



INDRAPRASTHA INSTITUTE *of*
INFORMATION TECHNOLOGY
DELHI

Department
of
Electronics & Communication Engineering

VLSI DESIGN FLOW | ECE-513

Dr. Sneh Saurabh

RTL to GDS Flow Automation
Project Part-I

Group Number 18
Animesh Pareek 2021131
Krishna Ayyagari 2021158
Rajat Vatwani 2021186

TABLE OF CONTENTS

1. Detailed Specifications:	6
Inputs:	6
Outputs:	7
Description:	8
Block Design:	9
Assumptions:	9
2. Verilog Code, Test Bench, and Code Coverage	10
Command used:	10
Verilog Code:	11
Module mesh	11
Module master	21
Module Processing_Unit	44
Module router	50
Test Bench 1:	59
Verilog Code	59
Explanation of test vectors: Here we have given Clock, reset and processor configuration signals to test the functionality of our code. This testbench is meant for only functionality testing and not for code coverage. We have configured our Test vectors in such a way that we can firstly test our acknowledgement protocol and the overall working of Processing units, router circuitry and the master module.	
Multiple Test Vectors in this case are tested to ensure structural completeness and the functional correctness of the overall code.	65
Output Waveform for Test Bench 1:	65
Code Coverage Report for Test Bench 1:	66
Toggle Coverage:	66
Block Coverage:	67
Explanation of coverage:	67
Test Bench 2:	68
Verilog Code	68
Explanation of test vectors: Here we have given Clock, reset and processor configuration signals to test the functionality of our code. This testbench is meant for only master module coverage. We have configured our Test vectors in such a way that we can firstly test our acknowledgement protocol and the overall working of the master module. Multiple Test Vectors in this case are tested to ensure structural completeness and the functional correctness of the overall code. This is to test coverage as well as the functionality of the testbench.	
Output Waveform for Test Bench 2:	73

Code Coverage Report for Test Bench 2:	74
Toggle Coverage:	75
Block Coverage:	75
Explanation of coverage:	76
Test Bench 3:	77
Verilog Code	77
Explanation of test vectors: Here we have given Clock, reset and processor configuration signals to test the functionality of our code. This testbench is meant for top module coverage. We have configured our Test vectors in such a way that we can firstly test our acknowledgement protocol and the overall working of the master module. Multiple Test Vectors in this case are tested to ensure structural completeness and the functional correctness of the overall code. This is to test coverage as well as the functionality of the testbench.	86
Output Waveform for Test Bench 3:	87
Code Coverage Report for Test Bench 3:	87
Toggle Coverage:	88
Block Coverage:	88
Explanation of coverage:	89
3. Logic Synthesis	90
Inputs:	90
RTL, Library file, Constraints file	90
Output:	90
Netlist (in .v format)	90
Library File:	90
Constraints:	90
Netlist:	90
Software Used:	90
Commands Used:	90
Synthesis for Minimum Area	91
TCL file:	91
Constraints provided for the minimum area	92
Area Report	92
Timing Report	92
Power Report	94
Cell Report	96
Synthesis for Best Timing	97
TCL file:	97
Constraints provided for just negative slack	97
Area Report	98
Timing Report	99
Power Report	101

Cell Report	101
Synthesis for Optimal Constraints	102
TCL file:	103
Constraints provided	103
Area Report	104
Timing Report	105
Power Report	107
Cell Report	108
Synthesized Schematic	109
QoR Comparison	109
Analysis of min area constraints	109
Analysis of tight clock constraints	110
Analysis of optimal clock constraints	110
RTL and Netlist comparison	111
4. Logical Equivalence Checking	117
Inputs: RTL, netlist generated after synthesis.	117
Output: Report which tells whether the design and mapped points are equivalent.	117
Software Used: Conformal (Cadence)	117
Commands Used:	117
Equivalence Checking for Minimum Area:	118
Equivalence Checking for Optimal Constraints:	120
Equivalence Checking for Best Timing:	121
Equivalence checking for Bad Netlist	124
1) Equivalence checking after manually editing the netlist to remove module master.	
124	
❖ As the module referred in is not defined the tool throws an error.	124
2) Equivalence checking after manually interchanging a port in a line in netlist has resulted in mapping error.	125
❖ The comparison revealed non-equivalence between two points, denoted as "NONEQ" for non-equivalence. The reason being unmapped points.	125
3) Equivalence checking after manually modifying a gate.	126
❖ We changed one NOR gate to OR gate, due to the above changes the compared 2 netlists have become inequivalent	126
5. STA	127
Inputs:	127
Output:	127
Software Used:	128
Cadence Tempus	128
Commands Used:	128
Remarks	128
STA for Minimum Area	128
TCL file:	128

Constraints used:	129
Violation report:	130
Timing report:	132
STA for Optimal Constraints	132
TCL file:	133
Constraints used:	133
Violation report:	134
Timing report:	136
STA for Best Timing	137
TCL file:	137
Constraints used:	138
Violation report:	139
Timing report:	141
Path Based Analysis	141
STA : ANALYSIS	142
6. Scan Insertion	143
Inputs: Liberty file, tcl file,constraints file (.sdc), Synthesized netlist	143
Output: Cell Report, Area Report, Power Report, Timing Report	143
Software Used: Cadence	143
Commands Used:	143
Schematic Design	143
Scan Insertion for Minimum Area	145
Constraints	146
Area Report	146
Timing Report	147
Power Report	148
Scan Insertion for Optimal Constraints	149
Constraints	149
Area Report	150
Timing Report	151
Power Report	153
Scan Insertion for Best timing	153
Constraints	154
Area Report	154
Timing Report	155
Power Report	156
Differences and Explanation of New Entities in netlist after Scan Chain Insertion	157
In our model we have inserted 2 scan chains. Scan changes introduce new ports to our design. Input ports introduced are scan_enable, test_mode, DFT_sdi_1 and DFT_sdi_2. Output ports introduced are DFT_sdo_1 and DFT_sdo_2. x_1 refers to the entities of the first scan chain while x_2 refers to second scan chain entities. When we did insertion of the scan chain all the DFF in the netlist were replaced by the special scan chain units	

from the library called SDFFRHQX1 cells.	157
Scan Chain Failure Detection Mechanism	159
QoR Analysis	159
Timing Report Analysis after Scan Chain Insertion	160

RTL to GDS Synthesis of a 2x2 Network on Chip

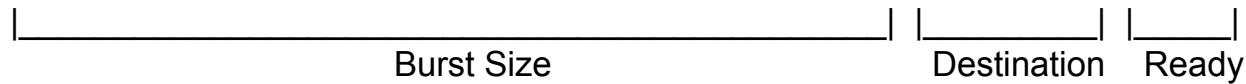
1. Detailed Specifications:

Inputs:

- Clock Signal (1 bit signal)
- Reset Signal (1 bit signal)
- Four [10:0] configure signals corresponding to each processor, that is,
 - p0_configure for processor 0
 - p1_configure for processor 1
 - p2_configure for processor 2
 - p3_configure for processor 3

Configure Signal [10:0]

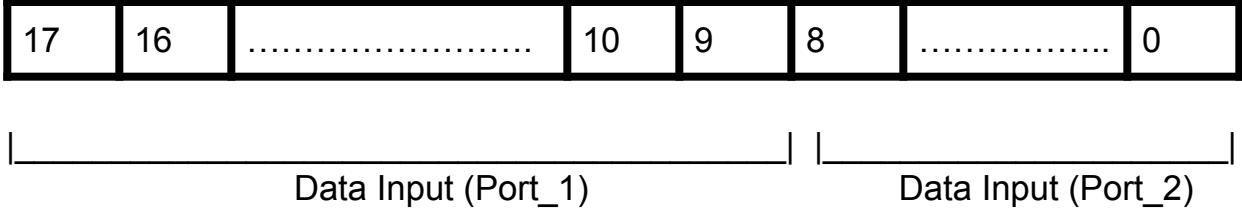
10	9	8	7	6	5	4	3	2	1	0
----	---	---	---	---	---	---	---	---	---	---



- ❖ 8 bits to indicate the number of transfers in a single transaction that is, the burst size
- ❖ 2 bits indicating the destination processor for the transfer from the current processor
- ❖ 1 bit indicates testbench is requesting to initiate a transfer from the current processor
- Block_all_paths signal (1 bit signal) is a signal used to block all the paths in the NOC. This is required for testing and coverage of code.
- Four [17:0] signals corresponding to each router for external input ports, that is,
 - R0_input for router 0
 - R1_input for router 1

- R2_input for router 2
- R3_input for router 3

Router input Signal [10:0]



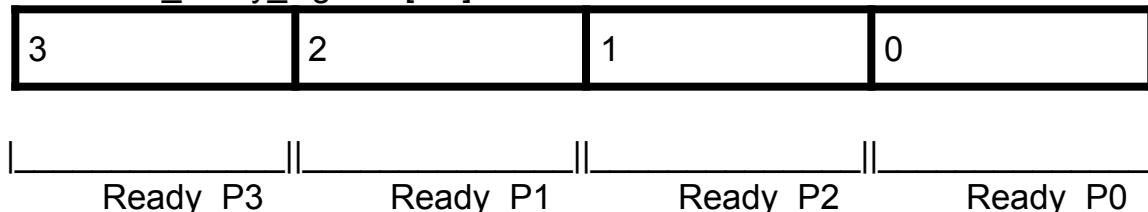
- ❖ Data Input size has 9 bits to indicate a last signal (1 bit) + data packet (8 bits)
- ❖ No. of external Ports => 2 per router
- ❖ External ports of the router =>

Router	Port_1 (Direction)	Port_2 (Direction)
R0	West	South
R1	East	South
R2	West	North
R3	East	North

Outputs:

- Processor_ready_signals =>A [3:0] output signal where each bit corresponds to Each of the Processor signals

Processor_ready_signals [3:0]



- Four [8:0] data signals corresponding to each processor, that is,
- P0_recieve_data for processor 0
- P1_recieve_data for processor 1

P2_recieve_data for processor 2

P3_recieve_data for processor 3

Where these 9 bits are used to indicate a last signal (1 bit) + data packet (1 byte) received by the processor.

- Four [17:0] signals corresponding to each router for external Output ports, that is,

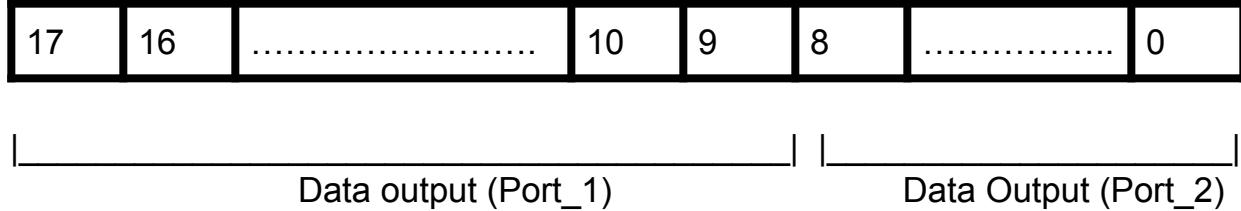
R0_output for router 0

R1_output for router 1

R2_output for router 2

R3_output for router 3

Router Output Signal [10:0]



❖ Data Output size =>

9 bits to indicate a last signal (1 bit) + data packet (1 byte)

❖ No. of external Ports => 2 per router

❖ External ports of the router =>

Router	Port_1 (Direction)	Port_2 (Direction)
R0	West	South
R1	East	South
R2	West	North
R3	East	North

Description:

NoC is a proper interconnection network that addresses the communication complexity among hundreds to thousands of cores in a many-core system on chip (SoC).

We have tried to design a simple 2x2 NoC design that helps in identifying the possible paths between any two routers to transfer a maximum of 256 packets.

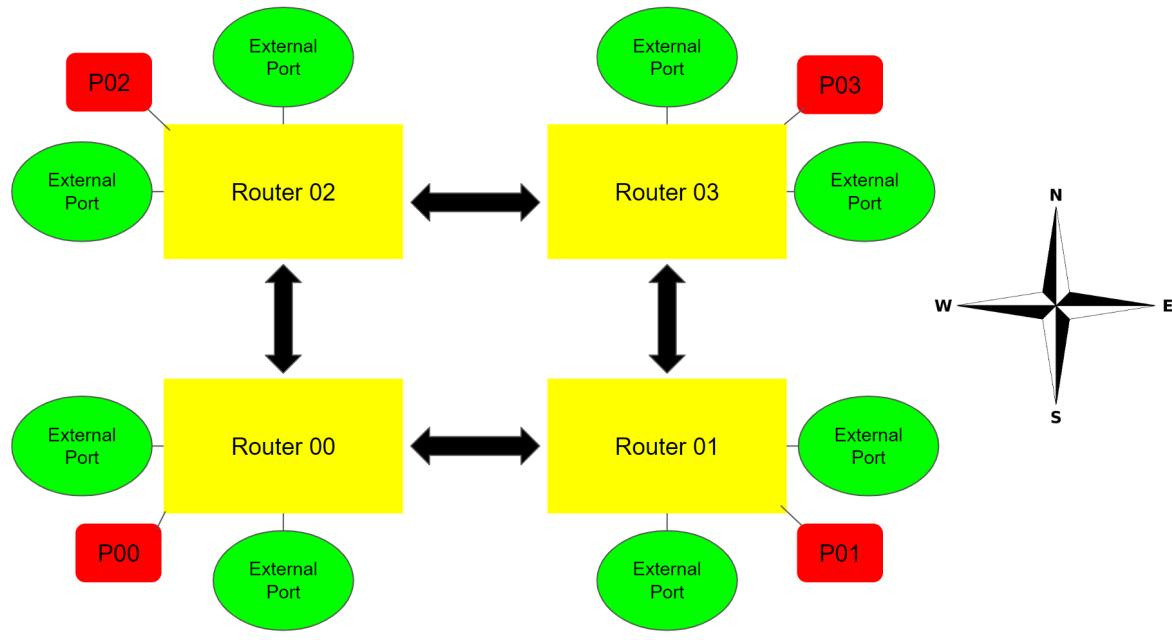
Router: This module assists in implementing each core and serves as the building block of our NoC mesh. It receives input of select lines from the master module and ensures the transmission of data to the next hop along the required path.

Master: This module helps in assigning the path for data transfer between any two processors

ProcessorUnit: This module demonstrates the working of the processor of each router. It has counters which ensure the data has reached the assigned destination up until the last filt.

Main: This is the top module for the entire design. It includes instances of Router, Master and ProcessorUnit.

Block Design:



Assumptions:

- We have taken an assumption that this NOC design is going to operate on a byte addressable system, i.e, the size of a single packet is going to be one byte.

- We have designed a scalable NOC structure which could be utilized in making up a bigger NOC design. This is to say that this design could be used even for a bigger NOC as a constituent black box, which would automatically route internal paths to process the transaction.
- To make the design more realistic we have utilized the concept of burst size factor.
- We have assumed that a processor in a single transfer can have a maximum burst size of 256 packets.
- NoC allows multiple transactions to occur parallelly. But we can't afford intermixing of packets from different processing units. To avoid this we utilize a mechanism of request and acknowledgement.
- We assume that a processor cannot request for another transfer until all the packets of previous transfer are not transmitted to its corresponding router.
- Now since all transfers of multiple processors are requested parallelly in the system we also needed a mechanism to ensure that requests don't overlap. We have no restrictions on processing units (P_unit (s)). They can request anything at any point in time.
- We assume that master processes all the requests at the rising edge of the clock and It processes requests of transfer in a priority fashion where the P_unit-0 gets the highest priority followed by P_unit-1 and so on.
- Most importantly, we assume that our design works in a synchronous fashion, that is the transfer of a packet from one unit to another happens only at posedge of the clock. (1 packet transfer from P0 to P1 takes 3 cycles after master approval)
- Also please note that since our design is synchronous and completely pipelined it takes a cycle to see the transfer. (+1 cycle to see the data at P1 {in above example})
- At Last we have an assumption that inputs to this design are constant at the rising edge of the clock. This is because we have flip flops at the input and the output ports.

2. Verilog Code, Test Bench, and Code Coverage

Command used:

```
ncverilog -access rwc testbench_name.v -coverage all -gui
```

Verilog Code:

Module mesh

```
/*
Module name:
    mesh
Module Description:
    This module helps in identifying the possible paths between any 2
routers:
                2 - 3
                  |
                  |
                0 - 1

Pin Description:
    Clock: 1 bit input port for the clock signal.
    Reset: 1 bit input port for the reset signal.
    configure_signals: (P0 to P3 in order)
        8 bit to indicate no of transfers in a single transaction (burst
size)
        2 bit indicating destination of transfer
        1 bit indicates testbench is requesting transfer
    processor_ready_signals: 4 bit output port to indicate the readiness
of the processors (in order P3 to P0)
*/
`include"Master_new.v"
`include"P_unit.v"
`include"router.v"

module mesh(
    input clock,
    input reset,
    input [10:0]p0_configure,
    input [10:0]p1_configure,
    input [10:0]p2_configure,
    input [10:0]p3_configure,
    input [17:0] r0_input,
    input [17:0] r1_input,
    input [17:0] r2_input,
```

```

input [17:0] r3_input,
input block_all_paths,
output [3:0] processor_ready_signals,
//output [19:0] temp_path_block_signals,
output [8:0] p0_recieve_data,
output [8:0] p1_recieve_data,
output [8:0] p2_recieve_data,
output [8:0] p3_recieve_data,
output [17:0] r0_output,
output [17:0] r1_output,
output [17:0] r2_output,
output [17:0] r3_output
);

reg [10:0] p0_configure1,p1_configure1,p2_configure1,p3_configure1;
reg block_all_paths1;
//reg [19:0] temp_path_block_signals2;
wire [3:0] processor_ready_signals1;
reg [3:0] processor_ready_signals2;
// wire [19:0] temp_path_block_signals1;
wire [17:0] r0_output1,r1_output1,r2_output1,r3_output1;
reg [17:0] r0_input1,r1_input1,r2_input1,r3_input1;
reg [17:0] r0_output2,r1_output2,r2_output2,r3_output2;
//assign temp_path_block_signals=temp_path_block_signals2;
assign processor_ready_signals=processor_ready_signals2;
assign r0_output=r0_output2;
assign r1_output=r1_output2;
assign r2_output=r2_output2;
assign r3_output=r3_output2;
always@(posedge clock)
begin
    r0_input1 <= r0_input;
    block_all_paths1 <= block_all_paths;
    r1_input1 <= r1_input;
    r2_input1 <= r2_input;
    r3_input1 <= r3_input;
end
always@(posedge clock)
begin
    r0_output2 <= r0_output1;
    r1_output2 <= r1_output1;

```

```

    r2_output2 <= r2_output1;
    r3_output2 <= r3_output1;
end
always@(posedge clock)
begin
    processor_ready_signals2 <= processor_ready_signals1;
    // temp_path_block_signals2 <= temp_path_block_signals1;
end
always@(posedge clock)
begin
    p0_configure1 <= p0_configure;
    p1_configure1 <= p1_configure;
    p2_configure1 <= p2_configure;
    p3_configure1 <= p3_configure;
end

wire [8:0] d01,d10,d23,d32,d02,d20,d13,d31;
wire [8:0] d00,d11,d22,d33;
wire [8:0] r00,r11,r22,r33;
wire Nr0,Sr0,Er0,Wr0;
wire Nr1,Sr1,Er1,Wr1;
wire Nr2,Sr2,Er2,Wr2;
wire Nr3,Sr3,Er3,Wr3;
wire Pr0,Pr1,Pr2,Pr3;

reg [6:0] Path_usage_bits_0;
reg [6:0] Path_usage_bits_1;
reg [6:0] Path_usage_bits_2;
reg [6:0] Path_usage_bits_3;

wire [27:0] Path_usage_bits;

wire [3:0] response_signals;
wire [3:1] P0_signals;
wire [3:1] P1_signals;
wire [3:1] P2_signals;
wire [3:1] P3_signals;
wire [19:0] R0_control_signals;
wire [19:0] R1_control_signals;
wire [19:0] R2_control_signals;

```

```

wire [19:0] R3_control_signals;
// wire [19:0] temp_path_block_signals;

// parameter NO_DATA=9'b0000000000;

//instantiation
master m0(
    .clock(clock),
    .reset(reset),
    .path_free_bits(Path_usage_bits),
    .P0_signals(P0_signals),
    .P1_signals(P1_signals),
    .P2_signals(P2_signals),
    .P3_signals(P3_signals),
    .block_all_paths(block_all_paths1),
    .R0_control_signals(R0_control_signals),
    .R1_control_signals(R1_control_signals),
    .R2_control_signals(R2_control_signals),
    .R3_control_signals(R3_control_signals),
    .response_signals(response_signals)
    // .temp_path_block_signals(temp_path_block_signals1)
);

Processing_unit p0(
    .clock(clock),
    .reset(reset),
    .master_response(response_signals[0]),
    .data_from_router(r00),
    .data_to_router(d00),
    .request_transfer(P0_signals[1]),
    .which_processor(P0_signals[3:2]),
    .processor_ready(processor_ready_signals1[0]),
    .data_got(p0_recieve_data),
    .tb_request(p0_configure1[0]),
    .tb_processor(p0_configure1[2:1]),
    .tb_len(p0_configure1[10:3])
);

//Set commands by master
router r0(
    .clock(clock),
    .reset(reset),

```

```

    .select_north(R0_control_signals[19:17]),
    .select_south(R0_control_signals[16:14]),
    .select_east(R0_control_signals[13:11]),
    .select_west(R0_control_signals[10:8]),
    .select_processor(R0_control_signals[7:5]),
    .data_north(d20),
    .data_south(r0_input1[8:0]),
    .data_east(d10),
    .data_west(r0_input1[17:9]),
    .data_processor(d00),
    .output_north(d02),
    .output_south(r0_output1[8:0]),
    .output_east(d01),
    .output_west(r0_output1[17:9]),
    .output_processor(r00),
    .north_ready(Nr0),
    .south_ready(Sr0),
    .east_ready(Er0),
    .west_ready(Wr0),
    .processor_ready(Pr0),
    .SetNR(R0_control_signals[4]),
    .SetSR(R0_control_signals[3]),
    .SetER(R0_control_signals[2]),
    .SetWR(R0_control_signals[1]),
    .SetPR(R0_control_signals[0])
);

Processing_unit p1(
    .clock(clock),
    .reset(reset),
    .master_response(response_signals[1]),
    .data_from_router(r11),
    .data_to_router(d11),
    .request_transfer(P1_signals[1]),
    .which_processor(P1_signals[3:2]),
    .processor_ready(processor_ready_signals1[1]),
    .data_got(p1_recieve_data),
    .tb_request(p1_configure1[0]),
    .tb_processor(p1_configure1[2:1]),
    .tb_len(p1_configure1[10:3])
);

```

```

) ;

router r1(
    .clock(clock),
    .reset(reset),
    .select_north(R1_control_signals[19:17]),
    .select_south(R1_control_signals[16:14]),
    .select_east(R1_control_signals[13:11]),
    .select_west(R1_control_signals[10:8]),
    .select_processor(R1_control_signals[7:5]),
    .data_north(d31),
    .data_south(r1_input1[8:0]),
    .data_east(r1_input1[17:9]),
    .data_west(d01),
    .data_processor(d11),
    .output_north(d13),
    .output_south(r1_output1[8:0]),
    .output_east(r1_output1[17:9]),
    .output_west(d10),
    .output_processor(r11),
    .north_ready(Nr1),
    .south_ready(Sr1),
    .east_ready(Er1),
    .west_ready(Wr1),
    .processor_ready(Pr1),
    .SetNR(R1_control_signals[4]),
    .SetSR(R1_control_signals[3]),
    .SetER(R1_control_signals[2]),
    .SetWR(R1_control_signals[1]),
    .SetPR(R1_control_signals[0])
);

Processing_unit p2(
    .clock(clock),
    .reset(reset),
    .master_response(response_signals[2]),
    .data_from_router(r22),
    .data_to_router(d22),
    .request_transfer(P2_signals[1]),
    .which_processor(P2_signals[3:2]),
    .processor_ready(processor_ready_signals1[2]),
    .data_got(p2_recieve_data),
);

```

```

        .tb_request(p2_configure1[0]),
        .tb_processor(p2_configure1[2:1]),
        .tb_len(p2_configure1[10:3])
    );
router r2(
    .clock(clock),
    .reset(reset),
    .select_north(R2_control_signals[19:17]),
    .select_south(R2_control_signals[16:14]),
    .select_east(R2_control_signals[13:11]),
    .select_west(R2_control_signals[10:8]),
    .select_processor(R2_control_signals[7:5]),
    .data_north(r2_input1[8:0]),
    .data_south(d02),
    .data_east(d32),
    .data_west(r2_input1[17:9]),
    .data_processor(d22),
    .output_north(r2_output1[8:0]),
    .output_south(d20),
    .output_east(d23),
    .output_west(r2_output1[17:9]),
    .output_processor(r22),
    .north_ready(Nr2),
    .south_ready(Sr2),
    .east_ready(Er2),
    .west_ready(Wr2),
    .processor_ready(Pr2),
    .SetNR(R2_control_signals[4]),
    .SetSR(R2_control_signals[3]),
    .SetER(R2_control_signals[2]),
    .SetWR(R2_control_signals[1]),
    .SetPR(R2_control_signals[0])
);
Processing_unit p3(
    .clock(clock),
    .reset(reset),
    .master_response(response_signals[3]),
    .data_from_router(r33),
    .data_to_router(d33),
    .request_transfer(P3_signals[1]),

```

```

        .which_processor(P3_signals[3:2]),
        .processor_ready(processor_ready_signals1[3]),
        .data_got(p3_recieve_data),
        .tb_request(p3_configure1[0]),
        .tb_processor(p3_configure1[2:1]),
        .tb_len(p3_configure1[10:3])
    );
router r3(
    .clock(clock),
    .reset(reset),
    .select_north(R3_control_signals[19:17]),
    .select_south(R3_control_signals[16:14]),
    .select_east(R3_control_signals[13:11]),
    .select_west(R3_control_signals[10:8]),
    .select_processor(R3_control_signals[7:5]),
    .data_north(r3_input1[8:0]),
    .data_south(d13),
    .data_east(r3_input1[17:9]),
    .data_west(d23),
    .data_processor(d33),
    .output_north(r3_output1[8:0]),
    .output_south(d31),
    .output_east(r3_output1[17:9]),
    .output_west(d32),
    .output_processor(r33),
    .north_ready(Nr3),
    .south_ready(Sr3),
    .east_ready(Er3),
    .west_ready(Wr3),
    .processor_ready(Pr3),
    .SetNR(R3_control_signals[4]),
    .SetSR(R3_control_signals[3]),
    .SetER(R3_control_signals[2]),
    .SetWR(R3_control_signals[1]),
    .SetPR(R3_control_signals[0])
);
always @ (*) //router 0
begin
    Path_usage_bits_0[0] = Pr0 ; //0 to 0

```

```

    Path_usage_bits_0[1] = Er0 & Pr1; //0 to 1 //flat
    Path_usage_bits_0[2] = Nr0 & Er2 & Sr3 & Pr1 ; //0 to 1 longer
0-2-3-1

    Path_usage_bits_0[3] = Nr0 & Pr2; //0 to 2 //vertical
    Path_usage_bits_0[4] = Er0 & Nr1 & Wr3 & Pr2; //0 to 2 longer
0-1-3-2

    Path_usage_bits_0[5] = Nr0  & Er2  & Pr3 ; //0 to 3 //diagonal
(vertical) 0-2-3
    Path_usage_bits_0[6] = Er0 & Nr1  & Pr3 ; //0 to 3 (flat) 0-1-3

end

always @ (*) //router1
begin
    Path_usage_bits_1[0] = Pr1 ; //1 to 1

    Path_usage_bits_1[1] = Er1 & Pr0; //1 to 0 //flat
    Path_usage_bits_1[2] = Nr1  & Wr3 & Sr2 & Pr0 ; //1 to 0 longer
1-3-2-0

    Path_usage_bits_1[3] = Nr1  & Pr3 ; //1 to 3 //vertical
    Path_usage_bits_1[4] = Wr1  & Nr0 & Er2 & Pr3 ; // 1 to 3 longer
1-0-2-3

    Path_usage_bits_1[5] = Nr1 & Wr3 & Pr2 ; // 1 to 2 //diagonal
(vertical) 1-3-2
    Path_usage_bits_1[6] = Wr1 & Nr0 & Pr2 ; //1 to 2 (flat) 1-0-2

end

always @ (*) //router2
begin
    Path_usage_bits_2[0] = Pr2; //2 to 2

    Path_usage_bits_2[1] = Er2 & Pr3 ; //2 to 3 //flat

```

```

    Path_usage_bits_2[2] = Sr2 & Er0 & Nr1 & Pr3 ; //2 to 3 longer
2-0-1-3

    Path_usage_bits_2[3] = Sr2 & Pr0; //2 to 0 //vertical
    Path_usage_bits_2[4] = Er2 & Sr3 & Wr1 & Pr0 ; //2 to 0 longer
2-3-1-0

    Path_usage_bits_2[5] = Sr2 & Er0 & Pr1 ; //2 to 1 //diagonal
(vertical) 2-0-1
    Path_usage_bits_2[6] = Er2 & Sr3 & Pr1 ; //2 to 1 (flat) 2-3-1

end

always @ (*) //router3
begin
    Path_usage_bits_3[0] = Pr3 ; // 3 to 3

    Path_usage_bits_3[1] = Wr3 & Pr2 ; //3 to 2 //flat
    Path_usage_bits_3[2] = Sr3 & Wr1 & Nr0 & Pr2 ; //3 to 2 longer
3-1-0-2

    Path_usage_bits_3[3] = Sr3 & Pr1 ; //3 to 1 //vertical
    Path_usage_bits_3[4] = Wr3 & Sr2 & Er0 & Pr1 ; //3 to 1 longer
3-2-0-1

    Path_usage_bits_3[5] = Sr3 & Wr1 & Pr0 ; //3 to 0 //diagonal
(vertical) 3-1-0
    Path_usage_bits_3[6] = Wr3 & Sr2 & Pr0; //3 to 0 (flat) 3-2-0

end

assign Path_usage_bits = { Path_usage_bits_3, Path_usage_bits_2,
Path_usage_bits_1, Path_usage_bits_0};
endmodule

```

Module master

```
/*
Module Pin Description :

Clock: 1 bit input port for the clock signal.

Path_free_bits: Indicates the usage of all 28 ((24 cross paths + 4
self paths )) paths available in the 2x2 NOC mesh.

Router_control_signals: (for all R0 to R3)
    3 bits each for selecting lines of a router -> (In order North,
South, East, West, Processor) i.e 3x5
    1 bit each for setting the ready regs of a router (In order
North, South, East, West, Processor) i.e 1x5

Processor_signals: (for all P0 to P3)
    2 bit to indicate to which processor is transaction requested
    1 bit each for requesting transfer

response_signals: 4 bit signal where each bit indicates the response
from the master to the processors (in order P3 to P0)

temp_path_block_signals:
    1 bit each for setting the ready regs of a router (In order
North, South, East, West, Processor) i.e 1x5
    and where 1 indicates the path is blocked
    (in order R0 to R3)

*/
```

```
module master(
    input clock,
    input reset,
    input [27:0] path_free_bits,
    input [2:0] P0_signals,
    input [2:0] P1_signals,
    input [2:0] P2_signals,
    input [2:0] P3_signals,
    input block_all_paths,
    output [19:0] R0_control_signals,
    output [19:0] R1_control_signals,
    output [19:0] R2_control_signals,
    output [19:0] R3_control_signals,
    output [3:0] response_signals
    // output [19:0] temp_path_block_signals
);
```

```

// //temporary variables
reg [19:0] R0_control_signals1, R1_control_signals1,
R2_control_signals1, R3_control_signals1;
reg [19:0] R0_control_signals2, R1_control_signals2,
R2_control_signals2, R3_control_signals2;
reg [3:0] response_signals1;
reg [3:0] response_signals2;
assign R0_control_signals = R0_control_signals2;
assign R1_control_signals = R1_control_signals2;
assign R2_control_signals = R2_control_signals2;
assign R3_control_signals = R3_control_signals2;

assign response_signals = response_signals2;
//Preliminary conditions for reset and and every posedge
always@(posedge clock or posedge reset)
begin
    if(reset==1'b1) //At reset data does not need to go to anywhere,
so all destinadirections of router are set to default
        begin
            R0_control_signals2 <= 20'b0;
            R1_control_signals2 <= 20'b0;
            R2_control_signals2 <= 20'b0;
            R3_control_signals2 <= 20'b0;
            response_signals2 <= 4'b0; //Master is saying free all paths
        end
    else //assign destination for every direction of a router
following the below computation at every clockedge
        begin
            R0_control_signals2 <= R0_control_signals1 ;
            R1_control_signals2 <= R1_control_signals1 ;
            R2_control_signals2 <= R2_control_signals1 ;
            R3_control_signals2 <= R3_control_signals1 ;
            response_signals2 <= response_signals1;
        end
    end

// P0 - Processing
reg [19:0] R0_control_signals1_0, R1_control_signals1_0,
R2_control_signals1_0, R3_control_signals1_0;
reg response_signals1_0;

```

```

always@(*)
begin
    if(P0_signals[0]==1'b1)
        begin
            case(P0_signals[2:1]) //iterating all cases of P0
                2'b00://0 to 0
                    begin
                        if(path_free_bits[0]==1'b1)
                            begin

R0_control_signals1_0[19:0]={R0_control_signals2[19:8],3'b100,4'b0,1'b1};

R1_control_signals1_0[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_0[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_0[19:0]={R3_control_signals2[19:5],5'b0};
                    response_signals1_0 = 1'b1;
                end
                else
                    begin

R0_control_signals1_0[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_0[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_0[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_0[19:0]={R3_control_signals2[19:5],5'b0};
                    response_signals1_0 = 1'b0;
                end
            end
            2'b01://0 to 1
            begin
                if(path_free_bits[1]==1'b1)
                    begin //direct

R1_control_signals1_0[19:0]={R1_control_signals2[19:8],3'b011,4'b0,1'b1};

```

```

R0_control_signals1_0[19:0]={R0_control_signals2[19:14],3'b100,R0_control_
signals2[10:5],2'b0,1'b1,2'b0};

R2_control_signals1_0[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_0[19:0]={R3_control_signals2[19:5],5'b0};
    response_signals1_0 = 1'b1;
end
else if(path_free_bits[2]==1'b1)      begin
//indirect 0-2-3-1

R1_control_signals1_0[19:0]={R1_control_signals2[19:8],3'b000,4'b0,1'b1};

R0_control_signals1_0[19:0]={3'b100,R0_control_signals2[16:5],1'b1,4'b0};

R2_control_signals1_0[19:0]={R2_control_signals2[19:14],3'b001,R2_control_
signals2[10:5],2'b0,1'b1,2'b0};

R3_control_signals1_0[19:0]={R3_control_signals2[19:17],3'b011,R3_control_
signals2[13:5],1'b0,1'b1,3'b0};
    response_signals1_0 = 1'b1;
end
else
begin

R0_control_signals1_0[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_0[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_0[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_0[19:0]={R3_control_signals2[19:5],5'b0};
    response_signals1_0 = 1'b0;
end
end
2'b10://from 0 to 2
begin
if(path_free_bits[3]==1'b1)
begin//direct

```

```

                    response_signals1_0 =
1'b1; //Processor of Router 1 active

R1_control_signals1_0[19:0]={R1_control_signals2[19:5],5'b0};

R0_control_signals1_0[19:0]={3'b100,R0_control_signals2[16:5],1'b1,4'b0};

R2_control_signals1_0[19:0]={R2_control_signals2[19:8],3'b001,4'b0,1'b1};

R3_control_signals1_0[19:0]={R3_control_signals2[19:5],5'b0};
    end
    else if (path_free_bits[4]==1'b1)
begin//indirect 0-1-3-2
                    response_signals1_0 =
1'b1;

R1_control_signals1_0[19:0]={3'b011,R1_control_signals2[16:5],1'b1,4'b0};

R0_control_signals1_0[19:0]={R0_control_signals2[19:14],3'b100,R0_control_
signals2[10:5],2'b0,1'b1,2'b0};

R2_control_signals1_0[19:0]={R2_control_signals2[19:8],3'b010,4'b0,1'b1};

R3_control_signals1_0[19:0]={R3_control_signals2[19:11],3'b001,R3_control_
signals2[7:5],3'b0,1'b1,1'b0};
    end
    else
begin

R0_control_signals1_0[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_0[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_0[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_0[19:0]={R3_control_signals2[19:5],5'b0};
                    response_signals1_0 = 1'b0;
    end
end
2'b11://0 to 3

```

```

begin
    if(path_free_bits[5]==1'b1)
        begin//0-2-3
            response_signals1_0 =
1'b1;

R1_control_signals1_0[19:0]={R1_control_signals2[19:5],5'b0};

R0_control_signals1_0[19:0]={3'b100,R0_control_signals2[16:5],1'b1,4'b0};

R2_control_signals1_0[19:0]={R2_control_signals2[19:14],3'b001,R2_control_
signals2[10:5],2'b0,1'b1,2'b0};

R3_control_signals1_0[19:0]={R3_control_signals2[19:8],3'b011,4'b0,1'b1};
    end
    else if(path_free_bits[6]==1'b1)
        begin//0-1-3
            response_signals1_0 =
1'b1;

R0_control_signals1_0[19:0]={R0_control_signals2[19:14],3'b100,R0_control_
signals2[10:5],2'b0,1'b1,2'b0};

R1_control_signals1_0[19:0]={3'b011,R1_control_signals2[16:5],1'b1,4'b0};

R2_control_signals1_0[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_0[19:0]={R3_control_signals2[19:8],3'b001,4'b0,1'b1};
    end
    else
        begin

R0_control_signals1_0[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_0[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_0[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_0[19:0]={R3_control_signals2[19:5],5'b0};
            response_signals1_0 = 1'b0;

```

```

        end
    end
endcase
end
else if(block_all_paths==1'b1)
begin
    R0_control_signals1_0[19:0]={20{1'b1}};
    R1_control_signals1_0[19:0]={20{1'b1}};
    R2_control_signals1_0[19:0]={20{1'b1}};
    R3_control_signals1_0[19:0]={20{1'b1}};
    response_signals1_0 = 1'b0;
end
else
begin
    R0_control_signals1_0[19:0]={R0_control_signals2[19:5],5'b0};
    R1_control_signals1_0[19:0]={R1_control_signals2[19:5],5'b0};
    R2_control_signals1_0[19:0]={R2_control_signals2[19:5],5'b0};
    R3_control_signals1_0[19:0]={R3_control_signals2[19:5],5'b0};
    response_signals1_0 = 1'b0;
end
end
// P1 - Processing
reg [19:0] R0_control_signals1_1, R1_control_signals1_1,
R2_control_signals1_1, R3_control_signals1_1;
reg response_signals1_1;
always@(*)
begin
    if(P1_signals[0]==1'b1) //iterating all cases of P1
    begin
        case(P1_signals[2:1])
            2'b00: //1 to 0
            begin
                if(path_free_bits[8]==1'b1)
                begin //direct
                    response_signals1_1 = 1'b1;
R0_control_signals1_1[19:0]={R0_control_signals2[19:8],3'b010,4'b0,1'b1};

R1_control_signals1_1[19:0]={R1_control_signals2[19:11],3'b100,R1_control_
signals2[7:5],3'b0,1'b1,1'b0};

```

```

R2_control_signals1_1[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_1[19:0]={R3_control_signals2[19:5],5'b0};
    end
    else if(path_free_bits[9]==1'b1)
begin//indirect
    response_signals1_1 = 1'b1;

R0_control_signals1_1[19:0]={R0_control_signals2[19:8],3'b000,4'b0,1'b1};

R1_control_signals1_1[19:0]={3'b100,R1_control_signals2[16:5],1'b1,4'b0};

R2_control_signals1_1[19:0]={R2_control_signals2[19:17],3'b010,R2_control_
signals2[13:5],1'b0,1'b1,3'b0};

R3_control_signals1_1[19:0]={R3_control_signals2[19:11],3'b001,R3_control_
signals2[7:5],3'b0,1'b1,1'b0};
    end
    else
begin

R0_control_signals1_1[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_1[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_1[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_1[19:0]={R3_control_signals2[19:5],5'b0};
    response_signals1_1 = 1'b0;
    end
end
2'b01://1 to 1
begin
    if(path_free_bits[7]==1'b1)
begin
    response_signals1_1 =
1'b1;

R0_control_signals1_1[19:0]={R0_control_signals2[19:5],5'b0};

```

```

R1_control_signals1_1[19:0]={R1_control_signals2[19:8],3'b100,4'b0,1'b1};

R2_control_signals1_1[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_1[19:0]={R3_control_signals2[19:5],5'b0};
    end
    else
begin

R0_control_signals1_1[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_1[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_1[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_1[19:0]={R3_control_signals2[19:5],5'b0};
    response_signals1_1 = 1'b0;
end
end
2'b10: //from 1 to 2
begin
    if(path_free_bits[12]==1'b1)
begin//1-3-2
    response_signals1_1 = 1'b1;

R0_control_signals1_1[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_1[19:0]={3'b100,R1_control_signals2[16:5],1'b1,4'b0};

R3_control_signals1_1[19:0]={R3_control_signals2[19:11],3'b001,R3_control_
signals2[7:5],3'b0,1'b1,1'b0};

R2_control_signals1_1[19:0]={R2_control_signals2[19:8],3'b010, 4'b0,
1'b1};
end
else if(path_free_bits[13]==1'b1)
begin//1-0-2
response_signals1_1 = 1'b1;

```

```

R0_control_signals1_1[19:0]={3'b010,R0_control_signals2[16:5],1'b1,4'b0};

R1_control_signals1_1[19:0]={R1_control_signals2[19:11],3'b100,R1_control_
signals2[7:5],3'b0,1'b1,1'b0};

R2_control_signals1_1[19:0]={R2_control_signals2[19:8],3'b001, 4'b0,
1'b1};

R3_control_signals1_1[19:0]={R3_control_signals2[19:5],5'b0};
      end
      else
      begin

R0_control_signals1_1[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_1[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_1[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_1[19:0]={R3_control_signals2[19:5],5'b0};
      response_signals1_1 = 1'b0;
      end
      end
      2'b11://1 to3
      begin
          if(path_free_bits[10]==1'b1)
          begin//direct
          response_signals1_1 = 1'b1;

R0_control_signals1_1[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_1[19:0]={3'b100,R1_control_signals2[16:5],1'b1,4'b0};

R2_control_signals1_1[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_1[19:0]={R3_control_signals2[19:8],3'b001, 4'b0,
1'b1};

```

```

        end
    else if(path_free_bits[11]==1'b1)
begin//indirect
    response_signals1_1 = 1'b1;

R0_control_signals1_1[19:0]={3'b010,R0_control_signals2[16:5],1'b1,4'b0};

R1_control_signals1_1[19:0]={R1_control_signals2[19:11],3'b100,R1_control_
signals2[7:5],3'b0,1'b1,1'b0};

R2_control_signals1_1[19:0]={R2_control_signals2[19:14],3'b001,R2_control_
signals2[10:5],2'b0,1'b1,2'b0};

R3_control_signals1_1[19:0]={R3_control_signals2[19:8],3'b011, 4'b0,
1'b1};

        end
    else
begin

R0_control_signals1_1[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_1[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_1[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_1[19:0]={R3_control_signals2[19:5],5'b0};
    response_signals1_1 = 1'b0;
end
end
endcase
end
else if(block_all_paths==1'b1)
begin
    R0_control_signals1_1[19:0]={20{1'b1}};
    R1_control_signals1_1[19:0]={20{1'b1}};
    R2_control_signals1_1[19:0]={20{1'b1}};
    R3_control_signals1_1[19:0]={20{1'b1}};
    response_signals1_1 = 1'b0;
end

```

```

    else
    begin
        R0_control_signals1_1[19:0]={R0_control_signals2[19:5],5'b0};
        R1_control_signals1_1[19:0]={R1_control_signals2[19:5],5'b0};
        R2_control_signals1_1[19:0]={R2_control_signals2[19:5],5'b0};
        R3_control_signals1_1[19:0]={R3_control_signals2[19:5],5'b0};
        response_signals1_1 = 1'b0;
    end
end

// P2 - Processing
reg [19:0] R0_control_signals1_2, R1_control_signals1_2,
R2_control_signals1_2, R3_control_signals1_2;
reg response_signals1_2;
always@(*)
begin
    if(P2_signals[0]==1'b1) //iterating all cases of P2
    begin
        case(P2_signals[2:1])
            2'b00: //2 to 0
            begin
                if(path_free_bits[17]==1'b1)
                begin//direct

                    response_signals1_2 = 1'b1;

R0_control_signals1_2[19:0]={R0_control_signals2[19:8],3'b000,4'b0,1'b1};

R1_control_signals1_2[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_2[19:0]={R2_control_signals2[19:17],3'b100,R2_control_
signals2[13:5],1'b0,1'b1,3'b0};

R3_control_signals1_2[19:0]={R3_control_signals2[19:5],5'b0};
                end
                else if (path_free_bits[18]==1'b1)
                begin //indirect

                    response_signals1_2 = 1'b1;

```

```

R0_control_signals1_2[19:0]={R0_control_signals2[19:8],3'b010,4'b0,1'b1};

R1_control_signals1_2[19:0]={R1_control_signals2[19:11],
3'b000,R1_control_signals2[7:5],3'b0,1'b1,1'b0};

R2_control_signals1_2[19:0]={R2_control_signals2[19:14],3'b100,
R2_control_signals2[10:5],2'b0,1'b1,2'b0};

R3_control_signals1_2[19:0]={R3_control_signals2[19:17],3'b011,R3_control_
signals2[13:5],1'b0,1'b1,3'b0};
begin
end
else
begin

R0_control_signals1_2[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_2[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_2[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_2[19:0]={R3_control_signals2[19:5],5'b0};
begin
    response_signals1_2 = 1'b0;
end
end
2'b01://2 to 1
begin// modified swapping of 19 and 20
if(path_free_bits[20]==1'b1)
begin//2-3-1
    response_signals1_2 = 1'b1;

R0_control_signals1_2[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_2[19:0]={R1_control_signals2[19:8],3'b000,4'b0,1'b1};

R2_control_signals1_2[19:0]={R2_control_signals2[19:14],3'b100,R2_control_
signals2[10:5],2'b0,1'b1,2'b0};

```

```

R3_control_signals1_2[19:0]={R3_control_signals2[19:17],3'b011,R3_control_
signals2[13:5],1'b0,1'b1,3'b0};
      end
      else if(path_free_bits[19]==1'b1)
begin//2-0-1
      response_signals1_2 = 1'b1;

R0_control_signals1_2[19:0]={R0_control_signals2[19:14],3'b000,R0_control_
signals2[10:5],2'b0,1'b1,2'b0};

R1_control_signals1_2[19:0]={R1_control_signals2[19:8],3'b011,4'b0,1'b1};

R2_control_signals1_2[19:0]={R2_control_signals2[19:17],3'b100,R2_control_
signals2[13:5], 1'b0, 1'b1,3'b0};

R3_control_signals1_2[19:0]={R3_control_signals2[19:5],5'b0};
      end
      else
begin

R0_control_signals1_2[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_2[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_2[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_2[19:0]={R3_control_signals2[19:5],5'b0};
      response_signals1_2 = 1'b0;
      end
      end
      2'b10://2 to 2
begin
      if(path_free_bits[14]==1'b1)
begin
      response_signals1_2 = 1'b1;

R0_control_signals1_2[19:0]={R0_control_signals2[19:5],5'b0};

```

```

R1_control_signals1_2[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_2[19:0]={R2_control_signals2[19:8],3'b100,4'b0,1'b1};

R3_control_signals1_2[19:0]={R3_control_signals2[19:5],5'b0};
    end
    else
begin

R0_control_signals1_2[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_2[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_2[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_2[19:0]={R3_control_signals2[19:5],5'b0};
    response_signals1_2 = 1'b0;
end
end
2'b11: //from 2 to 3
begin
    if(path_free_bits[15]==1'b1)
begin//direct
    response_signals1_2 = 1'b1;

R0_control_signals1_2[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_2[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_2[19:0]={R2_control_signals2[19:14],3'b100,R2_control_
signals2[10:5],2'b0,1'b1,2'b0};

R3_control_signals1_2[19:0]={R3_control_signals2[19:8],3'b011,4'b0,1'b1};
    end
    else if(path_free_bits[16]==1'b1)
begin//indirect
    response_signals1_2 = 1'b1;

R1_control_signals1_2[19:0]={3'b011,R1_control_signals2[16:5],1'b1,4'b0};

```

```

R0_control_signals1_2[19:0]={R0_control_signals2[19:14],3'b000,R0_control_
signals2[10:5],2'b0,1'b1,2'b0};

R2_control_signals1_2[19:0]={R2_control_signals2[19:17],3'b100,R2_control_
signals2[13:5],1'b0,1'b1,3'b0};

R3_control_signals1_2[19:0]={R3_control_signals2[19:8],3'b001,4'b0,1'b1};
begin
end
else
begin

R0_control_signals1_2[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_2[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_2[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_2[19:0]={R3_control_signals2[19:5],5'b0};
response_signals1_2 = 1'b0;
end
end
endcase
end
else if(block_all_paths==1'b1)
begin
R0_control_signals1_2[19:0]={20{1'b1}};
R1_control_signals1_2[19:0]={20{1'b1}};
R2_control_signals1_2[19:0]={20{1'b1}};
R3_control_signals1_2[19:0]={20{1'b1}};
response_signals1_2 = 1'b0;
end
else
begin
R0_control_signals1_2[19:0]={R0_control_signals2[19:5],5'b0};
R1_control_signals1_2[19:0]={R1_control_signals2[19:5],5'b0};
R2_control_signals1_2[19:0]={R2_control_signals2[19:5],5'b0};
R3_control_signals1_2[19:0]={R3_control_signals2[19:5],5'b0};
response_signals1_2 = 1'b0;
end

```

```

end

// P3 - Processing
reg [19:0] R0_control_signals1_3, R1_control_signals1_3,
R2_control_signals1_3, R3_control_signals1_3;
reg response_signals1_3;
always@(*)
begin
    if(P3_signals[0]==1'b1) //iterating all cases of P3
    begin
        case(P3_signals[2:1])
            2'b00://3 to 0
            begin
                if(path_free_bits[26]==1'b1)
                begin //3-1-0
                    response_signals1_3 = 1'b1;

R3_control_signals1_3[19:0]={R3_control_signals2[19:17],3'b100,R3_control_
signals2[13:5],1'b0,1'b1,3'b0};

R1_control_signals1_3[19:0]={R1_control_signals2[19:11],3'b000,R1_control_
signals2[7:5],3'b0,1'b1,1'b0};

R2_control_signals1_3[19:0]={R2_control_signals2[19:5],5'b0};

R0_control_signals1_3[19:0]={R0_control_signals2[19:8],3'b010,4'b0,1'b1};
        end
        else if(path_free_bits[27]==1'b1)
        begin//3-2-0
            response_signals1_3 = 1'b1;

R0_control_signals1_3[19:0]={R0_control_signals2[19:8],3'b000,4'b0,1'b1};

R1_control_signals1_3[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_3[19:0]={R2_control_signals2[19:17],3'b010,R2_control_
signals2[13:5],1'b0,1'b1,3'b0};

```

```

R3_control_signals1_3[19:0]={R3_control_signals2[19:11],3'b100,R3_control_
signals2[7:5],3'b0,1'b1,1'b0};
begin
end
else
begin

R0_control_signals1_3[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_3[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_3[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_3[19:0]={R3_control_signals2[19:5],5'b0};
response_signals1_3 = 1'b0;
end
end
2'b01://3 to 1
begin
if(path_free_bits[24]==1'b1)
begin//direct

response_signals1_3 = 1'b1;

R0_control_signals1_3[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_3[19:0]={R1_control_signals2[19:8],3'b000,4'b0,1'b1};

R2_control_signals1_3[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_3[19:0]={R3_control_signals2[19:17],3'b100,R3_control_
signals2[13:5],1'b0,1'b1,3'b0};
end
else if(path_free_bits[25]==1'b1)
begin//indirect
response_signals1_3 = 1'b1;

R0_control_signals1_3[19:0]={R0_control_signals2[19:14],3'b000,R0_control_
signals2[10:5],2'b0,1'b1,2'b0};

```

```

R1_control_signals1_3[19:0]={R1_control_signals2[19:8],3'b011, 4'b0,
1'b1};

R2_control_signals1_3[19:0]={R2_control_signals2[19:17],3'b010,R2_control_
signals2[13:5],1'b0,1'b1,3'b0};

R3_control_signals1_3[19:0]={R3_control_signals2[19:11],3'b100,R3_control_
signals2[7:5], 3'b0, 1'b1,1'b0};
begin
end
else
begin

R0_control_signals1_3[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_3[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_3[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_3[19:0]={R3_control_signals2[19:5],5'b0};
response_signals1_3 = 1'b0;
begin
end
end
2'b10://3 to 2
begin
if(path_free_bits[22]==1'b1)
begin//direct

response_signals1_3 = 1'b1;

R0_control_signals1_3[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_3[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_3[19:0]={R2_control_signals2[19:8],3'b010,4'b0,1'b1};

R3_control_signals1_3[19:0]={R3_control_signals2[19:11],3'b100,R3_control_
signals2[7:5],3'b0,1'b1,1'b0};
begin
else if(path_free_bits[23]==1'b1)

```

```

begin//indirect
    response_signals1_3 = 1'b1;

R0_control_signals1_3[19:0]={3'b010,R0_control_signals2[16:5],1'b1,4'b0};

R1_control_signals1_3[19:0]={R1_control_signals2[19:11],3'b000,R1_control_
signals2[7:5],3'b0,1'b1,1'b0};

R2_control_signals1_3[19:0]={R2_control_signals2[19:8],3'b001, 4'b0,
1'b1};

R3_control_signals1_3[19:0]={R3_control_signals2[19:17],3'b100,R3_control_
signals2[13:5],1'b0,1'b1,3'b0};
    end
    else
begin

R0_control_signals1_3[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_3[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_3[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_3[19:0]={R3_control_signals2[19:5],5'b0};
    response_signals1_3 = 1'b0;
    end
end
2'b11: //3 to 3
begin
    if(path_free_bits[21]==1'b1)
begin

    response_signals1_3 = 1'b1;

R0_control_signals1_3[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_3[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_3[19:0]={R2_control_signals2[19:5],5'b0};

```

```

R3_control_signals1_3[19:0]={R3_control_signals2[19:8],3'b100,4'b0,1'b1};

      end
    else
      begin

R0_control_signals1_3[19:0]={R0_control_signals2[19:5],5'b0};

R1_control_signals1_3[19:0]={R1_control_signals2[19:5],5'b0};

R2_control_signals1_3[19:0]={R2_control_signals2[19:5],5'b0};

R3_control_signals1_3[19:0]={R3_control_signals2[19:5],5'b0};
      response_signals1_3 = 1'b0;
    end
  end
  endcase
end
else if(block_all_paths==1'b1)
begin
  R0_control_signals1_3[19:0]={20{1'b1}};
  R1_control_signals1_3[19:0]={20{1'b1}};
  R2_control_signals1_3[19:0]={20{1'b1}};
  R3_control_signals1_3[19:0]={20{1'b1}};
  response_signals1_3 = 1'b0;
end
else
begin
  R0_control_signals1_3[19:0]={R0_control_signals2[19:5],5'b0};
  R1_control_signals1_3[19:0]={R1_control_signals2[19:5],5'b0};
  R2_control_signals1_3[19:0]={R2_control_signals2[19:5],5'b0};
  R3_control_signals1_3[19:0]={R3_control_signals2[19:5],5'b0};
  response_signals1_3 = 1'b0;
end
end

// Multi processing - prep block
wire [4:0] sig0_1;

```

```

    assign sig0_1 = (R0_control_signals1_0[4:0] &
R0_control_signals1_1[4:0]) | (R3_control_signals1_0[4:0] &
R3_control_signals1_1[4:0]) | (R1_control_signals1_0[4:0] &
R1_control_signals1_1[4:0]) | (R2_control_signals1_0[4:0] &
R2_control_signals1_1[4:0]);
    reg [19:0] reg0_1,reg1_1,reg2_1,reg3_1;
    always@(*)
    begin
        response_signals1[0]=response_signals1_0;
    end
    always@(*)
    begin
        if(sig0_1==5'b0)
        begin
            reg0_1 = R0_control_signals1_0 | R0_control_signals1_1;
            reg1_1 = R1_control_signals1_0 | R1_control_signals1_1;
            reg2_1 = R2_control_signals1_0 | R2_control_signals1_1;
            reg3_1 = R3_control_signals1_0 | R3_control_signals1_1;
            response_signals1[1]=response_signals1_1;
        end
        else
        begin
            response_signals1[1]=1'b0;
            reg0_1 = R0_control_signals1_0;
            reg1_1 = R1_control_signals1_0;
            reg2_1 = R2_control_signals1_0;
            reg3_1 = R3_control_signals1_0;
        end
    end
end

wire [4:0] sig01_2;
reg [19:0] reg0_2,reg1_2,reg2_2,reg3_2;
assign sig01_2 = (reg0_1[4:0] & R0_control_signals1_2[4:0]) |
(reg3_1[4:0] & R3_control_signals1_2[4:0]) | (reg1_1[4:0] &
R1_control_signals1_2[4:0]) | (reg2_1[4:0] & R2_control_signals1_2[4:0]);
always@(*)
begin
    if(sig01_2==5'b0)
    begin
        reg0_2 = reg0_1 | R0_control_signals1_2;
    end

```

```

    reg1_2 = reg1_1 | R1_control_signals1_2;
    reg2_2 = reg2_1 | R2_control_signals1_2;
    reg3_2 = reg3_1 | R3_control_signals1_2;
    response_signals1[2]=response_signals1_2;

end
else
begin
    response_signals1[2] =1'b0;
    reg0_2 = reg0_1;
    reg1_2 = reg1_1;
    reg2_2 = reg2_1;
    reg3_2 = reg3_1;
end
end

wire [4:0] sig012_3;
assign sig012_3 = (reg0_2[4:0] & R0_control_signals1_3[4:0]) |
(reg3_2[4:0] & R3_control_signals1_3[4:0]) | (reg1_2[4:0] &
R1_control_signals1_3[4:0]) | (reg2_2[4:0] & R2_control_signals1_3[4:0]);
always@(*)
begin
    if(sig012_3==5'b0)
    begin
        R0_control_signals1 = reg0_2 | R0_control_signals1_3;
        R1_control_signals1 = reg1_2 | R1_control_signals1_3;
        R2_control_signals1 = reg2_2 | R2_control_signals1_3;
        R3_control_signals1 = reg3_2 | R3_control_signals1_3;
        response_signals1[3] =response_signals1_3;
    end
    else
    begin
        response_signals1[3] =1'b0;
        R0_control_signals1 = reg0_2;
        R1_control_signals1 = reg1_2;
        R2_control_signals1 = reg2_2;
        R3_control_signals1 = reg3_2;
    end
end
end

endmodule

```

Module Processing_Unit

```
/*
Module name:
    Processing_unit
Module Description:
    This Module contains demonstrates the working of our processor.
Pin Description:
    Clock: 1 bit input port for the clock signal.
    Reset: 1 bit input port for the reset signal.
    master_response: 1 bit input which shows the availability of processor
(1->Master has accepted the request of processor)
    data_from_router: 9bit input reg which contains the data received from
the router to its corresponding processor. [8] ->last flit
    data_to_router: 9bit output reg which contains the data processor
sends to its corresponding router. [8] ->last flit
    request_transfer: 1bit output where processor requests master for
allocation (1->request is high)
    which_processor: 2 bit output register corresponding to the
destination router/processor
    processor_ready: 1 bit output indicates the processor is free to send
out data
    tb_request: 1 bit input which is the value user gives to processor to
use as request_transfer
    tb_processor: 2 bit output reg is the value user gives to processor to
use as which_processor
    tb_len: 8 bit output reg is the value user gives to processor the
burst size/number of packets (essentially the 8bit data we are using)
*/
module Processing_unit(
    input clock,
    input reset,
    input master_response,
    input [8:0] data_from_router,
    output [8:0] data_to_router,
    output request_transfer,
```

```

        output [1:0] which_processor,
        output processor_ready,
        output [8:0] data_got,
        input tb_request,
        input [1:0] tb_processor,
        input [7:0] tb_len
    ) ;
        reg [8:0] data_got1, data_to_router1;
        reg [1:0] which_processor1;
        reg request_transfer1;
        assign data_got=data_got1;
        assign which_processor=which_processor1;
        assign data_to_router=data_to_router1;
        assign request_transfer=request_transfer1;

    always@(posedge clock)
    begin
        begin
            data_got1<=data_from_router; //assign tlast which checks
whether last element or not
            //tlast_prev is assigned 1 clock cycle after tlast is
evaluated and assigned
        end
    end

//variables

//To check a valid request
    reg request_line;
//Temporarily stores value of processor_ready till clockedge
    reg processor_ready1;

//Keeps a note of packets received
    reg [7:0]counter_value;
//Keeps a note of packets received temporarily till clockedge
    reg [7:0]counter_value1;

// Checks whether last bit or no
    reg tlast;

```

```

//a variable that is a copy of tlast one clockcycle prior
reg tlast_prev;

//temporarily stores value of tlast till clockedge
reg tlast1;

always@(*)
begin
    request_line=tb_request & processor_ready1; //high only when
processor is free and user has asked for a request transfer from processor
end

always@(*)
begin
    if(reset==1'b1)
        begin
            which_processor1=2'b00; //set destination processor to default
        end
    else
        begin
            which_processor1=tb_processor; //and set which_processor
according to the destination the user has set
        end
    end

always@(*)
begin
    if(reset==1'b1)
        begin
            request_transfer1=1'b0; //if reset all requests are null and
void
        end
    else
        begin
            request_transfer1=request_line; //else raise request_transfer
if a valid request
        end
    end

always@(posedge clock or posedge reset)

```

```

begin
    if(reset==1'b1)
        begin
            tlast_prev<=1'b0; //if reset, no data, therefore definitely
not the last bit
        end
    else
        begin
            tlast_prev=tlast; //assign tlast which checks whether last
element or not
            //tlast_prev is assigned 1 clock cycle after tlast is
evaluated and assigned
        end
end

reg processor_ready2;
always@(*)
begin
    if(master_response==1'b1)
        begin
            processor_ready2=1'b0; //if master commands to utilise
processor, it is no more free to transmit
        end
    else
        begin
            processor_ready2=1'b1;
        end
end

always@(*)
begin
    if(reset || tlast_prev || master_response)
        begin
            processor_ready1<=processor_ready2;
        end
end

//set the values of processor_ready
assign processor_ready=processor_ready1;

```

```

//processor_ready is in 1 clock cycle lag as tlast_prev is assigned 1
clock cycle after tlast is evaluated and assigned

always@(*)
begin
    if(reset==1'b1)
        begin
            tlast1=1'b0; //if reset, no data, therefore definitely not the
last bit
            end
        else if(counter_value1==tb_len)
            begin
                tlast1=1'b1; //if the number of packets to be sent are
transmitted (counter = tb_len) it means it is the last packets so raise
tlast to 1
                end
            else
                begin
                    tlast1=1'b0; //else continue with 0 as either it is in middle
of transfer or no transfer at all, so no last flit
                    end
            end
        end

always@(posedge clock or posedge reset)
begin
    if(reset==1'b1)
        begin
            tlast<=1'b0; //if reset, no data, therefore definitely not the
last bit
            end
        else
            begin
                tlast<=tlast1; //assign values as computed above
            end
        end
    end

always@(*)
begin
    if(request_line==1 || counter_value==8'b11111111)
        begin

```

```

        counter_value1=8'b00000001; //if new request or overflow start
counter from 1
    end
else
begin
    counter_value1=counter_value+1; //else increment counter by 1
end
end

always@(posedge clock or posedge reset) //next packet at clockedge
begin
    if(reset==1'b1)
begin
    counter_value<=8'b00000000; //if reset, restart from 1
end
else
begin
    counter_value<=counter_value1; //else assign the value
obtained from computation shown above
end
end

always@(posedge clock or posedge reset)
begin
    if(reset==1'b1)
begin
    data_to_router1<=9'b0; //if it was a reset, send 0 as output
end
else
begin
    data_to_router1<={tlast,counter_value[7:0]}; //else
concatenate the last flit with counter value
end
end

endmodule

```

Module router

```
/*
Module name:
    router
Module Description:
    This Module contains basic Building block of NoC Mesh.
    It is a 5x5 (NEWS,P) router with 5 input and 5 output ports.
Data Packet:
    Packet size of data is 1 byte, 1 bit for last flit (indicates The
last Packet)
Module Pin Description :
    Clock: 1 bit input port for the clock signal.
    Reset: 1 bit input port for the reset signal.
    SetNR: 1 bit input port to set the North side port ready reg by
master.
    SetSR: 1 bit input port to set the South side port ready reg by
master.
    SetER: 1 bit input port to set the East side port ready reg by master.
    SetWR: 1 bit input port to set the West side port ready reg by master
    SetPR: 1 bit input port to set the Processor side port ready reg by
master.
    Select North: 3 bit input port to select the data for North side port.
    Select South: 3 bit input port to select the data for South side port.
    Select East: 3 bit input port to select the data for East side port.
    Select West: 3 bit input port to select the data for West side port.
    Select Processor: 3 bit input port to select the data for Processor
side port.
    Data North: 9 bit input port for the data from the North side port.
[8] ->last flit
    Data South: 9 bit input port for the data from the South side port.
[8] ->last flit
    Data East: 9 bit input port for the data from the East side port. [8]
->last flit
    Data West: 9 bit input port for the data from the West side port. [8]
->last flit
    Data Processor: 9 bit input port for the data from the Processor side
port. [8] ->last flit
    Output North: 9 bit output port for the data to the North side port.
[8] ->last flit
```

```

    Output South: 9 bit output port for the data to the South side port.
[8] ->last flit
    Output East: 9 bit output port for the data to the East side port. [8]
->last flit
    Output West: 9 bit output port for the data to the West side port. [8]
->last flit
    Output Processor: 9 bit output port for the data to the Processor side
port. [8] ->last flit
    North Ready: 1 bit output port to indicate the availability of the
North side port line.
    South Ready: 1 bit output port to indicate the availability of the
South side port line.
    East Ready: 1 bit output port to indicate the availability of the East
side port line.
    West Ready: 1 bit output port to indicate the availability of the West
side port line.
    Processor Ready: 1 bit output port to indicate the availability of the
Processor side port line.

Additional Notes:
regPR: Check whether the path is busy or not. 0->busy, 1->free

Output at each direction can have data from NEWS,P. Therefore,we need
3 select bits in each Router_control_signals in master.v

*/
module router(
    input clock,
    input reset,
    input [2:0] select_north,
    input [2:0] select_south,
    input [2:0] select_east,
    input [2:0] select_west,
    input [2:0] select_processor,
    input [8:0] data_north,
    input [8:0] data_south,
    input [8:0] data_east,
    input [8:0] data_west,
    input [8:0] data_processor,
    output [8:0] output_north,

```

```

    output [8:0] output_south,
    output [8:0] output_east,
    output [8:0] output_west,
    output [8:0] output_processor,
    output north_ready,
    output south_ready,
    output east_ready,
    output west_ready,
    output processor_ready,
    input SetNR,
    input SetSR,
    input SetER,
    input SetWR,
    input SetPR
);

// internal working

// temporary variables of output used till clockedge is not reached
reg [8:0]
output_north1,output_south1,output_east1,output_west1,output_processor1;
reg [8:0]
output_north2,output_south2,output_east2,output_west2,output_processor2;
assign output_north=output_north2;
assign output_south=output_south2;
assign output_east=output_east2;
assign output_west=output_west2;
assign output_processor=output_processor2;

// check whether route is free or not
reg regNR,regSR,regER,regWR,regPR;

//temporary variable used till clockedge is not reached
reg regNR1,regSR1,regER1,regWR1,regPR1;

// Data Packet Routing (Crossbar Working)

always@(*)

```

```

begin
    case(select_north) //choosing which input of router will go as
output on north, essentially a multiplexer
        3'b000: output_north1 = data_north;
        3'b001: output_north1 = data_south;
        3'b010: output_north1 = data_east;
        3'b011: output_north1 = data_west;
        3'b100: output_north1 = data_processor;
    default: output_north1 = 9'b0000000000;

    endcase
end

always@(*)
begin
    case(select_south) //choosing which input of router will go as
output on south, essentially a multiplexer
        3'b000: output_south1 = data_north;
        3'b001: output_south1 = data_south;
        3'b010: output_south1 = data_east;
        3'b011: output_south1 = data_west;
        3'b100: output_south1 = data_processor;
    default: output_south1 = 9'b0000000000;

    endcase
end

always@(*)
begin
    case(select_east) //choosing which input of router will go as
output on east, essentially a multiplexer
        3'b000: output_east1 = data_north;
        3'b001: output_east1 = data_south;
        3'b010: output_east1 = data_east;
        3'b011: output_east1 = data_west;
        3'b100: output_east1 = data_processor;
    default: output_east1 = 9'b0000000000;

    endcase
end

```

```

always@(*)
begin
    case(select_west) //choosing which input of router will go as
output on west, essentially a multiplexer
        3'b000: output_west1 = data_north;
        3'b001: output_west1 = data_south;
        3'b010: output_west1 = data_east;
        3'b011: output_west1 = data_west;
        3'b100: output_west1 = data_processor;
    default: output_west1 = 9'b0000000000;

    endcase
end

always@(*)
begin
    case(select_processor) //choosing which input of router will go as
output on processor, essentially a multiplexer
        3'b000: output_processor1 = data_north;
        3'b001: output_processor1 = data_south;
        3'b010: output_processor1 = data_east;
        3'b011: output_processor1 = data_west;
        3'b100: output_processor1 = data_processor;
    default: output_processor1 = 9'b0000000000;

    endcase
end

// Output Port Selection
always@(*)
begin
    if(output_north1[8]==1'b1)
    begin
        regNR1=1; // If last bit of data packet, make route free
    end
    else if(SetNR==1'b1)
    begin
        regNR1=0; // The master has granted permission to route
through this direction, make the path busy (for next incoming packet)
    end
end

```

```

    end
  else
    begin
      regNR1=regNR; //Else continue with the previous value
    end
  end
always@(*)
begin
  if(output_south1[8]==1'b1)
  begin
    regSR1=1;// If last bit of data packet, make route free
  end
  else if(SetSR==1'b1)
  begin
    regSR1=0; // The master has granted permission to route
through this direction, make the path busy (for next incoming packet)
  end
  else
  begin
    regSR1=regSR; //Else continue with the previous value
  end
end
always@(*)
begin
  if(output_east1[8]==1'b1)
  begin
    regER1=1;// If last bit of data packet, make route free
  end
  else if(SetER==1'b1)
  begin
    regER1=0; // The master has granted permission to route
through this direction, make the path busy (for next incoming packet)
  end
  else
  begin
    regER1=regER; //Else continue with the previous value
  end
end
always@(*)
begin

```

```

if(output_west1[8]==1'b1)
begin
    regWR1=1;// If last bit of data packet, make route free
end
else if(SetWR==1'b1)
begin
    regWR1=0; // The master has granted permission to route
through this direction, make the path busy (for next incoming packet)
end
else
begin
    regWR1=regWR; //Else continue with the previous value
end
end
always@(*)
begin
    if(output_processor1[8]==1'b1)
begin
    regPR1=1;// If last bit of data packet, make route free
end
else if(SetPR==1'b1)
begin
    regPR1=0; // The master has granted permission to route
through this direction, make the path busy (for next incoming packet)
end
else
begin
    regPR1=regPR; //Else continue with the previous value
end
end
//assigning output at posedge of clock or reset
always@(posedge clock)
begin
    output_north2<=output_north1; //assign output according to the
output of MUX, at clock edge
end
always@(posedge clock or posedge reset)
begin
    if(reset==1'b1)
begin

```

```

        regNR<=1; //if reset, all paths are made free
    end
else
begin
    regNR<=regNR1; //assign path freeness according to the
algorithm done above, at clock edge
end
always@(posedge clock)
begin
    output_south2<=output_south1; //assign output according to the
output of MUX, at clock edge
end
always@(posedge clock or posedge reset)
begin
    if(reset==1'b1)
begin
    regSR<=1; //if reset, all paths are made free
end
else
begin
    regSR<=regSR1; //assign path freeness according to the
algorithm done above, at clock edge
end
end
always@(posedge clock)
begin
    output_east2<=output_east1; //assign output according to the
output of MUX, at clock edge
end
always@(posedge clock or posedge reset)
begin
    if(reset==1'b1)
begin
    regER<=1; //if reset, all paths are made free
end
else
begin
    regER<=regER1; //assign path freeness according to the
algorithm done above, at clock edge

```

```

        end
    end
    always@ (posedge clock)
    begin
        output_west2<=output_west1; //assign output according to the
output of MUX done above, at clock edge
    end
    always@ (posedge clock or posedge reset)
    begin
        if(reset==1'b1)
        begin
            regWR<=1; //if reset, all paths are made free
        end
        else
        begin
            regWR<=regWR1; //assign path freeness according to the
algorithm done above, at clock edge
        end
    end
    always@ (posedge clock)
    begin
        output_processor2<=output_processor1; //assign output according to
the output of MUX done above, at clock edge
    end
    always@ (posedge clock or posedge reset)
    begin
        if(reset==1'b1)
        begin
            regPR<=1;//if reset, all paths are made free
        end
        else
        begin
            regPR<=regPR1; //assign path freeness according to the
algorithm done above, at clock edge
        end
    end

    // Assigning ussage bit (freeness of paths found using regNR, regSR,
regER, regWR, regPR)
    assign north_ready=regNR;

```

```

    assign south_ready=regSR;
    assign east_ready=regER;
    assign west_ready=regWR;
    assign processor_ready=regPR;

endmodule

```

Test Bench 1:

Verilog Code

```

`include "main.v"
module mesh_tb();

    reg clk=0;
    reg reset=0;
    wire [3:0] processor_ready_signals;
    reg [10:0] p0_configure,p1_configure,p2_configure,p3_configure;
    // wire [19:0] temp_path_block_signals;
    wire [8:0]
p0_recieve_data,p1_recieve_data,p2_recieve_data,p3_recieve_data;
    reg block_all_paths;
    integer i;
    parameter No_DATA=18'b0;
    mesh m1 (.clock(clk),
              .reset(reset),
              .r0_input(No_DATA),
              .r1_input(No_DATA),
              .r2_input(No_DATA),
              .r3_input(No_DATA),
              .p0_configure(p0_configure),
              .p1_configure(p1_configure),
              .p2_configure(p2_configure),
              .p3_configure(p3_configure),
              .block_all_paths(block_all_paths),
              .processor_ready_signals(processor_ready_signals),
              .p0_recieve_data(p0_recieve_data),
              .p1_recieve_data(p1_recieve_data),
              .p2_recieve_data(p2_recieve_data),

```

```

.p3_recieve_data(p3_recieve_data) ,
.r0_output() ,
.r1_output() ,
.r2_output() ,
.r3_output()
//.temp_path_block_signals(temp_path_block_signals)
);

initial begin
$dumpfile("noc_sim.vcd");
$dumpvars(0,mesh_tb);

reset = 1'b1;
block_all_paths = 1'b0;
#16 reset = 1'b0;

for(i = 0; i < 4; i = i + 1) begin
  case(i)
    0: begin
      p0_configure = 11'b00000001001;
      p1_configure = 11'b00000001001;
      p2_configure = 11'b00000001001;
      p3_configure = 11'b00000001001;
    end
    1: begin
      p0_configure = 11'b00000001011;
      p1_configure = 11'b00000001011;
      p2_configure = 11'b00000001011;
      p3_configure = 11'b00000001011;
    end
    2: begin
      p0_configure = 11'b00000001101;
      p1_configure = 11'b00000001101;
      p2_configure = 11'b00000001101;
      p3_configure = 11'b00000001101;
    end
    3: begin
      p0_configure = 11'b00000001111;
      p1_configure = 11'b00000001111;
      p2_configure = 11'b00000001111;
    end
  endcase
end

```

```

    p3_configure = 11'b00000001111;
end
endcase
#20; // Delay to ensure parallelism in simulation
end

reset = 1'b1;
#16 reset = 1'b0;

// #4000 $finish;
end

initial begin

#1 reset = 1'b1;
#16 reset = 1'b0;
#20 p0_configure = 11'b00000001001;
#20 p0_configure = 11'b00000010001;
#1 p0_configure = 11'b00000100001;
#1 p0_configure = 11'b00000001011;
#1 p0_configure = 11'b00000101011;
#1 p0_configure = 11'b00000111011;

#20 p1_configure = 11'b00000010001;
#20 p1_configure = 11'b00000100001;
#20 p1_configure = 11'b00000001101;
#20 p0_configure = 11'b00000001011;
#20 p0_configure = 11'b00000010011;

#1 p3_configure = 11'b00000100101;
#20 p3_configure = 11'b00000100001;
#20 p0_configure = 11'b00000100011;
#20 p0_configure = 11'b00000001101;
#20 p0_configure = 11'b00000010101;

#20 p2_configure = 11'b00000010001;
#20 p2_configure = 11'b00000100001;

```

```

#20 p2_configure = 11'b00000001101; #20 p2_configure =
11'b00000010001;
#20 p2_configure = 11'b00000100001;

#20 p1_configure = 11'b00000010001;
#20 p1_configure = 11'b00000100001;
#20 p1_configure = 11'b00000001101;
#20 p2_configure = 11'b00000001101;
#20 p0_configure = 11'b00000100101;
#20 p0_configure = 11'b00000100111;
#20 p0_configure = 11'b00000001111;
#20 p0_configure = 11'b00000010111;

//##20 p1_configure = 11'b00000001011;
//##20 p1_configure = 11'b00000001001;
#20 p1_configure = 11'b00000001011;
#20 p1_configure = 11'b000000010011;
#20 p1_configure = 11'b00000100011;

#20 p2_configure = 11'b00000001101;
#20 p2_configure = 11'b00000010101;
#20 p2_configure = 11'b00000100101;
#20 p1_configure = 11'b00000001001;

#20 p2_configure = 11'b00000010001;
#20 p2_configure = 11'b00000100001;
#20 p2_configure = 11'b00000001101;

#1 reset = 1'b1;
#16 reset = 1'b0;

#20 p1_configure = 11'b00000010001;
#20 p1_configure = 11'b00000100001;
#20 p1_configure = 11'b00000001101;
#20 p1_configure = 11'b00000010101;

#20 p0_configure = 11'b00000100011;
#20 p0_configure = 11'b00000001101;

```

```

#20 p0_configure = 11'b00000010101;
#20 p1_configure = 11'b00000100101;
#20 p1_configure = 11'b00000100111;

#20 p3_configure = 11'b00000100101;
#20 p3_configure = 11'b00000100001;
#20 p1_configure = 11'b00000001111;
#20 p1_configure = 11'b00000010111;

//#20 p2_configure = 11'b00000001011;
//#20 p2_configure = 11'b00000001001;
#20 p2_configure = 11'b00000001011;
#20 p2_configure = 11'b000000011011;
#20 p2_configure = 11'b000000100011;
#20 p2_configure = 11'b00000001001;

#20 p3_configure = 11'b00000001011;
#20 p3_configure = 11'b00000010011;
#20 p3_configure = 11'b00000100011;
#20 p3_configure = 11'b00000001101;

#20 p0_configure = 11'b00000100011;
#20 p0_configure = 11'b00000001101;
#20 p0_configure = 11'b00000010101;
#20 p2_configure = 11'b00000010001;

#20 p1_configure = 11'b00000010001;
#20 p1_configure = 11'b00000100001;
#20 p1_configure = 11'b00000001101;
#20 p2_configure = 11'b00000010001;
#20 p2_configure = 11'b000000010001;
#20 p2_configure = 11'b00000001101;
#20 p2_configure = 11'b00000010101;
#20 p2_configure = 11'b00000010101;
#20 p2_configure = 11'b00100100111;
#20 p2_configure = 11'b00000001111;

#1 reset = 1'b1;
#16 reset = 1'b0;

```

```

#20      p2_configure = 11'b000110010111;

//#20 p3_configure = 11'b00000001011;
//#20 p3_configure = 11'b00000001001;
#20  p3_configure = 11'b00000001011;
#20  p3_configure = 11'b00000010011;
#20 p3_configure = 11'b00000100011;
    #20 p3_configure = 11'b00000001101;
    #20 p3_configure = 11'b00000010101;

#20 p0_configure = 11'b00000100011;
    #20 p0_configure = 11'b00000001101;
    #20 p0_configure = 11'b00000010101;
#20 p3_configure = 11'b00000100101;
    #20 p3_configure = 11'b00000100001;

#20  p2_configure = 11'b00000001101;
    #20  p2_configure = 11'b00000010101;
#20  p2_configure = 11'b00000100101;

#20 p1_configure = 11'b00000010001;
#20  p1_configure = 11'b00000100001;
    #20 p1_configure = 11'b00000001101;
    #20 p3_configure = 11'b00000001001;
#20 p3_configure = 11'b00000010001;
#20 p3_configure = 11'b000000100111;
    #20 p3_configure = 11'b00000001111;
#20 p3_configure = 11'b000000010111;

#1 reset = 1'b1;
#16 reset = 1'b0;

#20 p0_configure = 11'b00000001011;
#20 p0_configure = 11'b00000011011;
#20 p0_configure = 11'b00000001011;
#20 p0_configure = 11'b000000101011;
#20 p0_configure = 11'b000000111011;

```

```

#500000 $finish;
end

always #10 clk = ~clk;

endmodule

```

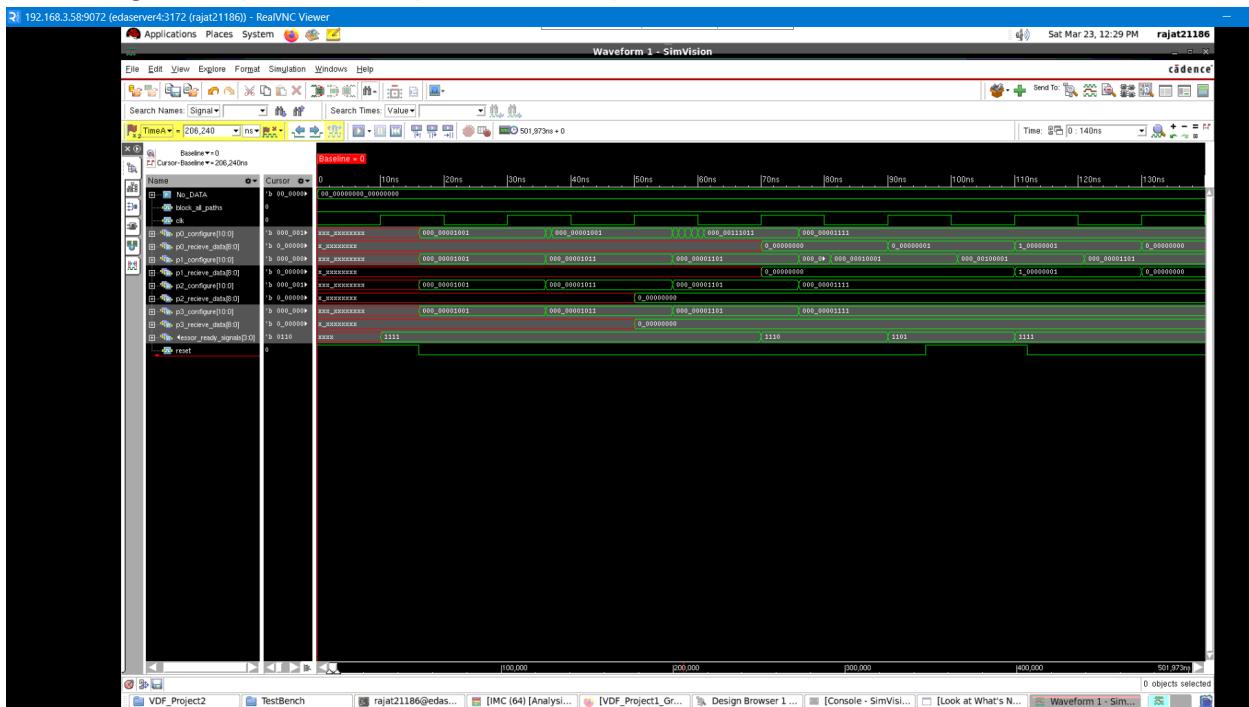
Explanation of test vectors:

Here we have given Clock, reset and processor configuration signals to test the functionality of our code. This testbench is meant for only functionality testing and not for code coverage. We have configured our Test vectors in such a way that we can firstly test our acknowledgement protocol and the overall working of Processing units, router circuitry and the master module.

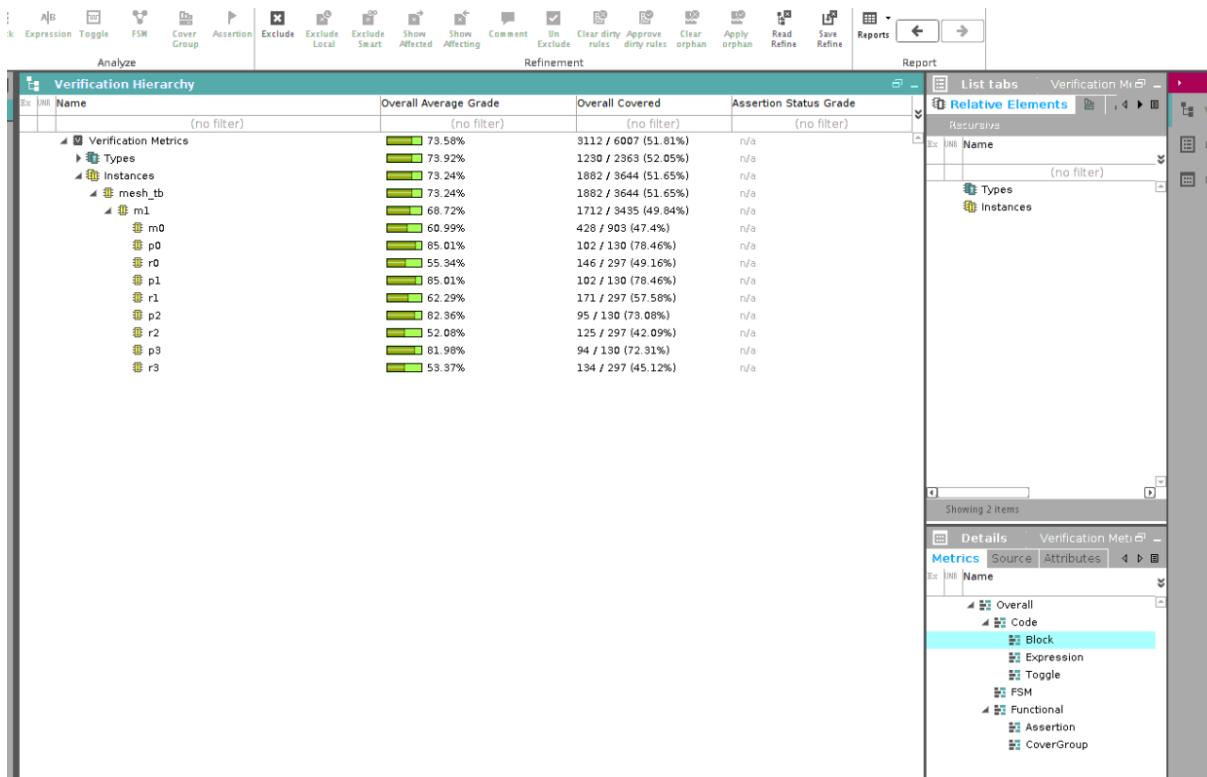
Multiple Test Vectors in this case are tested to ensure structural completeness and the functional correctness of the overall code.

Output Waveform for Test Bench 1:

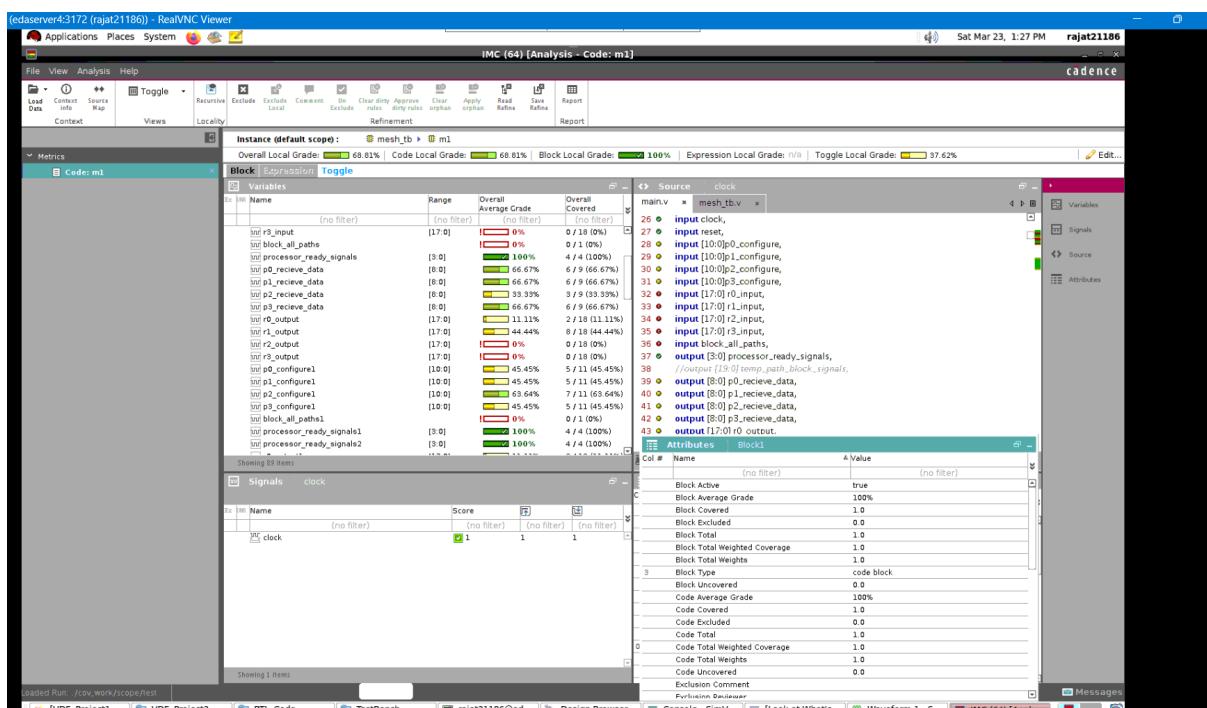
(Showing the input and output variables)



Code Coverage Report for Test Bench 1:



Toggle Coverage:



Block Coverage:

The screenshot shows the Cadence C Viewer interface for an analysis named 'IMC (64) [Analysis - Code: m1]'. The top status bar indicates the date as 'Sat Mar 23, 1:25 PM' and the user as 'rajat21186'. The main window has tabs for 'Blocks', 'Expression', and 'Toggle'. The 'Blocks' tab is selected, showing a table of code blocks with columns for Index, Block Type, Source Line, and Score. A total of 8 items are shown. The 'Source' tab displays the Verilog code for 'main.v' and 'mesh_tb.v'. The 'Attributes' tab lists various coverage metrics for the selected block, such as Block Active (true), Block Average Grade (100%), and Block Total (1.0). The bottom status bar shows the system path: 'ect2' > 'RTL Code' > 'TestBench' > 'rajat21186@ed...' > 'Design Browser...' > 'Console - SimV...' > 'Look at What's...' > 'Waveform 1 - S...' > 'IMC (64) [Analysis...]

Explanation of coverage:

We see a 100% block coverage by this testbench.

We see around 74% overall code coverage by this testbench. We have covered various cases including single transfers, parallel transfers and blocked transfers. We have also tested the functionality of reset and burst transfers. In fact our priority Circuitry has been also tested by the testbench. We verified the Result by manual speculation of the waveform. Everything here seems to be good except the toggling report. The tool reports only 38% coverage of toggles. Fortunately this report does not necessarily mean that our test vectors are incapable of covering the bench. The main reason for less coverage is because bits of some specific registers have not toggled in this testbench. Interestingly this is mainly due to logical assumptions under which we have designed this system. For example, we have a counter register in our P_unit module, since this is a counter circuitry, it would toggle from 7'b0 to {{7'b1}} always in increasing manner, but this won't account for all possible toggles. This is functionally correct and thus cannot have toggle coverage. Similarly in our design various toggles are not possible to

account for, as they are not practically possible as a result we don't see a good toggle coverage.

Test Bench 2:

Verilog Code

```
`include "main.v"

module tb_noc();
    reg clk=0;
    reg reset=0;
    wire [3:0] processor_ready_signals;
    reg [10:0] p0_configure,p1_configure,p2_configure,p3_configure;
    wire [8:0]
p0_recieve_data,p1_recieve_data,p2_recieve_data,p3_recieve_data;
    reg block_all_paths;
    parameter No_DATA=18'b0;
    mesh m1 (
        .clock(clk),
        .reset(reset),
        .r0_input(No_DATA),
        .r1_input(No_DATA),
        .r2_input(No_DATA),
        .r3_input(No_DATA),
        .p0_configure(p0_configure),
        .p1_configure(p1_configure),
        .p2_configure(p2_configure),
        .p3_configure(p3_configure),
        .block_all_paths(block_all_paths),
        .processor_ready_signals(processor_ready_signals),
        .p0_recieve_data(p0_recieve_data),
        .p1_recieve_data(p1_recieve_data),
        .p2_recieve_data(p2_recieve_data),
        .p3_recieve_data(p3_recieve_data),
        .r0_output(),
        .r1_output(),
        .r2_output(),
        .r3_output()
    );

```

```

initial begin
    $dumpfile("noc_sim.vcd");
    $dumpvars(0,tb_noc);
    reset=1'b0;
    block_all_paths=1'b0;
    p0_configure=11'b0;
    p1_configure=11'b0;
    p2_configure=11'b0;
    p3_configure=11'b0;
    #1 reset = 1'b1;
    #16 reset = 1'b0;

    #10 p0_configure = 11'b00000001011;
    #30 p0_configure = 11'b00000001010;
    p3_configure = 11'b00000001111;
    #10 p0_configure = 11'b00000001011;
    #10 p1_configure = 11'b00000000101;

    #20 p2_configure = 11'b00000001011;
    #40 p2_configure = 11'b00000001010;
    p3_configure = 11'b00000001111;
    #20 p2_configure = 11'b00000001011;

    #40 p3_configure = 11'b00000001011;
    #30 p3_configure = 11'b00000001010;
    p0_configure = 11'b00000001001;
    #10 p3_configure = 11'b00000001011;

    #60 p1_configure = 11'b000000001011;

    #50 block_all_paths=1'b1;
    #50 block_all_paths=1'b0;

    #1 reset = 1'b1;
    #16 reset = 1'b0;
    #200;

    #10 p0_configure = 11'b00000001101;
    p3_configure = 11'b00000001111;
    #10 p0_configure = 11'b00000001101;

```

```
#10 p1_configure = 11'b000000001010;

#20 p2_configure = 11'b00000001101;
#1 p3_configure = 11'b00000001111;

#40 p3_configure = 11'b00000001101;
#1 p1_configure = 11'b00000001010;
#10 p3_configure = 11'b00000001101;

#60 p1_configure = 11'b000000001101;
#10 p0_configure = 11'b000000001000;
#10 p1_configure = 11'b000000001101;

#50 block_all_paths=1'b1;
#50 block_all_paths=1'b0;

#1 reset = 1'b1;
#16 reset = 1'b0;

#200;

#60 p1_configure = 11'b000000001101;
#1 p3_configure = 11'b000000001101;
#10 p1_configure = 11'b000000001101;

#50 block_all_paths=1'b1;
#50 block_all_paths=1'b0;

#200;

#10 p0_configure = 11'b00000001111;
#10 p1_configure = 11'b00000001011;
#10 p0_configure = 11'b00000001111;
#10 p1_configure = 11'b00000000110;

#20 p2_configure = 11'b00000001111;
#10 p0_configure = 11'b00000001111;
#10 p2_configure = 11'b00000001111;

#40 p3_configure = 11'b00000001111;
```

```
p0_configure = 11'b00000001111;
#10 p3_configure = 11'b00000001111;

#60 p1_configure = 11'b00000001111;

#40 reset = 1'b1;
#16 reset = 1'b0;

#50 block_all_paths=1'b1;
#50 block_all_paths=1'b0;

#200;

#20 p0_configure = 11'b00000010001;
#1 p1_configure = 11'b00000010011;
#1 p2_configure = 11'b00000010101;
#1 p3_configure = 11'b00000010111;

#50 reset = 1'b1;
#16 reset = 1'b0;

#50 block_all_paths=1'b1;
#50 block_all_paths=1'b0;

#40 reset = 1'b1;
#16 reset = 1'b0;
#200;

//p0_configure = 11'b00000001001;

//#10 p1_configure = 11'b00000001001;
//#30 p1_configure = 11'b00000001000;
//p3_configure = 11'b00000001111;
```

```

//#10 p1_configure = 11'b00000001001;

#2 p1_configure = 11'b00000001001;
#1 p2_configure = 11'b00000001001;
#1 p3_configure = 11'b00000001001;

#50 p1_configure = 11'b00000001001;
#1 p2_configure = 11'b00000001001;
#1 p3_configure = 11'b00000001001;

#70 reset = 1'b1;
#16 reset = 1'b0;

#50 block_all_paths=1'b1;
#50 block_all_paths=1'b0;

#70 reset = 1'b1;
#16 reset = 1'b0;
#200;

#2 p1_configure = 11'b00000001001;

#40 p1_configure = 11'b00000001000;

#2 p1_configure = 11'b00000001001;

#10000 $finish;
end

always #10 clk = ~clk;

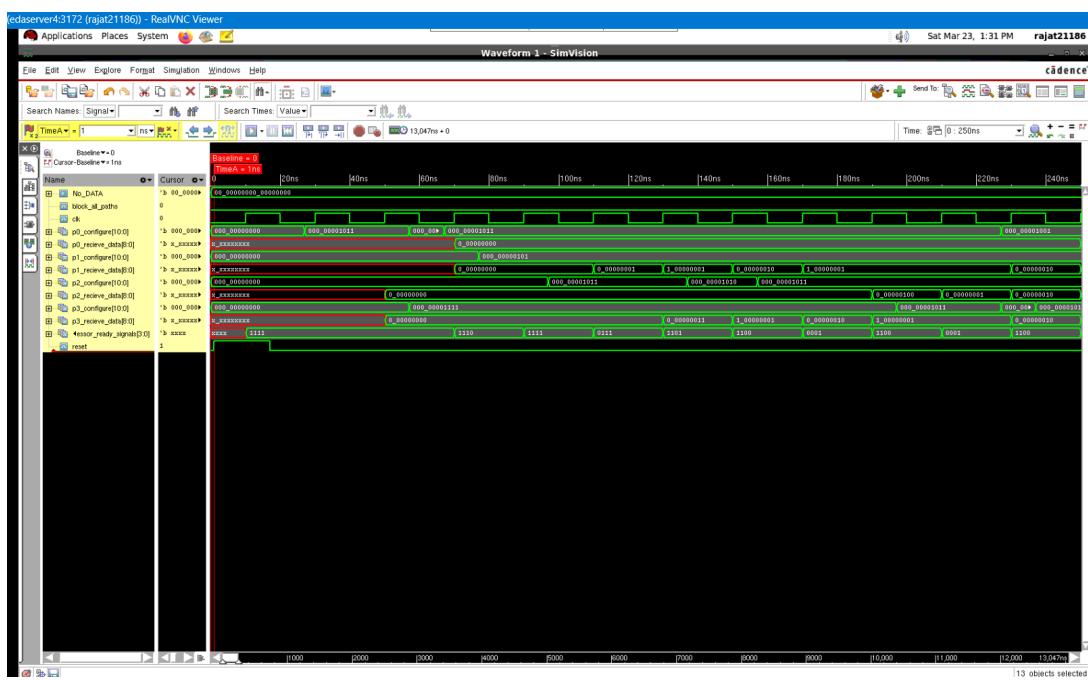
endmodule

```

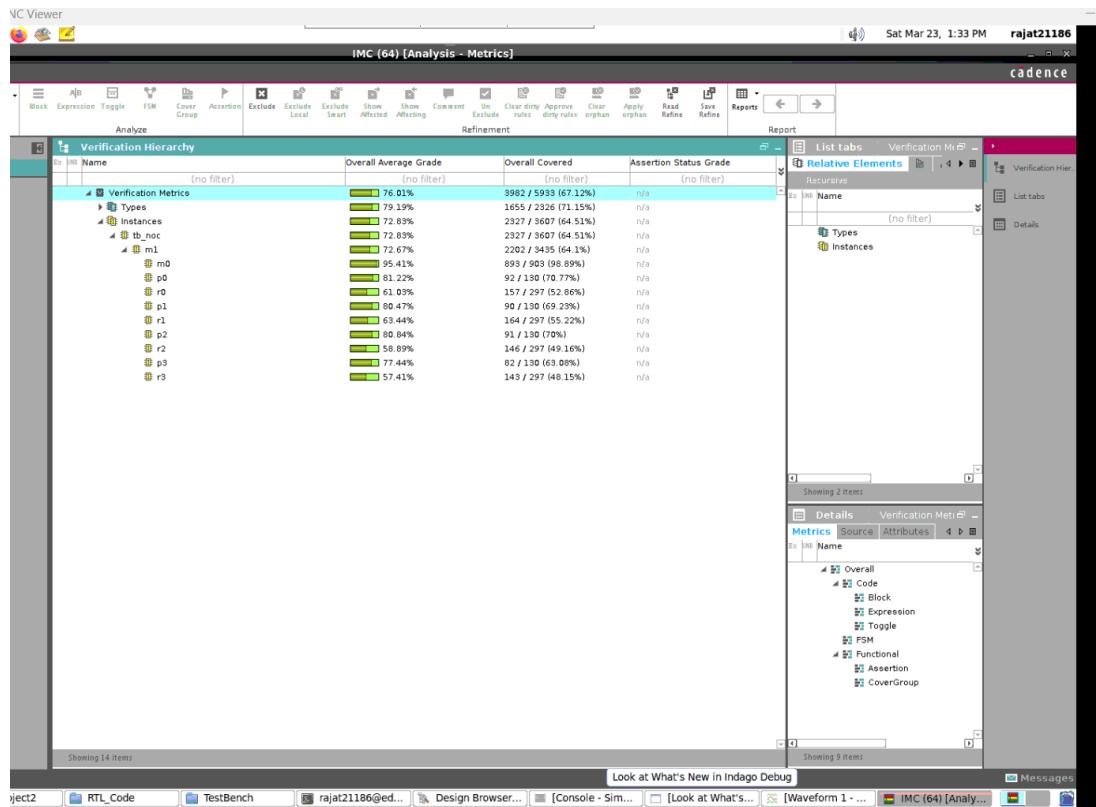
Explanation of test vectors:

Here we have given Clock, reset and processor configuration signals to test the functionality of our code. This testbench is meant for only master module coverage. We have configured our Test vectors in such a way that we can firstly test our acknowledgement protocol and the overall working of the master module. Multiple Test Vectors in this case are tested to ensure structural completeness and the functional correctness of the overall code. This is to test coverage as well as the functionality of the testbench.

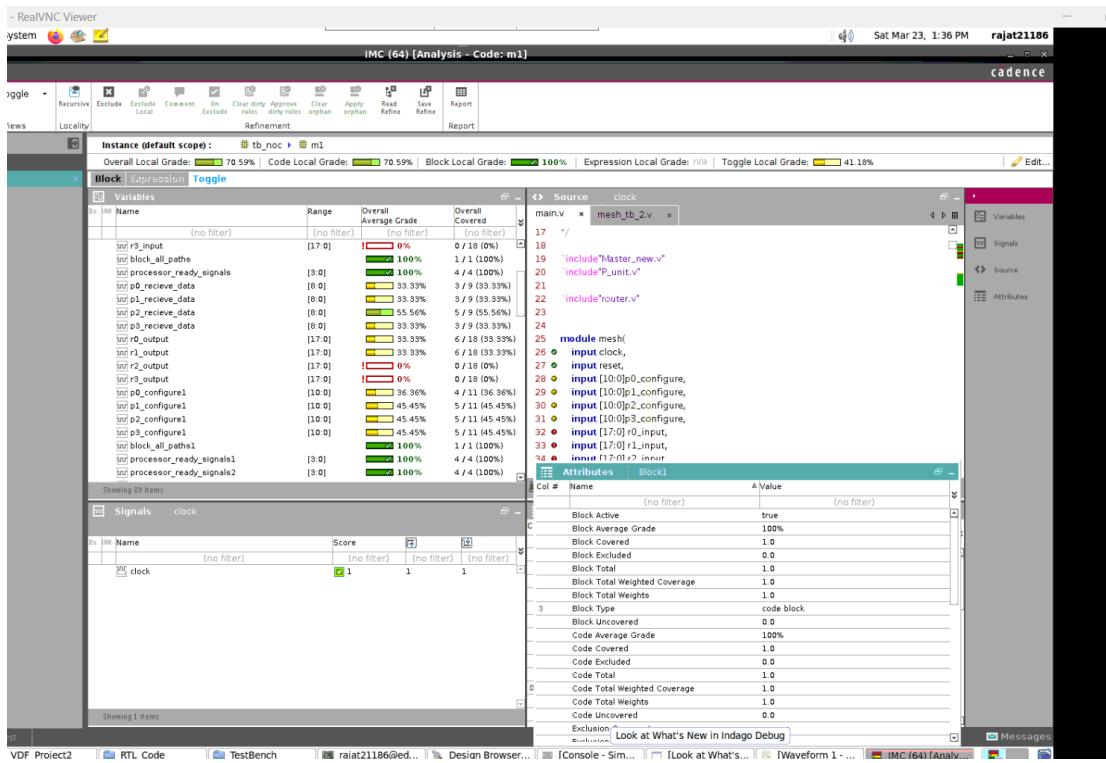
Output Waveform for Test Bench 2: (Showing the input and output variables)



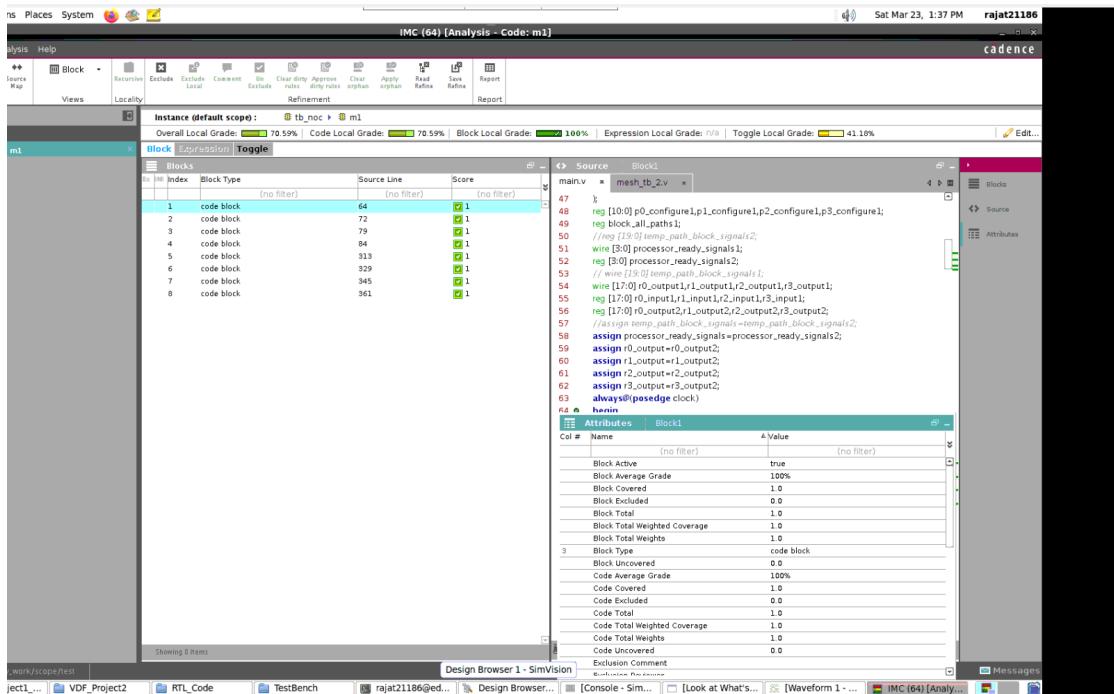
Code Coverage Report for Test Bench 2:



Toggle Coverage:



Block Coverage:



Explanation of coverage:

We see a 100% block coverage by this testbench. We see around 76% overall code coverage by this testbench. Here Master coverage has increased from 60% to 96 % from testbench. We have covered various cases including single transfers, parallel transfers and blocked transfers. We have also tested back-to-back transfers for all the processors. We have also tested the functionality of reset and block all path signals. In fact our priority Circuitry of the master has been also tested by the testbench. We verified the Result by manual speculation of the waveform. All possible path testing Has been carried out in this testbench.

Everything here seems to be good except the toggling report. The tool reports only 41% coverage of toggles. Fortunately this report does not necessarily mean that our test vectors are incapable of covering the bench. As discussed at the start we have a 11 bit configuration signal for the processor configuration, where the starting 8 bits denote the burst size. As a manual VLSI Engineer It is not possible to give test vectors which cover all toggles of these 8 bits. Also we have certain other inputs like router inputs denoting 9 bit data input from external ports to the specified router (a 18 bit signal per router). Similarly it is not logical as well as not possible for us to cover all toggles for this signal as this is a data packet signal. Similarly we observe various other signals in the design for which either it is not manually possible to cover all toggles or simply it is impractical to do so. Also, another main reason for less coverage is because bits of some specific registers have not toggled in this testbench. Interestingly this is mainly due to logical assumptions under which we have designed this system. For example, we have a counter register in our P_unit module, since this is a counter circuitry, it would toggle from 7'b0 to {{7'b1}} always in increasing manner, but this won't account for all possible toggles. This is functionally correct and thus cannot have toggle coverage. Similarly in our design various toggles are not possible to account for, as they are not practically possible as a result we don't see a good toggle coverage.

Test Bench 3:

Verilog Code

```
`include "main.v"

module tb_noc();
    reg clk=0;
    reg reset=0;
    wire [3:0] processor_ready_signals;
    reg [10:0] p0_configure,p1_configure,p2_configure,p3_configure;
    wire [8:0]
p0_recieve_data,p1_recieve_data,p2_recieve_data,p3_recieve_data;
    reg block_all_paths;
    integer i;
    parameter No_DATA=18'b0;
    mesh m1 (
        .clock(clk),
        .reset(reset),
        .r0_input(No_DATA),
        .r1_input(No_DATA),
        .r2_input(No_DATA),
        .r3_input(No_DATA),
        .p0_configure(p0_configure),
        .p1_configure(p1_configure),
        .p2_configure(p2_configure),
        .p3_configure(p3_configure),
        .block_all_paths(block_all_paths),
        .processor_ready_signals(processor_ready_signals),
        .p0_recieve_data(p0_recieve_data),
        .p1_recieve_data(p1_recieve_data),
        .p2_recieve_data(p2_recieve_data),
        .p3_recieve_data(p3_recieve_data),
        .r0_output(),
        .r1_output(),
        .r2_output(),
        .r3_output()
    );
    initial begin
        $dumpfile("noc_sim.vcd");
```

```

$dumpvars(0,tb_noc);
reset=1'b0;
block_all_paths=1'b0;
p0_configure=11'b0;
p1_configure=11'b0;
p2_configure=11'b0;
p3_configure=11'b0;
#1 reset = 1'b1;
#16 reset = 1'b0;

#10 p0_configure = 11'b00000001011;
#30 p0_configure = 11'b00000001010;
p3_configure = 11'b00000001111;
#10 p0_configure = 11'b00000001011;
#10 p1_configure = 11'b00000000101;

#20 p2_configure = 11'b00000001011;
#40 p2_configure = 11'b00000001010;
p3_configure = 11'b00000001111;
#20 p2_configure = 11'b00000001011;

#40 p3_configure = 11'b00000001011;
#30 p3_configure = 11'b00000001010;
p0_configure = 11'b00000001001;
#10 p3_configure = 11'b00000001011;

#60 p1_configure = 11'b000000001011;

#50 block_all_paths=1'b1;
#50 block_all_paths=1'b0;

#1 reset = 1'b1;
#16 reset = 1'b0;
#200;

#10 p0_configure = 11'b00000001101;
p3_configure = 11'b00000001111;
#10 p0_configure = 11'b00000001101;
#10 p1_configure = 11'b000000001010;

```

```

#20 p2_configure = 11'b00000001101;
#1 p3_configure = 11'b00000001111;

#40 p3_configure = 11'b00000001101;
#1 p1_configure = 11'b00000001010;
#10 p3_configure = 11'b00000001101;

#60 p1_configure = 11'b000000001101;
#10 p0_configure = 11'b000000001000;
#10 p1_configure = 11'b000000001101;

#50 block_all_paths=1'b1;
#50 block_all_paths=1'b0;

#1 reset = 1'b1;
#16 reset = 1'b0;

#200;

#60 p1_configure = 11'b000000001101;
#1 p3_configure = 11'b000000001101;
#10 p1_configure = 11'b000000001101;

#50 block_all_paths=1'b1;
#50 block_all_paths=1'b0;

#200;

#10 p0_configure = 11'b00000001111;
#10 p1_configure = 11'b00000001011;
#10 p0_configure = 11'b00000001111;
#10 p1_configure = 11'b00000000110;

#20 p2_configure = 11'b00000001111;
#10 p0_configure = 11'b00000001111;
#10 p2_configure = 11'b00000001111;

#40 p3_configure = 11'b00000001111;
p0_configure = 11'b00000001111;
#10 p3_configure = 11'b00000001111;

```

```
#60 p1_configure = 11'b00000001111;  
  
#40 reset = 1'b1;  
#16 reset = 1'b0;  
  
#50 block_all_paths=1'b1;  
#50 block_all_paths=1'b0;  
  
#200;  
  
#20 p0_configure = 11'b00000010001;  
#1 p1_configure = 11'b00000010011;  
#1 p2_configure = 11'b00000010101;  
#1 p3_configure = 11'b00000010111;  
  
#50 reset = 1'b1;  
#16 reset = 1'b0;  
  
#50 block_all_paths=1'b1;  
#50 block_all_paths=1'b0;  
  
#40 reset = 1'b1;  
#16 reset = 1'b0;  
#200;  
  
#2 p1_configure = 11'b00000001001;  
#1 p2_configure = 11'b00000001001;  
#1 p3_configure = 11'b00000001001;  
  
#50 p1_configure = 11'b00000001001;  
#1 p2_configure = 11'b00000001001;  
#1 p3_configure = 11'b00000001001;  
  
#70 reset = 1'b1;  
#16 reset = 1'b0;  
  
#50 block_all_paths=1'b1;
```

```

#50 block_all_paths=1'b0;

#70 reset = 1'b1;
#16 reset = 1'b0;
#200;

#2 p1_configure = 11'b00000001001;

#40 p1_configure = 11'b00000001000;

#2 p1_configure = 11'b00000001001;

#70 reset = 1'b1;
#16 reset = 1'b0;

#50 block_all_paths=1'b1;
#50 block_all_paths=1'b0;

#70 reset = 1'b1;
#16 reset = 1'b0;
#200;

for(i = 0; i < 4; i = i + 1) begin
case(i)
0: begin
    p0_configure = 11'b00000001001;
    p1_configure = 11'b00000001001;
    p2_configure = 11'b00000001001;
    p3_configure = 11'b00000001001;
end
1: begin
    p0_configure = 11'b00000001011;
    p1_configure = 11'b00000001011;
    p2_configure = 11'b00000001011;
    p3_configure = 11'b00000001011;
end
2: begin
    p0_configure = 11'b00000001101;
    p1_configure = 11'b00000001101;

```

```

    p2_configure = 11'b00000001101;
    p3_configure = 11'b00000001101;
  end
  3: begin
    p0_configure = 11'b00000001111;
    p1_configure = 11'b00000001111;
    p2_configure = 11'b00000001111;
    p3_configure = 11'b00000001111;
  end
endcase
#20; // Delay to ensure parallelism in simulation
end

reset = 1'b1;
#16 reset = 1'b0;

#10000 $finish;
end

initial begin

#1 reset = 1'b1;
#16 reset = 1'b0;
#20 p0_configure = 11'b00000001001;
#20 p0_configure = 11'b00000010001;
#1 p0_configure = 11'b00000100001;
#1 p0_configure = 11'b00000001011;
#1 p0_configure = 11'b00000101011;
#1 p0_configure = 11'b00000111011;

#20 p1_configure = 11'b00000010001;
#20 p1_configure = 11'b00000100001;
#20 p1_configure = 11'b00000001101;
#20 p0_configure = 11'b00000001011;
#20 p0_configure = 11'b00000010011;

#1 p3_configure = 11'b00000100101;
#20 p3_configure = 11'b00000100001;
#20 p0_configure = 11'b00000100011;

```

```

#20 p0_configure = 11'b00000001101;
#20 p0_configure = 11'b00000010101;

#20 p2_configure = 11'b00000010001;
#20 p2_configure = 11'b00000100001;
#20 p2_configure = 11'b00000001101; #20 p2_configure =
11'b00000010001;
#20 p2_configure = 11'b00000100001;

#20 p1_configure = 11'b00000010001;
#20 p1_configure = 11'b00000100001;
#20 p1_configure = 11'b00000001101;
#20 p2_configure = 11'b00000001101;
#20 p0_configure = 11'b00000100101;
#20 p0_configure = 11'b00000100111;
#20 p0_configure = 11'b00000001111;
#20 p0_configure = 11'b00000010111;

//#20 p1_configure = 11'b00000001011;
//#20 p1_configure = 11'b00000001001;
#20 p1_configure = 11'b00000001011;
#20 p1_configure = 11'b00000010011;
#20 p1_configure = 11'b00000100011;

#20 p2_configure = 11'b00000001101;
#20 p2_configure = 11'b00000010101;
#20 p2_configure = 11'b00000100101;
#20 p1_configure = 11'b00000001001;

#20 p2_configure = 11'b00000010001;
#20 p2_configure = 11'b00000100001;
#20 p2_configure = 11'b00000001101;

#1 reset = 1'b1;
#16 reset = 1'b0;

#20 p1_configure = 11'b00000010001;
#20 p1_configure = 11'b00000100001;

```

```

#20 p1_configure = 11'b00000001101;
#20 p1_configure = 11'b00000010101;

#20 p0_configure = 11'b00000100011;
#20 p0_configure = 11'b00000001101;
#20 p0_configure = 11'b00000010101;
#20 p1_configure = 11'b00000100101;
#20 p1_configure = 11'b00000100111;

#20 p3_configure = 11'b00000100101;
#20 p3_configure = 11'b00000100001;
#20 p1_configure = 11'b00000001111;
#20 p1_configure = 11'b00000010111;

//#20 p2_configure = 11'b00000001011;
//#20 p2_configure = 11'b00000001001;
#20 p2_configure = 11'b00000001011;
#20 p2_configure = 11'b000000011011;
#20 p2_configure = 11'b00000100011;
#20 p2_configure = 11'b00000001001;

#20 p3_configure = 11'b00000001011;
#20 p3_configure = 11'b00000010011;
#20 p3_configure = 11'b00000100011;
#20 p3_configure = 11'b00000001101;

#20 p0_configure = 11'b00000100011;
#20 p0_configure = 11'b00000001101;
#20 p0_configure = 11'b00000010101;
#20 p2_configure = 11'b00000010001;

#20 p1_configure = 11'b00000010001;
#20 p1_configure = 11'b000000100001;
#20 p1_configure = 11'b00000001101;
#20 p2_configure = 11'b000000100001;
#20 p2_configure = 11'b00000001101;
#20 p2_configure = 11'b00000010101;
#20 p2_configure = 11'b00000010101;
#20 p2_configure = 11'b00100100111;

```

```

#20  p2_configure = 11'b00000001111;

#1 reset = 1'b1;
#16 reset = 1'b0;

#20      p2_configure = 11'b00110010111;

#70 reset = 1'b1;
#16 reset = 1'b0;

#50 block_all_paths=1'b1;
#50 block_all_paths=1'b0;

#70 reset = 1'b1;
#16 reset = 1'b0;
#200;

//#20 p3_configure = 11'b00000001011;
//#20 p3_configure = 11'b00000001001;
#20 p3_configure = 11'b00000001011;
#20 p3_configure = 11'b000000010011;
#20 p3_configure = 11'b000000100011;
#20 p3_configure = 11'b00000001101;
#20 p3_configure = 11'b000000010101;

#20 p0_configure = 11'b00000100011;
#20 p0_configure = 11'b00000001101;
#20 p0_configure = 11'b000000010101;
#20 p3_configure = 11'b00000100101;
#20 p3_configure = 11'b00000100001;

#70 reset = 1'b1;
#16 reset = 1'b0;

#50 block_all_paths=1'b1;
#50 block_all_paths=1'b0;

#70 reset = 1'b1;
#16 reset = 1'b0;
#200;

```

```

#20 p2_configure = 11'b00000001101;
#20 p2_configure = 11'b00000010101;
#20 p2_configure = 11'b00000100101;

#20 p1_configure = 11'b00000010001;
#20 p1_configure = 11'b00001100001;
#20 p1_configure = 11'b00001001101;
#20 p3_configure = 11'b00100001001;
#20 p3_configure = 11'b00000010001;
#20 p3_configure = 11'b000000100111;
#20 p3_configure = 11'b00000001111;
#20 p3_configure = 11'b000000010111;

#1 reset = 1'b1;
#16 reset = 1'b0;

#20 p0_configure = 11'b00000001011;
#20 p0_configure = 11'b000000011011;
#20 p0_configure = 11'b000000001011;
#20 p0_configure = 11'b000000101011;
#20 p0_configure = 11'b000000111011;

# 10000 $finish;
end
always #10 clk = ~clk;

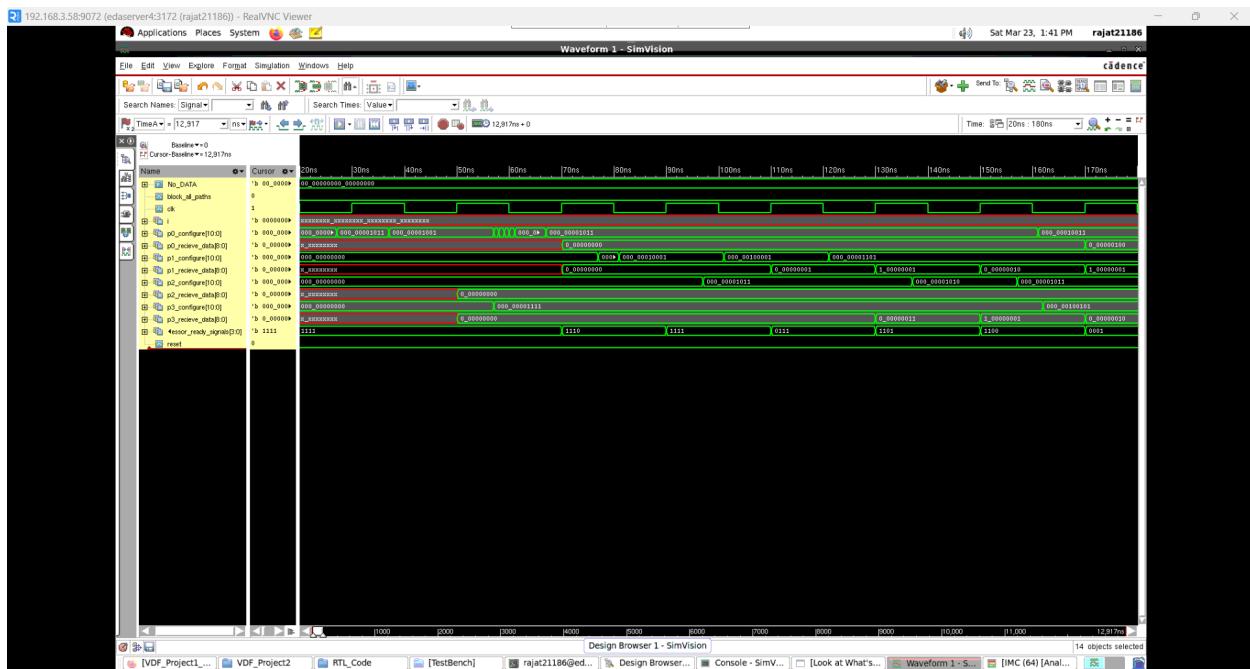
endmodule

```

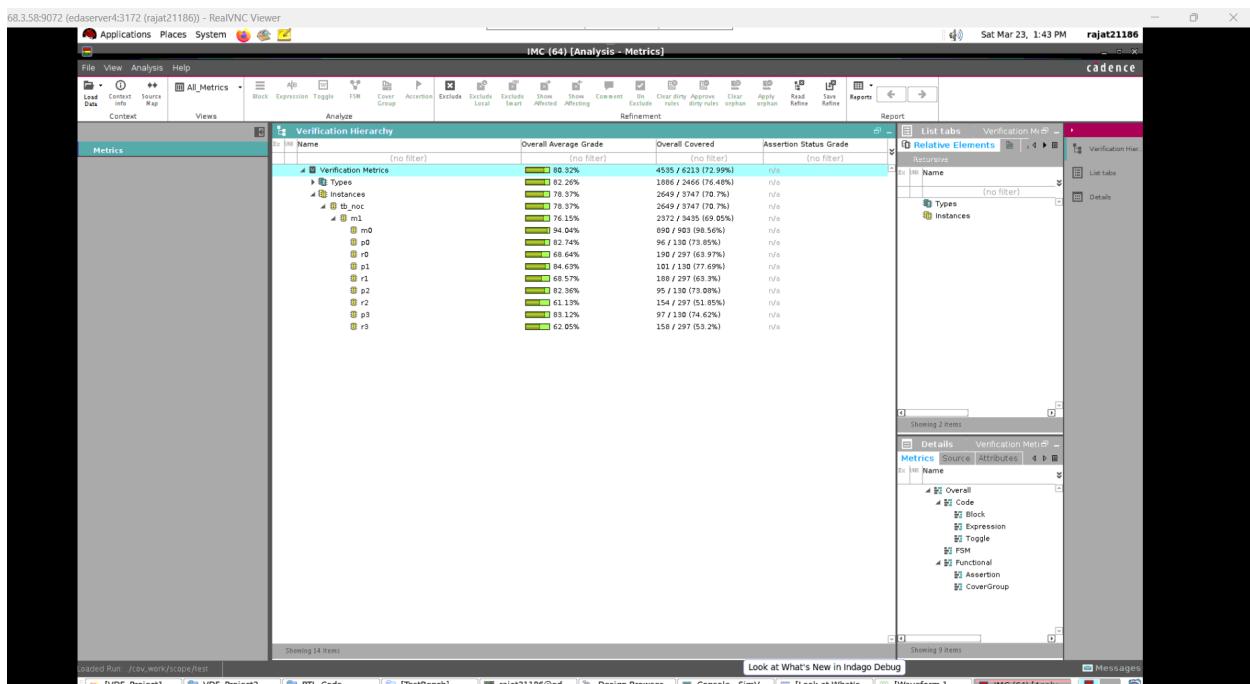
Explanation of test vectors:

Here we have given Clock, reset and processor configuration signals to test the functionality of our code. This testbench is meant for top module coverage. We have configured our Test vectors in such a way that we can firstly test our acknowledgement protocol and the overall working of the master module. Multiple Test Vectors in this case are tested to ensure structural completeness and the functional correctness of the overall code. This is to test coverage as well as the functionality of the testbench.

Output Waveform for Test Bench 3: (Showing the input and output variables)



Code Coverage Report for Test Bench 3:



Toggle Coverage:

The screenshot shows the Cadence IMC 64 Analysis interface with the 'Toggle' view selected. The top status bar indicates the date as Sat Mar 23, 1:44 PM and the session ID as rajat21186. The main window displays coverage data for various signals and variables. The 'Variables' section lists numerous signals like r0_input, r1_input, r2_input, r3_input, p0_receive_data, p1_receive_data, p2_receive_data, p3_receive_data, p0_configure, p1_configure, p2_configure, p3_configure, block_all_paths, processor_ready_signals1, and processor_ready_signals2, each with its range, overall average grade, and overall covered percentage. The 'Signals' section shows the clock signal with a score of 1 and a coverage of 1.0. The right side of the interface features a detailed 'Source' pane showing the Verilog code for the main module, and an 'Attributes' pane listing various coverage metrics such as Block Active (true), Block Average Grade (100%), and Block Total Weights (1.0).

Block Coverage:

The screenshot shows the Cadence IMC 64 Analysis interface with the 'Block' view selected. The top status bar indicates the date as Sat Mar 23, 1:45 PM and the session ID as rajat21186. The main window displays coverage data for code blocks. The 'Blocks' section lists 8 code blocks numbered 1 to 8, each with its index, block type (code block), source line, and score. The right side of the interface features a detailed 'Source' pane showing the Verilog code for the main module, and an 'Attributes' pane listing various coverage metrics such as Block Active (true), Block Average Grade (100%), and Block Total Weights (1.0).

Explanation of coverage:

We see a 100% block coverage by this testbench. We see around 80% overall code coverage by this testbench. Here Master coverage has increased from 60% to 95 % from testbench 1. Here Mesh coverage has increased from 70% to 77 % from testbench 1. We have covered all possible cases of transfers including single transfers, parallel transfers and blocked transfers. We have also tested back-to-back transfers for all the processors. We have also tested the functionality of reset and block all path signals. In fact our path usage circuitry and Input-Output flip flops of the mesh has been also tested by the testbench. Complete Structural and functional testing has been carried out by the testbench. We verified the Result by manual speculation of the waveform. All possible path testing Has been carried out in this testbench.

Everything here seems to be good except the toggling report. The tool reports around 50% coverage of toggles. Fortunately this report does not necessarily mean that our test vectors are incapable of covering the bench. As discussed at the start we have a 11 bit configuration signal for the processor configuration, where the starting 8 bits denote the burst size. As a manual VLSI Engineer It is not possible to give test vectors which cover all toggles of these 8 bits. Also we have certain other inputs like router inputs denoting 9 bit data input from external ports to the specified router (a 18 bit signal per router). Similarly it is not logical as well as not possible for us to cover all toggles for this signal as this is a data packet signal. Similarly we observe various other signals in the design for which either it is not manually possible to cover all toggles or simply it is impractical to do so. Also, another main reason for less coverage is because bits of some specific registers have not toggled in this testbench. Interestingly this is mainly due to logical assumptions under which we have designed this system. For example, we have a counter register in our P_unit module, since this is a counter circuitry, it would toggle from 7'b0 to {{7'b1}} always in increasing manner, but this won't account for all possible toggles. This is functionally correct and thus cannot have toggle coverage. Similarly in our design various toggles are not possible to account for, as they are not practically possible as a result we don't see a good toggle coverage.

3. Logic Synthesis

Inputs:

RTL, Library file, Constraints file

Output:

Netlist (in .v format)

After receiving the RTL, Library, and Constraints files, the tool generates a Netlist in the .v format. The tool processes these inputs to convert the RTL into a structural implementation using cells specified in the library file.

Library File:

Fast.lib

It encompasses crucial data about standard cells, including drive strength, input and output pins, cell delay, rise and fall transitions, power characteristics, and operational conditions(PVT). The file is formatted in Liberty format.

Constraints:

These are design-specific rules that the Cadence Tool must adhere to. They can also be utilized by the Cadence tool to enhance the Power, Performance, and the Area (PPA) of the design.

Netlist:

It represents the interconnection of logic gates using Verilog constructs.

Software Used:

Cadence Genus Synthesis Solution

Commands Used:

1. csh
2. source /cadence/cshrc
3. genus -legacy_ui/Synthesis_script
4. source (name of tcl file)

Synthesis for Minimum Area

TCL file:

```
main_area.tcl X
set_attr library /home/rajat21186/Desktop/VDF_Project/library/fast.lib
read_hdl /home/rajat21186/Desktop/VDF_Project/RTL_Code/main.v
elaborate mesh
read_sdc /home/rajat21186/Desktop/VDF_Project/Constraints/main_constraints_area.sdc
set_attribute information_level 2
synthesize -to_generic
synthesize -to_mapped -effort medium
write_sdf -timescale ns -nonegchecks -recrem split -edges check_edge > ../design/main_min_area_net
write_hdl -lec > ../design/main_min_area_netlist/main_syn.v
report area
report timing
write_sdc > ../design/main_min_area_netlist/main_for_physical_design.sdc
write_script > ../design/main_min_area_netlist/main_sdc.g
report timing > ../Synthesis_Reports/Synthesis_min_area_report/main_synthesis_timing_report.rep
report power > ../Synthesis_Reports/Synthesis_min_area_report/main_synthesis_power_report.rep
report gates > ../Synthesis_Reports/Synthesis_min_area_report/main_synthesis_cell_report.rep
report area > ../Synthesis_Reports/Synthesis_min_area_report/main_synthesis_area_report.rep
check_design
gui_show
```

Create_clock command is used to define the clock present in the circuit in which we define name of the clock, clock period, its duty cycle and name of the port used for clock. Set_clock_transition command is used to specify the transition time (slew) at the source of the clock. Set_clock_uncertainty command is to define the timing uncertainty of a clock period. This uncertainty of clock is used to model various factors that can reduce the effective clock period during hold and setup checks. Set_input_delay command defines the delay at all input ports and reset and set_output_delay defines delay at all output ports while set_load is used to specify capacitive load at all output ports.

Constraints provided for the minimum area

```
create_clock -name clk -period 4.32 [get_ports clock]
set_clock_transition -rise 0.001 [get_clocks clk]
set_clock_transition -fall 0.001 [get_clocks clk]
set_clock_uncertainty 0.01 [get_clocks clk]
set_clock_latency 0.01 [get_clocks clk]

set_input_delay -clock [get_clocks clk] 0.04 [all_inputs]
set_output_delay -clock [get_clocks clk] 0.04 [all_outputs]

set_load 0.5 [all_outputs]
```

Area Report

```
Generated by:          Genus(TM) Synthesis Solution 19.13-s073_1
Generated on:         Mar 23 2024  01:55:52 am
Module:               mesh
Technology library:   fast
Operating conditions: fast (balanced_tree)
Wireload mode:        enclosed
Area mode:            timing library
=====
```

Instance	Module	Cell Count	Cell Area	Net Area	Total Area	Wireload
mesh		1641	14361.420	0.000	14361.420	<none> (D)
m0	master	436	2615.846	0.000	2615.846	<none> (D)
r2	router_2	193	1476.712	0.000	1476.712	<none> (D)
r0	router	177	1397.994	0.000	1397.994	<none> (D)
r3	router_3	176	1375.287	0.000	1375.287	<none> (D)
r1	router_1	151	1245.100	0.000	1245.100	<none> (D)

(D) = wireload is default in technology library

Timing Report

Pin	Type	Fanout	Load	Slew	Delay	Arrival
		(ff)	(ps)	(ps)	(ps)	
(clock clk)	launch latency		0 R			
p0_Configure1_reg[0]/CK		1	+10	10 R		
p0_Configure1_reg[0]/Q	DFFQX1	1 1.8 12	+76	10 R		
g2140/B			+0	86		
g2140/Y	NAND2XL	2 4.7 29	+25	119 F		
g2127/B			+0	118		
g2127/Y	NOR2XL	13 35.5 314	+230	348 R		
m0/P0_signals[0]						
g14919/A			+0	340		
g14919/Y	CLKINVX1	10 20.0 114	+45	385		
g20263/A			+0	385		
g20263/Y	NOR2XL	3 8.6 88	+88	473 R		
g20233/B			+0	473		
g20233/Y	NAND2BX1	1 2.9 32	+17	491 F		
g20227/B0			+0	491		
g20227/Y	OAI2BB1X1	2 5.9 26	+27	518 R		
g20196/C0			+0	518		
g20196/Y	AOI211X1	1 2.9 31	+12	530 F		
g20191/A			+0	530		
g20191/Y	INVX1	2 4.6 20	+22	552 R		
g20185/AN			+0	552		
g20185/Y	NOR2BX1	1 2.9 26	+35	587 R		
g20183/B0			+0	587		
g20183/Y	AOI211X1	7 15.8 62	+31	610 F		
g20182/A			+0	618		
g20182/Y	CLKINVX1	6 15.7 52	+53	671 R		
g20178/B			+0	671		
g20178/Y	NOR2XL	5 13.3 62	+44	715 F		
g20179/B			+0	715		
g20179/Y	NAND3BXL	1 3.0 34	+36	751 R		
g20166/B			+0	751		
g20166/Y	NAND2BX1	2 4.7 24	+29	771 F		
g20162/A1			+0	771		
g20162/Y	AO22XL	1 2.9 19	+49	820 F		
g20157/B			+0	820		
g20157/Y	NOR4X1	11 30.0 280	+212	1032 R		

g20182/A				+0	618
g20182/Y	CLKINVX1	6 15.7	52	+53	671 R
g20178/B				+0	671
g20178/Y	NOR2XL	5 13.3	62	+44	715 F
g20170/B				+0	715
g20170/Y	NAND3BXL	1 3.0	34	+36	751 R
g20166/B				+0	751
g20166/Y	NAND2BX1	2 4.7	24	+20	771 F
g20162/A1				+0	771
g20162/Y	A022XL	1 2.9	19	+49	820 F
g20157/B				+0	820
g20157/Y	NOR4X1	11 30.0	280	+212	1032 R
g20156/A				+0	1032
g20156/Y	CLKINVX1	5 12.4	89	+25	1057 F
g20148/A1				+0	1057
g20148/Y	OAI21X1	2 4.8	48	+53	1110 R
g20137/A1				+0	1110
g20137/Y	AOI22X1	1 3.0	31	+25	1135 F
g20134/C0				+0	1135
g20134/Y	OAI211X1	1 2.9	46	+21	1156 R
g20133/A				+0	1156
g20133/Y	CLKINVX1	1 2.9	17	+10	1166 F
g20132/A1				+0	1166
g20132/Y	OAI22X1	1 3.0	48	+31	1197 R
g20131/C0				+0	1197
g20131/Y	AOI211X1	6 12.7	54	+29	1227 F
g20130/A				+0	1227
g20130/Y	CLKINVX1	8 16.7	53	+52	1279 R
g20375/A				+0	1279
g20375/Y	NAND3X1	6 12.8	62	+50	1329 F
g14831/B				+0	1329
g14831/Y	OR2X1	3 7.8	24	+60	1390 F
g14811/B				+0	1390
g14811/Y	OR2XL	1 1.7	13	+34	1424 F
g14778/B1				+0	1424
g14778/Y	AOI222XL	1 1.7	62	+56	1480 R
g14754/A				+0	1480
g14754/Y	NAND2XL	1 2.5	27	+20	1500 F
R3_control_signals2_reg[16]/D <<<	DFFRHQX1			+0	1500
R3_control_signals2_reg[16]/CK	setup			1 +89	1588 R
(clock clk)					
	capture				4320 R
	latency			+10	4330 R
	uncertainty			-10	4320 R

Cost Group : 'clk' (path_group 'clk')
Timing slack : 2732ps
Start-point : p0_configure1_reg[0]/CK
End-point : m0/R3_control_signals2_reg[16]/D

Power Report

Instance: /mesh

Power Unit: W

PDB Frames: /stim#0/frame#0

Category	Leakage	Internal	Switching	Total	Row%
memory	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
register	1.11203e-04	2.75150e-03	1.13273e-03	3.99543e-03	84.92%
latch	8.64628e-07	1.43892e-06	9.60990e-08	2.39965e-06	0.05%
logic	4.33201e-05	2.56076e-04	1.44229e-04	4.43625e-04	9.43%
bbox	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
clock	0.00000e+00	0.00000e+00	2.63483e-04	2.63483e-04	5.60%
pad	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
pm	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
Subtotal	1.55388e-04	3.00901e-03	1.54054e-03	4.70494e-03	100.00%
Percentage	3.30%	63.95%	32.74%	100.00%	100.00%

Cell Report

NAND4BBXL	2	16.652	fast
NAND4BXL	4	27.248	fast
NAND4XL	1	5.298	fast
NOR2BX1	25	113.535	fast
NOR2BXL	44	199.822	fast
NOR2XL	135	408.726	fast
NOR3BX1	14	84.773	fast
NOR3X1	7	31.790	fast
NOR3XL	6	27.248	fast
NOR4BXL	1	6.812	fast
NOR4X1	1	6.055	fast
NOR4XL	1	6.055	fast
OA21X1	2	13.624	fast
OA22X1	3	22.707	fast
OAI211X1	20	105.966	fast
OAI211XL	46	243.722	fast
OAI21X1	35	158.949	fast
OAI21XL	22	99.911	fast
OAI221X1	1	7.569	fast
OAI222XL	3	24.978	fast
OAI22X1	26	157.435	fast
OAI22XL	2	12.110	fast
OAI2BB1X1	6	31.790	fast
OAI2BB1XL	10	52.983	fast
OAI31X1	2	12.110	fast
OAI32X1	3	20.436	fast
OAI32XL	9	61.309	fast
OAI33X1	5	41.630	fast
OR2X1	7	31.790	fast
OR2XL	3	13.624	fast
OR3XL	2	12.110	fast
OR4XL	5	34.060	fast
SDFFSHQX1	5	136.242	fast
TLATNX1	4	57.524	fast
XNOR2X1	1	8.326	fast
XOR2XL	4	33.304	fast
<hr/>			
total	1641	14361.420	

Type	Instances	Area	Area %
sequential	527	9452.167	65.8
inverter	152	345.146	2.4
logic	962	4564.107	31.8
physical_cells	0	0.000	0.0
<hr/>			
total	1641	14361.420	100.0

Synthesis for Best Timing

TCL file:

```
set_attr library /home/rajat21186/Desktop/VDF_Project/library/fast.lib
read_hdl /home/rajat21186/Desktop/VDF_Project/RTL_Code/main.v
elaborate mesh
read_sdc ./home/rajat21186/Desktop/VDF_Project/Constraints/main_constraints_timing.sdc
set_attribute information_level 2
synthesize -to_generic
synthesize -to_mapped -effort medium
write_sdf -timescale ns -nonegchecks -recrem split -edges check_edge > ../design/main_timing_netlist/main_delays.sdf
write_hdl -lec > ../design/main_timing_netlist/main_syn.v
report area
report timing
write_sdc > ../design/main_timing_netlist/main_for_physical_design.sdc
write_script > ../design/main_timing_netlist/main_sdc.g
report timing > ./Synthesis_Reports/Synthesis_min_timing_report/main_synthesis_timing_report.rep
report power > ./Synthesis_Reports/Synthesis_min_timing_report/main_synthesis_power_report.rep
report gates > ./Synthesis_Reports/Synthesis_min_timing_report/main_synthesis_cell_report.rep
report area > ./Synthesis_Reports/Synthesis_min_timing_report/main_synthesis_area_report.rep
check_design
gui_show
```

Constraints provided for just negative slack

```
create_clock -name clk -period 0.78 [get_ports clock]
set_clock_transition -rise 0.001 [get_clocks clk]
set_clock_transition -fall 0.001 [get_clocks clk]
set_clock_uncertainty 0.01 [get_clocks clk]
set_clock_latency 0.01 [get_clocks clk]

set_input_delay -clock [get_clocks clk] 0.04 [all_inputs]
set_output_delay -clock [get_clocks clk] 0.04 [all_outputs]

set_load 0.5 [all_outputs]
```

Area Report

```
=====
Generated by:      Genus(TM) Synthesis Solution 19.13-s073_1
Generated on:      Mar 23 2024  02:03:26 am
Module:           mesh
Technology library: fast
Operating conditions: fast (balanced_tree)
Wireload mode:    enclosed
Area mode:        timing library
=====

Instance  Module   Cell Count  Cell Area  Net Area  Total Area  Wireload
-----
mesh          1779  14857.190  0.000  14857.190  <none> (D)
m0            master     564  3254.670  0.000  3254.670  <none> (D)
r2            router_2    183  1477.469  0.000  1477.469  <none> (D)
r0            router      176  1427.513  0.000  1427.513  <none> (D)
r3            router_3    176  1394.967  0.000  1394.967  <none> (D)
r1            router_1     150  1272.349  0.000  1272.349  <none> (D)

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

Timing Report

```
=====
Generated by: Genus(TM) Synthesis Solution 19.13-s073_1
Generated on: Mar 23 2024 02:03:25 am
Module: mesh
Technology library: fast
Operating conditions: fast (balanced_tree)
Wireload mode: enclosed
Area mode: timing library
=====
```

Pin	Type	Fanout	Load	Slew	Delay	Arrival
		(fF)	(ps)	(ps)	(ps)	
(clock clk)	launch latency				0 R	
r0				+10	10 R	
regPR_reg/CK				1	10 R	
regPR_reg/Q	SDFFSHQX4	5	13.7	16	+109	119 F
r0/processor_ready						
g937/B				+0	119	
g937/Y	NAND2X1	2	8.9	31	+27	146 R
g909/A				+0	146	
g909/Y	NOR2X2	5	13.4	21	+19	164 F
m0/path_free_bits[9]						
g15646/A0				+0	164	
g15646/Y	AOI22XL	1	2.8	43	+45	209 R
g15640/A				+0	209	
g15640/Y	NAND2X1	2	7.3	34	+26	235 F
g15599/A2				+0	235	
g15599/Y	AOI31X2	2	8.7	36	+36	271 R
g15589/C				+0	271	
g15589/Y	NAND3BX2	1	11.6	36	+30	301 F
g15587/B				+0	301	
g15587/Y	NOR2X4	6	18.6	32	+33	333 R
g15578/A0N				+0	333	
g15578/Y	OAI2BB1X1	2	4.6	21	+41	374 R
g15554/A1				+0	374	
g15554/Y	AOI21X1	1	4.3	24	+21	395 F
g15550/A				+0	395	
g15550/Y	CLKAND2X3	1	18.4	20	+47	442 F
g15547/A				+0	442	
g15547/Y	NAND3X6	11	30.7	25	+25	468 R
g15531/A1				+0	468	
g15531/Y	OAI21X1	2	4.7	29	+20	488 F
g15515/B				+0	488	
g15515/Y	NAND2BX1	1	2.9	17	+19	507 R
g15508/B				+0	507	
g15508/Y	NAND4X1	1	5.8	44	+39	546 F
g15505/B				+0	546	
g15505/Y	NOR2X2	2	8.4	31	+34	579 R

		NUM2X2	0	15.4	<1	+19	104	R
g989/Y	m0/path_free_bits[9]							
g15646/A0					+0	164		
g15646/Y		AOI22XL	1	2.8	43	+45	209	R
g15640/A					+0	209		
g15640/Y		NAND2X1	2	7.3	34	+26	235	F
g15599/A2					+0	235		
g15599/Y		AOI31X2	2	8.7	36	+36	271	R
g15589/C					+0	271		
g15589/Y		NAND3BX2	1	11.6	36	+30	301	F
g15587/B					+0	301		
g15587/Y		NOR2X4	6	18.6	32	+33	333	R
g15578/A0N					+0	333		
g15578/Y		OAI2BB1X1	2	4.6	21	+41	374	R
g15554/A1					+0	374		
g15554/Y		AOI21X1	1	4.3	24	+21	395	F
g15550/A					+0	395		
g15550/Y		CLKAND2X3	1	18.4	20	+47	442	F
g15547/A					+0	442		
g15547/Y		NAND3X6	11	30.7	25	+25	468	R
g15531/A1					+0	468		
g15531/Y		OAI21X1	2	4.7	29	+20	488	F
g15515/B					+0	488		
g15515/Y		NAND2BX1	1	2.9	17	+19	507	R
g15508/B					+0	507		
g15508/Y		NAND4X1	1	5.8	44	+39	546	F
g15505/B					+0	546		
g15505/Y		NOR2X2	2	8.4	31	+34	579	R
g15504_5/B					+0	579		
g15504_5/Y		CLKAND2X3	6	15.1	19	+34	613	R
g11683/B					+0	613		
g11683/Y		NAND2X2	3	11.4	27	+20	633	F
fopt15992/A					+0	633		
fopt15992/Y		CLKINVX1	2	8.5	28	+28	661	R
fopt15991/A					+0	661		
fopt15991/Y		INVX2	4	7.0	14	+10	672	F
g11585/A					+0	672		
g11585/Y		NAND3XL	1	1.8	19	+20	691	R
R3_control_signals2_reg[14]/D <<<		DFFRX1			+0	691		
R3_control_signals2_reg[14]/CK		setup			1	+94	785	R
<hr/>								
(clock clk)		capture					780	R
		latency				+10	790	R
		uncertainty				-10	780	R

Cost Group : 'clk' (path_group 'clk')
 Timing slack : -5ps (TIMING VIOLATION)
 Start-point : r0/regPR_reg/CK
 End-point : m0/R3_control_signals2_reg[14]/D

Power Report

Instance: /mesh

Power Unit: W

PDB Frames: /stim#0/frame#0

Category	Leakage	Internal	Switching	Total	Row%
memory	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
register	1.03454e-04	1.44834e-02	6.25682e-03	2.08437e-02	86.26%
latch	8.64628e-07	6.23454e-06	6.82784e-07	7.78195e-06	0.03%
logic	5.06980e-05	1.16285e-03	6.83566e-04	1.89711e-03	7.85%
bbox	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
clock	0.00000e+00	0.00000e+00	1.41508e-03	1.41508e-03	5.86%
pad	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
pm	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
Subtotal	1.55017e-04	1.56525e-02	8.35615e-03	2.41636e-02	100.00%
Percentage	0.64%	64.78%	34.58%	100.00%	100.00%

Cell Report

NOR3BX1	10	60.552	fast
NOR3BXL	4	24.221	fast
NOR3X1	8	36.331	fast
NOR3X2	1	9.083	fast
NOR3XL	6	27.248	fast
NOR4BXL	1	6.812	fast
NOR4X1	1	6.055	fast
NOR4XL	3	18.166	fast
OA21XL	2	13.624	fast
OAI211X1	5	26.492	fast
OAI211XL	44	233.125	fast
OAI21X1	46	208.904	fast
OAI21X2	1	8.326	fast
OAI21XL	41	186.197	fast
OAI221X1	13	98.397	fast
OAI221XL	2	15.138	fast
OAI222XL	1	8.326	fast
OAI22X1	11	66.607	fast
OAI22XL	8	48.442	fast
OAI2BB1X1	7	37.088	fast
OAI2BB1XL	5	26.492	fast
OAI31X1	2	12.110	fast
OAI31XL	3	18.166	fast
OAI32X1	3	20.436	fast
OAI32XL	2	13.624	fast
OAI33X1	1	8.326	fast
OR2X1	4	18.166	fast
OR2XL	7	31.790	fast
SDFFSHQX1	1	27.248	fast
SDFFSHQX2	3	86.287	fast
SDFFSHQX4	9	286.108	fast
TLATNX1	4	57.524	fast
XNOR2XL	1	8.326	fast
XOR2XL	6	49.955	fast
<hr/>			
total	1779	14857.190	

Type	Instances	Area	Area %
sequential	527	9250.832	62.3
inverter	202	489.714	3.3
buffer	27	125.645	0.8
logic	1023	4990.999	33.6
physical_cells	0	0.000	0.0
<hr/>			
total	1779	14857.190	100.0

□ Synthesis for Optimal Constraints

TCL file:

```
set_attr library /home/rajat21186/Desktop/VDF_Project/library/fast.lib
read_hdl /home/rajat21186/Desktop/VDF_Project/RTL_Code/main.v
elaborate mesh
read_sdc /home/rajat21186/Desktop/VDF_Project/Constraints/main_constraints_optimal.sdc
set_attribute information_level 2
synthesize -to_generic
synthesize -to_mapped -effort medium
write_sdf -timescale ns -nonegchecks -recrern split -edges check_edge > ../design/main_optimal_netlist/main_delays.sdf
write_hdl -lec > ../design/main_optimal_netlist/main_syn.v
report area
report timing
write_sdc > ../design/main_optimal_netlist/main_for_physical_design.sdc
write_script > ../design/main_optimal_netlist/main_sdc.g
report timing > ../Synthesis_Reports/Synthesis_optimal_report/main_synthesis_timing_report.rep
report power > ../Synthesis_Reports/Synthesis_optimal_report/main_synthesis_power_report.rep
report gates > ../Synthesis_Reports/Synthesis_optimal_report/main_synthesis_cell_Report.rep
report area > ../Synthesis_Reports/Synthesis_optimal_report/main_synthesis_area_report.rep
check_design
gui_show
```

Constraints provided

```
create_clock -name clk -period 2.55 [get_ports clock]

set_clock_transition -rise 0.001 [get_clocks clk]
set_clock_transition -fall 0.001 [get_clocks clk]
set_clock_uncertainty 0.01 [get_clocks clk]
set_clock_latency 0.01 [get_clocks clk]

set_input_delay -clock [get_clocks clk] 0.04 [all_inputs]
set_output_delay -clock [get_clocks clk] 0.04 [all_outputs]

set_load 0.5 [all_outputs]
```

Area Report

```
=====
Generated by:          Genus(TM) Synthesis Solution 19.13-s073_1
Generated on:          Mar 23 2024  02:05:52 am
Module:                mesh
Technology library:   fast
Operating conditions: fast (balanced_tree)
Wireload mode:        enclosed
Area mode:            timing library
=====
```

Instance	Module	Cell Count	Cell Area	Net Area	Total Area	Wireload
mesh		1649	14384.884	0.000	14384.884	<none> (D)
m0	master	439	2613.576	0.000	2613.576	<none> (D)
r2	router_2	193	1476.712	0.000	1476.712	<none> (D)
r0	router	177	1397.994	0.000	1397.994	<none> (D)
r3	router_3	177	1382.856	0.000	1382.856	<none> (D)
r1	router_1	151	1245.100	0.000	1245.100	<none> (D)

(D) = wireload is default in technology library

Timing Report

```
=====
Generated by: Genus(TM) Synthesis Solution 19.13-s073_1
Generated on: Mar 23 2024 02:05:51 am
Module: mesh
Technology library: fast
Operating conditions: fast (balanced_tree)
Wireload mode: enclosed
Area mode: timing library
=====
```

	Pin	Type	Fanout	Load	Slew	Delay	Arrival
			(fF)	(ps)	(ps)	(ps)	
(clock clk)		launch latency			+10	10	R
p0_configure1_reg[0]/CK				1		10	R
p0_configure1_reg[0]/Q		DFFQX1	1	1.8	12	+76	86 R
g2369/B					+0	86	
g2369/Y		NAND2XL	2	4.7	29	+25	110 F
g2353/B					+0	110	
g2353/Y		NOR2XL	14	33.7	298	+219	329 R
m0/P0_signals[0]							
g14752/A					+0	329	
g14752/Y		CLKINVX1	10	21.2	112	+49	378 F
g17101/B					+0	378	
g17101/Y		NAND2XL	7	18.4	106	+104	481 R
g17034/A					+0	481	
g17034/Y		CLKINVX1	5	12.5	48	+30	511 F
g16923/C0					+0	511	
g16923/Y		AOI221X1	2	4.7	68	+55	566 R
g16918/A					+0	566	
g16918/Y		INVX1	2	4.7	26	+14	580 F
g16898/A0N					+0	580	
g16898/Y		OAI2BB1XL	1	1.7	19	+34	614 F
g16897/A					+0	614	
g16897/Y		NOR2XL	8	20.1	180	+137	751 R
g16896/A					+0	751	
g16896/Y		CLKINVX1	7	18.3	77	+43	793 F
g16893/B					+0	793	
g16893/Y		NOR2XL	5	13.6	128	+110	904 R
g16882/A1					+0	904	
g16882/Y		AOI21X1	2	5.9	54	+30	934 F
g16877/A1					+0	934	
g16877/Y		OAI22X1	1	1.7	38	+37	972 R
g16875/A					+0	972	
g16875/Y		NAND2XL	1	3.0	25	+21	993 F
g16874/C0					+0	993	
g16874/Y		OAI211X1	1	1.7	34	+17	1010 R
g16873/B					+0	1010	
g16873/Y		OR2X1	11	30.3	87	+81	1092 R

g16893/Y	NOR2XL	5	13.6	128	+110	904	R	
g16882/A1					+0	904		
g16882/Y	AOI21X1	2	5.9	54	+30	934	F	
g16877/A1					+0	934		
g16877/Y	OAI22X1	1	1.7	38	+37	972	R	
g16875/A					+0	972		
g16875/Y	NAND2XL	1	3.0	25	+21	993	F	
g16874/C0					+0	993		
g16874/Y	OAI211X1	1	1.7	34	+17	1010	R	
g16873/B					+0	1010		
g16873/Y	OR2X1	11	30.3	87	+81	1092	R	
g16872/A					+0	1092		
g16872/Y	CLKINVX1	6	13.7	45	+32	1124	F	
g16867/B1					+0	1124		
g16867/Y	AOI221X1	2	4.7	62	+57	1181	R	
g16862/A					+0	1181		
g16862/Y	CLKINVX1	1	3.0	21	+9	1190	F	
g16852/A1					+0	1190		
g16852/Y	AOI22X1	1	1.8	28	+30	1220	R	
g16848/C					+0	1220		
g16848/Y	AND3X1	1	3.0	18	+44	1264	R	
g16847/C0					+0	1264		
g16847/Y	OAI211X1	7	16.1	92	+60	1324	F	
g16846/A					+0	1324		
g16846/Y	CLKINVX1	10	18.5	64	+67	1391	R	
g14724/A					+0	1391		
g14724/Y	NAND2XL	5	13.7	129	+64	1455	F	
g14692/B					+0	1455		
g14692/Y	NOR2XL	3	8.7	95	+98	1553	R	
g14682/A					+0	1553		
g14682/Y	CLKINVX1	1	3.0	28	+8	1560	F	
g14662/A2					+0	1560		
g14662/Y	OAI32X1	2	3.5	60	+47	1608	R	
g14588/B0					+0	1608		
g14588/Y	AOI21XL	1	1.6	29	+11	1618	F	
g14570/A					+0	1618		
g14570/Y	NAND3XL	1	2.4	25	+26	1645	R	
R3_control_signals2_reg[10]/D <<<	DFFRHQX1				+0	1645		
R3_control_signals2_reg[10]/CK	setup				1	+91	1736	R
<hr/>								
clock clk)	capture					2550	R	
	latency					+10	2560	R
	uncertainty					-10	2550	R

```
lost Group    : 'clk' (path_group 'clk')
Timing slack :     814ps
Start-point  : p0_configure1_reg[0]/CK
End-point    : m0/R3_control_signals2_reg[10]/D
```

Power Report

Instance: /mesh

Power Unit: W

PDB Frames: /stim#0/frame#0

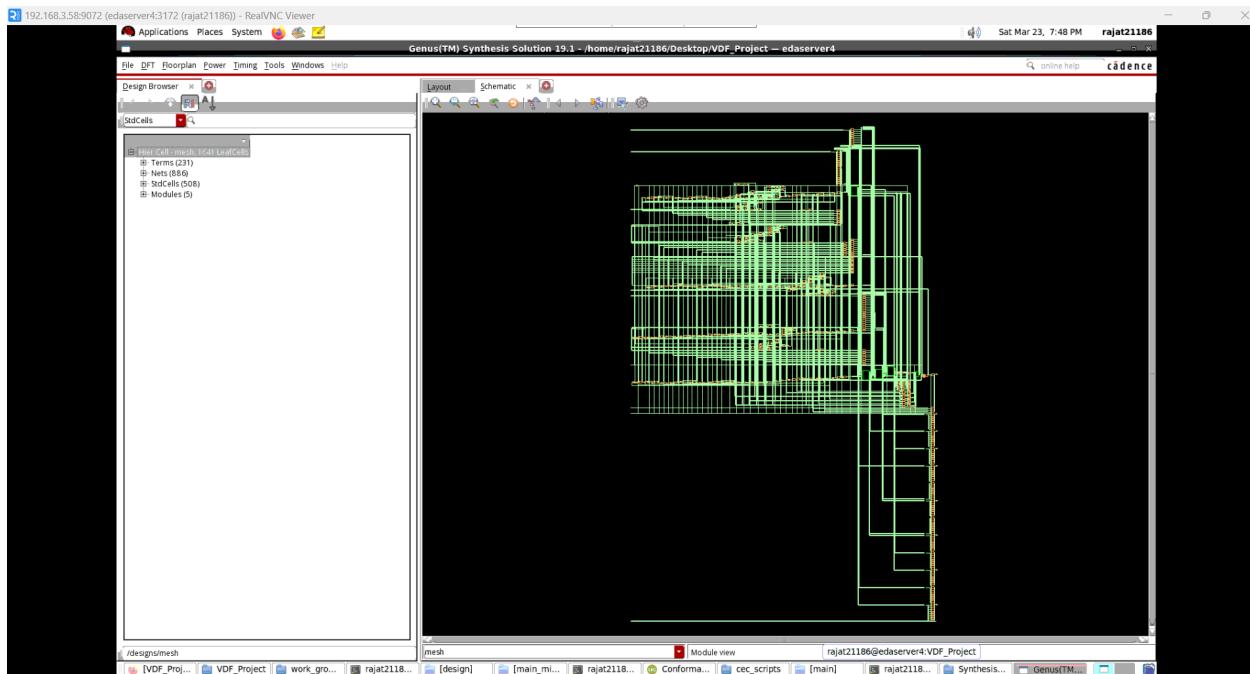
Category	Leakage	Internal	Switching	Total	Row%
memory	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
register	1.11615e-04	4.64139e-03	1.91408e-03	6.66708e-03	85.34%
latch	8.64628e-07	2.20777e-06	1.47988e-07	3.22039e-06	0.04%
logic	4.27235e-05	4.14528e-04	2.38638e-04	6.95890e-04	8.91%
bbox	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
clock	0.00000e+00	0.00000e+00	4.46371e-04	4.46371e-04	5.71%
pad	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
pm	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
Subtotal	1.55203e-04	5.05813e-03	2.59924e-03	7.81257e-03	100.00%
Percentage	1.99%	64.74%	33.27%	100.00%	100.00%

Cell Report

NAND4BBXL	1	8.326	fast
NAND4BXL	1	6.812	fast
NOR2BX1	24	108.994	fast
NOR2BXL	45	204.363	fast
NOR2XL	127	384.505	fast
NOR3BX1	17	102.938	fast
NOR3BXL	2	12.110	fast
NOR3X1	6	27.248	fast
NOR3XL	7	31.790	fast
NOR4X1	1	6.055	fast
NOR4XL	1	6.055	fast
OAI21X1	2	13.624	fast
OAI22XL	1	7.569	fast
OAI211X1	23	121.861	fast
OAI211XL	48	254.318	fast
OAI21X1	35	158.949	fast
OAI21XL	16	72.662	fast
OAI221X1	2	15.138	fast
OAI222XL	2	16.652	fast
OAI22X1	27	163.490	fast
OAI22XL	1	6.055	fast
OAI2BB1X1	4	21.193	fast
OAI2BB1XL	12	63.580	fast
OAI31X1	3	18.166	fast
OAI31XL	2	12.110	fast
OAI32X1	5	34.060	fast
OAI32XL	8	54.497	fast
OAI33X1	7	58.281	fast
OR2X1	3	13.624	fast
OR2XL	3	13.624	fast
OR4XL	4	27.248	fast
SDFFSHQX1	5	136.242	fast
TLATNX1	4	57.524	fast
XNOR2X1	2	16.652	fast
XNOR2XL	1	8.326	fast
XOR2XL	3	24.978	fast
<hr/>			
total	1649	14384.884	

Type	Instances	Area	Area %
sequential	527	9464.277	65.8
inverter	163	370.124	2.6
logic	959	4550.483	31.6
physical_cells	0	0.000	0.0
<hr/>			
total	1649	14384.884	100.0

Synthesized Schematic



QoR Comparison

Analysis of min area constraints

1. By changing the value of the time period and increasing it initially, it was observed that at the clock period was saturated, that is no more reduction in the value of the area was observed by increasing the value of the time period beyond 4.31ns.
2. In the area report there 1641 cells and the cell area of 14361.420.
3. The start point and end point and slack shown in the timing report, obtained from the netlist. The start point is p0_configure1_reg[0]/CK and the end point is taken at m0/R3_control_signals2_reg[16]/D
4. The arrival time computation starts from the start point and all the delays in the path are added. The final arrival time (AT) is computed by the tool by adding the setup margin of the capture flip flop used in the design and it is observed to be 50ps , hence the net arrival time computed by the tool is $1500 + 89 = 1589$ but tool is showing 1588 due to uncertainty of clock. The required time is decided by the clock period as the data is captured by the capture flop at the next clock edge , so here the clock period is 4320ps, but the uncertainty is 10ps, and the latency is 10ps, so the net required time is => $4320 - 10 + 10$ ps => 4320ps.

5. The slack value is calculated using Required time - Arrival time.
6. When synthesizing for minimum area of our design the tool will choose the smallest standard cells from the library.
7. The smallest cells means that the current through the cells is low hence the delay will be higher.
8. Since the delay is higher, so we have to increase the time period of the clock until minimum area is achieved.

Analysis of tight clock constraints

1. For this tight clock case we iterate over and over again finally we finalize our time period as 0.78 ns when we got the minimum negative slack as -5ps.
2. The start point in this case is r0/regPR_reg/CK and the end point is m0/R3_control_signals2_reg[14]/D.
3. So by the same explanation given as above in the case of min area the arrival time in this case is the arrival time is $691 + 94 = 785$ ps and the required time is $780 + 10(\text{latency}) - 10(\text{uncertainty}) = 780$ ps.
4. The slack = RT - AT = $780 - 785 = -5$ ps.
5. When synthesizing for the best timing, the tool will pick the cells from the library with the least delay.
6. The current will be higher for such cells and hence the area.
7. The time period of the clock is chosen as such the slack is just negative.

Analysis of optimal clock constraints

1. For this tight clock case we iterate over and over again finally we finalize our time period as 2.55 ns when we got the minimum negative slack as 814 ps.
2. The start point in this case is p0_configure1_reg[0]/CK and the end point is m0/R3_control_signals2_reg[10]/D.
3. So by the same explanation given as above in the case of min area the arrival time in this case is the arrival time is $1645 + 91 = 1736$ ps and the required time is $2550 + 10(\text{latency}) - 10(\text{uncertainty}) = 2550$ ps.
4. The slack = RT - AT = $2550 - 1736 = 814$ ps.
5. In this case, the constraints are provided between the above two cases, so we take the average of the two time periods in the above cases.

Above we analyzed the effect of changing area on changing the time period and now we have explained how the tool calculates the slack , RT and AT using the given constraints such as time period, uncertainty, latency, input delay, output delay, set load, etc.

So, we have explained the changing impact of constraints on the QoR.

	CLOCK PERIOD	CELL COUNT	AREA	TIMING SLACK
MIN. AREA	4.32ns	1641	14361.420	2732 ps
OPTIMAL	2.55ns	1649	14384.884	814 ps
BEST TIMING	0.78ns	1779	14857.190	-5 ps

RTL and Netlist comparison

RTL code: Input-output flip flops

```

main.v
File Edit View
reg [17:0] r0_output2,r1_output2,r2_output2,r3_output2;
//assign temp_path_block_signals=temp_path_block_signals2;
assign processor_ready_signals=processor_ready_signals2;
assign r0_output=r0_output2;
assign r1_output=r1_output2;
assign r2_output=r2_output2;
assign r3_output=r3_output2;
always@(posedge clock)
begin
    r0_input1 <= r0_input;
    block_all_paths1 <= block_all_paths;
    r1_input1 <= r1_input;
    r2_input1 <= r2_input;
    r3_input1 <= r3_input;
end
always@(posedge clock)
begin
    r0_output2 <= r0_output1;
    r1_output2 <= r1_output1;
    r2_output2 <= r2_output1;
    r3_output2 <= r3_output1;
end
always@(posedge clock)
begin
    processor_ready_signals2 <= processor_ready_signals1;
    // temp_path_block_signals2 <= temp_path_block_signals1;
end
always@(posedge clock)
begin
    p0_configure1 <= p0_configure;
    p1_configure1 <= p1_configure;
    p2_configure1 <= p2_configure;
    p3_configure1 <= p3_configure;
end

```

Ln 1, Col 1 | 18,221 characters

Netlist code: Input-output flip flops

```
File Edit View

_Control_Signals[16], UNCONNECTED_HIER_246, R3_Control_Signals[15])), .select_east ((UNCONNECTED_HIER_246, _Control_Signals[16])), .select_west (R3_Control_Signals[10:8]), .select_processor (R3_Control_Signals[10:8]), .data_east (r3_input1[17:9]), .data_west (d23), .data_processor (d33), .output_north (output1[17:9]), .output_west (d32), .output_processor (r33), .north_ready (Nr3), .south_ready (Pr3), .SetNR (UNCONNECTED_HIER_249), .SetSR (R3_Control_Signals[3]), .SetER (UNCONNECTED_HIER_249), .Control_Signals[8]));
DFQFQXL p0_configure1_reg[0] (.CK (clock), .D (p0_configure[0]), .Q (p0_configure1[0]));
DFQFQXL p0_configure1_reg[1] (.CK (clock), .D (p0_configure[1]), .Q (p0_configure1[1]));
DFQFQXL p0_configure1_reg[2] (.CK (clock), .D (p0_configure[2]), .Q (p0_configure1[2]));
DFQFQXL p0_configure1_reg[3] (.CK (clock), .D (p0_configure[3]), .Q (p0_configure1[3]));
DFQFQXL p0_configure1_reg[4] (.CK (clock), .D (p0_configure[4]), .Q (p0_configure1[4]));
DFQFQXL p0_configure1_reg[5] (.CK (clock), .D (p0_configure[5]), .Q (p0_configure1[5]));
DFQFQXL p0_configure1_reg[6] (.CK (clock), .D (p0_configure[6]), .Q (p0_configure1[6]));
DFQFQXL p0_configure1_reg[7] (.CK (clock), .D (p0_configure[7]), .Q (p0_configure1[7]));
DFQFQXL p0_configure1_reg[8] (.CK (clock), .D (p0_configure[8]), .Q (p0_configure1[8]));
DFQFQXL p0_configure1_reg[9] (.CK (clock), .D (p0_configure[9]), .Q (p0_configure1[9]));
DFQFQXL p0_configure1_reg[10] (.CK (clock), .D (p0_configure[10]), .Q (p0_configure1[10]));
DFFRHQXL p0_tlast_reg(.RN (n_0), .CK (clock), .D (n_197), .Q (p0_tlast));
DFQFQXL p1_configure1_reg[0] (.CK (clock), .D (p1_configure[0]), .Q (p1_configure1[0]));
DFQFQXL p1_configure1_reg[1] (.CK (clock), .D (p1_configure[1]), .Q (p1_configure1[1]));
DFQFQXL p1_configure1_reg[2] (.CK (clock), .D (p1_configure[2]), .Q (p1_configure1[2]));
DFQFQXL p2_configure1_reg[0] (.CK (clock), .D (p2_configure[0]), .Q (p2_configure1[0]));
DFQFQXL p2_configure1_reg[1] (.CK (clock), .D (p2_configure[1]), .Q (p2_configure1[1]));
DFQFQXL p2_configure1_reg[2] (.CK (clock), .D (p2_configure[2]), .Q (p2_configure1[2]));
DFQFQXL p3_configure1_reg[0] (.CK (clock), .D (p3_configure[0]), .Q (p3_configure1[0]));
DFQFQXL p3_configure1_reg[1] (.CK (clock), .D (p3_configure[1]), .Q (p3_configure1[1]));
DFQFQXL p3_configure1_reg[2] (.CK (clock), .D (p3_configure[2]), .Q (p3_configure1[2]));
DFQFQXL p3_configure1_reg[3] (.CK (clock), .D (p3_configure[3]), .Q (p3_configure1[3]));
DFQFQXL p3_configure1_reg[4] (.CK (clock), .D (p3_configure[4]), .Q (p3_configure1[4]));
DFQFQXL p3_configure1_reg[5] (.CK (clock), .D (p3_configure[5]), .Q (p3_configure1[5]));
DFQFQXL p3_configure1_reg[6] (.CK (clock), .D (p3_configure[6]), .Q (p3_configure1[6]));
DFQFQXL p3_configure1_reg[7] (.CK (clock), .D (p3_configure[7]), .Q (p3_configure1[7]));
DFQFQXL p3_configure1_reg[8] (.CK (clock), .D (p3_configure[8]), .Q (p3_configure1[8]));
DFQFQXL p3_configure1_reg[9] (.CK (clock), .D (p3_configure[9]), .Q (p3_configure1[9]));
```

RTL code: master-interlogic circuit and its ffs

```
main.v          main.syn.v          Untitled          Master_new.v X
File Edit View

assign response_signals = response_signals2;
//Preliminary conditions for reset and and every posedge
always(poedge clock or poedge reset)
begin
    if(reset==1'b1) //At reset data does not need to go to anywhere, so all destinadirections of router are set to default
    begin
        R0_control_signals2 <= 20'b00;
        R1_control_signals2 <= 20'b00;
        R2_control_signals2 <= 20'b00;
        R3_control_signals2 <= 20'b00;
        response_signals2 <= 4'b00; //Master is saying free all paths
    end
    else //Assign destination for every direction of a router following the below computation at every clockedge
    begin
        R0_control_signals2 <= R0_control_signals1 ;
        R1_control_signals2 <= R1_control_signals1 ;
        R2_control_signals2 <= R2_control_signals1 ;
        R3_control_signals2 <= R3_control_signals1 ;
        response_signals2 <= response_signals1;
    end
end

// P0 - Processing
reg [19:0] R0_control_signals1_0, R1_control_signals1_0, R2_control_signals1_0, R3_control_signals1_0;
reg response_signals1_0;
always@(*)
begin
    if(P0_signals[0]==1'b1)
    begin
        case(P0_signals[2:1]) //iterating all cases of P0
            2'000://0 to 0
            begin
                if(path_free_bits[0]==1'b1)
                begin
```

```

File Edit View main_syn.v Untitled Master_new.v x + - o ⊖
assign sig01_2 = (reg0_1[4:0] & R0_control_signals1_2[4:0]) | (reg3_1[4:0] & R3_control_signals1_2[4:0]) | (reg1_1[4:0] & R1_control_signals1_2[4:0]);
always@(*)
begin
if(sig01_2==5'b00)
begin
reg8_2 = reg8_1 & R0_control_signals1_2;
reg1_2 = reg1_1 & R1_control_signals1_2;
reg2_2 = reg2_1 | R2_control_signals1_2;
reg3_2 = reg3_1 | R3_control_signals1_2;
reg8_3 = reg8_1[2]>>2&reg8_1[2];
end
else
begin
response_signals1[2] = 1'b0;
reg8_2 = reg8_1;
reg1_2 = reg1_1;
reg2_2 = reg2_1;
reg3_2 = reg3_1;
end
end
wire [4:0] sig02_3;
assign sig02_3 = (reg0_2[4:0] & R0_control_signals1_3[4:0]) | (reg3_2[4:0] & R3_control_signals1_3[4:0]) | (reg1_2[4:0] & R1_control_signals1_3[4:0]);
always@(*)
begin
if(sig02_3==5'b00)
begin
R0_control_signals1 = reg2_3 & R0_control_signals1_3;
R1_control_signals1 = reg1_2 & R1_control_signals1_3;
R2_control_signals1 = reg2_2 & R2_control_signals1_3;
R3_control_signals1 = reg3_2 & R3_control_signals1_3;
response_signals1[3] = response_signals1_3;
end
end

```

Ln 1, Col 1 73,822 characters 100% Unix (LF) UTF-8

Netlist: master-interlogic circuit and its fffs

```

File Edit View main_syn.v x +
DFFRHQX1 \response_signals2_reg[0].(RN(n_15), .CK(clock), .D(n_294), .Q(response_signals[0]));
DFFRHQX1 \response_signals2_reg[1].(RN(n_15), .CK(clock), .D(n_307), .Q(response_signals[1]));
DFFRHQX1 \response_signals2_reg[2].(RN(n_15), .CK(clock), .D(n_328), .Q(response_signals[2]));
DFFRHQX1 \response_signals2_reg[3].(RN(n_15), .CK(clock), .D(n_348), .Q(response_signals[3]));
INVXL g16828.(A(n_360), Y(n_361));
AOI131XL g16830.(A(n_252), A1(n_188), A2(n_4), B0(n_339), Y(n_360));
NAND3BXL g16831.(AN(n_367), B(n_338), C(n_281), Y(n_359));
AOI2B8B1XL g16832.(AN(n_383), A1N(n_4), B0(n_335), Y(n_358));
AOI131XL g16834.(A(n_251), A1(n_189), A2(n_371), B(n_338), Y(n_357));
NAND4BXL g16835.(AN(n_373), B(n_308), C(n_350), D(n_308), Y(n_356));
AOI2B8B1XL g16837.(AN(n_381), A1N(n_4), B0(n_334), Y(n_355));
AOI211XL g16838.(A(n_374), A1(n_287), B0(n_315), C0(n_5), Y(n_354));
OR4XL g16839.(A(n_442), B(n_369), C(n_373), D(n_367), Y(n_353));
AOI211XL g16840.(A(n_371), A1(n_277), B0(n_310), C0(n_329), Y(n_352));
AOI2B8B1XL g16841.(AN(n_292), A1N(n_4), B0(n_332), Y(n_351));
NAND2XL g16842.(A(n_4), B(n_264), Y(n_356));
CLKINVXL g16843.(A(n_5), Y(n_367));
AOI212XL g16844.(A(n_242), A1(n_298), B0(n_371), Y(n_348));
NAND2XL g16845.(A(n_4), B(n_368), Y(n_5));
CLKINVXL g16846.(A(n_371), Y(n_4));
AOI211XL g16847.(A(n_189), A1(n_344), B0(n_343), C0(n_346), Y(n_371));
AND3XL g16848.(A(n_341), A1(n_345), C(n_342), Y(n_346));
AOI212XL g16849.(A(n_266), A1(n_372), B0(n_319), Y(n_345));
AOI222XL g16850.(A(n_252), A1(n_339), B0(n_250), B1(n_337), Y(n_344));
AOI222XL g16851.(A(n_291), A1(n_331), B0(n_303), B1(n_336), Y(n_343));
AOI222XL g16852.(A(n_292), A1(n_333), B0(n_273), B1(n_340), Y(n_342));
NAND2BXL g16853.(AN(n_334), B(n_301), Y(n_341));
INVXL g16855.(A(n_337), Y(n_338));
AOI223XL g16857.(A(n_272), A1(n_374), B0(n_314), Y(n_340));
AOI231XL g16858.(A(n_187), A1(n_289), A2(n_374), B0(n_318), Y(n_372));
AOI221XL g16859.(A(n_241), A1(n_374), B0(n_311), Y(n_339));
AOI311XL g16860.(A(n_187), A1(n_255), A2(n_374), B0(n_312), Y(n_337));
CLKINVXL g16861.(A(n_335), Y(n_356));
CLKINVXL g16862.(A(n_332), Y(n_333));

```

Ln 102, Col 65 1,51,167 characters

RTL code: Router interlogic and ffs

```

reg [8:0] output_north1,output_south1,output_east1,output_
reg [8:0] output_north2,output_south2,output_east2,output_
assign output_north=output_north;
assign output_south=output_south;
assign output_east=output_east;
assign output_west=output_west;
assign output_processor=output_processor2;

// check whether route is free or not
reg regNR,regSR,regER,regW1,regPR;

//temporary variable used till clockedge is not reached
reg regNR1,regSR1,regER1,regW11,regPR1;

// Data Packet Routing (Crossbar Working)

always@(*)
begin
    case(select_north) //choosing which input of router will
        3'b000: output_north1 = data_north;
        3'b001: output_north1 = data_south;
        3'b010: output_north1 = data_east;
        3'b011: output_north1 = data_west;
        3'b100: output_north1 = data_processor;
        default: output_north1 = 9'b00000000;
    //3'b101: output_north1 = 9'b00000000;
    //3'b110: output_north1 = 9'b00000000;
    //3'b111: output_north1 = 9'b00000000;
    endcase
end

// assign output according to the output of MUX, at clock edge
end
always@(posedge clock or posedge reset)
begin
    if(reset==1'b1)
        begin
            regER<=1;
        end
    else
        begin
            regER=regR1; //assign path freeness according to the algorithm done above, at clock edge
        end
end
always@(posedge clock)
begin
    output_west2=<output_west1; //assign output according to the output of MUX done above, at clock edge
end
always@(posedge clock or posedge reset)
begin
    if(reset==1'b1)
        begin
            regW1<=1;
        end
    else
        begin
            regW1=regW11; //assign path freeness according to the algorithm done above, at clock edge
        end
end

```

Netlist: Router interlogic and ffs

```

main_syn.v          Untitled           Master_new.v          router.v          main.v
File   Edit   View
AOI22XL g661(.A0 (select_processor[0]), .A1 (n_114), .B0 (select_processor[2]), .B1 (n_116), .C0 (n_118), .Y (n_121));
AOI22XL g663(.A0 (select_north[0]), .A1 (n_115), .B0 (select_north[2]), .B1 (n_117), .C0 (n_119), .Y (n_120));
OAI33X1 g664(.A0 (select_north[2]), .A1 (data_north[8]), .A2 (n_111), .B0 (data_east[8]), .B1 (select_north[0]), .B2 (n_108), .Y (n_119));
OAI33X1 g665(.A0 (select_processor[2]), .A1 (data_north[8]), .A2 (n_113), .B0 (data_east[8]), .B1 (select_processor[0]), .B2 (n_109), .Y (n_118));
NAND2XL g666(.A (n_110), .B (data_processor[8]), .Y (n_117));
NAND2XL g667(.A (n_112), .B (data_processor[8]), .Y (n_116));
OAI22XL g668(.A0 (select_north[1]), .A1 (data_south[8]), .B0 (data_west[8]), .B1 (n_108), .Y (n_115));
OAI22XL g669(.A0 (select_processor[1]), .A1 (data_south[8]), .B0 (data_west[8]), .B1 (n_109), .Y (n_114));
CLKINVX1 g670(.A (n_112), .Y (n_113));
NOR2XL g671(.A (select_processor[0]), .B (select_processor[1]), .Y (n_112));
CLKINVX1 g672(.A (n_110), .Y (n_111));
NOR2XL g673(.A (select_north[0]), .B (select_north[1]), .Y (n_110));
CLKINVX1 g674(.A (data_processor[4]), .Y (n_128));
CLKINVX1 g675(.A (data_processor[3]), .Y (n_127));
CLKINVX1 g676(.A (data_processor[0]), .Y (n_129));
CLKINVX1 g677(.A (data_processor[5]), .Y (n_132));
INVXL g678(.A (data_processor[8]), .Y (n_143));
CLKINVX1 g679(.A (data_processor[7]), .Y (n_130));
CLKINVX1 g680(.A (data_processor[2]), .Y (n_126));
CLKINVX1 g681(.A (data_processor[1]), .Y (n_125));
CLKINVX1 g682(.A (data_processor[6]), .Y (n_131));
CLKINVX1 g683(.A (select_processor[1]), .Y (n_109));
CLKINVX1 g684(.A (select_north[1]), .Y (n_108));
SDFSSHQX1 regER_reg(.SN (n_67), .CK (clock), .D (n_98), .SI (east_ready), .SE (n_99), .Q (east_ready));
DFQFX1 \output_north2_reg[6] (.CK (clock), .D (n_106), .Q (output_north[6]));
DFQFX1 \output_north2_reg[5] (.CK (clock), .D (n_105), .Q (output_north[5]));
DFQFX1 \output_north2_reg[8] (.CK (clock), .D (n_107), .Q (output_north[8]));
DFQFX1 \output_north2_reg[4] (.CK (clock), .D (n_100), .Q (output_north[4]));
DFQFX1 \output_north2_reg[2] (.CK (clock), .D (n_102), .Q (output_north[2]));
DFQFX1 \output_north2_reg[1] (.CK (clock), .D (n_101), .Q (output_north[1]));
DFQFX1 \output_north2_reg[7] (.CK (clock), .D (n_104), .Q (output_north[7]));
DFQFX1 \output_north2_reg[3] (.CK (clock), .D (n_103), .Q (output_north[3]));
OAI32XL g1237(.A0 (n_86), .A1 (select_north[2]), .A2 (n_97), .B0 (n_77), .B1 (n_129), .Y (n_107));
OAI32XL g1238(.A0 (n_90), .A1 (select_north[2]), .A2 (n_91), .B0 (n_77), .B1 (n_131), .Y (n_106));

```

RTL: P_unit interlogic

```

main_syn.v          Untitled           main.v
File   Edit   View
reg tlstart1;
always@(*)
begin
    request_line=tb_request & processor_ready1; //high only when processor is free and user has asked for a request transfer from processor
end
always@(*)
begin
    if(reset==1'b1)
    begin
        which_processor1=2'b00; //set destination processor to default
    end
    else
    begin
        which_processor1=tb_processor; //and set which_processor according to the destination the user has set
    end
end
always@(*)
begin
    if(reset==1'b1)
    begin
        request_transfer1=1'b0; //if reset all requests are null and void
    end
    else
    begin
        request_transfer1=request_line; //else raise request_transfer if a valid request
    end
end
always@(posedge clock or posedge reset)
begin
    if(reset==1'b1)

```

Netlist: P_unit interlogic

File Edit View

```
UrrQX4 \p0_data_got1_reg[0] (.CK (clock), .D (r00[0]), .Q (p0_receive_data[0]));
DFFQX4 \p0_data_got1_reg[1] (.CK (clock), .D (r00[1]), .Q (p0_receive_data[1]));
DFFQX4 \p0_data_got1_reg[2] (.CK (clock), .D (r00[2]), .Q (p0_receive_data[2]));
DFFQX4 \p0_data_got1_reg[3] (.CK (clock), .D (r00[3]), .Q (p0_receive_data[3]));
DFFQX4 \p0_data_got1_reg[4] (.CK (clock), .D (r00[4]), .Q (p0_receive_data[4]));
DFFQX4 \p0_data_got1_reg[5] (.CK (clock), .D (r00[5]), .Q (p0_receive_data[5]));
DFFQX4 \p0_data_got1_reg[6] (.CK (clock), .D (r00[6]), .Q (p0_receive_data[6]));
DFFQX4 \p0_data_got1_reg[7] (.CK (clock), .D (r00[7]), .Q (p0_receive_data[7]));
DFFQX4 \p0_data_got1_reg[8] (.CK (clock), .D (r00[8]), .Q (p0_receive_data[8]));
DFFRHQX1 \p0_data_to_router1_reg[0] (.RN (n_0), .CK (clock), .D (p0_counter_value[0]), .Q (d00[0]));
DFFRHQX1 \p0_data_to_router1_reg[1] (.RN (n_0), .CK (clock), .D (p0_counter_value[1]), .Q (d00[1]));
DFFRHQX1 \p0_data_to_router1_reg[2] (.RN (n_0), .CK (clock), .D (p0_counter_value[2]), .Q (d00[2]));
DFFRHQX1 \p0_data_to_router1_reg[3] (.RN (n_0), .CK (clock), .D (p0_counter_value[3]), .Q (d00[3]));
DFFRHQX1 \p0_data_to_router1_reg[4] (.RN (n_0), .CK (clock), .D (p0_counter_value[4]), .Q (d00[4]));
DFFRHQX1 \p0_data_to_router1_reg[5] (.RN (n_0), .CK (clock), .D (p0_counter_value[5]), .Q (d00[5]));
DFFRHQX1 \p0_data_to_router1_reg[6] (.RN (n_0), .CK (clock), .D (p0_counter_value[6]), .Q (d00[6]));
DFFRHQX1 \p0_data_to_router1_reg[7] (.RN (n_0), .CK (clock), .D (p0_counter_value[7]), .Q (d00[7]));
DFFRHQX1 \p0_data_to_router1_reg[8] (.RN (n_0), .CK (clock), .D (p0_tlast), .Q (d00[8]));
DFFQX1 \p1_configure1_reg[3] (.CK (clock), .D (p1_configure[3]), .Q (p1_configure1[3]));
DFFQX1 \p1_configure1_reg[4] (.CK (clock), .D (p1_configure[4]), .Q (p1_configure1[4]));
DFFQX1 \p1_configure1_reg[5] (.CK (clock), .D (p1_configure[5]), .Q (p1_configure1[5]));
DFFQX1 \p1_configure1_reg[6] (.CK (clock), .D (p1_configure[6]), .Q (p1_configure1[6]));
DFFQX1 \p1_configure1_reg[7] (.CK (clock), .D (p1_configure[7]), .Q (p1_configure1[7]));
DFFQX1 \p1_configure1_reg[8] (.CK (clock), .D (p1_configure[8]), .Q (p1_configure1[8]));
DFFQX1 \p1_configure1_reg[9] (.CK (clock), .D (p1_configure[9]), .Q (p1_configure1[9]));
DFFQX1 \p1_configure1_reg[10] (.CK (clock), .D (p1_configure[10]), .Q (p1_configure1[10]));
DFFQX4 \p1_data_got1_reg[0] (.CK (clock), .D (r11[0]), .Q (p1_receive_data[0]));
DFFQX4 \p1_data_got1_reg[1] (.CK (clock), .D (r11[1]), .Q (p1_receive_data[1]));
DFFQX4 \p1_data_got1_reg[2] (.CK (clock), .D (r11[2]), .Q (p1_receive_data[2]));
DFFQX4 \p1_data_got1_reg[3] (.CK (clock), .D (r11[3]), .Q (p1_receive_data[3]));
DFFQX4 \p1_data_got1_reg[4] (.CK (clock), .D (r11[4]), .Q (p1_receive_data[4]));
DFFQX4 \p1_data_got1_reg[5] (.CK (clock), .D (r11[5]), .Q (p1_receive_data[5]));
DFFQX4 \p1_data_got1_reg[6] (.CK (clock), .D (r11[6]), .Q (p1_receive_data[6]));
DFFQX4 \p1_data_got1_reg[7] (.CK (clock), .D (r11[7]), .Q (p1_receive_data[7]));
```

RTL: mesh interlogic

File Edit View

```
// parameter NO_DATA=9'b00000000;
/ instantiation
master m#(
    .clock(clock),
    .reset(reset),
    .path_free_bits(Path_usage_bits),
    .P0_signals(P0_signals),
    .P1_signals(P1_signals),
    .P2_signals(P2_signals),
    .P3_signals(P3_signals),
    .block_all_paths(block_all_paths1),
    .R0_control_signals(R0_control_signals),
    .R1_control_signals(R1_control_signals),
    .R2_control_signals(R2_control_signals),
    .R3_control_signals(R3_control_signals),
    .response_signals(response_signals)
) response;

Processing_unit p0(
    .clock(clock),
    .reset(reset),
    .master_response(response_signals[0]),
    .data_from_router(r00),
    .data_to_router(d00),
    .request_transfer(P0_signals[1]),
    .which_processor(P0_signals[3:2]),
    .processor_ready(processor_ready_signals1[0]),
    .data_got(p0_receive_data),
    .tb_request(tb_configure1[0]),
    .tb_processor(tb_configure1[2:1]),
    .tb_len(tb_configure1[10:3])
);
/ Set commands by master
master m#(
    .SetWR(R3_control_signals[1]),
    .SetPR(R3_control_signals[0])
);

always @ (*) //router 0
begin
    Path_usage_bits_0[0] = Pr0 ; //0 to 0
    Path_usage_bits_0[1] = Er0 & Pr1; //0 to 1 //flat
    Path_usage_bits_0[2] = Nr0 & Er2 & Sr3 & Pr1 ; //0 to 1 longer 0-2-3-1
    Path_usage_bits_0[3] = Nr0 & Pr2; //0 to 2 //vertical
    Path_usage_bits_0[4] = Er0 & Nr1 & Wr3 & Pr2; //0 to 2 longer 0-1-3-2
    Path_usage_bits_0[5] = Nr0 & Er2 & Pr3 ; //0 to 3 //diagonal (vertical) 0-2-3
    Path_usage_bits_0[6] = Er0 & Nr1 & Pr3 ; //0 to 3 (flat) 0-1-3
end

always @ (*) //router1
begin
    Path_usage_bits_1[0] = Pr1 ; //1 to 1
    Path_usage_bits_1[1] = Er1 & Pr0; //1 to 0 //flat
    Path_usage_bits_1[2] = Nr1 & Wr3 & Sr2 & Pr0 ; //1 to 0 longer 1-3-2-0
    Path_usage_bits_1[3] = Nr1 & Pr3 ; //1 to 3 //vertical
    Path_usage_bits_1[4] = Nr1 & Er0 & Er2 & Pr3 ; // 1 to 3 longer 1-0-2-3
    Path_usage_bits_1[5] = Nr1 & Wr3 & Pr2 ; // 1 to 2 //diagonal (vertical) 1-3-2
    Path_usage_bits_1[6] = Nr1 & Nr0 & Pr2 ; //1 to 2 (flat) 1-0-2
end
```

In 143. Col 1 1 of 18,221 characters

Netlist: mesh interlogic

```

// synthesis_merge
// merged rep p0/data_to_router1_reg[8]
// merge + p0/tlast_prev_req
// merged rep p1/data_to_router1_reg[8]
// merge + p1/tlast_prev_req
// merged rep p2/data_to_router1_reg[8]
// merge + p2/tlast_prev_req
// merged rep p3/tlast_prev_req
// merge + p3/tlast_prev_req
// merged rep m0/R0_control_signals2_reg[8]
// merge + m0/R0_control_signals2_reg[9]
// merge + m0/R0_control_signals2_reg[10]
// merge + m0/R0_control_signals2_reg[14]
// merge + m0/R0_control_signals2_reg[15]
// merge + m0/R1_control_signals2_reg[16]
// merge + m0/R1_control_signals2_reg[8]
// merge + m0/R1_control_signals2_reg[9]
// merge + m0/R1_control_signals2_reg[11]
// merge + m0/R1_control_signals2_reg[12]
// merge + m0/R1_control_signals2_reg[13]
// merge + m0/R1_control_signals2_reg[14]
// merge + m0/R1_control_signals2_reg[15]
// merge + m0/R1_control_signals2_reg[16]
// merge + m0/R2_control_signals2_reg[8]
// merge + m0/R2_control_signals2_reg[10]
// merge + m0/R2_control_signals2_reg[14]
// merge + m0/R2_control_signals2_reg[17]
// merge + m0/R2_control_signals2_reg[18]
// merge + m0/R2_control_signals2_reg[19]
// merge + m0/R3_control_signals2_reg[11]
// merge + m0/R3_control_signals2_reg[12]
// merge + m0/R3_control_signals2_reg[13]
// merge + m0/R3_control_signals2_reg[17]
// ...

wire p0_n_58, p0_n_59, p0_tlast, p1_request_line, p1_tlast, p2_request_line, p2_tlast, p3_n_51;
wire p3_n_52, p3_n_53, p3_n_54, p3_n_57, p3_n_58, p3_n_59, p3_tlast;
master m0(.clock(clock), .reset(reset), .path_free_bits(Path_usage_bits_0[6:1], Pr0), P0_signals(P0_signals), P1_signals(P1_signals), P2_signals(P2_signals), P3_signals(P3_signals), block_all_paths(block_all_paths1), R0_control_signals({R0_control_signals[19:16], UNCONNECTED6, UNCONNECTED5, R0_control_signals[13:12], UNCONNECTED4, UNCONNECTED3, UNCONNECTED2, UNCONNECTED1, R0_control_signals[7:4], UNCONNECTED6, R0_control_signals[2], UNCONNECTED, R0_control_signals[0]}), R1_control_signals({R1_control_signals[19:18], UNCONNECTED17, UNCONNECTED16, UNCONNECTED15, UNCONNECTED14, UNCONNECTED13, UNCONNECTED12, UNCONNECTED11, R1_control_signals[10], UNCONNECTED10, UNCONNECTED9, R1_control_signals[7:6], UNCONNECTED2, R1_control_signals[4], UNCONNECTED7, R1_control_signals[2:0]}, R2_control_signals({UNCONNECTED26, UNCONNECTED25, UNCONNECTED24, R2_control_signals[16:15], UNCONNECTED23, UNCONNECTED18, R2_control_signals[8]}, R3_control_signals({UNCONNECTED35, UNCONNECTED34, UNCONNECTED33, R3_control_signals[16:15], UNCONNECTED32, UNCONNECTED31, UNCONNECTED30, UNCONNECTED29, R3_control_signals[10:5], UNCONNECTED28, R3_control_signals[3], UNCONNECTED27, R3_control_signals[1:0]}), response_signals(response_signals));
router r0(.clock(clock), .reset(reset), .select_north(R0_control_signals[19:17]), .select_south({UNCONNECTED_HIER_Z1, UNCONNECTED_HIER_Z0, UNCONNECTED_HIER_Z2}), .select_east({R0_control_signals[13], UNCONNECTED_HIER_Z22, R0_control_signals[12]}), .select_west({UNCONNECTED_HIER_Z4, UNCONNECTED_HIER_Z3, R0_control_signals[16]}), .select_processor(R0_control_signals[7:5]), .data_north(d20), .data_south(r0_input1[8:0]), .data_east(d10), .data_west(r0_input1[17:9]), .data_processor(d00), .output_north(d02), .output_south(r0_output1[8:0]), .output_east(d01), .output_west(r0_output1[17:9]), .output_processor(r0), .south_ready(Sr0), .east_ready(Er0), .west_ready(Wr0), .north_ready(Nr0), .south_ready(Sr1), .east_ready(Er1), .west_ready(Wr1), .processor_ready(Pr0), .SetNR(R0_control_signals[4]), .SetSR(UNCONNECTED_HIER_Z5), .SetER(R0_control_signals[2]), .SetWR(UNCONNECTED_HIER_Z6), .SetPR(R0_control_signals[0]));
router_r1 r1(.clock(clock), .reset(reset), .select_north({R1_control_signals[19], UNCONNECTED_HIER_Z7, R1_control_signals[18]}), .select_south({UNCONNECTED_HIER_Z10, UNCONNECTED_HIER_Z9, UNCONNECTED_HIER_Z8}), .select_east({UNCONNECTED_HIER_Z13, UNCONNECTED_HIER_Z12, UNCONNECTED_HIER_Z11}), .select_west({R1_control_signals[10], UNCONNECTED_HIER_Z14, R0_control_signals[16]}), .select_processor({R1_control_signals[7], UNCONNECTED_HIER_Z15, R1_control_signals[6]}, .data_north(d31), .data_south({UNCONNECTED_HIER_Z24, UNCONNECTED_HIER_Z23, UNCONNECTED_HIER_Z22, UNCONNECTED_HIER_Z21, UNCONNECTED_HIER_Z20, UNCONNECTED_HIER_Z19, UNCONNECTED_HIER_Z18, UNCONNECTED_HIER_Z17, UNCONNECTED_HIER_Z16}), .data_east({UNCONNECTED_HIER_Z33, UNCONNECTED_HIER_Z32, UNCONNECTED_HIER_Z31, UNCONNECTED_HIER_Z30, UNCONNECTED_HIER_Z29, UNCONNECTED_HIER_Z28, UNCONNECTED_HIER_Z27, UNCONNECTED_HIER_Z26, UNCONNECTED_HIER_Z25}), .data_west(d01), .data_processor(d11), .output_north(d13), .output_south(r1_output1[8:0]), .output_east(r1_output1[17:9]), .output_west(d10), .output_processor(r11), .north_ready(Nr1), .south_ready(Sr1), .east_ready(Er1), .west_ready(Wr1), .processor_ready(Pr1), .SetNR(R1_control_signals[4]), .SetSR(UNCONNECTED_HIER_Z34), .SetER(R1_control_signals[2]), .SetWR(R1_control_signals[1]), .SetPR(R1_control_signals[0]));
router_r2 r2(.clock(clock), .reset(reset), .select_north({UNCONNECTED_HIER_Z37, UNCONNECTED_HIER_Z36, UNCONNECTED_HIER_Z35}), .select_south({R2_control_signals[16:15], UNCONNECTED_HIER_Z38}), .select_east({R2_control_signals[13:11], UNCONNECTED_HIER_Z40, UNCONNECTED_HIER_Z39, R0

```

Please Note -> Signal bandwidths are too high as well as we have too many signals

As a result it is not feasible to show all of the logic here. Though we have identified their place in the netlist.

We have manually cross verified the complete Netlist and our RTL design.

4. Logical Equivalence Checking

Inputs: RTL, netlist generated after synthesis.

Output: Report which tells whether the design and mapped points are equivalent.

Software Used: Conformal (Cadence)

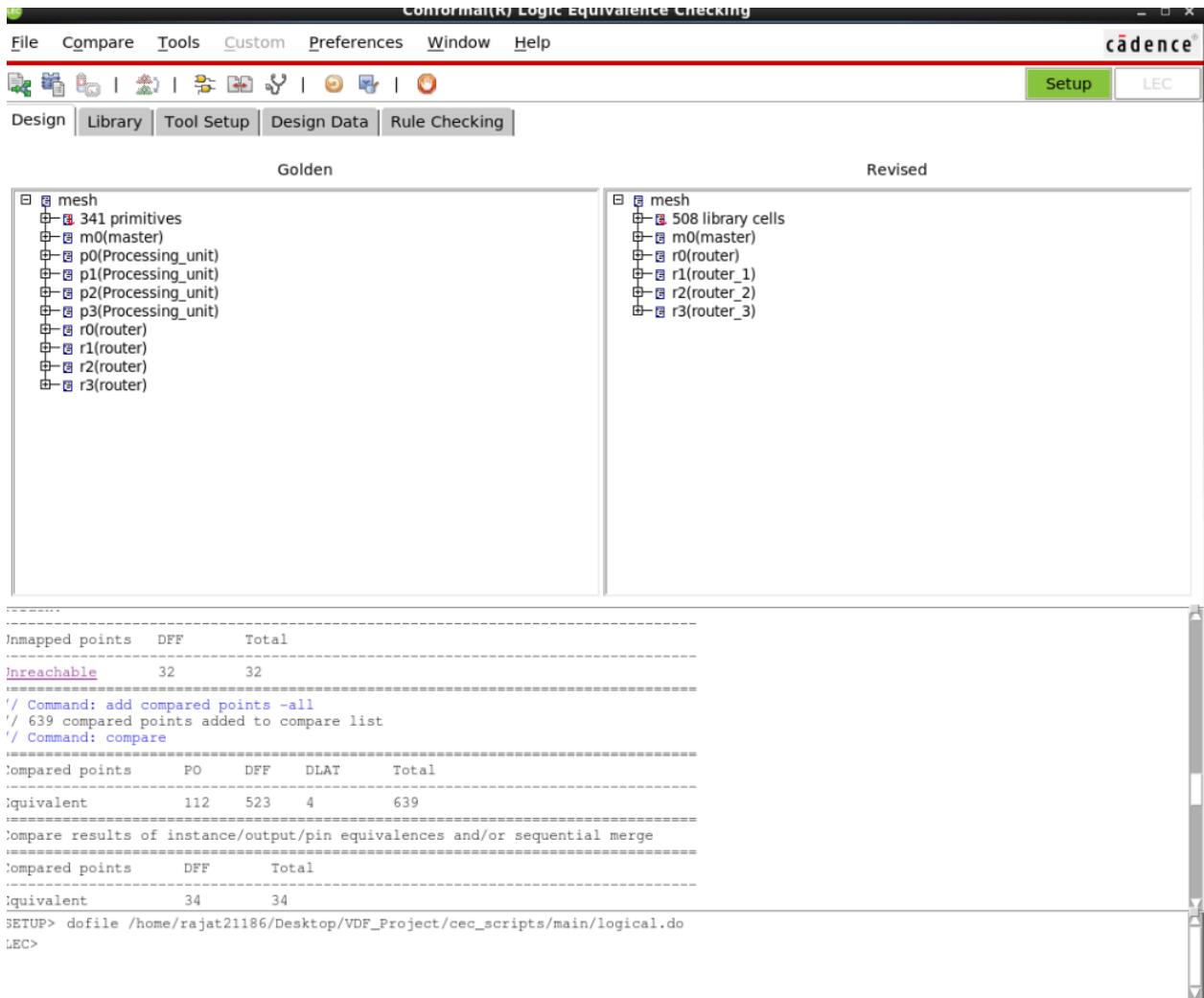
Commands Used:

1. csh
2. Source /cadence/cshrc
3. Lec -lpgxl -dofile -logical.do

Logical Equivalence Checking is done to verify that the two representations of the same design will exhibit the same behavior.

One-to-one port matching is done. All the ports and pins are mapped in golden(RTL) and revised (synthesized netlist), and the tool compares ports other than the inputs.

Equivalence Checking for Minimum Area:



rajab21186@edaserver4:work ground

Mapping Manager

Close Schematics Refresh Preferences Window Help

Jnmapped Points **Unreachable**

	DFF	444	m0/R3_control_signals2_reg[4]
0	DFF	446	m0/R3_control_signals2_reg[2]
0	DFF	464	m0/R2_control_signals2_reg[4]
0	DFF	467	m0/R2_control_signals2_reg[1]
0	DFF	485	m0/R1_control_signals2_reg[3]
0	DFF	505	m0/R0_control_signals2_reg[3]
0	DFF	507	m0/R0_control_signals2_reg[1]

Mapped Points

(+)	PI	1	clock
(+)	PI	2	reset
(+)	PI	3	p0_configure[10]
(+)	PI	4	p0_configure[9]
(+)	PI	5	p0_configure[8]
(+)	PI	6	p0_configure[7]
(+)	PI	7	p0_configure[6]

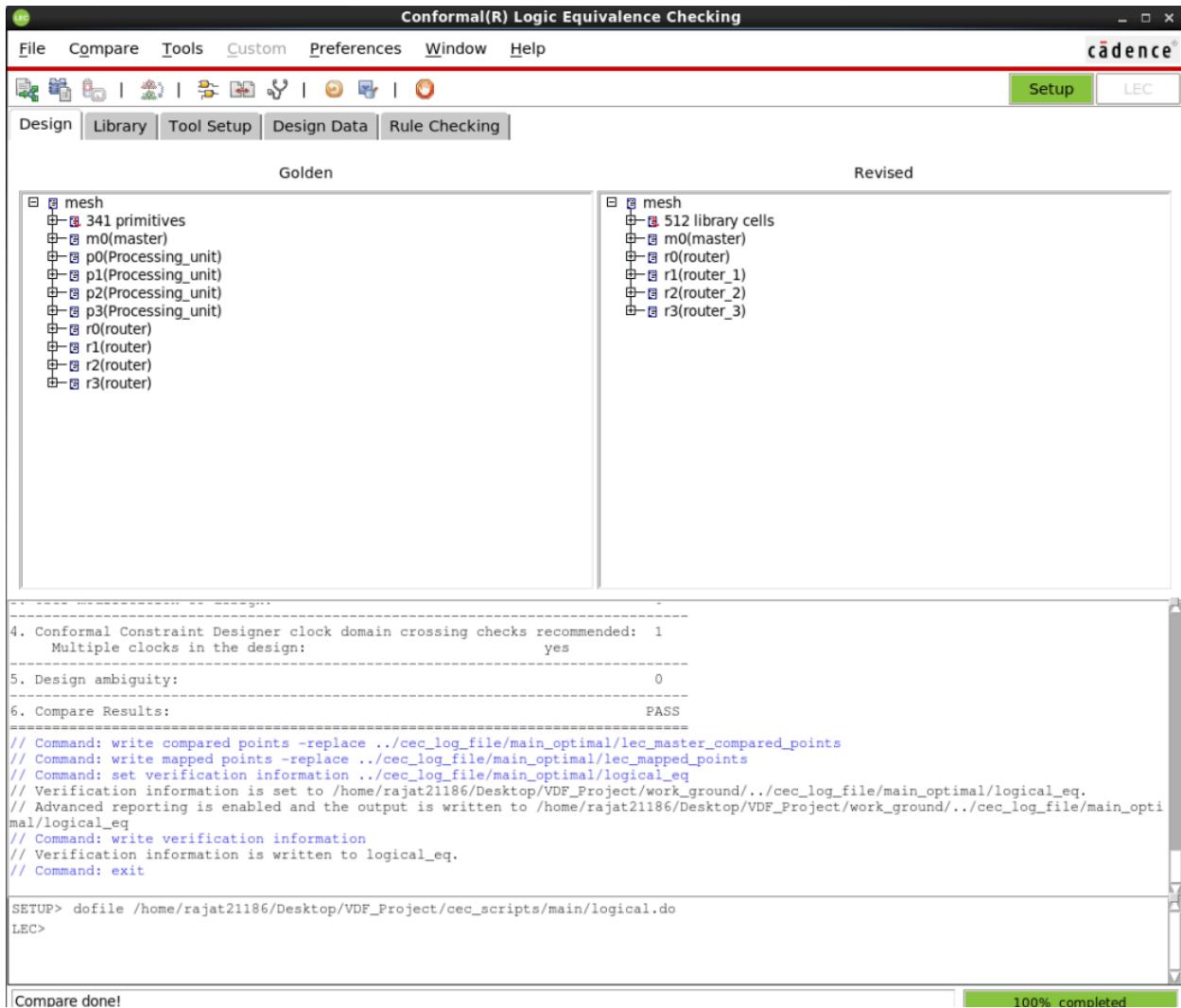
(+)	PI	1	clock
(+)	PI	2	reset
(+)	PI	3	p0_configure[10]
(+)	PI	4	p0_configure[9]
(+)	PI	5	p0_configure[8]
(+)	PI	6	p0_configure[7]
(+)	PI	7	p0_configure[6]

Compared Points **Compared Class** **Support Size** **Support Size**

(+)	S	DFF	597	p1/data_to_router1_reg[8]
(+)	S	DFF	753	p3/data_to_router1_reg[8]
(+)	S	DFF	675	p2/data_to_router1_reg[8]
(+)	S	DFF	519	p0/data_to_router1_reg[8]
(+)	S	DFF	500	m0/R0_control_signals2_reg[8]
(+)	S	DFF	434	m0/R3_control_signals2_reg[14]
(+)	S	DFF	471	m0/R1_control_signals2_reg[17]
(+)	S	DFF	497	m0/R0_control_signals2_reg[11]
(+)	S	DFF	483	m0/R1_control_signals2_reg[5]

(+)	DFF	568	p1_data_to_router1_reg[8]/U\$1/U\$1
(+)	DFF	612	p3_data_to_router1_reg[8]/U\$1/U\$1
(+)	DFF	594	p2_data_to_router1_reg[8]/U\$1/U\$1
(+)	DFF	542	p0_data_to_router1_reg[8]/U\$1/U\$1
(+)	DFF	252	m0/R0_control_signals2_reg[8]/U\$1/U\$1
(+)	DFF	277	m0/R3_control_signals2_reg[14]/U\$1/U\$1
(+)	DFF	261	m0/R1_control_signals2_reg[17]/U\$1/U\$1
(+)	DFF	253	m0/R0_control_signals2_reg[11]/U\$1/U\$1
(+)	DFF	258	m0/R1_control_signals2_reg[5]/U\$1/U\$1

Equivalence Checking for Optimal Constraints:



Mapping Manager

Close Schematics Refresh Preferences Window Help

Unmapped Points Unmapped Class

- (!) DFF 444 m0/R3_control_signals2_reg[4]
- (!) DFF 446 m0/R3_control_signals2_reg[2]
- (!) DFF 464 m0/R2_control_signals2_reg[4]
- (!) DFF 467 m0/R2_control_signals2_reg[1]
- (!) DFF 485 m0/R1_control_signals2_reg[3]
- (!) DFF 505 m0/R0_control_signals2_reg[3]
- (!) DFF 507 m0/R0_control_signals2_reg[1]

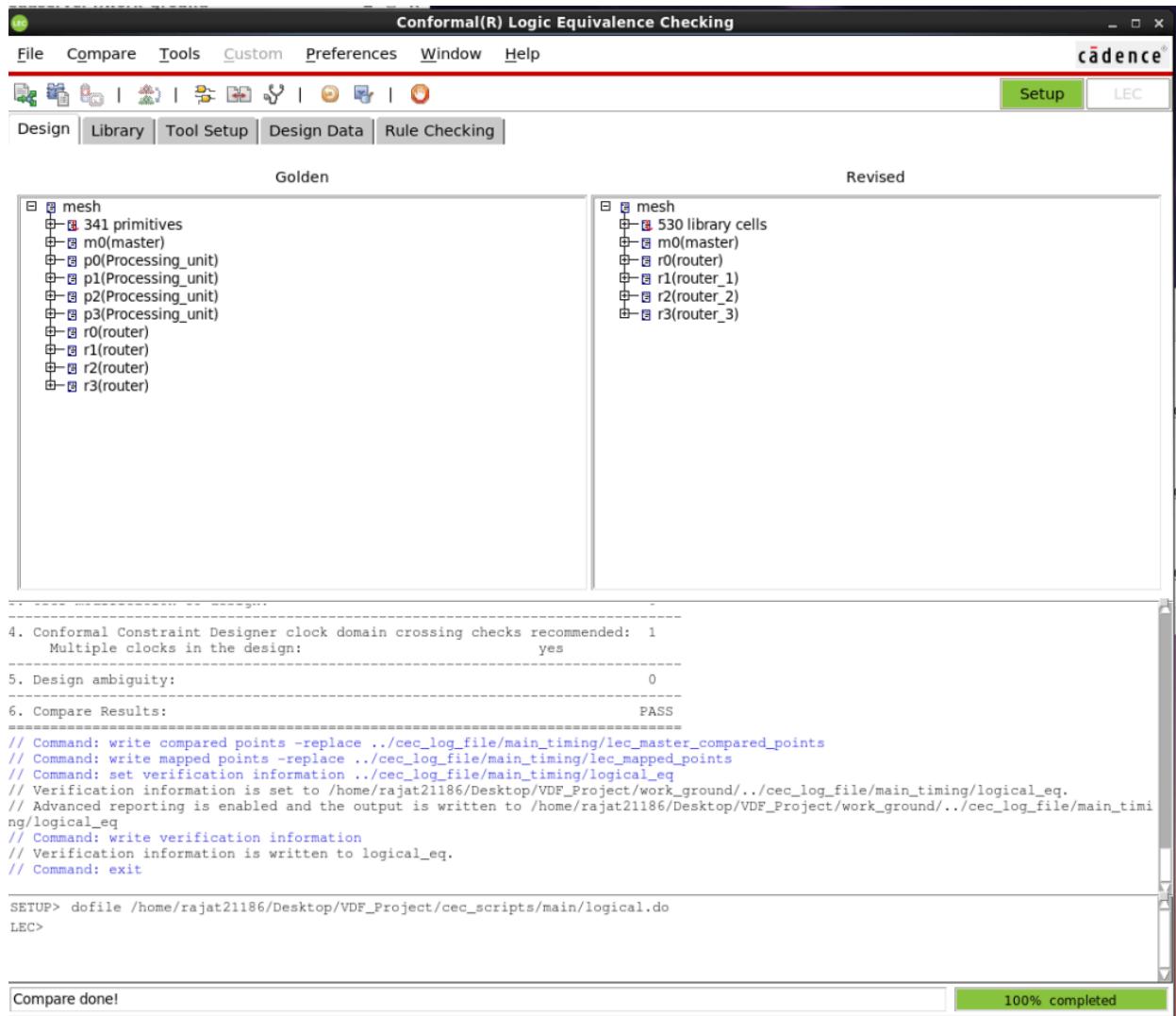
Mapped Points

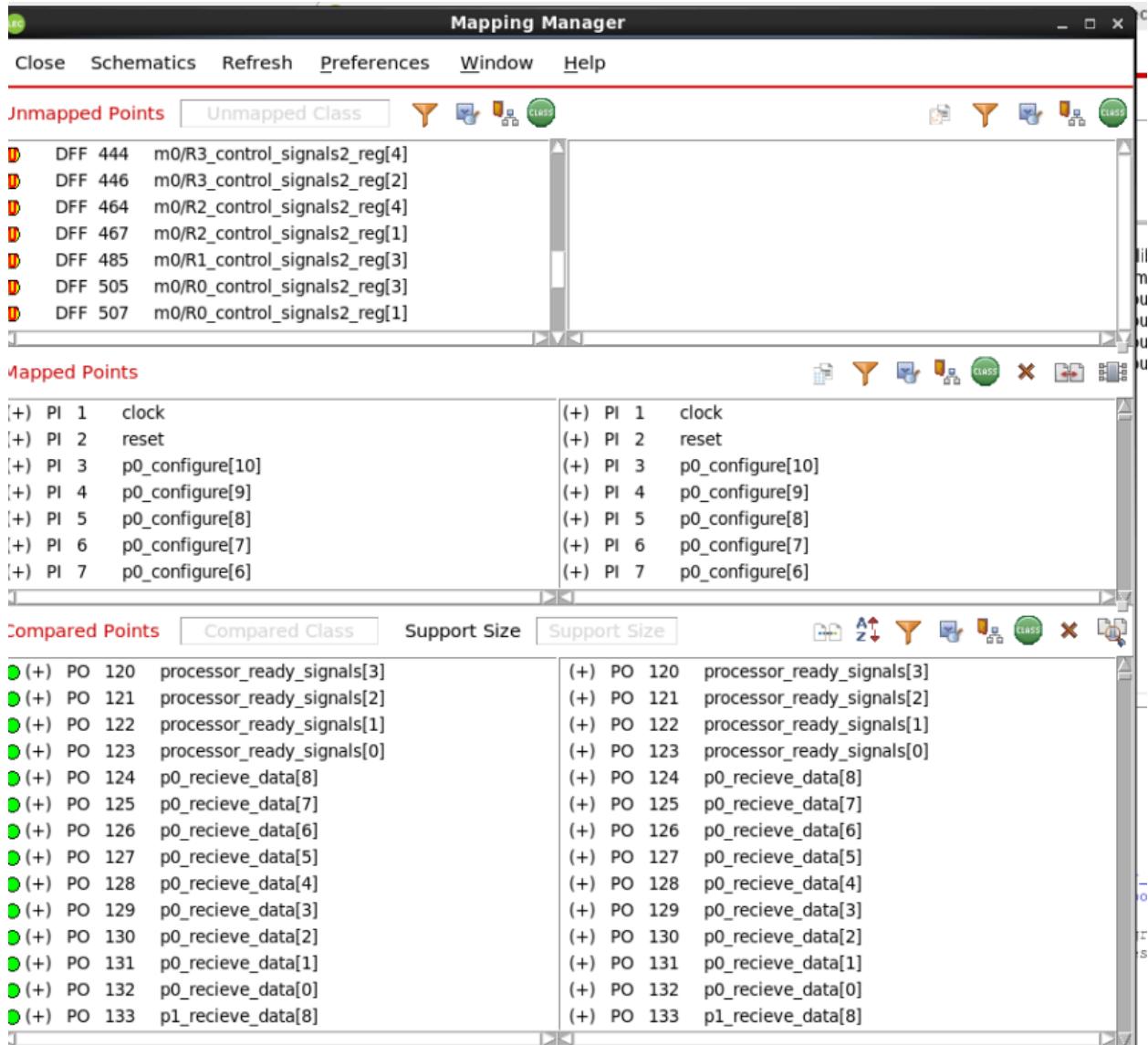
(+) PI 1	clock	(+) PI 1	clock
(+) PI 2	reset	(+) PI 2	reset
(+) PI 3	p0_configure[10]	(+) PI 3	p0_configure[10]
(+) PI 4	p0_configure[9]	(+) PI 4	p0_configure[9]
(+) PI 5	p0_configure[8]	(+) PI 5	p0_configure[8]
(+) PI 6	p0_configure[7]	(+) PI 6	p0_configure[7]
(+) PI 7	p0_configure[6]	(+) PI 7	p0_configure[6]

Compared Points Compared Class Support Size

(+) PO 120	processor_ready_signals[3]	(+) PO 120	processor_ready_signals[3]
(+) PO 121	processor_ready_signals[2]	(+) PO 121	processor_ready_signals[2]
(+) PO 122	processor_ready_signals[1]	(+) PO 122	processor_ready_signals[1]
(+) PO 123	processor_ready_signals[0]	(+) PO 123	processor_ready_signals[0]
(+) PO 124	p0_recieve_data[8]	(+) PO 124	p0_recieve_data[8]
(+) PO 125	p0_recieve_data[7]	(+) PO 125	p0_recieve_data[7]
(+) PO 126	p0_recieve_data[6]	(+) PO 126	p0_recieve_data[6]
(+) PO 127	p0_recieve_data[5]	(+) PO 127	p0_recieve_data[5]
(+) PO 128	p0_recieve_data[4]	(+) PO 128	p0_recieve_data[4]
(+) PO 129	p0_recieve_data[3]	(+) PO 129	p0_recieve_data[3]
(+) PO 130	p0_recieve_data[2]	(+) PO 130	p0_recieve_data[2]
(+) PO 131	p0_recieve_data[1]	(+) PO 131	p0_recieve_data[1]
(+) PO 132	p0_recieve_data[0]	(+) PO 132	p0_recieve_data[0]
(+) PO 133	p1_recieve_data[8]	(+) PO 133	p1_recieve_data[8]

Equivalence Checking for Best Timing:





The reason you see unmapped points in the design because while synthesizing the tool says that these variables do not drive any main output. In our mesh design we can see that router zero has no south or west , router 1 has no south or east , router 2 has no north or west, router 3 has no north or east and the unmapped are related to this so they do not do anything because that is not possible in our design.

Also we are getting Conformal Constraint Designer Crossing checks because our Processing_unit module contains a latch because we need this in our functionality.

☐ Equivalence checking for Bad Netlist

```

// Generated by Cadence Genus(TM) Synthesis Solution 19.13-s073 1
// Generated on: Mar 23 2024 01:55:51 IST (Mar 22 2024 20:25:51 UTC)
// Verification Directory fv/mesh

module router(clock, reset, select_north, select_south, select_east, select_west, select_processor, data_north, data_south, data_east, data_west, data_processor, output_north, output_south, output_east, output_west, output_processor, data_north, data_south, data_east, data_west, data_processor);
input clock, reset, SetNR, SetSR, SetER, SetWR, SetPR;
input [8:0] select_north, select_south, select_east, select_west, select_processor;
input [8:0] data_north, data_south, data_east, data_west, GtA_processor;
output [8:0] output_north, output_south, output_east, output_west, output_processor;
output [8:0] select_north, select_south, select_east, select_west, processor_ready;
wire [8:0] select_north, select_south, select_east, select_west, select_processor;
wire [8:0] data_north, data_south, data_east, data_west, data_processor;
wire [8:0] output_north, output_south, output_east, output_west, output_processor;
wire [8:0] select_north, select_south, select_east, select_west, ready;
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, n_36, n_37, n_38, n_39;
wire n_40, n_41, n_42, n_43, n_44, n_45, n_46, n_47;
wire n_48, n_49, n_50, n_51, n_52, n_53, n_54, n_55;
wire n_56, n_57, n_58, n_59, n_60, n_61, n_62, n_63;
wire n_64, n_65, n_66, n_67, n_68, n_69, n_70, n_71, n_72, n_73;
wire n_74, n_75, n_76, n_77, n_78, n_79, n_80, n_81, n_82, n_83;
wire n_84, n_85, n_86, n_87, n_88, n_89, n_90, n_91, n_92;
wire n_93, n_94, n_95, n_96, n_97, n_98, n_99, n_100, n_101;
wire n_102, n_103, n_104, n_105, n_106, n_107, n_108, n_109;
wire n_110, n_111, n_112, n_113, n_114, n_115, n_116, n_117;
wire n_118, n_119, n_120, n_121, n_122, n_123, n_124, n_125, n_126, n_127, n_128;
wire n_129, n_130, n_131, n_132, n_133, n_134, n_135, n_136;
wire n_206;

AO122XL g661.AB (select_processor[0]), .A1(n_114), .B0(select_processor[1]), .B1(n_116), .C0(n_119), .Y(n_121);
AO122XL g662.AB (select_processor[1]), .A1(n_115), .B0(select_processor[0]), .B1(n_117), .C0(n_119), .Y(n_120);
OA133XL g664.AB (select_north[2]), .A1(data_north[8]), .A2(n_111), .B0(data_east[8]), .B1(select_north[8]), .B2(n_108), .Y(n_119);
OA133XL g665.AB (select_processor[2]), .A1(data_north[8]), .A2(n_113), .B0(data_east[8]), .B1(select_processor[8]), .B2(n_109), .Y(n_118);
NAND2XL g666.A (n_110), .B (data_processor[8]), .Y (n_117);
NOR2XL g667.A (data_processor[8]), .B (n_110), .Y (n_116);
OA122XL g668.AB (select_north[1]), .A1(data_south[8]), .B0(data_west[8]), .B1(n_108), .Y(n_115);
OA122XL g669.AB (select_processor[1]), .A1(data_south[8]), .B0(data_west[8]), .B1(n_109), .Y(n_114);
CLKINVXL g670.A (n_112), .Y (n_113);
NOR2XL g671.A (data_processor[4]), .B (n_110), .B0(select_processor[1]), .Y (n_112);
CLKINVXL g672.A (n_110), .Y (n_111);
NOR2XL g673.A (select_north[8]), .B (select_north[1]), .Y (n_110);
CLKINVXL g674.A (data_processor[4]), .Y (n_128);

```

1) Equivalence checking after manually editing the netlist to remove module master.

- ❖ As the module referred in is not defined the tool throws an error.

```

Conformal(R) Logic Equivalence Checking
File Compare Tools Custom Preferences Window Help
Setup LEC
Design Library Tool Setup Design Data Rule Checking
Golden Revised
Golden
Revised
// Command: read_design /home/rajat21186/Desktop/VDF_Project/RTL_Code/main.v -verilog -golden
// Parsing file /home/rajat21186/Desktop/VDF_Project/RTL_Code/main.v ...
// Parsing file /home/rajat21186/Desktop/VDF_Project/RTL_Code/Master_new.v ...
// Parsing file /home/rajat21186/Desktop/VDF_Project/RTL_Code/P/unit.v ...
// Parsing file /home/rajat21186/Desktop/VDF_Project/RTL_Code/router.v ...
// Golden root module is set to 'mesh'
// Note: (BTL#(2)) Latch(em) are inferred due to occurrence!(occurrence!);
// Note: (BTL#(2)) Latch(em) are inferred due to self assignment (occurrence!);
// Warning: (BTL#(4)) Signal has input but it has no output (occurrence?);
// Note: Read VERILOG design successfully
CPU time : 0.70 seconds
Elapsed time : 0.70 seconds
Memory usage : 290.81 M bytes
// Command: read_design /home/rajat21186/Desktop/VDF_Project/design/main_min_area_netlist/main_syn.v -verilog -revised
// Parsing file /home/rajat21186/Desktop/VDF_Project/design/main_min_area_netlist/main_syn.v ...
// Error: /home/rajat21186/Desktop/VDF_Project/design/main_min_area_netlist/main_syn.v:195 Module 'master' is referenced but not defined.
// Dofile /home/rajat21186/Desktop/VDF_Project/cec_scripts/main/logical.do is aborted at line 7
SETUP> dofile /home/rajat21186/Desktop/VDF_Project/cec_scripts/main/logical.do
SETUP>

```

2) Equivalence checking after manually interchanging a port in a line in netlist has resulted in mapping error.

- ❖ We had changed OA22X1 g2115 parameter p3_configure[5] to p3_configure[4]
- ❖ The comparison revealed non-equivalence between two points, denoted as "NONEQ" for non-equivalence. The reason being unmapped points.

```

NAND2XL g2107(.A (n_141), .B (p0_counter_value[5]), .Y (n_143));
NAND2XL g2108(.A (n_140), .B (p3_counter_value[5]), .Y (n_142));
AOI21X1 g2109(.A0 (n_125), .A1 (n_137), .B0 (n_141), .Y (p0_n_56));
NOR2XL g2111(.A (n_125), .B (n_137), .Y (n_141));
NOR2XL g2112(.A (n_121), .B (n_136), .Y (n_140));
OAI21X1 g2114(.A0 (p0_counter_value[3]), .A1 (n_134), .B0 (n_137), .Y (n_139));
OA22X1 g2115(.A0 (p3_configure1[5]), .A1 (n_135), .B0 (p3_configure1[4]), .B1 (n_128), .Y (n_138));
OA21X1 g2116(.A0 (p3_counter_value[3]), .A1 (n_133), .B0 (n_136), .Y (p3_n_55));
NAND2XL g2117(.A (n_134), .B (p0_counter_value[3]), .Y (n_137));
NAND2XL g2118(.A (n_133), .B (p3_counter_value[3]), .Y (n_136));
INVX1 g2119(.A (p3_n_54), .Y (n_135));

```

```

AOI21X1 g2109(.A0 (n_125), .A1 (n_137), .B0 (n_141), .Y (p0_n_56));
NOR2XL g2111(.A (n_125), .B (n_137), .Y (n_141));
NOR2XL g2112(.A (n_121), .B (n_136), .Y (n_140));
OAI21X1 g2114(.A0 (p0_counter_value[3]), .A1 (n_134), .B0 (n_137), .Y (n_139));
OA22X1 g2115(.A0 (p3_configure1[4]), .A1 (n_135), .B0 (p3_configure1[4]), .B1 (n_128), .Y (n_138));
OA21X1 g2116(.A0 (p3_counter_value[3]), .A1 (n_133), .B0 (n_136), .Y (p3_n_55));
NAND2XL g2117(.A (n_134), .B (p0_counter_value[3]), .Y (n_137));
NAND2XL g2118(.A (n_133), .B (p3_counter_value[3]), .Y (n_136));
INVX1 g2119(.A (p3_n_54), .Y (n_135));

```

```

=====
Due to these Non-equivalent points:
[DATA]
(G) + 8525 DC /p3/tlast_reg
(R) + 6082 INV /g2053/U$2

1 Non-equivalent point(s) reported
1 compared point(s) reported
=====
Compared points      P0      DFF     DLAT    Total
-----
Equivalent          112     522      4       638
-----
Non-equivalent      0       1       0       1
=====

0 Non-equivalent point(s) reported
0 compared point(s) reported
Compare results of instance/output/pin equivalences and/or sequential merge
=====
Compared points      DFF      Total
-----
Equivalent          34       34
-----
// Command: report verification
=====
Verification Report
=====
Category           Count
-----
1. Non-standard modeling options used:          0
-----
2. Incomplete verification:                     0
-----
3. User modification to design:                0
-----
4. Conformal Constraint Designer clock domain crossing checks recommended: 1
   Multiple clocks in the design:               yes
-----
5. Design ambiguity:                           0
-----
6. Compare Results:                           FAIL:NONEQ
=====

// Command: write compared points -replace ../../cec_log_file/main_optimal/lec_master_compared_points
// Command: write mapped points -replace ../../cec_log_file/main_optimal/lec_mapped_points
// Command: set verification information ../../cec_log_file/main_optimal/logical_eq
// Verification information is set to /home/rajat21186/Desktop/VDF_Project/work_ground/../../cec_log_file/main_optimal/logical_eq
// Advanced reporting is enabled and the output is written to /home/rajat21186/Desktop/VDF_Project/work_ground/../../cec_log_file/main_optimal/logical_eq
// Command: write verification information
// Verification information is written to logical_eq.
// Command: exit!

```

3) Equivalence checking after manually modifying a gate.

- ❖ We changed one NOR gate to OR gate, due to the above changes the compared 2 netlists have become inequivalent

```
-----  
AOI211X1 g2451(.A0 (n_120), .A1 (n_126), .B0 (n_134), .C0 (p0_n_51), .Y (n_318));  
AOI211X1 g2452(.A0 (n_121), .A1 (n_136), .B0 (n_140), .C0 (p3_n_51), .Y (n_319));  
AOI211X1 g2453(.A0 (n_123), .A1 (n_142), .B0 (n_147), .C0 (p3_n_51), .Y (n_320));  
NOR2XL g2454(.A (n_139), .B (p0_n_51), .Y (n_321));  
NOR2XL g2455(.A (n_144), .B (p3_n_51), .Y (n_322));  
-----
```

```
-----  
AOI211X1 g2451(.A0 (n_120), .A1 (n_126), .B0 (n_134), .C0 (p0_n_51), .Y (n_318));  
AOI211X1 g2452(.A0 (n_121), .A1 (n_136), .B0 (n_140), .C0 (p3_n_51), .Y (n_319));  
AOI211X1 g2453(.A0 (n_123), .A1 (n_142), .B0 (n_147), .C0 (p3_n_51), .Y (n_320));  
NOR2XL g2454(.A (n_139), .B (p0_n_51), .Y (n_321));  
OR2XL g2455(.A (n_144), .B (p3_n_51), .Y (n_322));  
-----
```

```

=====
(G) + 8525 DC  /p3/tlast_reg
(R) + 6081 INV  /g2053/U$2

Compared points are: Non-equivalent
(G) + 747 DFF  /p3/counter_value_reg[5]
(R) + 522 DFF  /p3_counter_value_reg[5]/U$1/U$1
Due to these Non-equivalent points:
[DATA]
(G) + 8167 MUX  /p3/U$4/U$6
(R) + 6030 OR   /g2455/U$1

2 Non-equivalent point(s) reported
2 compared point(s) reported
=====
Compared points      PO      DFF      DLAT      Total
-----
Equivalent          112     521      4        637
-----
Non-equivalent      0       2        0        2
=====
0 Non-equivalent point(s) reported
0 compared point(s) reported
Compare results of instance/output/pin equivalences and/or sequential merge
=====
Compared points      DFF      Total
-----
Equivalent          34       34
-----
// Command: report verification
=====
                         Verification Report
-----
Category                                     Count
-----
1. Non-standard modeling options used:           0
-----
2. Incomplete verification:                     0
-----
3. User modification to design:                0
-----
4. Conformal Constraint Designer clock domain crossing checks recommended: 1
   Multiple clocks in the design:               yes
-----
5. Design ambiguity:                           0
-----
6. Compare Results:                           FAIL:NONEQ
-----
// Command: write compared points -replace ../cec_log_file/main_optimal/lec_master_compared_points
// Command: write mapped points -replace ../cec_log_file/main_optimal/lec_mapped_points

```

5.STA

Inputs:

Synthesized netlist, library file and constraints file

Output:

Violation report

Software Used:

Cadence Tempus

Commands Used:

1. csh
2. source /cadence/cshrc
3. tempus -nowin
4. Source (name of TCL file)

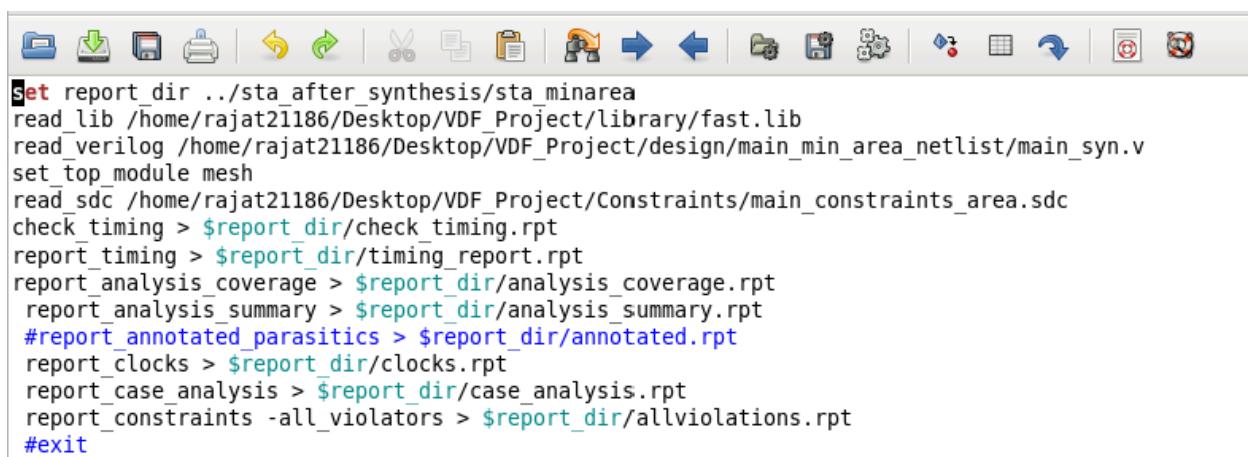
Remarks

The Static Timing Analysis (STA) indicates that there are no Setup Violations observed. All data reliably reaches the input of the flip-flops before the setup time of the subsequent clock edge. Hold violations will be resolved during clock tree synthesis.

STA for Minimum Area

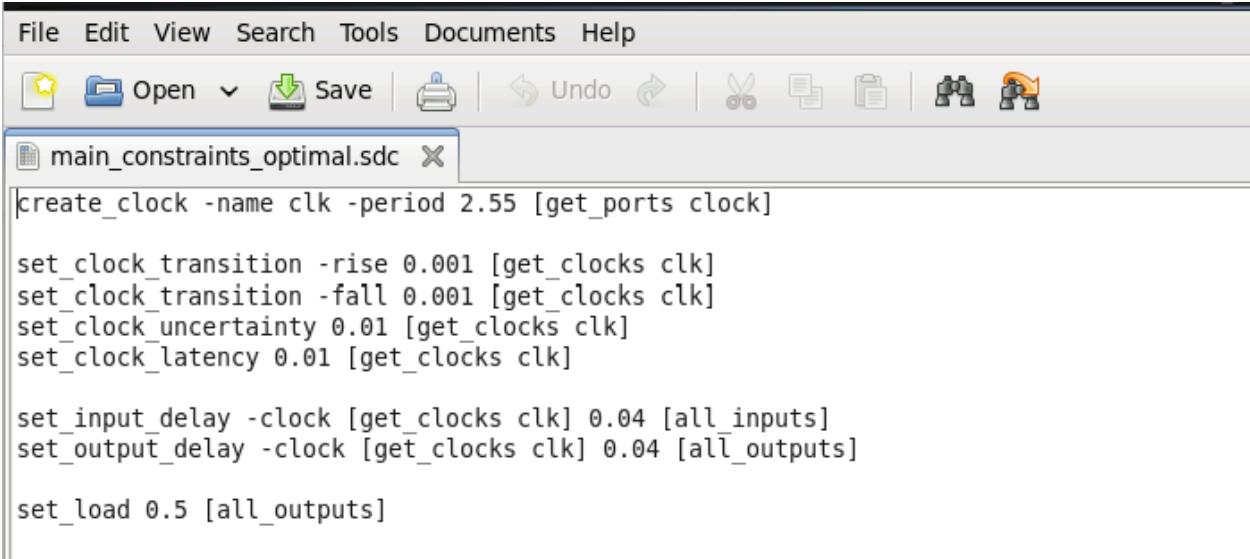
It can be seen that the start point is reset and the end point is taken at m0/R3_control_signals2_reg[16]/D which is the worst case for the problem statement. We have obtained a positive slack of 0.545.

TCL file:



```
set report_dir ..\sta_after_synthesis\sta_minarea
read_lib /home/rajat21186/Desktop\VDF_Project/library/fast.lib
read_verilog /home/rajat21186/Desktop\VDF_Project/design/main_min_area_netlist/main_syn.v
set_top_module mesh
read_sdc /home/rajat21186/Desktop\VDF_Project/Constraints/main_constraints_area.sdc
check_timing > $report_dir/check_timing.rpt
report_timing > $report_dir/timing_report.rpt
report_analysis_coverage > $report_dir/analysis_coverage.rpt
report_analysis_summary > $report_dir/analysis_summary.rpt
#report_annotated_parasitics > $report_dir/annotated.rpt
report_clocks > $report_dir/clocks.rpt
report_case_analysis > $report_dir/case_analysis.rpt
report_constraints -all_violators > $report_dir/allviolations.rpt
#exit
```

Constraints used:



The screenshot shows a software application window with a menu bar at the top containing File, Edit, View, Search, Tools, Documents, and Help. Below the menu is a toolbar with icons for Open, Save, Undo, Redo, Cut, Copy, Paste, and Find. The main area is a code editor with the title "main_constraints_optimal.sdc". The code in the editor is as follows:

```
create_clock -name clk -period 2.55 [get_ports clock]

set_clock_transition -rise 0.001 [get_clocks clk]
set_clock_transition -fall 0.001 [get_clocks clk]
set_clock_uncertainty 0.01 [get_clocks clk]
set_clock_latency 0.01 [get_clocks clk]

set_input_delay -clock [get_clocks clk] 0.04 [all_inputs]
set_output_delay -clock [get_clocks clk] 0.04 [all_outputs]

set_load 0.5 [all_outputs]
```

Violation report:

File Edit View Search Tools Documents Help

Open Save Undo | Print | Cut Copy Paste | Find Replace | Zoom | Help

allviolations.rpt x

```
#####
# Generated by: Cadence Tempus 20.10-p003_1
# OS: Linux x86_64(Host ID edaserver4)
# Generated on: Sat Mar 23 02:07:40 2024
# Design: mesh
# Command: report_constraints -all_violators > ../../sta_after_synthesis/sta_minarea/allviolations.rpt
#####
# format : frame 0 : split 1

max_delay/setup
-----
No paths found

min_delay/hold
-----
-----
```

End Point	Slack	Cause
r3_input1_reg[17]/D f	-0.196	VIOLATED
r3_input1_reg[16]/D f	-0.196	VIOLATED
r3_input1_reg[15]/D f	-0.196	VIOLATED
r3_input1_reg[14]/D f	-0.196	VIOLATED
r3_input1_reg[13]/D f	-0.196	VIOLATED
r3_input1_reg[12]/D f	-0.196	VIOLATED
r3_input1_reg[11]/D f	-0.196	VIOLATED
r3_input1_reg[10]/D f	-0.196	VIOLATED
r3_input1_reg[9]/D f	-0.196	VIOLATED
r3_input1_reg[8]/D f	-0.196	VIOLATED
r3_input1_reg[7]/D f	-0.196	VIOLATED
r3_input1_reg[6]/D f	-0.196	VIOLATED
r3_input1_reg[5]/D f	-0.196	VIOLATED
r3_input1_reg[4]/D f	-0.196	VIOLATED
r3_input1_reg[3]/D f	-0.196	VIOLATED
r3_input1_reg[2]/D f	-0.196	VIOLATED
r3_input1_reg[1]/D f	-0.196	VIOLATED
r3_input1_reg[0]/D f	-0.196	VIOLATED
r2_input1_reg[17]/D f	-0.196	VIOLATED
r2_input1_reg[16]/D f	-0.196	VIOLATED
r2_input1_reg[15]/D f	-0.196	VIOLATED
r2_input1_reg[14]/D f	-0.196	VIOLATED
r2_input1_reg[13]/D f	-0.196	VIOLATED
r2_input1_reg[12]/D f	-0.196	VIOLATED
r2_input1_reg[11]/D f	-0.196	VIOLATED
r2_input1_reg[10]/D f	-0.196	VIOLATED
r2_input1_reg[9]/D f	-0.196	VIOLATED
r2_input1_reg[8]/D f	-0.196	VIOLATED
r2_input1_reg[7]/D f	-0.196	VIOLATED
r2_input1_reg[6]/D f	-0.196	VIOLATED
r2_input1_reg[5]/D f	-0.196	VIOLATED
r2_input1_reg[4]/D f	-0.196	VIOLATED
r2_input1_reg[3]/D f	-0.196	VIOLATED
r2_input1_reg[2]/D f	-0.196	VIOLATED

```
Check type : clock_period
-----
| No paths found
|
Check type : skew
-----
| No paths found

Check type : pulse_width
-----
| No violating Checks with given description found

Check type : max_transition
-----
| No Violations found

Check type : min_transition
-----
| No Violations found

Check type : max_capacitance
-----
| No Violations found

Check type : min_capacitance
-----
| No Violations found

Check type : max_fanout
-----
| No Violations found

Check type : min_fanout
-----
| No Violations found
```

Timing report:

File Edit View Search Tools Documents Help

Open Save Undo Redo Cut Copy Paste Find Replace

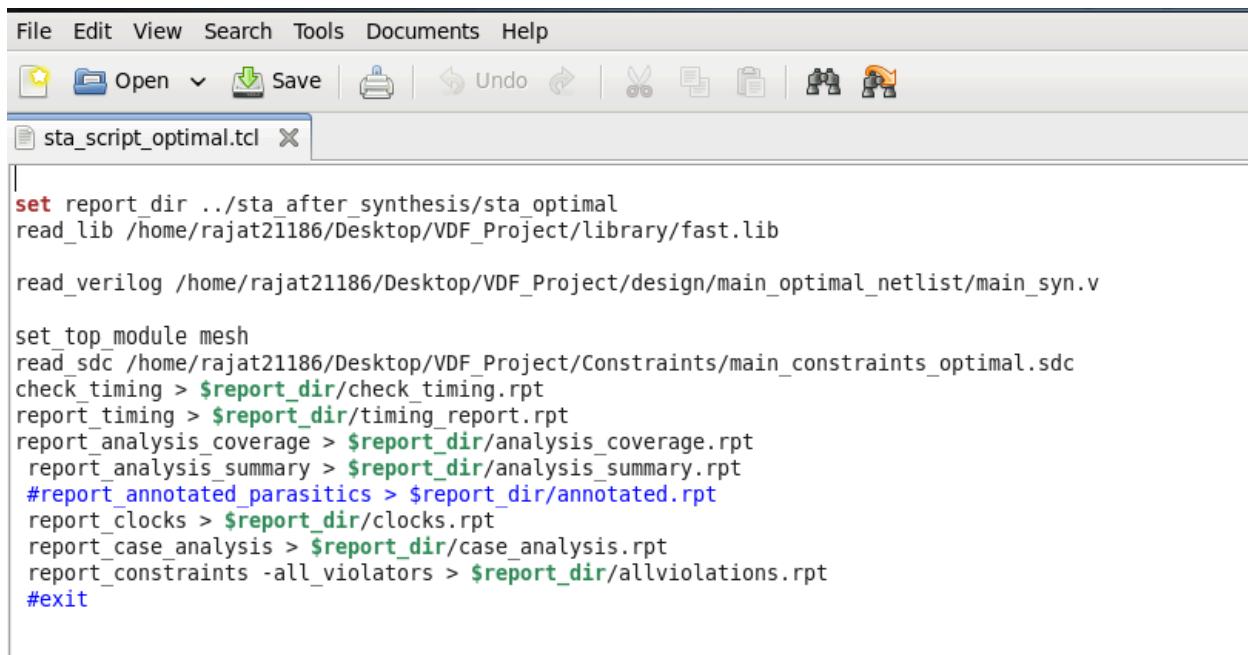
timing_report.rpt X

```
#####
# Generated by: Cadence Tempus 20.10-p003_1
# OS: Linux x86_64(Host ID edaserver4)
# Generated on: Sat Mar 23 02:07:40 2024
# Design: mesh
# Command: report_timing > $report_dir/timing_report.rpt
#####
Path 1: MET Setup Check with Pin m0/R3_control_signals2_reg[16]/CK
Endpoint: m0/R3_control_signals2_reg[16]/D (v) checked with leading edge of 'clk'
Beginpoint: reset (v) triggered by leading edge of 'clk'
Path Groups: {clk}
Other End Arrival Time 0.010
- Setup 0.089
+ Phase Shift 2.550
- Uncertainty 0.010
= Required Time 2.461
- Arrival Time 1.916
= Slack Time 0.545
Clock Rise Edge 0.000
+ Input Delay 0.040
+ Network Insertion Delay 0.010
= Beginpoint Arrival Time 0.050
-----
Instance Arc Cell Delay Arrival Required
Time Time
-----
- reset v - - 0.050 0.595
g2183 A v -> Y ^ CLKINVX1 0.643 0.693 1.238
g2131 A ^ -> Y ^ AND2X1 0.062 0.755 1.300
m0/g20313 B ^ -> Y v NAND2BX1 0.051 0.806 1.351
m0/g20263 B v -> Y ^ NOR2XL 0.077 0.883 1.428
m0/g20233 B ^ -> Y v NAND2BX1 0.018 0.901 1.446
m0/g20227 B0 v -> Y ^ OAI2BB1X1 0.027 0.928 1.473
m0/g20196 C0 ^ -> Y v AOI211X1 0.012 0.940 1.486
m0/g20191 A v -> Y ^ INVX1 0.022 0.962 1.508
m0/g20185 AN ^ -> Y ^ NOR2BX1 0.035 0.998 1.543
m0/g20183 B0 ^ -> Y v AOI211X1 0.031 1.029 1.574
m0/g20182 A v -> Y ^ CLKINVX1 0.053 1.082 1.627
m0/g20178 B ^ -> Y v NOR2XL 0.044 1.126 1.671
m0/g20170 B v -> Y ^ NAND3BXL 0.037 1.163 1.708
m0/g20166 B ^ -> Y v NAND2BX1 0.020 1.183 1.728
m0/g20162 A1 v -> Y v A022XL 0.049 1.232 1.777
m0/g20157 B v -> Y ^ NOR4X1 0.213 1.445 1.990
m0/g20156 A ^ -> Y v CLKINVX1 0.026 1.471 2.016
m0/g20148 A1 v -> Y ^ AOI211X1 0.054 1.524 2.070
m0/g20137 A1 ^ -> Y v AOI22X1 0.025 1.550 2.095
m0/g20134 C0 v -> Y ^ AOI211X1 0.022 1.572 2.117
m0/g20133 A ^ -> Y v CLKINVX1 0.010 1.582 2.127
m0/g20132 A1 v -> Y ^ AOI22X1 0.031 1.613 2.158
m0/g20131 C0 ^ -> Y v AOI211X1 0.030 1.642 2.188
m0/g20130 A v -> Y ^ CLKINVX1 0.053 1.695 2.240
m0/n20375 A ^ -> Y v NAND3X1 0.051 1.746 2.291
```

STA for Optimal Constraints

It can be seen that the start point is reset and the end point is taken at m0/R3_control_signals2_reg[10]/D which is the worst case for the problem statement. We have obtained a positive slack of 0.460.

TCL file:



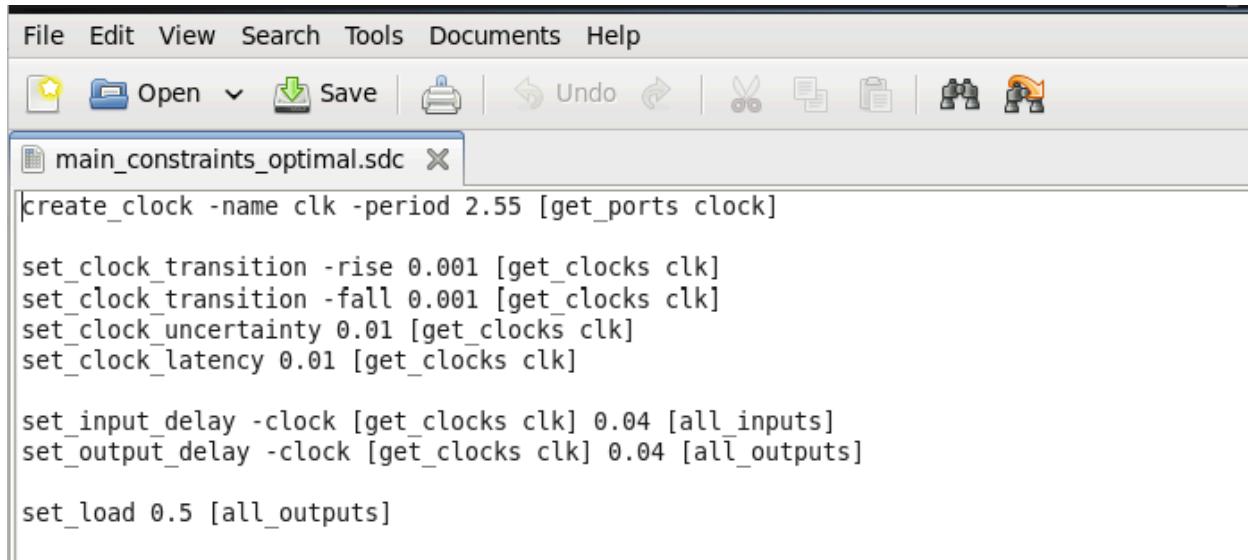
```
File Edit View Search Tools Documents Help
Open Save Undo Redo Cut Copy Paste Find Replace
sta_script_optimal.tcl X

set report_dir ..../sta_after_synthesis/sta_optimal
read_lib /home/rajat21186/Desktop/VDF_Project/library/fast.lib

read_verilog /home/rajat21186/Desktop/VDF_Project/design/main_optimal_netlist/main_syn.v

set_top_module mesh
read_sdc /home/rajat21186/Desktop/VDF_Project/Constraints/main_constraints_optimal.sdc
check_timing > $report_dir/check_timing.rpt
report_timing > $report_dir/timing_report.rpt
report_analysis_coverage > $report_dir/analysis_coverage.rpt
report_analysis_summary > $report_dir/analysis_summary.rpt
#report_annotated_parasitics > $report_dir/annotated.rpt
report_clocks > $report_dir/clocks.rpt
report_case_analysis > $report_dir/case_analysis.rpt
report_constraints -all_violators > $report_dir/allviolations.rpt
#exit
```

Constraints used:



```
File Edit View Search Tools Documents Help
Open Save Undo Redo Cut Copy Paste Find Replace
main_constraints_optimal.sdc X

create_clock -name clk -period 2.55 [get_ports clock]

set_clock_transition -rise 0.001 [get_clocks clk]
set_clock_transition -fall 0.001 [get_clocks clk]
set_clock_uncertainty 0.01 [get_clocks clk]
set_clock_latency 0.01 [get_clocks clk]

set_input_delay -clock [get_clocks clk] 0.04 [all_inputs]
set_output_delay -clock [get_clocks clk] 0.04 [all_outputs]

set_load 0.5 [all_outputs]
```

Violation report:

File Edit View Search Tools Documents Help

Open Save Undo |

allviolations.rpt X

```
#####
# Generated by: Cadence Tempus 20.10-p003_1
# OS: Linux x86_64(Host ID edaserver4)
# Generated on: Sat Mar 23 02:11:19 2024
# Design: mesh
# Command: report_constraints -all_violators > ../sta_after_synthesis/sta_optimal/allviolations.rpt
#####
# format : frame 0 : split 1

max_delay/setup
-----
No paths found

min_delay/hold
-----
-----
```

End Point	Slack	Cause
r3_input1_reg[17]/D f	-0.196	VIOLATED
r3_input1_reg[16]/D f	-0.196	VIOLATED
r3_input1_reg[15]/D f	-0.196	VIOLATED
r3_input1_reg[14]/D f	-0.196	VIOLATED
r3_input1_reg[13]/D f	-0.196	VIOLATED
r3_input1_reg[12]/D f	-0.196	VIOLATED
r3_input1_reg[11]/D f	-0.196	VIOLATED
r3_input1_reg[10]/D f	-0.196	VIOLATED
r3_input1_reg[9]/D f	-0.196	VIOLATED
r3_input1_reg[8]/D f	-0.196	VIOLATED
r3_input1_reg[7]/D f	-0.196	VIOLATED
r3_input1_reg[6]/D f	-0.196	VIOLATED
r3_input1_reg[5]/D f	-0.196	VIOLATED
r3_input1_reg[4]/D f	-0.196	VIOLATED
r3_input1_reg[3]/D f	-0.196	VIOLATED
r3_input1_reg[2]/D f	-0.196	VIOLATED
r3_input1_reg[1]/D f	-0.196	VIOLATED
r3_input1_reg[0]/D f	-0.196	VIOLATED
r2_input1_reg[17]/D f	-0.196	VIOLATED
r2_input1_reg[16]/D f	-0.196	VIOLATED
r2_input1_reg[15]/D f	-0.196	VIOLATED
r2_input1_reg[14]/D f	-0.196	VIOLATED
r2_input1_reg[13]/D f	-0.196	VIOLATED
r2_input1_reg[12]/D f	-0.196	VIOLATED
r2_input1_reg[11]/D f	-0.196	VIOLATED
r2_input1_reg[10]/D f	-0.196	VIOLATED
r2_input1_reg[9]/D f	-0.196	VIOLATED
r2_input1_reg[8]/D f	-0.196	VIOLATED
r2_input1_reg[7]/D f	-0.196	VIOLATED
r2_input1_reg[6]/D f	-0.196	VIOLATED
r2_input1_reg[5]/D f	-0.196	VIOLATED
r2_input1_reg[4]/D f	-0.196	VIOLATED
r2_input1_reg[3]/D f	-0.196	VIOLATED
r2_input1_reg[2]/D f	-0.196	VIOLATED

```
-----  
Check type : clock_period  
-----  
No paths found  
  
Check type : skew  
-----  
No paths found  
  
Check type : pulse_width  
-----  
No violating Checks with given description found  
  
Check type : max_transition  
-----  
No Violations found  
  
Check type : min_transition  
-----  
No Violations found  
  
Check type : max_capacitance  
-----  
No Violations found  
  
Check type : min_capacitance  
-----  
No Violations found  
  
Check type : max_fanout  
-----  
No Violations found  
  
Check type : min_fanout  
-----  
No Violations found
```

Timing report:

File Edit View Search Tools Documents Help

Open Save Undo

timing_report.rpt X

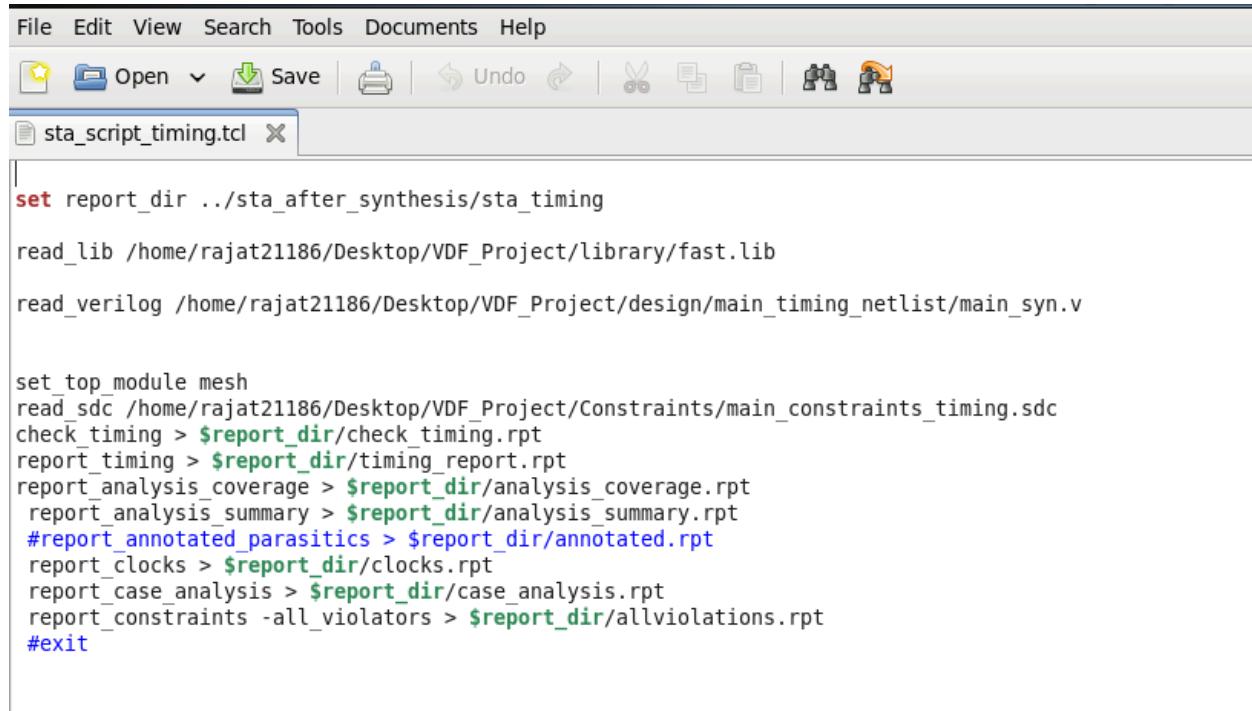
```
#####
# Generated by: Cadence Tempus 20.10-p003_1
# OS: Linux x86_64(Host ID edaserver4)
# Generated on: Sat Mar 23 02:11:19 2024
# Design: mesh
# Command: report_timing > $report_dir/timing_report.rpt
#####
Path 1: MET Setup Check with Pin m0/R3_control_signals2_reg[10]/CK
Endpoint: m0/R3_control_signals2_reg[10]/D (^) checked with leading edge of 'clk'
Beginpoint: reset (v) triggered by leading edge of 'clk'
Path Groups: {clk}
Other End Arrival Time 0.010
- Setup 0.090
+ Phase Shift 2.550
- Uncertainty 0.010
= Required Time 2.460
- Arrival Time 2.000
= Slack Time 0.460
    Clock Rise Edge 0.000
    + Input Delay 0.040
    + Network Insertion Delay 0.010
    = Beginpoint Arrival Time 0.050
-----
Instance Arc Cell Delay Arrival Required
Time Time
-----
- reset v - 0.050 0.510
g2407 A v -> Y ^ CLKINVX1 0.647 0.697 1.157
g2363 A ^ -> Y ^ AND2X1 0.060 0.756 1.216
m0/g17025 AN ^ -> Y ^ NAND2BX1 0.055 0.811 1.271
m0/g16996 B ^ -> Y v NOR2XL 0.024 0.836 1.295
m0/g16984 A v -> Y ^ INVX1 0.024 0.860 1.320
m0/g16920 A0 ^ -> Y v OAI211X1 0.035 0.894 1.354
m0/g16906 B0 v -> Y ^ AOI21X1 0.046 0.941 1.400
m0/g16899 B1 ^ -> Y v OAI22XL 0.022 0.963 1.422
m0/g16897 B v -> Y ^ NOR2XL 0.137 1.099 1.559
m0/g16896 A ^ -> Y v CLKINVX1 0.044 1.143 1.603
m0/g16893 B v -> Y ^ NOR2XL 0.110 1.254 1.713
m0/g16882 A1 ^ -> Y v AOI21X1 0.031 1.285 1.745
m0/g16877 A1 v -> Y ^ OAI22X1 0.038 1.322 1.782
m0/g16875 A ^ -> Y v NAND2XL 0.022 1.344 1.804
m0/g16874 C0 v -> Y ^ OAI211X1 0.018 1.362 1.821
m0/g16873 B ^ -> Y ^ OR2X1 0.081 1.443 1.903
m0/g16872 A ^ -> Y v CLKINVX1 0.033 1.476 1.935
m0/g16867 B1 v -> Y ^ AOI221X1 0.057 1.533 1.992
m0/g16862 A ^ -> Y v CLKINVX1 0.010 1.542 2.002
m0/g16852 A1 v -> Y ^ AOI22X1 0.030 1.572 2.032
m0/g16848 C ^ -> Y ^ AND3X1 0.045 1.616 2.076
m0/g16847 C0 ^ -> Y v OAI211X1 0.059 1.676 2.136
m0/g16846 A v -> Y ^ CLKINVX1 0.068 1.744 2.204
m0/g14724 A ^ -> Y v NAND2XL 0.064 1.808 2.268
m0/g14692 R v -> Y ^ NOR2XI 0.099 1.907 2.367
```

The above shown are the critical paths in our design and the tool has calculated the RT and AT using the delays which we goto from the fast.lib library.

STA for Best Timing

It can seen that the start point is reset and the end point is taken at m0/R0_control_signals2_reg[2]/RN which is the worst case for the problem statement. We have obtained a positive slack of 1.422.

TCL file:

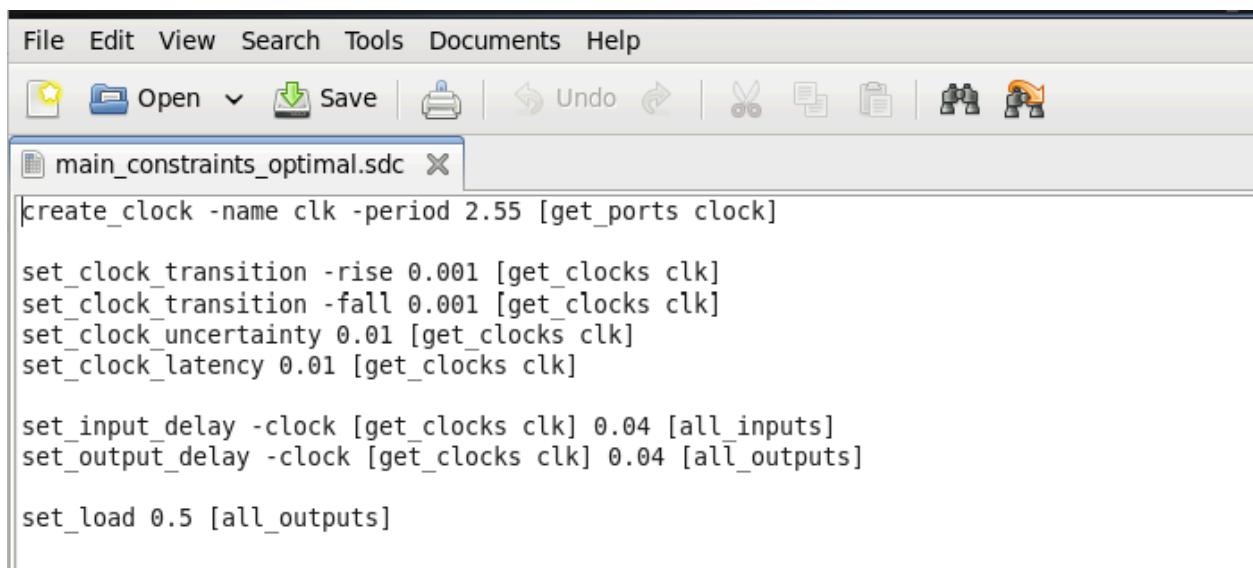


The screenshot shows a software interface for editing a TCL script. The menu bar includes File, Edit, View, Search, Tools, Documents, and Help. The toolbar contains icons for Open, Save, Undo, Redo, Cut, Copy, Paste, Find, and Replace. The main window displays the contents of the file 'sta_script_timing.tcl'. The code in the file is as follows:

```
set report_dir ..../sta_after_synthesis/sta_timing
read_lib /home/rajat21186/Desktop/VDF_Project/library/fast.lib
read_verilog /home/rajat21186/Desktop/VDF_Project/design/main_timing_netlist/main_syn.v

set_top_module mesh
read_sdc /home/rajat21186/Desktop/VDF_Project/Constraints/main_constraints_timing.sdc
check_timing > $report_dir/check_timing.rpt
report_timing > $report_dir/timing_report.rpt
report_analysis_coverage > $report_dir/analysis_coverage.rpt
report_analysis_summary > $report_dir/analysis_summary.rpt
#report_annotated_parasitics > $report_dir/annotated.rpt
report_clocks > $report_dir/clocks.rpt
report_case_analysis > $report_dir/case_analysis.rpt
report_constraints -all_violators > $report_dir/allviolations.rpt
#exit
```

Constraints used:



The screenshot shows a software application window with a menu bar (File, Edit, View, Search, Tools, Documents, Help) and a toolbar with various icons. A tab labeled "main_constraints_optimal.sdc" is selected. The main area contains the following Xilinx SDC code:

```
create_clock -name clk -period 2.55 [get_ports clock]

set_clock_transition -rise 0.001 [get_clocks clk]
set_clock_transition -fall 0.001 [get_clocks clk]
set_clock_uncertainty 0.01 [get_clocks clk]
set_clock_latency 0.01 [get_clocks clk]

set_input_delay -clock [get_clocks clk] 0.04 [all_inputs]
set_output_delay -clock [get_clocks clk] 0.04 [all_outputs]

set_load 0.5 [all_outputs]
```

Violation report:

File Edit View Search Tools Documents Help

Open Save Undo | Scissors Copy Paste Find Replace

allviolations.rpt X

```
#####
# Generated by: Cadence Tempus 20.10-p003_1
# OS: Linux x86_64(Host ID edaserver4)
# Generated on: Sat Mar 23 02:09:36 2024
# Design: mesh
# Command: report_constraints -all violators > ../sta_after_synthesis/sta_timing/allviolations.rpt
#####
# format : frame 0 : split 1

max_delay/setup
-----
No paths found

min_delay/hold
-----
-----
```

End Point	Slack	Cause
r3_input1_reg[17]/D f	-0.196	VIOLATED
r3_input1_reg[16]/D f	-0.196	VIOLATED
r3_input1_reg[15]/D f	-0.196	VIOLATED
r3_input1_reg[14]/D f	-0.196	VIOLATED
r3_input1_reg[13]/D f	-0.196	VIOLATED
r3_input1_reg[12]/D f	-0.196	VIOLATED
r3_input1_reg[11]/D f	-0.196	VIOLATED
r3_input1_reg[10]/D f	-0.196	VIOLATED
r3_input1_reg[9]/D f	-0.196	VIOLATED
r3_input1_reg[8]/D f	-0.196	VIOLATED
r3_input1_reg[7]/D f	-0.196	VIOLATED
r3_input1_reg[6]/D f	-0.196	VIOLATED
r3_input1_reg[5]/D f	-0.196	VIOLATED
r3_input1_reg[4]/D f	-0.196	VIOLATED
r3_input1_reg[3]/D f	-0.196	VIOLATED
r3_input1_reg[2]/D f	-0.196	VIOLATED
r3_input1_reg[1]/D f	-0.196	VIOLATED
r3_input1_reg[0]/D f	-0.196	VIOLATED
r2_input1_reg[17]/D f	-0.196	VIOLATED
r2_input1_reg[16]/D f	-0.196	VIOLATED
r2_input1_reg[15]/D f	-0.196	VIOLATED
r2_input1_reg[14]/D f	-0.196	VIOLATED
r2_input1_reg[13]/D f	-0.196	VIOLATED
r2_input1_reg[12]/D f	-0.196	VIOLATED
r2_input1_reg[11]/D f	-0.196	VIOLATED
r2_input1_reg[10]/D f	-0.196	VIOLATED
r2_input1_reg[9]/D f	-0.196	VIOLATED
r2_input1_reg[8]/D f	-0.196	VIOLATED
r2_input1_reg[7]/D f	-0.196	VIOLATED
r2_input1_reg[6]/D f	-0.196	VIOLATED
r2_input1_reg[5]/D f	-0.196	VIOLATED
r2_input1_reg[4]/D f	-0.196	VIOLATED
r2_input1_reg[3]/D f	-0.196	VIOLATED

```
-----  
Check type : clock_period  
-----  
No paths found  
  
Check type : skew  
-----  
No paths found  
  
Check type : pulse_width  
-----  
No violating Checks with given description found  
  
Check type : max_transition  
-----  
No Violations found  
  
Check type : min_transition  
-----  
No Violations found  
  
Check type : max_capacitance  
-----  
No Violations found  
  
Check type : min_capacitance  
-----  
No Violations found  
  
Check type : max_fanout  
-----  
No Violations found  
  
Check type : min_fanout  
-----  
No Violations found
```

Timing report:

```

#####
# Generated by: Cadence Tempus 20.10-p003_1
# OS: Linux x86_64(Host ID edaserver4)
# Generated on: Sat Mar 23 02:09:36 2024
# Design: mesh
# Command: report_timing > $report_dir/timing_report.rpt
#####
Path 1: MET Recovery Check with Pin m0/R0_control_signals2_reg[2]/CK
Endpoint: m0/R0_control_signals2_reg[2]/RN (^) checked with leading edge of 'clk'
Beginpoint: reset (v) triggered by leading edge of 'clk'
Path Groups: {async_default}
Other End Arrival Time      0.010
- Recovery                  0.394
+ Phase Shift                2.550
- Uncertainty                0.010
= Required Time              2.156
- Arrival Time               0.734
= Slack Time                 1.422
    Clock Rise Edge          0.000
    + Input Delay             0.040
    + Network Insertion Delay 0.010
    = Beginpoint Arrival Time 0.050
-----
Instance                      Arc      Cell     Delay   Arrival  Required
                           Time      Time
-----
-                         reset v - - 0.050  1.472
m0/g15836                  A v -> Y ^ INVXL 0.684  0.734  2.156
m0/R0_control_signals2_reg[2] RN ^ DFFRHQX1 0.000  0.734  2.156
-----
```

The above shown are the critical paths in our design and the tool has calculated the RT and AT using the delays which we goto from the fast.lib library. In this case we have got the least number of critical paths because we used less time period and therefore the cells used while synthesis were having less delay and so the critical paths are less.

Path Based Analysis

	Required Time (in ns)	Arrival Time (in ns)	Slack (in ns)
MIN. AREA	2.461	1.916	0.545

OPTIMAL	2.460	2.000	0.460
BEST TIMING	2.156	0.734	1.422

1. Required Time is the difference of clock period and setup time
2. Arrival Time is the time data takes to reach the input pin of the flip flop
3. Slack is the difference of required time and arrival time.
As our slack is positive, we have no setup up violations
4. Critical path is the path with the worst slack, which is shown by the tool.
5. In all three cases, we utilized the optimal constraint file to strike a balance between area and speed. Additionally to ensure consistency across evaluations.

STA : ANALYSIS

1. For the Minimum area netlist which is a highly relaxed netlist in terms of timing we observe the slack is 0.545.
2. Setup time in this case is 0.089 and uncertainty is 0.010. So the required time (RT) is phase shift time(clock period) - setup time - constraint time => $2.550 - 0.089 - 0.010 \Rightarrow 2.461$ ns and Arrival time (AT) is 1.916 and slack is defined as $RT - AT \Rightarrow 0.545$ ns . Positive value of the slack implies that we can increase the value of arrival time by slack value without affecting the overall delay of circuit.
3. For the best timing netlist we got the maximum slack which is very evident because netlist of the best timing case has the tightest constraint of 2.55ns. So the $2.55 - 0.394(\text{recovery}) - 0.010(\text{uncertainty}) = 2.156$ ns

Arrival Time = 0.734ns

Slack = RT - AT = 1.422ns

4. For the optimal case $RT = 2.55 - 0.09(\text{setup}) - 0.010(\text{uncertainty}) = 2.460$.
 $AT = 2.000$
 $\text{Slack} = RT - AT = 0.460$ ns

6. Scan Insertion

Inputs: Liberty file, tcl file, constraints file (.sdc), Synthesized netlist

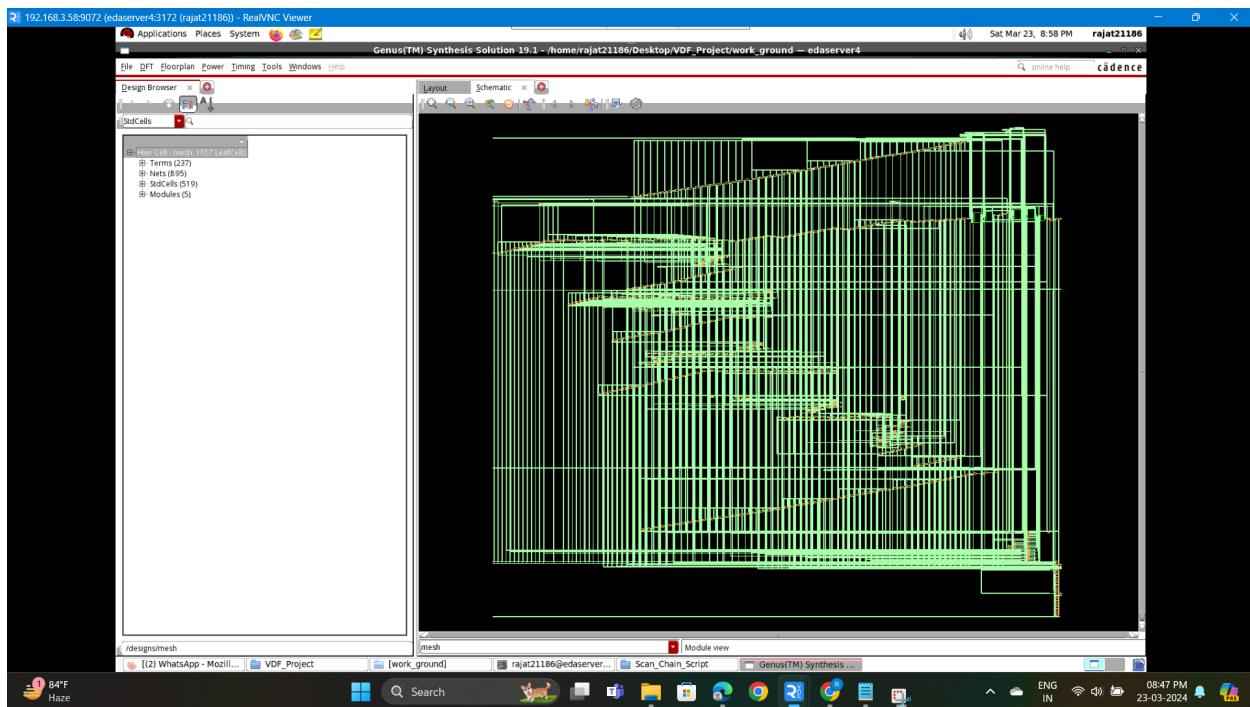
Output: Cell Report, Area Report, Power Report, Timing Report

Software Used: Cadence

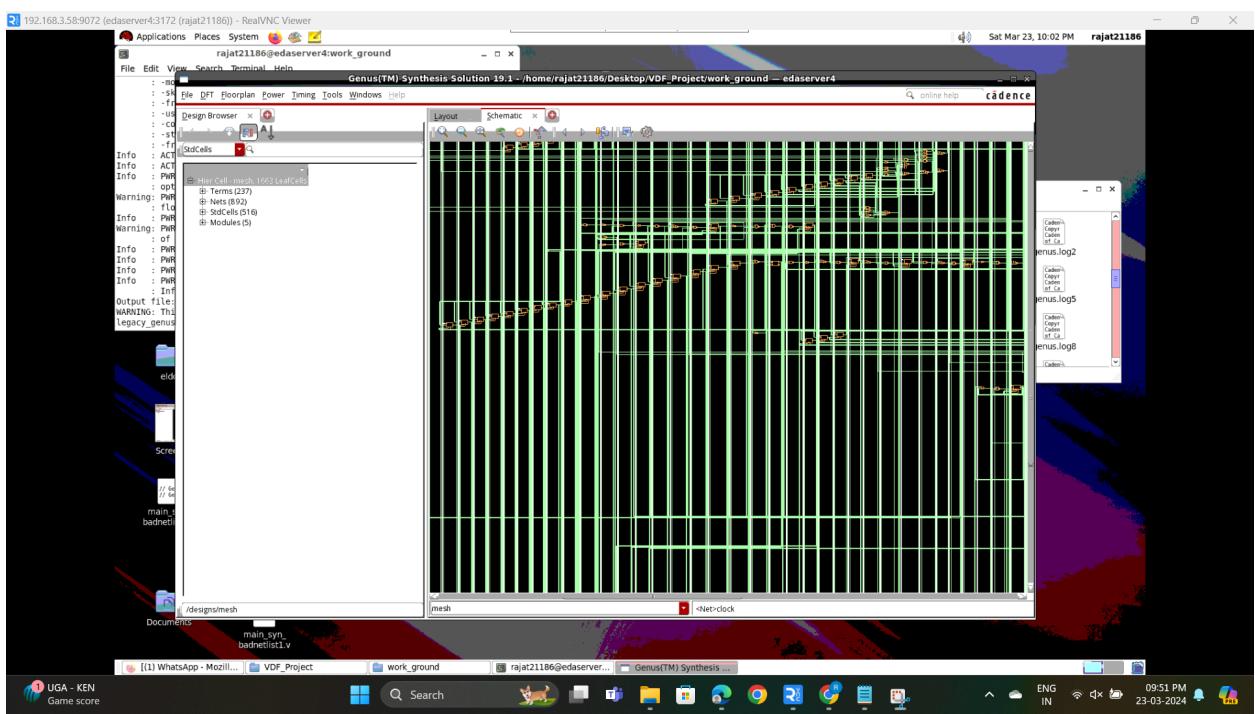
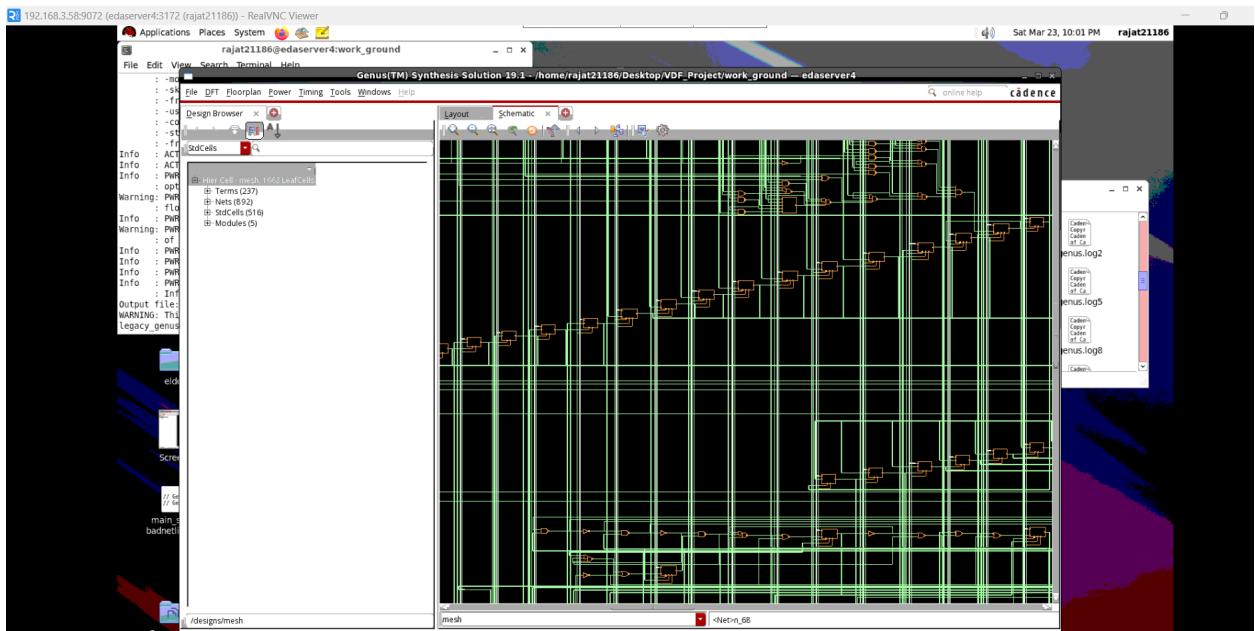
Commands Used:

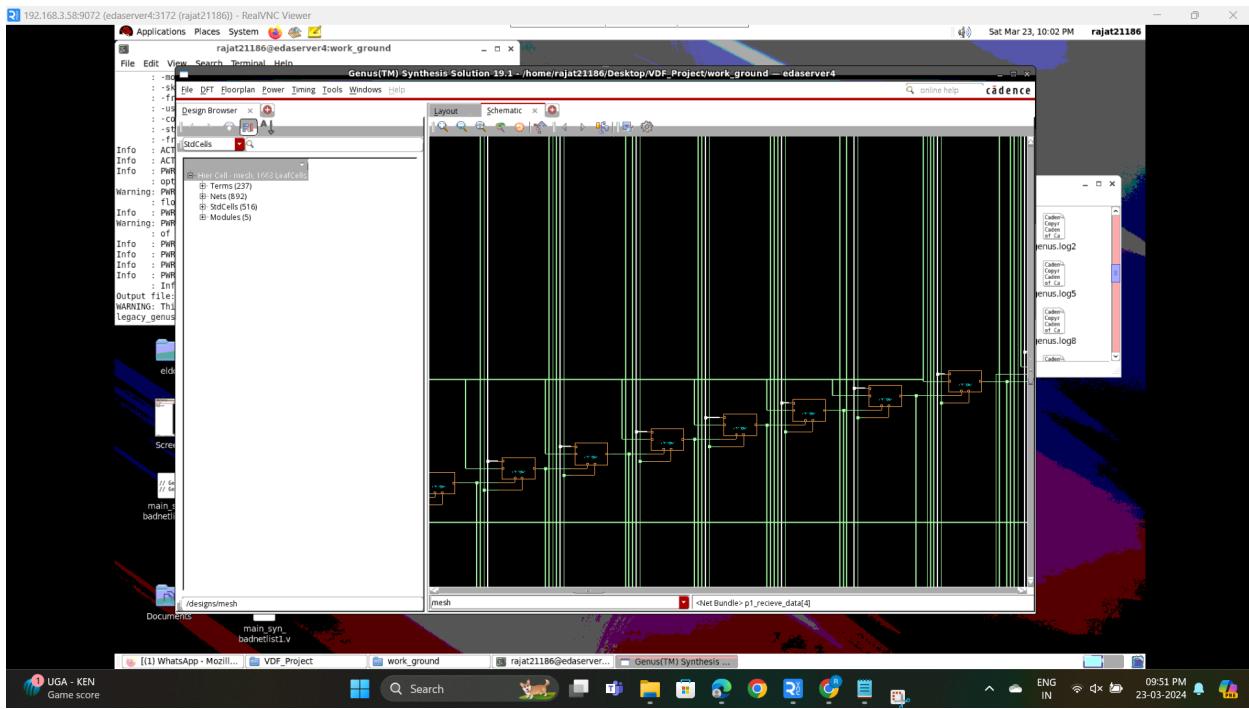
1. csh
2. source /cadence/cshrc
3. Genus -legacy_ui

Schematic Design



Some of the changes in design





Scan Insertion for Minimum Area

```

drt_script_area.tcl  dit_script_optimal.tcl  dit_script_timing.tcl
set_attr lib_search_path /home/rajat21186/Desktop/VDF_Project/library
set_attr hdl_search_path /home/rajat21186/Desktop/VDF_Project/RTL_Code/mesh
set_attr library fast.lib
read_hdl /home/rajat21186/Desktop/VDF_Project/RTL_Code/main.v
elaborate
read_sdc /home/rajat21186/Desktop/VDF_Project/Constraints/main_constraints_area.sdc
report_timing -lint
set_attribute dft_scan_style muxed_scan
define_dft shift_enable -active high -create_port scan_en
define_dft test_mode -active high -create_port test_mode
define_dft test_clock clock
report dft_setup
check_dft_rules >dft_report/dft_rules_report
fix_dft_violations -test_control test_mode -async_set -async_reset -clock
synthesize -to_mapped
set_attr dft_min_number_of_scan_chains 2 mesh
set_attr dft_mix_clock_edges_in_scan_chains true mesh
connect_scan_chains -auto_create_chains -preview
connect_scan_chains -auto_create_chains
report qor
write_atpg -cadence > ../DFT_Results/main_min_area/main/test.atpg
write_atpg_stil > ../DFT_Results/main_min_area/main/still.atpg
write_scandef > ../DFT_Results/main_min_area/main.d_ef
write_sdf -timescale ns -nonechecks -recrern split -edges check_edge > ../DFT_Results/main_min_area/main_delays_optimal.sdf
write_hdl > ../DFT_Results/main_min_area/synthesised/netlist.v
write_sdc > ../DFT_Results/main_min_area/sdc_file_for_physical_design.sdc
write_script > ../DFT_Results/main_min_area/synthesis_script_sdc.g
report_timing > ../DFT_Results/main_min_area/synthesis_timing_report.rep
report_power > ../DFT_Results/main_min_area/synthesis_power_report.rep
report_gates > ../DFT_Results/main_min_area/synthesis_cell_report.rep
report_area > ../DFT_Results/main_min_area/synthesis_area_report.rep
#exit
gui_show

```

- Firstly, all the RTL files along with the library are read and elaborated. DFT_scan_style, offers two design options: Muxed scan style and Clocked Level-Sensitive Scanned

Design (LSSD) scan style. This command dictates the preferred design style for DFT. In our project, we employ the muxed scan design style.

- Next, `define_dft shift_enable -active high -create_port scan_en` is used to establish three modes for scan design: normal, shift, and capture. In the shift mode, our scan cells function as shift registers, facilitated by the `scan_en` port specified within this command, alongside the creation of test mode and clock ports.
- Following this, the `report dft_setup` command offers a comprehensive overview of the entire DFT environment setup, capturing both user-defined choices and tool-inferred configurations.
- Subsequently, `fix_dft_violations` identifies and resolves any DFT-related violations, adhering to specified rules, while requiring manual intervention for user-defined issues.
- `Synthesize -to_mapped` further enhances our design implementation by synthesizing it to meet constraints, leveraging cells from the library and conducting logic optimizations.
- `connect_scan_chains -auto_create_chains` –preview efficiently configures and interconnects scan cells into chains, with the flexibility to generate additional chains if needed, and provides a preview of the connection report.
- Finally, `write_atpg` generates appropriate test patterns by utilizing scan chain information to assist the ATPG tool.

Constraints

Same as used in synthesis section of minimum area

Area Report

===== Generated by: Genus(TM) Synthesis Solution 19.13-s073_1 Generated on: Mar 23 2024 08:57:08 pm Module: mesh Technology library: fast Operating conditions: fast (balanced_tree) Wireload mode: enclosed Area mode: timing library =====						
Instance	Module	Cell Count	Cell Area	Net Area	Total Area	Wireload
mesh		1657	16512.530	0.000	16512.530	<none> (D)
m0	master	436	2829.292	0.000	2829.292	<none> (D)
r2	router_354	194	1699.997	0.000	1699.997	<none> (D)
r0	router	178	1621.280	0.000	1621.280	<none> (D)
r3	router_353	178	1599.330	0.000	1599.330	<none> (D)
r1	router_355	152	1474.441	0.000	1474.441	<none> (D)

(D) = wireload is default in technology library

Timing Report

```
=====
Generated by:      Genus(TM) Synthesis Solution 19.13-s073_1
Generated on:     Mar 23 2024  08:57:07 pm
Module:          mesh
Technology library: fast
Operating conditions: fast (balanced_tree)
Wireload mode:    enclosed
Area mode:       timing library
=====
```

Pin	Type	Fanout	Load	Slew	Delay	Arrival
		(fF)	(ps)	(ps)	(ps)	
(clock clk)	launch latency				0	R
p0_configure1_reg[0]/CK				+10	10	R
p0_configure1_reg[0]/Q	SDFFQX1	2	3.6	16	+78	88 R
j2148/B				+0	88	
j2148/Y	NAND2XL	2	4.7	29	+25	114 F
j2135/B				+0	114	
j2135/Y	NOR2XL	13	35.5	314	+230	343 R
m0/P0_signals[0]						
g14921/A				+0	343	
g14921/Y	CLKINVX1	10	20.0	114	+45	388 F
g20263/A				+0	388	
g20263/Y	NOR2XL	3	8.6	88	+88	477 R
g20233/B				+0	477	
g20233/Y	NAND2BX1	1	2.9	32	+17	494 F
g20227/B0				+0	494	
g20227/Y	OAI2BB1X1	2	5.9	26	+27	521 R
g20196/C0				+0	521	
g20196/Y	AOI211X1	1	2.9	31	+12	534 F
g20191/A				+0	534	
g20191/Y	INVX1	2	4.6	20	+22	556 R
g20185/AN				+0	556	
g20185/Y	NOR2BX1	1	2.9	26	+35	591 R
g20183/B0				+0	591	
g20183/Y	AOI211X1	7	17.0	65	+33	623 F
g20182/A				+0	623	
g20182/Y	CLKINVX1	6	15.7	52	+54	677 R
g20178/B				+0	677	
g20178/Y	NOR2XL	5	13.3	62	+44	721 F
g20170/B				+0	721	
g20170/Y	NAND3BXL	1	3.0	34	+36	757 R
g20166/B				+0	757	
g20166/Y	NAND2BX1	2	4.7	24	+20	777 F
g20162/A1				+0	777	
g20162/Y	A022XL	1	2.9	19	+49	826 F
g20157/B				+0	826	
g20157/Y	NOR4X1	11	30.0	280	+212	1038 R

192.168.3.58:9072 (edaserver4:3172) (rajat21186) - RealVNC Viewer

Applications Places System

File Edit View Search Tools Documents Help

synthesis_timing_report.rep (~/Desktop/VDF_Project/DFT_Results/main_min_area) - gedit

File Open Save Undo

synthesis_timing_report.rep

g20182/A				+0	623
g20178/Y	CLKINVX1	6	15.7	52	+0 677 R
g20178/B				+0	677
g20178/Y	NOR2XL	5	13.3	62	+44 721 F
g20178/B				+0	721
g20166/Y	NAND3XL	1	3.0	34	+0 757 R
g20166/B				+0	757
g20166/Y	NAND2BX1	2	4.7	24	+20 777 F
g20166/B				+0	777
g20157/A	A022XL	1	2.9	19	+49 826 F
g20157/Y				+0	826
g20157/Y	NOR4X1	11	30.0	280	+212 1039 R
g20156/A				+0	1039
g20156/Y	CLKINVX1	5	13.7	92	+28 1066 F
g20148/A1				+0	1066
g20148/Y	OAI221X1	2	4.8	49	+54 1120 R
g20148/A1				+0	1120
g20137/Y	OAI222X1	1	3.0	31	+25 1145 F
g20134/C0				+0	1145
g20134/Y	OAI2211X1	1	2.9	46	+22 1167 R
g20134/A				+0	1167
g20133/Y	CLKINVX1	1	2.9	17	+10 1176 F
g20132/A1				+0	1176
g20132/Y	OAI222X1	1	3.0	48	+31 1207 R
g20132/C0				+0	1207
g20131/Y	OAI2211X1	6	12.7	55	+29 1237 F
g20130/Y				+0	1237
g20130/Y	CLKINVX1	8	21.2	63	+61 1296 R
g20130/A				+0	1296
g20375/Y	NAND3X1	6	12.8	63	+51 1349 F
g14811/B				+0	1349
g14811/Y	OR2X1	3	7.8	24	+61 1410 R
g14811/B				+0	1410
g14811/Y	OR2XL	1	1.7	13	+34 1444 F
g14778/B1				+0	1444
g14778/Y	A01222XL	1	1.7	62	+50 1499 R
g14754/Y				+0	1499
g14754/Y	NAND2XL	1	2.9	29	+22 1521 F
R3_control_signals2_reg[16]/D	SDFFRHQX1			+0	1521
R3_control_signals2_reg[16]/CK	setup	1	+86	1668 R	
(clock clk)					
	capture				4320 R
	latency			+10	4330 R
	uncertainty			-10	4320 R
Cost Group : 'clk' (path group 'clk')					
Timing slack : 2772ps					
Start-point : p0 configure1 reg[8]/CK					
End-point : m0/f3 control_signals2_reg[16]/0					

main_min_area Plain Text Tab Width: 8 v Ln 1, Col 1 INS

[2] WhatsApp - Mozilla... [] (work_group) [] (rajat21186@edaserv... []) VDF_Project [] DFT_Results [] main_min_area [] synthesis_timing_repo...

UGA - KEN Game score ENG IN 09:01 PM 23-03-2024

Power Report

Instance: /mesh

Power Unit: W

PDB Frames: /stim#0/frame#0

Category	Leakage	Internal	Switching	Total	Row%
memory	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
register	1.40790e-04	2.82315e-03	1.16349e-03	4.12743e-03	85.05%
latch	8.64628e-07	1.43564e-06	1.01752e-07	2.40202e-06	0.05%
logic	4.42326e-05	2.63958e-04	1.52184e-04	4.60375e-04	9.49%
bbox	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
clock	0.00000e+00	0.00000e+00	2.63007e-04	2.63007e-04	5.42%
pad	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
pm	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
Subtotal	1.85887e-04	3.08854e-03	1.57878e-03	4.85321e-03	100.01%
Percentage	3.83%	63.64%	32.53%	100.00%	100.00%

Scan Insertion for Optimal Constraints

```
set_attr lib_search_path /home/rajat21186/Desktop/VDF_Project/library
set_attr hdl_search_path /home/rajat21186/Desktop/VDF_Project/RTL_Code/mesh
set_attr library fast.lib
read_hdl /home/rajat21186/Desktop/VDF_Project/RTL_Code/main.v
elaborate
read_sdc /home/rajat21186/Desktop/VDF_Project/Constraints/main_constraints_optimal.sdc
report timing -lint
set_attribute dft_scan_style muxed_scan
define_dft shift_enable -active high -create_port scan_en
define_dft test_mode -active high -create_port test_mode
define_dft test_clock clock
report dft_setup
check_dft_rules >dft_report/dft_rules_report
fix_dft_violations -test_control test_mode -async_set -async_reset -clock

synthesize -to_mapped
set_attr dft_min_number_of_scan_chains 2 mesh
set_attr dft_mix_clock_edges_in_scan_chains true mesh
connect_scan_chains -auto_create_chains -preview
connect_scan_chains -auto_create_chains
report qor
write_atpg -cadence > ../DFT_Results/main_optimal/main_test.atpg
write_atpg -stil > ../DFT_Results/main_optimal/main_still.atpg
write_scandef > ../DFT_Results/main_optimal/main.ddef
write_sdf -timescale ns -nonegchecks -recrun split -edges check_edge > ../DFT_Results/main_optimal/main_delays_optimal.sdf
write_hdl > ../DFT_Results/main_optimal/synthesised.netlist.v
write_sdc > ../DFT_Results/main_optimal/sdc_file_for_physical_design.sdc
write_script > ../DFT_Results/main_optimal/synthesis_script_sdc.g
report_timing > ../DFT_Results/main_optimal/synthesis_timing_report.rep
report_power > ../DFT_Results/main_optimal/synthesis_power_report.rep
report_gates > ../DFT_Results/main_optimal/synthesis_cell_report.rep
report_area > ../DFT_Results/main_optimal/synthesis_area_report.rep
#exit
gui_show
```

- Firstly, all the RTL files along with the library are read and elaborated. DFT_scan_style, offers two design options: Muxed scan style and Clocked Level-Sensitive Scanned Design (LSSD) scan style. This command dictates the preferred design style for DFT. In our project, we employ the muxed scan design style.
- Next, define_dft shift_enable -active high -create_port scan_en is used to establish three modes for scan design: normal, shift, and capture. In the shift mode, our scan cells function as shift registers, facilitated by the scan_en port specified within this command, alongside the creation of test mode and clock ports.
- Following this, the report dft_setup command offers a comprehensive overview of the entire DFT environment setup, capturing both user-defined choices and tool-inferred configurations.
- Subsequently, fix_dft_violations identifies and resolves any DFT-related violations, adhering to specified rules, while requiring manual intervention for user-defined issues.
- Synthesize -to_mapped further enhances our design implementation by synthesizing it to meet constraints, leveraging cells from the library and conducting logic optimizations.
- connect_scan_chains -auto_create_chains -preview efficiently configures and interconnects scan cells into chains, with the flexibility to generate additional chains if needed, and provides a preview of the connection report.
- Finally, write_atpg generates appropriate test patterns by utilizing scan chain information to assist the ATPG tool.

Constraints

Same as used in synthesis section of optimal case

Area Report

```
=====  
Generated by:          Genus(TM) Synthesis Solution 19.13-s073_1  
Generated on:         Mar 23 2024 09:01:46 pm  
Module:               mesh  
Technology library:   fast  
Operating conditions: fast (balanced_tree)  
Wireload mode:        enclosed  
Area mode:            timing library  
=====
```

Instance	Module	Cell Count	Cell Area	Net Area	Total Area	Wireload
mesh		1663	16504.204	0.000	16504.204	<none> (D)
m0	master	446	2830.049	0.000	2830.049	<none> (D)
r2	router_354	194	1699.997	0.000	1699.997	<none> (D)
r0	router	178	1621.280	0.000	1621.280	<none> (D)
r3	router_353	177	1603.114	0.000	1603.114	<none> (D)
r1	router_355	152	1474.441	0.000	1474.441	<none> (D)

(D) = wireload is default in technology library

Timing Report

Pin	Type	Fanout	Load	Slew	Delay	Arrival
		(fF)	(ps)	(ps)	(ps)	
(clock clk)	launch latency				0 R	
p0_configure1_reg[0]/CK				+10	10 R	
p0_configure1_reg[0]/Q	SDFFQX1	2 3.6	16	+78	88 R	
g1893/B				+0	88	
g1893/Y	NAND2XL	2 4.7	29	+25	114 F	
g1875/B				+0	114	
g1875/Y	NOR2XL	14 34.8	308	+225	339 R	
m0/P0 signals[0]						
g14752/A				+0	339	
g14752/Y	CLKINVX1	10 20.1	112	+46	385 F	
g17086/B				+0	385	
g17086/Y	NAND2XL	7 18.4	106	+104	488 R	
g17035/A				+0	488	
g17035/Y	CLKINVX1	5 11.4	47	+28	516 F	
g16924/C0				+0	516	
g16924/Y	AOI221X1	2 4.7	68	+54	571 R	
g16919/A				+0	571	
g16919/Y	INVX1	2 4.7	26	+14	584 F	
g16899/A0N				+0	584	
g16899/Y	OAI2BB1XL	1 1.7	19	+34	619 F	
g16898/A				+0	619	
g16898/Y	NOR2XL	8 21.4	192	+144	763 R	
g16897/A				+0	763	
g16897/Y	CLKINVX1	7 18.3	80	+42	806 F	
g16894/B				+0	806	
g16894/Y	NOR2XL	5 13.6	128	+111	917 R	
g16884/A2				+0	917	
g16884/Y	AOI32X1	2 6.0	75	+50	967 F	
g16876/A1				+0	967	
g16876/Y	OAI211X1	1 3.0	44	+47	1014 R	
g16875/C0				+0	1014	
g16875/Y	AOI211X1	7 17.9	65	+36	1050 F	
g16874/A				+0	1050	
g16874/Y	CLKINVX1	10 28.7	85	+79	1129 R	
g16873/B				+0	1129	
g16873/Y	NOR2RX1	7 14.0	46	+32	1161 F	

g16899/Y	OAI2BB1XL	1	1.7	19	+34	619	F	
g16898/A					+0	619		
g16898/Y	NOR2XL	8	21.4	192	+144	763	R	
g16897/A					+0	763		
g16897/Y	CLKINVX1	7	18.3	80	+42	806	F	
g16894/B					+0	806		
g16894/Y	NOR2XL	5	13.6	128	+111	917	R	
g16884/A2					+0	917		
g16884/Y	AOI32X1	2	6.0	75	+50	967	F	
g16876/A1					+0	967		
g16876/Y	OAI211X1	1	3.0	44	+47	1014	R	
g16875/C0					+0	1014		
g16875/Y	AOI211X1	7	17.9	65	+36	1050	F	
g16874/A					+0	1050		
g16874/Y	CLKINVX1	10	28.7	85	+79	1129	R	
g16873/B					+0	1129		
g16873/Y	NOR2BX1	7	14.0	46	+32	1161	F	
g16867/C0					+0	1161		
g16867/Y	AOI211X1	2	3.5	46	+45	1207	R	
g16863/A					+0	1207		
g16863/Y	INVXL	1	3.0	22	+15	1222	F	
g16853/B1					+0	1222		
g16853/Y	AOI22X1	1	3.0	37	+28	1249	R	
g16849/B0					+0	1249		
g16849/Y	OAI211X1	7	18.5	101	+70	1320	F	
g16848/A					+0	1320		
g16848/Y	CLKINVX1	10	18.5	66	+70	1389	R	
g14724/A					+0	1389		
g14724/Y	NAND2XL	5	13.7	126	+64	1453	F	
g14692/B					+0	1453		
g14692/Y	NOR2XL	3	8.7	94	+97	1550	R	
g14682/A					+0	1550		
g14682/Y	CLKINVX1	1	3.0	28	+8	1558	F	
g14662/A2					+0	1558		
g14662/Y	AOI32X1	2	4.6	67	+52	1611	R	
g14594/BN					+0	1611		
g14594/Y	NAND4BBXL	1	2.9	30	+37	1648	R	
R2_control_signals2_reg[15]/D <<<	SDFFRHQX1				+0	1648		
R2_control_signals2_reg[15]/CK	setup				1	+80	1728	R
(clock clk)								
	capture					2550	R	
	latency					+10	2560	R
	uncertainty					-10	2550	R

Cost Group : 'clk' (path_group 'clk')
Timing slack : 822ps
Start-point : p0_configure1_reg[0]/CK
End-point : m0/R2_control_signals2_reg[15]/D

Power Report

Category	Leakage	Internal	Switching	Total	Row%
memory	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
register	1.40790e-04	4.76742e-03	1.95772e-03	6.86593e-03	85.37%
latch	8.64628e-07	2.20476e-06	1.56093e-07	3.22548e-06	0.04%
logic	4.34342e-05	4.31708e-04	2.52815e-04	7.27957e-04	9.05%
bbox	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
clock	0.00000e+00	0.00000e+00	4.45565e-04	4.45565e-04	5.54%
pad	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
pm	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
Subtotal	1.85089e-04	5.20133e-03	2.65626e-03	8.04268e-03	100.00%
Percentage	2.30%	64.67%	33.03%	100.00%	100.00%

Scan Insertion for Best timing

```

set_attr lib_search_path /home/rajat21186/Desktop/VDF_Project/library
set_attr hdl_search_path /home/rajat21186/Desktop/VDF_Project/RTL_Code/mesh
set_attr library fast.lib
read_hdl /home/rajat21186/Desktop/VDF_Project/RTL_Code/main.v
elaborate
read_sdc /home/rajat21186/Desktop/VDF_Project/Constraints/main_constraints_timing.sdc
report timing -lint
set_attribute dft_scan_style muxed_scan
define_dft shift_enable -active high -create_port scan_en
define_dft test_mode -active high -create_port test_mode
define_dft test_clock clock
report_dft_setup
check_dft_rules >dft_report/dft_rules_report
fix_dft_violations -test_control test_mode -async_set -async_reset -clock

synthesize -to_mapped
set_attr dft_min_number_of_scan_chains 2 mesh
set_attr dft_mix_clock_edges_in_scan_chains true mesh
connect_scan_chains -auto_create_chains -preview
connect_scan_chains -auto_create_chains
report qor
write_atpg -cadence > ../DFT_Results/main_timing/main_test.atpg
write_atpg -stil > ../DFT_Results/main_timing/main_still.atpg
write_scandef > ../DFT_Results/main_timing/main.d.def
write_sdf -timescale ns -nonegchecks -recrm split -edges check_edge > ../DFT_Results/main_timing/main_delays_optimal.sdf
write_hdl > ../DFT_Results/main_timing/synthesised_netlist.v
write_sdc > ../DFT_Results/main_timing/sdc_file_for_physical_design.sdc
write_script > ../DFT_Results/main_timing/synthesis_script_sdc.g
report timing > ../DFT_Results/main_timing/synthesis_timing_report.rep
report power > ../DFT_Results/main_timing/synthesis_power_report.rep
report gates > ../DFT_Results/main_timing/synthesis_cell_report.rep
report area > ../DFT_Results/main_timing/synthesis_area_report.rep
#exit
gui_show

```

- Firstly, all the RTL files along with the library are read and elaborated. DFT_scan_style, offers two design options: Muxed scan style and Clocked Level-Sensitive Scanned Design (LSSD) scan style. This command dictates the preferred design style for DFT. In our project, we employ the muxed scan design style.

- Next, `define_dft shift_enable -active high -create_port scan_en` is used to establish three modes for scan design: normal, shift, and capture. In the shift mode, our scan cells function as shift registers, facilitated by the `scan_en` port specified within this command, alongside the creation of test mode and clock ports.
- Following this, the `report dft_setup` command offers a comprehensive overview of the entire DFT environment setup, capturing both user-defined choices and tool-inferred configurations.
- Subsequently, `fix_dft_violations` identifies and resolves any DFT-related violations, adhering to specified rules, while requiring manual intervention for user-defined issues.
- `Synthesize -to_mapped` further enhances our design implementation by synthesizing it to meet constraints, leveraging cells from the library and conducting logic optimizations.
- `connect_scan_chains -auto_create_chains -preview` efficiently configures and interconnects scan cells into chains, with the flexibility to generate additional chains if needed, and provides a preview of the connection report.
- Finally, `write_atpg` generates appropriate test patterns by utilizing scan chain information to assist the ATPG tool.

Constraints

Same as used in synthesis section of best timing

Area Report

```
=====
Generated by:          Genus(TM) Synthesis Solution 19.13-s073_1
Generated on:         Mar 23 2024 09:02:40 pm
Module:               mesh
Technology library:   fast
Operating conditions: fast (balanced_tree)
Wireload mode:        enclosed
Area mode:            timing library
=====
```

Instance	Module	Cell Count	Cell Area	Net Area	Total Area	Wireload
mesh		1821	17626.687	0.000	17626.687	<none> (D)
m0	master	578	3739.086	0.000	3739.086	<none> (D)
r2	router_354	186	1696.213	0.000	1696.213	<none> (D)
r0	router	178	1640.202	0.000	1640.202	<none> (D)
r3	router_353	180	1614.468	0.000	1614.468	<none> (D)
r1	router_355	152	1487.309	0.000	1487.309	<none> (D)

(D) = wireload is default in technology library

Timing Report

```
=====
Generated by:      Genus(TM) Synthesis Solution 19.13-s073_1
Generated on:      Mar 23 2024  09:02:39 pm
Module:            mesh
Technology library: fast
Operating conditions: fast (balanced_tree)
Wireload mode:     enclosed
Area mode:         timing library
=====
```

Pin	Type	Fanout	Load (fF)	Slew (ps)	Delay (ps)	Arrival (ps)
(clock clk)	launch latency				+10	0 R
r2						10 R
regPR_reg/CK					1	10 R
regPR_reg/Q					+111	121 F
r2/processor_ready						
g869/B					+0	121
g869/Y	NAND3X1	5	18.2	17	+24	144 R
fopt/A					+0	144
fopt/Y	CLKINVX2	4	8.3	14	+12	156 F
m0/path_free_bits[13]						
g15713_0/AN					+0	156
g15713_0/Y	NOR2BX1	1	3.0	13	+29	186 F
g15538/B					+0	186
g15538/Y	NAND2X1	1	3.0	14	+15	200 R
g15511/B0					+0	200
g15511/Y	OAI21X1	1	5.7	27	+22	222 F
g15481/B0					+0	222
g15481/Y	AOI21X2	2	7.4	30	+30	252 R
g15459/A1					+0	252
g15459/Y	OAI21X2	1	11.6	26	+23	275 F
g15454/B					+0	275
g15454/Y	NOR2X4	3	19.7	31	+30	306 R
g15438/A1					+0	306
g15438/Y	OAI21X4	6	15.1	28	+19	324 F
g15426/A1					+0	324
g15426/Y	OAI21XL	1	3.0	42	+40	364 R
fopt15789/A					+0	364
fopt15789/Y	BUFX3	2	14.2	19	+38	402 R
g15412/A1					+0	402
g15412/Y	AOI21X4	1	17.9	27	+20	423 F
g15411/C					+0	423
g15411/Y	NAND3X6	10	30.0	24	+24	447 R
fopt15744/A					+0	447
fopt15744/Y	INVX3	3	11.4	13	+11	458 F
g15396/A1					+0	458
g15396/Y	AOI21X1	2	4.8	36	+32	490 R

					✓	✓
g15400/A1						
g15468/Y	AOI21X2	3	9.3	34	+34	265 R
g15454/B					+0	265
g15454/Y	NAND2X2	1	4.1	26	+13	278 F
g15445/B					+0	278
g15445/Y	CLKAND2X3	5	15.1	21	+42	321 F
g15442/A					+0	321
g15442/Y	INVX2	5	12.4	21	+22	342 R
g15436/B0					+0	342
g15436/Y	AOI21X1	1	3.0	29	+11	354 F
g15420/B0					+0	354
g15420/Y	OAI21X1	2	3.4	33	+21	374 R
g15405/A					+0	374
g15405/Y	NAND2XL	1	2.8	24	+20	395 F
g15400/A					+0	395
g15400/Y	NAND2X1	1	5.5	23	+24	419 R
g15398/A					+0	419
g15398/Y	INVX2	1	11.7	16	+14	433 F
g15395/C					+0	433
g15395/Y	NAND3X4	10	24.6	29	+23	456 R
g15383/A1					+0	456
g15383/Y	OAI21XL	2	4.6	34	+29	484 F
g15362/A					+0	484
g15362/Y	NAND2X1	2	4.4	22	+25	510 R
g15352/A					+0	510
g15352/Y	NAND2X1	1	3.0	16	+15	525 F
g15350/B					+0	525
g15350/Y	NOR2X1	1	5.6	35	+30	555 R
g15344/D					+0	555
g15344/Y	NAND4X2	1	8.3	39	+31	586 F
fopt15735/A					+0	586
fopt15735/Y	INVX3	8	30.3	33	+35	621 R
g10895/B					+0	621
g10895/Y	NAND2XL	2	3.6	35	+24	645 F
g10787/B					+0	645
g10787/Y	AND2X1	5	9.9	23	+45	690 F
g10758/A					+0	690
g10758/Y	NAND3X1	1	1.8	17	+19	709 R
R2_control_signals2_reg[13]/D <<	SDFFSRX1				+0	709
R2_control_signals2_reg[13]/CK	setup			1	+92	801 R
<hr/>						
(clock clk)						
	capture					780 R
	latency				+10	790 R
	uncertainty				-10	780 R
<hr/>						
Cost Group	:	'clk'	(path_group 'clk')			
Timing slack	:	-21ps	(TIMING VIOLATION)			
Start-point	:	r3/regPR_reg/CK				
End-point	:	m0/R2_control_signals2_reg[13]/D				

Power Report

Category	Leakage	Internal	Switching	Total	Row%
memory	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
register	1.43656e-04	1.55008e-02	6.59694e-03	2.22414e-02	85.13%
latch	8.64628e-07	7.21784e-06	1.06671e-06	9.14918e-06	0.04%
logic	5.48396e-05	1.46314e-03	8.63564e-04	2.38154e-03	9.12%
bbox	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
clock	0.00000e+00	0.00000e+00	1.49451e-03	1.49451e-03	5.72%
pad	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
pm	0.00000e+00	0.00000e+00	0.00000e+00	0.00000e+00	0.00%
Subtotal	1.99360e-04	1.69712e-02	8.95608e-03	2.61266e-02	100.01%
Percentage	0.76%	64.96%	34.28%	100.00%	100.00%

Differences and Explanation of New Entities in netlist after Scan Chain Insertion

In our model we have inserted 2 scan chains. Scan changes introduce new ports to our design. Input ports introduced are scan_enable, test_mode, DFT_sdi_1 and DFT_sdi_2. Output ports introduced are DFT_sdo_1 and DFT_sdo_2. x_1 refers to the entities of the first scan chain while x_2 refers to second scan chain entities. When we did insertion of the scan chain all the DFF in the netlist were replaced by the special scan chain units from the library called SDFFRHQX1 cells.

Here DFT_sdi_(i) represents scan chain input for the (i) scan chain.

Here DFT_sdo_(i) represents scan chain output for the (i) scan chain.

All of these scan chain cells operate in three modes. These modes operate via the above specified inputs.

Scan change modes:

Mode	test_mode	scan_en
Normal	0	0
Shift	1	1
Capture	1	0

Normal mode is the simple functionality based mode where design operates as per the normal designed functionality. (scan cells work as DFF (s))

Shift mode is the mode where the scan chain works as a chain of shift registers. (value of scan input is captured)

Capture mode is the mode where the response of a combinational circuit applied at Dpin is captured.

File Edit View

```

(R2_control_signals[0]));
SDFFRHQX1 \R2_control_signals2_reg[2] (.RN (n_0), .CK (clock), .D
(n_343), .SI (R2_control_signals[0]), .SE (DFT_sen), .Q
(R2_control_signals[2]));
SDFFRHQX1 \R2_control_signals2_reg[3] (.RN (n_0), .CK (clock), .D
(n_352), .SI (R2_control_signals[2]), .SE (DFT_sen), .Q
(R2_control_signals[3]));
SDFFRHQX1 \R3_control_signals2_reg[0] (.RN (n_0), .CK (clock), .D
(n_346), .SI (R2_control_signals[16]), .SE (DFT_sen), .Q
(R3_control_signals[0]));
SDFFRHQX1 \R3_control_signals2_reg[1] (.RN (n_0), .CK (clock), .D
(n_351), .SI (R3_control_signals[0]), .SE (DFT_sen), .Q
(R3_control_signals[1]));
SDFFRHQX1 \R3_control_signals2_reg[3] (.RN (n_0), .CK (clock), .D
(n_344), .SI (R3_control_signals[1]), .SE (DFT_sen), .Q
(R3_control_signals[3]));
SDFFRHQX1 \response_signals2_reg[0] (.RN (n_0), .CK (clock), .D
(n_277), .SI (R3_control_signals[16]), .SE (DFT_sen), .Q
(response_signals[0]));
SDFFRHQX1 \response_signals2_reg[1] (.RN (n_0), .CK (clock), .D
(n_162), .SI (response_signals[0]), .SE (DFT_sen), .Q
(response_signals[1]));
SDFFRHQX1 \response_signals2_reg[2] (.RN (n_0), .CK (clock), .D
(n_321), .SI (response_signals[1]), .SE (DFT_sen), .Q
(response_signals[2]));
SDFFRHQX1 \response_signals2_reg[3] (.RN (n_0), .CK (clock), .D
(n_350), .SI (response_signals[2]), .SE (DFT_sen), .Q
(response_signals[3]));
OR3XL g20110(.A (n_308), .B (n_365), .C (n_359), .Y (n_355));
AO21X1 g20112(.A0 (n_252), .A1 (n_341), .B0 (n_326), .Y (n_354));
OAI2BB1XL g20114(.A0N (n_243), .A1N (n_341), .B0 (n_324), .Y (n_353));
OAI21XL g20115(.A0 (n_267), .A1 (n_342), .B0 (n_323), .Y (n_352));
OAI21X1 g20117(.A0 (n_292), .A1 (n_363), .B0 (n_325), .Y (n_351));
AOI211X1 g20118(.A0 (n_231), .A1 (n_265), .B0 (n_169), .C0 (n_363),
v /n_35011.

```

```

wire n_339, n_340, n_341, n_342, n_343, n_344, n_345, n_346;
wire n_348, n_350, n_351, n_352, n_353, n_354, n_355, n_356;
wire n_357, n_358, n_359, n_360, n_361, n_363, n_365, n_366;
wire n_367, n_368, n_369, n_370, n_371, n_372, n_373, n_374;
wire n_375, n_376, n_377, n_378, n_379, n_380, n_381, n_382;
wire n_383, n_384, n_385, n_386, n_387, n_393, n_394, n_395;
wire n_396, n_397, n_401, n_438, n_439, n_440, n_442, n_443;
wire n_444, n_445, n_446, n_447, n_448, n_450, n_451, n_452;
wire n_453, n_454, n_455, n_456, n_457, n_458, n_886_BAR, n_2444_BAR;
CLKINVX1 g12475(.A (n_401), .Y (n_363));
DFFRHQX1 \R0_control_signals2_reg[2] (.RN (n_15), .CK (clock), .D (n_352), .Q (R0_control_signals[2]));
DFFRHQX1 \R0_control_signals2_reg[4] (.RN (n_15), .CK (clock), .D (n_356), .Q (R0_control_signals[4]));
DFFRHQX1 \R1_control_signals2_reg[0] (.RN (n_15), .CK (clock), .D (n_357), .Q (R1_control_signals[0]));
DFFRHQX1 \R1_control_signals2_reg[1] (.RN (n_15), .CK (clock), .D (n_172), .Q (R1_control_signals[1]));
DFFRHQX1 \R1_control_signals2_reg[2] (.RN (n_15), .CK (clock), .D (n_353), .Q (R1_control_signals[2]));
DFFRHQX1 \R1_control_signals2_reg[4] (.RN (n_15), .CK (clock), .D (n_359), .Q (R1_control_signals[4]));
DFFRHQX1 \R2_control_signals2_reg[0] (.RN (n_15), .CK (clock), .D (n_361), .Q (R2_control_signals[0]));
DFFRHQX1 \R2_control_signals2_reg[2] (.RN (n_15), .CK (clock), .D (n_354), .Q (R2_control_signals[2]));
DFFRHQX1 \R2_control_signals2_reg[3] (.RN (n_15), .CK (clock), .D (n_355), .Q (R2_control_signals[3]));
DFFRHQX1 \R3_control_signals2_reg[0] (.RN (n_15), .CK (clock), .D (n_173), .Q (R3_control_signals[0]));
DFFRHQX1 \R3_control_signals2_reg[1] (.RN (n_15), .CK (clock), .D (n_358), .Q (R3_control_signals[1]));
DFFRHQX1 \R3_control_signals2_reg[3] (.RN (n_15), .CK (clock), .D (n_351), .Q (R3_control_signals[3]));
DFFRHQX1 \response_signals2_reg[0] (.RN (n_15), .CK (clock), .D (n_294), .Q (response_signals[0]));
DFFRHQX1 \response_signals2_reg[1] (.RN (n_15), .CK (clock), .D (n_307), .Q (response_signals[1]));
DFFRHQX1 \response_signals2_reg[2] (.RN (n_15), .CK (clock), .D (n_328), .Q (response_signals[2]));
DFFRHQX1 \response_signals2_reg[3] (.RN (n_15), .CK (clock), .D (n_348), .Q (response_signals[3]));
INVXL g16828(.A (n_360), .Y (n_361));
AOI31XL g16830(.A0 (n_252), .A1 (n_188), .A2 (n_4), .B0 (n_339), .Y (n_360));
NAND3BXL g16831(.AN (n_367), .B (n_330), .C (n_281), .Y (n_359));
OAI2BB1XL g16832(.A0N (n_303), .A1N (n_4), .B0 (n_335), .Y (n_358));
OAI31XL g16834(.A0 (n_251), .A1 (n_189), .A2 (n_371), .B0 (n_338), .Y (n_357));
NAND4BXL g16835(.AN (n_373), .B (n_300), .C (n_350), .D (n_308), .Y (n_356));
OAI2BB1XL g16837(.A0N (n_301), .A1N (n_4), .B0 (n_334), .Y (n_355));
OAI211XL g16838(.A0 (n_374), .A1 (n_287), .B0 (n_315), .C0 (n_5), .Y (n_354));

```

Scan Chain Failure Detection Mechanism

Advantage of a single Scan chain system is that it helps to detect the failure in linear time instead of exponential time failure. In fact we have used 2 scan chain chains so it further decreased by a factor of 2. As for the mechanism, it is a simple shift-Normal-capture and shift cycle.

Basically we provide the test vectors via the scan chain input port which is then shifted to the combinational circuit launch flip flop after which the design works in the Normal mode resulting in the generation of the output bit which is then Captured by scan cell. After capturing the output it is shifted out.

These shifted out bits are then judged by the testing machine against the golden (correct) bits. If they don't match, a failure is detected.

This scan chain based testing helps us to do testing more efficiently in a much faster way.

QoR Analysis

Before DFT

	Clock Period	Cell Count	Area	Timing Slack
Min Area	4.32ns	1641	14361.420	2732ps

Optimal Control	2.55ns	1649	14384.884	814ps
Best Timing	0.78ns	1779	14857.190	-5ps

After DFT

	Clock Period	Cell Count	Area	Timing Slack
Min Area	4.32ns	1657	16512.530	2712ps
Optimal Control	2.55ns	1663	16504.204	812ps
Best Timing	0.78ns	1821	17626.687	-24ps

From the above table we can see that the area of the design always increases after DFT as compared to before DFT. This is due to extra pins in scan cells and extra routing resources required to form scan chains. For minimum area and optimal we see that, the number of cells is also increased this also leads to increase in area.

Timing slack decreases after DFT as compared to pre DFT. This is due to extra delay encountered due to routing the scan cells. In minimum area and the optimal case numbers of cells increment also leads to decrease in slack.

Timing Report Analysis after Scan Chain Insertion

In the pre DFT case timing report of tight clock case we have the required time, arrival time and slack as follows:

$$AT \Rightarrow 691 + 94(\text{setup}) \Rightarrow 785$$

$$RT \Rightarrow 780 + 10(\text{latency}) - 10(\text{uncertainty}) = 780$$

$$\text{Slack} \Rightarrow RT - AT = 780 - 785 = -5\text{ps}$$

In the post DFT case timing report of tight clock we have the required time, arrival time and slack as follows:

$$AT \Rightarrow 709 + 92(\text{setup}) = 801 \text{ ps}$$

$$RT \Rightarrow 780 + 10(\text{latency}) - 10(\text{uncertainty}) = 780\text{ps}$$

$$\text{Slack} \Rightarrow 780 - 801 = -21\text{ps}$$

We observe that the setup time has increased after the scan chain insertion. Also we can observe that the delays are less after the scan chain got inserted so the required time and the arrival time both are decreasing hence slack is reduced.