**Assignment**

**Hardware Design, Simulation, Synthesis and Implementation**

**Objective**: In this assignment, you will be designing, simulating and synthesizing various digital blocks including adders, multiplexers, decoder, parity generator, flipflops and an updown counter. A hardware software co-design approach is also introduced where the software part involves the processor being used solely to provide test inputs.

**Tools used**: Modelsim, Xilinx Vivado, Python, Synopsys

**Hardware used**: Zybo board, Digilent Discovery kit.

**Guidelines to be followed**:

* **Naming convention**: File name should match with the module name (eg: if the module is named “full-adder”, the design file name should be named as “full-adder.v” and the testbench should be named as “full-adder\_tb.v”)
* **Formatting**: Each Verilog file should have a proper header: giving the name, description, version history as given in the sample file. Comments should be given carefully so that program is understandable without too much clutter (You may refer the uploaded sample file)
* **Submission**: The completed word document in the given template (converted to pdf) and uploaded in AUMS. The same pdf along with all Verilog design and testbench files should be submitted in the given one drive link [ESLD-Assignment](https://amritavishwavidyapeetham-my.sharepoint.com/:f:/g/personal/b_karthi_cb_amrita_edu/EhoNanbAYFlIpNGdwEFp_QYBt5uO4R9K5pc6NbrCxUU-FA?e=iyey6P).

You can create a folder structure for yourself (XXX stands for the 3 digits of your roll number)

* + “ECE19xxx” -> Assignment 1A, Assignment 1 B, 1C, 1D, 1E

**PART E- ASIC Synthesis using Synopys**

1. Synthesize the counter as an ASIC using Synopsys tool using a clock constraint and paste the following
   * Synthesized output
   * Power report
   * Timing report (with a given clock constraint)

Date : 19/10/2023

1. **Synthesized output for the following code using Synopsis – DC ( Design Vision )**

/\* **Module** - OneHotEncUpDwn2Bit.v

19ECE347 - ELECTRONIC SYSTEM LEVEL DESIGN AND VERIFICATION ~ ASSIGNMENT 1 PART 1A

Roll no - CB.EN.U4ECE20101

Change history: 07/09/23 - V1.0 - Initial working version created (owner: Abinav Balachandar - CB.EN.U4ECE20101)

\*/

/\*

**Description**:

This is a 2-bit updown counter implemented using FSM and one "hot" encoding.

reset: A signal that resets the counter to 0.

clk: A clock signal that increments the counter on the rising edge.

count: The current value of the counter.

**Returns**:

The next value of the counter.

\*/

module OneHotEncUpDwn2Bit (

input clk,

input reset,

input count,

output [3:0]y

);

reg [3:0]y\_out;

reg [1:0] current\_state;

reg [1:0] next\_state;

wire cl1;

parameter CNT0 = 2'b00;

parameter CNT1 = 2'b01;

parameter CNT2 = 2'b11;

parameter CNT3 = 2'b10;

always @ (posedge cl1)

begin

if (count==1'b1)

begin if (reset==1'b1)

current\_state<=CNT0;

else

current\_state<=next\_state; end

else

begin if (reset==1'b1)

current\_state<=CNT3;

else

current\_state<=next\_state;

end

end

always @(current\_state)

begin

if (count==1'b1)

case (current\_state)

CNT0:begin next\_state<=CNT1;

y\_out<=4'b0001;end

CNT1: begin next\_state<=CNT2;

y\_out<=4'b0010;end

CNT2: begin next\_state<=CNT3;

y\_out<=4'b0100;end

CNT3: begin next\_state<=CNT0;

y\_out<=4'b1000;end

endcase

else

case (current\_state)

CNT0: begin next\_state<=CNT3;

y\_out<=4'b0001;end

CNT1: begin next\_state<=CNT0;

y\_out<=4'b0010;end

CNT2: begin next\_state<=CNT1;

y\_out<=4'b0100;end

CNT3: begin next\_state<=CNT2;

y\_out<=4'b1000;end

endcase

end

clk\_divider cl(

.clk\_125MHz(clk),

.clk\_1Hz(cl1)

);

assign y=y\_out;

endmodule

// Clk divider

module clk\_divider (

input wire clk\_125MHz, // 125MHz input clock

output reg clk\_1Hz // 1Hz output clock

);

reg [26:0] counter; // 27 bits for dividing 125MHz to 1Hz

always @(posedge clk\_125MHz) begin

if (counter == 27'd62499999) begin // 125e6 cycles = 1Hz

counter <= 0;

clk\_1Hz <= ~clk\_1Hz; // Toggle the output every 125e6 cycles

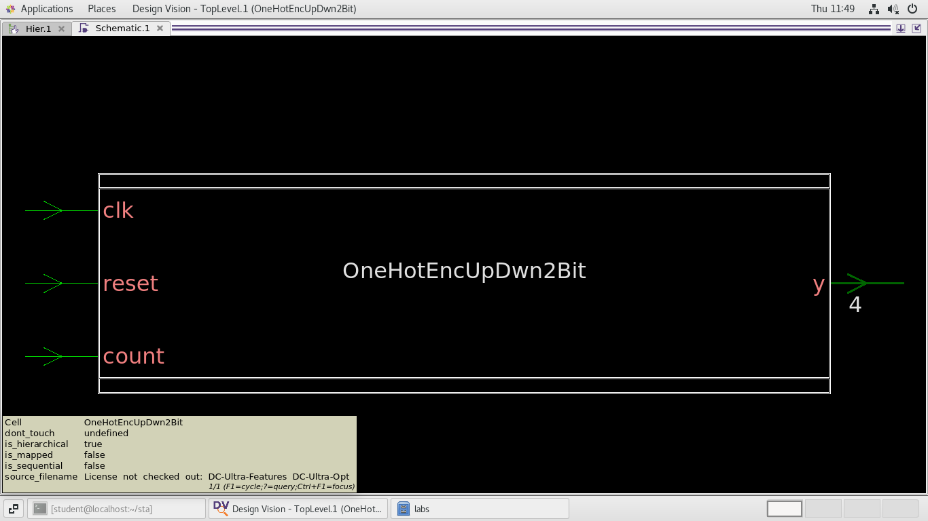
end else begin

counter <= counter + 1;

end

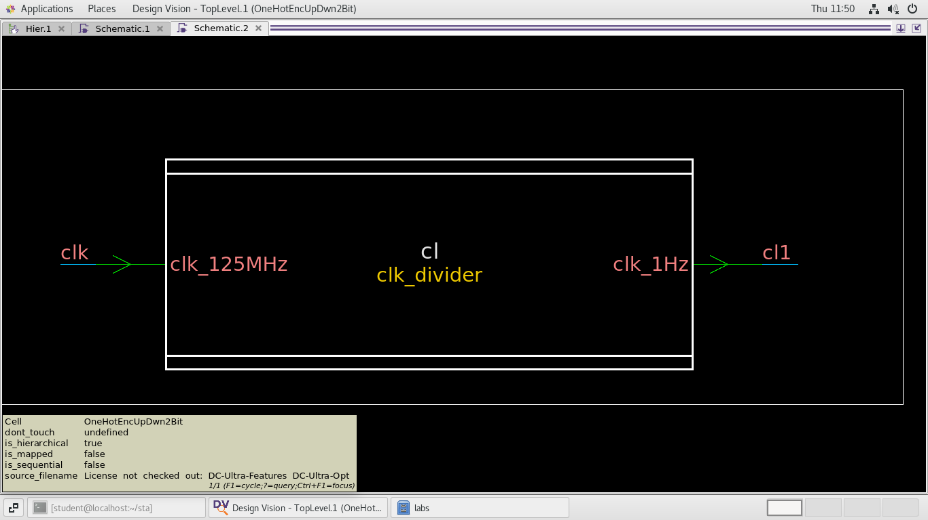
end

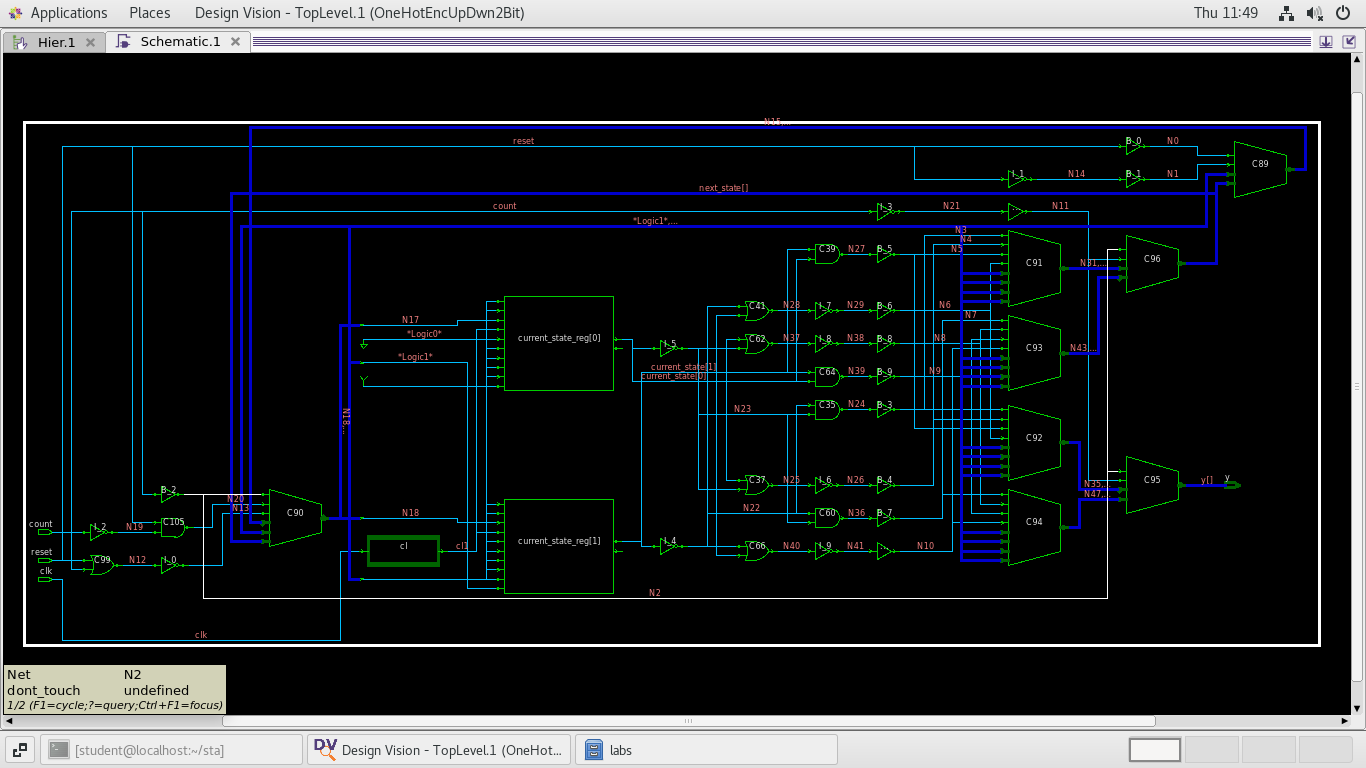
endmodule



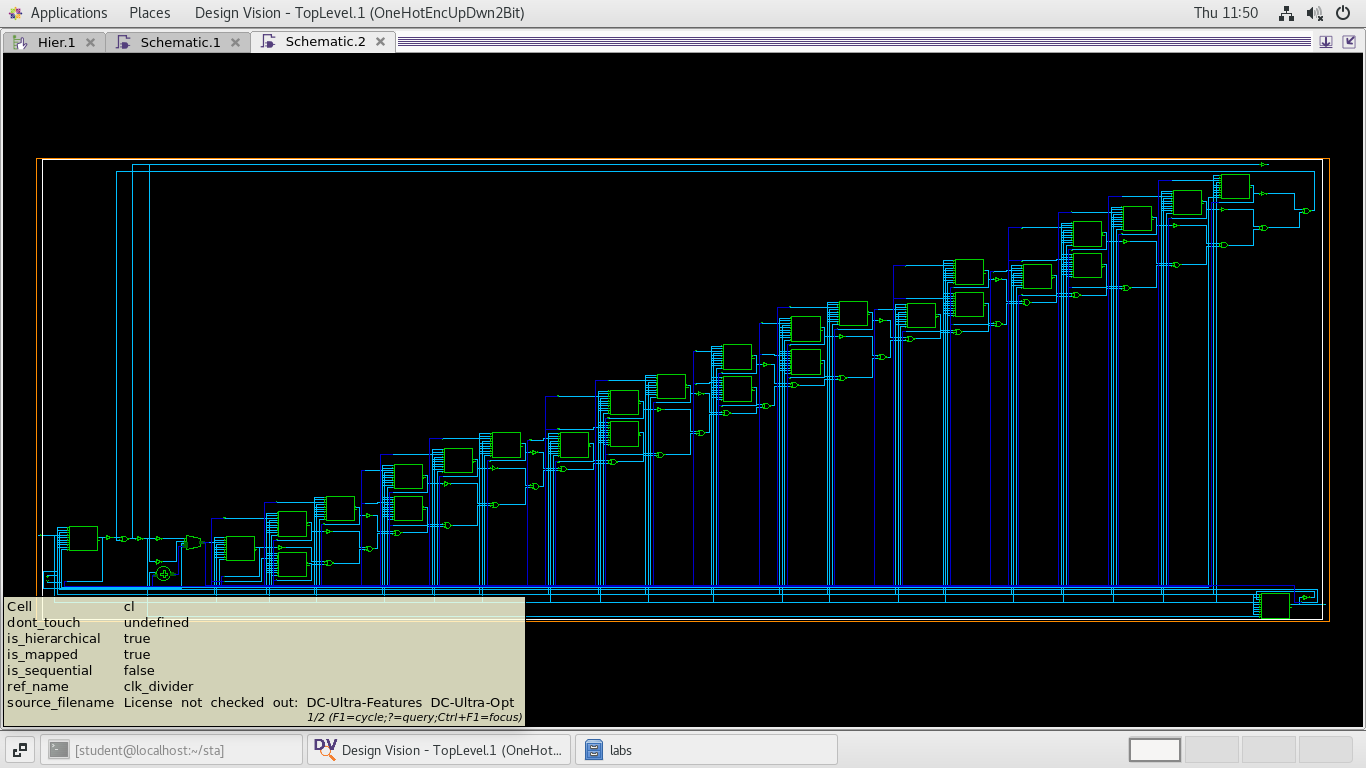
**Fig1. Block representation** of my Top Module **– OneHotEncUpDwn2Bit** after “analyze” and “elaborate” process.

**Fig2. Block representation** of **Clk\_divider** module, a sub module which is connected to the top block via wire cl1



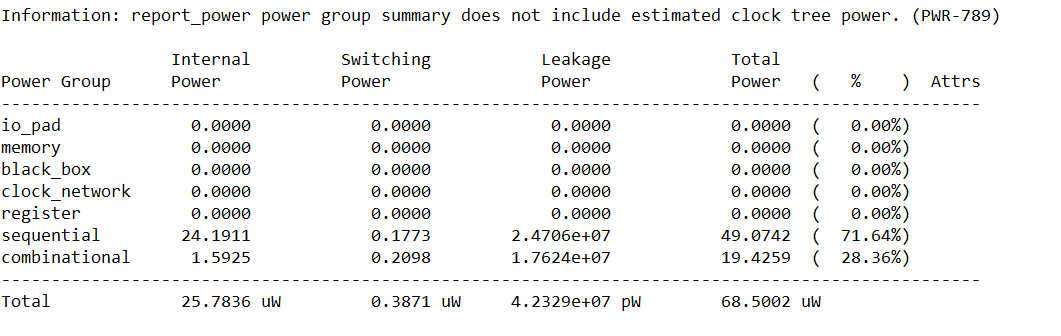
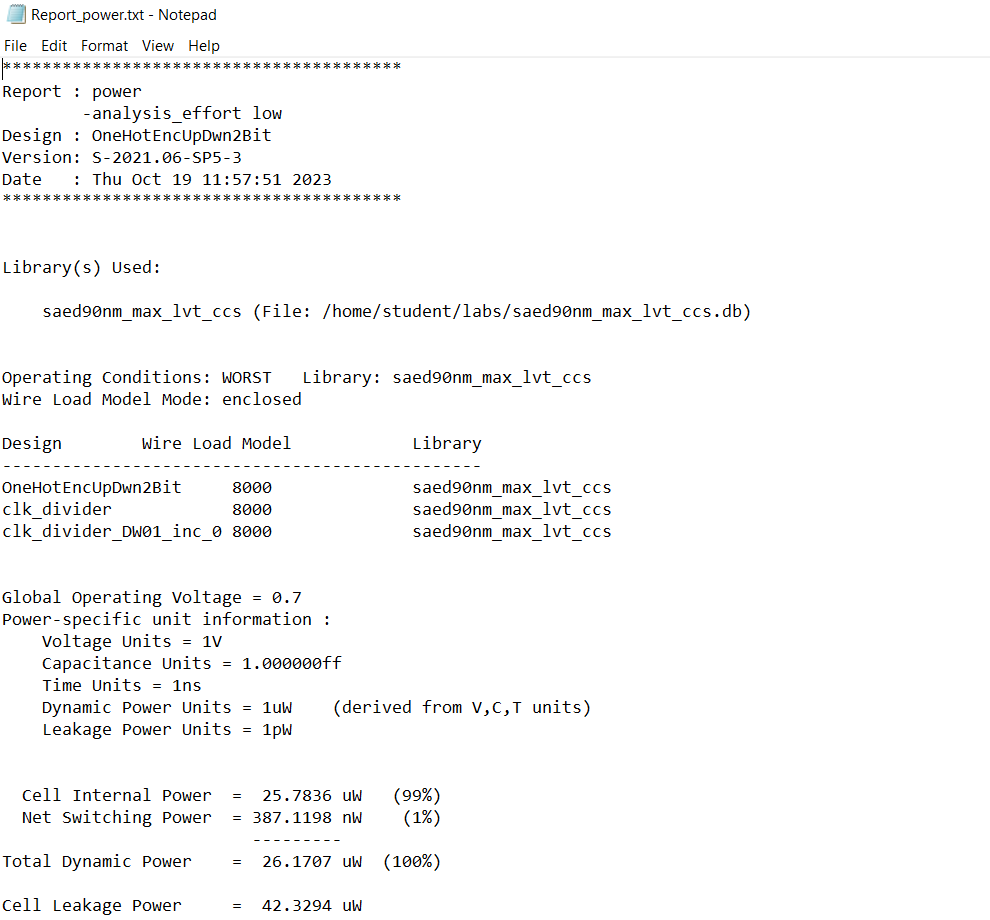


**Fig3. Schematic representation of synthesised code.**



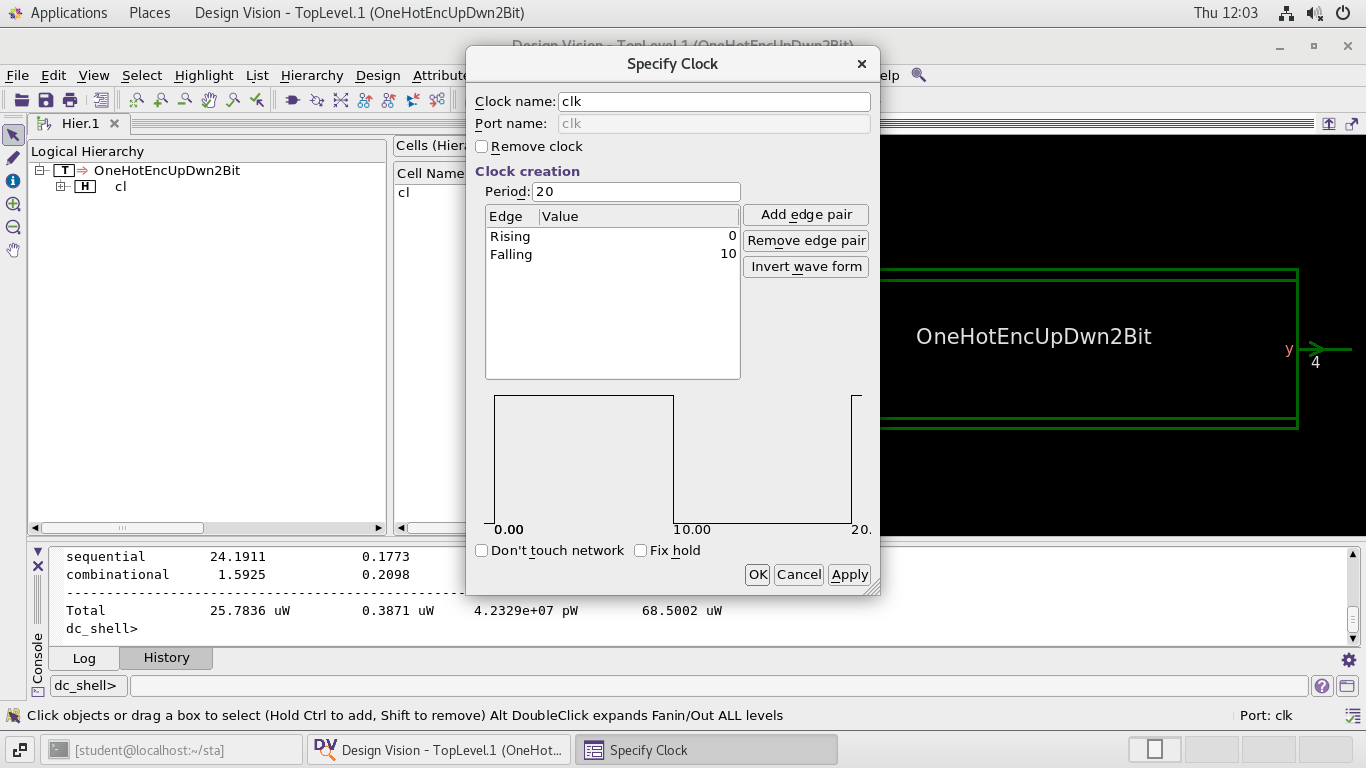
**Fig4. Schematic representation of Clk divider module.**

1. **Power report :**

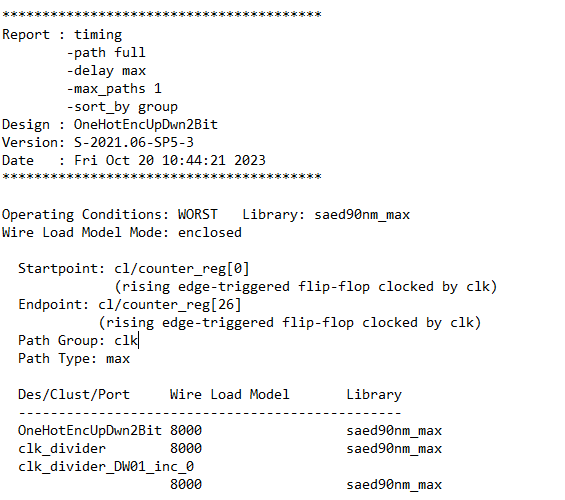


**Fig5. The above figure represents the power report generated, .txt can be found here.**

1. **Timing report (with constraints)**

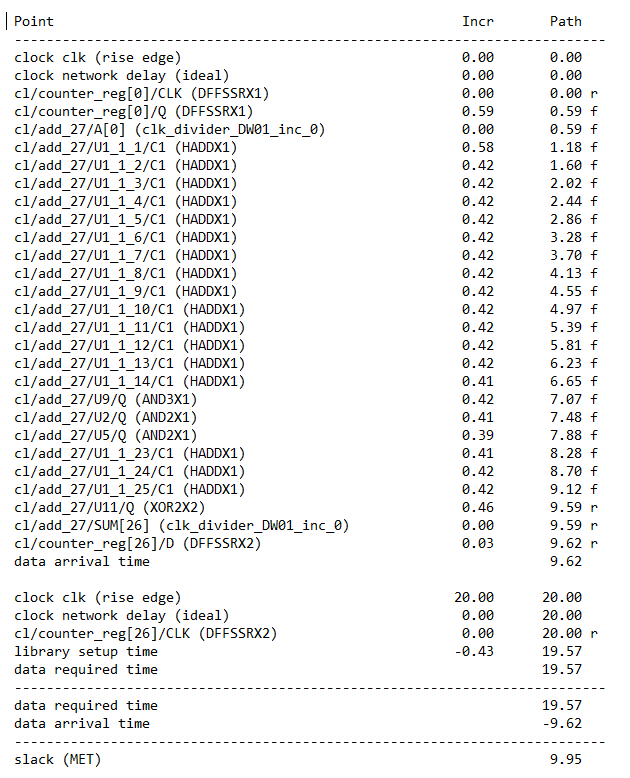
**Fig6. Specifying clock.**

**Period = 20ns**

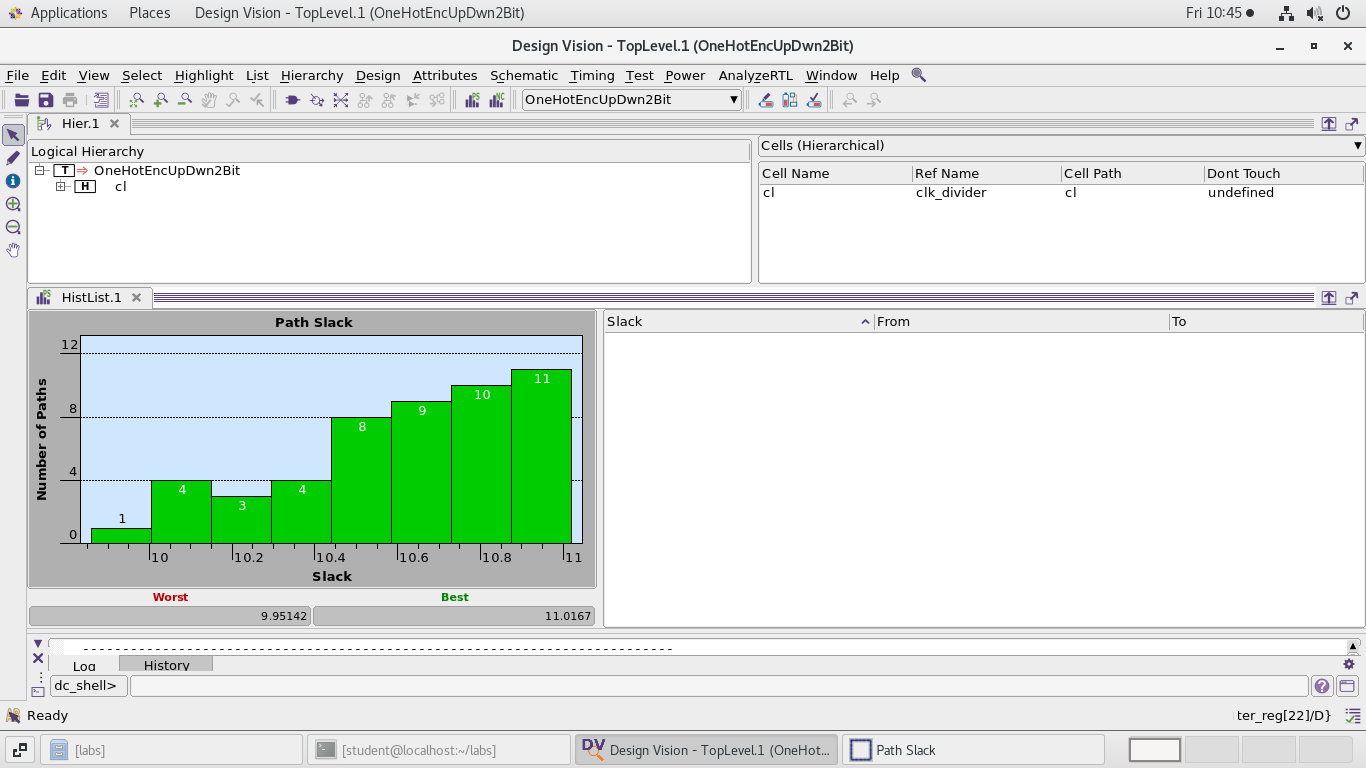


**Fig7. Part(i) Timing Report**

( constraint given was equal rise and fall time of 2ns)

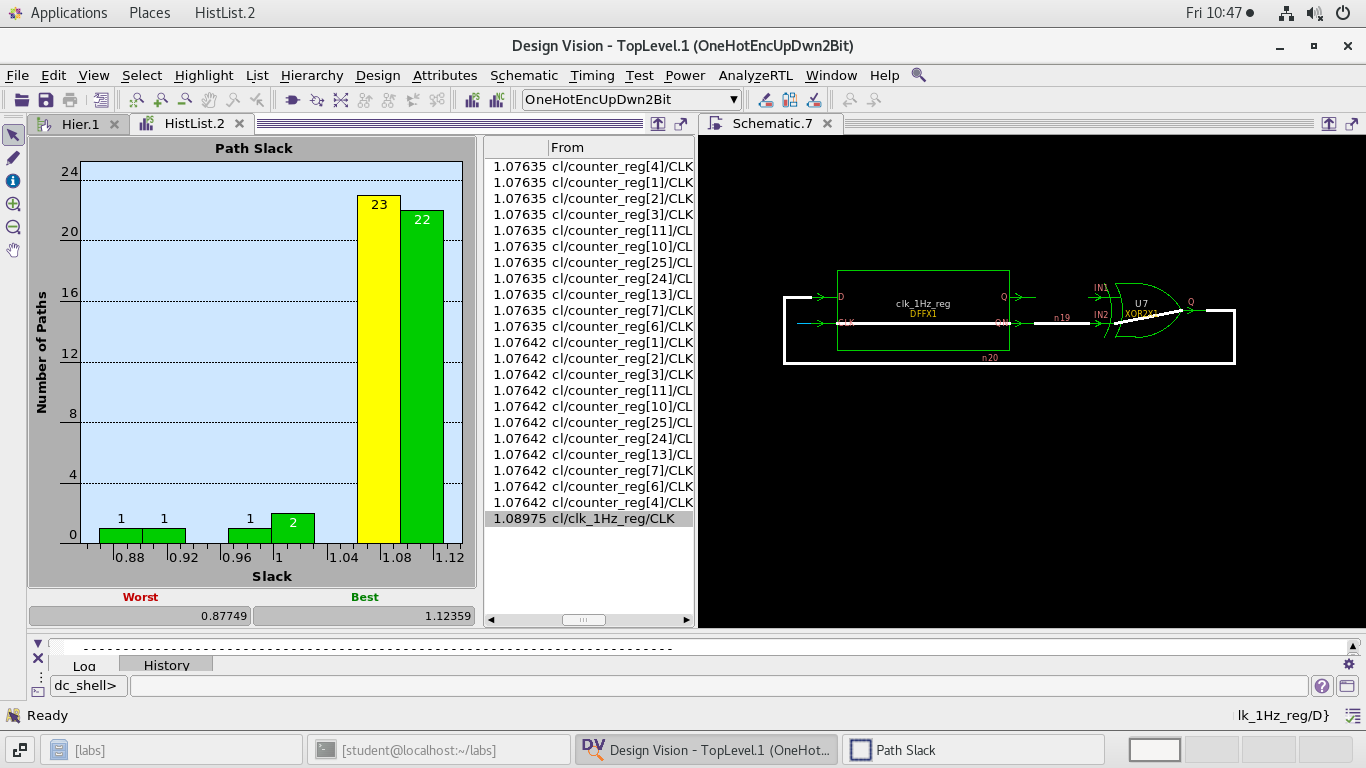


**Fig8. Part(ii) Timing report**



**Fig9 . Path slack – for Max conditions.**

Path slack : Slack of a timing path is the amount of time by which a violation is avoided.



**Fig10. HistList.2 represents – path Slack for min conditions.**

In the above figure, the middle column represents different connections through it. It has from and to addresses. The Schematic represents the travel path chosen; I have chosen **from : cl/clk\_1Hz\_reg/clk**

**To: U7(XOR2)**