

VLSI DESIGN
INDUSTRIAL TRAINING REPORT

for the course
18ECP105L - Industrial Training- II

Submitted by

Pushpal Das (RA1911004010565)

Semester – VI

Academic Year: 2021–22

In partial fulfilment for the award of the degree

Of

BACHELOR OF TECHNOLOGY

in

ELECTRONICS AND COMMUNICATION ENGINEERING of

FACULTY OF ENGINEERING AND TECHNOLOGY



S.R.M. Nagar, Kattankulathur, Kancheepuram District

March 2022

1

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

This is to certify that this report titled “VLSI DESIGN” is the Bonafide work of “PUSHPAL DAS (RA1911004010565)”, who carried out the industrial training. Certified further, that to the best of my knowledge the work reported herein does

not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

Signature Signature

Mr. Pushpal Das **Mr.G.Elavel Visuvanathan** ECE III Year Industrial training
coordinator Dept. of Electronics and Dept. of Electronics and Communication
Engineering Communication Engineering

2

ACKNOWLEDGEMENT

The success and final outcome of this Project required a lot of guidance, assistance from many Professors and I am extremely privileged to have got this all along the completion of my project. I owe my deep gratitude to my training company Vyorius, which took keen interest in my project work and guided all along, till the completion of my project work by providing all the necessary information for developing a practical approach. I am thankful and fortunate enough to get constant encouragement, support and guidance from all Professors of ECE Department. I would like to express my gratitude towards my parents and seniors for their kind co-operation and encouragement which helped me in completion of the Project.





VLSI DESIGN PROJECTS/INTERNSHIP

College Name

- SRM Institute of Science and Technology.

Company Name

- Vyorius.

Domain

- VLSI design.

Project topics

1. Basic circuits written in Verilog/VHDL, simulated and implemented on the FPGA
2. UART communication to print a single character
3. FSM Designs: Mealy & Moore Machines and Up Down Counter
4. 16-Bit RISC Processor
5. Digital Design

Hardware and software used

Hardware,

Mimas A7 Mini FPGA Development Board

Mimas A7 Mini is an easy-to-use FPGA Development board featuring Artix 7 FPGA (XC7A35T – FTG256C package) with FTDI's

FT2232H Dual-Channel USB device. It is an Artix-7 based replacement and upgrades of Mimas Spartan 6 FPGA Board. It is specially designed for the development and integration of FPGA based accelerated features to other designs. The USB 2.0 host interface based on popular FT2232H offers high bandwidth data transfer and board programming without the need for any external programming adapters.

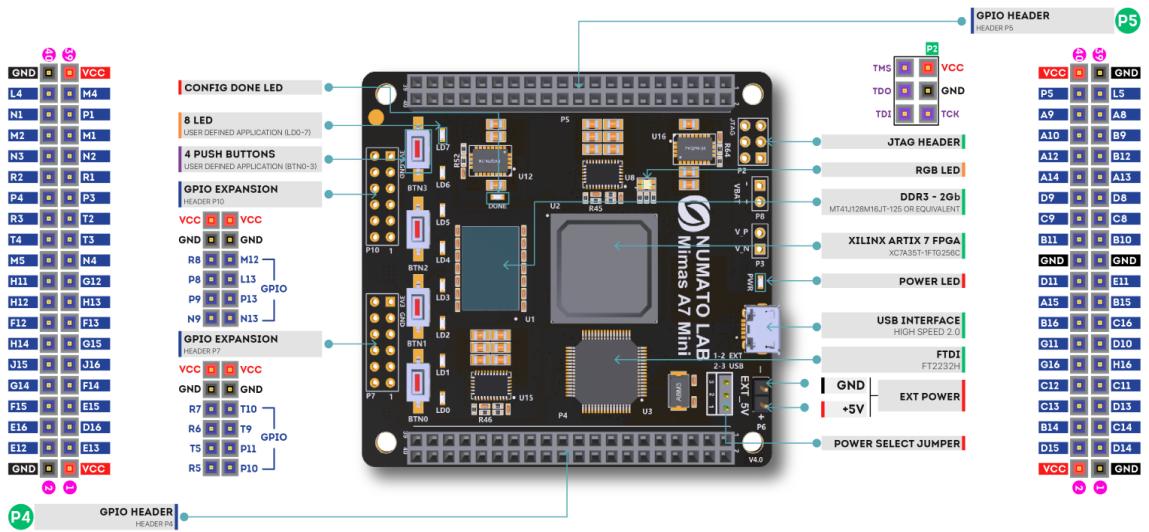
Features,

- Device: Xilinx Artix 7 FPGA (XC7A35T-1FTG256C)
- DDR3: 2Gb DDR3 (MT41J128M16JT-125 or equivalent)
- Built-in programming interface. No expensive JTAG adapters needed for programming the board
- Onboard 128Mb flash memory for FPGA configuration storage and custom user data storage
- High-Speed USB 2.0 interface for On-board flash programming. FT2232H Channel B is dedicated to JTAG Programming. Channel A can be used for custom applications
- 100MHz CMOS oscillator
- 8 LEDs, 1 RGB LED and 4 Push Buttons for user-defined purposes
- FPGA configuration via JTAG and USB
- Maximum IOs for user-defined purposes

FPGA – 70 IOs (35 professionally length matched Differential Pairs) and two 2×6 Expansion Headers

Applications,

- Product Prototype Development
- Accelerated computing integration
- Development and testing of custom embedded processors
- Communication devices development
- Educational tool for Schools and Universities



Specifications,

Attribute	Value
Dimensions	6 × 4 × 1 in
FPGA	<u>XC7A35T – 1FTG256C</u>
Memory	<u>DDR3 – 2Gb</u>
Configuration Options	<u>JTAG, USB</u>
Host Interface	<u>USB 2.0</u>
Primary Clock Frequency	<u>100MHz</u>
Number Of GPIOs (Max)	<u>70</u>
Number Of Diff Pairs	<u>35</u>

Software



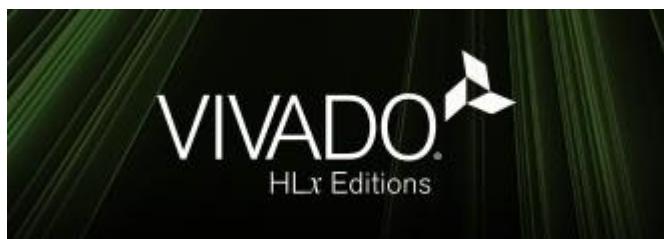
Icarus Verilog is a Verilog simulation and synthesis tool. It operates as a compiler, compiling source code written in Verilog (IEEE-1364) into some target format. For batch simulation, the compiler can

generate an intermediate form called vvp assembly. This intermediate form is executed by the ``vvp'' command. For synthesis, the compiler generates netlists in the desired format.

The compiler proper is intended to parse and elaborate design descriptions written to the IEEE standard *IEEE Std 1364-2005*. This is a fairly large and complex standard, so it will take some time to fill all the dark alleys of the standard, but that's the goal.

Icarus Verilog is a work in progress, and since the language standard is not standing still either, it probably always will be. That is as it should be. However, I will make stable releases from time to time, and will endeavour to not retract any features that appear in these stable releases. The quick links above will show the current stable release.

The main porting target is Linux, although it works well on many similar operating systems. Various people have contributed precompiled binaries of stable releases for a variety of targets. These releases are ported by volunteers, so what binaries are available depends on who takes the time to do the packaging. Icarus Verilog has been ported to That Other Operating System, as a command line tool, and there are installers for users without compilers. You can compile it entirely with free tools, too, although there are precompiled binaries of stable releases.



Vivado Design Suite is a software suite produced by [Xilinx](#) for synthesis and analysis of [HDL](#) designs, superseding [Xilinx ISE](#) with additional features for [system on a chip](#) development and [high-level synthesis](#).^{[11][12][13]} Vivado represents a ground-up rewrite and re-thinking of the entire design flow (compared to ISE).^{[14][15][16]}

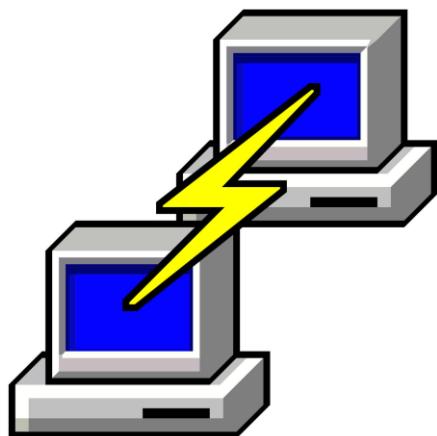
Like the later versions of [ISE](#), Vivado includes the in-built logic simulator [ISIM](#).^[17] Vivado also introduces high-level synthesis, with a toolchain that converts C code into programmable logic.^[18]

Replacing the 15-year-old ISE with Vivado Design Suite took 1000 [person-years](#) and cost US\$200 million.



Tenagra is an FPGA System management tool for configuring and communicating with Numato Lab's supported FPGA modules and development platforms. This software is designed to be a single interface for managing the devices and exercising some of the available features. Currently, Tenagra supports configuring the FPGA module/board (programming) and Memory Exerciser that can transfer data between various memories on the device. This includes both external DDR Memory and Block RAM available within the FPGA device. With Tenagra, you can create multiple configuration setups with different bitstreams and settings for each device model so that switching between multiple bitstreams is a breeze. This is especially helpful during development where the device may need to be reprogrammed with various bitstreams repeatedly.

Driver for Mimas A7 Mini Module



PuTTY is a free implementation of SSH (**and telnet**) for PCs running **Microsoft Windows** (it also includes an xterm terminal emulator). You will find PuTTY useful if you want to access an account on a Unix or other multi-user system from a PC (for example your own or one in an internet cafe).

1. Basic circuits written in Verilog/VHDL, simulated and implemented on the FPGA

Software to download

Notepad++

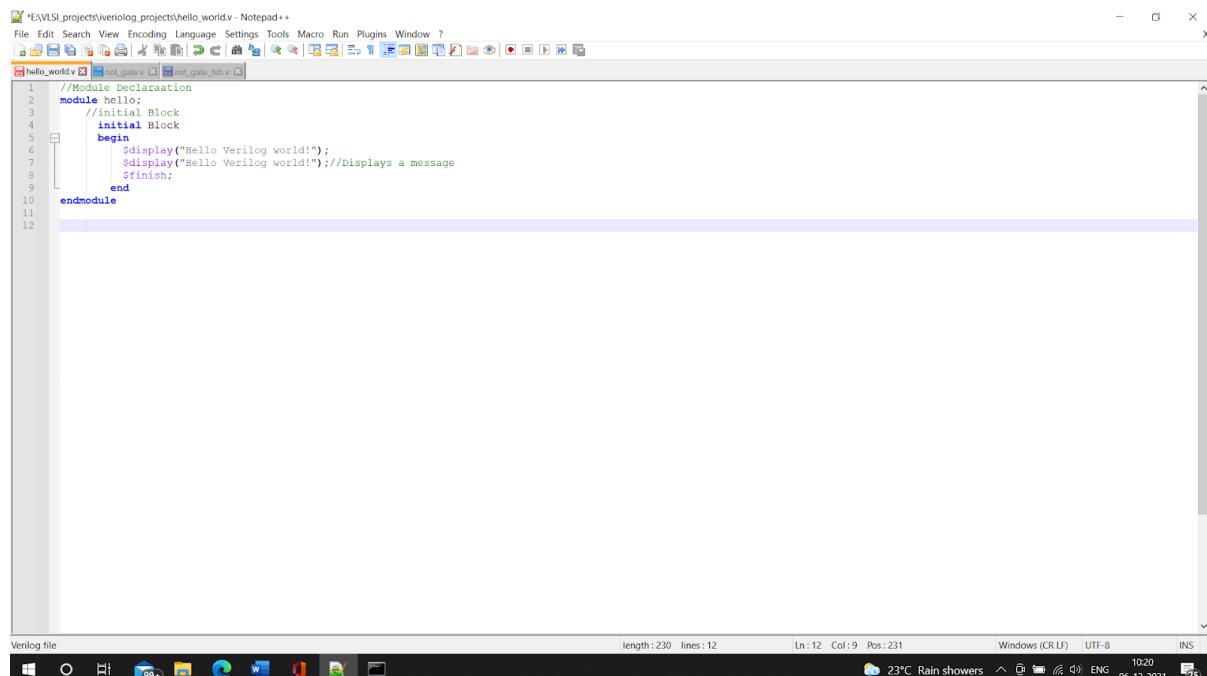
Step 1- Write the verilog program in notepad ++

Step 2-Open command prompt to initialise the data on the particular pc or laptop.

Iverilog ,should be already installed to interface command prompt with the code in the file destination where we will be saving our verilog program file in with 'xyz.v' extension.

Step 3- Now run with the same algorithm with basic program as shown below in notepad ++

1



The screenshot shows the Notepad++ application window. The title bar reads "EVLISI_projects\verilog_projects\hello_world.v - Notepad++". The menu bar includes File, Edit, Search, View, Encoding, Language, Settings, Tools, Macro, Run, Plugins, Window, and ?.

The code editor contains the following Verilog code:

```
1 //Module Declaration
2 module hello;
3   //initial Block
4   initial Block
5   begin
6     $display("Hello Verilog world!");
7     $display("Hello Verilog world!"); //Displays a message
8     $finish;
9   end
10 endmodule
11
12
```

The status bar at the bottom shows "Verilog file", "length: 230 lines: 12", "Ln: 12 Col: 9 Pos: 231", "Windows (CR LF)", "UTF-8", "INS", "23°C Rain showers", "ENG", "10:20", "06-12-2021", and "25".

Output-The reading is displays in command prompt after opening the file directory.

Step 4-Now let's do it with not gate and get the output,

First of all, we have to write the Verilog program for nor gate and save it in the directory where the command prompt will be initiating it. Then after that we have to generate an output file in that very directory only with Verilog, and the dump file will also be executed here, so as a total there will be 4 files in our directory

File1

The screenshot shows a Notepad++ window with the title "E\VLSI_projects\verilog_projects\not_gate\not_gate.v - Notepad++". The code is a Verilog module named "not_gate" with behavioral modeling:

```
1 //Module Declaration
2 module not_gate(
3     //Ports Declaration
4     //Inputs
5     input not_input,
6     //Output
7     output not_output
8 );
9
10 //Gate modelling
11 not(not_output,not_input);
12 /*
13 //Data Flow modelling
14 assign not_output = ~not_input;
15
16 //Behavioural modelling
17 always@(not_input)
18 begin
19     if(not_input == 1'b0)
20         not_output <= 1'b1;
21     else
22         not_output <= 1'b0;
23     end
24 */
25
26
27
28 endmodule
```

Below the code, the status bar shows: length:410 lines:28 Ln:27 Col:1 Pos:407 Windows (CR LF) UTF-8 INS.

File2

The screenshot shows a Notepad++ window with the title "E\VLSI_projects\verilog_projects\not_gate\not_gate_tb.v - Notepad++". The code is a testbench module "not_gate_tb" with an initial block and waveform dumping:

```
1 //Test Bench for not_gate
2 module not_gate_tb;
3
4     //wire/reg Declarations
5     reg not_in;      //input
6     wire not_out;    //output
7
8     //Instantiation
9     not_gate notl(not_in, not_out);
10
11    //Initial block
12    initial
13    begin
14        $dumpfile("not_gate_output.vcd");
15        $dumpvars(0,not_gate_tb);
16
17        not_in = 1'b0;
18        #5 not_in = 1'b1;
19        #5 not_in = 1'b0;
20        #5 not_in = 1'b1;
21        #5 not_in = 1'b0;
22    end
23
```

Below the code, the status bar shows: length:445 lines:23 Ln:23 Col:10 Pos:446 Windows (CR LF) UTF-8 INS.

COMMAND PROMPT

The screenshot shows a Windows Command Prompt window with the following text:

```
P:\VLSI_projects\iverilog_projects\not_gate_tb> Notepad +>
File Edit Search V not_gate
File Home Share View
not_gate.v Local Disk (P:) > VLSI_projects > iverilog_projects > not_gate
Name Date modified Type Size
not_gate.v 08-08-2021 23:42 V File 1 KB
not_gate_output 08-08-2021 23:52 File 2 KB
not_gate_output.vcd 08-08-2021 23:52 VCD File 1 KB
not_gate_tb.v 08-08-2021 23:50 V File 1 KB
C:\Windows\System32\cmd.exe -gtkwave not_gate_output.vcd
P:\VLSI_projects\iverilog_projects\hello_world>iverilog -o hello_out1 hello_world.v
Hello Verilog world!
Welcome to VLSI Projects!
Hello_world.vb: $finish called at 0 (s)
P:\VLSI_projects\iverilog_projects\hello_world>cd ../
P:\VLSI_projects\iverilog_projects>cd not_gate
P:\VLSI_projects\iverilog_projects\hello_world>iverilog -o not_gate_output not_gate_tb.v not_gate.v
P:\VLSI_projects\iverilog_projects>not_gate>vvp not_gate_output
VCD info: dumpfile not_gate_output.vcd opened for output.
P:\VLSI_projects\iverilog_projects\hello_world>gtkwave
GTKWave Analyzer v3.3.108 (w)1999-2020 BS1
GTKWAVE | Use the -h, --help command line flags to display help.
WM Destroy
#5 not_in = 1'b1;
#5 not_in = 1'b0;
#5 not_in = 1'b1;
#5 not_in = 1'b0;
end
```

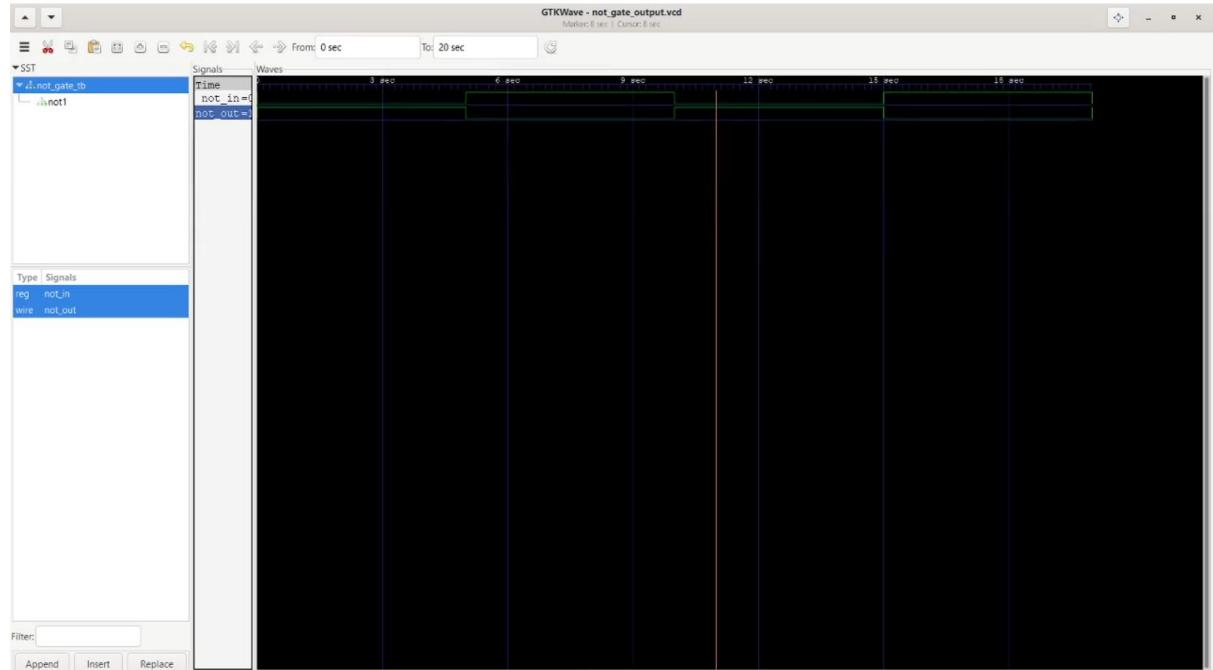
Below the command prompt, there is a terminal window showing the output of the gtkwave command:

```
P:\VLSI_projects\iverilog_projects\hello_world>gtkwave not_gate_output.vcd
GTKWave Analyzer v3.3.108 (w)1999-2020 BS1
[0] start time.
[20] end time.
```

At the bottom of the screen, status information is displayed:

length: 419 lines: 26 Ln: 13 Col: 10 Pos: 222 Windows (CR LF) UTF-8 OVR

Step 4-Then run the gtkwave in the command prompt , and call it with the output file extension, then in gtkwave we will see the test bench getting added up and thereby we can view the signal wave in gtkwave



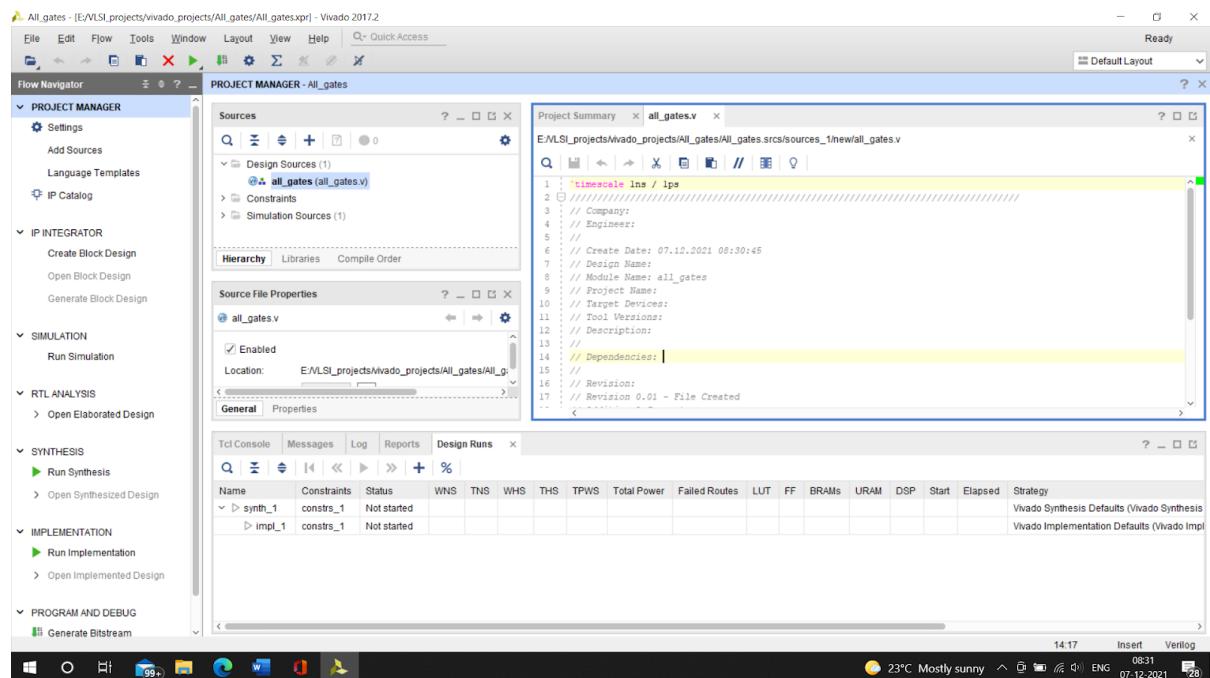
NOW WITH ALL BASIC GATES

Get the board support files from google by downloading the zip of Numato lab from GitHub link given

[numato \(Numato Lab\) \(github.com\)](https://github.com/numatolab)

After that as we know that we are dealing with Mimas A7 mini, all we have to do is extract the Mimas a7 mini files from numato labs and copy and paste it to xilinx vivado extension to it, that is we are simply taking a board file from one origin that is numato labs and extracting it and pasting it to xilinx vivavo software.

After that we have to open the vivado software where we have to create a new project and save in a directory, and after that through the process we have to select the board on it, that is Mimas A7 mini.



STEP 1- Write the code for all gates implementation on vivado

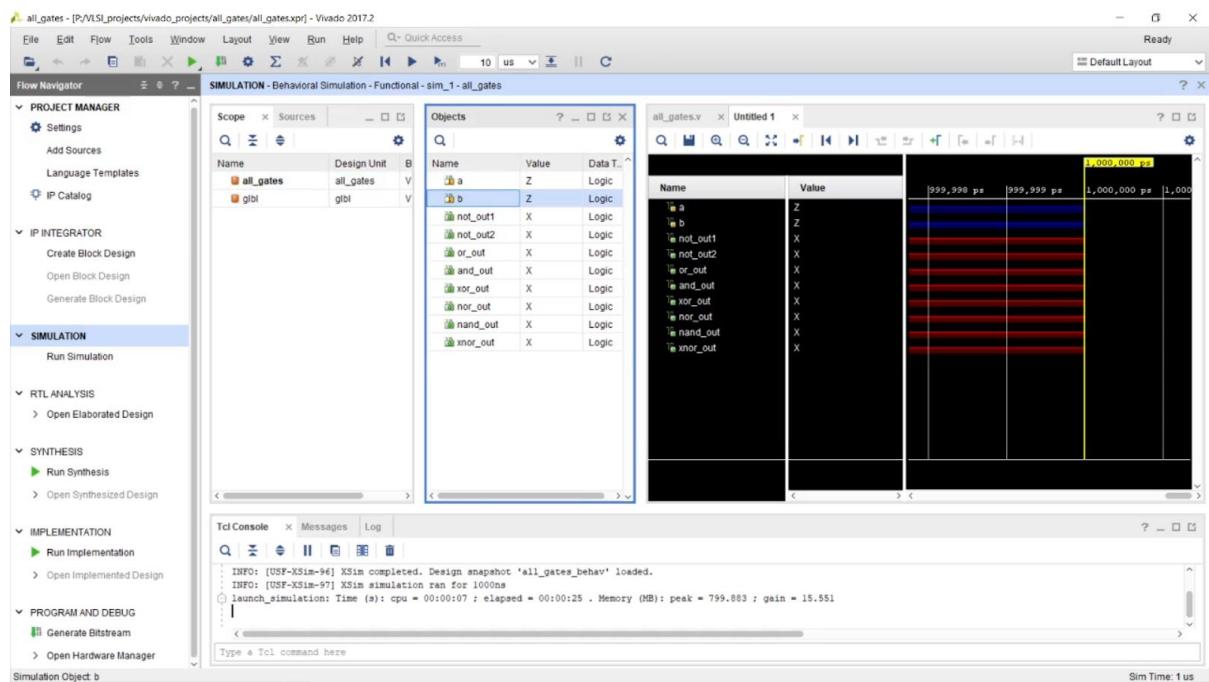
The screenshot shows the Vivado 2017.2 interface with the 'PROJECT MANAGER - All_gates' window open. The 'Sources' tab is selected, displaying the Verilog source code for the 'all_gates' module. The code defines inputs 'a' and 'b', and outputs 'not_out1', 'not_out2', 'or_out', 'and_out', 'xor_out', 'nor_out', 'nand_out', and 'xnor_out'. It also includes a section for 'Gate level modelling' using primitives like not, or, and, xor, not_and, nand, and xnor.

```

1: timescale 1ns / 1ps
2:
3:
4: module all_gates(
5:   //port declaration
6:   //input
7:   input a,
8:   input b,
9:
10:  //outputs
11:  output not_out1,
12:  output not_out2,
13:  output or_out,
14:  output and_out,
15:  output xor_out,
16:  output nor_out,
17:  output nand_out,
18:  output xnor_out
19: );
20:
21:
22: //Gate level modelling
23: not(not_out1,a);
24: not(not_out2,b);
25: or(or_out,a,b);
26: and(and_out,a,b);
27: xor(xor_out,a,b);
28: not(not_out,a,b);
29: nand(nand_out,a,b);
30: xnor(xnor_out,a,b);
31:
32 endmodule

```

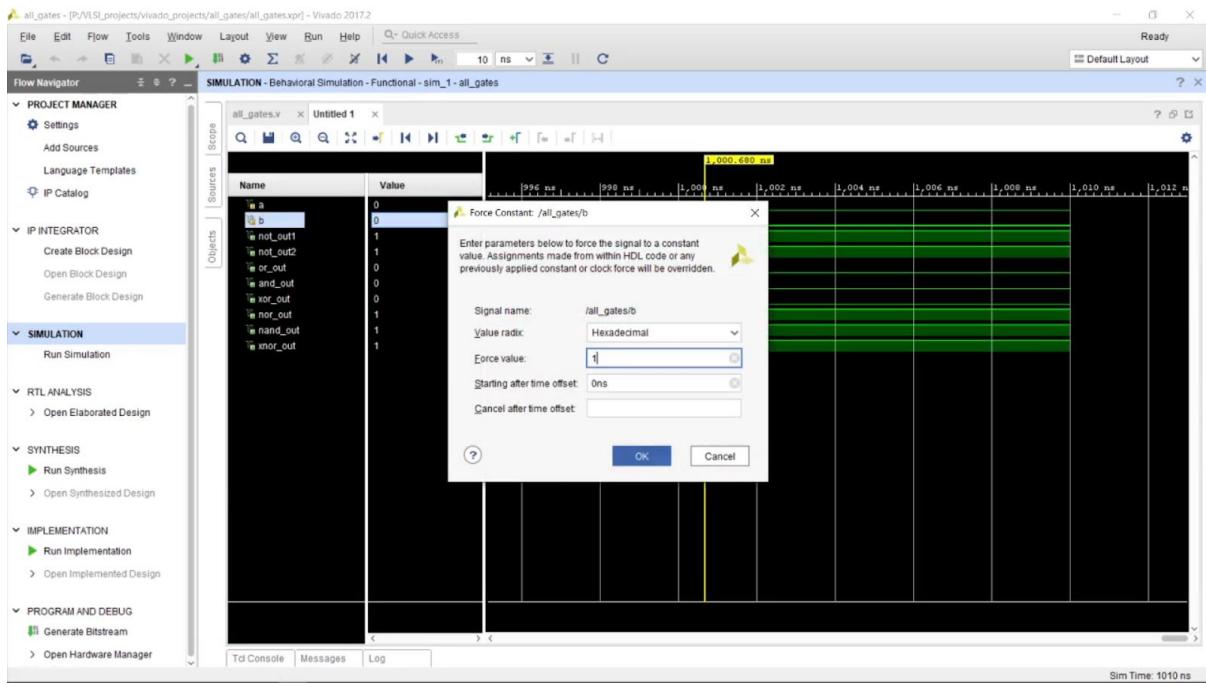
After that we have to run simulation to our program



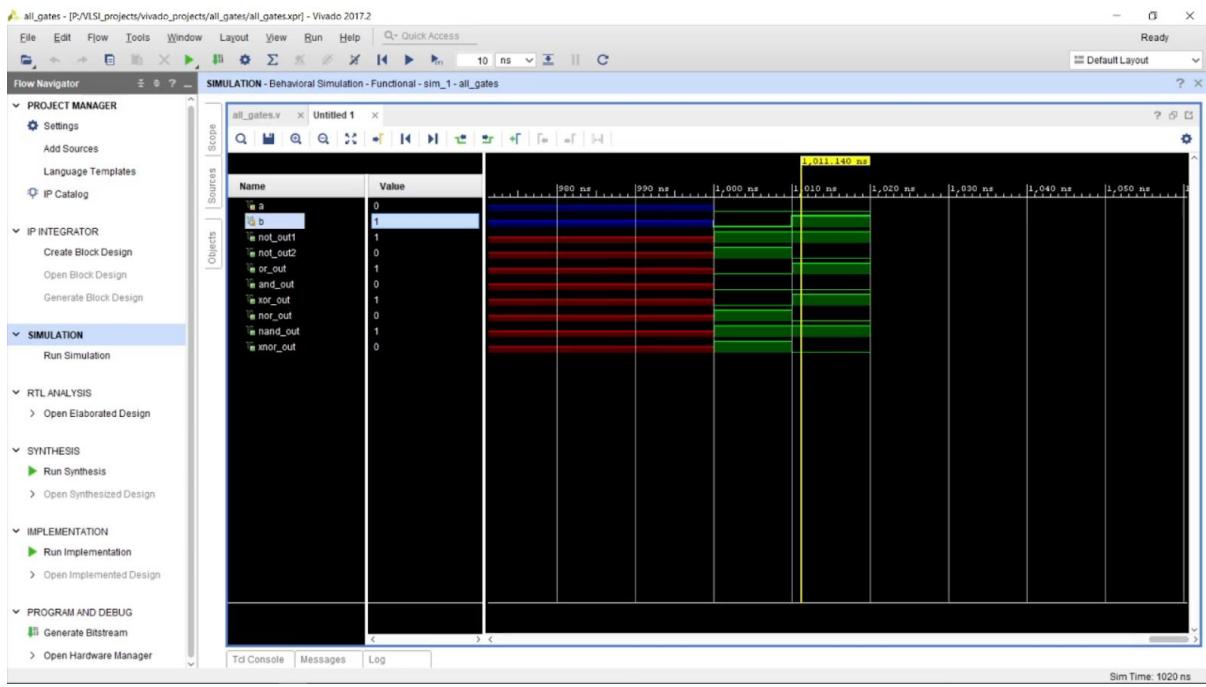
As we can see the inputs and the outputs.

After this window all we have to do is input value to the input's variable a and b, by forcing values or force constant

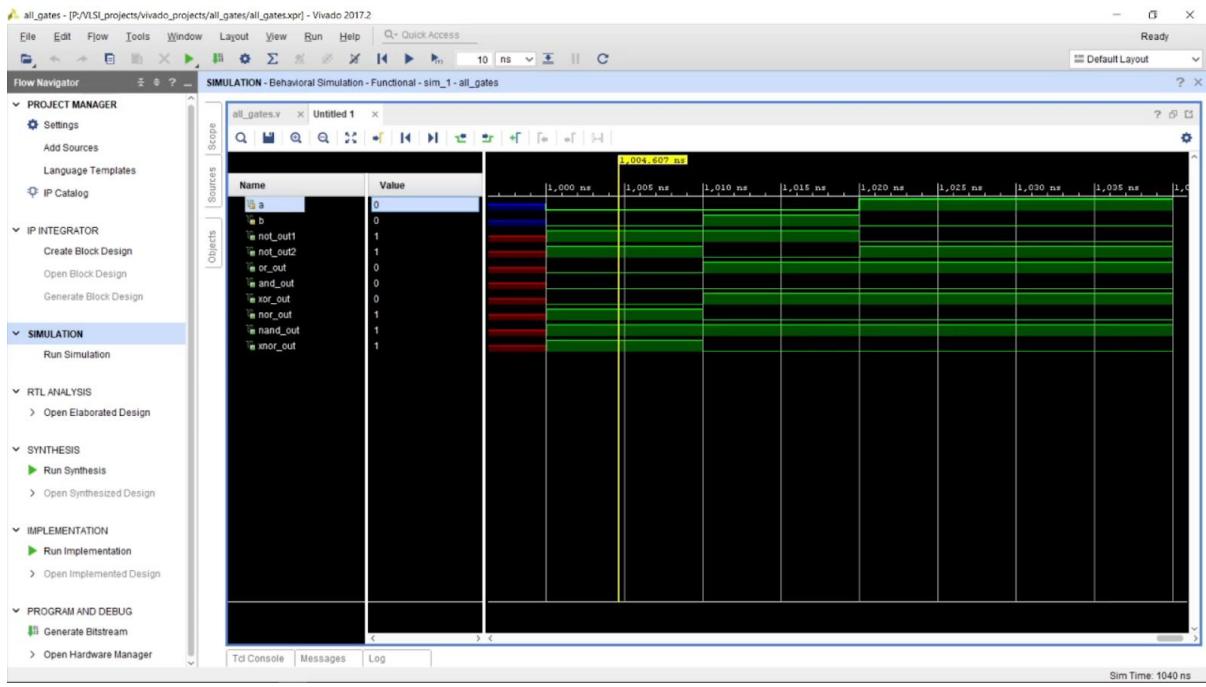
We can choose any type of value as hexadecimal or decimal as we are only going to deal with 0 and 1, Force value for a will be 0 and b will be 1, with 10 ns .



So, this is the output with only two combinations that is a 0 and b 0 , a 0 and b 1.



Now let us observe with 4 combinations that is a with 0011 and b with 1010.



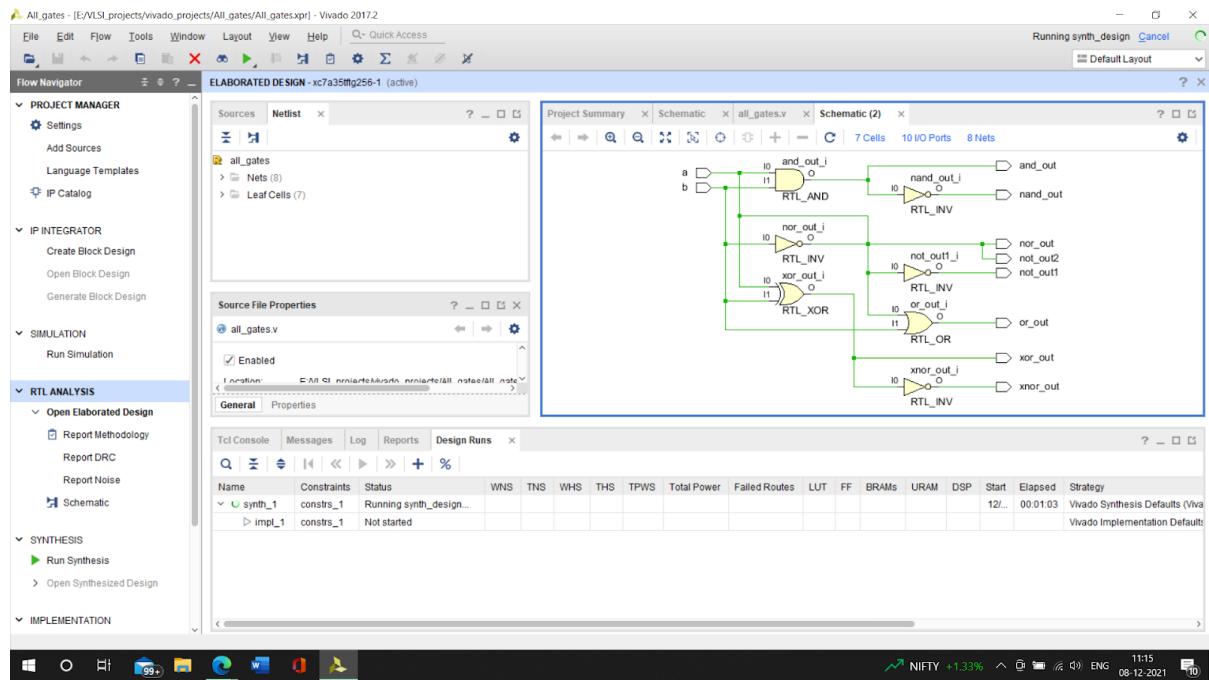
With respect to the values of a and b, and or not xor nor xnot nand output will change its value eventually

Let's come to the **HARDWARE** part!

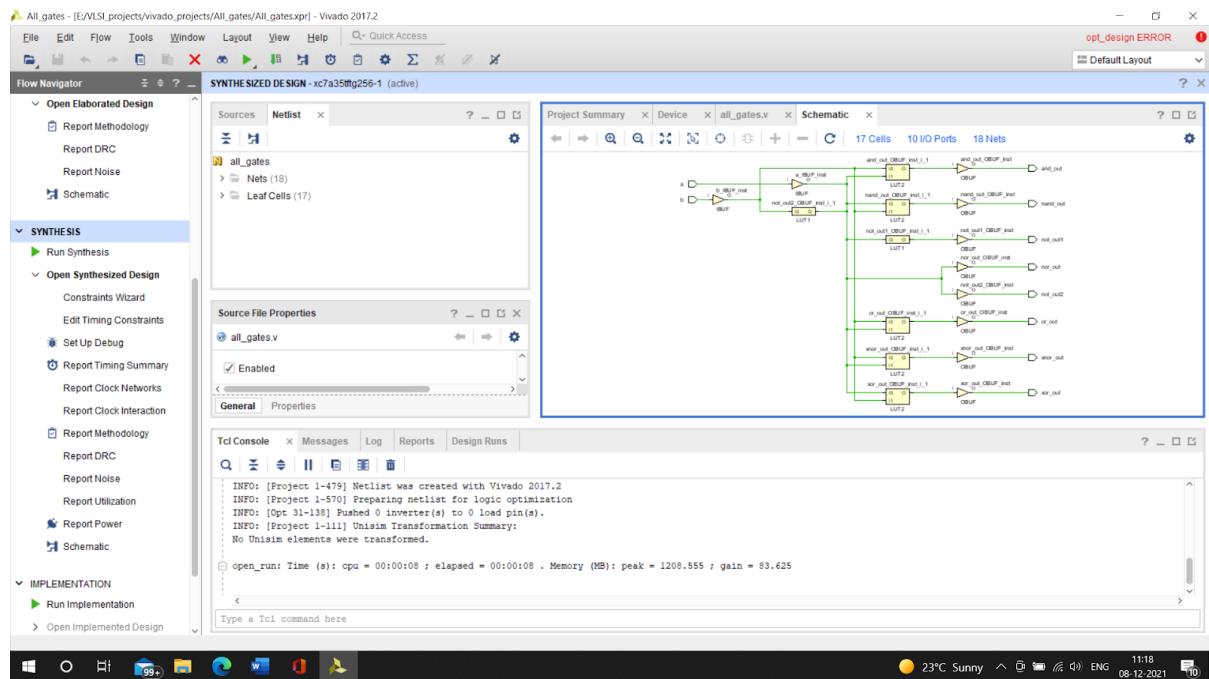
To implement it we first have to synthesis is by run synthesis , we can also elaborate the synthesis, under RTL analysis by choosing Schematic.

The RTL analysis is based on our Verilog code.

Below we can see the schematic based on our Verilog code, we got from RTL analysis



After synthesis is done, we can see the elaborated schematic of the synthesis, so here we might not be understanding with respect to basic gates or it will be having something like buffers, nothing but LUT, look up tables, with single input double output.



No, we have to bring in the pins which we need to connect, as we know we have to input as in our Verilog code with two input a and b, and when we come to the integrated circuit, we know we are using Mimas A7 mini so all we need to do is

We need to download the XDC file for vivado designs from the link
[Mimas A7 Mini FPGA Development Board | Numato Lab](#)

After downloading it open up with the notepad where we can view the XDC file and have the necessary requirements, and when I say requirements, we only need the Push button and the LEDS, and then we have to add these two constrains in VIVADO

```

#####
# QSP1 - FLASH
#####
set_property -dict { PACKAGE_PIN "P12" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { FLASH_CS_N }] ; # IO_16P_T0_FCS_B_14 Sch = FLASH_CS_N
set_property -dict { PACKAGE_PIN "P13" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { FLASH_Q[0] }] ; # IO_16P_T0_D09_MOSI_14 Sch = FLASH_D09
set_property -dict { PACKAGE_PIN "P14" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { FLASH_Q[1] }] ; # IO_16N_T0_D01_DIN_14 Sch = FLASH_D01
set_property -dict { PACKAGE_PIN "P15" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { FLASH_Q[2] }] ; # IO_16P_T0_D02_14 Sch = FLASH_D02
set_property -dict { PACKAGE_PIN "P16" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { FLASH_Q[3] }] ; # IO_16N_T0_D03_14 Sch = FLASH_D03
set_property -dict { PACKAGE_PIN "E8" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { FLASH_CLK }] ; # CCLK_0 Sch = FLASH_CLK

#####
# Push Buttons
#####
set_property -dict { PACKAGE_PIN "F5" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { sw_in[0] }] ; # IO_13P_T2_MRCC_35 Sch = SW0
set_property -dict { PACKAGE_PIN "J4" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { sw_in[1] }] ; # IO_13N_T3_VREF_35 Sch = SW1
set_property -dict { PACKAGE_PIN "M6" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { sw_in[2] }] ; # IO_13P_T3_A10_D26_14 Sch = SW2
set_property -dict { PACKAGE_PIN "N6" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { sw_in[3] }] ; # IO_13N_T3_A09_D25_VREF_14 Sch = SW3

#####
# LEDs
#####
set_property -dict { PACKAGE_PIN "K12" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LED[0] }] ; # IO_0_14 Sch = LED0
set_property -dict { PACKAGE_PIN "K13" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LED[1] }] ; # IO_15P_T0_D06_14 Sch = LED1
set_property -dict { PACKAGE_PIN "R10" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LED[2] }] ; # IO_17P_T2_A14_D30_14 Sch = LED2
set_property -dict { PACKAGE_PIN "R13" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LED[3] }] ; # IO_16P_T2_CS1_14 Sch = LED3
set_property -dict { PACKAGE_PIN "T13" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LED[4] }] ; # IO_16N_T2_A15_D31_14 Sch = LED4
set_property -dict { PACKAGE_PIN "R12" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LED[5] }] ; # IO_15P_T2_DQS_RDWR_B_14 Sch = LED5
set_property -dict { PACKAGE_PIN "T12" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LED[6] }] ; # IO_15N_T2_DQS_DOUT_CSO_B_14 Sch = LED6
set_property -dict { PACKAGE_PIN "R11" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LED[7] }] ; # IO_17N_T2_A13_D29_14 Sch = LED7

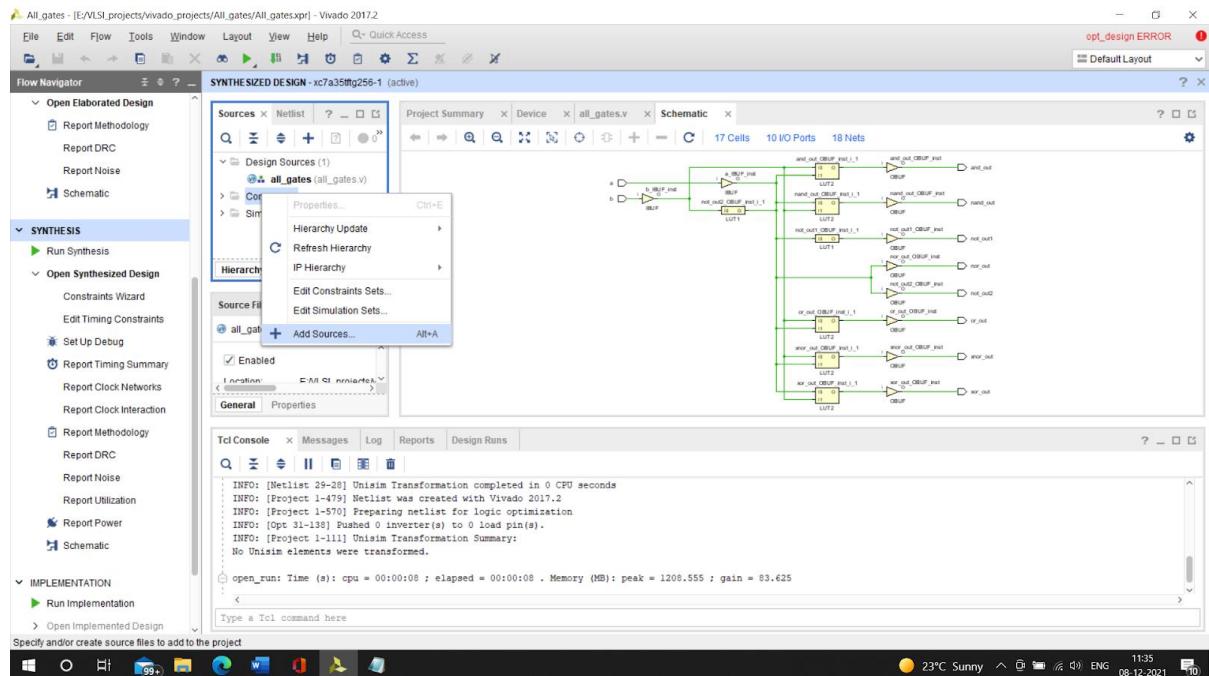
#####
# RGB LED
#####
set_property -dict { PACKAGE_PIN "M15" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LEDR }] ; # IO_13N_T0_DQ5_EMCLK_14 Sch = LED_R
set_property -dict { PACKAGE_PIN "M14" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LEDG }] ; # IO_14P_T0_D04_14 Sch = LED_G
set_property -dict { PACKAGE_PIN "M13" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LEDB }] ; # IO_14N_T0_D05_14 Sch = LED_B

#####
# Header PA
#####
set_property -dict { PACKAGE_PIN "E12" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { PA[0] }] ; # IO_13P_T2_MRCC_15 Sch = GPIO_1_P

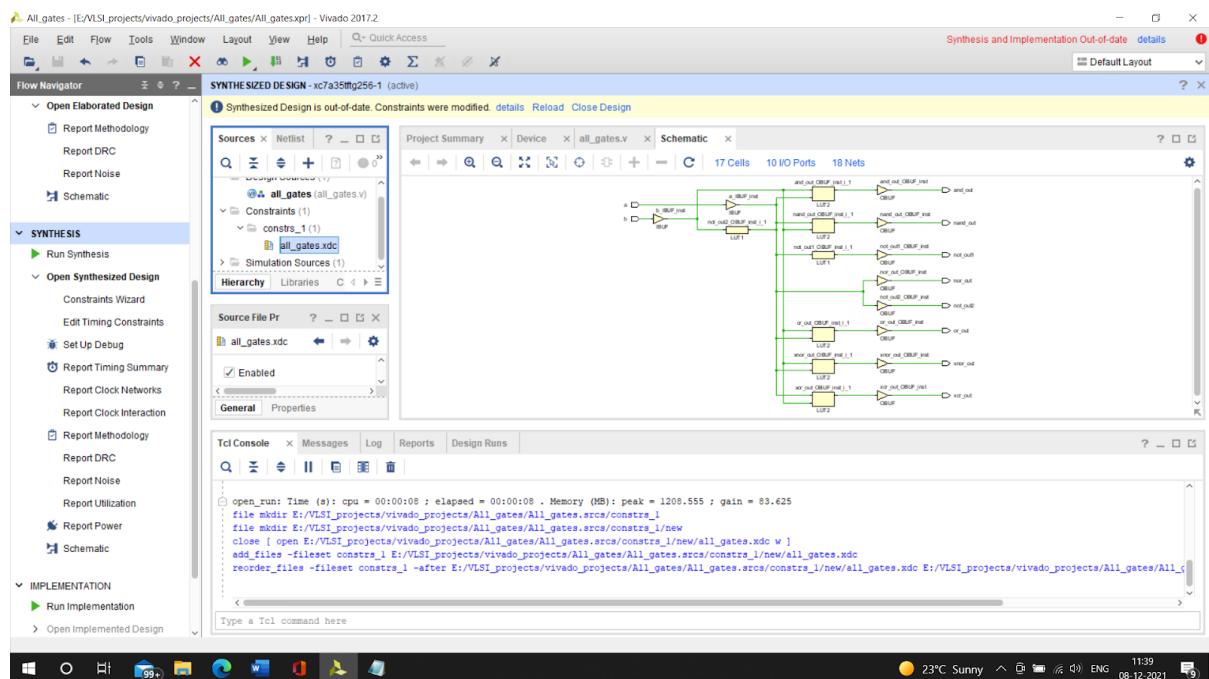
<

```

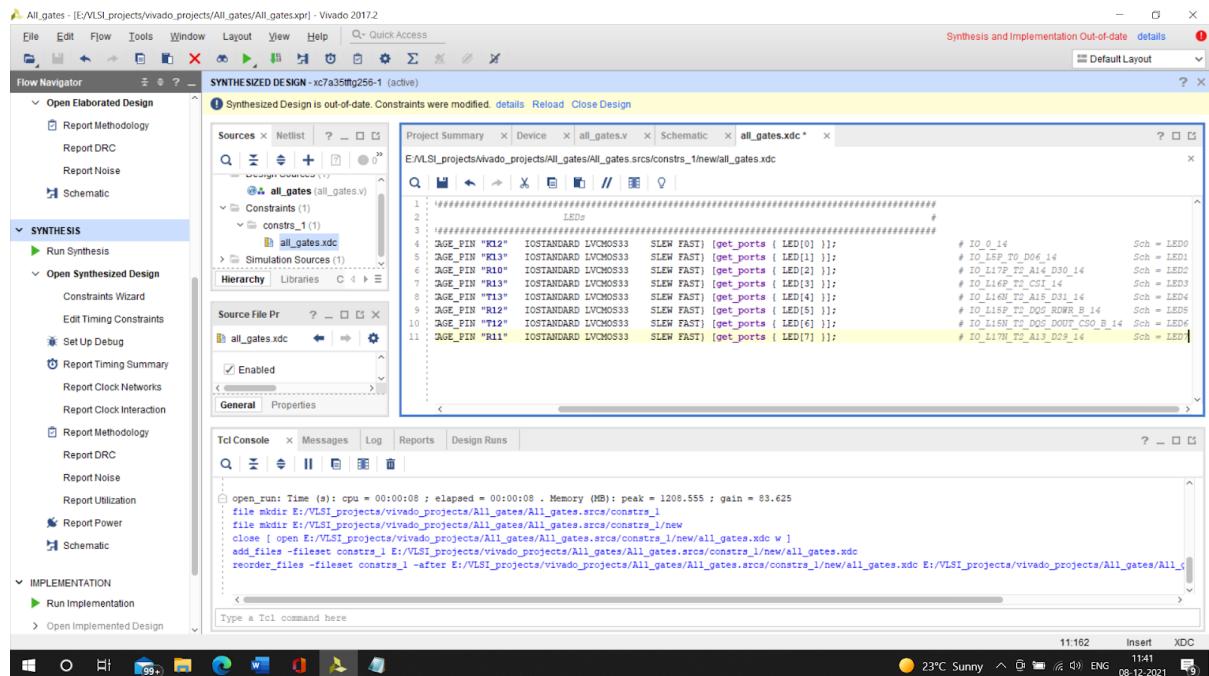
Now let's add the XDC file to vivado by adding sources to constraints, add or create constraints, then create a file name as "all_gates.xdc"



There by it is added!

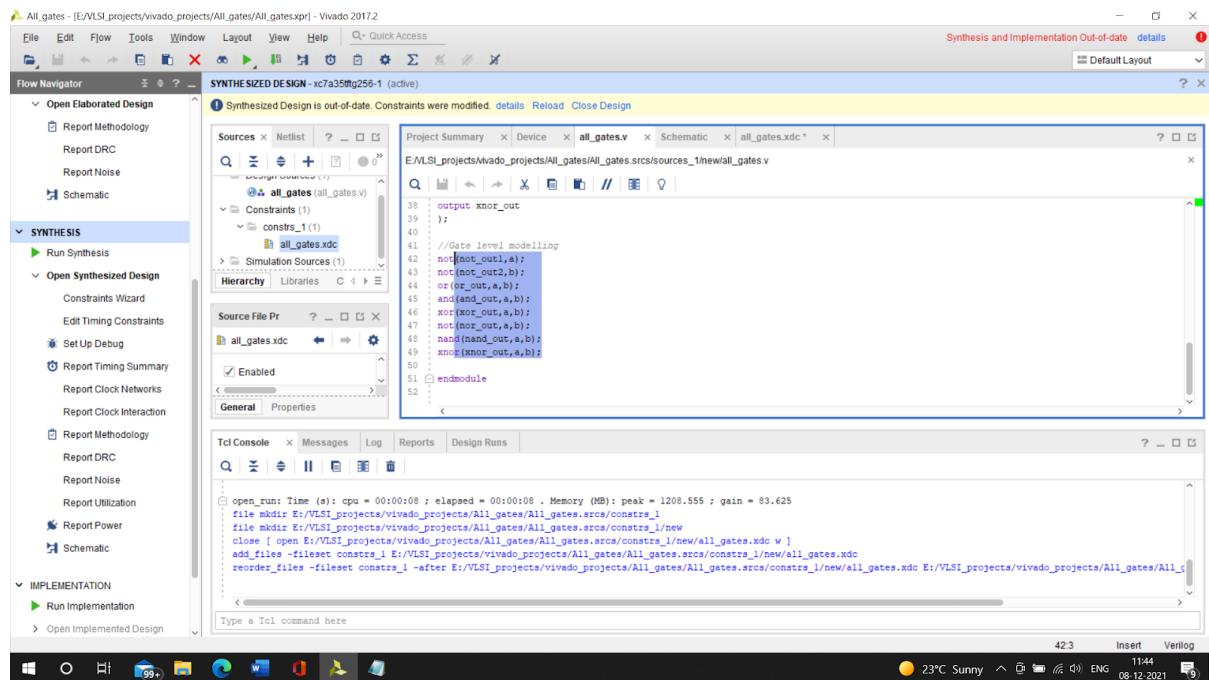


Ok so now it is time to get the xdc file which we download, copy the xdc file for the needed requirements, for us it is Leds and pushbutton and we will paste it here in the constraints file as shown below.

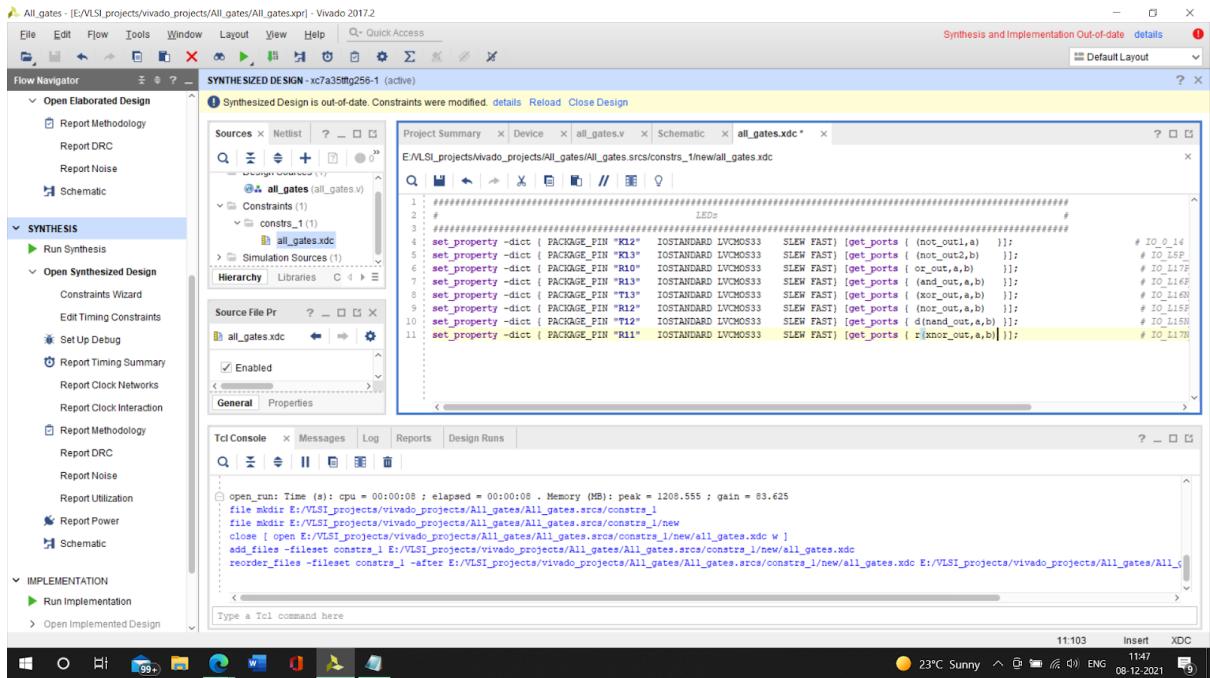


Now we have to change the identifiers in the LED pins so all we will do will take the output from our Verilog code and paste it here as shown below

Copy



Paste it here



Now coming to the switch buttons, we need two switch buttons so we will again go to the xdc file and copy the push button.

```
Mimasa7Mini - Notepad
File Edit Format View Help

#####
# OSPI - FLASH
#####
set_property dict { PACKAGE_PIN "L12" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { FLASH_CS_N }]; # IO_L6P_T0_FCS_B_14 Sch = FLASH_CS_N
set_property dict { PACKAGE_PIN "J13" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { FLASH_DQ[0] }]; # IO_L1P_T0_D00_MOSI_14 Sch = FLASH_DQ0
set_property dict { PACKAGE_PIN "J14" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { FLASH_DQ[1] }]; # IO_L1N_T0_D01_DIN_14 Sch = FLASH_DQ1
set_property dict { PACKAGE_PIN "K15" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { FLASH_DQ[2] }]; # IO_L2P_T0_D02_14 Sch = FLASH_DQ2
set_property dict { PACKAGE_PIN "K16" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { FLASH_DQ[3] }]; # IO_L2N_T0_D03_14 Sch = FLASH_DQ3
set_property dict { PACKAGE_PIN "E0" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { FLASH_CLK }]; # CCLK_0 Sch = FLASH_CLK

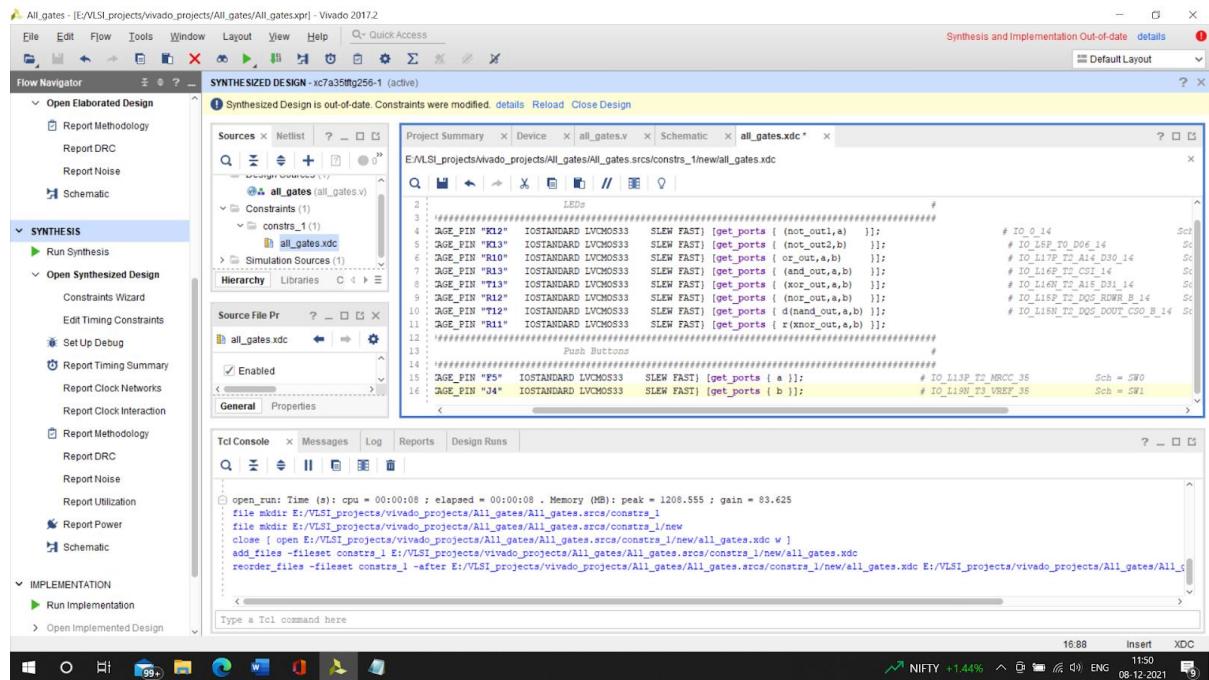
#####
# Push Buttons
#####
set_property dict { PACKAGE_PIN "F5" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { sw_in[0] }]; # IO_L13P_T2_MRCC_35 Sch = SW0
set_property dict { PACKAGE_PIN "J4" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { sw_in[1] }]; # IO_L19N_T3_VREF_35 Sch = SW1
set_property dict { PACKAGE_PIN "M6" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { sw_in[2] }]; # IO_L19P_T3_A10_D26_14 Sch = SW2
set_property dict { PACKAGE_PIN "N6" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { sw_in[3] }]; # IO_L19N_T3_A09_D25_VREF_14 Sch = SW3

#####
# LEDs
#####
set_property dict { PACKAGE_PIN "K12" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LED[0] }]; # IO_O_14 Sch = LED0
set_property dict { PACKAGE_PIN "K13" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LED[1] }]; # IO_L5P_T0_D06_14 Sch = LED1
set_property dict { PACKAGE_PIN "R10" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LED[2] }]; # IO_L17P_T2_A14_D30_14 Sch = LED2
set_property dict { PACKAGE_PIN "R13" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LED[3] }]; # IO_L16P_T2_CS1_14 Sch = LED3
set_property dict { PACKAGE_PIN "T13" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LED[4] }]; # IO_L16N_T2_A15_D31_14 Sch = LED4
set_property dict { PACKAGE_PIN "R12" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LED[5] }]; # IO_L15P_T2_DQS_RDWR_B_14 Sch = LED5
set_property dict { PACKAGE_PIN "T12" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LED[6] }]; # IO_L15N_T2_DQS_DOUT_CSO_B_14 Sch = LED6
set_property dict { PACKAGE_PIN "R11" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LED[7] }]; # IO_L17N_T2_A13_D29_14 Sch = LED7

#####
# RGB LED
#####
set_property dict { PACKAGE_PIN "M15" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LEDR }]; # IO_L3N_T0_DQS_ENMCCLK_14 Sch = LED_R
set_property dict { PACKAGE_PIN "L14" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LEDG }]; # IO_L4P_T0_D04_14 Sch = LED_G
set_property dict { PACKAGE_PIN "M14" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { LEDB }]; # IO_L4N_T0_D05_14 Sch = LED_B

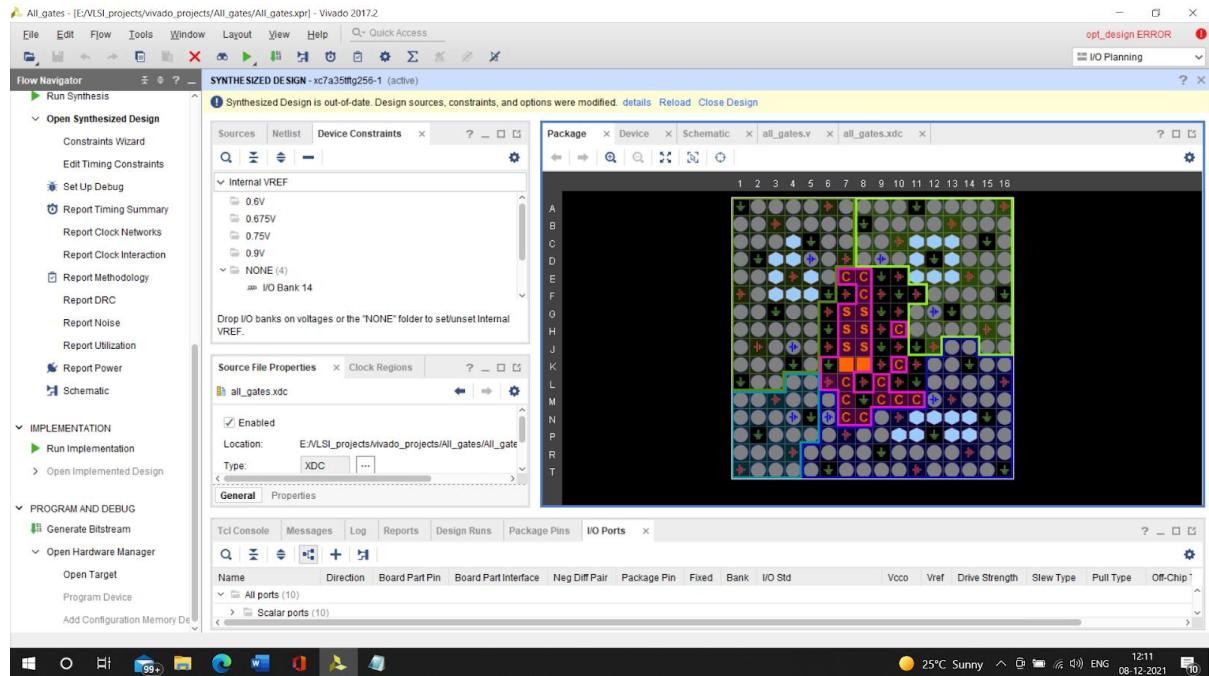
#####
# Header P4
#####
set_property dict { PACKAGE_PIN "E12" IOSTANDARD LVCMS33 SLEW FAST } [get_ports { P4[0] }]; # IO_L13P_T2_MRCC_15 Sch = GPIO_1_P
```

And again, paste it in the added constraints in vivado.

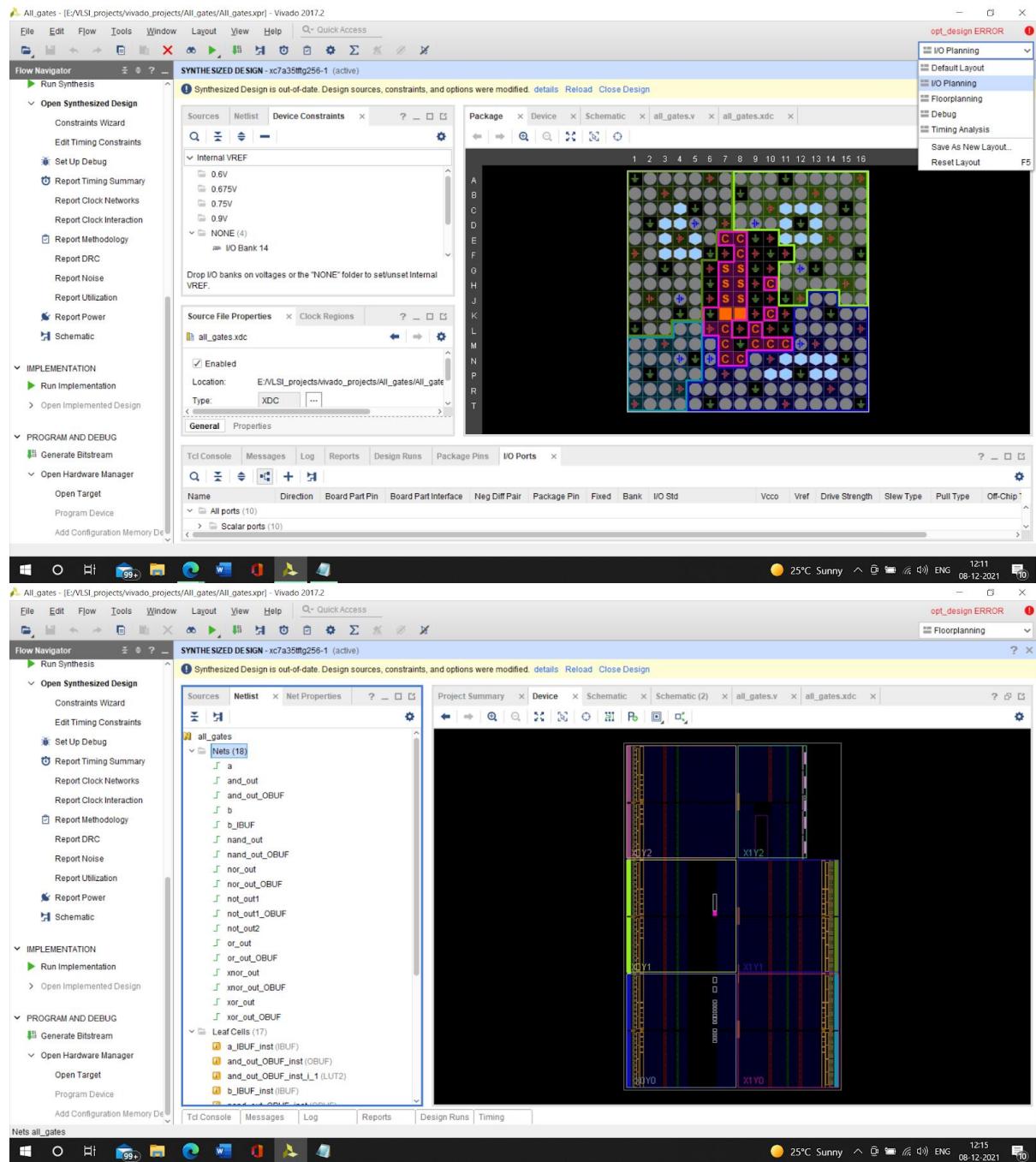


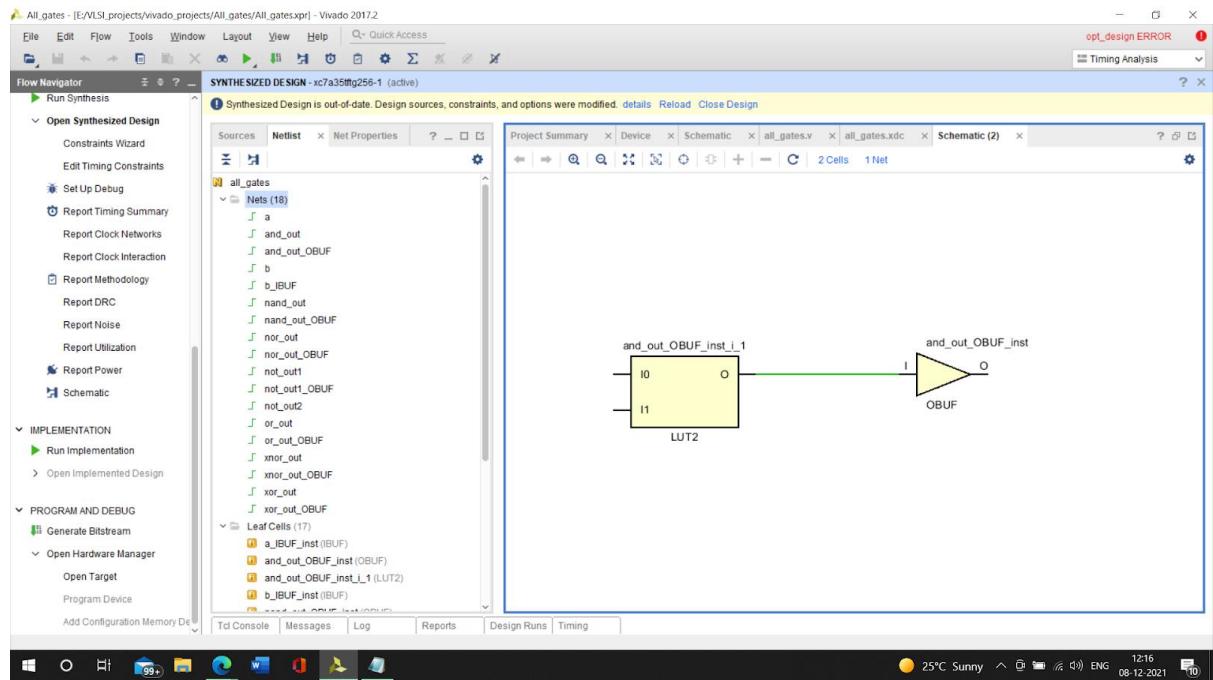
After that we under program and debug we have to choose generate bitstream

After running, I/O planning is what we can see.



Equivalently we can deal with floor mapping, timing constraints





Now if we want to connect it with the real time Mimas mini A7, firstly we have to connect it with our pc/laptop with the FPGA, then through Tenegra application we have to synch vivado, we have to automatically connect to the hardware by vivado and add a local server, with a local host, then it will be auto connected and again we have to choose program and debug option to upload it in the FPGA.

2. UART communication to print a single character

Software to download

- Notepad++

Step 1- Write the verilog program in notepad ++ .

```

*EVLISI.projects\ivado_projects\uart_tx.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
uart_tx.v
1 //Module declaration
2 module uart_transmit(
3   input clk,           //100MHz
4   input rst,
5   input uart_en,
6   //outputs
7   output [1:0] uart_tx,
8   output uart_busy
9 );
10 //Identifier Declaration
11 reg [3:0] bitcount;
12 reg [3:0] shift;
13 reg [1:0] data = "00";
14 wire sending;
15 //Baud rate as 115200Hz
16 reg [28:0] clk_count;
17 wire [28:0] clk_inc_count = clk_count[28] ? (115200) : (115200 - 10000000);
18 wire [28:0] clk_inc = clk_count + clk_inc_count;
19 wire gen_clk;
20
21 //Generating clock
22 always#(posedge clk)
23   clk_count += clk_inc;
24
25 assign gen_clk = <clk_count[28]; //115200Hz
26 assign uart_busy = [bitcount[3:1];
27 assign sending = [bitcount;
28 //Uart Transmission block
29 always#(posedge clk) begin
30   if(rst) begin
31     uart_tx = 1'nl;
32     bitcount <= 0;
33     shift <= 0;
34   end else begin
35     //Shifter init
36     if(uart_en & ~uart_busy) begin
37       bitcount <= (1 + 8 + 1 + 1);
38       shift <= (data[1:0], 1'b0);
39     end
40   //Data aTransmission
41   if(sending & gen_clk) begin
42     (shift, uart_tx) <= ( 1'h1, shift);
43     bitcount <= bitcount - 1;
44   end
45 end
46
47 endmodule

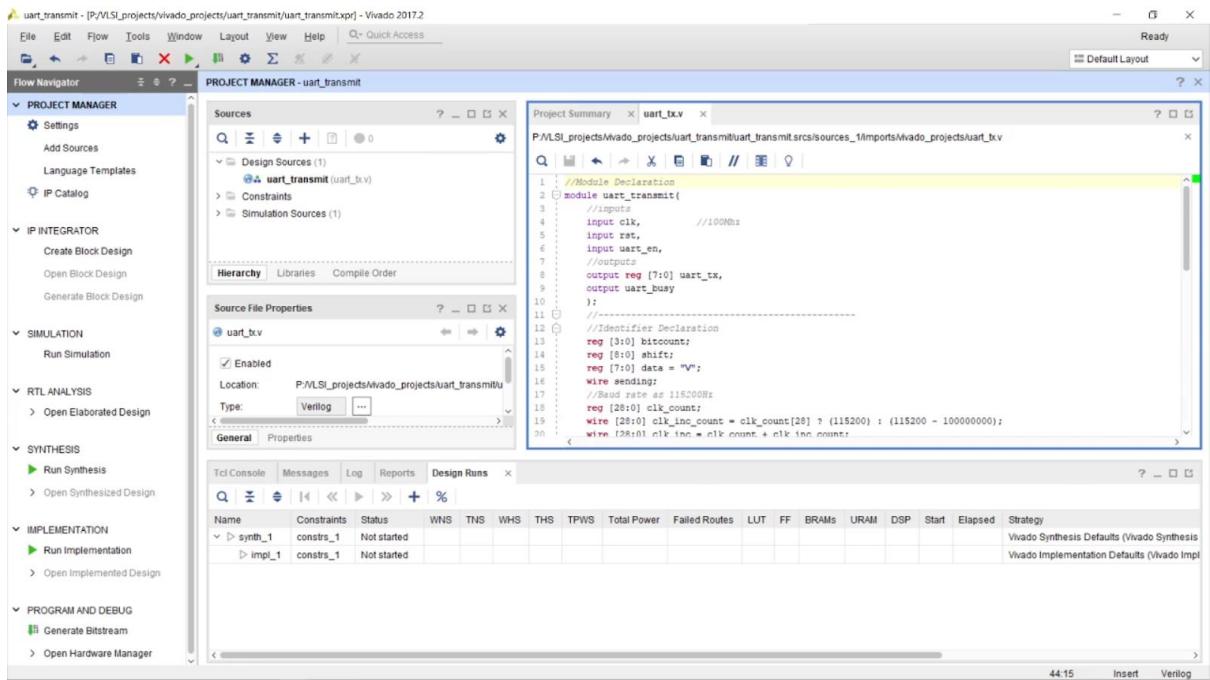
```

length: 1,102 lines: 54 ln: 22 Col: 19 Pos: 486 Windows (CR LF) UTF-8 INS

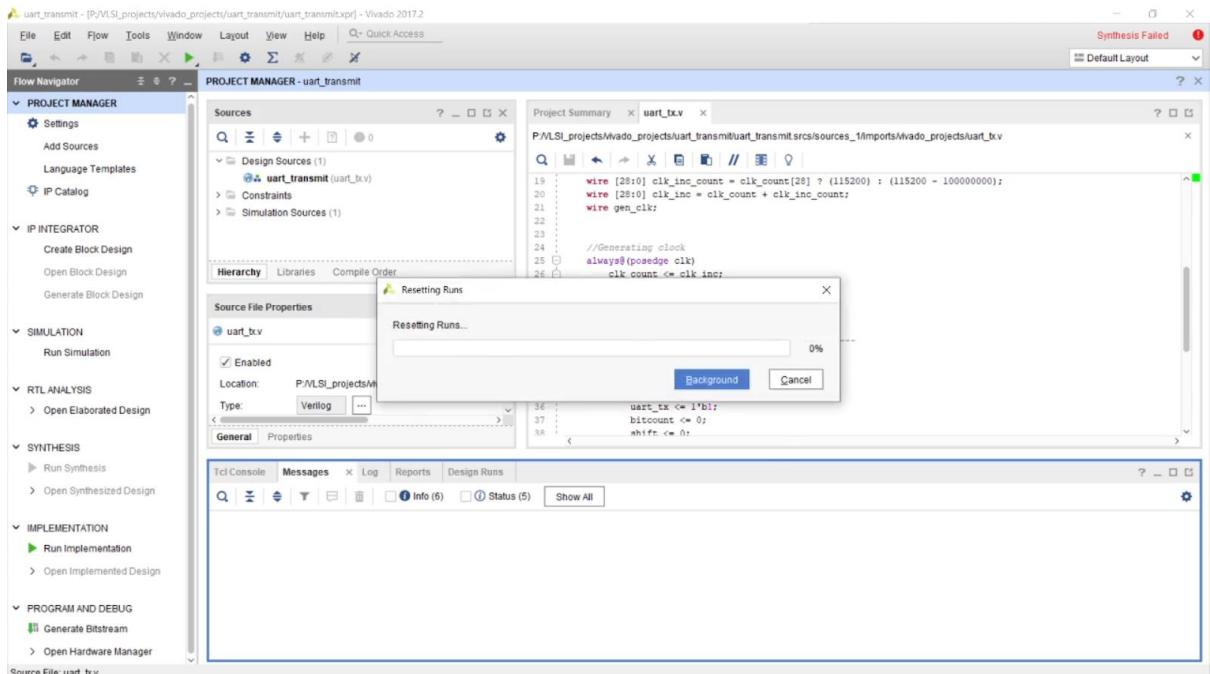
Verilog file

75°F Haze 11:19 15-12-2021 ENG

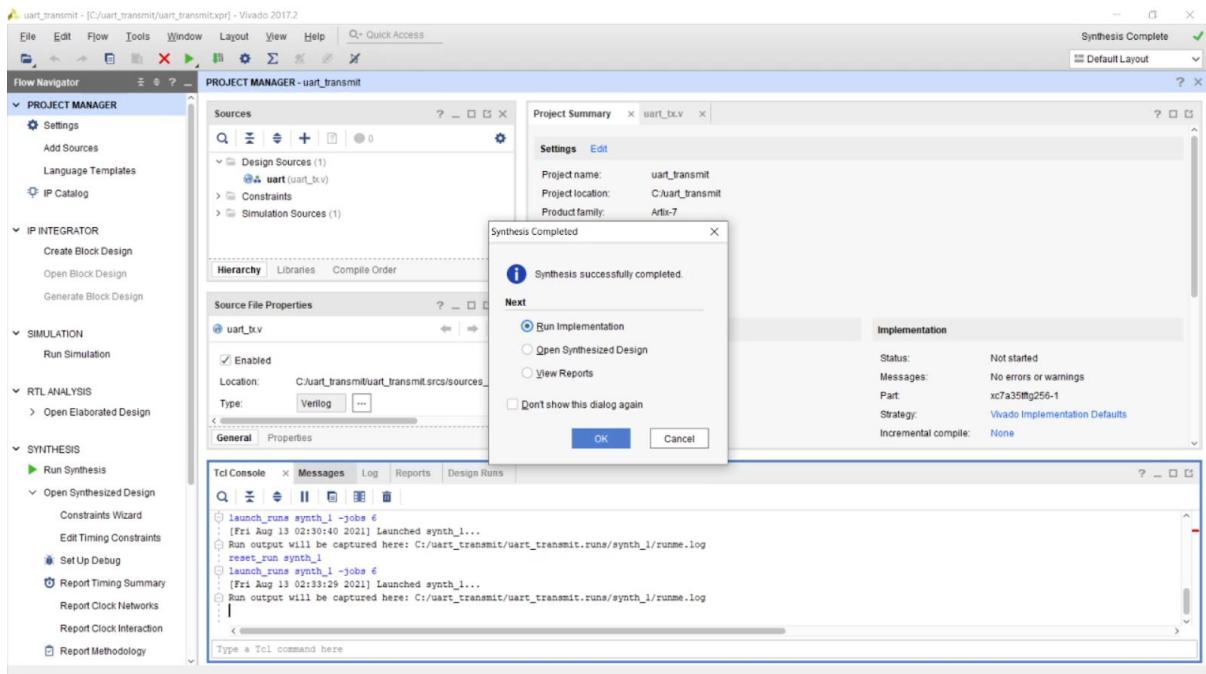
Step 2-Open Vivado software create a new project and add this notepad verilog extension to it



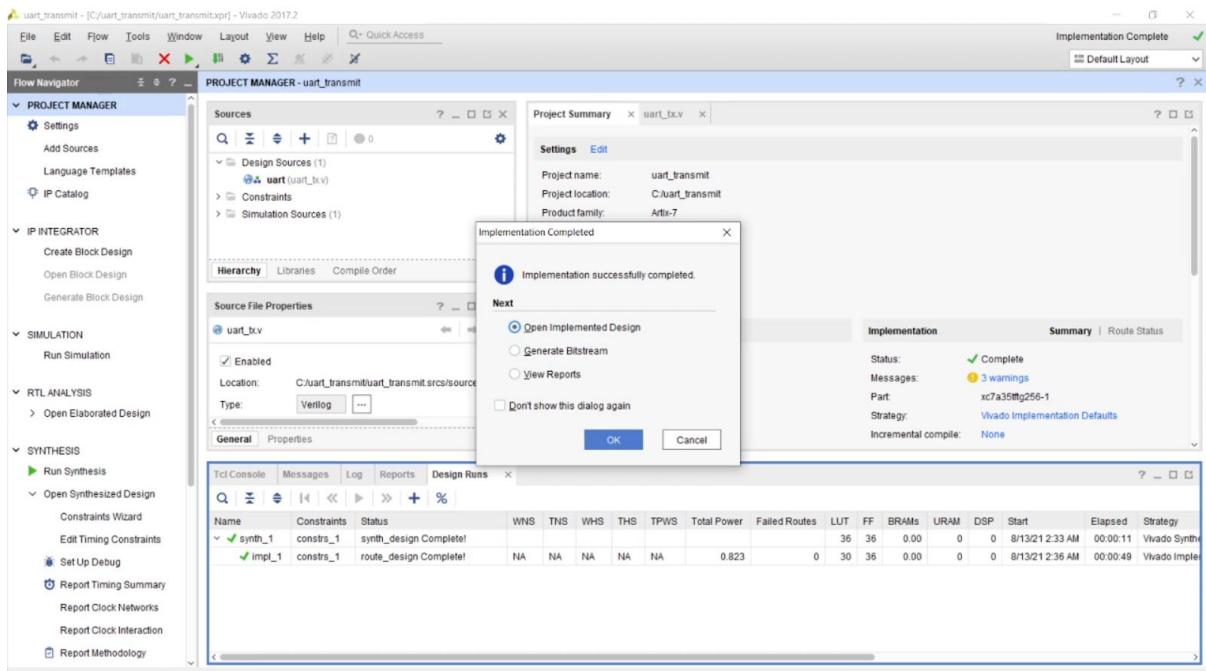
Synthesis the code extracted and compile the errors if there are any.



After synthesis run implementation



Open synthesis implemented design



Now one of the most vital part we have to change port design with respect to the XDC file we got for vivado designs from the link [Mimas A7 Mini FPGA Development Board | Numato Lab](#)

After downloading it open up with the notepad where we can view the XDC file

```

MimasA7Mini - Notepad
File Edit Format View Help
#####
set_property -dict { PACKAGE_PIN "L12" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { FLASH_CS_N }] ; # IO_L1P_T0_FCS_B_14 Sch = FLASH_CS_N
set_property -dict { PACKAGE_PIN "J13" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { FLASH_DQ[0] }] ; # IO_L1P_T0_D00_MOSI_14 Sch = FLASH_DQ0
set_property -dict { PACKAGE_PIN "J14" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { FLASH_DQ[1] }] ; # IO_L1N_TO_D01_DIN_14 Sch = FLASH_DQ1
set_property -dict { PACKAGE_PIN "K15" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { FLASH_DQ[2] }] ; # IO_L2P_T0_D02_14 Sch = FLASH_DQ2
set_property -dict { PACKAGE_PIN "K16" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { FLASH_DQ[3] }] ; # IO_L2N_TO_D03_14 Sch = FLASH_DQ3
set_property -dict { PACKAGE_PIN "E8" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { FLASH_CLK }] ; # CCLK_0 Sch = FLASH_CLK
#####
# Push Buttons #
#####
set_property -dict { PACKAGE_PIN "F5" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { sw_in[0] }]; # IO_L1P_T2_MRCC_35 Sch = SW0
set_property -dict { PACKAGE_PIN "J4" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { sw_in[1] }]; # IO_L19N_T3_VREF_35 Sch = SW1
set_property -dict { PACKAGE_PIN "M6" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { sw_in[2] }]; # IO_L19P_T3_A10_D26_14 Sch = SW2
set_property -dict { PACKAGE_PIN "N6" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { sw_in[3] }]; # IO_L19N_T3_A09_D25_VREF_14 Sch = SW3
#####
LEDs #
#####
set_property -dict { PACKAGE_PIN "K12" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { LED[0] }]; # IO_O_14 Sch = LED0
set_property -dict { PACKAGE_PIN "K13" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { LED[1] }]; # IO_L5P_T0_D06_14 Sch = LED1
set_property -dict { PACKAGE_PIN "R10" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { LED[2] }]; # IO_L17P_T2_A14_D30_14 Sch = LED2
set_property -dict { PACKAGE_PIN "R13" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { LED[3] }]; # IO_L16P_T2_CSI_14 Sch = LED3
set_property -dict { PACKAGE_PIN "T13" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { LED[4] }]; # IO_L16N_T2_A15_D31_14 Sch = LED4
set_property -dict { PACKAGE_PIN "R12" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { LED[5] }]; # IO_L15P_T2_DQS_RDWR_B_14 Sch = LED5
set_property -dict { PACKAGE_PIN "T12" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { LED[6] }]; # IO_L15N_T2_DQS_DOUT_CSO_B_14 Sch = LED6
set_property -dict { PACKAGE_PIN "R11" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { LED[7] }]; # IO_L17N_T2_A13_D29_14 Sch = LED7
#####
RGB LED #
#####
set_property -dict { PACKAGE_PIN "M15" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { LEDR }]; # IO_L3N_TO_DQS_EMCCCLK_14 Sch = LED_R
set_property -dict { PACKAGE_PIN "L14" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { LEDG }]; # IO_L4P_TO_D04_14 Sch = LED_G
set_property -dict { PACKAGE_PIN "M14" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { LEDB }]; # IO_L4N_TO_D05_14 Sch = LED_B
#####
Header P4 #
#####
set_property -dict { PACKAGE_PIN "E12" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { P4[0] }]; # IO_L13P_T2_MRCC_15 Sch = GPIO_1_P
set_property -dict { PACKAGE_PIN "E13" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { P4[1] }]; # IO_L13N_T2_MRCC_15 Sch = GPIO_1_N

```

```

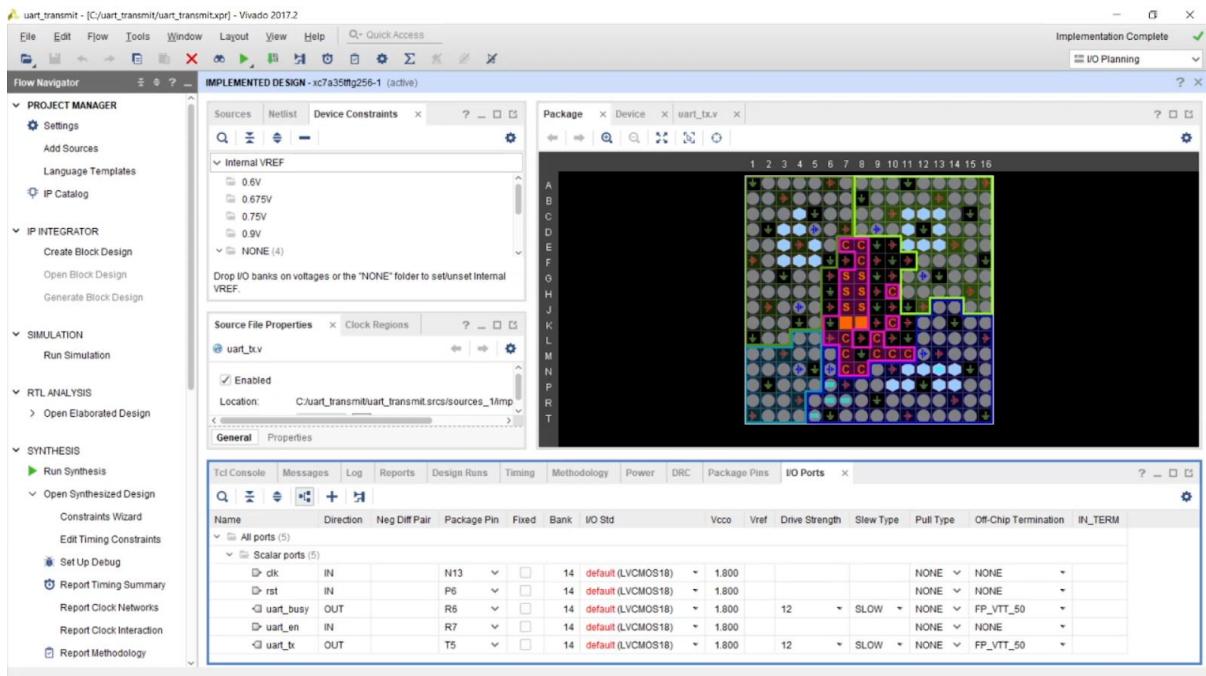
Ln 1, Col 1 60% Windows (CRLF) UTF-8

MimasA7Mini - Notepad
File Edit Format View Help
set_property CFGBVS VCCO [current_design]
set_property CONFIG_VOLTAGE 3.3 [current_design]
#####
CLOCK 100MHz #
#####
set_property -dict { PACKAGE_PIN "N11" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { CLK1 }]; # IO_L13P_T2_MRCC_14 Sch = CLK
#####
FT232H Signals #
#####
set_property -dict { PACKAGE_PIN "M16" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { DATA[0] }]; # IO_L7P_T1_D09_14 Sch = FTDI_D0
set_property -dict { PACKAGE_PIN "N16" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { DATA[1] }]; # IO_L7N_T1_D10_14 Sch = FTDI_D1
set_property -dict { PACKAGE_PIN "P15" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { DATA[2] }]; # IO_L8P_T1_D11_14 Sch = FTDI_D2
set_property -dict { PACKAGE_PIN "P16" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { DATA[3] }]; # IO_L8N_T1_D12_14 Sch = FTDI_D3
set_property -dict { PACKAGE_PIN "R15" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { DATA[4] }]; # IO_L9P_T1_DQS_14 Sch = FTDI_D4
set_property -dict { PACKAGE_PIN "R16" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { DATA[5] }]; # IO_L9N_T1_DQS_D13_14 Sch = FTDI_D5
set_property -dict { PACKAGE_PIN "T14" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { DATA[6] }]; # IO_L10P_T1_D14_14 Sch = FTDI_D6
set_property -dict { PACKAGE_PIN "T15" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { DATA[7] }]; # IO_L10N_T1_D15_14 Sch = FTDI_D7
#####
set_property -dict { PACKAGE_PIN "T8" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { TXE_N }]; # IO_L21N_T3_DQS_A06_D22_14 Sch = FTDI_TXE_N
set_property -dict { PACKAGE_PIN "T7" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { RXE_N }]; # IO_L21P_T3_DQS_14 Sch = FTDI_RXE_N
set_property -dict { PACKAGE_PIN "P14" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { WR_N }]; # IO_L12N_T1_MRCC_14 Sch = FTDI_WR_N
set_property -dict { PACKAGE_PIN "N12" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { RD_N }]; # IO_L13N_T2_MRCC_14 Sch = FTDI_RD_N
set_property -dict { PACKAGE_PIN "N14" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { CLKOUT}]; # IO_L12P_T1_MRCC_14 Sch = FTDI_CLKOUT
set_property -dict { PACKAGE_PIN "L15" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { OE_N }]; # IO_L3P_TO_DQS_PUDC_B_14 Sch = FTDI_OE#
set_property -dict { PACKAGE_PIN "M12" IOSTANDARD LVCMOS33 SLEW FAST } [get_ports { SIWUA }]; # IO_L6N_TO_D08_VREF_14 Sch = FTDI_SIWUA
#####
DDR3 : MT41J128M16XX-125 #
Frequency : 400 MHz
Data Width : 16 #
#####
set_property -dict { PACKAGE_PIN "G2" IOSTANDARD SSTL15 SLEW FAST } [get_ports { ddri3_dq[0] }]; # IO_L17P_T2_35 Sch = DDR3_DQ0
set_property -dict { PACKAGE_PIN "F3" IOSTANDARD SSTL15 SLEW FAST } [get_ports { ddri3_dq[1] }]; # IO_L14N_T2_SRCC_35 Sch = DDR3_DQ1
set_property -dict { PACKAGE_PIN "H4" IOSTANDARD SSTL15 SLEW FAST } [get_ports { ddri3_dq[2] }]; # IO_L18N_T2_SRCC_35 Sch = DDR3_DQ2
set_property -dict { PACKAGE_PIN "G5" IOSTANDARD SSTL15 SLEW FAST } [get_ports { ddri3_dq[3] }]; # IO_L16P_T2_35 Sch = DDR3_DQ3
set_property -dict { PACKAGE_PIN "G1" IOSTANDARD SSTL15 SLEW FAST } [get_ports { ddri3_dq[4] }]; # IO_L17N_T2_35 Sch = DDR3_DQ4
set_property -dict { PACKAGE_PIN "F4" IOSTANDARD SSTL15 SLEW FAST } [get_ports { ddri3_dq[5] }]; # IO_L14P_T2_SRCC_35 Sch = DDR3_DQ5
set_property -dict { PACKAGE_PIN "H5" IOSTANDARD SSTL15 SLEW FAST } [get_ports { ddri3_dq[6] }]; # IO_L18P_T2_35 Sch = DDR3_DQ6

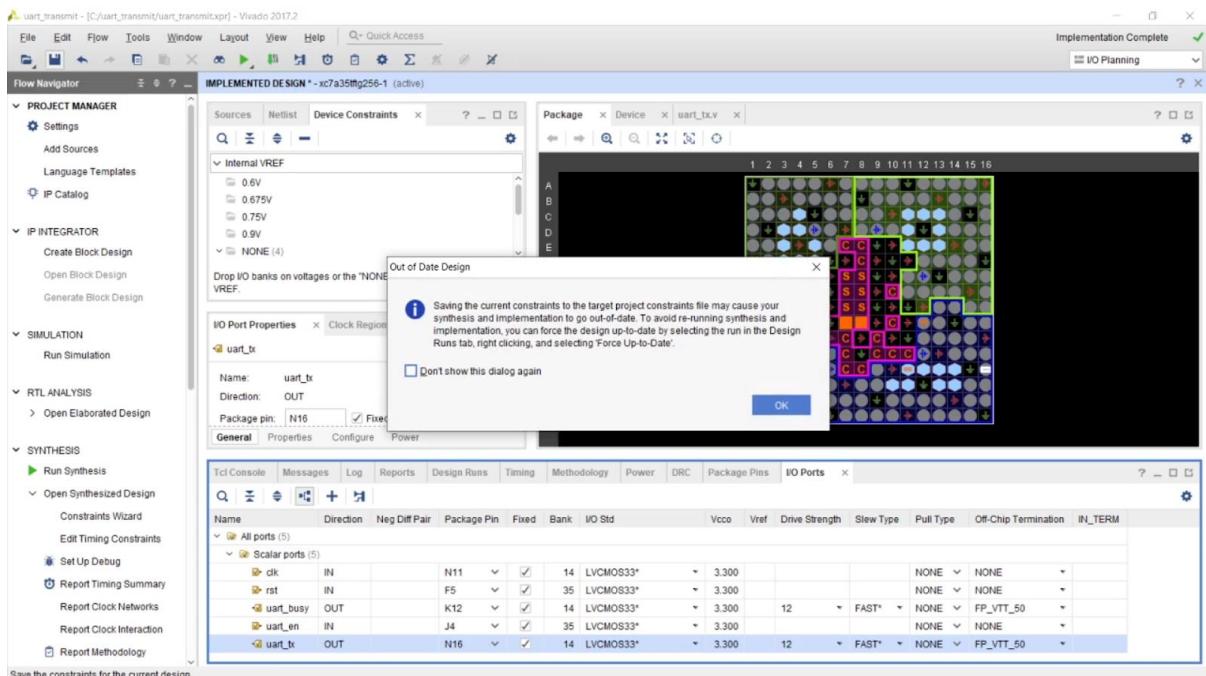
```

And have the necessary requirements, and when I say requirements, we need many ports for signals(user and programming channel), clock, Mimas A7 Mini does not have any reset we will use the push button pins,LED

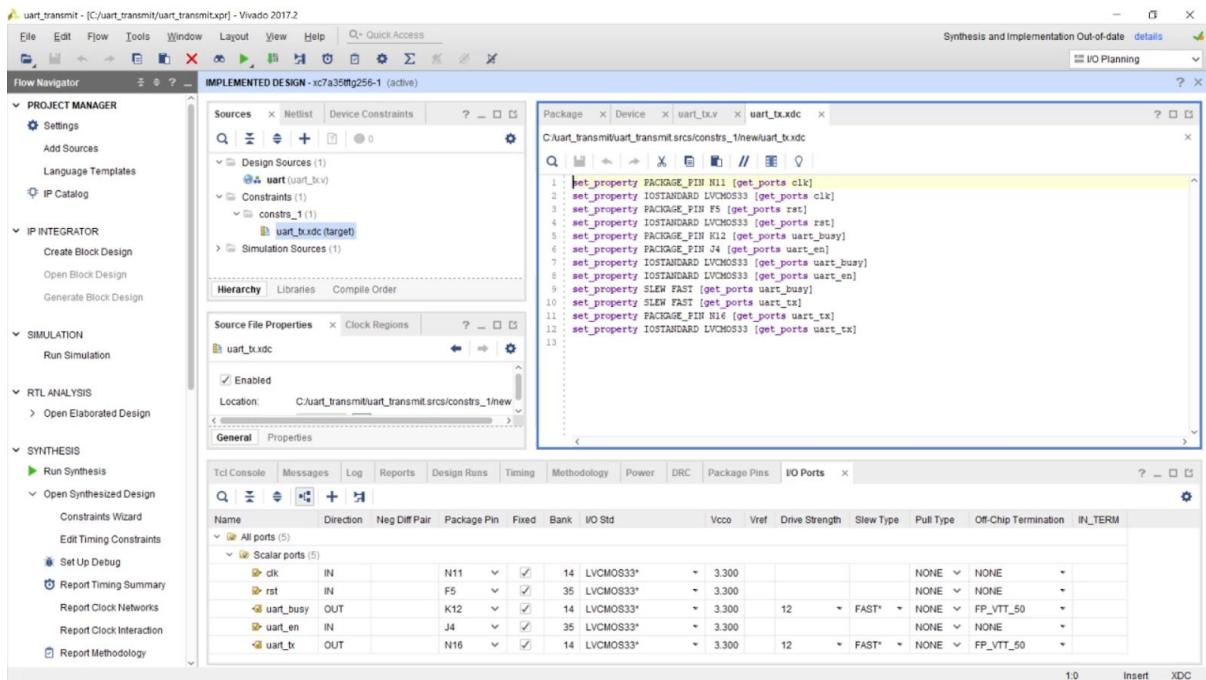
We will use low voltage CMOS, LVCMOS33



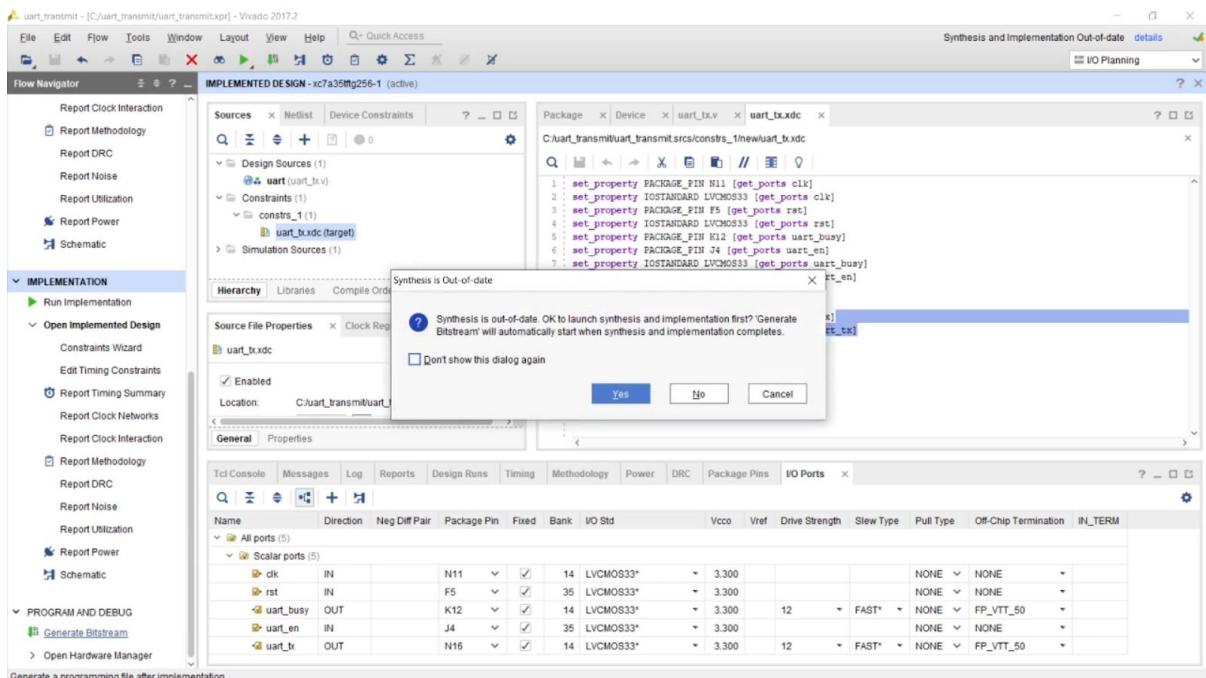
After the necessary changes save it and name the file as “uart_tx”



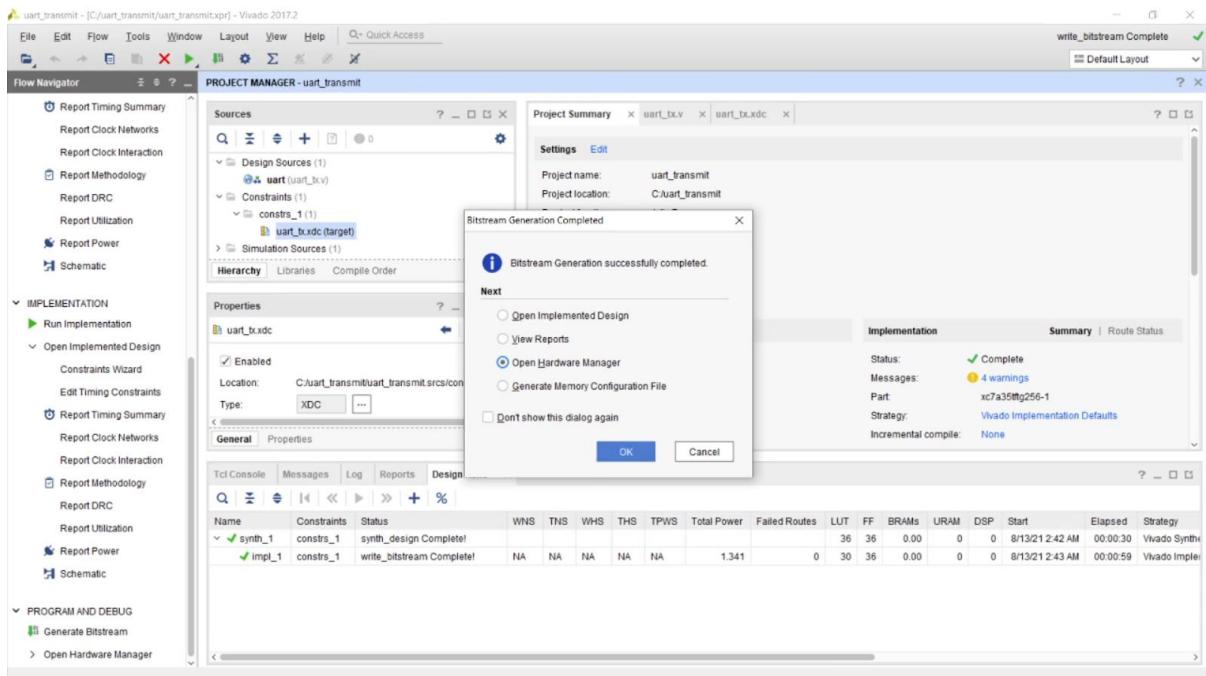
After that we can see in the source file under constrains and the I/O standards are given



Generate bit stream after that, resulting in synthesis and implementation



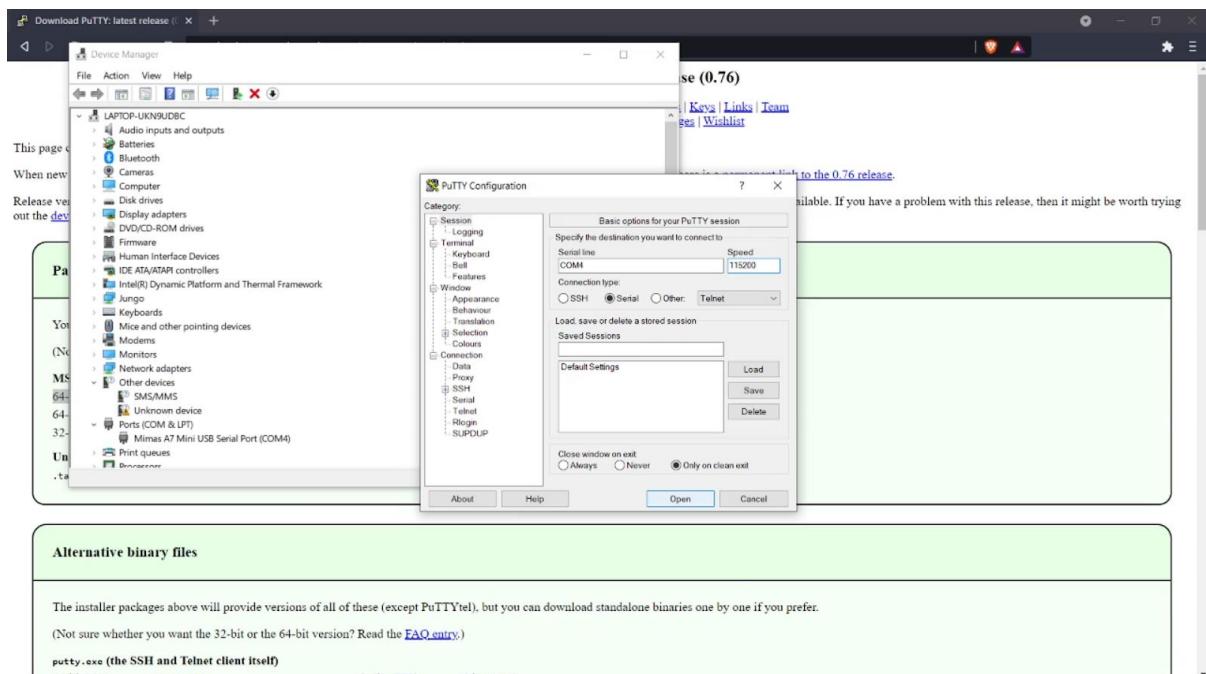
After the generation of bit stream open hardware manager



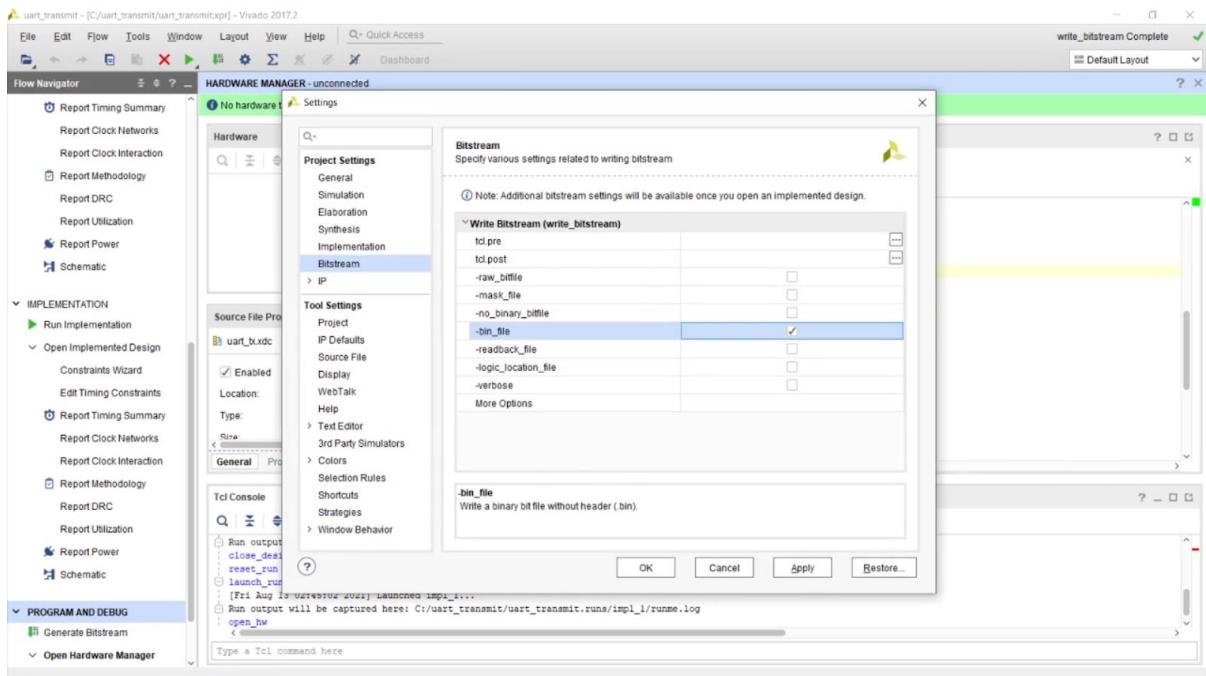
Step 3- Download a serial monitor , we will be using a light weight serial monitor software

[Download PuTTY - a free SSH and telnet client for Windows](#) link to download

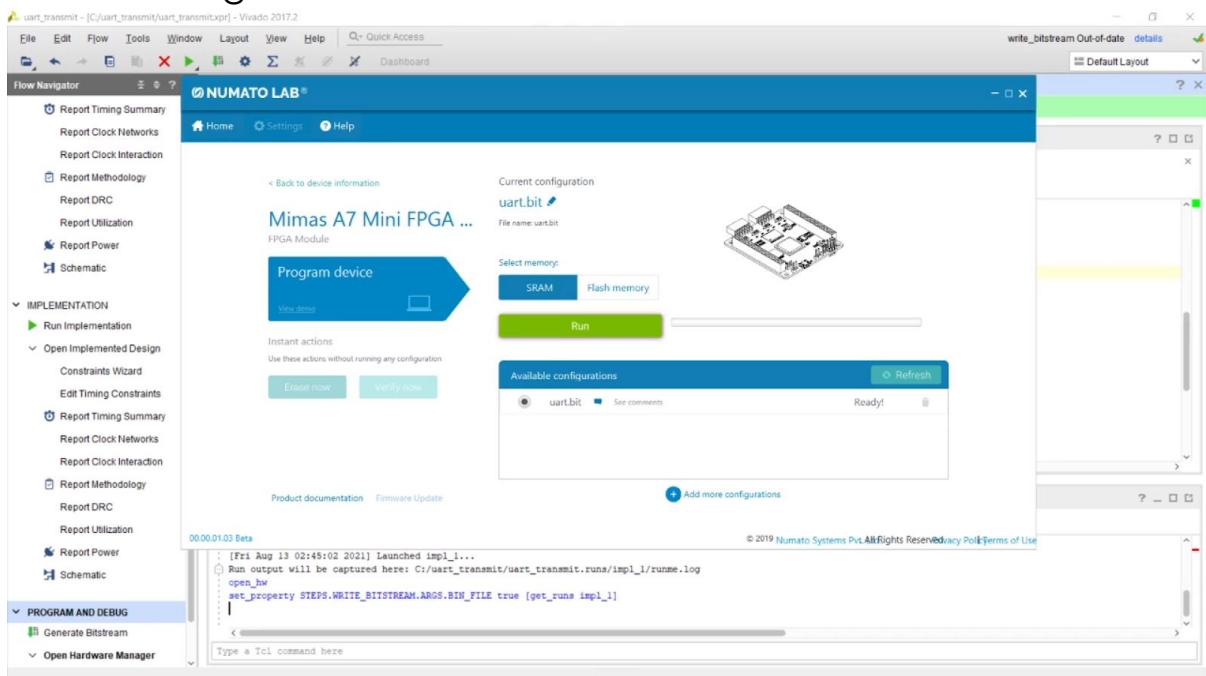
After that launch Putty with a a serial line as “COM 4” shown in device manager used in the following case , and the speed we know set as 15200, and then open the terminal.



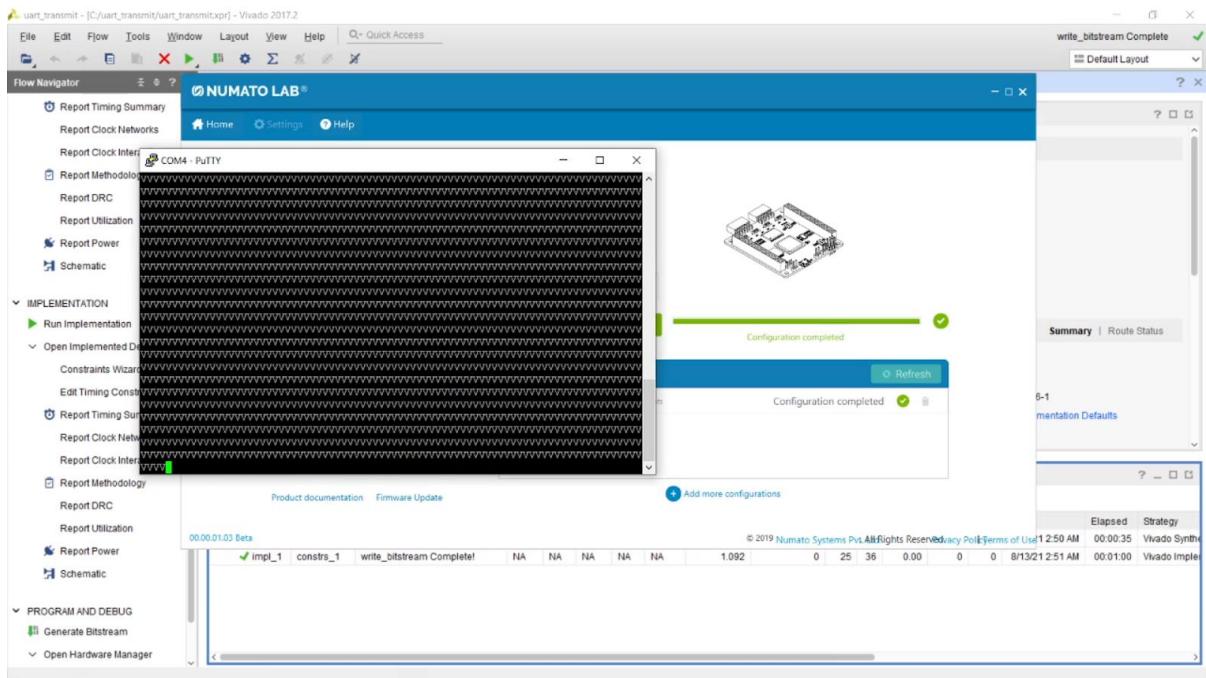
Step 4- Open Tenegra application and synch vivado bitstream file , you can go to setting and create one .



After choosing the file select SRAM and run it



As we know that with push button is enabled with the input of letter we have created so in our case push button will work that way. Every time we press the push button the letter V gets automatically printed in its own.



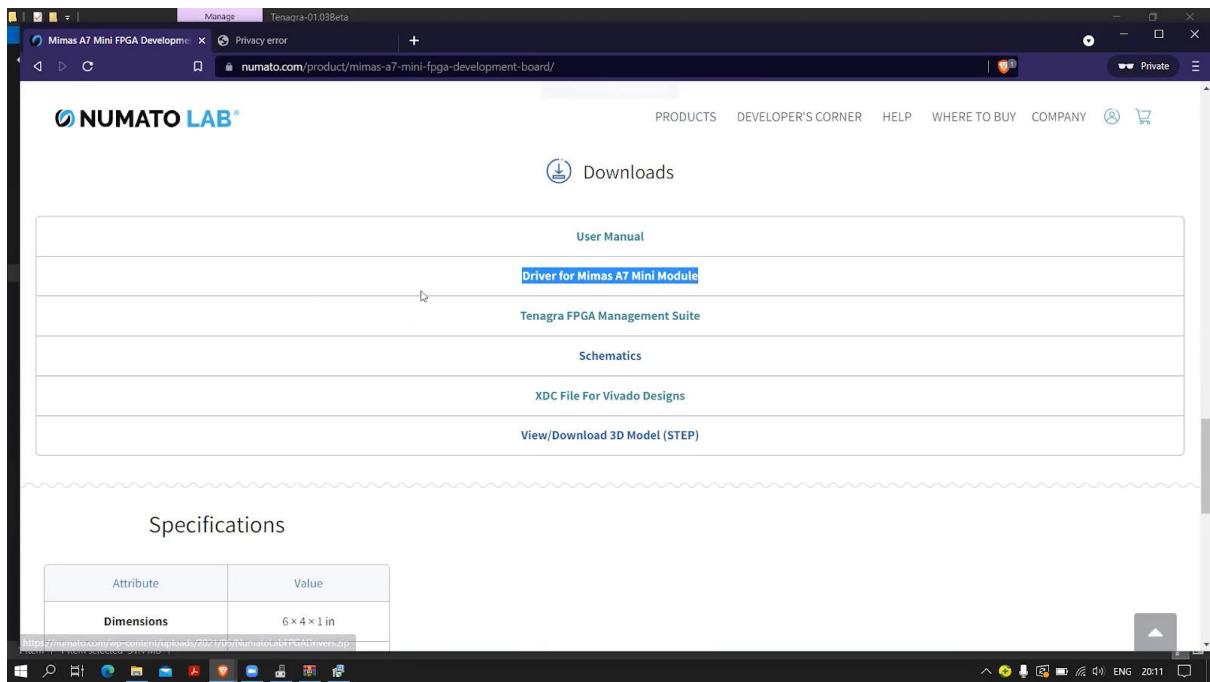
3. FSM Designs: Mealy & Moore Machines and Up Down Counter

Step 1- Connect the real time FPGA Mimas A7 mini board, open the device manager to set its configuration. Now we have to get tenegra a software which acts like a bridge between the FPGA board and the pc, where the program will be uploaded. You can get tenegra under Numato Lab

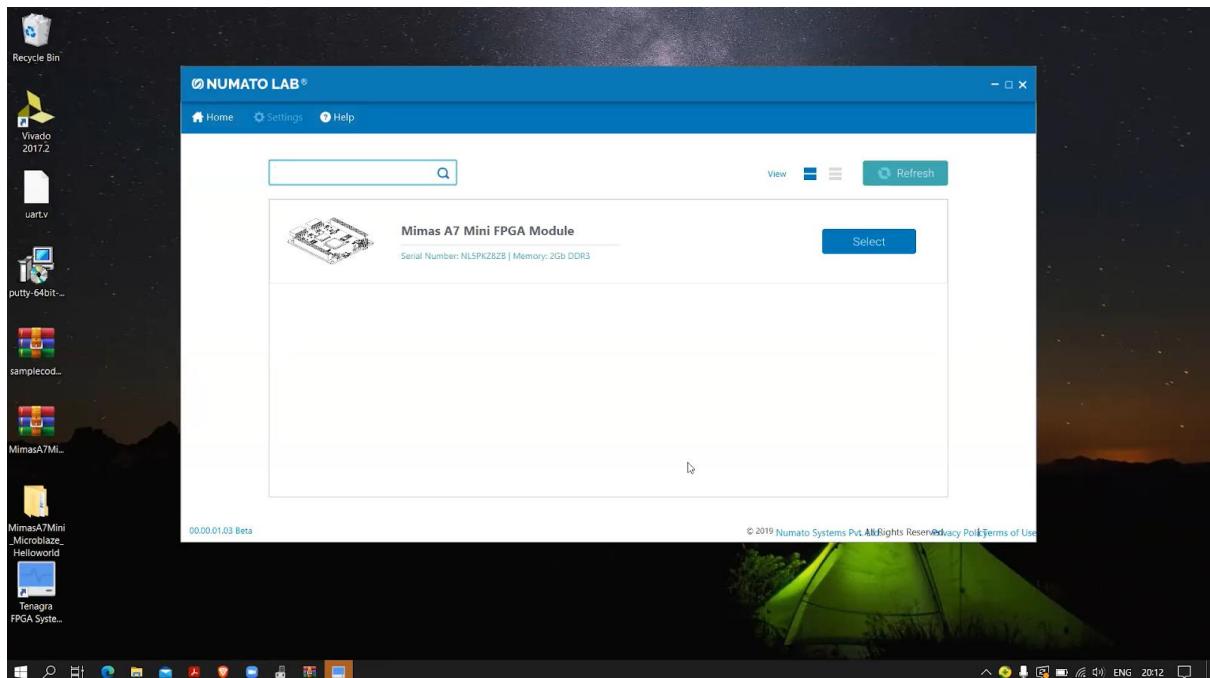
The screenshot shows the Numato Lab website homepage. At the top, there's a navigation bar with links for 'PRODUCTS', 'DEVELOPER'S CORNER', 'HELP', 'WHERE TO BUY', 'COMPANY', and a shopping cart icon. A search bar is also present. A banner on the left side of the page features the text 'Easy to use products & helpful tools' and 'Transform your ideas in to working products with minimum lead time and competitive cost.' Below the banner is a 'Get started' button. On the right side, there are two columns of product categories: 'AUTOMATION & DATA ACQUISITION' and 'FPGA & ACCELERATED COMPUTING'. Under 'AUTOMATION & DATA ACQUISITION', there are links for GPIO Modules (Bluetooth GPIO, Ethernet GPIO, Modbus GPIO, USB GPIO, WiFi GPIO), Relay Modules (Bluetooth Relay, Ethernet Relay, Modbus Relay, Relay Controller, Solid State Relay, USB Relay, WiFi Relay), and OTHER PRODUCTS (Power Over Ethernet, Adapters & Connectors, Power Supply, USB To Serial). Under 'FPGA & ACCELERATED COMPUTING', there are links for Xilinx Spartan 3, Xilinx Spartan 6, Xilinx Spartan 7, Xilinx Artix7, Xilinx Kintex 7, Xilinx Zynq, Intel MAX 10, FMC Modules, Expansion Modules, X0-Bus Lite Host Interface IP, Tenagra FPGA System Mgmt Software, Rhea Device Management Software, and Theia Android Application.

This screenshot shows a specific page for the Tenagra FPGA System Management Software. It features a 'Ask a question' button at the top. Below it, there are two sections: 'Downloads' and 'Knowledge Base'. The 'Downloads' section contains links for 'User Manual' (Tenagra-01.03-Beta Release, Tenagra-01.02-Beta Release, Tenagra (Older release)). The 'Knowledge Base' section contains a link for 'Getting started with Tenagra FPGA System Management Software'. Further down the page, there's a 'Specifications' section with a table showing attributes and values, and a link to a zip file for the software release.

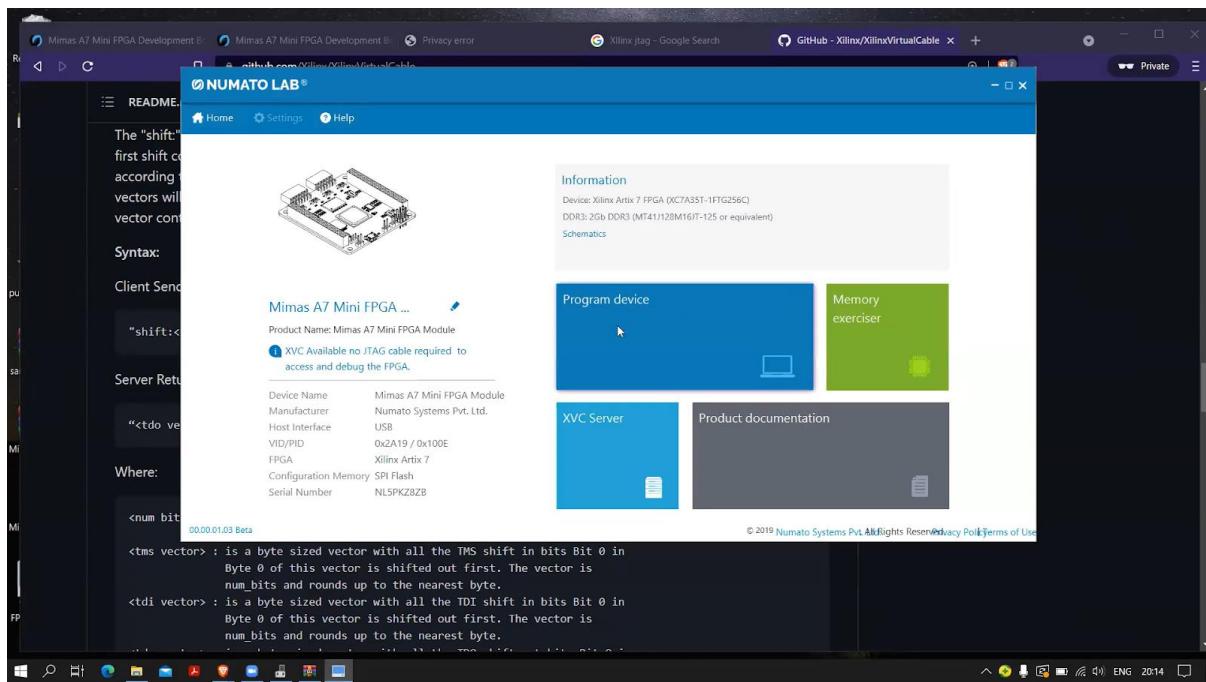
After this we have to download few drivers' fir Mimas A7 Mini Module.



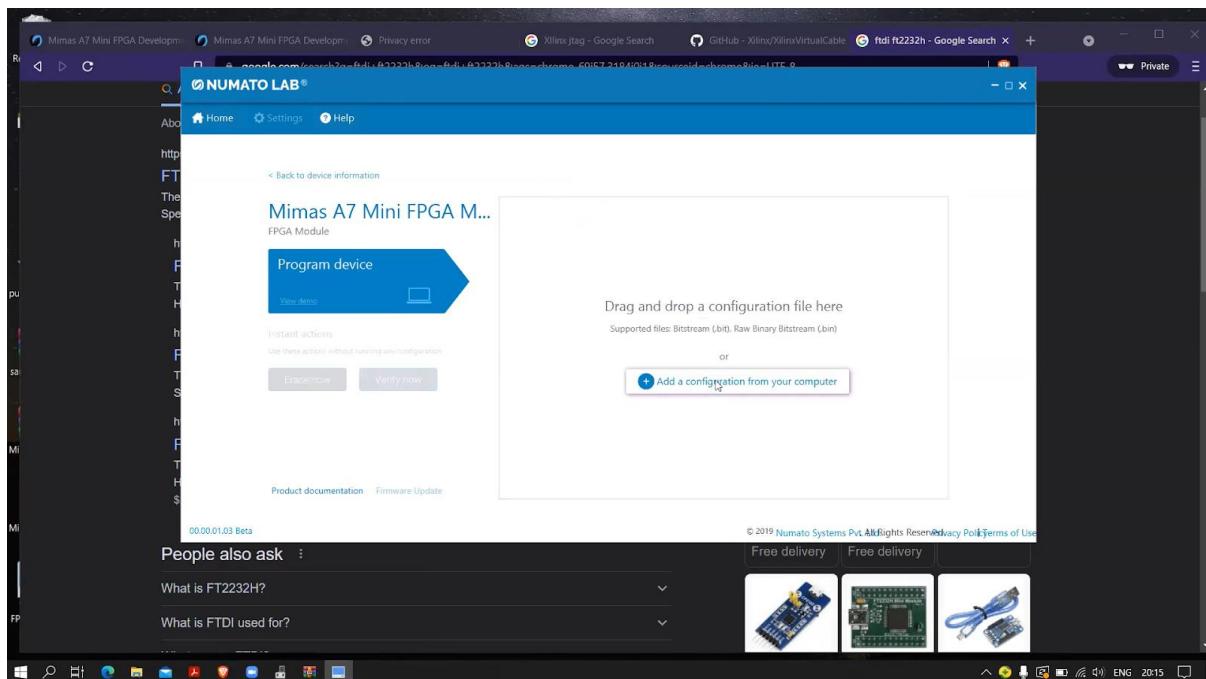
Launch tenegra where we will be viewing two ways or methods to put the program in our FPGA .



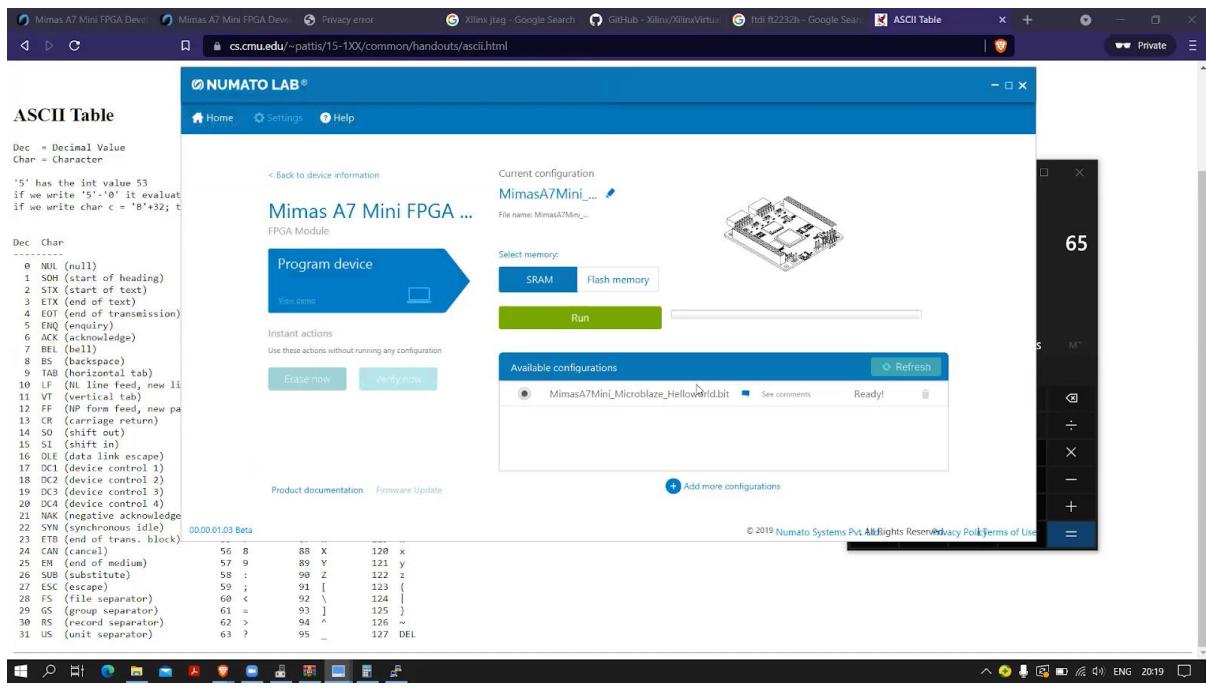
We could have used Xilinx jtag , a Xilinx platform cable, as it is quite expensive, we will not be using it rather we will using either of the two ways, XVC Server or Program device.



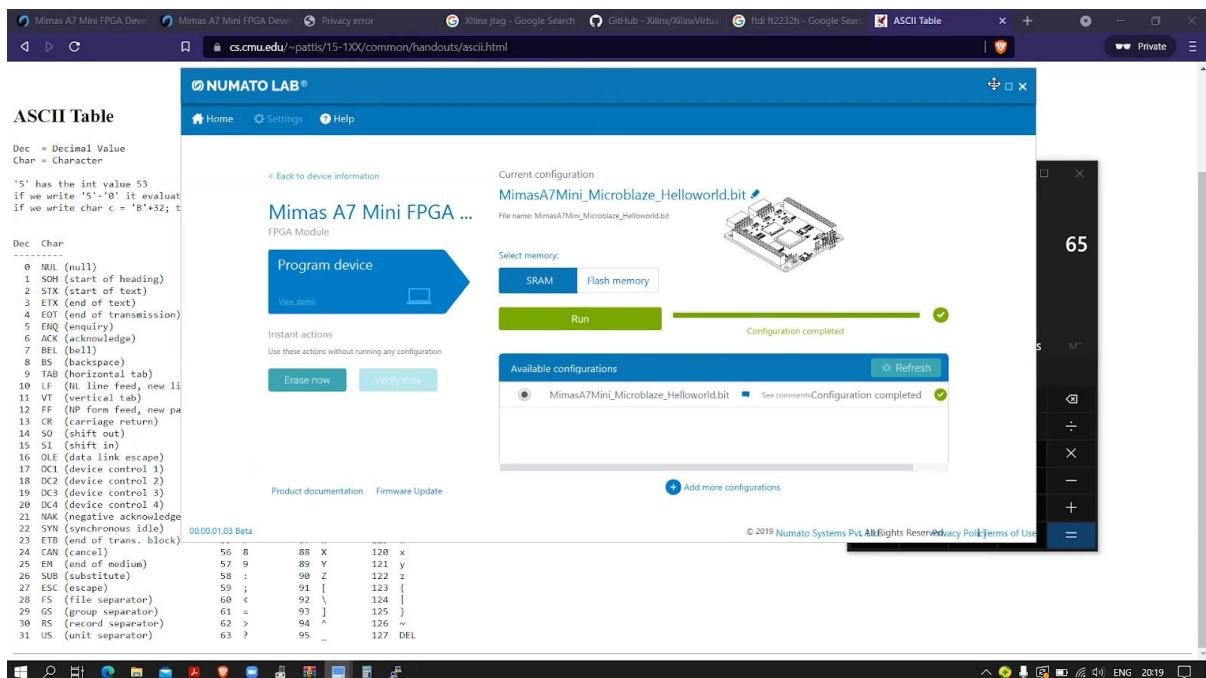
XVC server is nothing but Xilinx virtual cable, you can research and find out what it is. So, this is one way another way which we will be using to program is by an integrated circuit, FT2232HQ an onboard chip already available on Mimas A7 Mini, used for usb communication usb 2.0 communication not usb 3.0 with a speed of 480Mb/s. So, if we conclude we are using normal option that is program device to install the program on the FPGA.



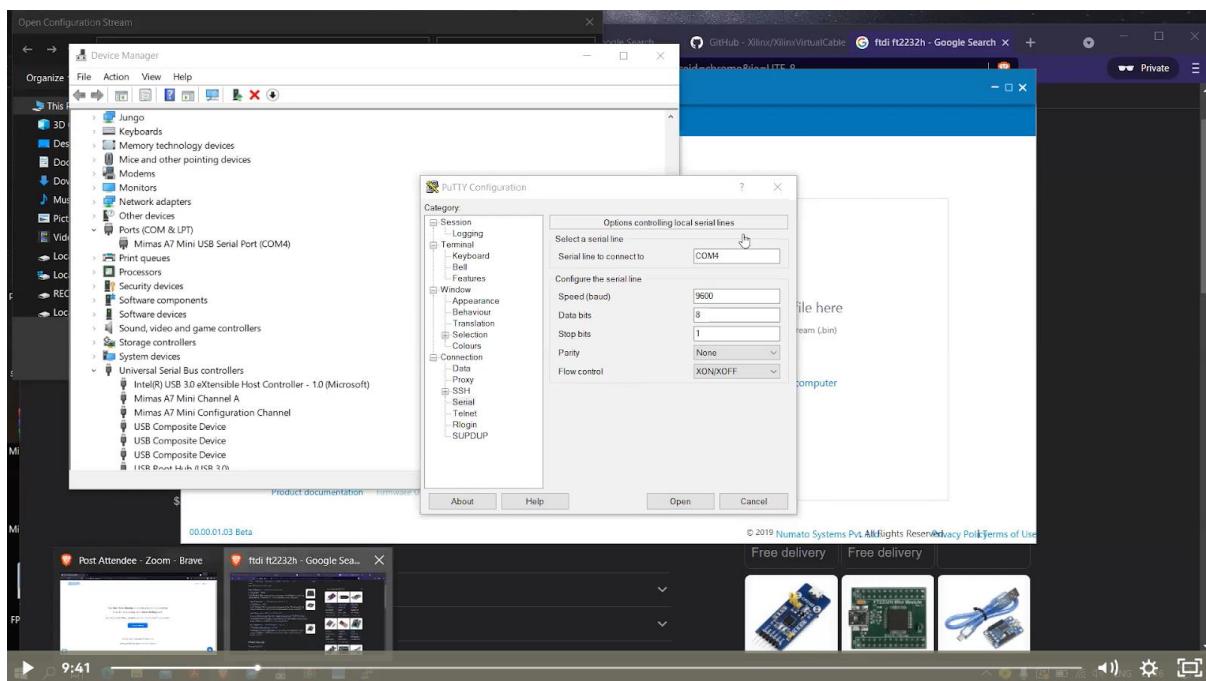
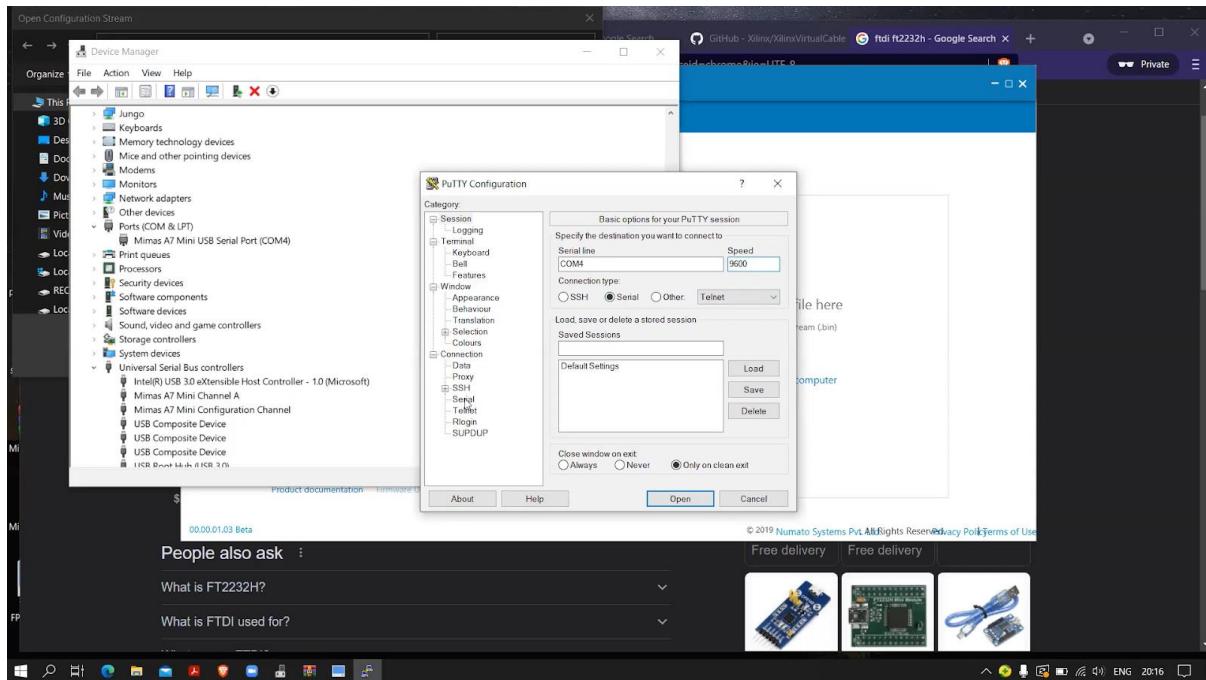
Add the bit file saved from Xilinx vivado integrated with this FPGA.



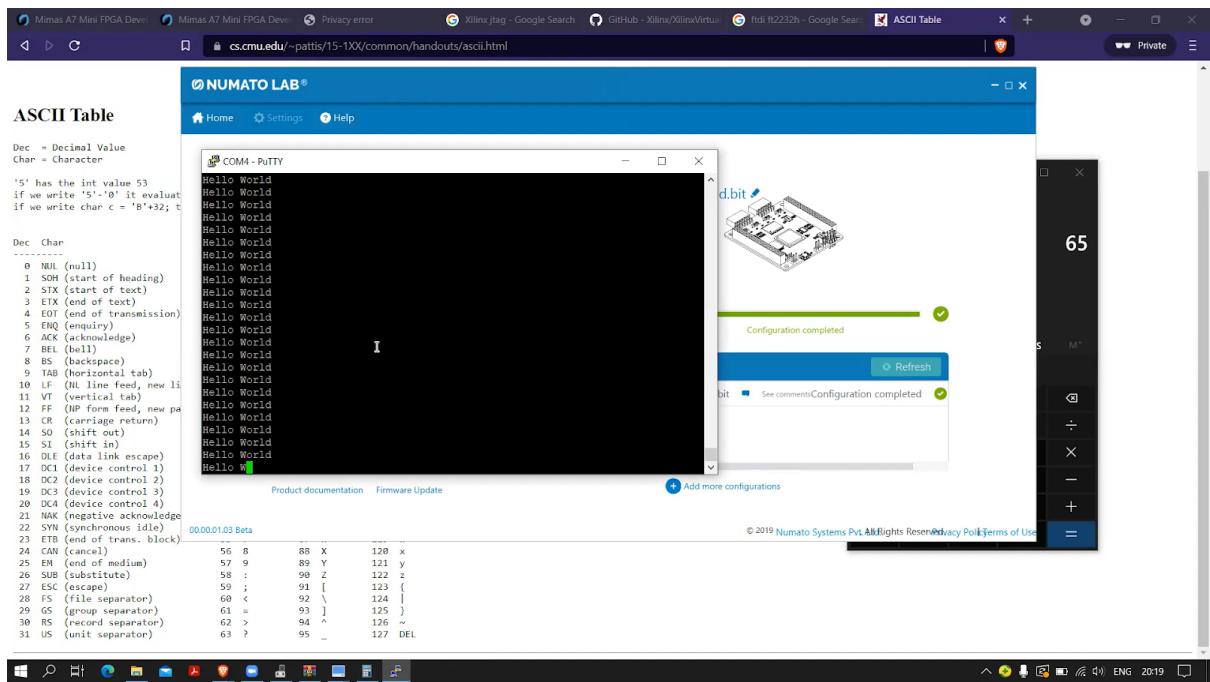
Run and configure the board with the program and after that if we have to launch putty software.



Step 2- Launch putty our serial monitors where it will receive data and make necessary changes with respect to port and connection type and so on.



Putty session will continuously print Hello world after we upload the program to our FPGA and even, we can stop it from happening with a reset button present on the board.



Open the C file where the program for hello world is written.

```
File Home Share Help
File Edit Format View Help
/helloworld.c - Notepad
/*
 * Copyright (C) 2009 - 2014 Xilinx, Inc. All rights reserved.
 *
 * Permission is hereby granted, free of charge, to any person obtaining a copy
 * of this software and associated documentation files (the "Software"), to deal
 * in the Software without restriction, including without limitation the rights
 * to use, copy, modify, merge, publish, distribute, sublicense, and/or sell
 * copies of the Software, and to permit persons to whom the Software is
 * furnished to do so, subject to the following conditions:
 *
 * The above copyright notice and this permission notice shall be included in
 * all copies or substantial portions of the Software.
 *
 * Use of the Software is limited solely to applications:
 * (a) running on a Xilinx device, or
 * (b) that interact with a Xilinx device through a bus or interconnect.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 *
 * (a) running on a Xilinx device, or
 * (b) that interact with a Xilinx device through a bus or interconnect.
 *
 * THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
 * IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
 * FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL
 * XILINX BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
 * WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF
 * OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE
 * SOFTWARE.
 *
 * Except as contained in this notice, the name of the Xilinx shall not be used
 * in advertising or otherwise to promote the sale, use or other dealings in
 * this Software without prior written authorization from Xilinx.
 */
*/
* helloworld.c: simple test application
*
* This application configures UART 16550 to baud rate 9600.
* PS7 UART (Zynq) is not initialized by this application, since
* bootrom/bsp configures it to baud rate 115200
```

```

helloworld.c - Notepad
File Edit Format View Help
* helloworld.c: simple test application
*
* This application configures UART 16550 to baud rate 9600.
* PS7 UART (Zynq) is not initialized by this application, since
* bootrom/bsp configures it to baud rate 115200
*
*
* -----
* | UART TYPE BAUD RATE
* -----
* uartns550 9600
* uartlite Configurable only in HW design
* ps7_uart 115200 (configured by bootrom/bsp)
*/
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"

int main()
{
    while(1)
    {
        |
        * -----
        * | UART TYPE BAUD RATE
        * -----
        * uartns550 9600
        * uartlite Configurable only in HW design
        * ps7_uart 115200 (configured by bootrom/bsp)
        */
#include <stdio.h>
#include "platform.h"
#include "xil_printf.h"

|
int main()
{
    while(1)
    {
        print("Hello World\n\r");
    }
}

|
Ln 52, Col 1 100% Unix (LF) UTF-8

```

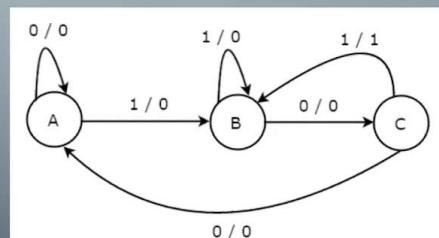
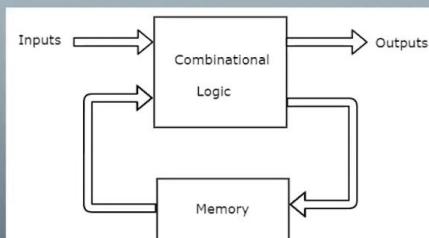
Things to know before proceeding,

Sequential circuits

- Finite State Machines
 - The behavior of synchronous sequential circuits can be represented in the graphical form and it is known as **state diagram**.
 - Decision making logic and Control of the Digital System.
 - There are two types of FSMs.
 - Mealy State Machine
 - Moore State Machine

Sequential circuits

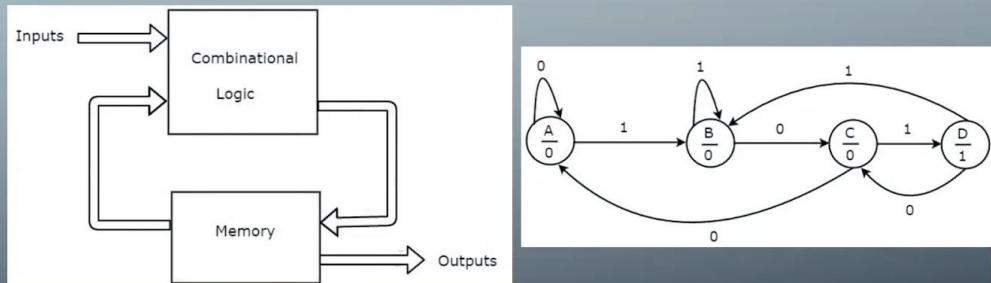
- Finite State Machines
 - Mealy State Machine
A Finite State Machine is said to be Mealy state machine, if outputs depend on both present inputs & present states.



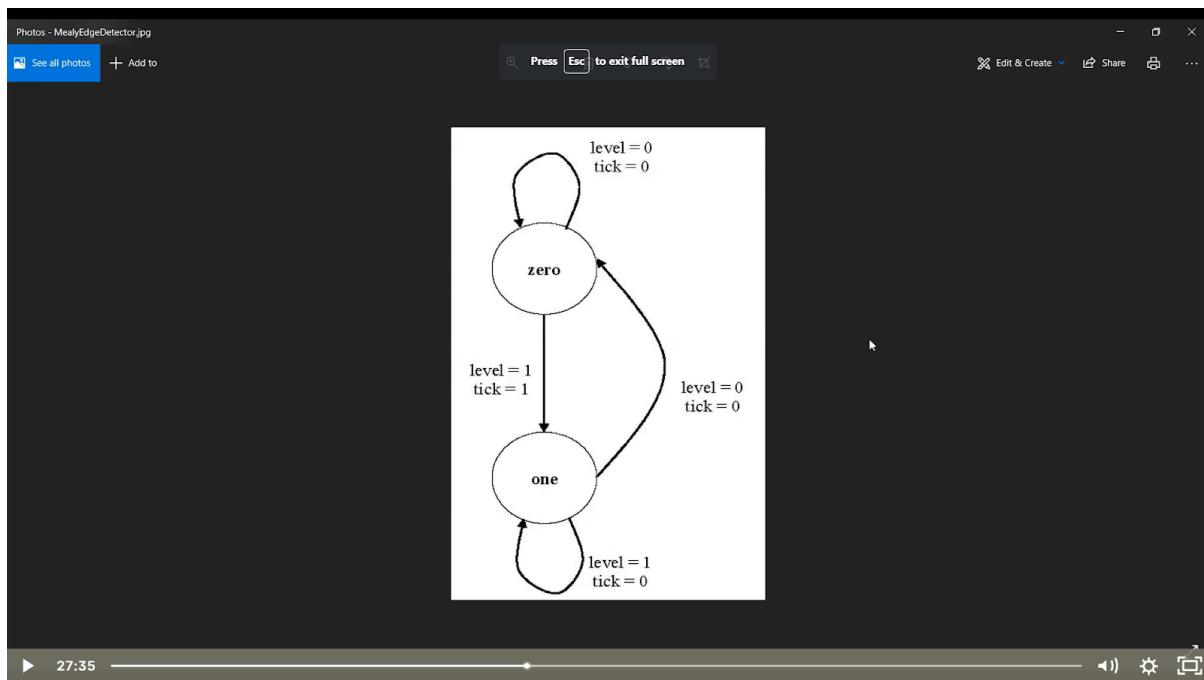
Sequential circuits

- Finite State Machines
 - Moore State Machine

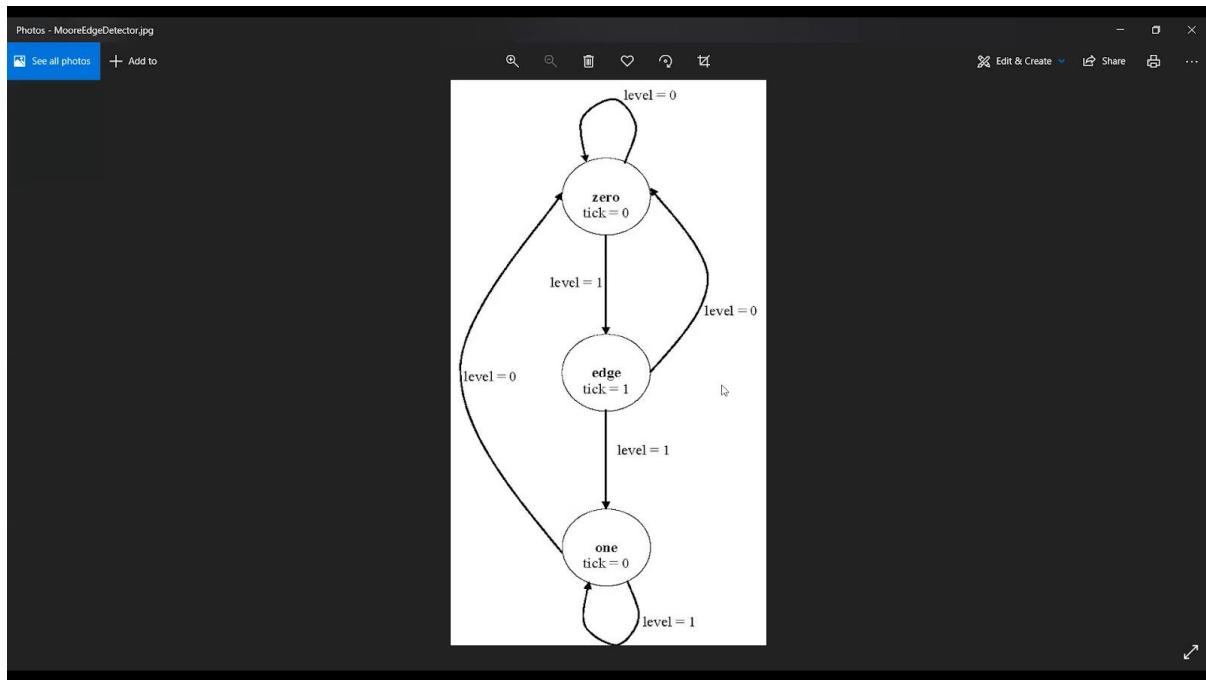
A Finite State Machine is said to be Moore state machine, if outputs depend only on present states.



Mealy state machines



Moore state machines



Step 3- Open notepad to write the Verilog code, a code to explain the basics of mealy and Moore state machine

```

C:\Users\Kishore Damam\Desktop\tsmsv - Notepad+
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Verilog file
fsm.sv
1 // edgeDetector.v
2 // Moore and Mealy Implementation
3
4 module edgeDetector
5 (
6   input wire clk, reset,
7   input wire level,
8   output reg Mealy_tick, Moore_tick
9 );
10
11 localparam // 2 states are required for Mealy
12   zeroMealy = 1'b0,
13   oneMealy = 1'b1;
14
15 localparam [1:0] // 3 states are required for Moore
16   zeroMoore = 2'b00,
17   edgeMoore = 2'b01,
18   oneMoore = 2'b10;
19
20 reg stateMealy_reg, stateMealy_next;
21 reg[1:0] stateMoore_reg, stateMoore_next;
22
23 always @ (posedge clk, posedge reset)
24 begin

```

length:2,603 lines:81 Ln:7 Col:23 Pos:132 Windows (CR LF) UTF-8 INS


```

C:\Users\Kishore Damam\Desktop\tsmsv - Notepad+
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
Verilog file
fsm.sv
16   zeroMoore = 2'b00,
17   edgeMoore = 2'b01,
18   oneMoore = 2'b10;
19
20 reg stateMealy_reg, stateMealy_next;
21 reg[1:0] stateMoore_reg, stateMoore_next;
22
23 always @ (posedge clk, posedge reset)
24 begin
25   if(reset) // go to state zero if rese
26     begin
27       stateMealy_reg <= zeroMealy;
28       stateMoore_reg <= zeroMoore;
29     end
30   else // otherwise update the states
31     begin
32       stateMealy_reg <= stateMealy_next;
33       stateMoore_reg <= stateMoore_next;
34     end
35 end
36
37 // Mealy Design
38 always @ (stateMealy_reg, level)
39 begin

```

length:2,603 lines:81 Ln:16 Col:14 Sel:9|1 Windows (CR LF) UTF-8 INS

The image shows two side-by-side Notepad++ windows displaying Verilog code. The left window contains Mealy state machine logic, and the right window contains Moore state machine logic. Both windows have identical toolbars and status bars at the bottom.

Mealy State Machine Logic (Left Window):

```
31 begin
32     stateMealy_reg <= stateMealy_next;
33     stateMoore_reg <= stateMoore_next;
34 end
35
36 // Mealy Design
37 always @(stateMealy_reg, level)
38 begin
39     // store current state as next
40     stateMealy_next = stateMealy_reg; // required: when no case statement is satisfied
41
42     Mealy_tick = 1'b0; // set tick to zero (so that 'tick = 1' is available for 1 cycle on
43     case(stateMealy_reg)
44         zeroMealy: // set 'tick = 1' if state = zero and level = '1'
45             if(level)
46                 begin // if level is 1, then go to state one,
47                     stateMealy_next = oneMealy; // otherwise remain in same state.
48                     Mealy_tick = 1'b1;
49                 end
50         oneMealy:
51             if(~level) // if level is 0, then go to zero state,
52                 stateMealy_next = zeroMealy; // otherwise remain in one state.
53     endcase
54 end
```

Moore State Machine Logic (Right Window):

```
43 Mealy_tick = 1'b0; // set tick to zero (so that 'tick = 1' is available for 1 cycle on
44 case(stateMealy_reg)
45     zeroMealy: // set 'tick = 1' if state = zero and level = '1'
46         if(level)
47             begin // if level is 1, then go to state one,
48                 stateMealy_next = oneMealy; // otherwise remain in same state.
49                 Mealy_tick = 1'b1;
50         end
51     oneMealy:
52         if(~level) // if level is 0, then go to zero state,
53             stateMealy_next = zeroMealy; // otherwise remain in one state.
54     endcase
55 end
56
57 // Moore Design
58 always @(stateMoore_reg, level)
59 begin
60     // store current state as next
61     stateMoore_next = stateMoore_reg; // required: when no case statement is satisfied
62
63     Moore_tick = 1'b0; // set tick to zero (so that 'tick = 1' is available for 1 cycle on
64     case(stateMoore_reg)
65         zeroMoore: // if state is zero,
66             if(level) // and level is 1
```

The screenshot shows a Notepad+ window titled 'C:\Users\Kishore Damam\Desktop\ fsm.sv - Notepad+'. The code is a Verilog module for a state machine:

```
64 case(stateMoore_reg)
65     zeroMoore: // if state is zero,
66         if(level) // and level is 1
67             stateMoore_next = edgeMoore; // then go to state edge.
68     edgeMoore:
69         begin
70             Moore_tick = 1'b1; // set the tick to 1.
71             if(level) // if level is 1,
72                 stateMoore_next = oneMoore; // go to state one,
73             else
74                 stateMoore_next = zeroMoore; // else go to state zero.
75         end
76     oneMoore:
77         if(~level) // if level is 0,
78             stateMoore_next = zeroMoore; // then go to state zero.
79     endcase
80 end
81 endmodule
```

Verilog file

length: 2,603 lines: 81 Ln: 16 Col: 14 Sel: 9 | 1 Windows (CR/LF) UTF-8 INS

Open notepad to write the verilog code, a code to explain the UART module

The screenshot shows a Notepad+ window titled 'C:\Users\Kishore Damam\Desktop\uart_tx.v - Notepad+'. The code is a Verilog module for a UART transmitter:

```
1 //////////////////////////////////////////////////////////////////
2 // File Downloaded from http://www.nandland.com
3 //////////////////////////////////////////////////////////////////
4 // This file contains the UART Transmitter. This transmitter is able
5 // to transmit 8 bits of serial data, one start bit, one stop bit,
6 // and no parity bit. When transmit is complete o_Tx_Done will be
7 // driven high for one clock cycle.
8 //
9 // Set Parameter CLKS_PER_BIT as follows:
10 // CLKS_PER_BIT = (Frequency of i_Clock)/(Frequency of UART)
11 // Example: 25 MHz Clock, 115200 baud UART i
12 // (25000000)/(115200) = 217
13
14 module UART_TX
15     #(parameter CLKS_PER_BIT = 217)
16     (
17         input      i_Clock,
18         input      i_TX_DV,
19         input [7:0] i_TX_Byte,
20         output     o_TX_Active,
21         output reg o_TX_Serial,
22         output     o_TX_Done
23     );
24
```

Verilog file

length: 4,079 lines: 147 Ln: 135 Col: 7 Pos: 3,906 Windows (CR/LF) UTF-8 INS

```
C:\Users\Kishore Damam\Desktop\uart_tx.v - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
Sims.v uart_tx.v  
7 // driven high for one clock cycle.  
8 //  
9 // Set Parameter CLKS_PER_BIT as follows:  
10 // CLKS_PER_BIT = (Frequency of i_Clock)/(Frequency of UART)  
11 // Example: 25 MHz Clock, 115200 baud UART  
12 // (25000000)/(115200) = 217  
13  
14 module UART_TX  
15     #(parameter CLKS_PER_BIT = 217)  
16     (  
17         input      i_Clock,  
18         input      i_TX_DV,  
19         input [7:0] i_TX_Byte,  
20         output     o_TX_Active,  
21         output reg o_TX_Serial,  
22         output     o_TX_Done  
23     );  
24  
25 parameter IDLE          = 3'b000;  
26 parameter TX_START_BIT  = 3'b001;  
27 parameter TX_DATA_BITS  = 3'b010;  
28 parameter TX_STOP_BIT   = 3'b011;  
29 parameter CLEANUP       = 3'b100;  
30  
Verilog file length:4,079 lines:147 Ln:135 Col:7 Pos:3,906 Windows (CR LF) UTF-8 INS
```

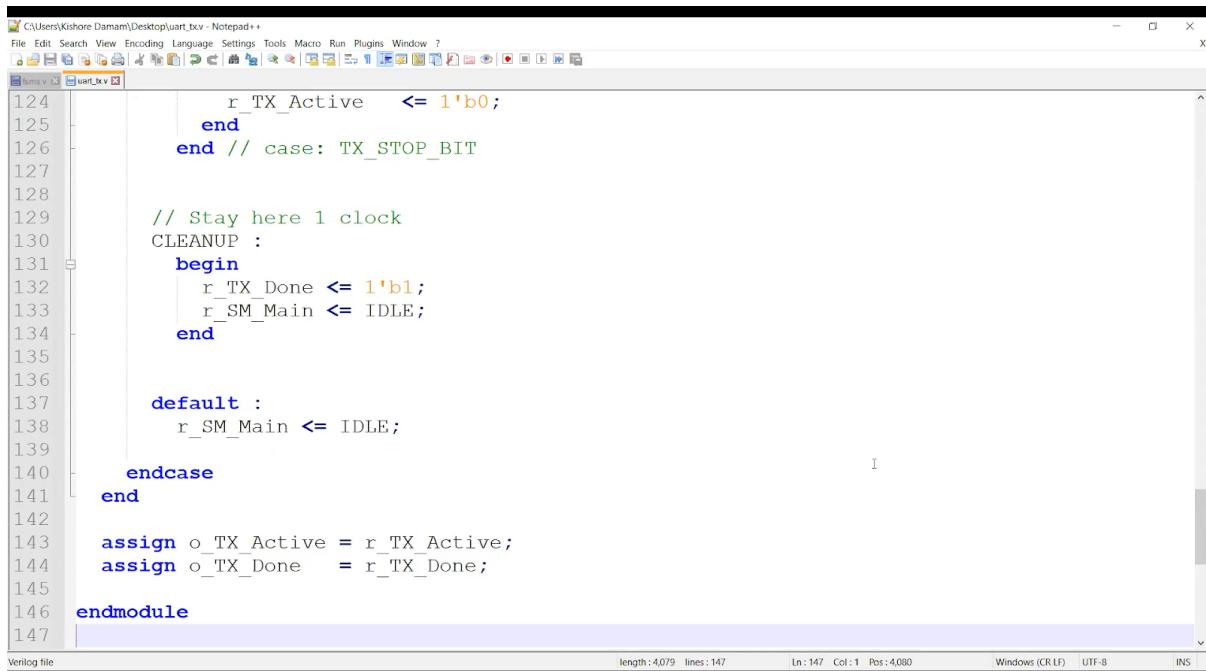
```
C:\Users\Kishore Damam\Desktop\uart_tx.v - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
Sims.v uart_tx.v  
25 parameter IDLE          = 3'b000;  
26 parameter TX_START_BIT  = 3'b001;  
27 parameter TX_DATA_BITS  = 3'b010;  
28 parameter TX_STOP_BIT   = 3'b011;  
29 parameter CLEANUP       = 3'b100;  
30  
31 reg [2:0] r_SM_Main    = 0;  
32 reg [7:0] r_Clock_Count = 0;  
33 reg [2:0] r_Bit_Index  = 0;  
34 reg [7:0] r_TX_Data    = 0;  
35 reg      r_TX_Done    = 0;  
36 reg      r_TX_Active  = 0;  
37  
38 always @ (posedge i_Clock)  
39 begin  
40  
41     case (r_SM_Main)  
42         IDLE :  
43             begin  
44                 o_TX_Serial  <= 1'b1;           // Drive Line High for Idle  
45                 r_TX_Done    <= 1'b0;  
46                 r_Clock_Count <= 0;  
47                 r_Bit_Index  <= 0;
```

```
43 begin
44     o_TX_Serial    <= 1'b1;           // Drive Line High for Idle
45     r_TX_Done      <= 1'b0;
46     r_Clock_Count <= 0;
47     r_Bit_Index    <= 0;
48
49     if (i_TX_DV == 1'b1)
50     begin
51         r_TX_Active <= 1'b1;
52         r_TX_Data   <= i_TX_Byte;
53         r_SM_Main   <= TX_START_BIT;
54     end
55     else
56         r_SM_Main <= IDLE;
57 end // case: IDLE
58
59 // Send out Start Bit. Start bit = 0
60 TX_START_BIT :
61 begin
62     o_TX_Serial <= 1'b0;
63
64     // Wait CLKS_PER_BIT-1 clock cycles for start bit to finish
65     if (r_Clock_Count < CLKS_PER_BIT-1)
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
```

```
58
59
60 // Send out Start Bit. Start bit = 0
61 TX_START_BIT :
62 begin
63     o_TX_Serial <= 1'b0;
64
65     // Wait CLKS_PER_BIT-1 clock cycles for start bit to finish
66     if (r_Clock_Count < CLKS_PER_BIT-1)
67     begin
68         r_Clock_Count <= r_Clock_Count + 1;
69         r_SM_Main     <= TX_START_BIT;
70     end
71     else
72     begin
73         r_Clock_Count <= 0;
74         r_SM_Main     <= TX_DATA_BITS;
75     end
76 end // case: TX_START_BIT
77
78
79 // Wait CLKS_PER_BIT-1 clock cycles for data bits to finish
80 TX_DATA_BITS :
81 begin
```

```
C:\Users\Kishore Damam\Desktop\uart_tx.v - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
Sims.v uart_tx.v  
76 end // case: TX_START_BIT  
77  
78  
79 // Wait CLKS_PER_BIT-1 clock cycles for data bits to finish  
80 TX_DATA_BITS :  
81 begin  
82 o_TX_Serial <= r_TX_Data[r_Bit_Index];  
83  
84 if (r_Clock_Count < CLKS_PER_BIT-1)  
85 begin  
86 r_Clock_Count <= r_Clock_Count + 1;  
87 r_SM_Main <= TX_DATA_BITS;  
88 end  
89 else  
90 begin  
91 r_Clock_Count <= 0;  
92  
93 // Check if we have sent out all bits  
94 if (r_Bit_Index < 7)  
95 begin  
96 r_Bit_Index <= r_Bit_Index + 1;  
97 r_SM_Main <= TX_DATA_BITS;  
98 end  
99 else  
Verilog file length:4,079 lines:147 Ln:135 Col:7 Pos:3,906 Windows (CR LF) UTF-8 INS
```

```
C:\Users\Kishore Damam\Desktop\uart_tx.v - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
Sims.v uart_tx.v  
97 r_SM_Main <= TX_DATA_BITS;  
98 end  
99 else  
100 begin  
101 r_Bit_Index <= 0;  
102 r_SM_Main <= TX_STOP_BIT;  
103 end  
104 end  
105 end // case: TX_DATA_BITS  
106  
107  
108 // Send out Stop bit. Stop bit = 1  
109 TX_STOP_BIT :  
110 begin  
111 o_TX_Serial <= 1'b1;  
112  
113 // Wait CLKS_PER_BIT-1 clock cycles for Stop bit to finish  
114 if (r_Clock_Count < CLKS_PER_BIT-1)  
115 begin  
116 r_Clock_Count <= r_Clock_Count + 1;  
117 r_SM_Main <= TX_STOP_BIT;  
118 end  
119 else  
120 begin  
Verilog file length:4,079 lines:147 Ln:135 Col:7 Pos:3,906 Windows (CR LF) UTF-8 INS
```



```

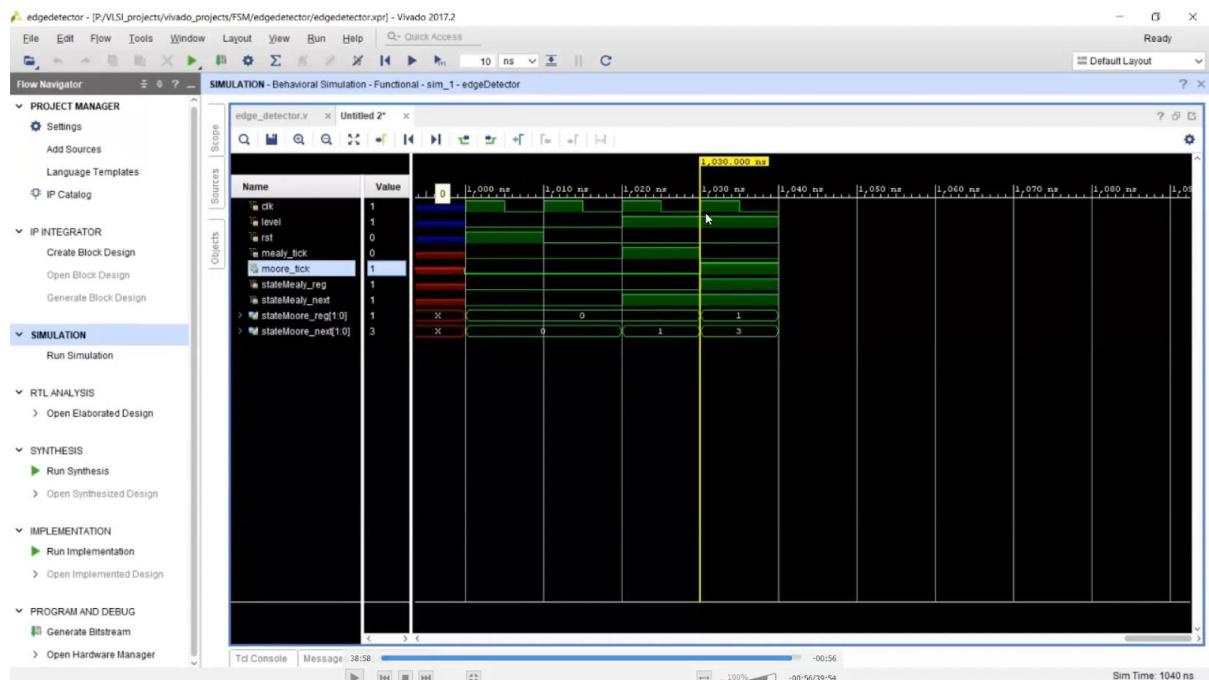
124      r_TX_Active    <= 1'b0;
125      end
126      end // case: TX_STOP_BIT
127
128
129      // Stay here 1 clock
130      CLEANUP :
131      begin
132          r_TX_Done <= 1'b1;
133          r_SM_Main <= IDLE;
134      end
135
136
137      default :
138          r_SM_Main <= IDLE;
139
140      endcase
141
142
143      assign o_TX_Active = r_TX_Active;
144      assign o_TX_Done   = r_TX_Done;
145
146  endmodule
147

```

Verilog file

length:4,079 lines:147 Ln:147 Col:1 Pos:4,080 Windows (CR/LF) UTF-8 INS

Output



What Is RISC?

A Reduced Instruction Set Computer is a type of microprocessor architecture that utilizes a small, highly-optimized set of instructions rather than the highly-specialized set of instructions typically found in other architectures. RISC is an alternative to the Complex Instruction Set Computing (CISC) architecture and is often considered the most efficient CPU architecture technology available today.

With RISC, a central processing unit (CPU) implements the processor design principle of simplified instructions that can do less but can execute more rapidly. The result is improved performance. A key RISC feature is that it allows developers to increase the register set and increase internal parallelism by increasing the number of parallel threads executed by the CPU and increasing the speed of the CPU's executing instructions. ARM, or "Advanced RISC Machine" is a specific family of instruction set architecture that's based on reduced instruction set architecture developed by Arm Ltd. Processors based on this architecture are common in smartphones, tablets, laptops, gaming consoles and desktops, as well as a growing number of other intelligent devices.

Why Is RISC Important?

RISC provides high performance per watt for battery operated devices where energy efficiency is key. A RISC processor executes one action per instruction. By taking just one cycle to complete, operation execution time is optimized.

Because the architecture uses a fixed length of instruction, it's easier to pipeline. And because it lacks complex instruction decoding logic, it supports more registers and spends less time on loading and storing values to memory.

For chip designers, RISC processors simplify the design and deployment process and provide a lower per-chip cost due to the smaller components required. Because of the reduced instruction set and simple decoding logic, less chip space is used, fewer transistors are required, and more general-purpose registers can fit into the central processing unit.

Instruction Format

Arithmetic

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE				Rd			Ra			Rb			X	X	

Immediate

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OPCODE (1000X)					Rd			Imm							



Operations

OPCODE					ALU Operation
0	0	0	0	0	Addition (Unsigned)
0	0	0	0	1	Addition (Signed)
0	0	0	1	X	Bitwise OR
0	0	1	0	X	Bitwise AND
0	0	1	1	X	Bitwise XOR
0	1	0	0	X	Bitwise NOT
0	1	0	1	X	Read Memory
0	1	1	0	X	Write Memory
1	0	0	0	0	Load Register (Low)
1	0	0	0	1	Load Register (High)
1	0	0	1	0	Compare (Unsigned)
1	0	0	1	1	Compare (Signed)
1	0	1	0	X	Shift Left
1	0	1	1	X	Shift Right
1	1	0	0	0	Jump (Imm.)
1	1	0	0	1	Jump (Register)

Design Modules

- 6. Instruction Decoder**
- 7. Control Unit**
- 8. ALU**
- 9. Registers handler**
- 10. Program counter**
- 11. RAM**

Testing Modules

- 12.Top test bench**
- 13.Instruction Decoder**
- 14.Control Unit**
- 15.ALU**
- 16.Registers handler**
- 17.Program counter**
- 18.RAM**

4. 16-Bit RISC

Processor

Software to download

- Notepad++

Step 1- Write the verilog program in notepad ++ .

DESIGN CODES

Module inst_dec

```
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
inst_dec.v
1 //TimeScale
2 'timescale 1ns / 1ps
3
4 //Module Definition
5 module inst_dec(
6     // Inputs
7     input I_clk,
8     input I_en,
9     input [15:0] I_inst,
10    // Outputs
11    output O_aluop,
12    output [3:0] O_selA,
13    output [3:0] O_selB,
14    output [3:0] O_selD,
15    output [15:0] O_imm,
16    output O_regwe
17 );
18
19
20 endmodule
```

Verilog file length : 298 lines : 20 Ln : 17 Col : 3 Pos : 284 Windows (CR LF) UTF-8 INS 1446

```
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
inst_dec.v
17 );
18 // Initial Block
19 initial begin
20
21     O_aluop <= 0;
22     O_selA <= 0;
23     O_selB <= 0;
24     O_selD <= 0;
25     O_imm <= 0;
26     O_regwe <= 0;
27
28 end
29
30 //Instruction Decoder Block
31 always@(negedge I_clk) begin
32
33     O_aluop <= I_inst[15:11];
34
35 end
36
37
38
```

Verilog file length : 576 lines : 40 Ln : 34 Col : 9 Pos : 551 Windows (CR LF) UTF-8 INS 1450

The screenshot shows a Notepad++ window displaying Verilog code for an instruction decoder module. The file is named `inst_dec.v`. The code implements a block for decoding 16-bit instructions. It includes assignments for output registers based on the opcode and source register selection, and a case statement for the REG Write Enable field. The code is annotated with comments explaining the purpose of each assignment.

```
27 //Instruction Decoder Block
28 always@(negedge I_clk) begin
29
30     if(I_en) begin
31         O_aluop <= I_inst[15:11]; //Opcode
32         O_selA <= I_inst[10:8]; //REG A
33         O_selB <= I_inst[7:5]; //REG B
34         O_selD <= I_inst[4:2]; //REG D
35         O_imm <= I_inst[7:0]; //Imm Data
36
37     //REG Write Enable
38     case(I_inst[15:12])
39         4'b0111: O_regwe <= 0;
40         4'b1100: O_regwe <= 0;
41         4'b1101: O_regwe <= 0;
42         default: O_regwe <= 1;
43     endcase
44
45     end
46 end
47
48 endmodule
```

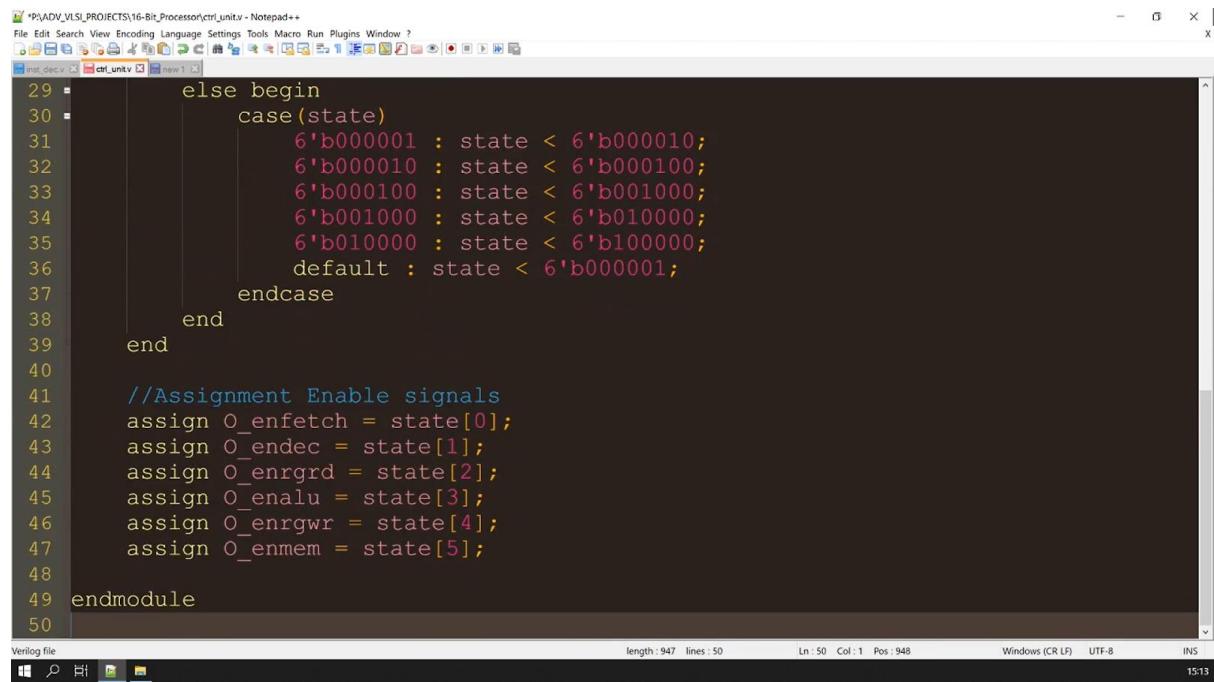
Verilog file length : 920 lines : 48 Ln : 32 Col : 27 Pos : 549 Windows (CR LF) UTF-8 INS 14:58

Module ctrl_unit

```

1 // TimeScale
2 'timescale 1ns / 1ps
3
4 // Module Definition
5 module ctrl_unit(
6     // Inputs
7     input I_clk,
8     input I_reset,
9     // Outputs
10    output O_enfetch,
11    output O_endec,
12    output O_enrgrd,
13    output O_enalu,
14    output O_enrgwr,
15    output O_enmem
16 );
17 // Reg Declaration
18 reg [5:0] state;
19
20 initial begin
21     state <= 6'b000001;
22 end
23
24
25 // State Select Block
26 always@(posedge I_clk) begin
27     if(I_reset)
28         state <= 6'000001;
29     else begin
30         case(state)
31             6'b000001 : state < 6'b000010;
32             6'b000010 : state < 6'b000100;
33             6'b000100 : state < 6'b001000;
34             6'b001000 : state < 6'b010000;
35             6'b010000 : state < 6'b100000;
36             default : state < 6'b000001;
37         endcase
38     end
39 end
40
41 //Assignment Enable signals
42 assign O_enfetch = state[0];
43 assign O_enfetch = state[0];
44 assign O_enfetch = state[0];

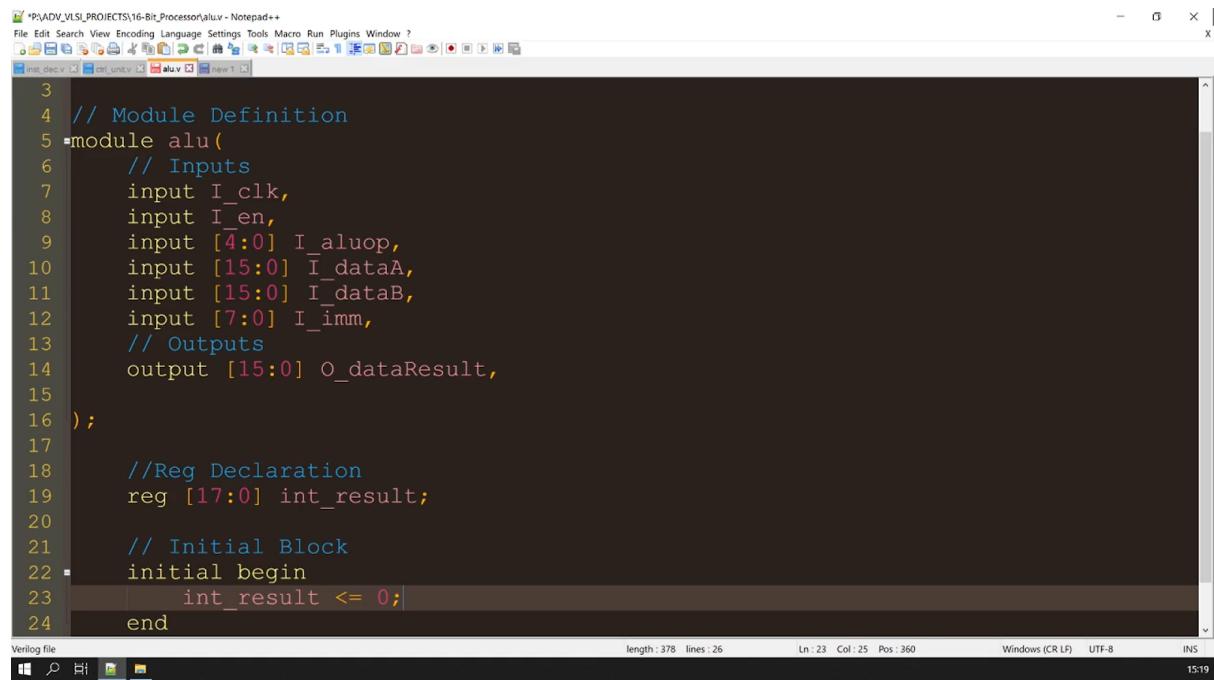
```



```
29 else begin
30     case(state)
31         6'b000001 : state < 6'b000010;
32         6'b000010 : state < 6'b000100;
33         6'b000100 : state < 6'b001000;
34         6'b001000 : state < 6'b010000;
35         6'b010000 : state < 6'b100000;
36         default : state < 6'b000001;
37     endcase
38 end
39
40 //Assignment Enable signals
41 assign O_enfetch = state[0];
42 assign O_endec = state[1];
43 assign O_enrgrd = state[2];
44 assign O_enalu = state[3];
45 assign O_enrgwr = state[4];
46 assign O_enmem = state[5];
47
48
49 endmodule
50
```

length : 947 lines : 50 Ln : 50 Col : 1 Pos : 948 Windows (CR LF) UTF-8 INS 15:19

Module alu



```
3
4 // Module Definition
5 module alu(
6     // Inputs
7     input I_clk,
8     input I_en,
9     input [4:0] I_aluop,
10    input [15:0] I_dataA,
11    input [15:0] I_dataB,
12    input [7:0] I_imm,
13    // Outputs
14    output [15:0] O_dataResult,
15
16 );
17
18 //Reg Declaration
19 reg [17:0] int_result;
20
21 // Initial Block
22 initial begin
23     int_result <= 0;
24 end
```

length : 378 lines : 26 Ln : 23 Col : 25 Pos : 360 Windows (CR LF) UTF-8 INS 15:19

```
*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\alu.v - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
Inst. dec.v Ctl. unity alu.v new 1  
14     output [15:0] O_dataResult,  
15  
16 );  
17  
18 //Reg/wire Declaration  
19 reg [17:0] int_result;  
20 wire op_lsb;  
21 reg [3:0] opcode;  
22  
23 // Parameter Declaration  
24 localparam Add = 0,  
25         Sub = 1,  
26         OR = 2,  
27         AND = 3,  
28         XOR = 4,  
29         NOT = 5,  
30         Rdmem = 6,  
31         Wrmem = 7,  
32         Load = 8,  
33         Cmp = 9,  
34         SHL = 10,  
35         SHR = 11,  
36  
Verilog file length : 936 lines : 67 Ln : 19 Col : 26 Sel : 10 | 1 Windows (CR LF) UTF-8 INS 15:27
```

```
*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\alu.v - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
Inst. dec.v Ctl. unity alu.v new 1  
29 NOT = 5,  
30 Rdmem = 6,  
31 Wrmem = 7,  
32 Load = 8,  
33 Cmp = 9,  
34 SHL = 10,  
35 SHR = 11,  
36 JMPA = 12,  
37 JMPR = 13;  
38  
39  
40 // Initial Block  
41 initial begin  
42     int_result <= 0;  
43 end  
44  
45 //Assigning values  
46 assign op_lsb <= I_aluop[0];  
47 assign opcode <= I_aluop[4:1];  
48  
49 // ALU Operations  
50 always@ (negedge I_clk) begin  
Verilog file length : 936 lines : 67 Ln : 56 Col : 25 Pos : 876 Windows (CR LF) UTF-8 INS 15:27
```

The screenshot shows a Verilog code editor with the following code:

```
43 end
44
45 //Assigning values
46 assign op_lsb <= I_aluop[0];
47 assign opcode <= I_aluop[4:1];
48
49 // ALU Operations
50 always@(posedge I_clk) begin
51
52 if(I_en) begin
53 case(opcode)
54
55 Add : begin
56     int_result <= (op_lsb ? ($signed(I_dataA) + $signed(I_dataB)) :
57         end
58
59 Sub : begin
60     int_result <= (op_lsb ? ($signed(I_dataA) - $signed(I_dataB)) :
61         end
62
63 OR : begin
64     int_result <= I_dataA | I_dataB;
65
66 end
```

The code implements an ALU with four operations: Add, Sub, OR, and AND. The result is assigned based on the value of `op_lsb`. The code uses signed arithmetic for addition and subtraction.

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

Sub : begin
 | int_result <= (op_lsb ? (\$signed(I_dataA) - \$signed(I_dataB)) :
 | end
 OR : begin
 | int_result <= I_dataA | I_dataB;
 | end
 AND : begin
 | int_result <= I_dataA & I_dataB;
 | end
 XOR : begin
 | int_result <= I_dataA ^ I_dataB;
 | end
 NOT : begin
 | int_result <= ~I_dataA;
 | end

Cmp : begin
 if(op_lsb) begin
 | int_result[0] <= (\$signed(I_dataA) == \$signed(I_dataB)) ? 1
 | int_result[1] <= (\$signed(I_dataA) == 0) ? 1 : 0;
 | int_result[2] <= (\$signed(I_dataB) == 0) ? 1 : 0;
 | int_result[3] <= (\$signed(I_dataA) > \$signed(I_dataB)) ? 1
 | int_result[4] <= (\$signed(I_dataA) < \$signed(I_dataB)) ? 1
 end else begin
 | int_result[0] <= (I_dataA == I_dataB) ? 1 : 0;
 | int_result[1] <= (I_dataA == 0) ? 1 : 0;
 | int_result[2] <= (I_dataB == 0) ? 1 : 0;
 | int_result[3] <= (I_dataA > I_dataB) ? 1 : 0;
 | int_result[4] <= (I_dataA < I_dataB) ? 1 : 0;
 end
 end
 SHL : begin
 | int_result <= I_dataA << (I_dataB[3:0]);
 end

```

97      int_result[1] <= (I_dataA == 0) ? 1 : 0;
98      int_result[2] <= (I_dataB == 0) ? 1 : 0;
99      int_result[3] <= (I_dataA > I_dataB) ? 1 : 0;
100     int_result[4] <= (I_dataA < I_dataB) ? 1 : 0;
101   end
102   O_shldBranch <= 0;
103 end
104
105   SHL : begin
106     int_result <= I_dataA << (I_dataB[3:0]);
107     O_shldBranch <= 0;
108   end
109
110   SHR : begin
111     int_result <= I_dataA >> (I_dataB[3:0]);
112     O_shldBranch <= 0;
113   end
114
115   JMPA : begin
116     int_result <= (op_lsb ? I_dataA : I_imm);
117     O_shldBranch <= 1;
118   end
119

```

```

Verilog file length : 2,748 lines : 126 Ln : 113 Col : 21 Pos : 2,568 Windows (CR LF) UTF-8 INS
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
inst_desc.v car_unit.v aluv new1
105   SHL : begin
106     int_result <= I_dataA << (I_dataB[3:0]);
107     O_shldBranch <= 0;
108   end
109
110   SHR : begin
111     int_result <= I_dataA >> (I_dataB[3:0]);
112     O_shldBranch <= 0;
113   end
114
115   JMPA : begin
116     int_result <= (op_lsb ? I_dataA : I_imm);
117     O_shldBranch <= 1;
118   end
119
120   JMPR : begin
121     int_result <= I_dataA;
122     O_shldBranch <= I_dataB({op_lsb , I_imm[1:0]});
123   end
124
125   endcase
126 end
127 endmodule

```

Verilog file length : 2,826 lines : 127 Ln : 122 Col : 67 Pos : 2,783 Windows (CR LF) UTF-8 INS

PS- O_shldBranch defined at the input and also in the ALU operations for the input one as

O_shldBranch <= 0;

Module pc_unit

```
2 'timescale 1ns / 1ps
3
4 // Module Definition
5 module pc_unit(
6     // Inputs
7     input I_clk,
8     input [1:0] I_opcode,
9     input [15:0] I_pc,
10    // Outputs
11    output reg [15:0] O_pc
12 );
13
14 // Initial Block
15 initial begin
16     O_pc <= 0;
17 end
18
19 // ALU Operations
20 always@(negedge I_clk) begin
21     case(I_opcode)
22         2'b00 :
23         2'b01 :
24         2'b10 :
25
26
27
28
29
```

```
7     input I_clk,
8     input [1:0] I_opcode,
9     input [15:0] I_pc,
10    // Outputs
11    output reg [15:0] O_pc
12 );
13
14 // Initial Block
15 initial begin
16     O_pc <= 0;
17 end
18
19 // Program Counter State
20 always@(negedge I_clk) begin
21     case(I_opcode)
22         2'b00 : O_pc <= O_pc;
23         2'b01 : O_pc <= O_pc + 1;
24         2'b10 : O_pc <= I_pc;
25         2'b11 : O_pc <= 0;
26     endcase
27 end
28
29 endmodule
```

Module reg_file

```
PAADV_VLSI_PROJECTS\16-Bit_Processor\reg_file.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
inst_dlev.v inst_ulev.v inst_vlev.v pc_ulev.v reg_file.v new.v

2 'timescale 1ns / 1ps
3
4 // Module Definition
5 module reg_file(
6     // Inputs
7     input I_clk,
8     input I_en,
9     input I_we,
10    input [3:0] I_selA,
11    input [3:0] I_selB,
12    input [3:0] I_selD,
13    input [3:0] I_dataD,
14     // Outputs
15    output [15:0] O_dataA,
16    output [15:0] O_dataB
17 );
18
19     // Internal Register declaration
20 reg [15:0] regs [7:0];
21
22     // Loop variable
23 integer count;
24
25 endmodule
```

```
Verilog file length : 401 lines : 25 Ln : 23 Col : 19 Pos : 389 Windows (CR LF) UTF-8 INS

PAADV_VLSI_PROJECTS\16-Bit_Processor\reg_file.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
inst_dlev.v inst_ulev.v inst_vlev.v pc_ulev.v reg_file.v new.v

23 integer count;
24
25     // Initialize registers
26 initial begin
27     O_dataA <= 0;
28     O_dataB <= 0;
29
30     for(count = 0; count < 8; count = count + 1) begin
31         regs[count] <= 0;
32     end
33
34     // Assigning correct values to OP regs
35 always@ (negedge I_clk) begin
36
37     if(I_en) begin
38         if(I_we)
39             regs[I_selD] <= I_dataD;
40
41         O_dataA <= regs[I_selA];
42         O_dataB <= regs[I_selB];
43     end
44
45 endmodule
```

Verilog file length : 775 lines : 45 Ln : 37 Col : 23 Pos : 650 Windows (CR LF) UTF-8 INS

Module fake_ram

```

P:\ADV_VLSI_PROJECTS\16-Bit_Processor\fake_ram.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
inst_decoder.v inst_unit.v inst_main.v reg_file.v fake_ram.v
3
4 // Module Definition
5 module fake_ram(
6     // Inputs
7     input I_clk,
8     input I_we,
9     input [15:0] I_addr,
10    input [15:0] I_data,
11    // Outputs
12    output [15:0] O_data
13 );
14 //Memory declaration
15 reg [15:0] mem [8:0];
16
17 // Initialize registers
18 initial begin
19
20     mem[0] = 16'b1000000011111110;
21     mem[1] = 16'b1000100111101101;
22     mem[2] = 16'b0010001000100000;
23     mem[3] = 16'b1000001100000001;
24     mem[4] = 16'b1000010000000001;
25     mem[5] = 16'b0000001101110000;
26     mem[6] = 16'b11000000000000101;

```

Verilog file length : 817 lines : 43 Ln : 14 Col : 24 Pos : 226 Windows (CR LF) UTF-8 INS

```

P:\ADV_VLSI_PROJECTS\16-Bit_Processor\fake_ram.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
inst_decoder.v inst_unit.v inst_main.v reg_file.v fake_ram.v
21     mem[1] = 16'b1000100111101101;
22     mem[2] = 16'b0010001000100000;
23     mem[3] = 16'b1000001100000001;
24     mem[4] = 16'b1000010000000001;
25     mem[5] = 16'b0000001101110000;
26     mem[6] = 16'b11000000000000101;
27     mem[7] = 0;
28     mem[8] = 0;
29
30     O_data = 16'b0000000000000000;
31 end
32
33 // RAM Operation
34 always@(negedge I_clk) begin
35
36     if(I_we) begin
37         mem[I_addr[15:0]] <= I_data;
38     end
39     O_data <= mem[I_addr[15:0]];
40
41 end
42
43 endmodule

```

Verilog file length : 817 lines : 43 Ln : 28 Col : 13 Pos : 598 Windows (CR LF) UTF-8 INS

TESTING CODES

Module decoder unit_test

```
PADEV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\decoder_unittest.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
take_rnm.v inst_decoder.v pc_untv.v reg_file.v reg_v.v ctrl_untv.v decoder_unittest.v regfile_unittest.v

1 // TimeScale
2 'timescale 1ns / 1ps
3
4 // Module Definition
5 module decoder_unittests();
6     // Variable declaration
7     //Regs
8     reg I_Clk;
9     reg I_En;
10    reg [15:0] I_Inst;
11    //Wires
12    wire [4:0] O_Aluop;
13    wire [3:0] O_SelA;
14    wire [3:0] O_SelB;
15    wire [3:0] O_SelD;
16    wire [15:0] O_Imm;
17    wire O_Regwe;
18
19 inst_decoder(
20     // Inputs
21     I_Clk,
22     I_En,
23     I_Inst,
24     // Outputs
25
Verilog file length : 667 lines : 51 Ln : 8 Col : 15 Pos : 136 Windows (CR LF) UTF-8 INS
```

Module regfile unit_test

```
PADEV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\regfile_unittest.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
take_rnm.v inst_decoder.v pc_untv.v reg_file.v reg_v.v ctrl_untv.v decoder_unittest.v regfile_unittest.v

3
4 // Module Definition
5 module regfile_unittest();
6     // Variable declaration
7     //Regs
8     reg I_clk;
9     reg I_dataD;
10    reg I_en;
11    reg I_selA;
12    reg I_selB;
13    reg I_selD;
14    reg I_we;
15
16
17
18
19 reg_file(
20     // Inputs
21     I_clk,
22     I_en,
23     I_we,
24     I_selA,
25     I_selB,
26     I_selD,
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
288
289
289
290
291
292
293
294
295
296
297
298
299
299
300
301
302
303
304
305
306
307
308
309
309
310
311
312
313
314
315
316
317
318
319
319
320
321
322
323
324
325
326
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
749
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
849
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
889
889
890
891
892
893
894
895
896
897
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
949
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2088
2089
2089
2090
2091
2092
2093

```

```

12 reg I_selB;
13 reg I_selD;
14 reg I_we;
15 //Wires
16 wire O_dataA;
17 wire O_dataB;
18
19 reg_file(
20     // Inputs
21     I_clk,
22     I_en,
23     I_we,
24     I_selA,
25     I_selB,
26     I_selD,
27     I_dataD,
28     // Outputs
29     O_dataA,
30     O_dataB
31 );
32
33
34
35
Verilog file length : 407 lines : 37 Ln : 34 Col : 1 Pos : 393 Windows (CR LF) UTF-8 INS

```

```

29 O_dataA,
30 O_dataB
31 );
32
33 /* Testing Process for reference
34 * 1) Read r0 and r1, write 0xFFFF to r0
35 * 2) Ensure 0xFFFF appears on data out line, write 0x2222 to r2
36 * 3) Write 0x3333 to r2, testing multiple writes to same location
37 * 4) Set up as tho writing 0xFEED to r0 but dont enable the I_we
38 * 5) Write 0x4444 to r4, ensure 0xFEED was not written to r0
39 * 6) After waiting multiple clock cycles, read r4 on both output A and B
40 */
41
42 initial begin
43     //Reset all Inputs
44     I_clk = 1'b0;
45     I_dataD = 0;
46     I_en = 0;
47     I_selA = 0;
48     I_selB = 0;
49     I_selD = 0;
50     I_we = 0;
51
52     //Start Test

```

length : 1,108 lines : 63 Ln : 34 Col : 6 Pos : 432 Windows (CR LF) UTF-8 INS

```

41
42     initial begin
43         //Reset all Inputs
44         I_clk = 1'b0;
45         I_dataD = 0;
46         I_en = 0;
47         I_selA = 0;
48         I_selB = 0;
49         I_selD = 0;
50         I_we = 0;
51
52         //Start Test
53         //Time = 7
54         #7
55         I_en = 1'b1;
56
57         I_selA = 3'b000;
58         I_selB = 3'b001;
59         I_selD = 3'b000;
60
61         I_dataD = 16'hFFFF;
62         I_we = 1'b1;
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84     end

```

Verilog file

length : 1,175 lines : 69 Ln : 53 Col : 19 Pos : 1,017 Windows (CR LF) UTF-8 INS


```

61         I_dataD = 16'hFFFF;
62         I_we = 1'b1;
63
64         //Time = 17
65         #10;
66         I_we = 1'b0;
67         I_selD = 3'b010;
68         I_dataD = 16'h2222;
69
70         //Time = 27
71         #10;
72         I_we = 1;
73
74         //Time = 37
75         #10;
76         I_dataD = 16'h3333;
77
78         //Time = 47
79         #10;
80         I_we = 0;
81         I_selD = 3'b000;
82         I_dataD = 16'hFEED;
83
84     end

```

Verilog file

length : 1,431 lines : 88 Ln : 82 Col : 28 Pos : 1,405 Windows (CR LF) UTF-8 INS

```

P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\regfile_unittest.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
File Ram.v inst_decy.v pc_unit.v reg_file.v slice.v cfrUnit.v decoder_unittest.v regfile_unittest.v
78 //Time = 47
79 #10;
80 I_we = 0;
81 I_selD = 3'b000;
82 I_dataD = 16'hFEED;
83
84 //Time = 57
85 #10;
86 I_selD = 3'b100;
87 I_dataD = 16'h4444;
88
89 //Time = 67
90 #10;
91 I_we = 1;
92
93 //Time = 117
94 #50;
95 I_selA = 3'b100;
96 I_selB = 3'b100;
97
98 end
99
100 //Clock |
101 always begin
Verilog file length : 1,669 lines : 108 Ln : 100 Col : 13 Pos : 1,606 Windows (CR LF) UTF-8 INS
P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\regfile_unittest.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
File Ram.v inst_decy.v pc_unit.v reg_file.v slice.v cfrUnit.v decoder_unittest.v regfile_unittest.v
86 I_selD = 3'b100;
87 I_dataD = 16'h4444;
88
89 //Time = 67
90 #10;
91 I_we = 1;
92
93 //Time = 117
94 #50;
95 I_selA = 3'b100;
96 I_selB = 3'b100;
97
98 end
99
100 //Clock generation
101 always begin
102 #5;
103 I_Clk = ~I_Clk;
104 end
105
106
107
108 endmodule
Verilog file length : 1,679 lines : 108 Ln : 99 Col : 1 Pos : 1,595 Windows (CR LF) UTF-8 INS

```

Module regfile main_test

```
4 // Module Definition
5 module main_test();
6     // Variable declaration
7     //Regs
8     reg clk;
9     reg reset;
10    reg we = 0;
11    reg [15:0] dataI = 0;
12    //Wire
13    wire [2:0] selA;
14    wire [2:0] selB;
15    wire [2:0] selD;
16    wire [15:0] dataA;
17    wire [15:0] dataB;
18    wire [15:0] dataD;
19    wire [4:0] aluop;
20    wire [7:0] imm;
21    wire [15:0] dataO;
22    wire [1:0] opcode;
23    wire [15:0] pcO;
24
25    wire shldBranch;
26    wire enfetch;
27    wire enalu;
```

```
22    wire [1:0] opcode;
23    wire [15:0] pcO;
24
25    wire shldBranch;
26    wire enfetch;
27    wire enalu;
28    wire endec;
29    wire enmem;
30    wire enrgrd;
31    wire engrwr;
32
33    //Assignments
34    assign opcode = (reset) ? 2'b11 : ((shldBranch) ? 2'b10 : ((we) ? 2'b01 : 2'b00));
35
36 //Instantiations
37 reg_file main_reg(
38     // Inputs
39     clk,
40     en,
41     we,
42     selA,
43     selB,
44     selD,
45     dataD,
```

```
Verilog file  length : 1,209  lines : 110  Ln : 25  Col : 20  Sel : 10 | 1  Windows (CR LF)  UTF-8  INS
*PAADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_testv - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
take_ram.v inst_decs.v pc_uvm.v reg_file.v alu.v ctrl_units.v decoder_units.v regfile_units.v main_testv.v

19      wire [4:0] aluop;
20      wire [7:0] imm;
21      wire [15:0] dataA;
22      wire [1:0] opcode;
23      wire [15:0] pc0;
24
25      wire shldBranch;
26      wire enfetch;
27      wire enalu;
28      wire endec;
29      wire enmem;
30      wire enrgrd;
31      wire enrgwr;
32
33 //Instantiations
34 reg_file main_reg(
35   // Inputs
36   clk,
37   en,
38   we,
39   selA,
40   selB,
41   selD,
42   dataD,
```

```
Verilog file  length : 1,209  lines : 110  Ln : 25  Col : 20  Sel : 10 | 1  Windows (CR LF)  UTF-8  INS
*PAADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_testv - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
take_ram.v inst_decs.v pc_uvm.v reg_file.v alu.v ctrl_units.v decoder_units.v regfile_units.v main_testv.v

34 reg_file main_reg(
35   // Inputs
36   clk,
37   en,
38   we,
39   selA,
40   selB,
41   selD,
42   dataD,
43   // Outputs
44   dataA,
45   dataB
46 );
47
48 inst_dec main_inst(
49   // Inputs
50   clk,
51   en,
52   inst,
53   // Outputs
54   aluop,
55   selA,
56   selB,
57   selD,
```

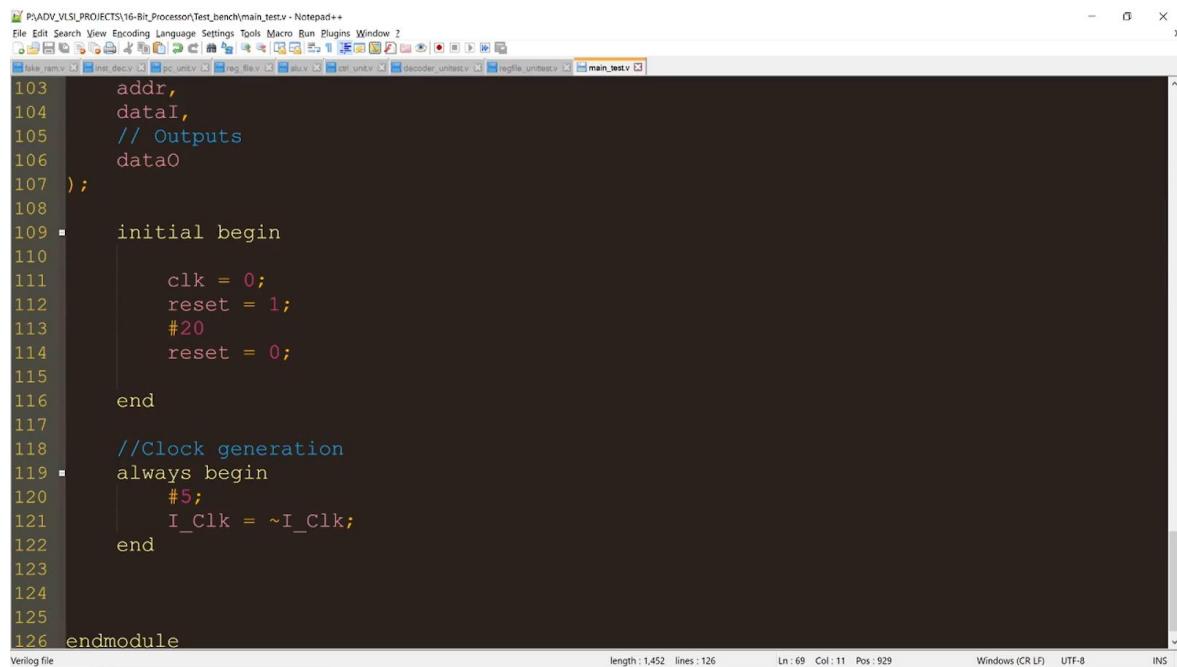
```
*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_testv - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
file_name.v inst_decoder.v pc_univ.v reg_file.v alu.v ctrl_unit.v decoder_untest.v regfile_untest.v main_testv.v  
43     selB,  
44     selD,  
45     dataD,  
46     // Outputs  
47     dataA,  
48     dataB  
49 );  
50  
51 inst_dec main_inst(  
52     // Inputs  
53     clk,  
54     en,  
55     inst,  
56     // Outputs  
57     aluop,  
58     sela,  
59     selB,  
60     selD,  
61     imm,  
62     regwe  
63 );  
64  
65 alu main_alu(  
66     // Inputs  
Verilog file length : 1,261 lines : 113 Ln : 34 Col : 34 Pos : 565 Windows (CR LF) UTF-8 INS
```

```
*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_testv - Notepad++  
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?  
file_name.v inst_decoder.v pc_univ.v reg_file.v alu.v ctrl_unit.v decoder_untest.v regfile_untest.v main_testv.v  
61     imm,  
62     regwe  
63 );  
64  
65 alu main_alu(  
66     // Inputs  
67     clk,  
68     en,  
69     aluop,  
70     ,  
71     dataB,  
72     imm,  
73     // Outputs  
74     dataResult,  
75     shldBranch  
76 );  
77  
78 ctrl_unit main_ctrl(  
79     // Inputs  
80     clk,  
81     reset,  
82     // Outputs  
83     enfetch,  
84     endec,  
Verilog file length : 1,292 lines : 113 Ln : 34 Col : 67 Pos : 598 Windows (CR LF) UTF-8 INS
```

```

*D:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_testv - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
file_name.v inst_decy pc_unit reg_filev reg_filev decoder_unitev reg_filev main_testv
54 `ctrl unit main_ctrl(
55     // Inputs
56     I_clk,
57     I_reset,
58     // Outputs
59     O_enfetch,
60     O_endec,
61     O_enrgrd,
62     O_enalu,
63     O_enrgwr,
64     O_enmem
65 );
66
67 `pc_unit pc_main(
68     // Inputs
69     I_clk,
70     I_opcode,
71     I_pc,
72     // Outputs
73     O_pc
74 );
75
76
77 Verilog file length : 798 lines : 78 Ln : 76 Col : 1 Pos : 785 Windows (CR LF) UTF-8 INS
*D:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_testv - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
file_name.v inst_decy pc_unit reg_filev reg_filev decoder_unitev reg_filev main_testv
71 `pc_unit pc_main(
72     // Inputs
73     clk,
74     opcode,
75     pc,
76     // Outputs
77     pc
78 );
79
80 `fake_ram main_ram(
81     // Inputs
82     clk,
83     we,
84     addr,
85     data,
86     // Outputs
87     data
88 );
89
90
91
92
93 endmodule
Verilog file length : 875 lines : 93 Ln : 10 Col : 11 Sel : 2|1 Windows (CR LF) UTF-8 INS

```



```

103     addr,
104     dataI,
105     // Outputs
106     dataO
107   );
108
109   initial begin
110     clk = 0;
111     reset = 1;
112     #20
113     reset = 0;
114
115   end
116
117   //Clock generation
118   always begin
119     #5;
120     I_Clk = ~I_Clk;
121   end
122
123
124
125
126 endmodule

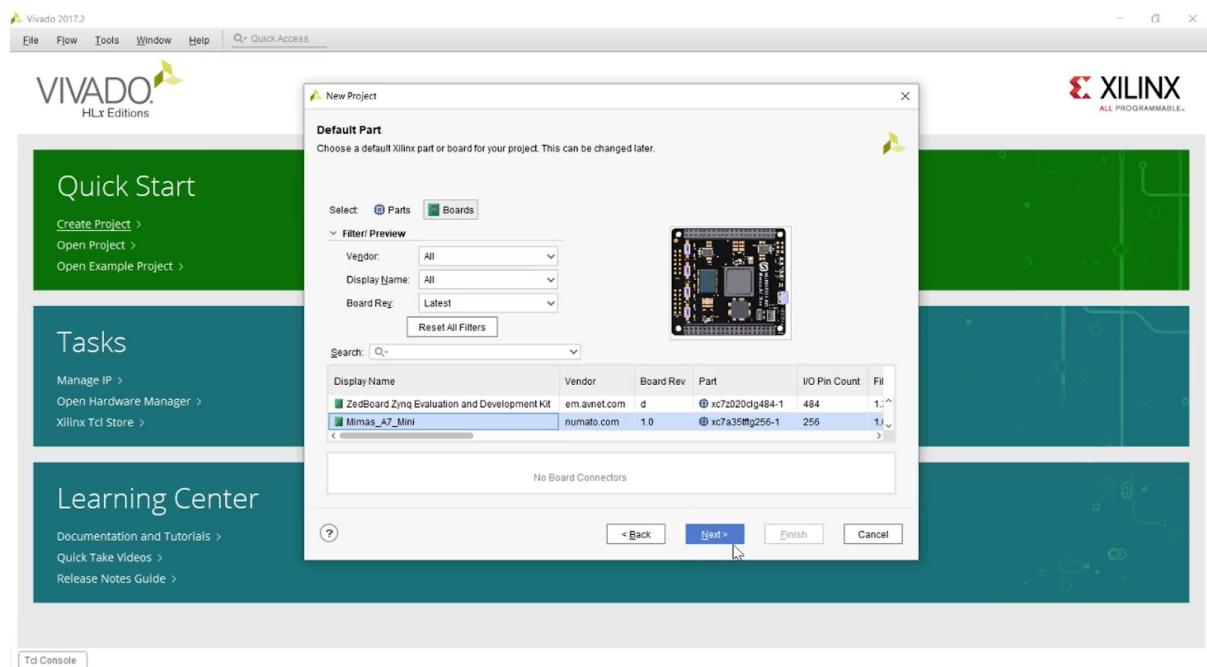
```

Verilog file length : 1,452 lines : 126 Ln : 69 Col : 11 Pos : 929 Windows (CR LF) UTF-8 INS

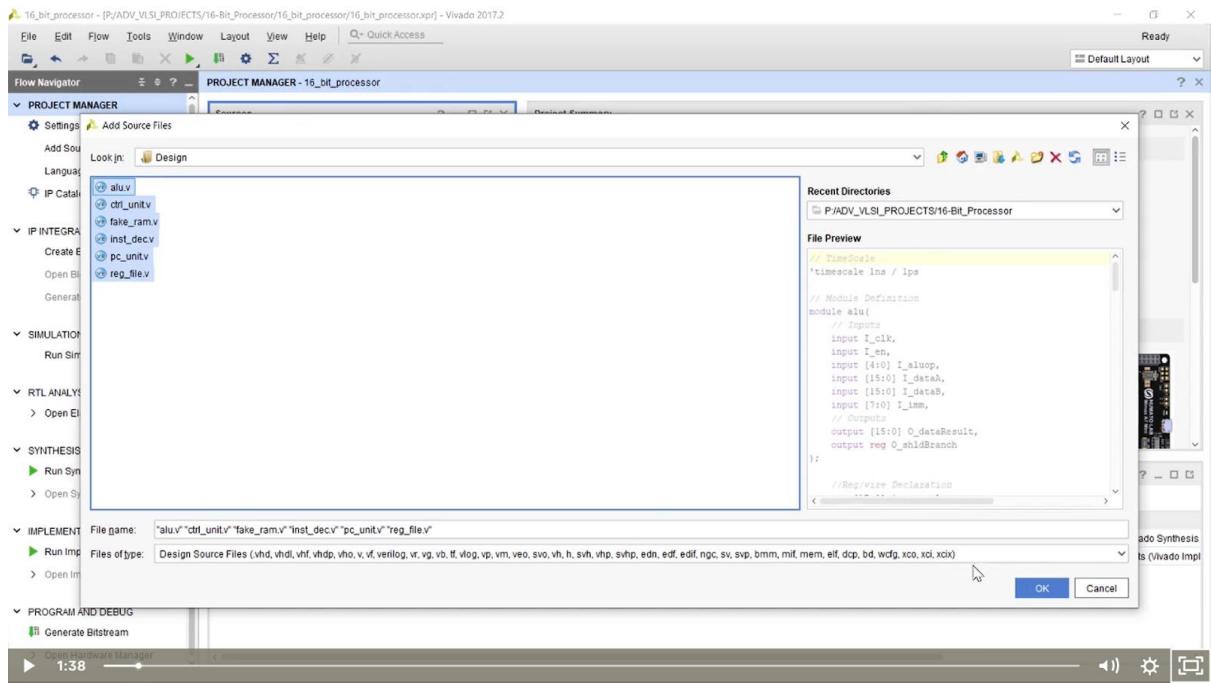
Time to simulate!

Create a project with a name.

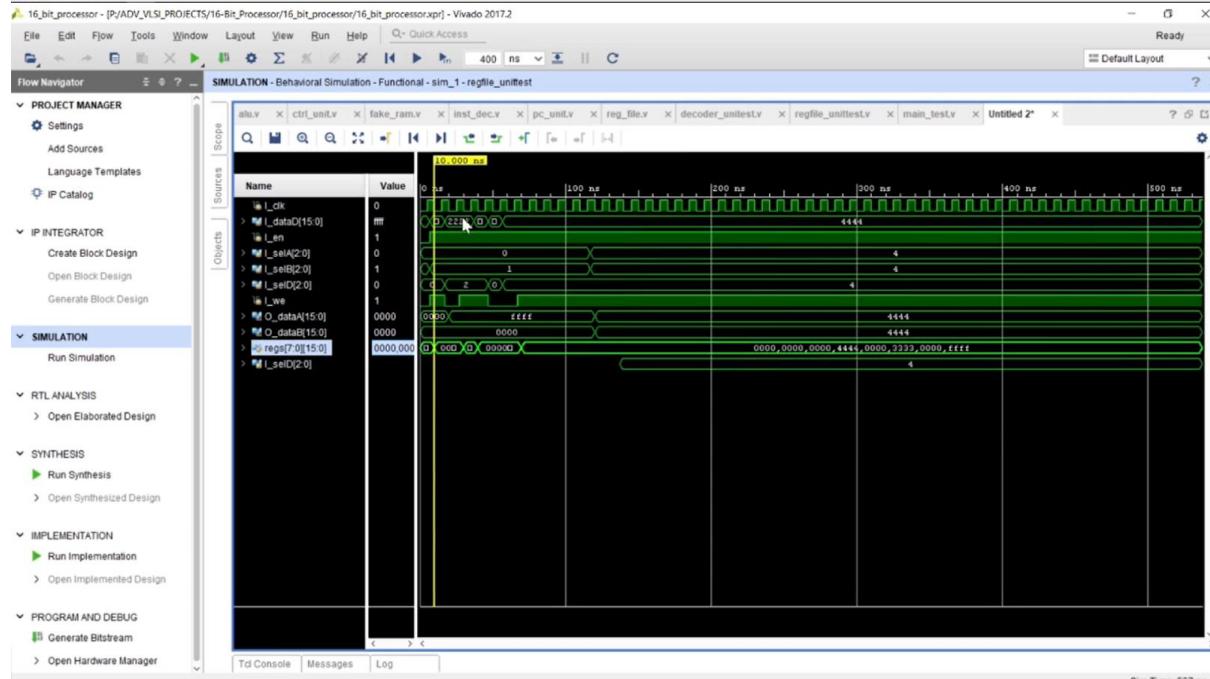
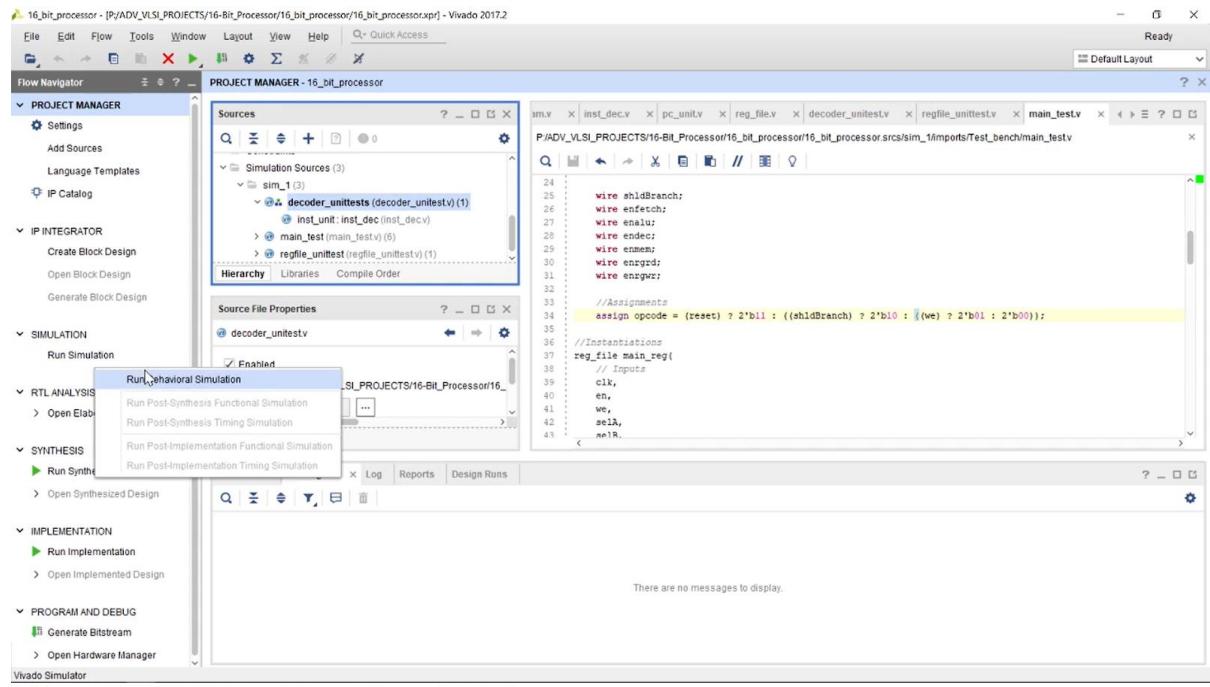
Chose the board, Mimas_A7_Mini



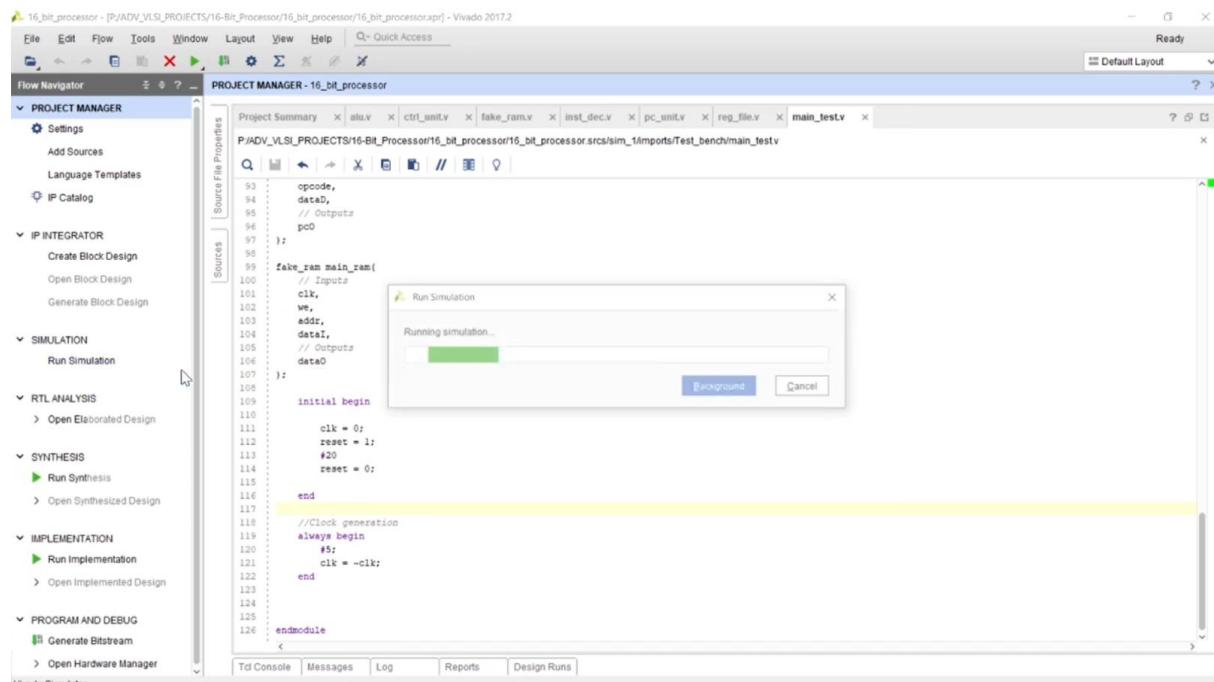
Import all the verilog files



After successfully compiling the errors, run Behavioral simulation model of our instruction decoder.



Run the main test bench simulation



Top Window (Vivado 2017.2):

The top window shows the Vivado interface for a project named "16_bit_processor". The left sidebar contains a tree view of the project structure under "PROJECT MANAGER" and "SIMULATION". The main area is titled "SIMULATION - Behavioral Simulation - Functional - sim_1 - main_test". It displays a logic analyzer-like waveform viewer with multiple signal traces. A table on the left lists signal names and their current values. The time axis ranges from 0 ns to 900 ns.

Name	Value
clk	0
reset	0
data[15:0]	0000
sel[2:0]	0
sel[2:0]	0
sel[2:0]	0
data[15:0]	0000
data[15:0]	0000
data[15:0]	2222
aluOp[4:0]	00
imm[7:0]	00
data[15:0]	XXXX
opcode[1:0]	X
pc[15:0]	0000
shiftBranch	Z
enfetch	0
endec	0
engrid	1
enalu	0
engrw	0
enmem	0
en	2
rewe	0
inst	2
dataResult	Z

Bottom Window (Vivado 2017.2):

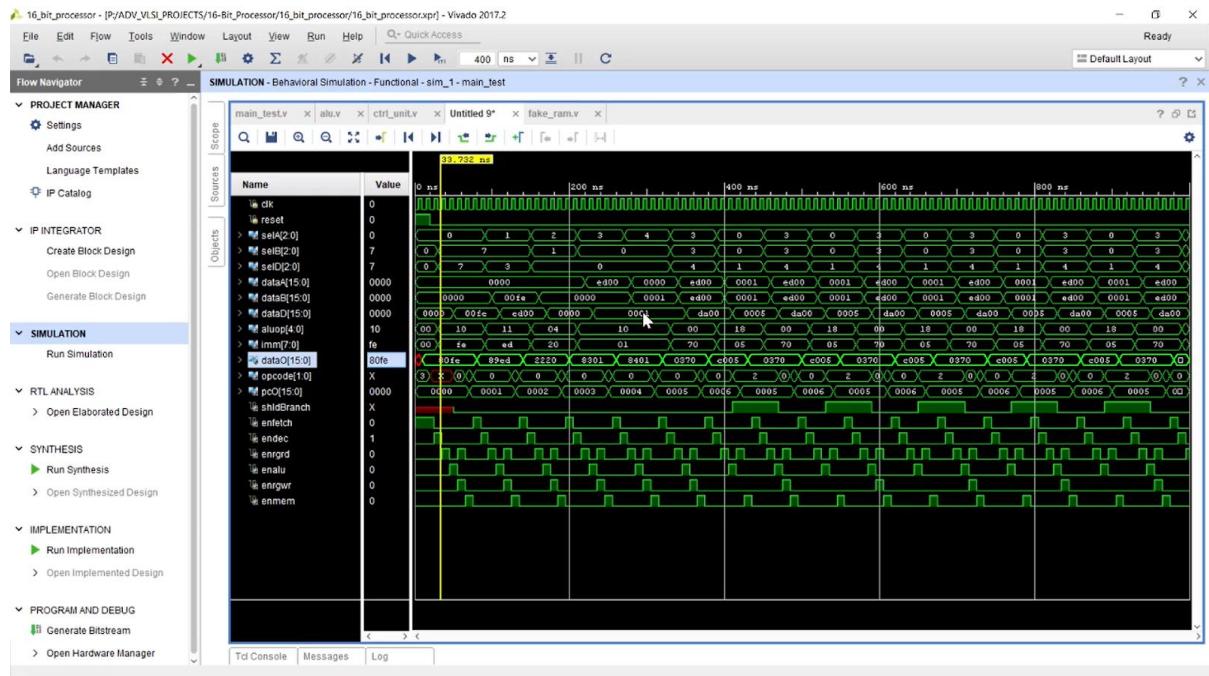
The bottom window also shows the Vivado interface for the same project. The left sidebar is identical. The main area is titled "SIMULATION - Behavioral Simulation - Functional - sim_1 - main_test". It displays a code editor window showing Verilog source code for "fake_ram.v". The code defines a memory interface with methods for reading and writing data to memory.

```

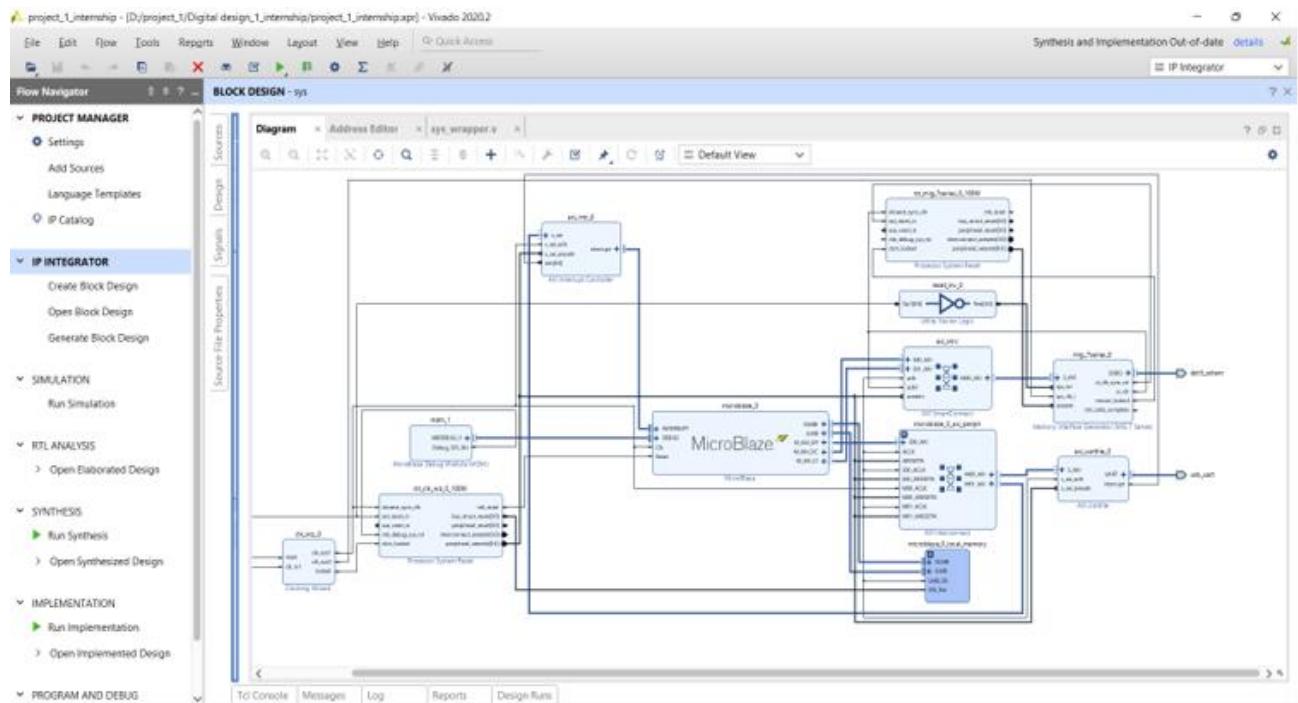
main_test.v | alu.v | ctrl_unit.v | inst_dec.v | pc_unit.v | reg_file.v | main_test.v | Untitled 9 | fake_ram.v
P/ADV_VLSI_PROJECTS/16-Bit_Processor/16_bit_processor/srcs/sim_1/imports/Test_bench/main_test.v

97:     opcode,
98:     dataD,
99:     // Outputs
100:    pcO
101: );
102
103     fake_ram main_ram(
104         .Inputs
105         .clk(clk),
106         .ram_WE,
107         .pcO,
108         .dataI,
109         // Outputs
110         .dataO
111     );
112
113     initial begin
114         clk = 0;
115         reset = 1;
116         #20
117         reset = 0;
118     end
119
120
121     //Clock generation
122     always begin
123         #5;
124         clk = ~clk;
125     end
126
127
128
129
130 endmodule

```



5 Digital Design



<https://drive.google.com/drive/folders/1j1CQQcNiZkKO7DBlcB5UqeO1-nQJEAIK?usp=sharing>