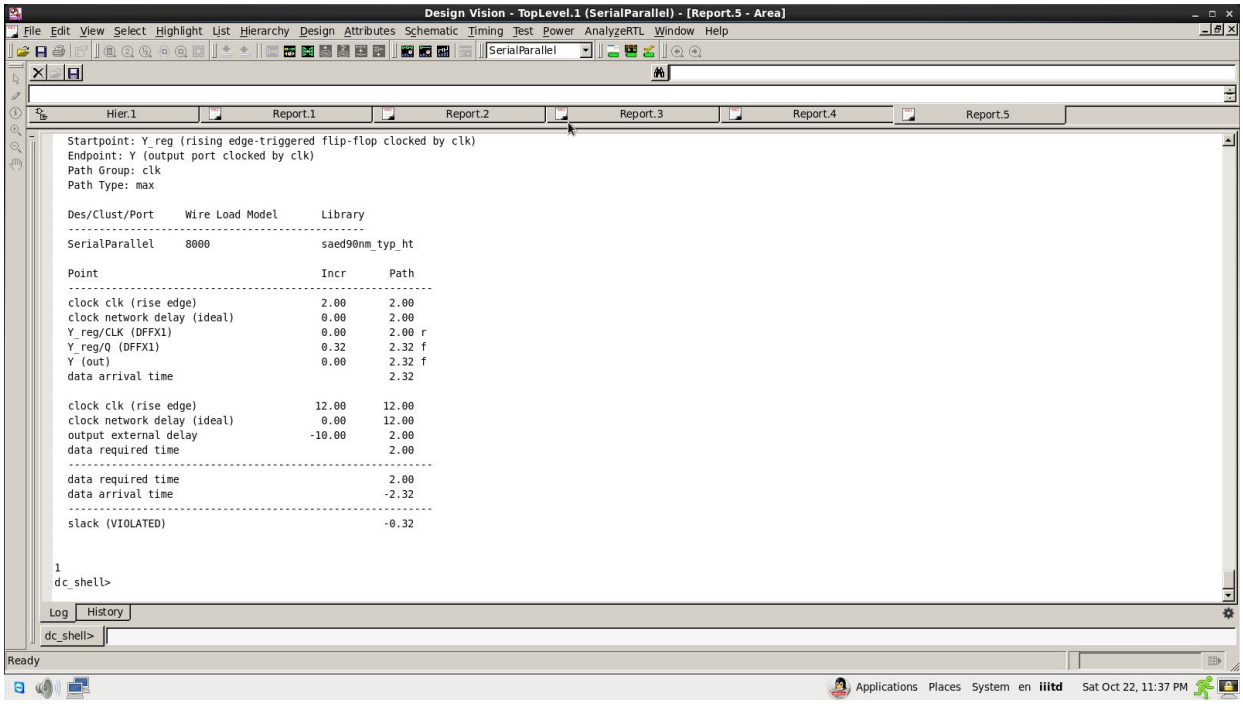
##

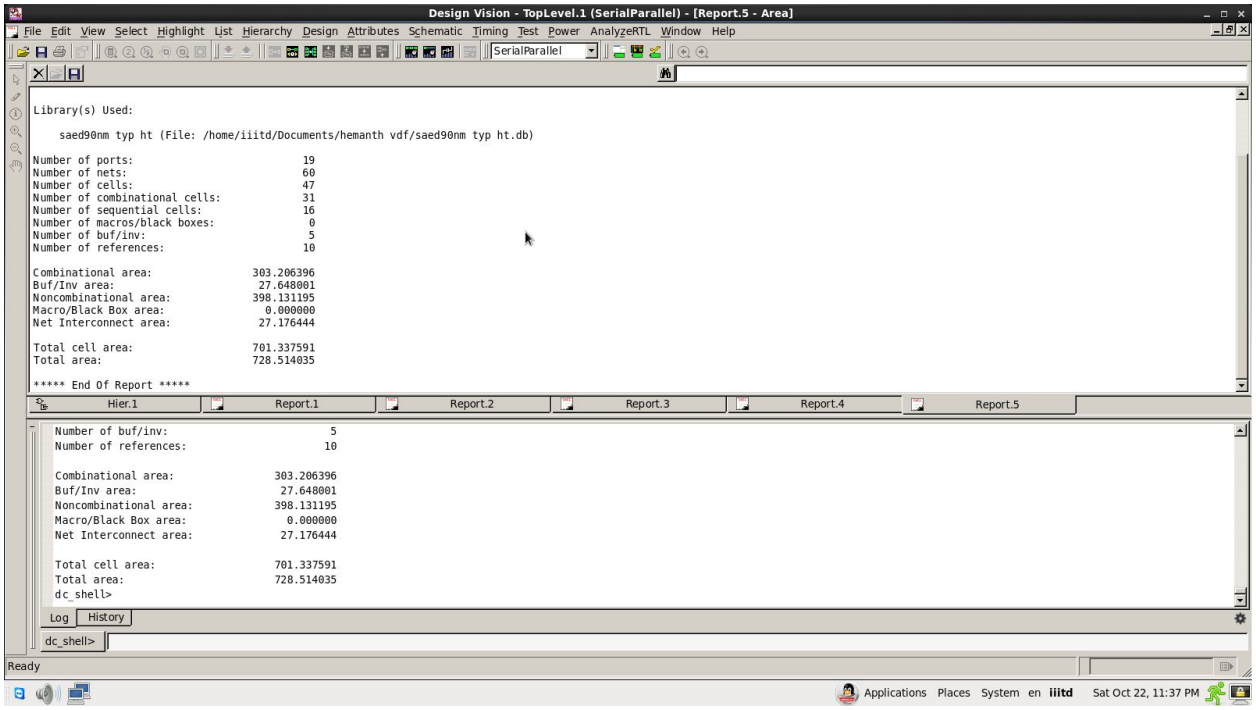
 # Created by write_sdc on Fri Oct 21 17:26:38 2016

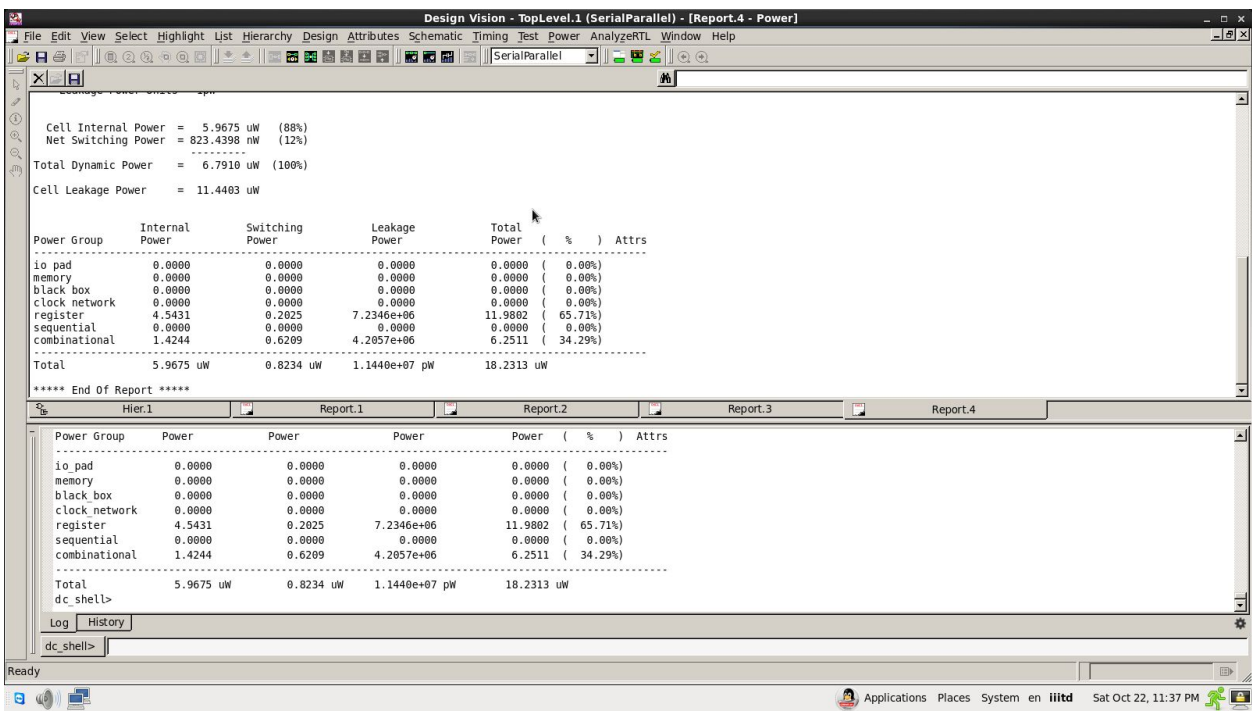
##

set sdc_version 2.0

set_units -time ns -resistance MOhm -capacitance fF -voltage V -current uA
set_max_fanout 10 [current_design]
create_clock [get_ports clk] -period 10 -waveform {2 3}
set_clock_transition -max -rise 0.2 [get_clocks clk]
set_clock_transition -min -rise 0.2 [get_clocks clk]
set_clock_transition -max -fall 0.3 [get_clocks clk]
set_clock_transition -min -fall 0.3 [get_clocks clk]
set_input_delay -clock clk -max 6 [get_ports rst]
set_output_delay -clock clk -max 10 [get_ports Y]
set_clock_uncertainty 0.03 [get_ports clk]





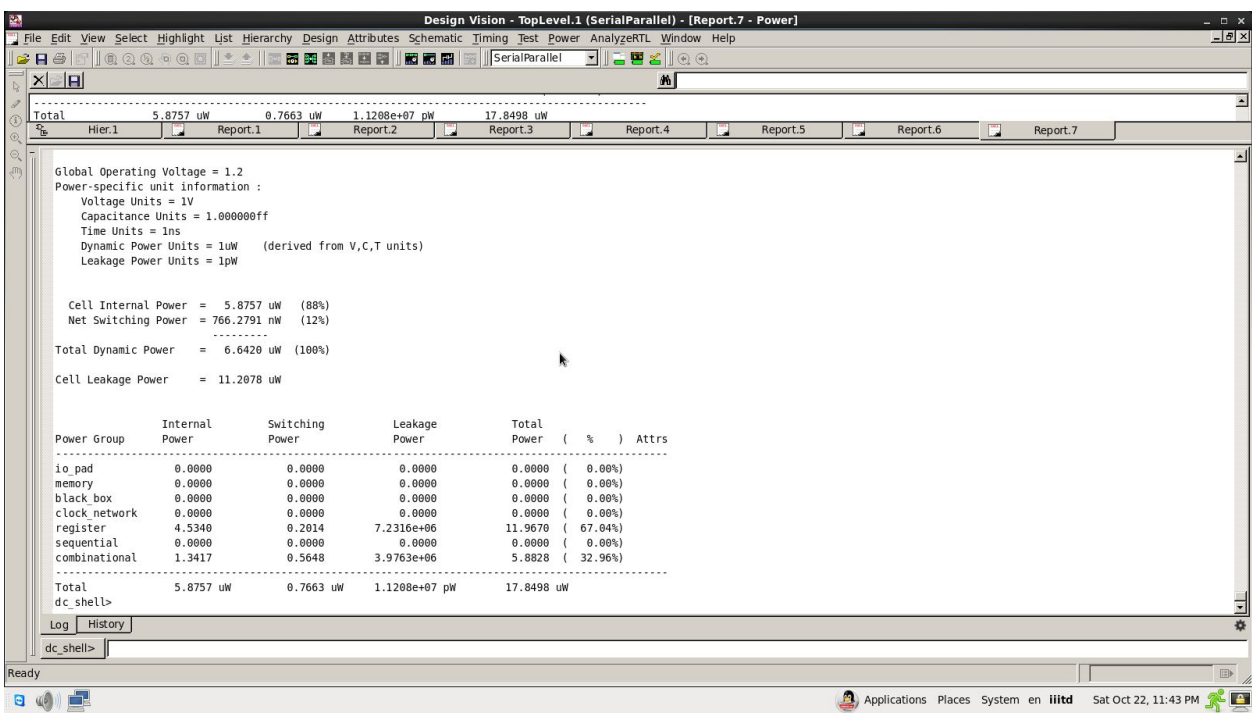


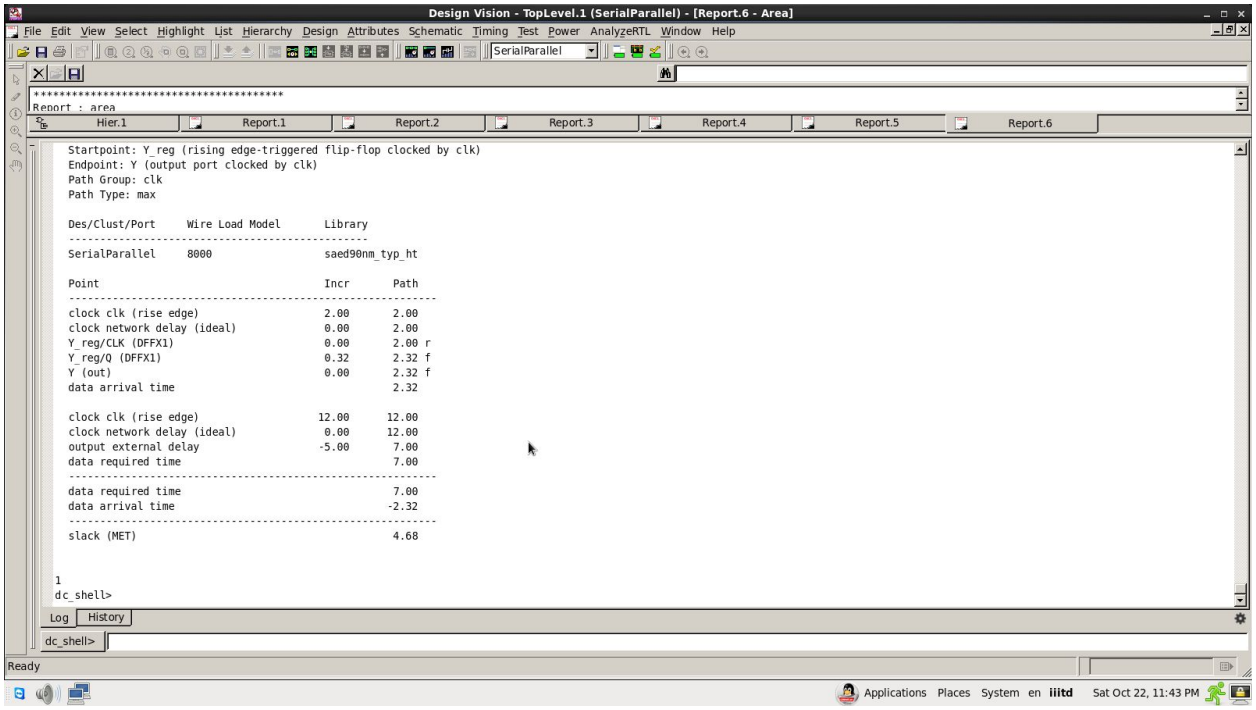
The slack is -0.32 which is calculated by required time - arrival time. As given in the question, the timing constraints are kept in such a way that the slack is negative and for this constraints the area that is coming out to be approx 728 with a power of 18.2uW. The netlist is saved as ‘netlist2.v’

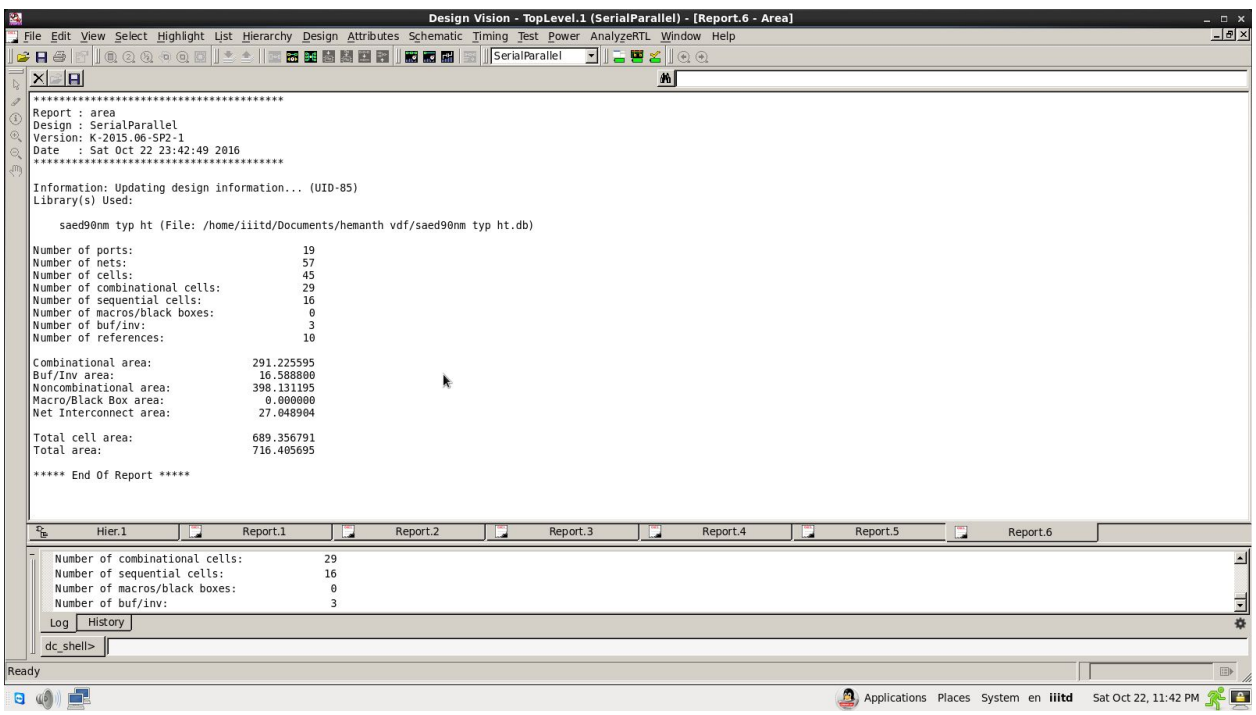
Constraints 3:

Area and timing:

```
##########################################################################

# Created by write_sdc on Fri Oct 21 17:27:49 2016

##########################################################################
set sdc_version 2.0

set_units -time ns -resistance MOhm -capacitance fF -voltage V -current uA
set_max_area -ignore_tns 717
set_max_fanout 10 [current_design]
create_clock [get_ports clk]  -period 10  -waveform {2 3}
set_clock_transition -max -rise 0.2 [get_clocks clk]
set_clock_transition -min -rise 0.2 [get_clocks clk]
set_clock_transition -max -fall 0.3 [get_clocks clk]
set_clock_transition -min -fall 0.3 [get_clocks clk]
set_input_delay -clock clk  -max 6  [get_ports rst]
set_output_delay -clock clk  -max 10  [get_ports Y]
set_clock_uncertainty 0.03  [get_ports clk]
```







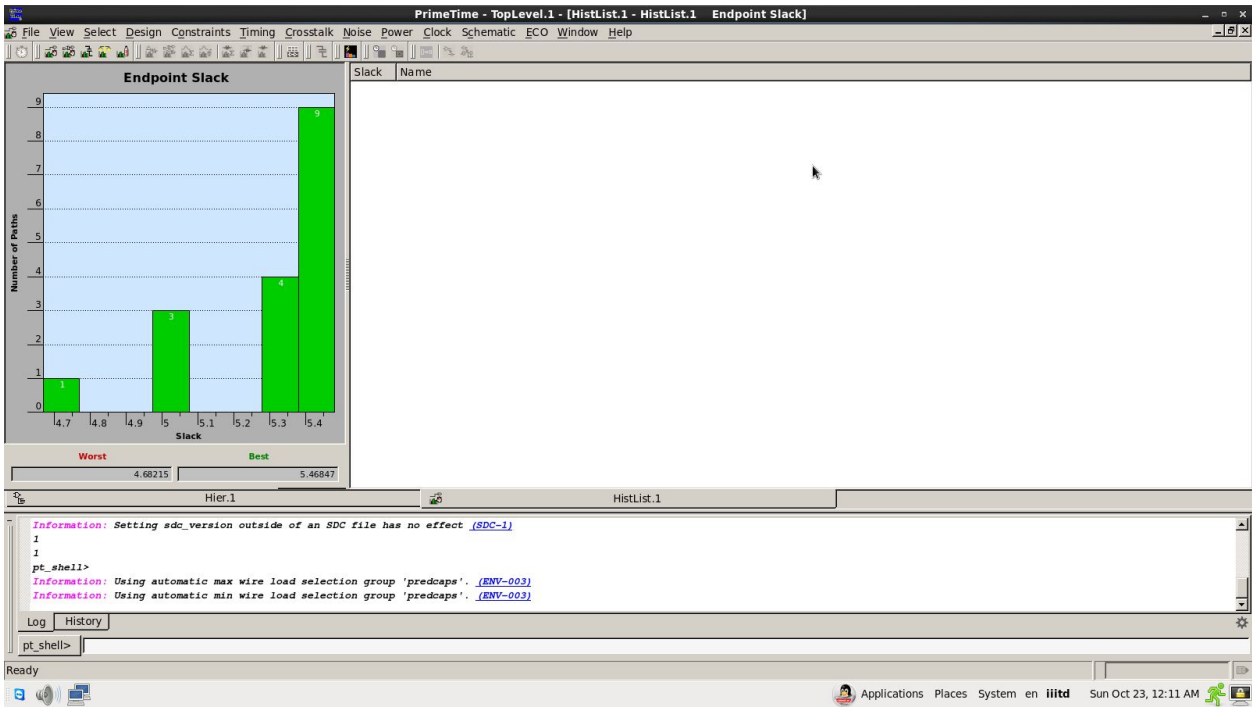
Last constraint deals with positive slack which not too high as constraint 1 and has a area optimization using set_max_area. As we can see the area is optimized from the previous design where didn't try to limit area. This time we tried to optimize the area keeping the using area optimization by disabling total negative slack. Hence the area of the design is a bit optimized as we can see. The netlist is saved as 'netlist3.v'.

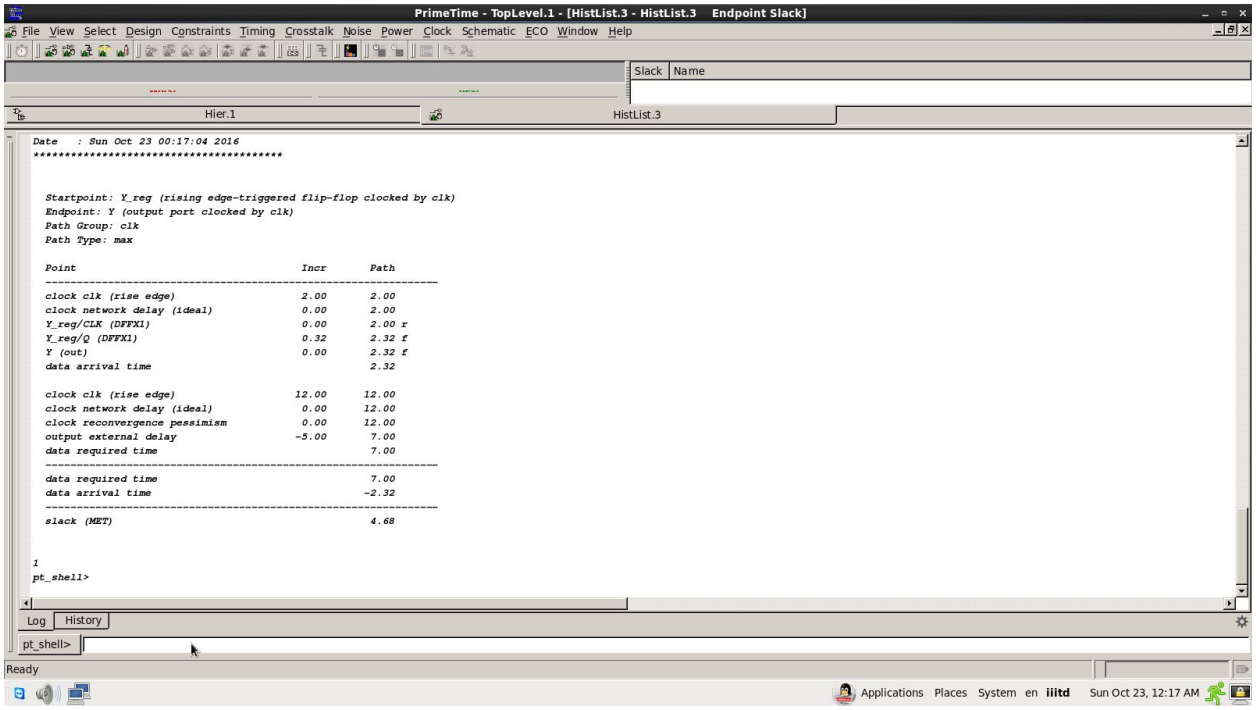
Commands used for Synthesis: dc_shell
start_gui

STA: I am using PrimeTime for STA. Static timing analysis for performed for all the netlists that are generated in the synthesis.

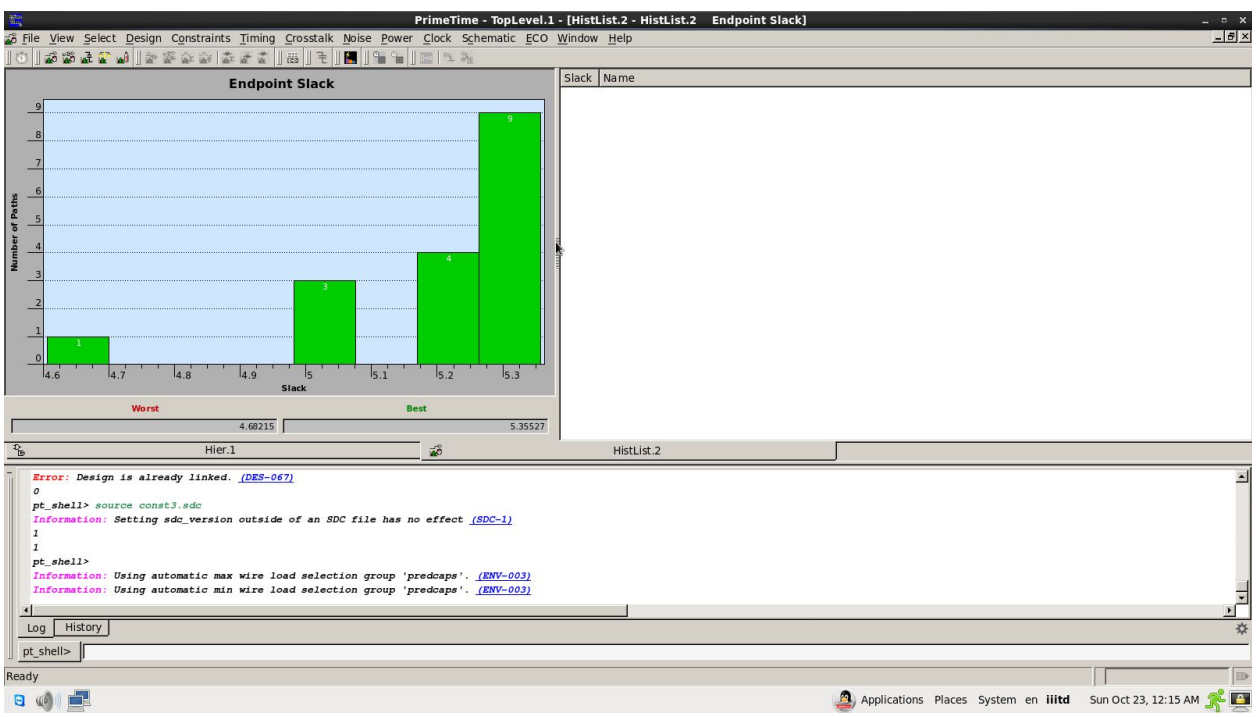
‘

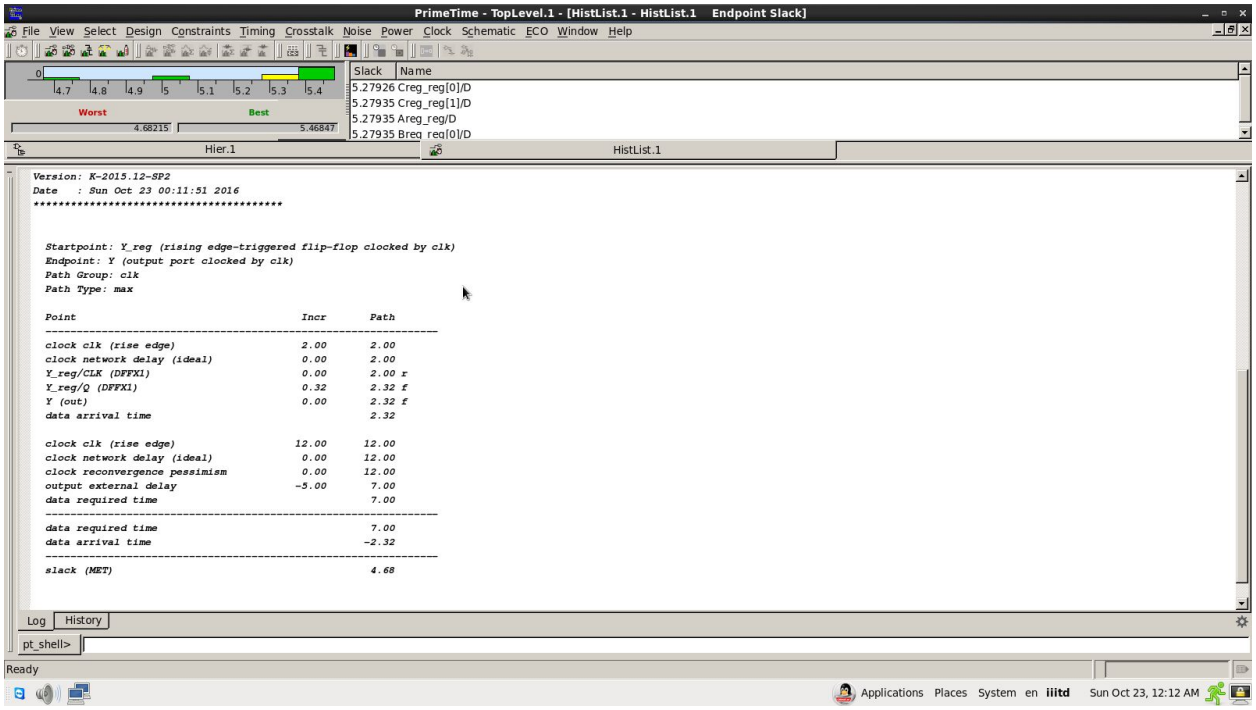
Netlist 1:



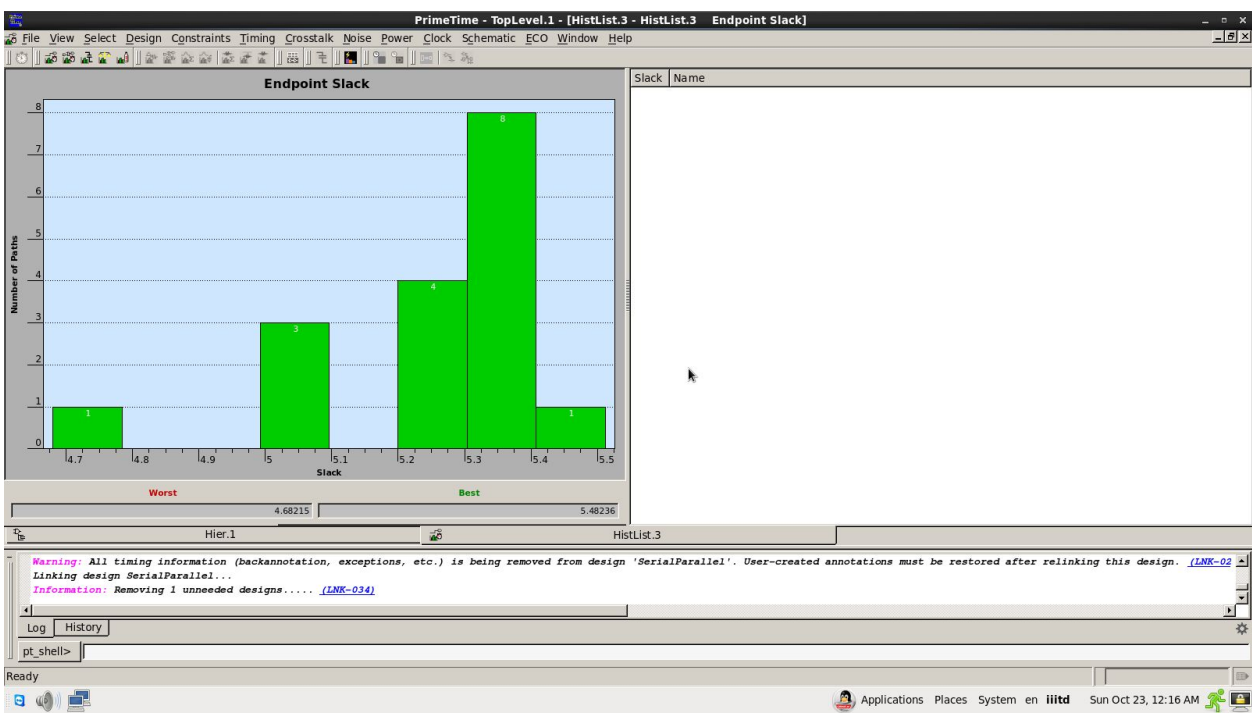

PrimeTime - TopLevel.1 - [HistList.3 - HistList.3 Endpoint Slack]
File View Select Design Constraints Timing Crosstalk Noise Power Clock Schematic ECO Window Help
Slack Name
Hier.1
HistList.3
Date : Sun Oct 23 00:17:04 2016
****
Startpoint: Y_reg (rising edge-triggered flip-flop clocked by clk)
Endpoint: Y (output port clocked by clk)
Path Group: clk
Path Type: max
Point Incr Path
clock clk (rise edge) 2.00 2.00
clock network delay (ideal) 0.00 2.00
Y_reg/CLK (DFFX1) 0.00 2.00 r
Y_reg/Q (DFFX1) 0.32 2.32 f
Y (out) 0.00 2.32 f
data arrival time 2.32
clock clk (rise edge) 12.00 12.00
clock network delay (ideal) 0.00 12.00
clock reconvergence pessimism 0.00 12.00
output external delay -5.00 7.00
data required time 7.00
data required time 7.00
data arrival time -2.32
slack (MET) 4.68
1
pt_shell>
Log History
pt_shell>
Ready
Applications Places System en iiitd Sun Oct 23, 12:17 AM

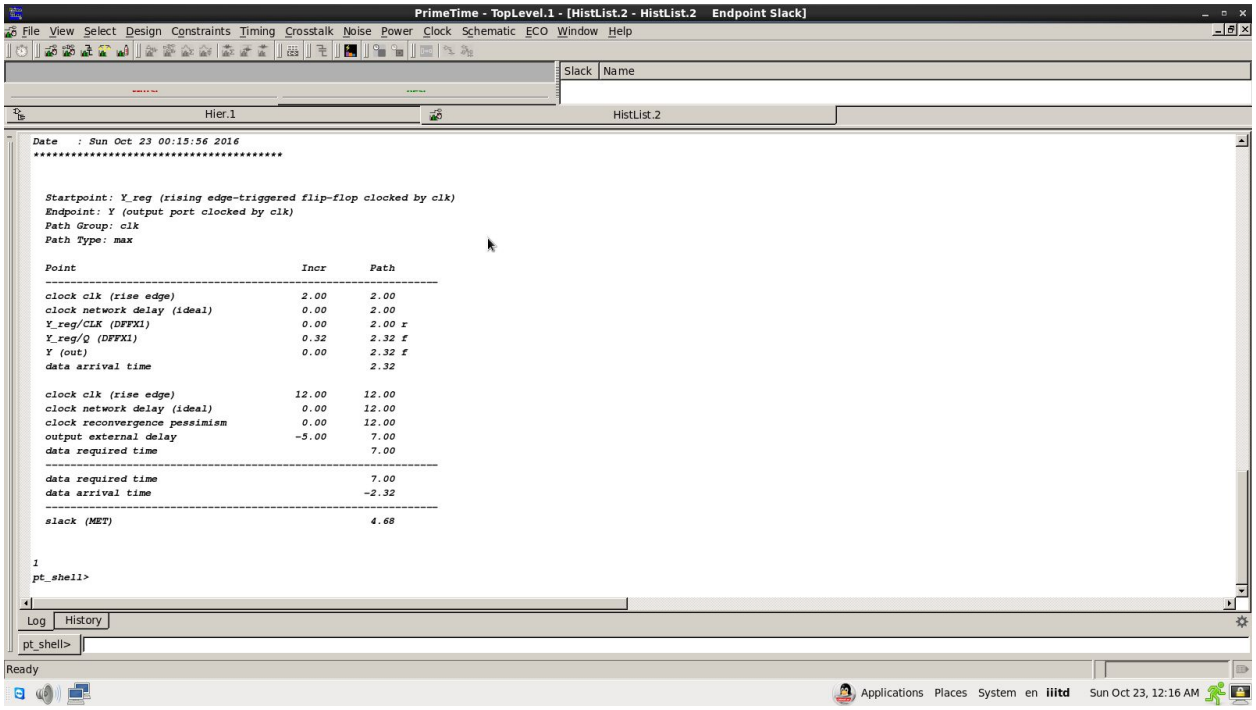
Netlist 2:


PrimeTime - TopLevel.1 - [HistList.2 - HistList.2 Endpoint Slack]
File View Select Design Constraints Timing Crosstalk Noise Power Clock Schematic ECO Window Help
Endpoint Slack
Number of Paths
Slack
Worst
Best
4.68215
5.35527
Slack Name
Hier.1
HistList.2
Error: Design is already linked. (DES-067)
0
pt_shell> source const3.sdc
Information: Setting sdc_version outside of an SDC file has no effect (SDC-1)
1
1
pt_shell>
Information: Using automatic max wire load selection group 'predcaps'. (ENV-003)
Information: Using automatic min wire load selection group 'predcaps'. (ENV-003)
Log History
pt_shell>
Ready
Applications Places System en iiitd Sun Oct 23, 12:15 AM


PrimeTime - TopLevel.1 - [HistList.1 - HistList.1 Endpoint Slack]
File View Select Design Constraints Timing Crosstalk Noise Power Clock Schematic ECO Window Help
Worst
Best
4.68215
5.46847
Slack Name
5.27926 Creg_reg[0]/D
5.27935 Creg_reg[1]/D
5.27935 Areg_reg/D
5.27935 Breg_reg[0]/D
Hier.1
HistList.1
Version: K-2015.12-SP2
Date : Sun Oct 23 00:11:51 2016
Startpoint: Y_reg (rising edge-triggered flip-flop clocked by clk)
Endpoint: Y (output port clocked by clk)
Path Group: clk
Path Type: max
Point Incr Path
clock clk (rise edge) 2.00 2.00
clock network delay (ideal) 0.00 2.00
Y_reg/CLK (DFFX1) 0.00 2.00 r
Y_reg/Q (DFFX1) 0.32 2.32 f
Y (out) 0.00 2.32 f
data arrival time 2.32
clock clk (rise edge) 12.00 12.00
clock network delay (ideal) 0.00 12.00
clock reconvergence pessimism 0.00 12.00
output external delay -5.00 7.00
data required time 7.00
data required time 7.00
data arrival time -2.32
slack (MET) 4.68
Log History
pt_shell>
Ready
Applications Places System en iiitd Sun Oct 23, 12:12 AM

Netlist 3:


PrimeTime - TopLevel.1 - [HistList.3 - HistList.3 Endpoint Slack]
File View Select Design Constraints Timing Crosstalk Noise Power Clock Schematic ECO Window Help
Endpoint Slack
Number of Paths
Slack
Worst
Best
4.68215
5.48236
Slack Name
Hier.1
HistList.3
Warning: All timing information (backannotation, exceptions, etc.) is being removed from design 'SerialParallel'. User-created annotations must be restored after relinking this design. (LNK-02
Linking design SerialParallel...
Information: Removing 1 unneeded designs..... (LNK-034)
Log History
pt_shell>
Ready
Applications Places System en iiitd Sun Oct 23, 12:16 AM



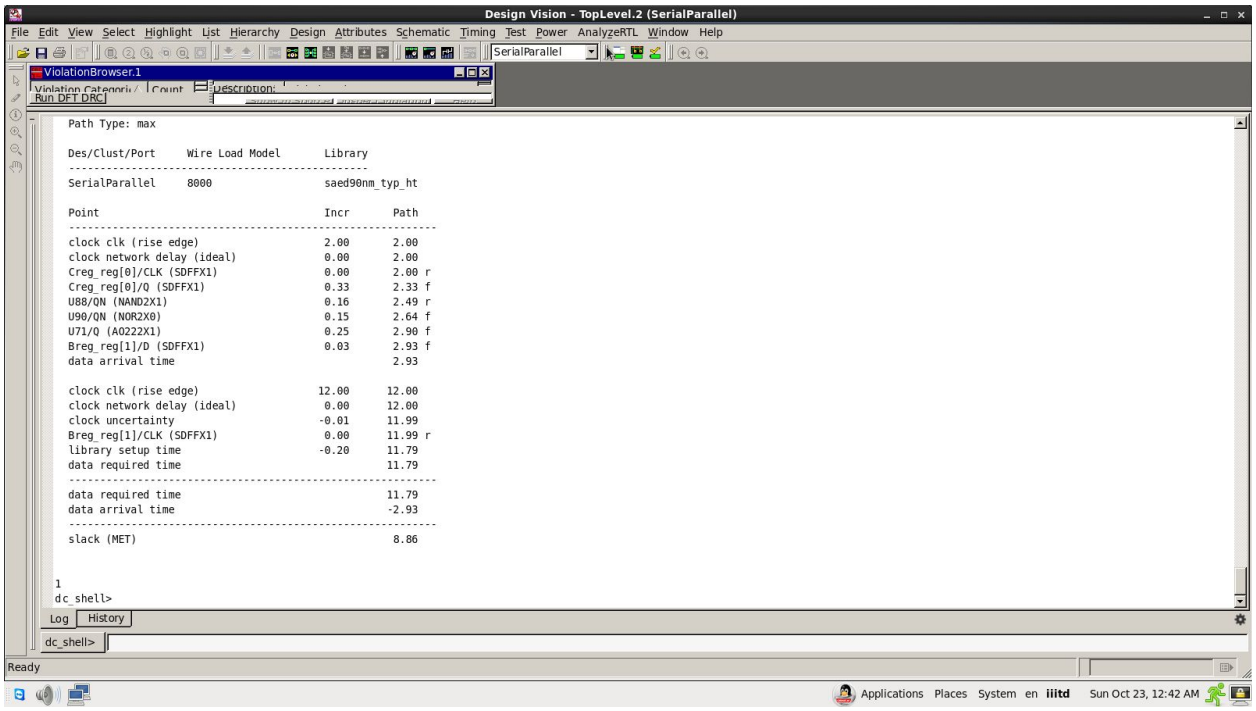
Analysis: report_timing gives the worst slack by default since same constraint file used for all the netlist the worst slack is same for all which is coming out to be 4.68. We can also see the startpoint and endpoint of the worst case slack in the timing report. In case of a negative slack we can inspect the path and schematic try to add a buffer or something to fix it. Since the netlist are different the best slack and the remaining paths has different slacks. The constraints used are initially to set the link to library ad read the netlist. Later on we need to link the library files to netlist and source the constraint file and start gui.

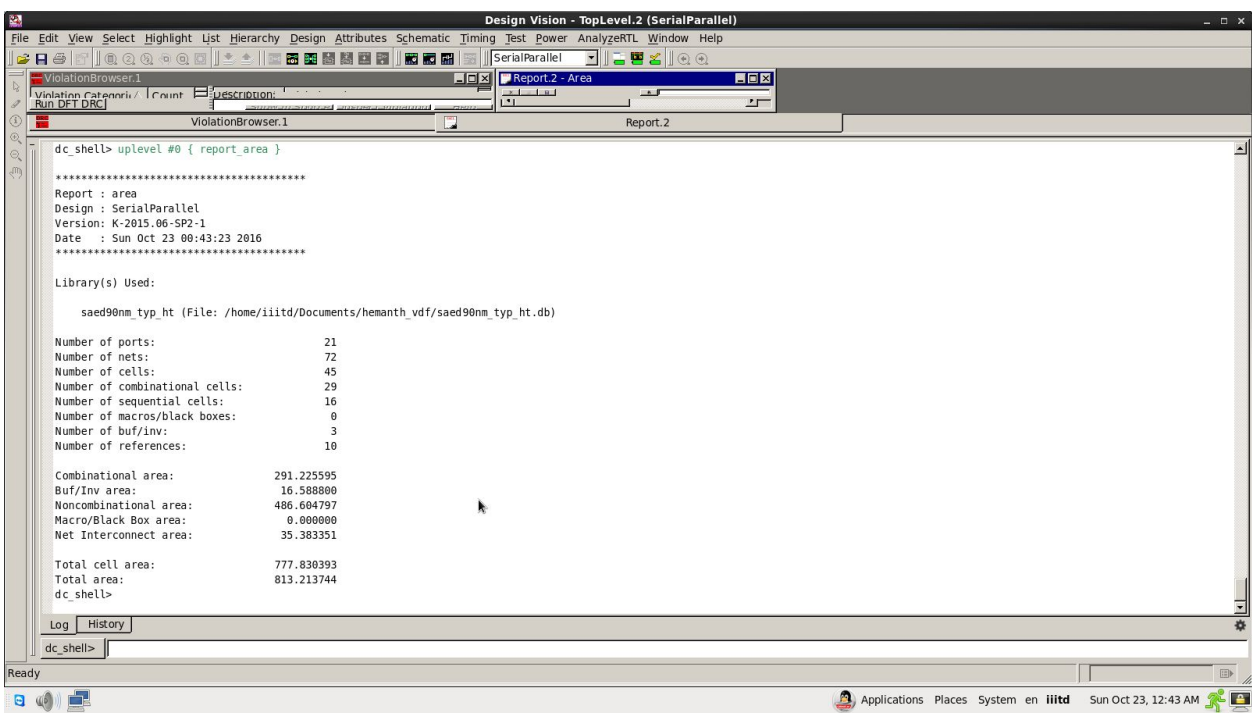
Commands used for PrimeTime:

```
set link_library {* path}
read_verilog netlist.v
link
source constraints.sdc
start_gui
```

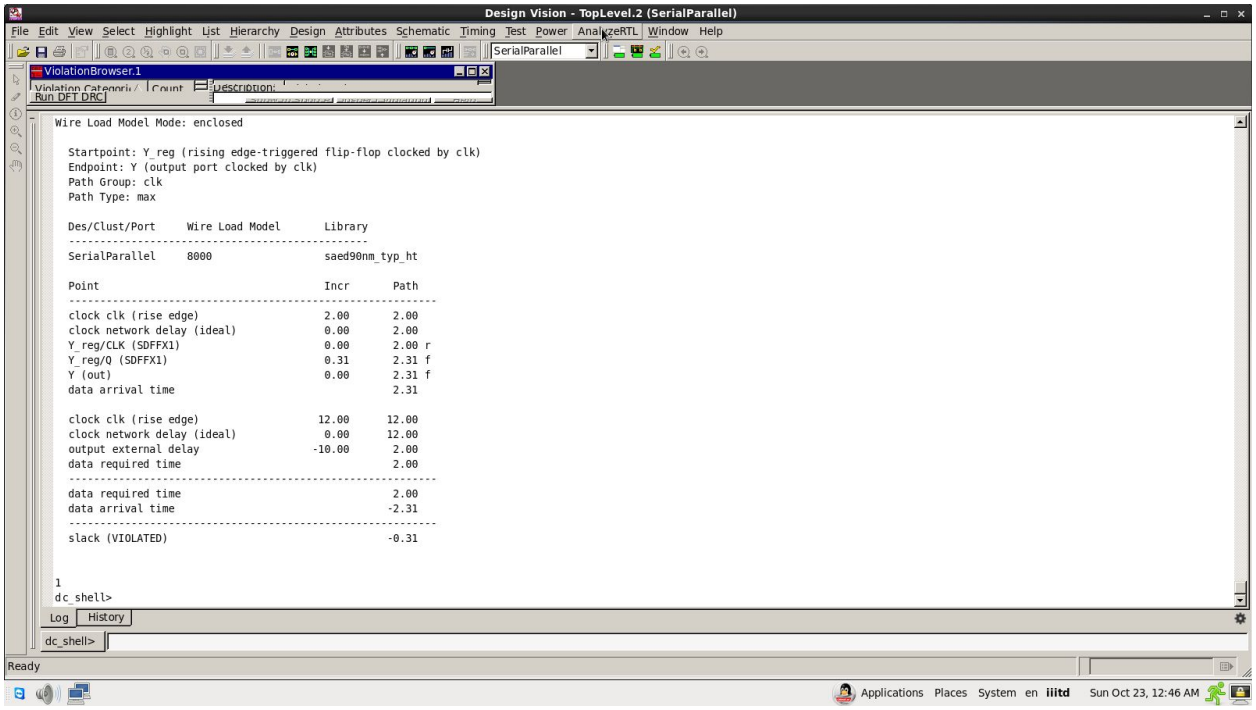
DFT: The tool used to perform DFT is Design Compiler. I performed DFT for all the constraints that were created in step 3. Area and timing of the previous and new netlist is compared.

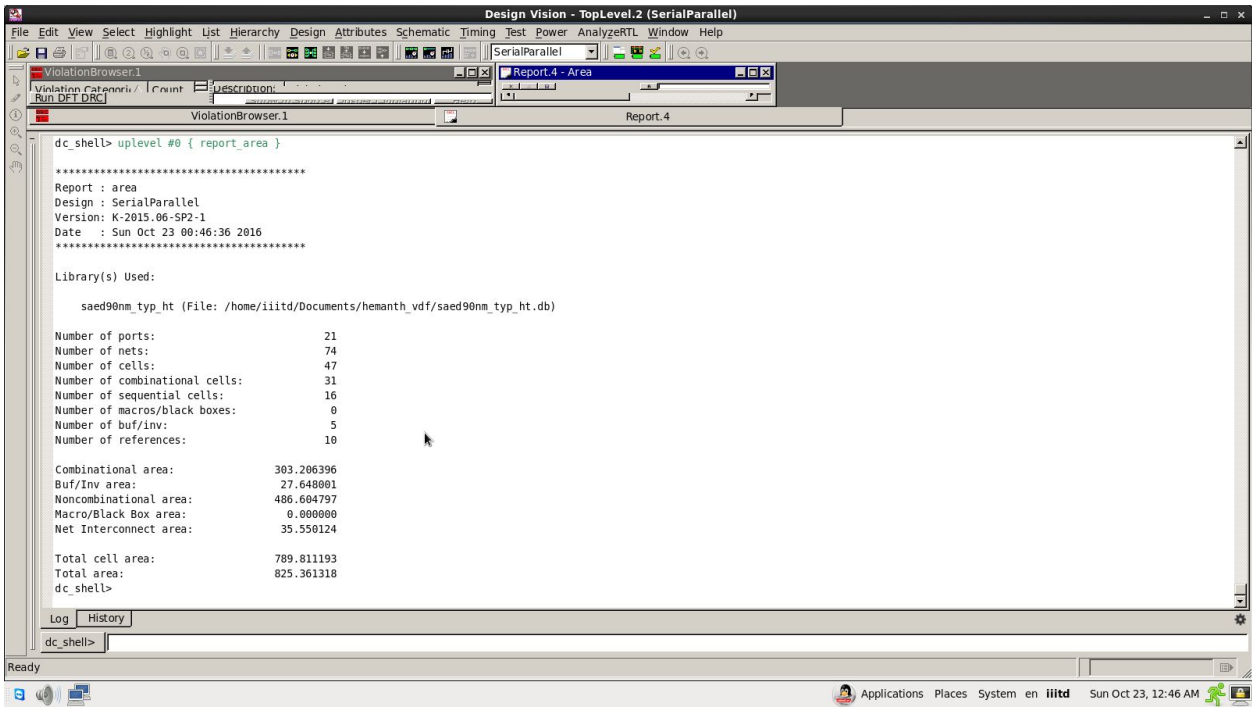
Constraints 1:



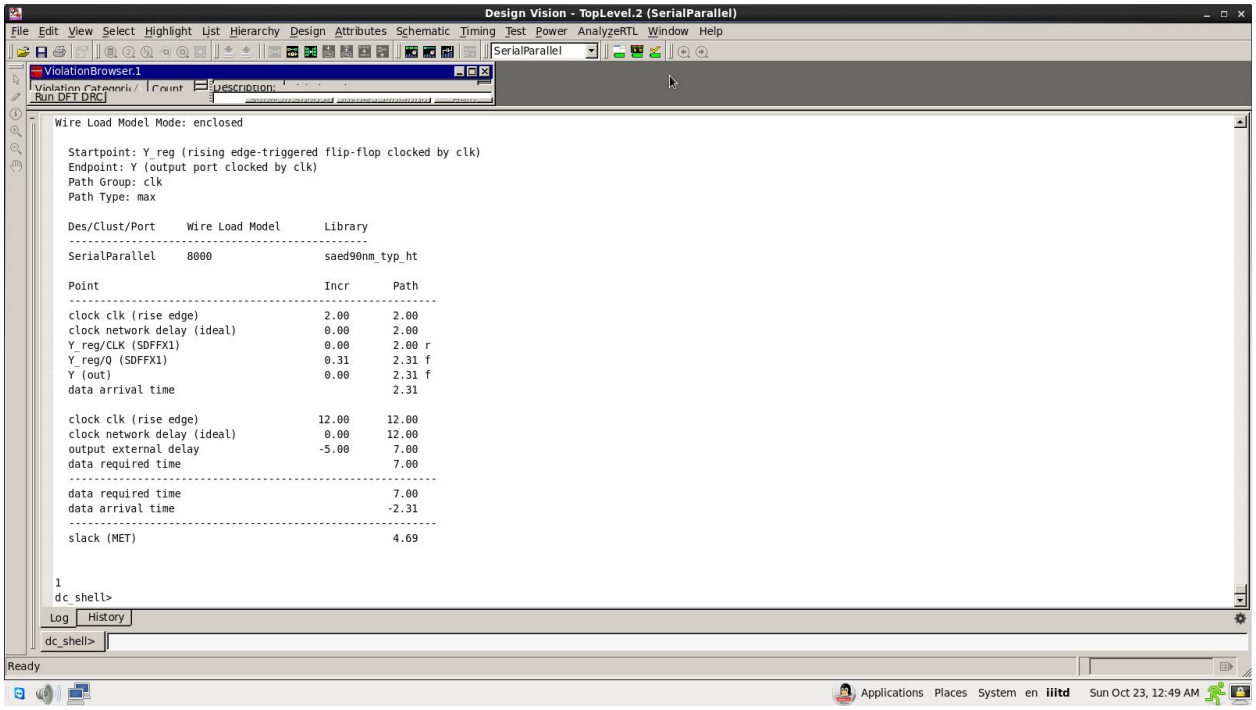


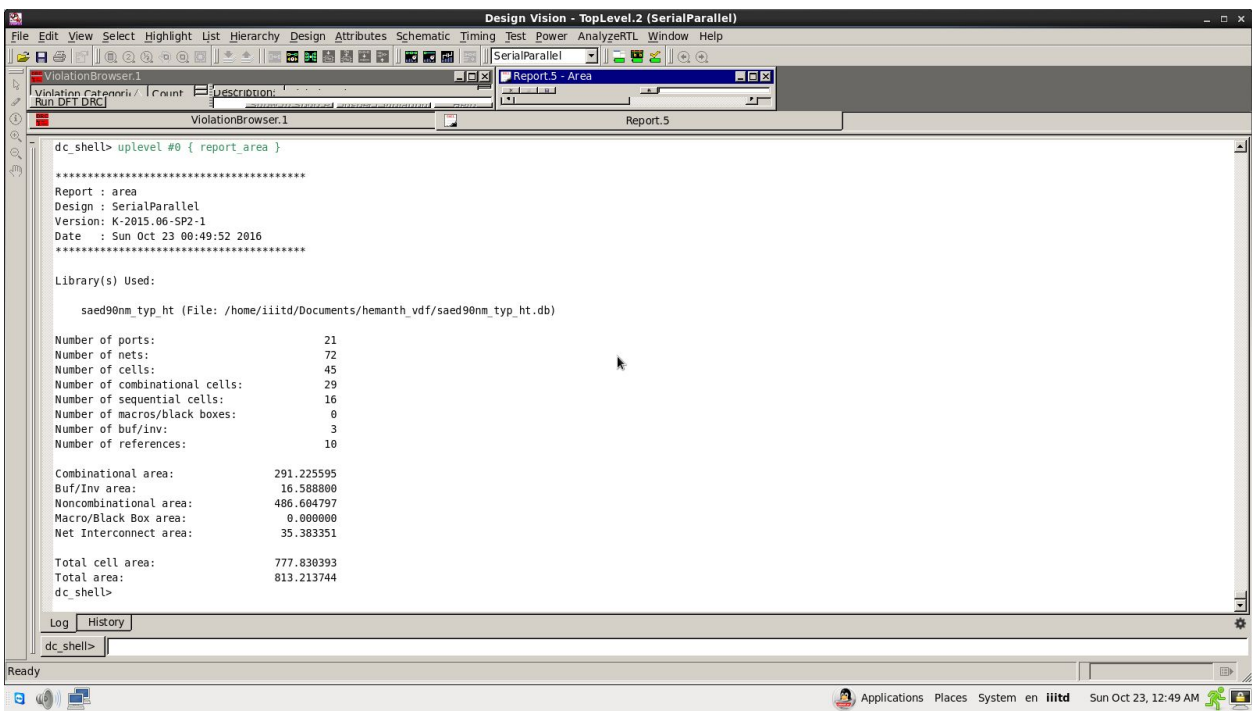
Constraints 2:





Constraints 3:





Analysis: As we can see the slack of the design is earlier almost the same or a bit optimize. Of course the area of the design should be increased to some extent because we are introducing scan flipflops into the design which increases the area. This is applied to all the design comparison above from constraint 1 to constraint 3.

Commands used for DFT:

```
set_scan_configuration -style multiplexed_flip_flop
compile -scan
set_dft_signal -view existing_dft -type ScanClock -port clk -timing [list 40 60]
create_test_protocol
dft_drc
set_scan_configuration -chain_count 1
preview_dft
 insert_dft
```