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

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

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.


VYORIUS
INTERNSHIP
CERTIFICATE
THIS IS TO CERTIFY THAT
Pushpal Das
has successfully completed internship program in IoT..
from 10th Jan, 2022 to 10th Mar, 2022. During the internship, the student
was found to be dedicated, hardworking and diligent.
istop


CERTIFICATE
OF PROJECT COMPLETION
VYORIUS
THIS CERTIFICATE IS PROUDLY PRESENTED TO
Pushpal Das
has successfully undergone Industrial Program on IoT..
from VYORIUS from 20th Nov, 2021 to 20th Jan, 2022 and
successfully completed the projects on
- IoT based on smart Agriculture System.
- Obstacle Avoidance Robot using Ultrasonic Sensor
- RGB Pattern
Under the guidance of the mentor and company representative
22-Jan-2022
DATE
istop
PROJECT HEAD
VYO-2201000099

Techfest™
INDIAN INSTITUTE OF TECHNOLOGY, BOMBAY

VYORIUS

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.

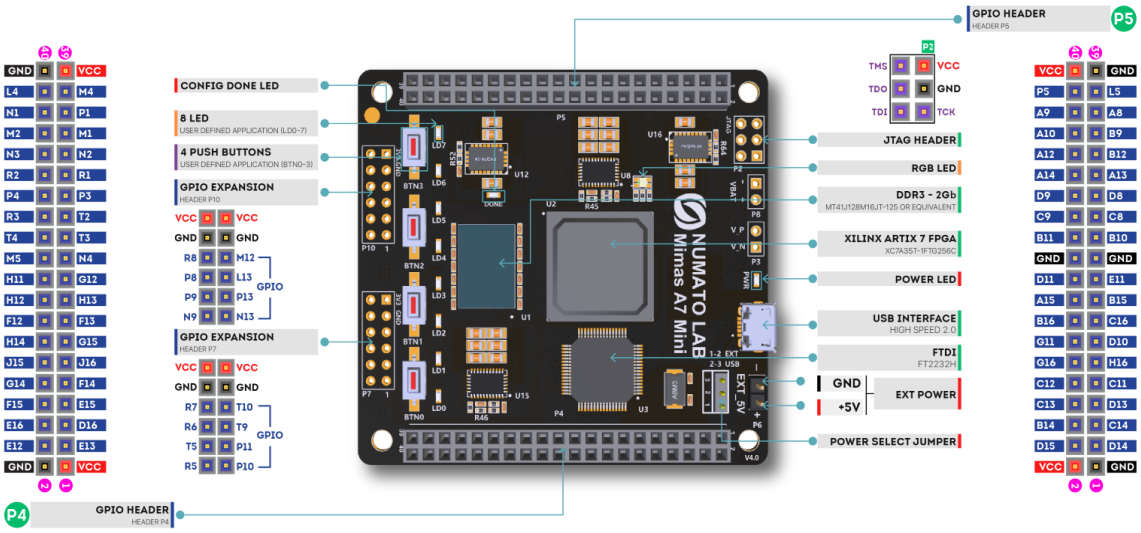
Features,

- 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	**XC7A35T – 1FTG256C**
Memory	**DDR3 – 2Gb**
Configuration Options	**JTAG**, **USB**
Host Interface	**USB 2.0**
Primary Clock Frequency	**100MHz**
Number Of GPIOs (Max)	**70**
Number Of Diff Pairs	**35**

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.[1][5][6][7] Vivado represents a ground-up rewrite and re-thinking of the entire design flow (compared to ISE).[8][9][10]

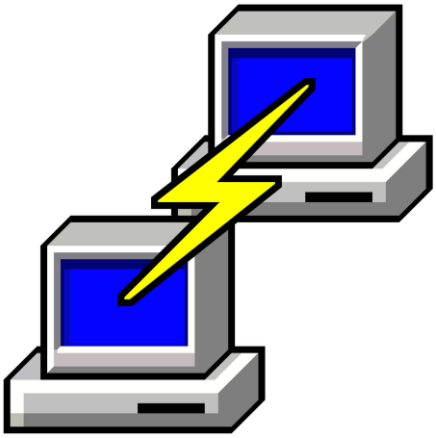
Like the later versions of ISE, Vivado includes the in-built logic simulator ISIM.[11] Vivado also introduces high-level synthesis, with a toolchain that converts C code into programmable logic.[6]

Replacing the 15-year-old ISE with Vivado Design Suite took 1000 person-years and cost US$200 million.

NUMATO LAB®

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++

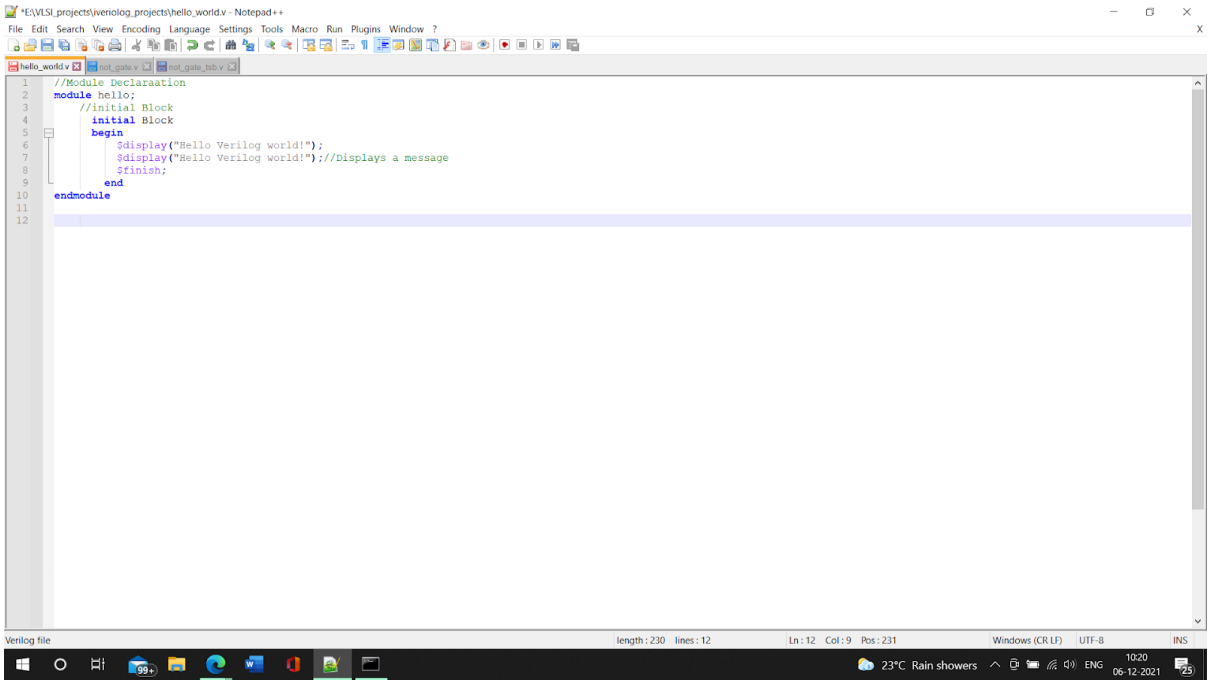
Step 1- Write the verilog program in notepad ++

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

Iverilog ,should be already installed to interface command promt 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

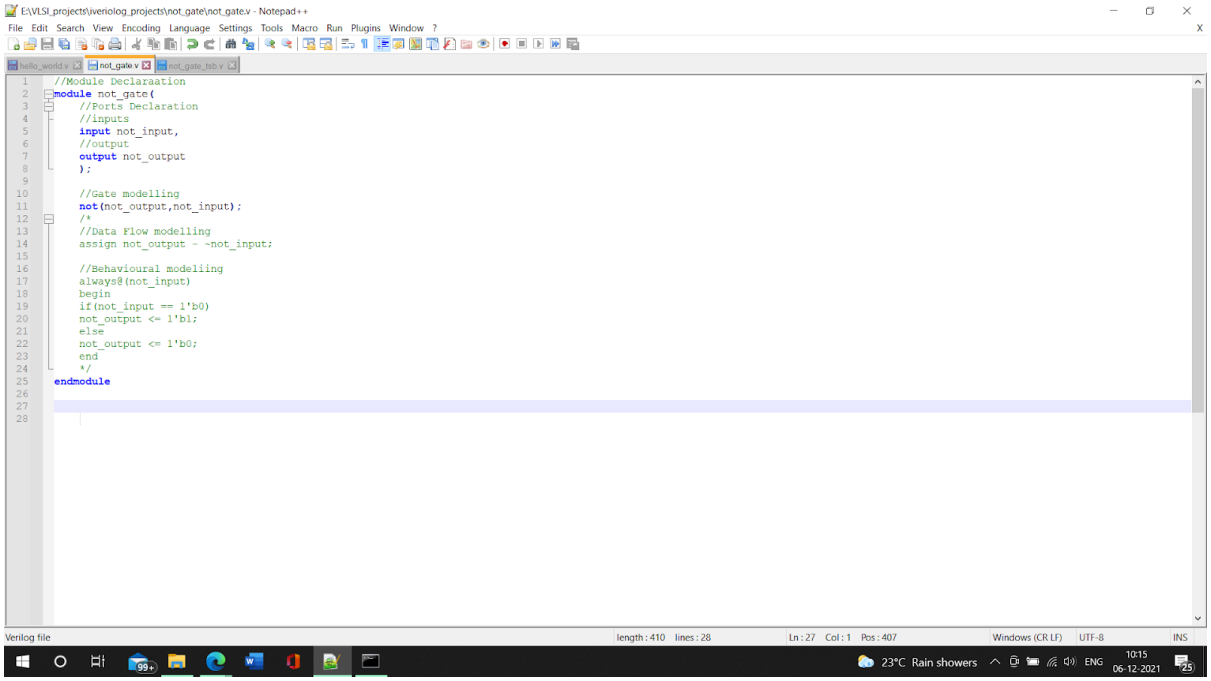


Output-The reading is displays in command prompt after opening the file directory.

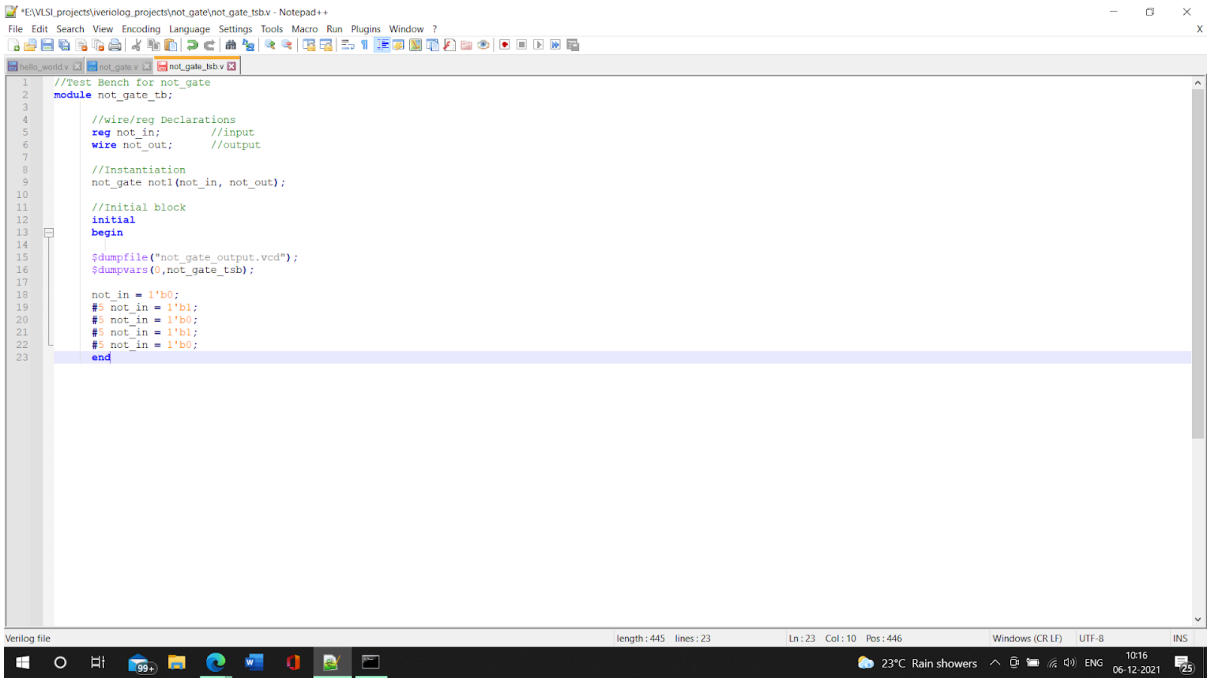
Step 4-Now let's do it with not gate and get the output,

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

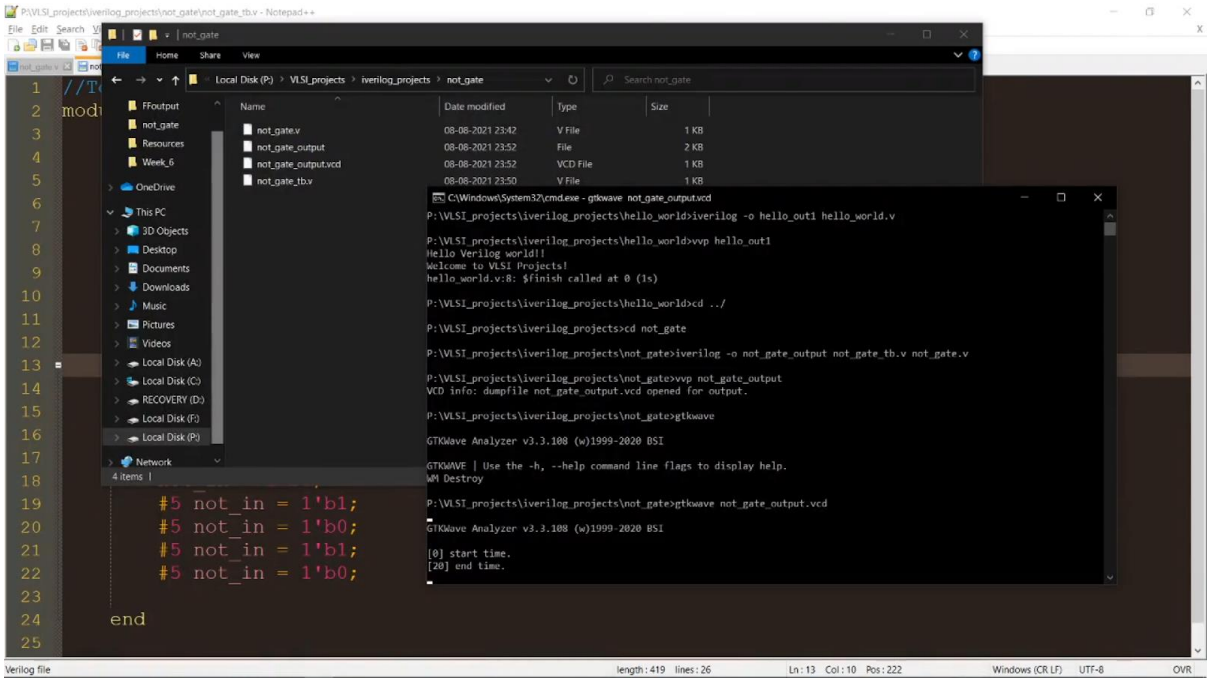
File1


E:\VLSI_projects\iveriolog_projects\not_gate\not_gate.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
hello_world.v
not_gate.v
not_gate_tsb.v
//Module Declaraation
module not_gate(
//Ports Declaration
//inputs
input not_input,
//output
output not_output
);
//Gate modelling
not(not_output,not_input);
/*
//Data Flow modelling
assign not_output = ~not_input;
//Behavioural modelling
always@(not_input)
begin
if(not_input == 1'b0)
not_output <= 1'b1;
else
not_output <= 1'b0;
end
*/
endmodule
Verilog file
length : 410 lines : 28
Ln : 27 Col : 1 Pos : 407
Windows (CR LF)
UTF-8
INS
23°C Rain showers
ENG
10:15
06-12-2021

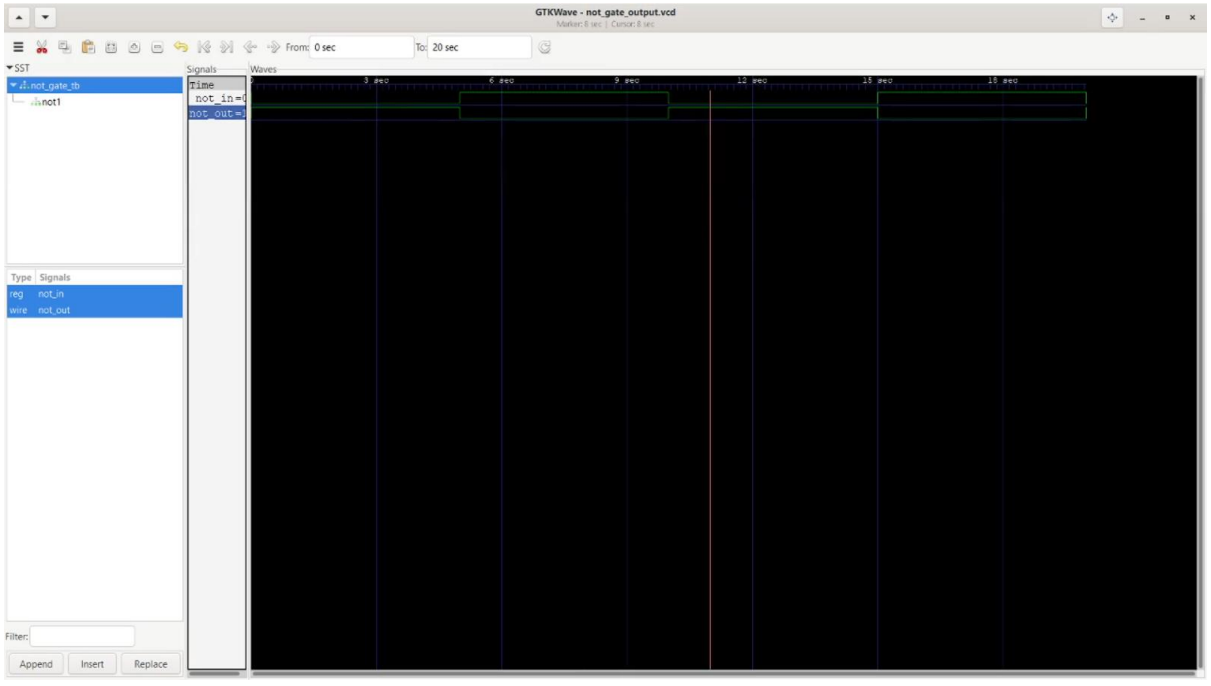
File2


*E:\VLSI_projects\iveriolog_projects\not_gate\not_gate_tsb.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
hello_world.v
not_gate.v
not_gate_tsb.v
//Test Bench for not_gate
module not_gate_tb;
//wire/reg Declarations
reg not_in; //input
wire not_out; //output
//Instantiation
not_gate not1(not_in, not_out);
//Initial block
initial
begin
$dumpfile("not_gate_output.vcd");
$dumpvars(0,not_gate_tsb);
not_in = 1'b0;
#5 not_in = 1'b1;
#5 not_in = 1'b0;
#5 not_in = 1'b1;
#5 not_in = 1'b0;
end
Verilog file
length : 445 lines : 23
Ln : 23 Col : 10 Pos : 446
Windows (CR LF)
UTF-8
INS
23°C Rain showers
ENG
10:16
06-12-2021

COMMAND PROMPT



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



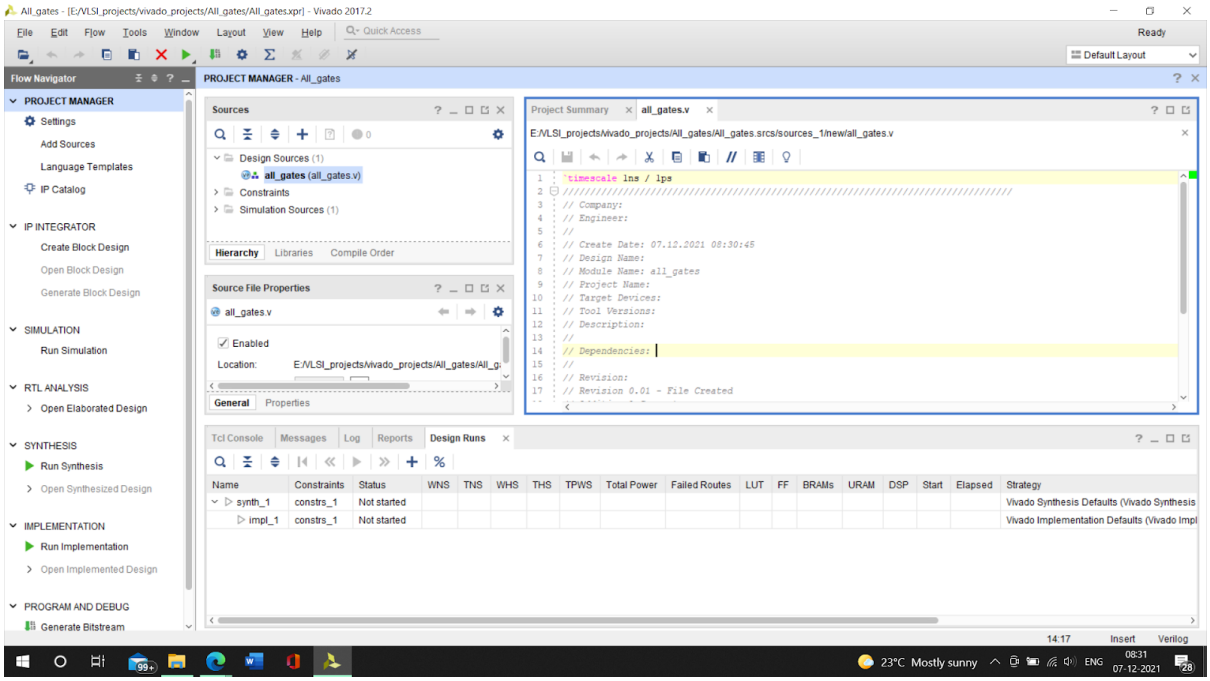
NOW WITH ALL BASIC GATES

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

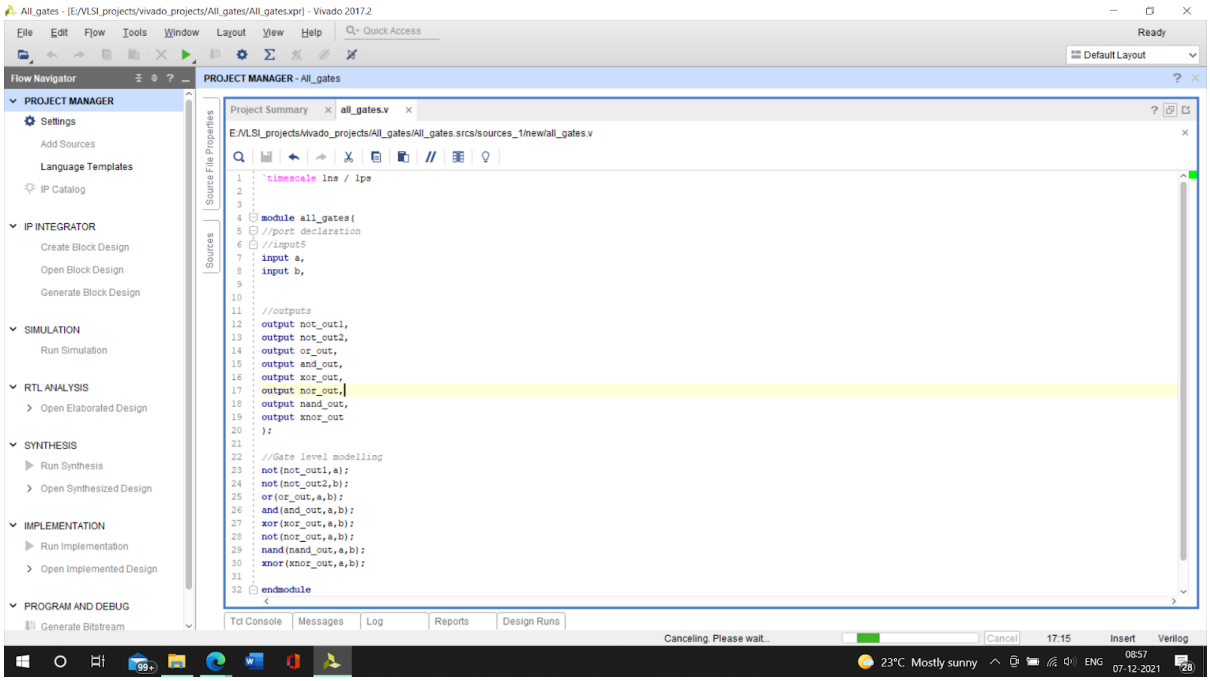
numato (Numato Lab) (github.com)

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 t numato labs and extracting it and pasting it to xilinx vivavo software.

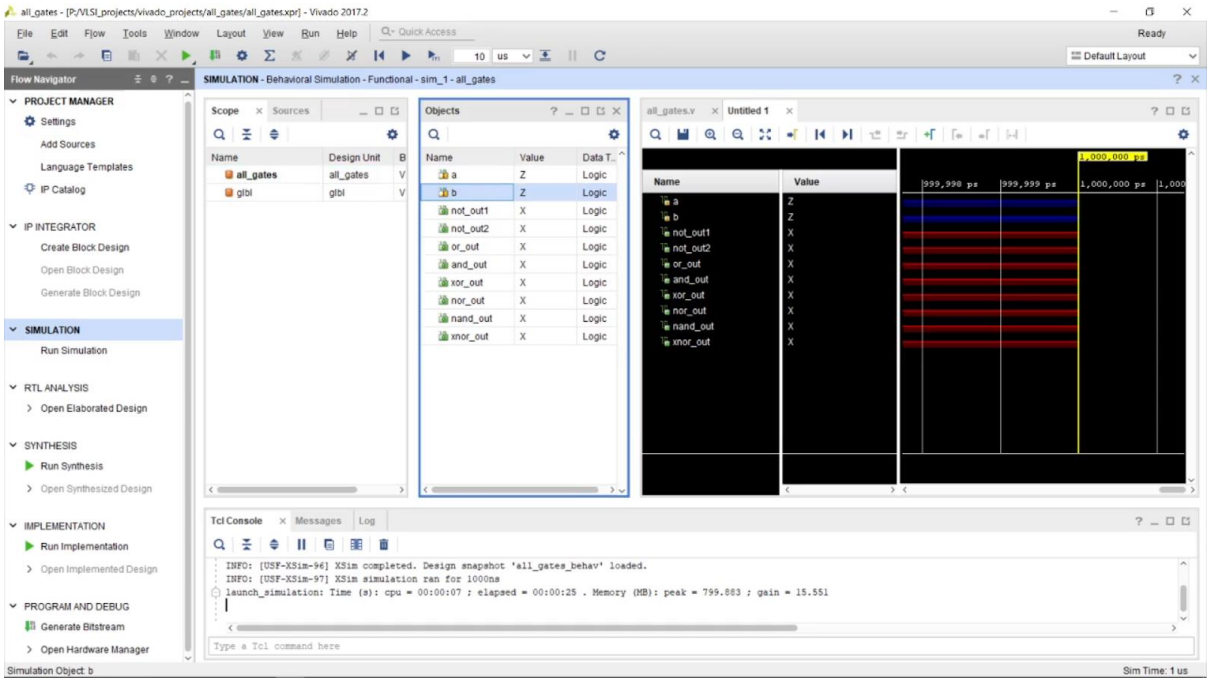
After that we have to open the vivado software where we have to create a new project and save in a directory, and after that through the process we have to select the board on it, that is Mimas A7 mini.



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



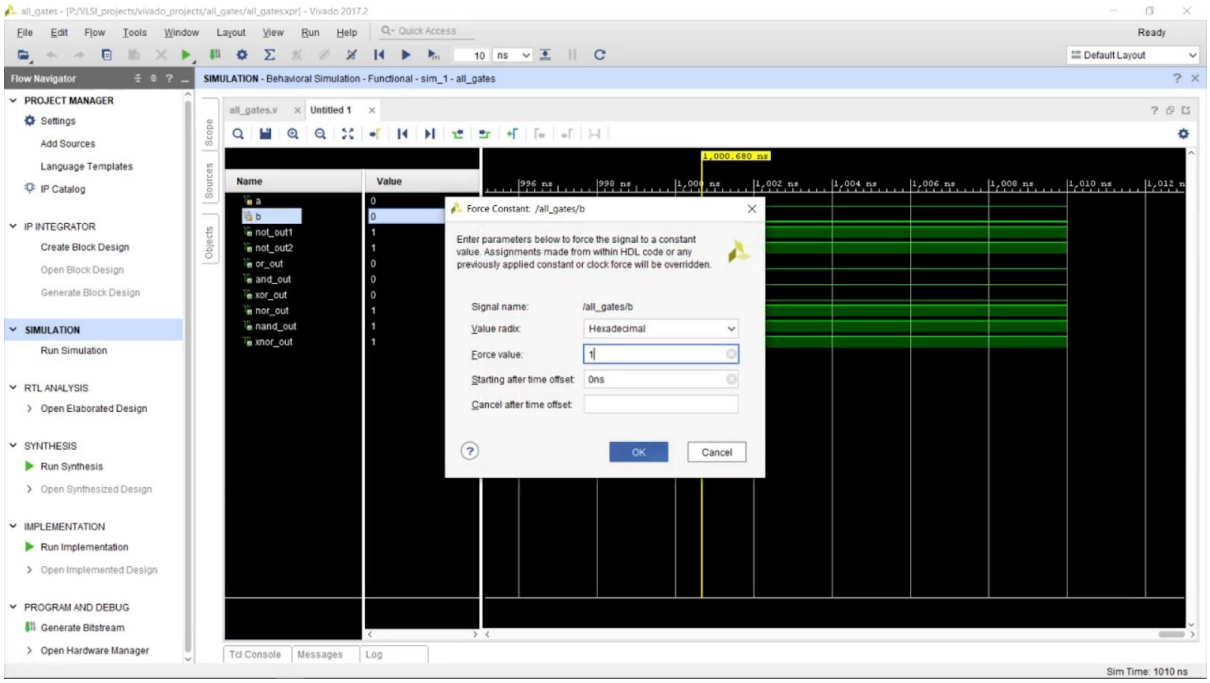
After that we have to run simulation to our program



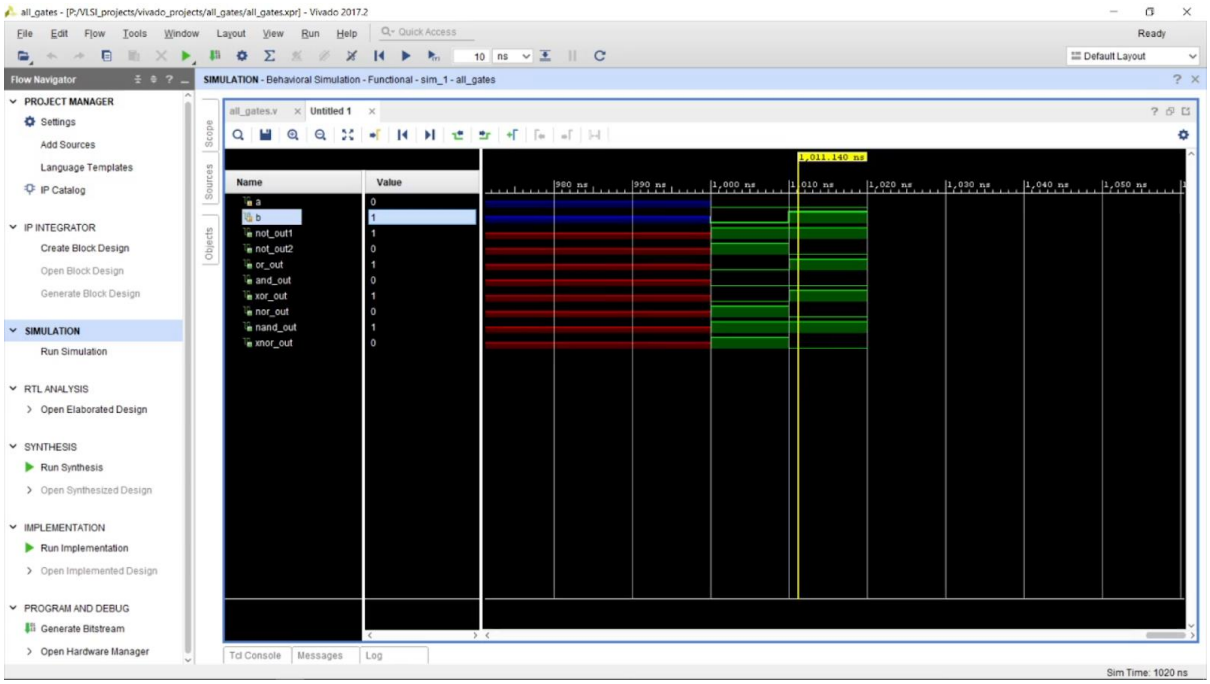
As we can see the inputs and the outputs.

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

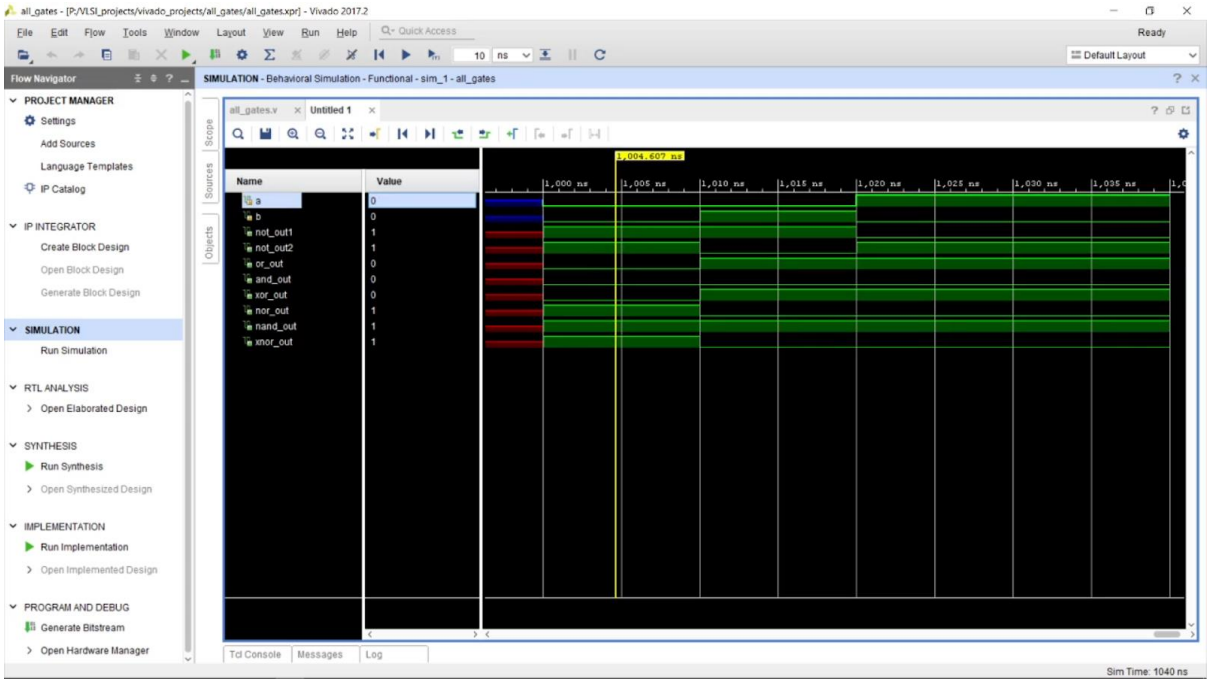
We can choose any type of value as hexadecimal or decimal as we are only going to deal with 0 and 1, Force value for a will be 0 and b will be 1, with 10 ns .



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.



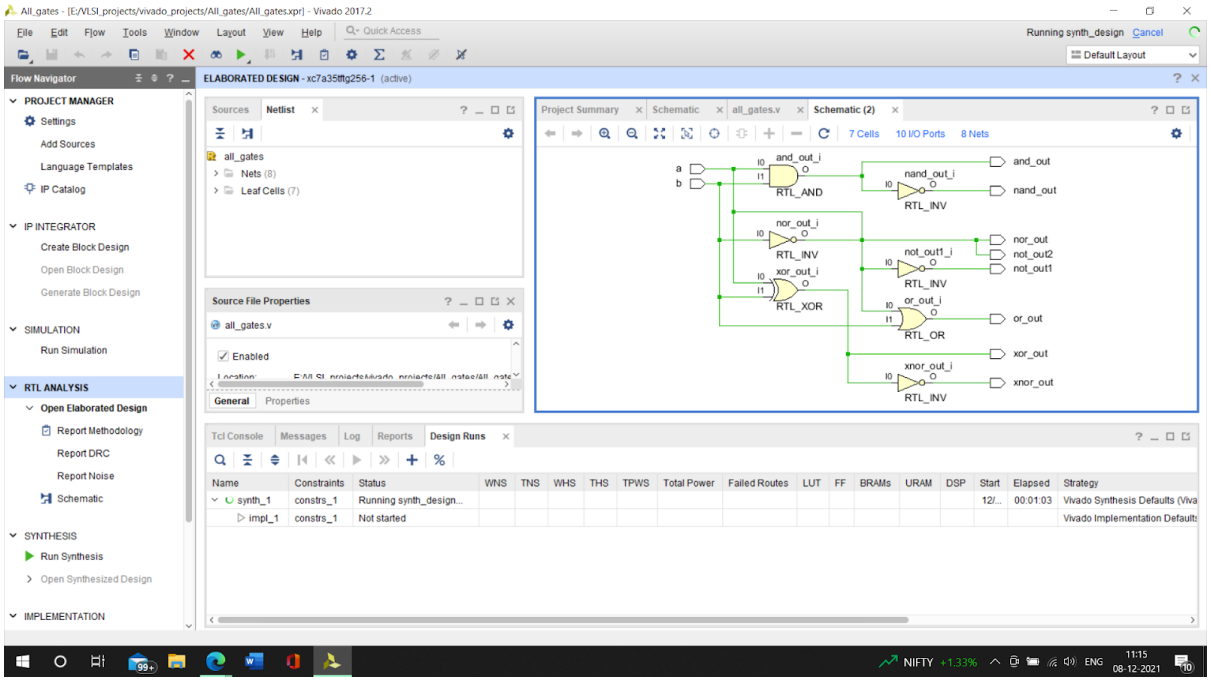
With respect to the values of a and b, and or not xor nor xnot nand output will change its value eventually

Let's come to the **HARDWAR**E part!

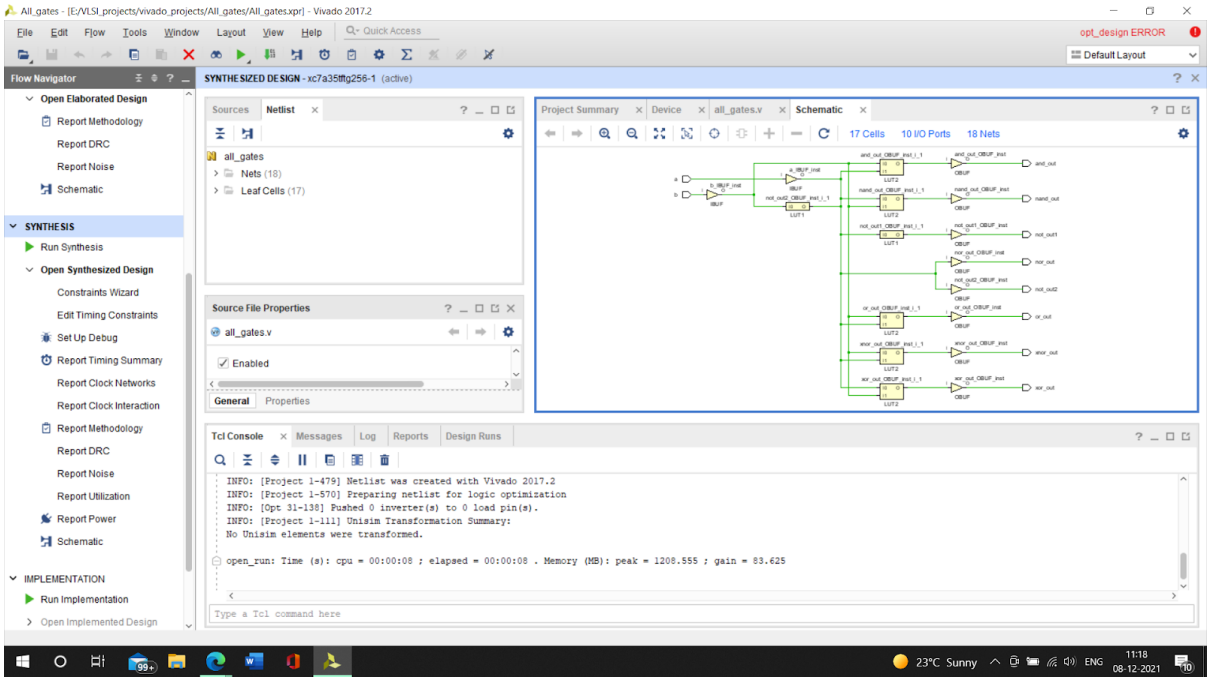
To implement it we first have to synthesis is by run synthesis , we can also elaborate the synthesis, under RTL analysis by choosing Schematic.

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



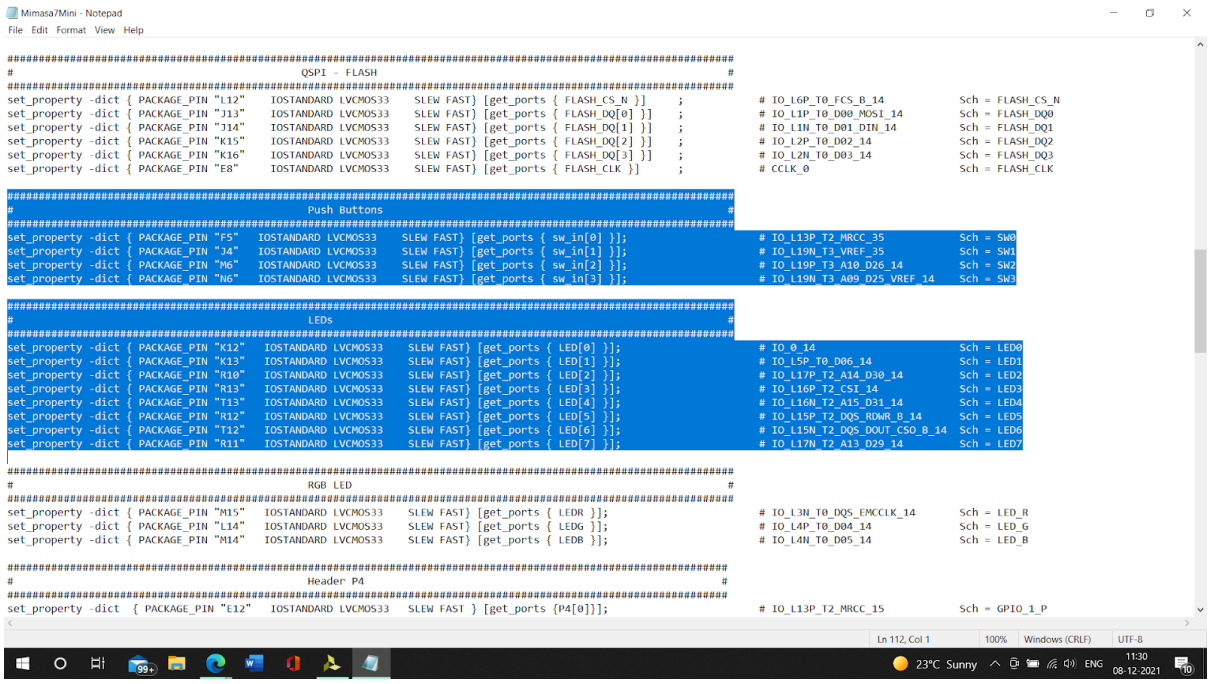
After synthesis is done, we can see the elaborated schematic of the synthesis, so here we might not be understanding with respect to basic gates or it will be having something like buffers, nothing but LUT, look up tables, with single input double input.



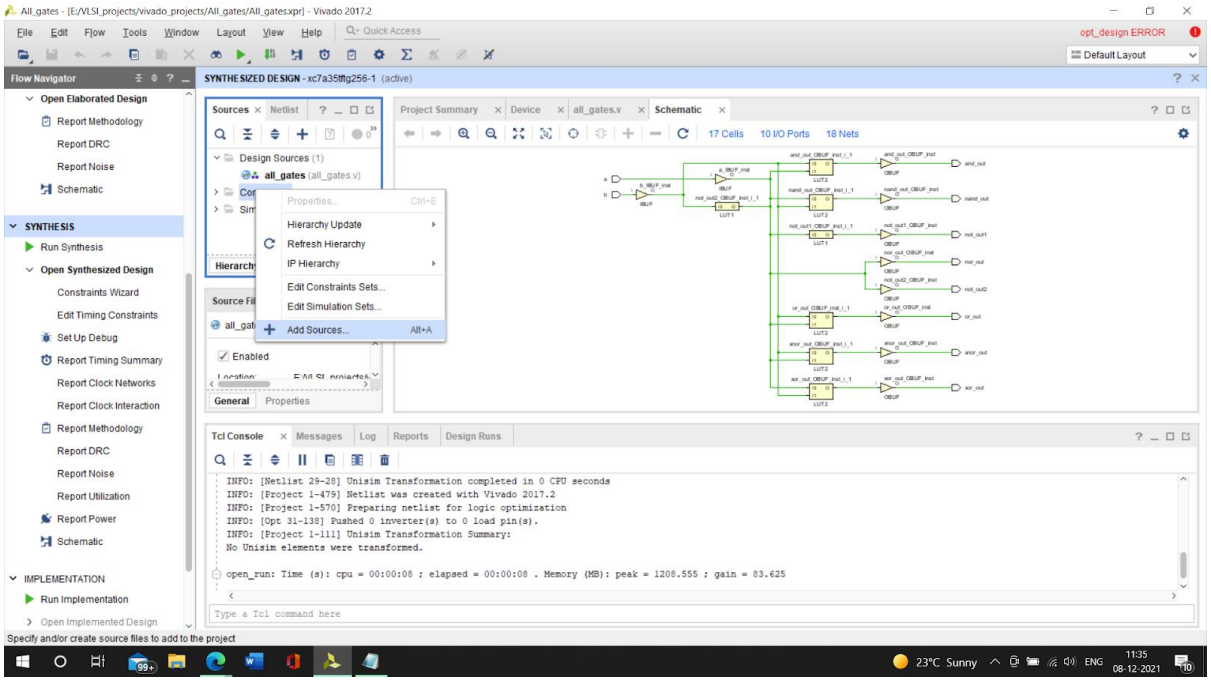
No, we have to bring in the pins which we need to connect, as we know we have to input as in our Verilog code with two input a and b, and when we come to the integrated circuit, we know we are using Mimas A7 mini so all we need to do is

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

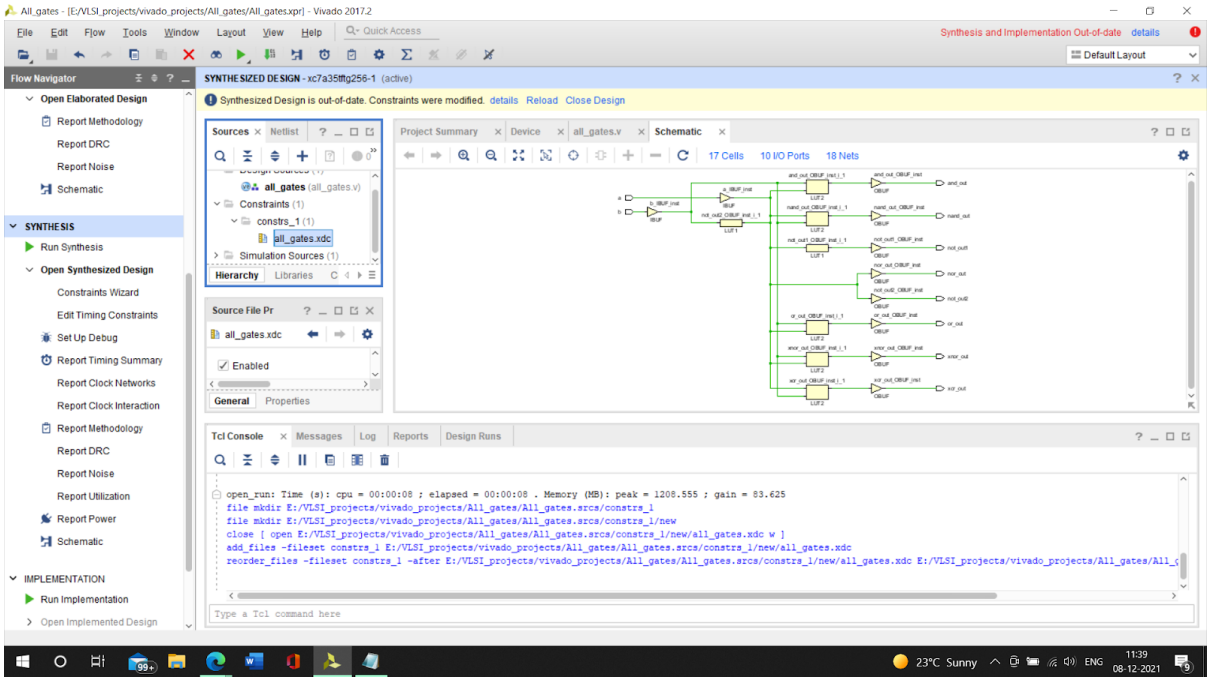
After downloading it open up with the notepad where we can view the XDC file and have the necessary requirements, and when I say requirements, we only need the Push button and the LEDS, and then we have to add these two constrains in VIVADO



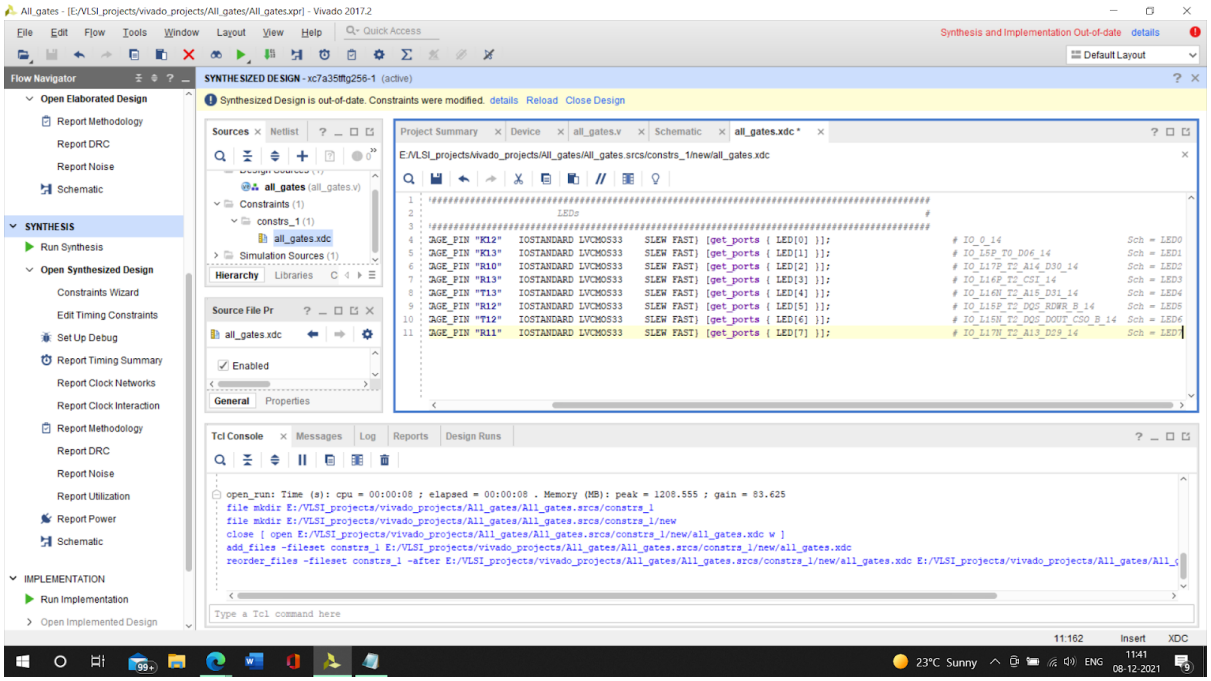
Now let's add the XDC file to vivado by adding sources to constrains, add or create constraints, then create a file name as "all_gates.xdc"



There by it is added!

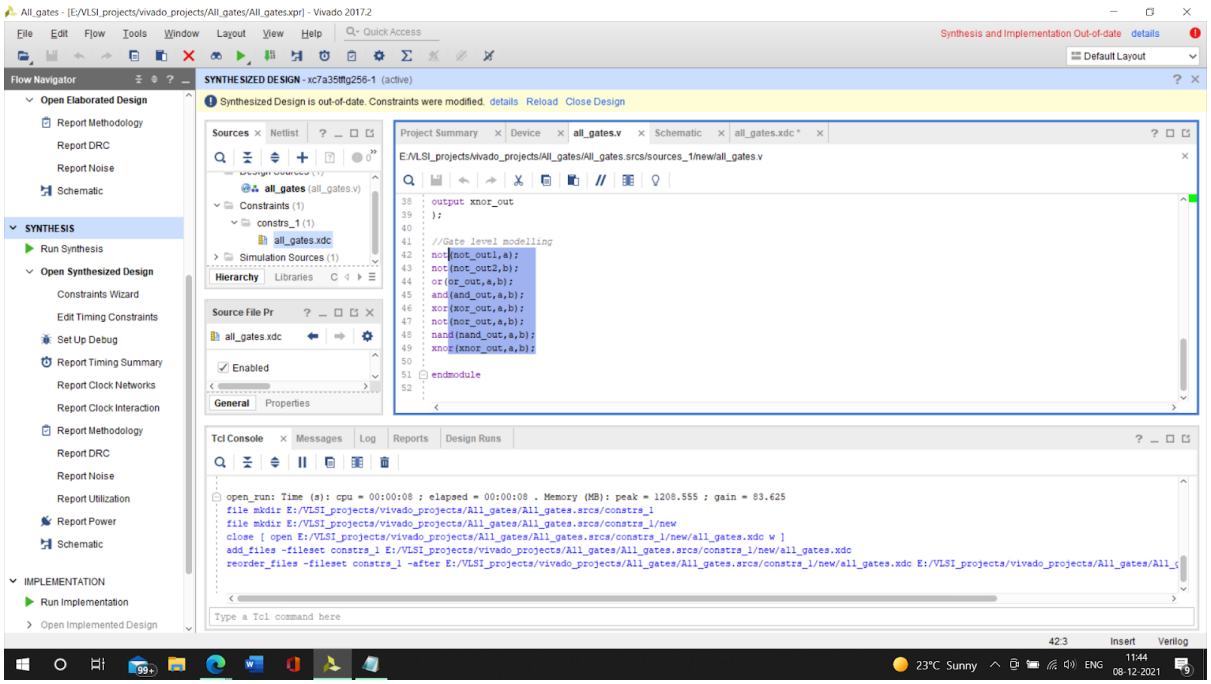


Ok so now it is time to get the xdc file which we download, copy the xdc file for the needed requirements, for us it is Leds and pushbutton and we will paste it here in the constraints file as shown below.

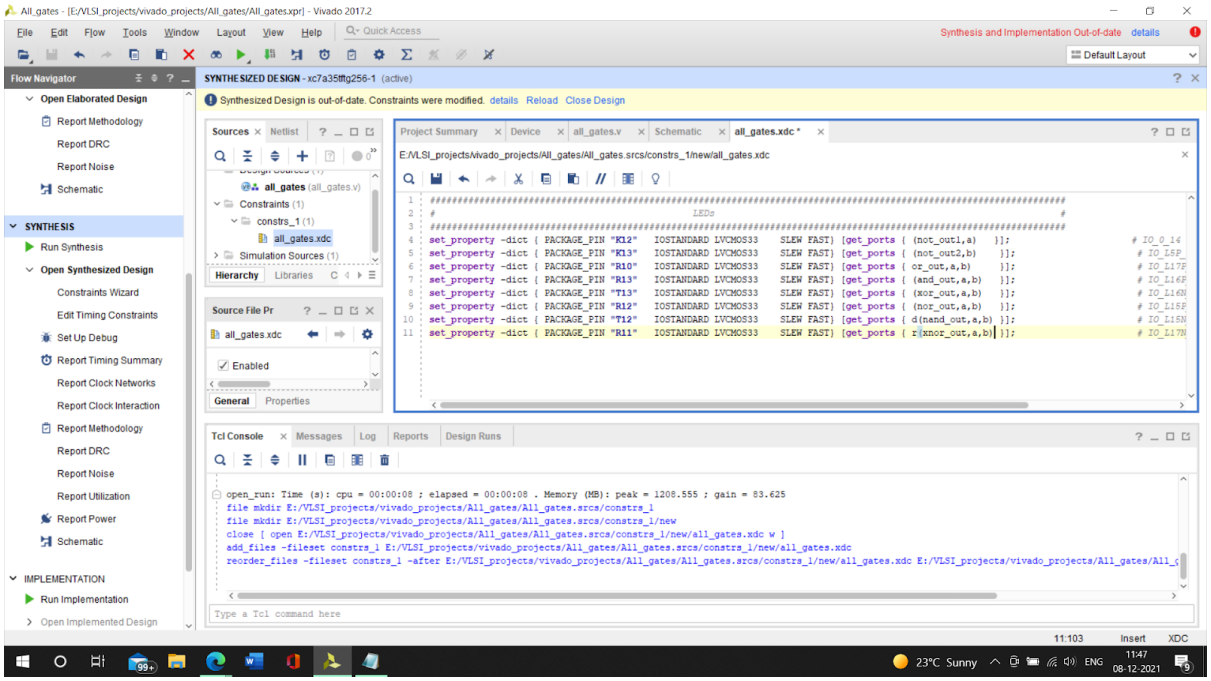


Now we have to change the identifiers in the LED pins so all we will do will take the output from our Verilog code and paste it here as shown below

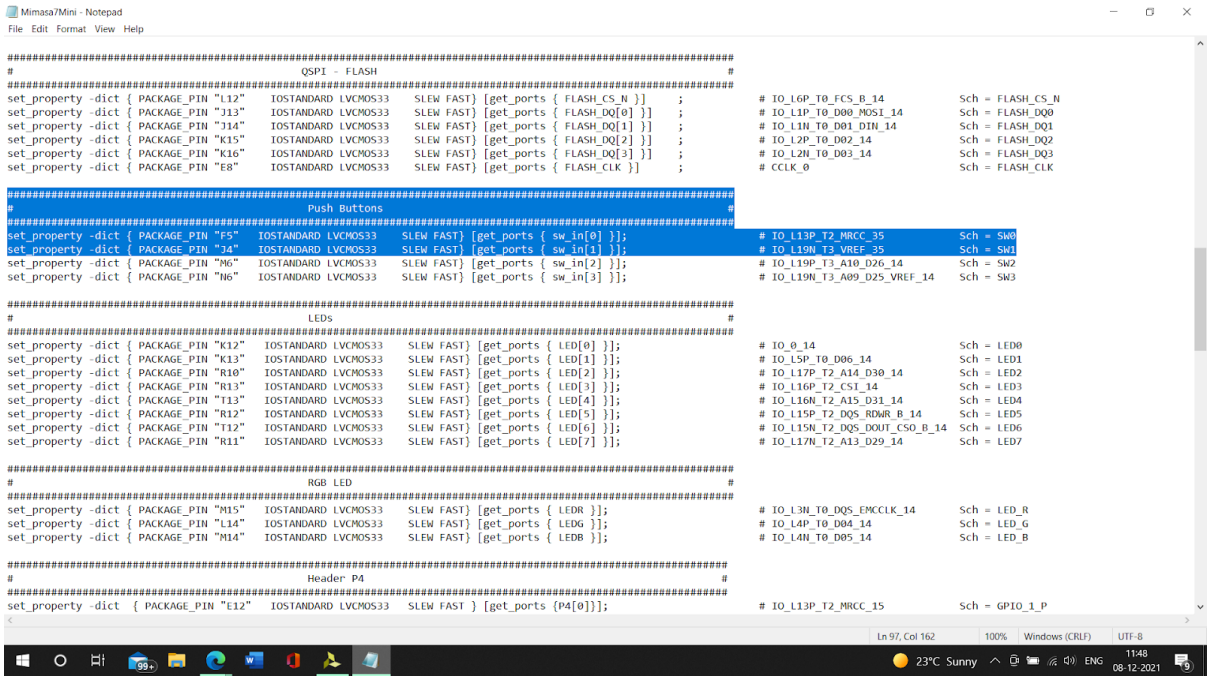
Copy



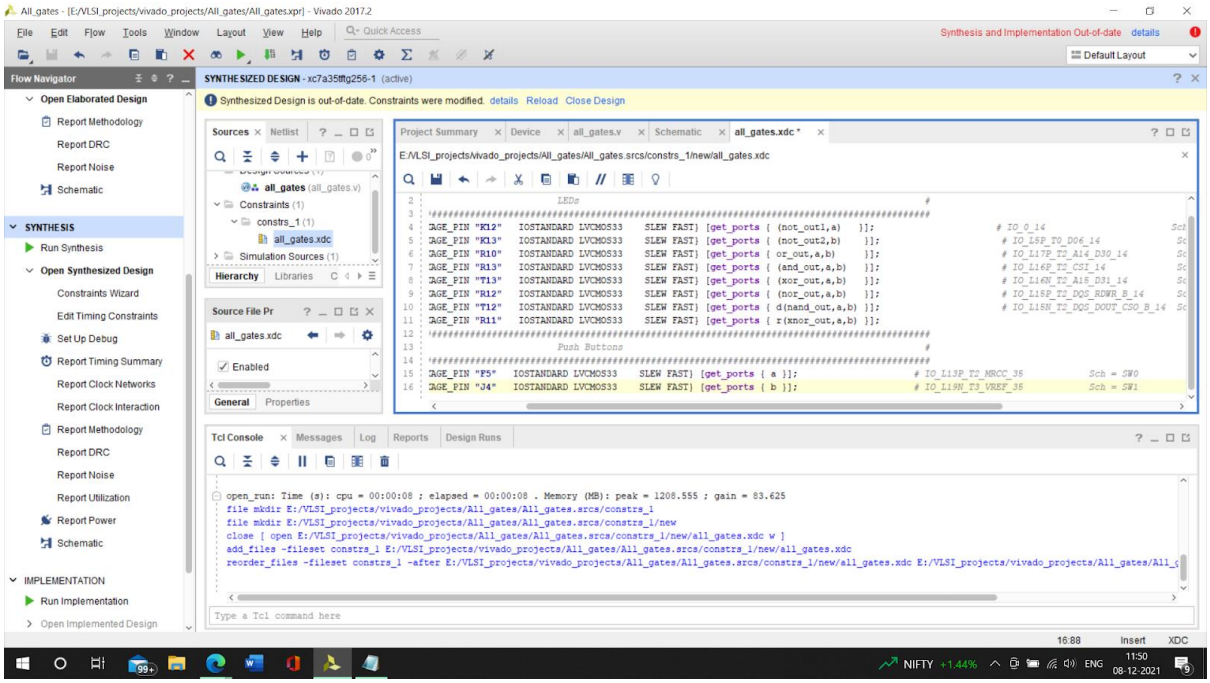
Paste it here



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.

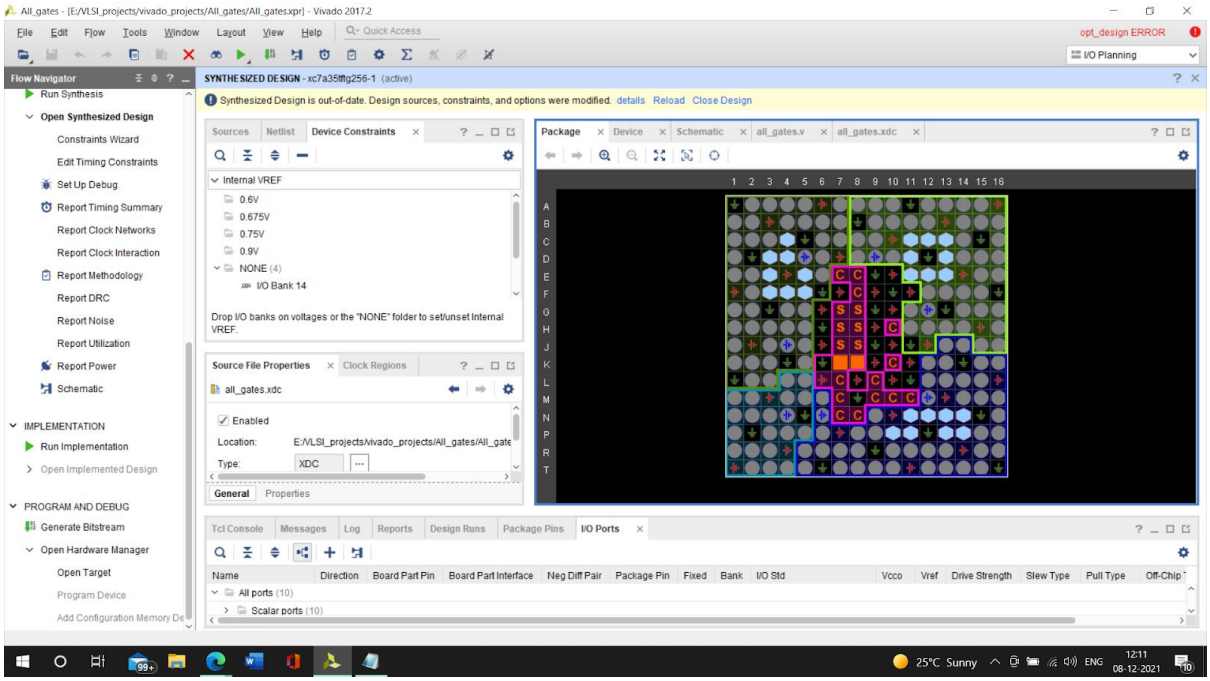


And again, paste it in the added constraints in vivado.

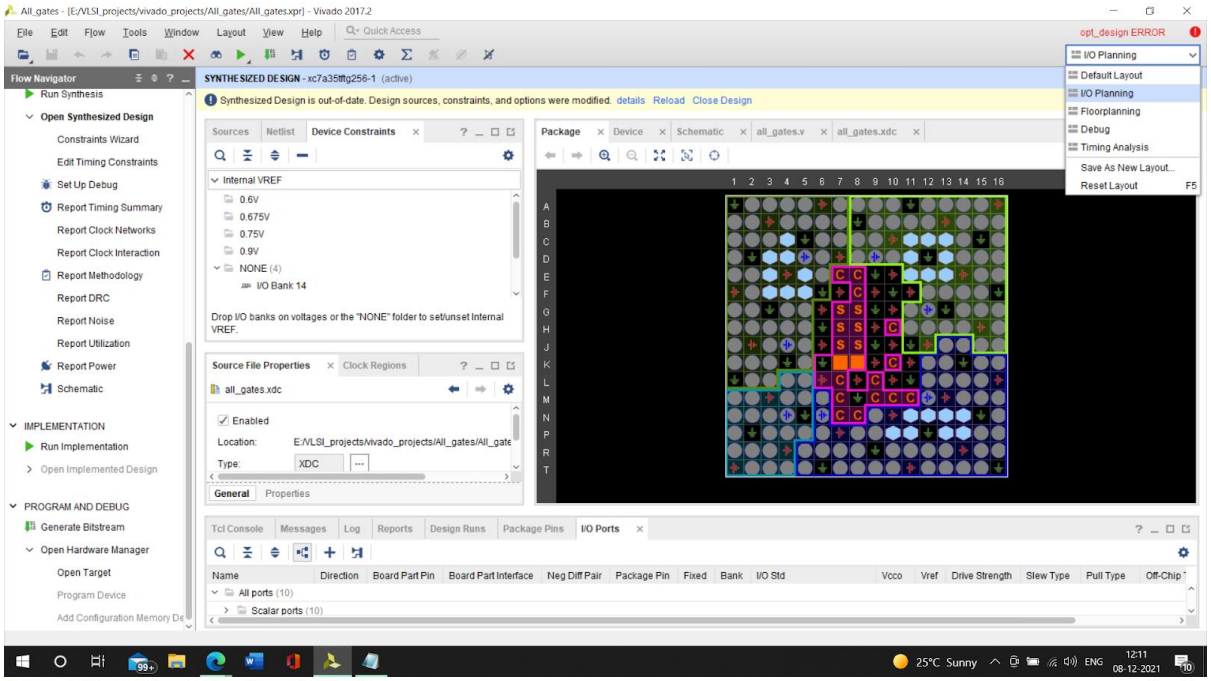


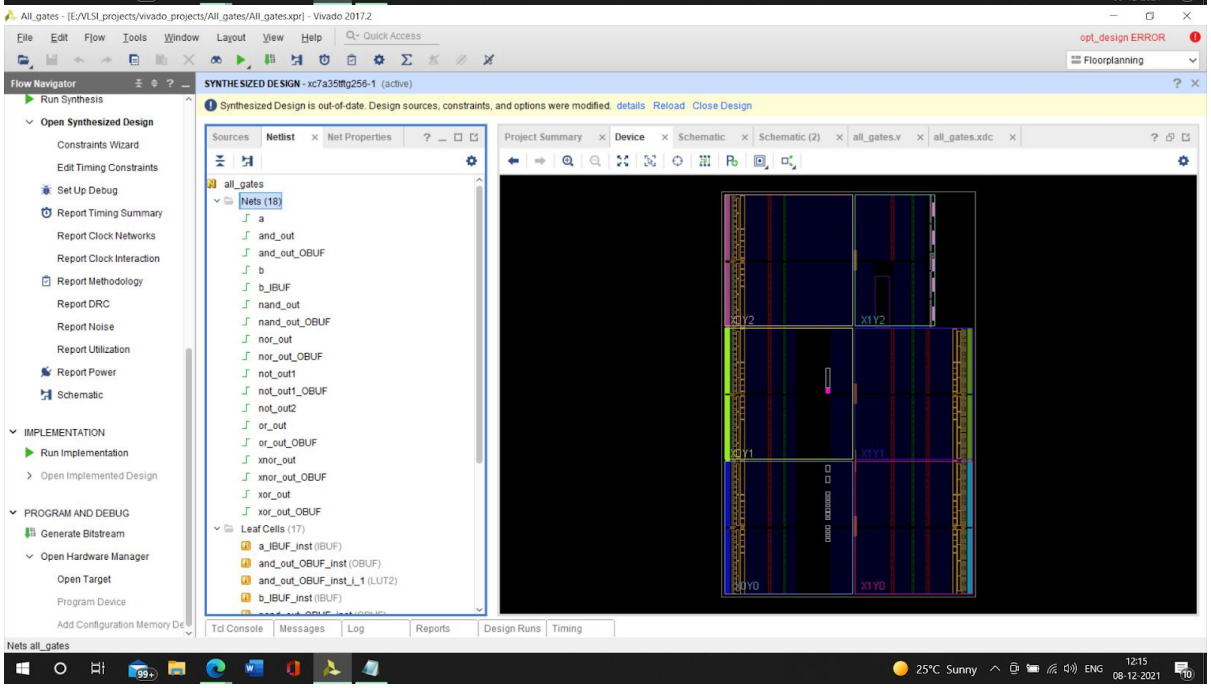
After that we under program and debug we have to choose generate bitstream

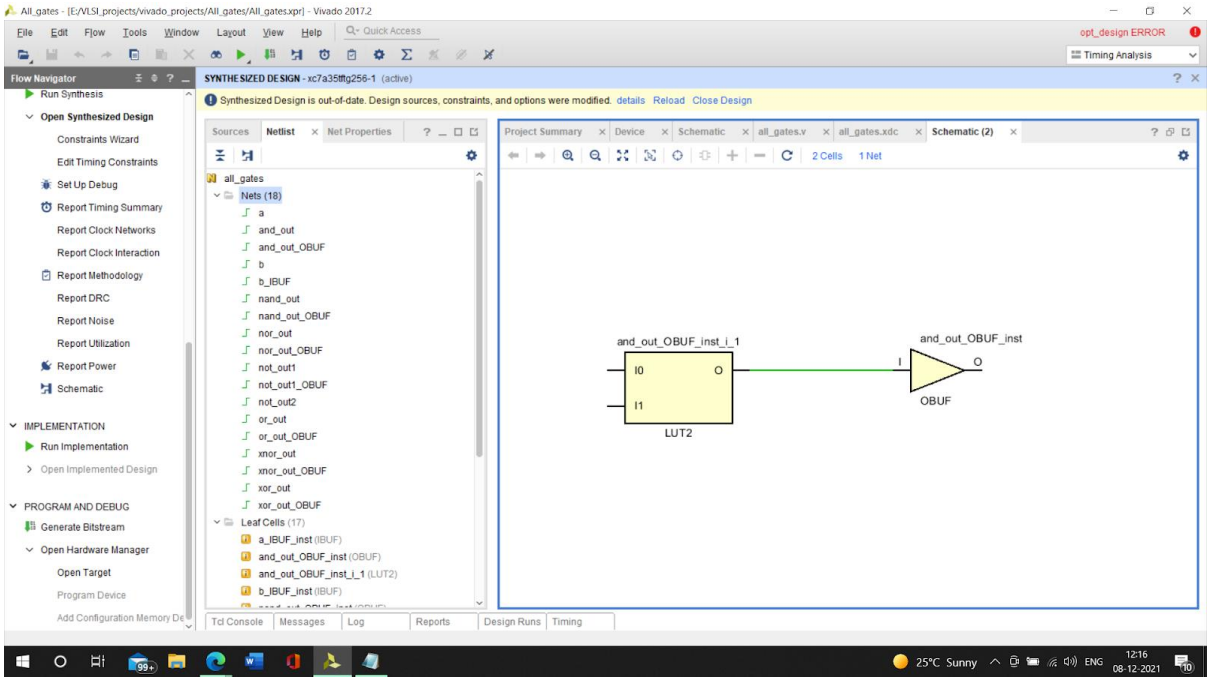
After running, I/O planning is what we can see.



Equivalently we can deal with floor mapping, timing constraints


All_gates - [E:/VLSI_projects/vivado_projects/All_gates/All_gates.xpr] - Vivado 2017.2
File Edit Flow Tools Window Layout View Help
Quick Access
opt_design ERROR
I/O Planning
Default Layout
I/O Planning
Floorplanning
Debug
Timing Analysis
Save As New Layout...
Reset Layout F5
Flow Navigator
Run Synthesis
Open Synthesized Design
Constraints Wizard
Edit Timing Constraints
Set Up Debug
Report Timing Summary
Report Clock Networks
Report Clock Interaction
Report Methodology
Report DRC
Report Noise
Report Utilization
Report Power
Schematic
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
Open Target
Program Device
Add Configuration Memory De
SYNTHESIZED DESIGN - xc7a35tftg256-1 (active)
Synthesized Design is out-of-date. Design sources, constraints, and options were modified. details Reload Close Design
Sources Netlist Device Constraints
Internal VREF
0.6V
0.675V
0.75V
0.9V
NONE (4)
I/O Bank 14
Drop I/O banks on voltages or the "NONE" folder to set/unset Internal VREF.
Source File Properties Clock Regions
all_gates.xdc
Enabled
Location: E:/VLSI_projects/vivado_projects/All_gates/All_gate
Type: XDC
General Properties
Package Device Schematic all_gates.xdc
Tcl Console Messages Log Reports Design Runs Package Pins I/O Ports
Name Direction Board Part Pin Board Part Interface Neg Diff Pair Package Pin Fixed Bank I/O Std Vcco Vref Drive Strength Slew Type Pull Type Off-Chip
All ports (10)
Scalar ports (10)
25°C Sunny
ENG
12:11
08-12-2021


All_gates - [E:/VLSI_projects/vivado_projects/All_gates/All_gates.xpr] - Vivado 2017.2
File Edit Flow Tools Window Layout View Help
Quick Access
opt_design ERROR
Floorplanning
Flow Navigator
Run Synthesis
Open Synthesized Design
Constraints Wizard
Edit Timing Constraints
Set Up Debug
Report Timing Summary
Report Clock Networks
Report Clock Interaction
Report Methodology
Report DRC
Report Noise
Report Utilization
Report Power
Schematic
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
Open Target
Program Device
Add Configuration Memory De
SYNTHESIZED DESIGN - xc7a35tftg256-1 (active)
Synthesized Design is out-of-date. Design sources, constraints, and options were modified. details Reload Close Design
Sources Netlist Net Properties
all_gates
Nets (18)
a
and_out
and_out_OBUF
b
b_IBUF
nand_out
nand_out_OBUF
nor_out
nor_out_OBUF
not_out1
not_out1_OBUF
not_out2
or_out
or_out_OBUF
xnor_out
xnor_out_OBUF
xor_out
xor_out_OBUF
Leaf Cells (17)
a_IBUF_inst (IBUF)
and_out_OBUF_inst (OBUF)
and_out_OBUF_inst_i_1 (LUT2)
b_IBUF_inst (IBUF)
Project Summary Device Schematic Schematic (2) all_gates.xdc
X0Y2
X1Y2
X0Y1
X1Y1
X0Y0
X1Y0
Tcl Console Messages Log Reports Design Runs Timing
Nets all_gates
25°C Sunny
ENG
12:15
08-12-2021



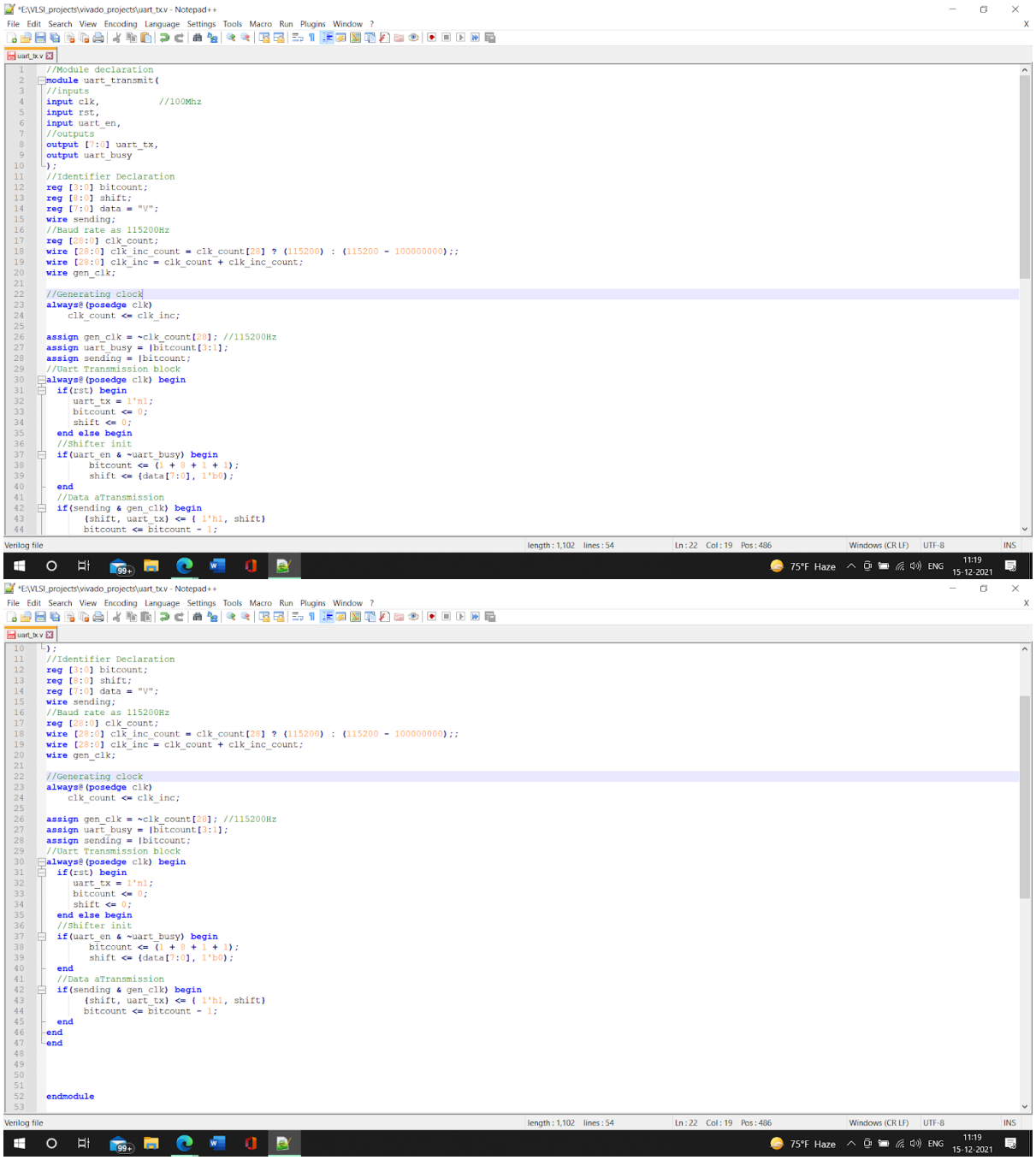
Now if we want to connect it with the real time Mimas mini A7, firstly we have to connect it with our pc/laptop with the FPGA, then through Tenegra application we have to synch vivado, we have to automatically connect to the hardware by vivado and add a local server, with a local host, then it will be 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

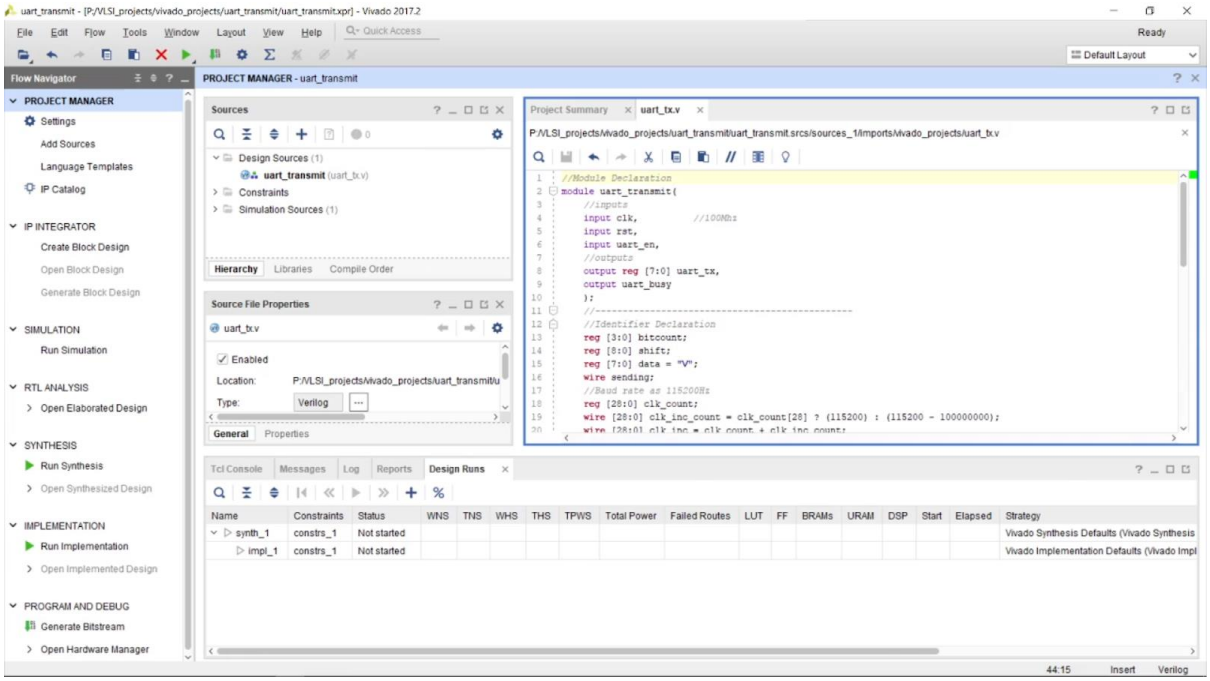
Software to download

- Notepad++

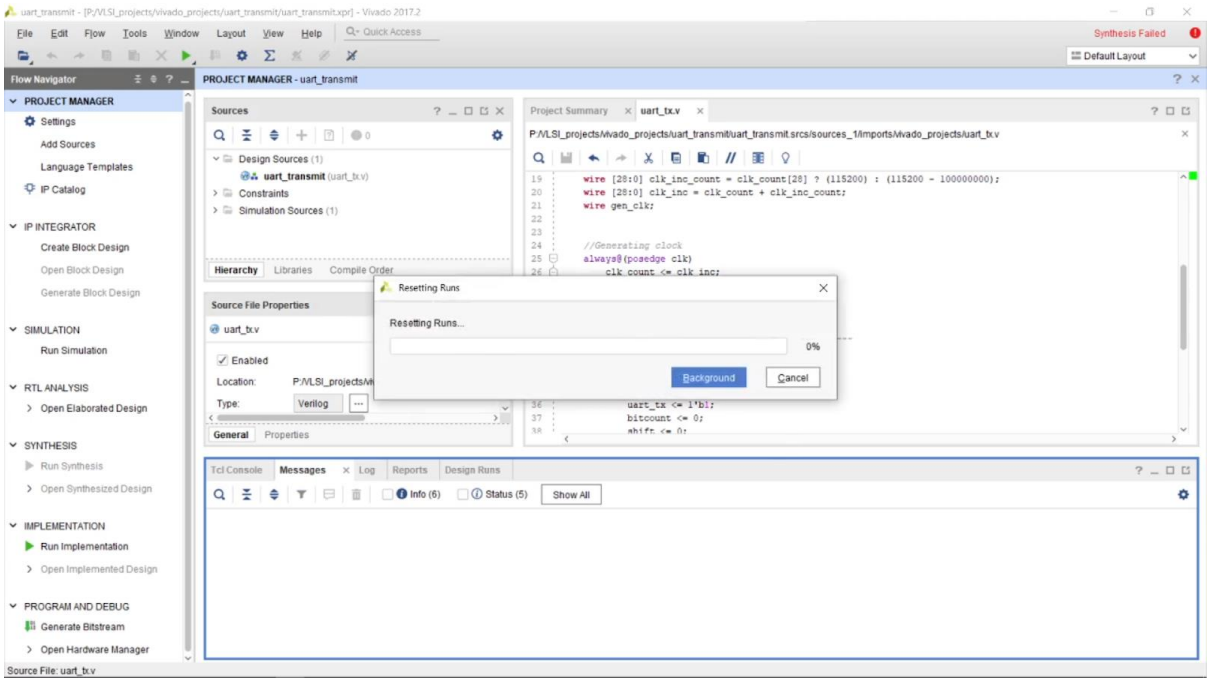
Step 1- Write the verilog program in notepad ++ .



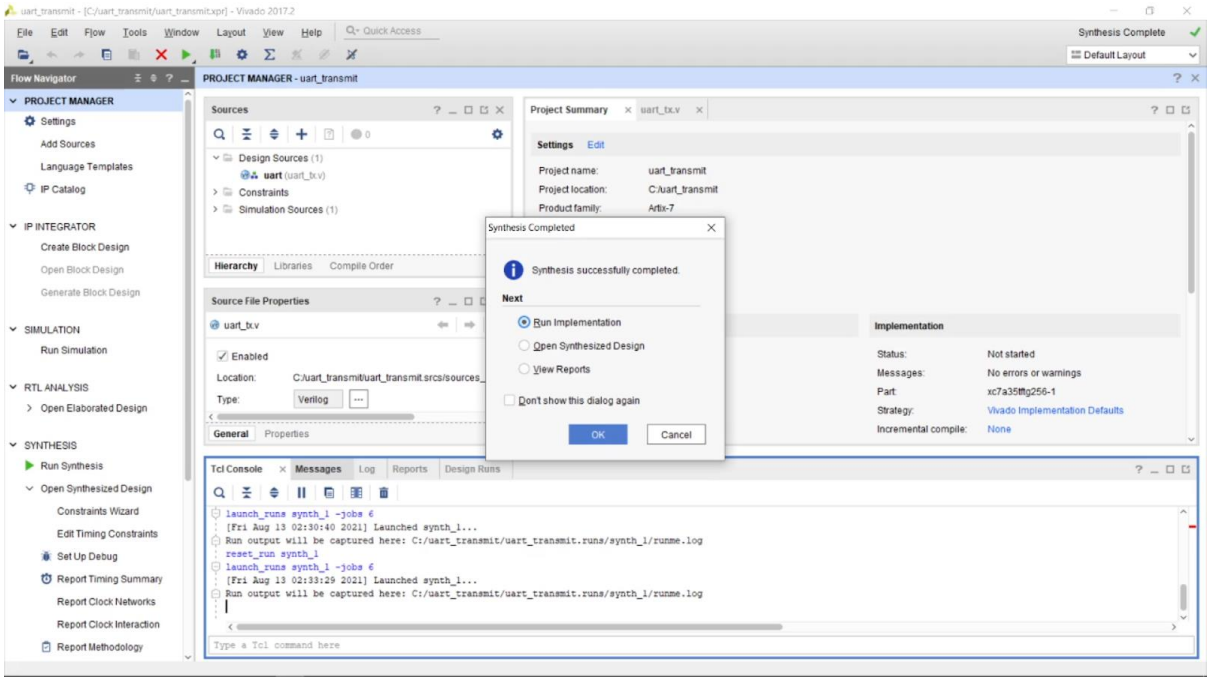
Step 2-Open Vivado software create a new project and add this notepad verilog extension to it



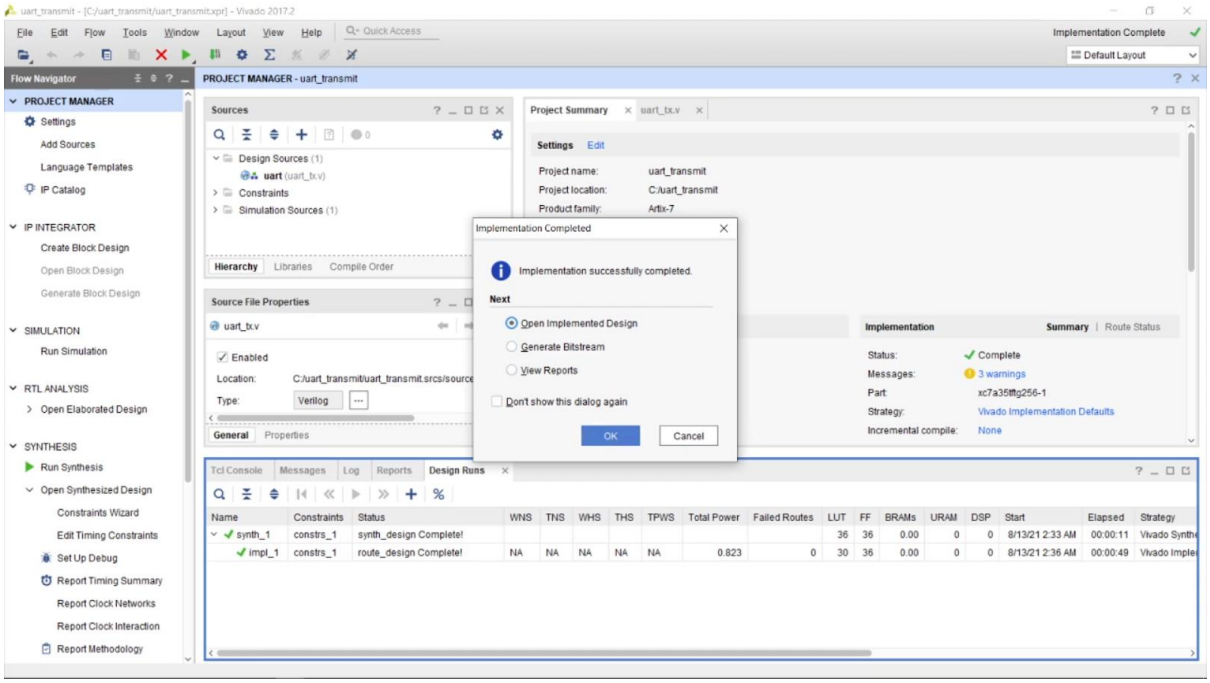
Synthesis the code extracted and compile the errors if there are any.



After synthesis run implementation

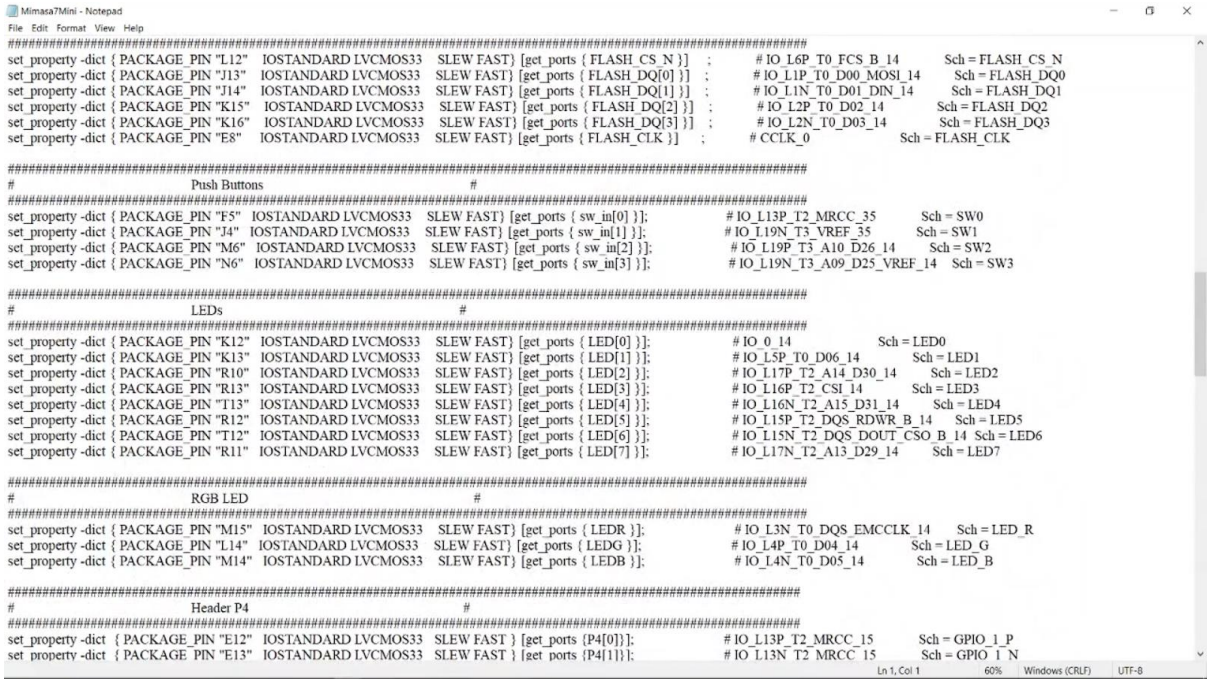


Open synthesis implemented design

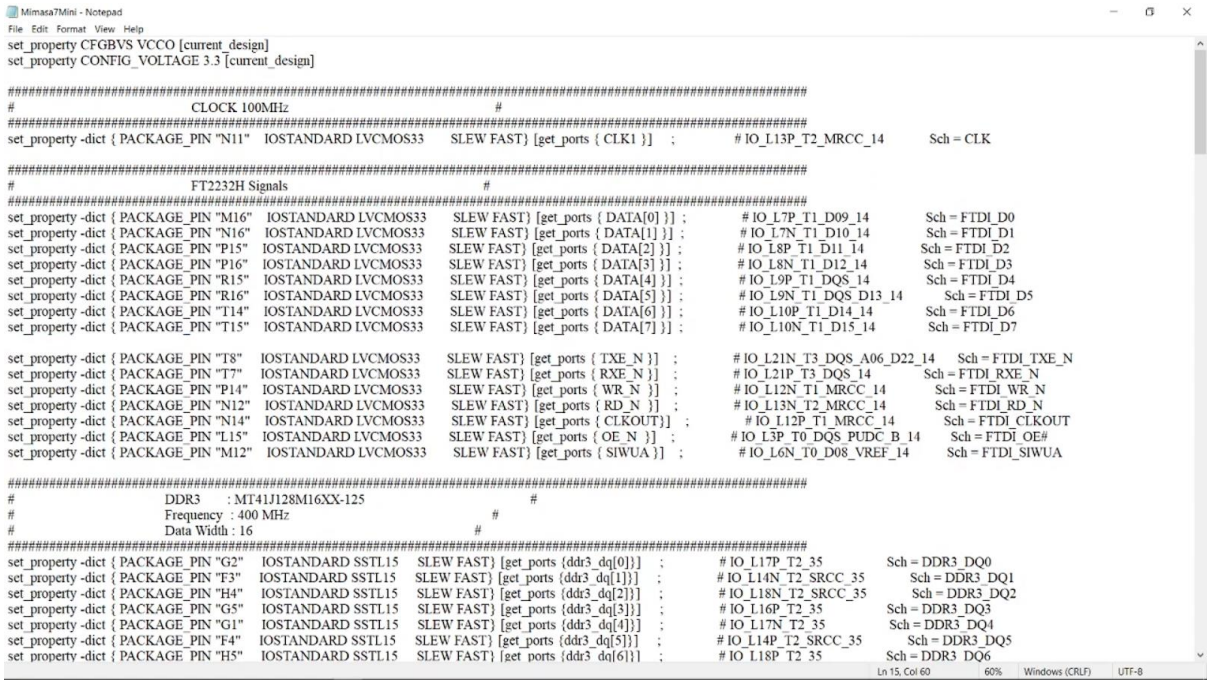


Now one of the most vital part we have to change port design with respect to the XDC file we got for vivado designs from the link [Mimas A7 Mini FPGA Development Board | Numato Lab]

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



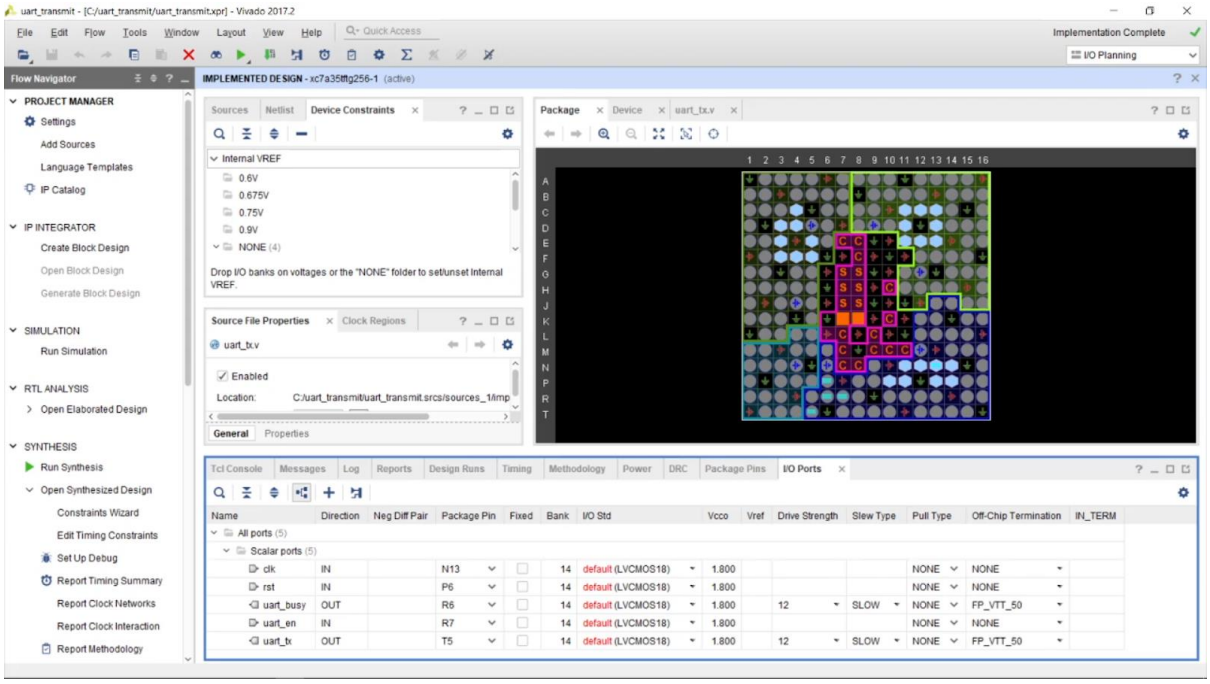
```
Mimasa7Mini - Notepad
File  Edit  Format  View  Help
##############################################################################################################
set_property -dict { PACKAGE_PIN "L12"   IOSTANDARD LVCMOS33   SLEW FAST} [get_ports { FLASH_CS_N }]    ;          # IO_L6P_T0_FCS_B_14          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_T0_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_T0_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_L13P_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_0_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_T0_DQS_EMCCLK_14     Sch = LED_R
set_property -dict { PACKAGE_PIN "L14"   IOSTANDARD LVCMOS33   SLEW FAST} [get_ports { LEDG }];              # IO_L4P_T0_D04_14            Sch = LED_G
set_property -dict { PACKAGE_PIN "M14"   IOSTANDARD LVCMOS33   SLEW FAST} [get_ports { LEDB }];              # IO_L4N_T0_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
```



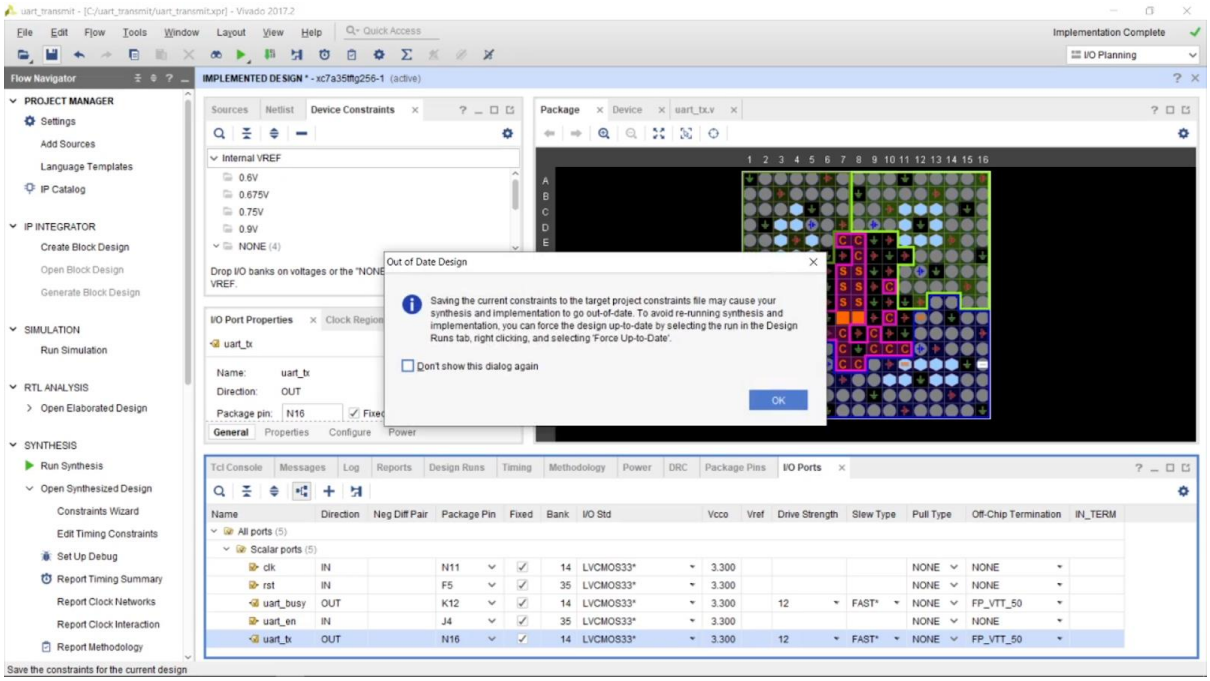
```
Mimasa7Mini - Notepad
File  Edit  Format  View  Help
set_property CFGBVS VCCO [current_design]
set_property CONFIG_VOLTAGE 3.3 [current_design]

##############################################################################################################
#                                  CLOCK 100MHz                                                              #
##############################################################################################################
set_property -dict { PACKAGE_PIN "N11"   IOSTANDARD LVCMOS33     SLEW FAST} [get_ports { CLK1 }]    ;          # IO_L13P_T2_MRCC_14          Sch = CLK

##############################################################################################################
#                                  FT2232H 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_T0_DQS_PUDC_B_14     Sch = FTDI_OE#
set_property -dict { PACKAGE_PIN "M12"   IOSTANDARD LVCMOS33     SLEW FAST} [get_ports { SIWUA }]   ;          # IO_L6N_T0_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 {ddr3_dq[0]}]    ;      # IO_L17P_T2_35               Sch = DDR3_DQ0
set_property -dict { PACKAGE_PIN "F3"    IOSTANDARD SSTL15    SLEW FAST} [get_ports {ddr3_dq[1]}]    ;      # IO_L14N_T2_SRCC_35          Sch = DDR3_DQ1
set_property -dict { PACKAGE_PIN "H4"    IOSTANDARD SSTL15    SLEW FAST} [get_ports {ddr3_dq[2]}]    ;      # IO_L18N_T2_SRCC_35          Sch = DDR3_DQ2
set_property -dict { PACKAGE_PIN "G5"    IOSTANDARD SSTL15    SLEW FAST} [get_ports {ddr3_dq[3]}]    ;      # IO_L16P_T2_35               Sch = DDR3_DQ3
set_property -dict { PACKAGE_PIN "G1"    IOSTANDARD SSTL15    SLEW FAST} [get_ports {ddr3_dq[4]}]    ;      # IO_L17N_T2_35               Sch = DDR3_DQ4
set_property -dict { PACKAGE_PIN "F4"    IOSTANDARD SSTL15    SLEW FAST} [get_ports {ddr3_dq[5]}]    ;      # IO_L14P_T2_SRCC_35          Sch = DDR3_DQ5
set_property -dict { PACKAGE_PIN "H5"    IOSTANDARD SSTL15    SLEW FAST} [get_ports {ddr3_dq[6]}]    ;      # IO_L18P_T2_35               Sch = DDR3_DQ6
                                                                                                                                  Ln 15, Col 60   60%   Windows (CRLF)   UTF-8
```

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

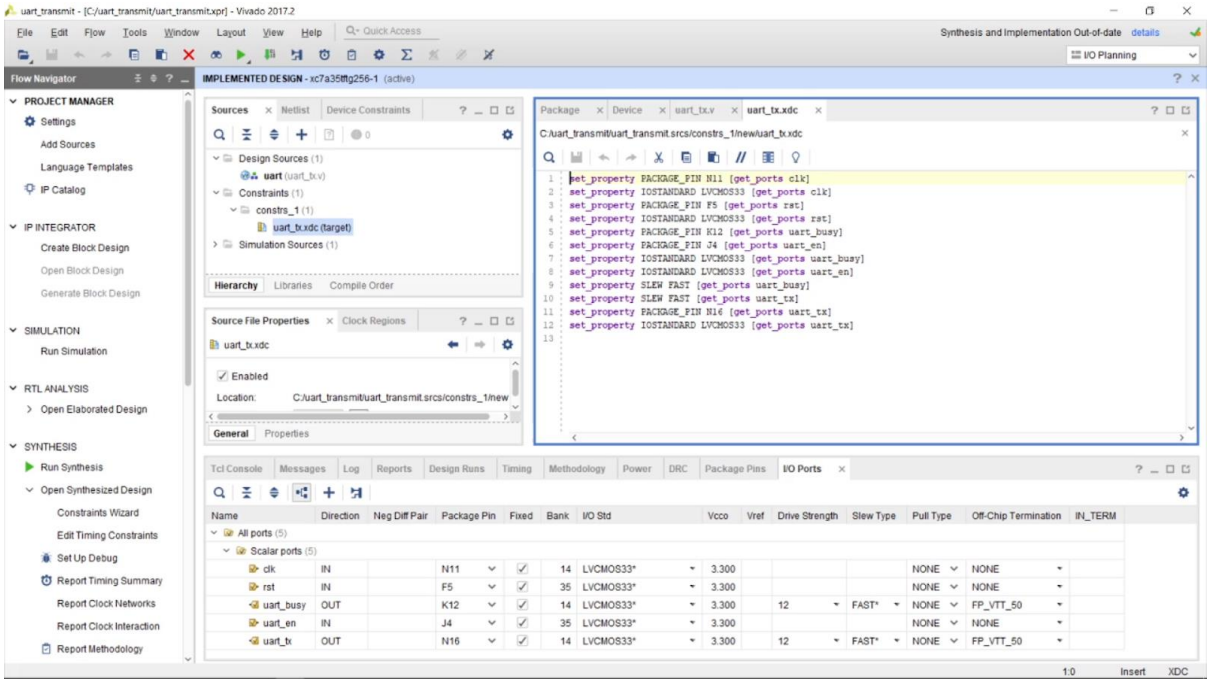
We will use low voltage CMOS, LVCMOS33



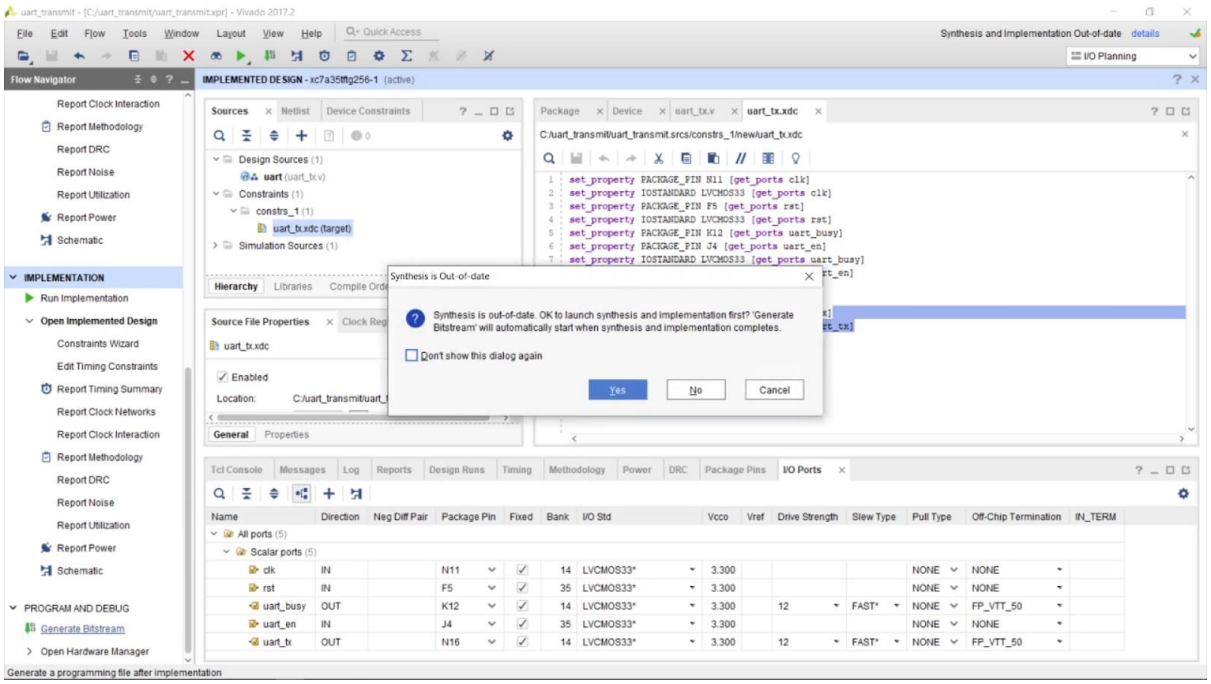
After the necessary changes save it and name the file as “uart_tx”



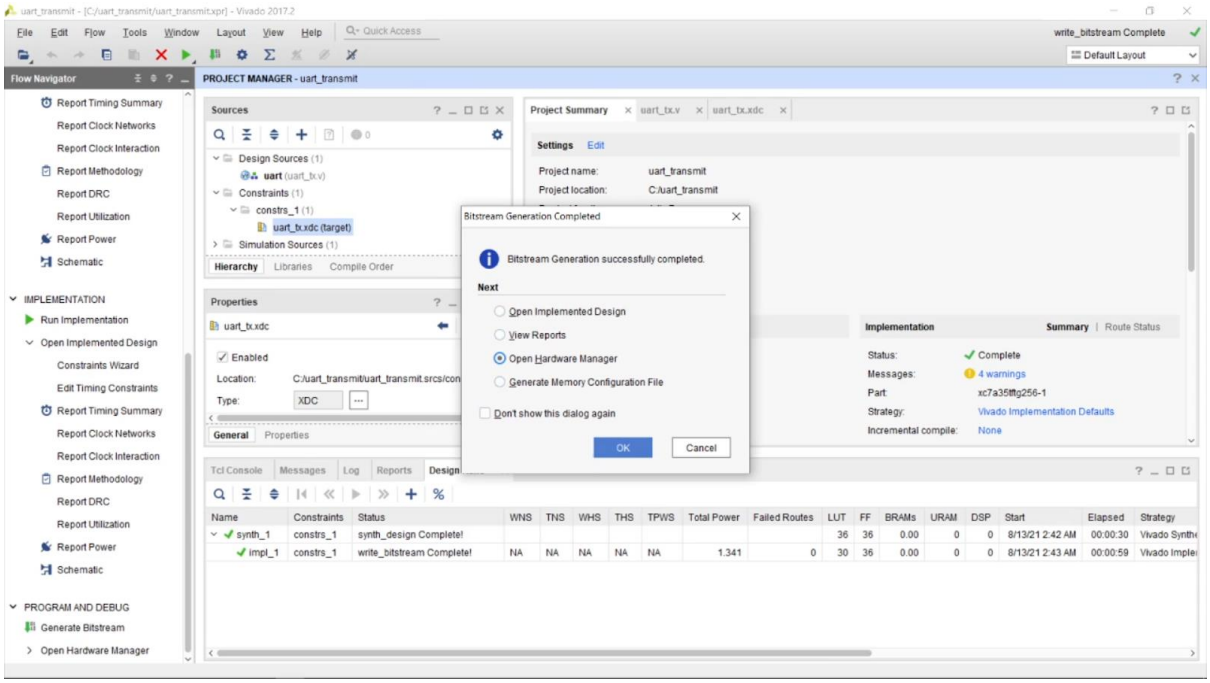
After that we can see in the source file under constrains and the I/O standards are given



Generate bit stream after that, resulting in synthesis and implementation



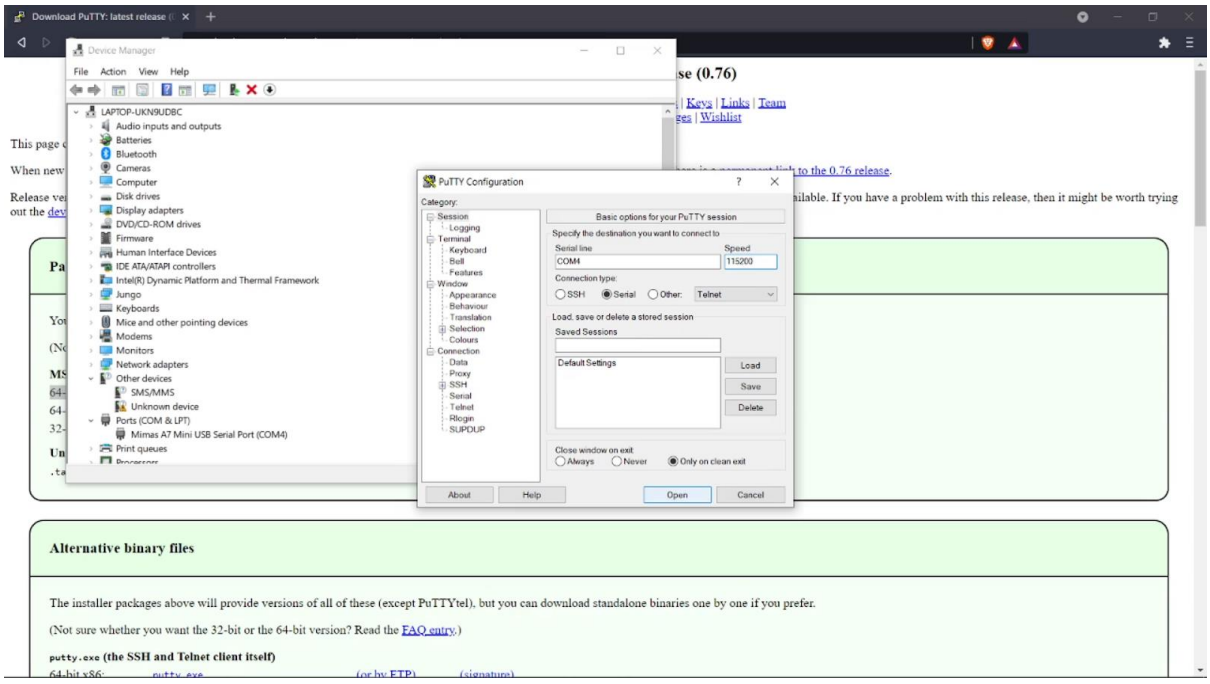
After the generation of bit stream open hardware manager



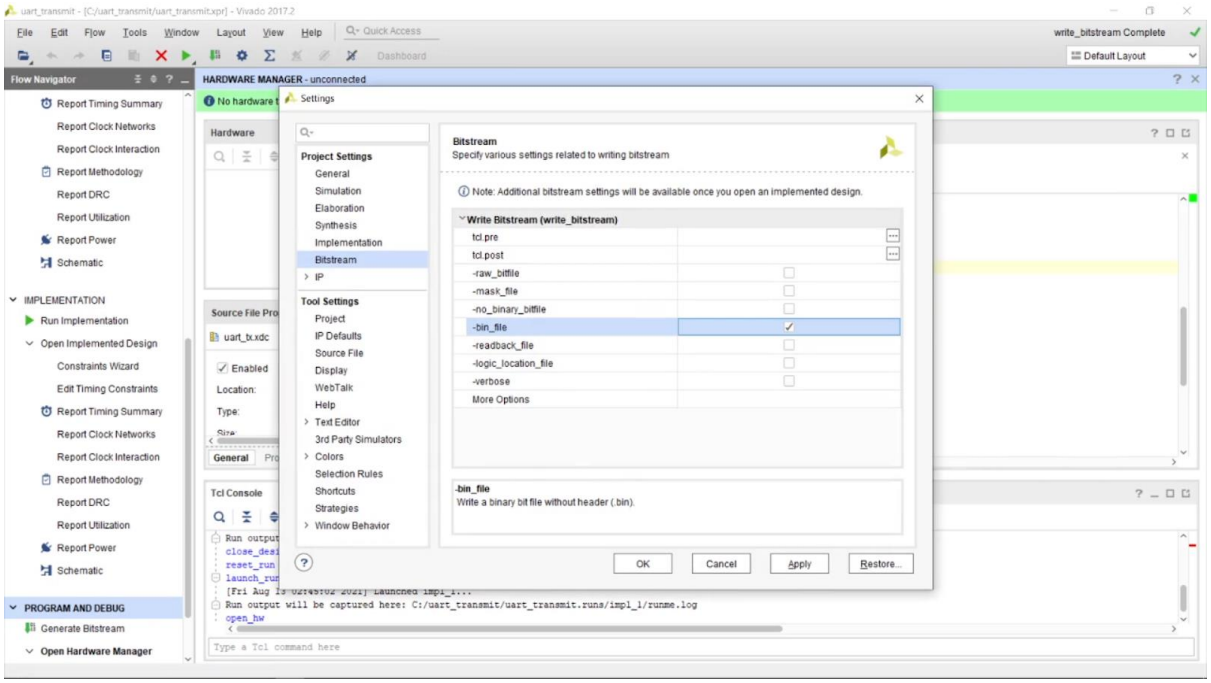
Step 3- Download a serial monitor , we will be using a light weight serial monitor software

Download PuTTY - a free SSH and telnet client for Windows link to download

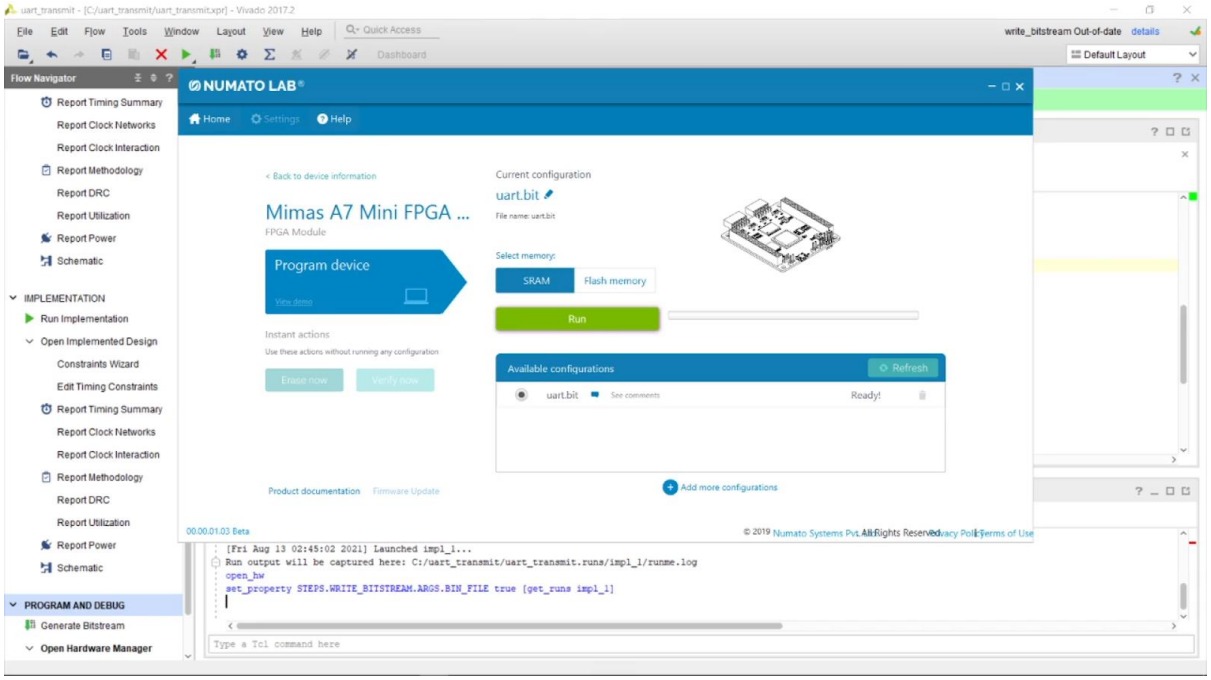
After that launch Putty with a a serial line as "COM 4" shown in device manager used in the following case , and the speed we know set as 15200, and then open the termnal.



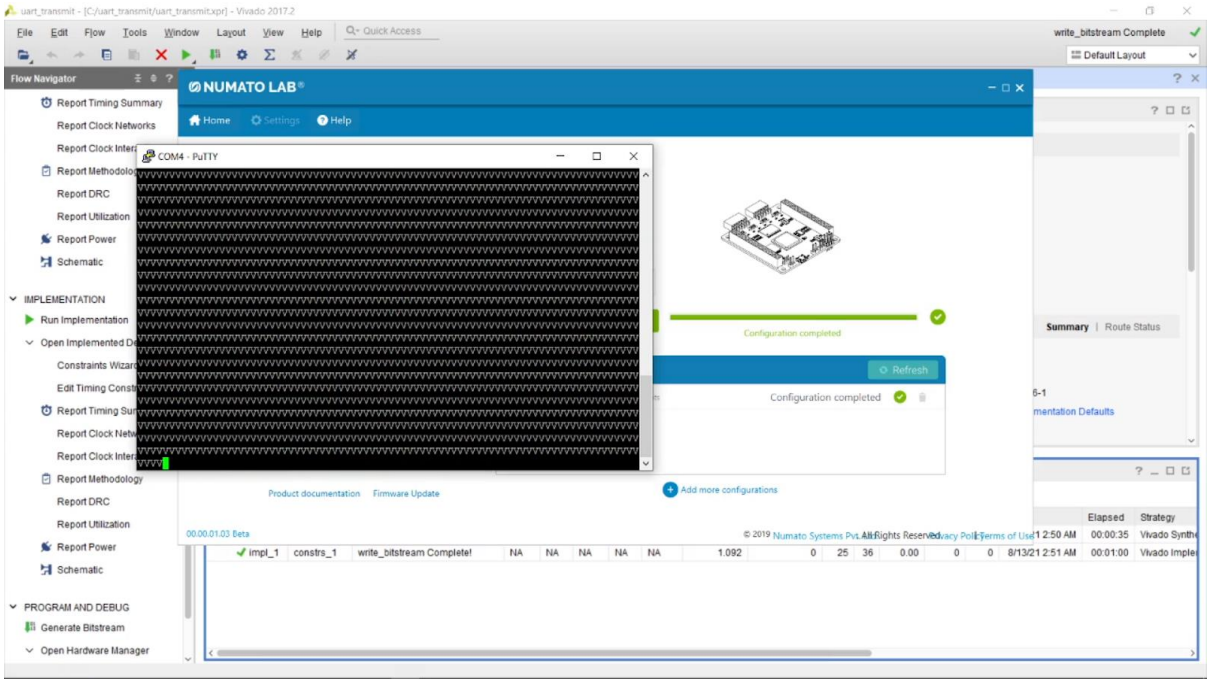
Step 4- Open Tenegra application and synch vivado bitstream file , you can go to setting and create one .



After chosing the file select SRAM and run it

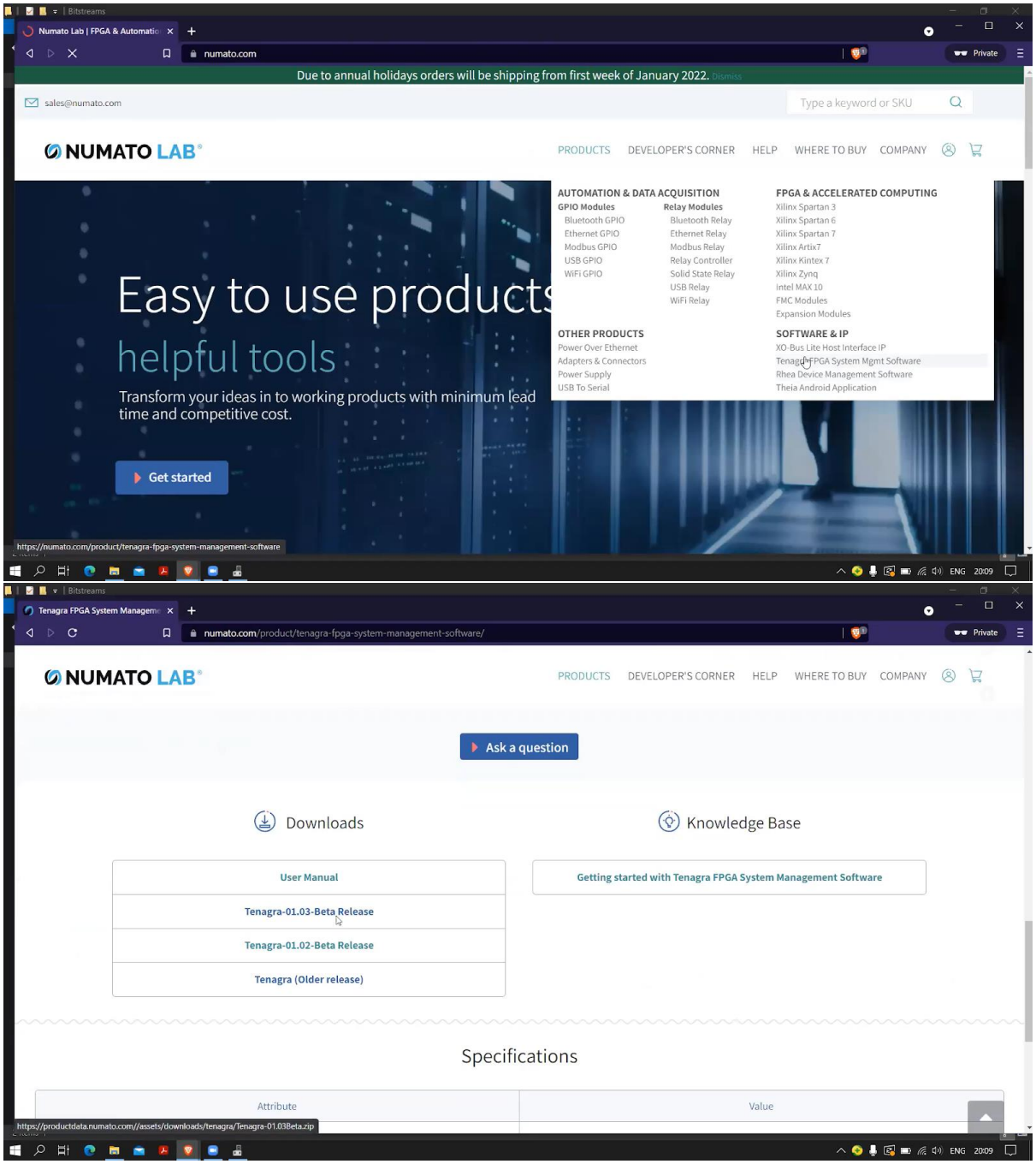


As we know that with push button is enabled with the input of letter we have created so in our case push button will work that way.Every time we press the push button the letter V gets automatically printed in its own.

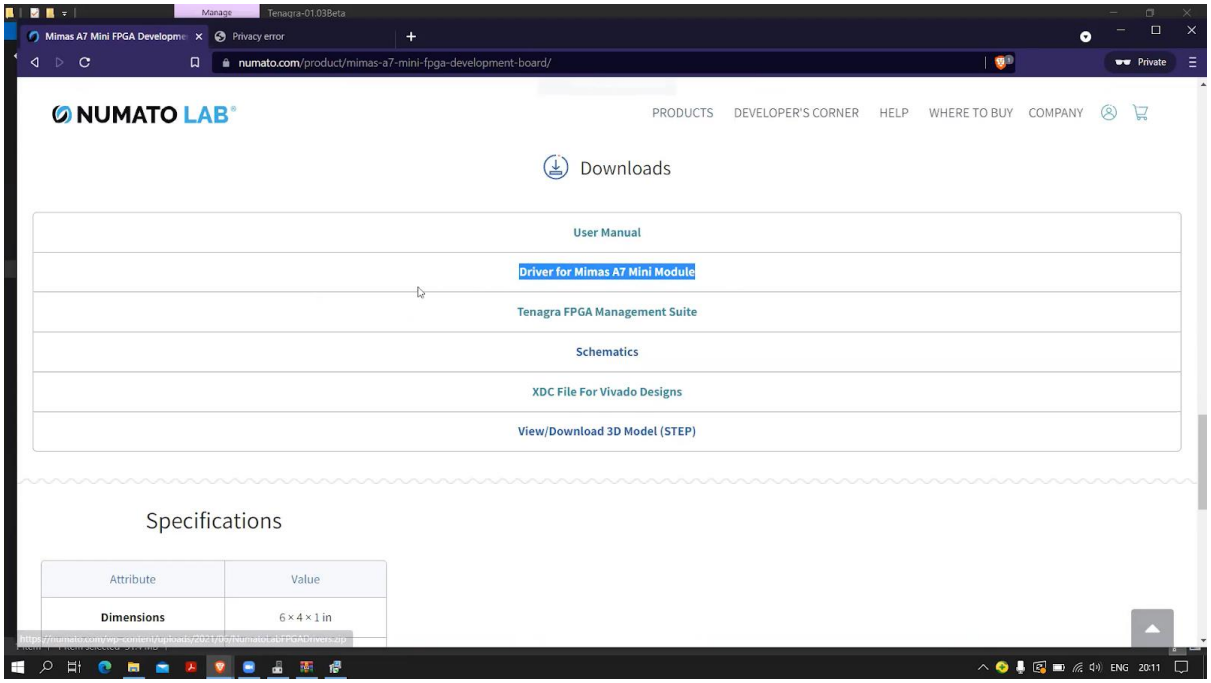


3. FSM Designs: Mealy & Moore Machines and Up Down Counter

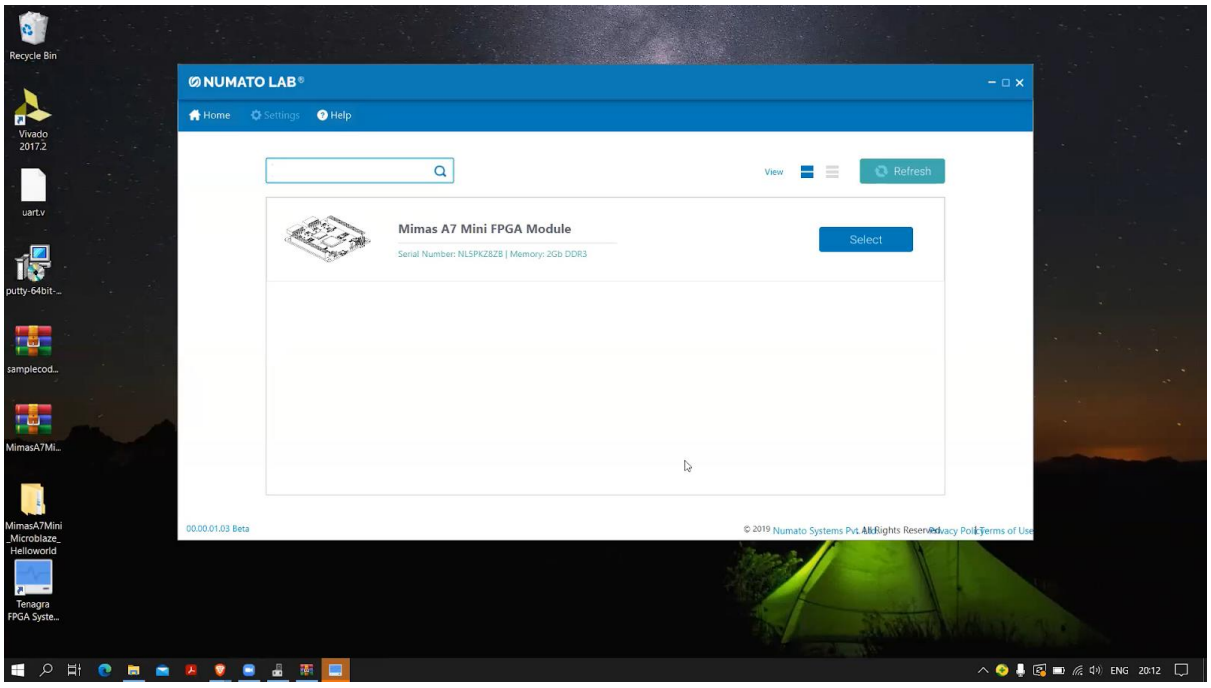
Step 1- Connect the real time FPGA Mimas A7 mini board, open the device manager to set its configuration. Now we have to get tenegra a software which acts like a bridge between the FPGA board and the pc, where the program will be uploaded. You can get tenegra under Numato Lab



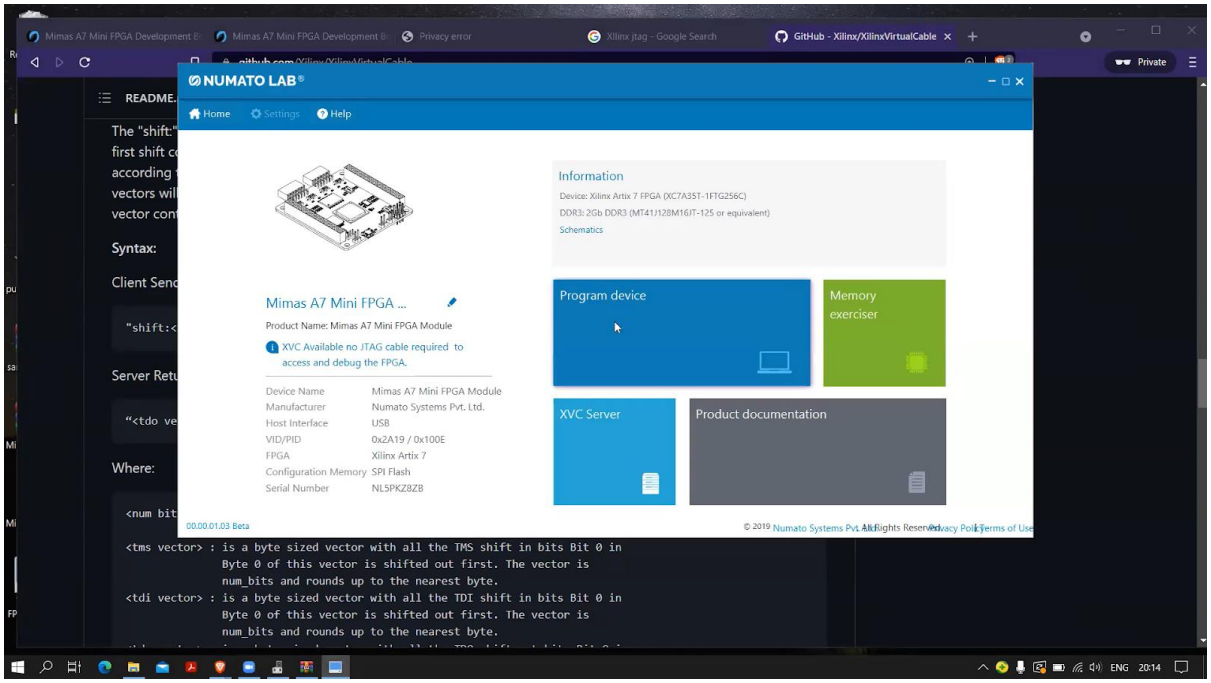
After this we have to download few drivers' fir Mimas A7 Mini Module.



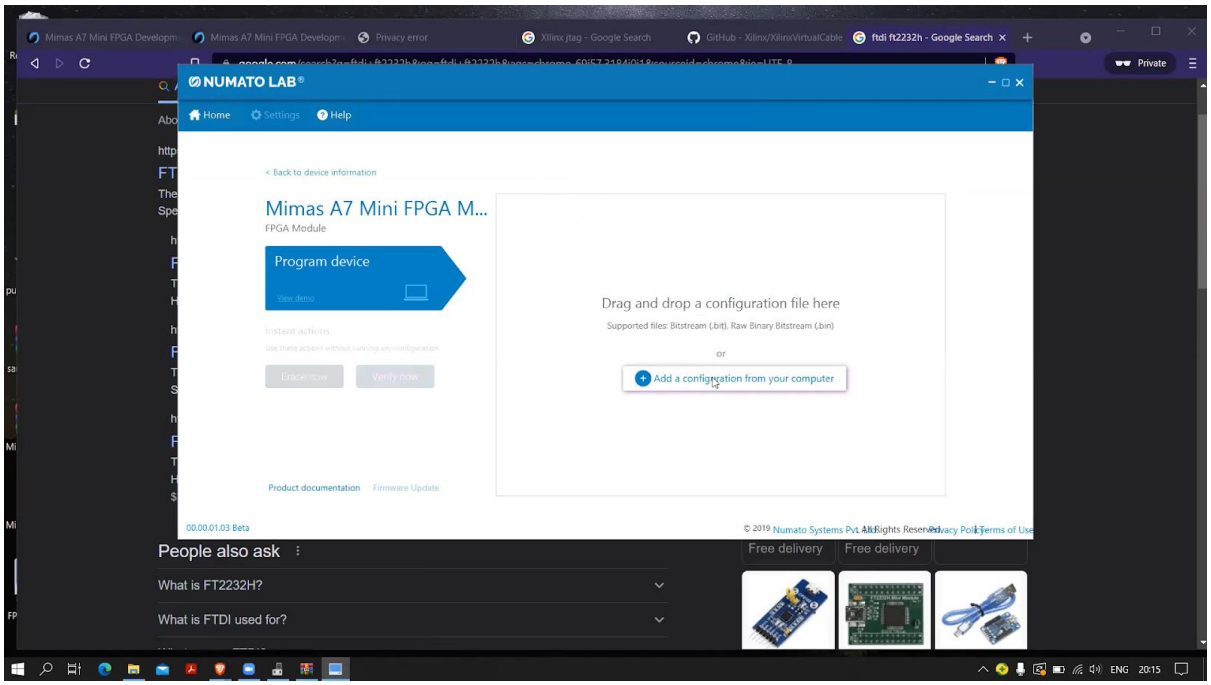
Launch tenegra where we will be viewing two ways or methods to put the program in our FPGA .



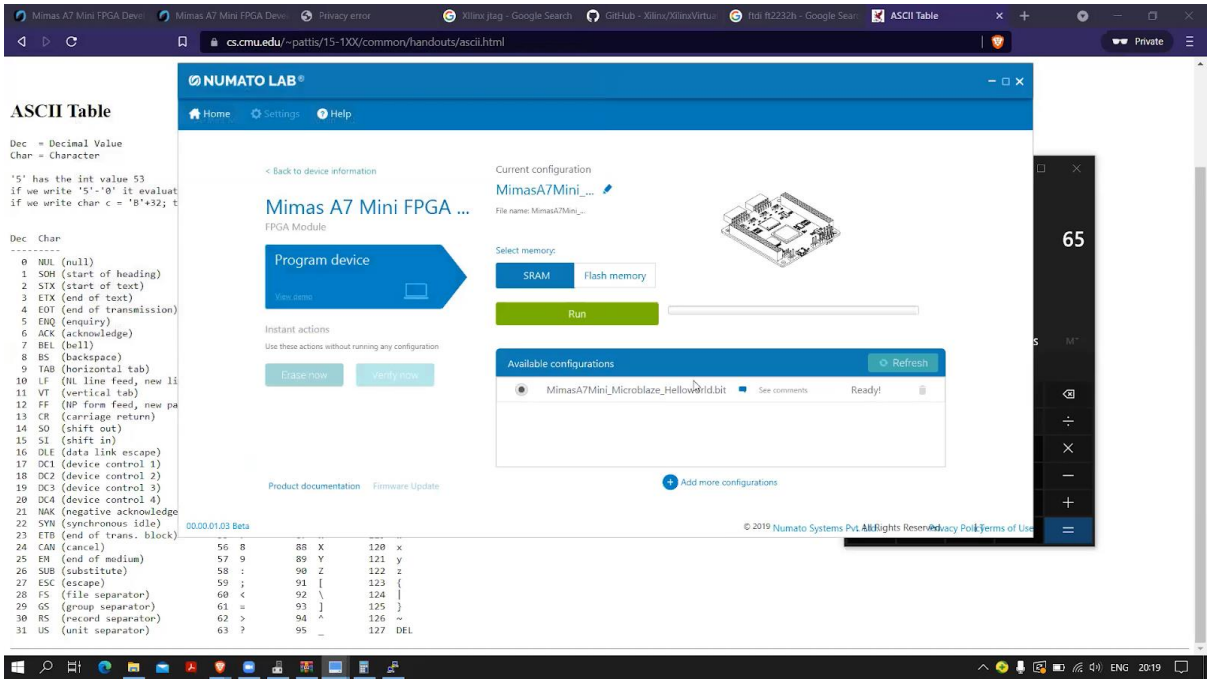
We could have used Xilinx jtag , a Xilinx platform cable, as it is quite expensive, we will not be using it rather we will using either of the two ways, XVC Server or Program device.



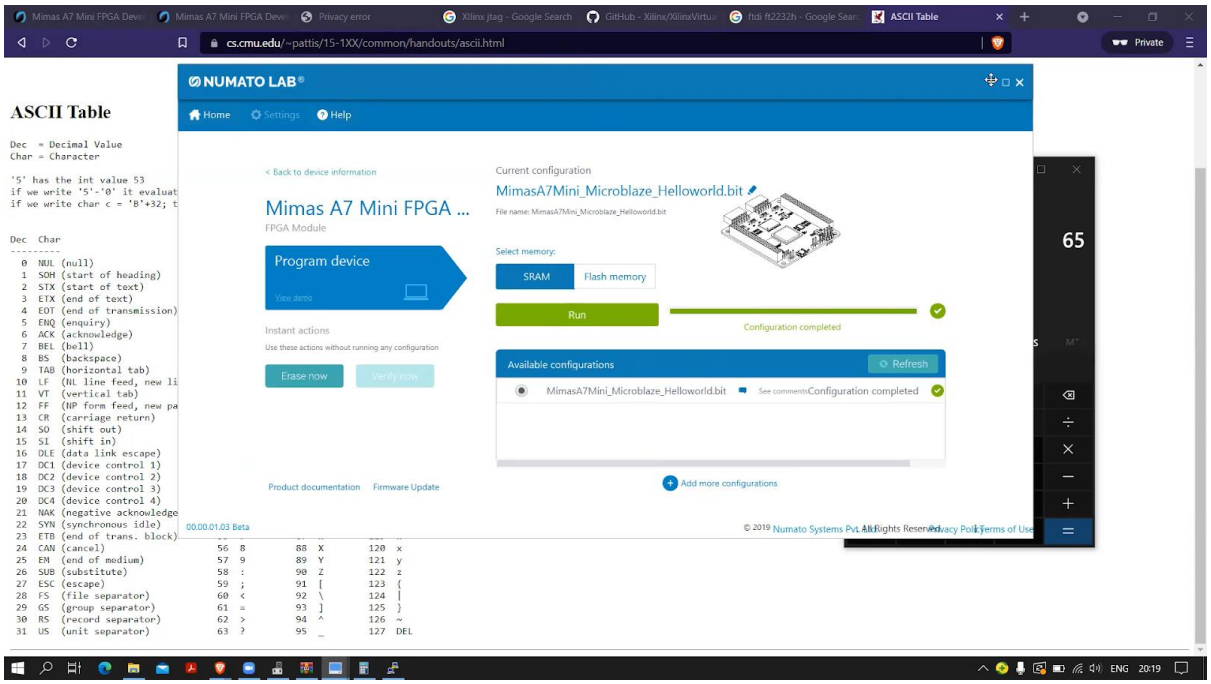
XVC server is nothing but Xilinx virtual cable, you can research and find out what it is. So, this is one way another way which we will be using to program is by an integrated circuit, FT2232HQ an onboard chip already available on Mimas A7 Mini, used for usb communication usb 2.0 communication not usb 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.



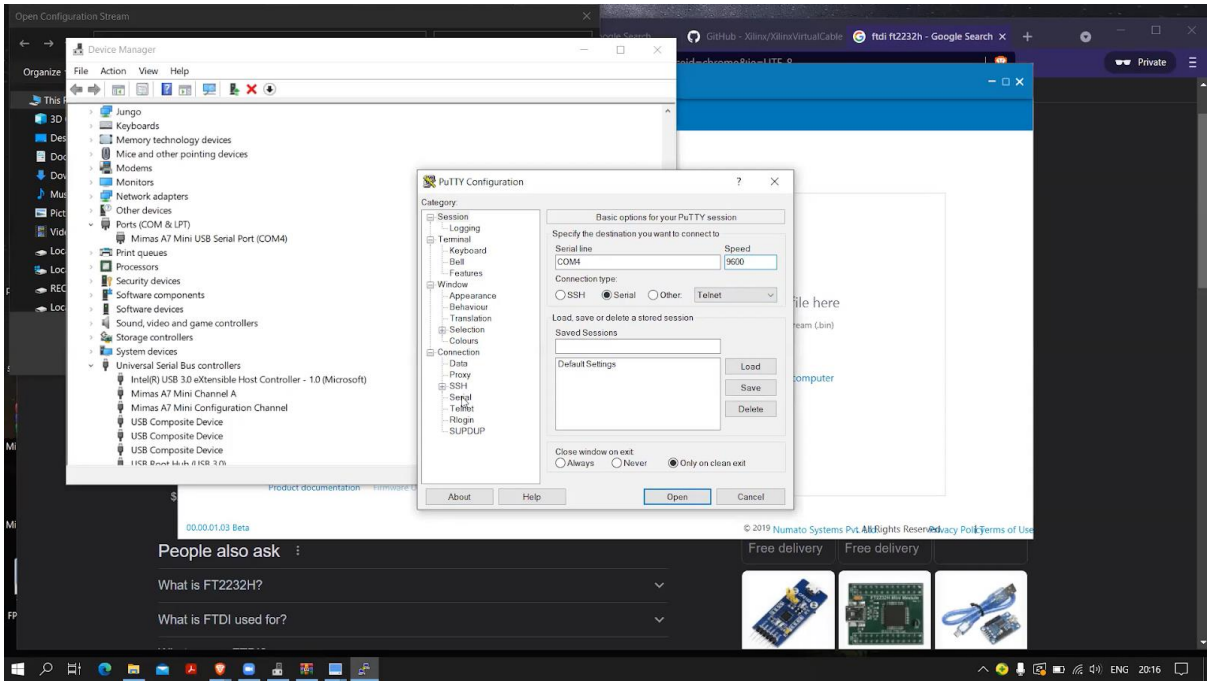
Add the bit file saved from Xilinx vivado integrated with this FPGA.

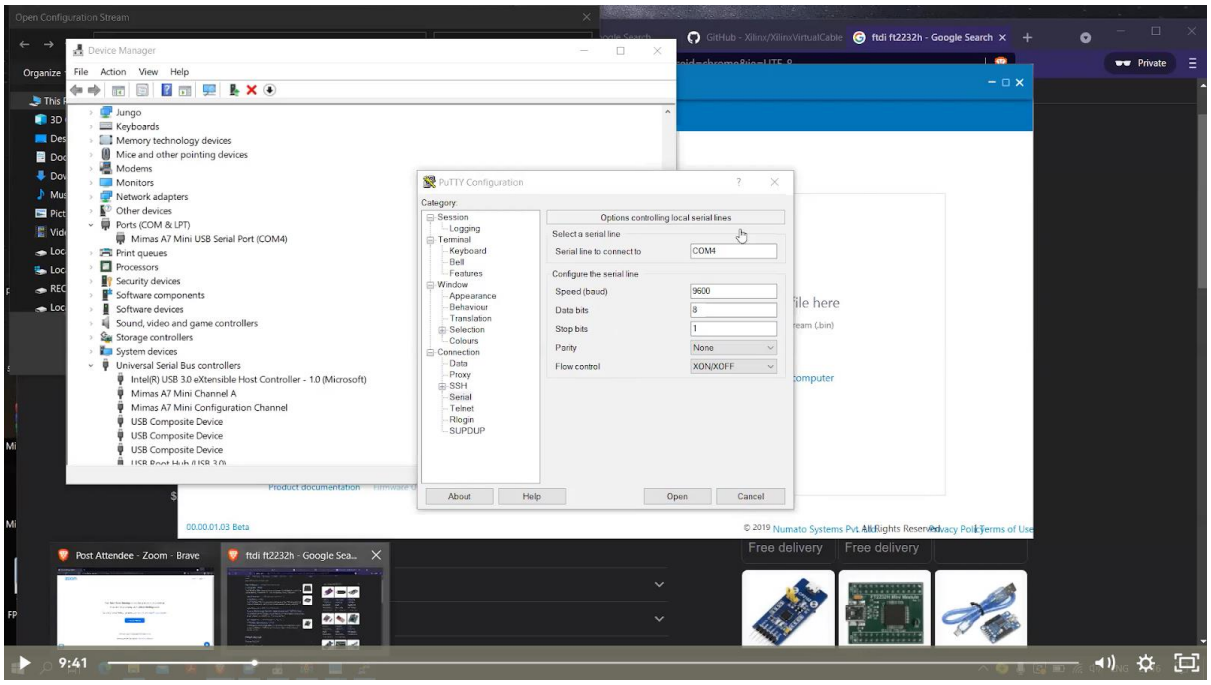


Run and configure the board with the program and after that if we have to launch putty software.

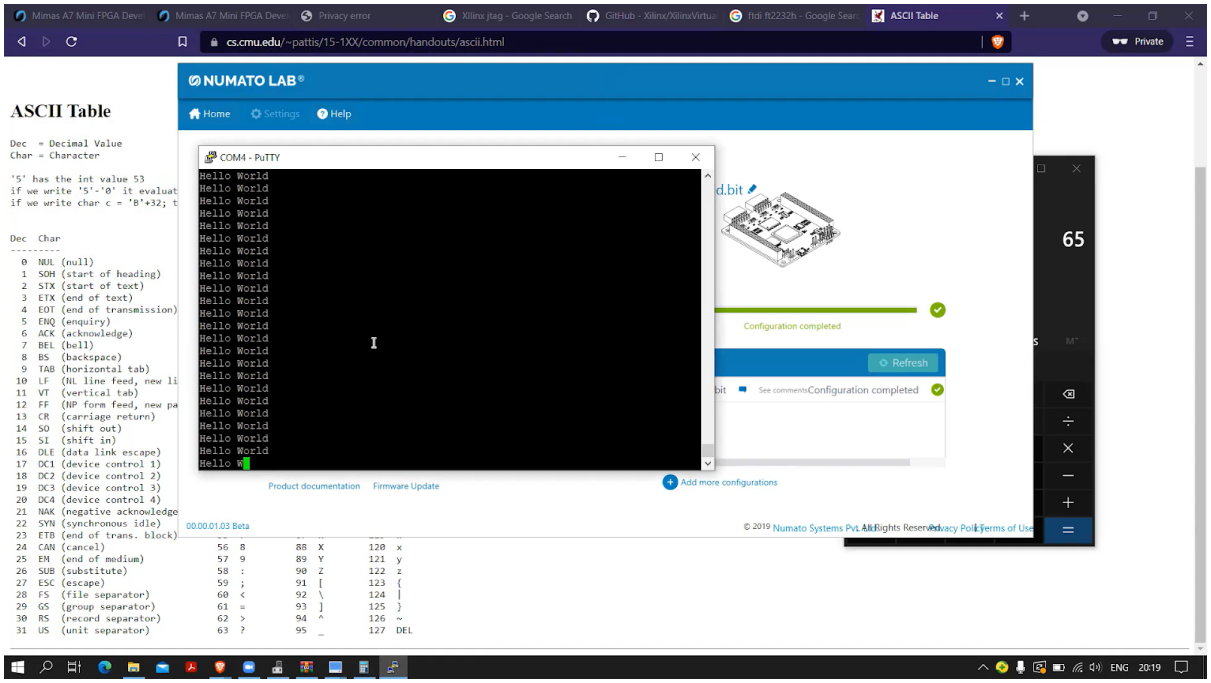


Step 2- Launch putty our serial monitors where it will receive data and make necessary changes with respect to port and connection type and so on.

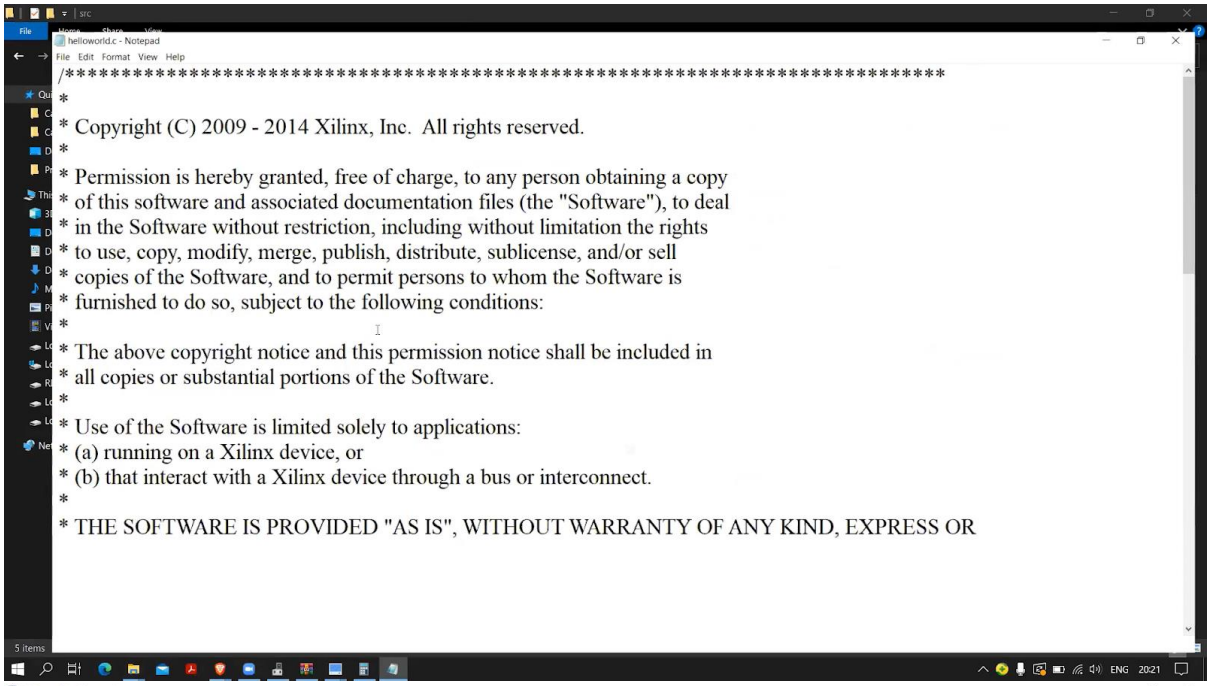


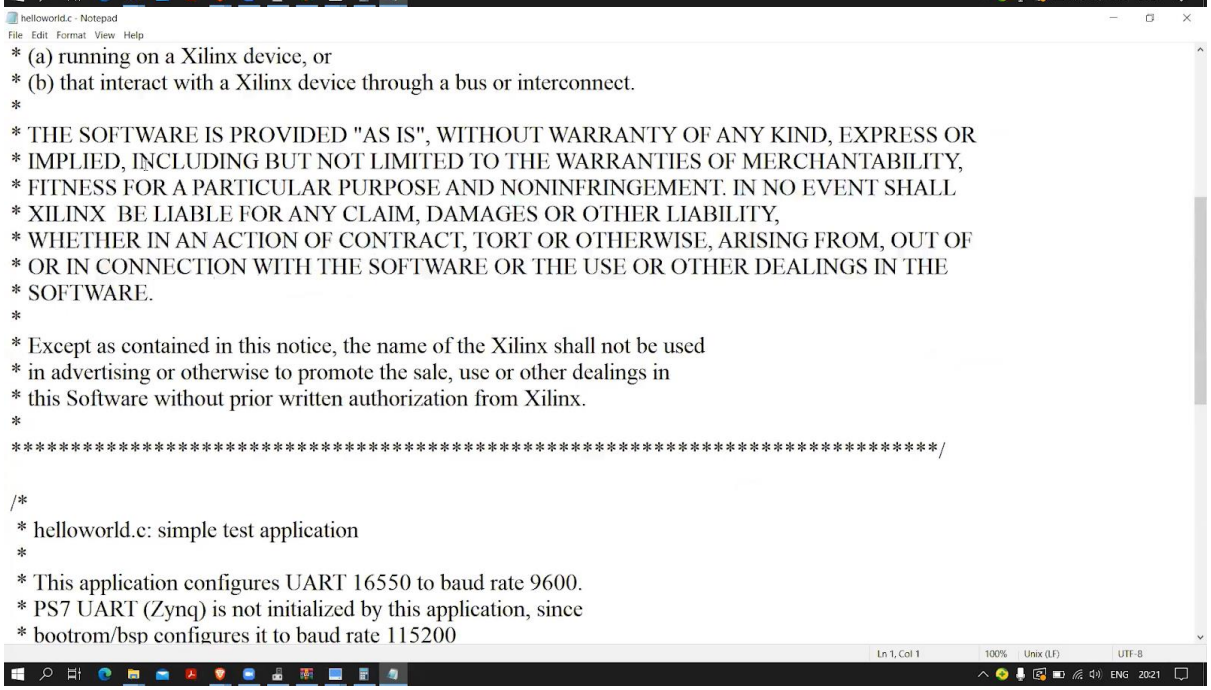


Putty session will continuously print Hello world after we upload the program to our FPGA and even, we can stop it from happening with a reset button present on the board.

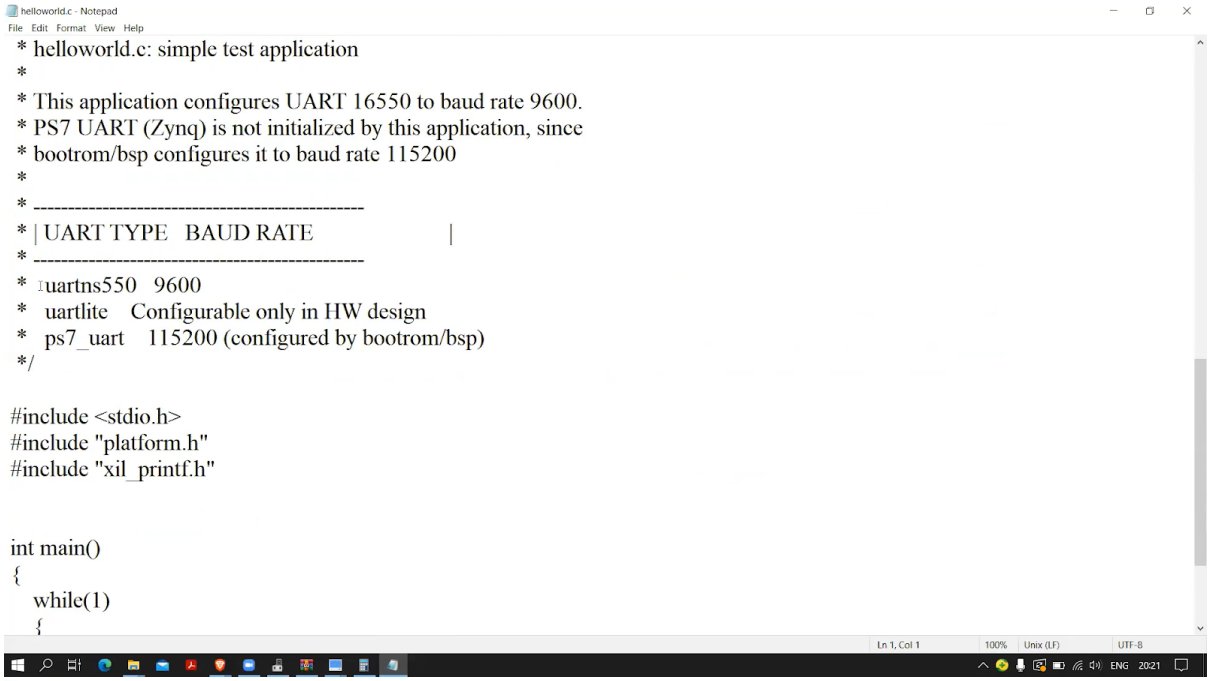


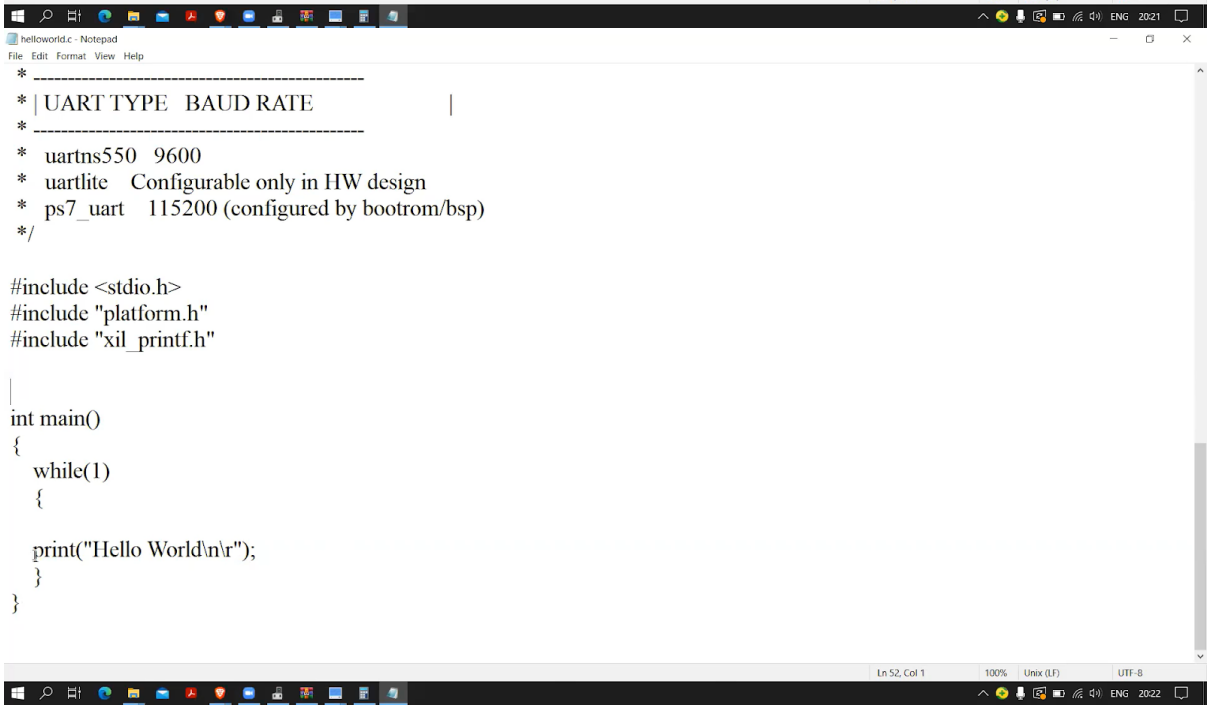
Open the C file where the program for hello world is written.


helloworld.c - Notepad
File Edit Format View Help
/***
*
* 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
5 items
ENG 20:21


helloworld.c - Notepad
File Edit Format View Help
* (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
Ln 1, Col 1
100%
Unix (LF)
UTF-8
ENG 20:21





Things to know before proceeding,

Sequential circuits

Slide Show
Resume Slide Show

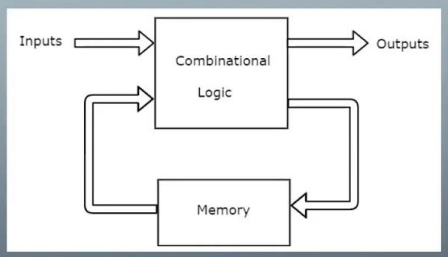
- Finite State Machines
  - The behavior of synchronous sequential circuits can be represented in the graphical form and it is known as **state diagram**.
  - Decision making logic and Control of the Digital System.
  - There are two types of FSMs.
    - Mealy State Machine
    - Moore State Machine

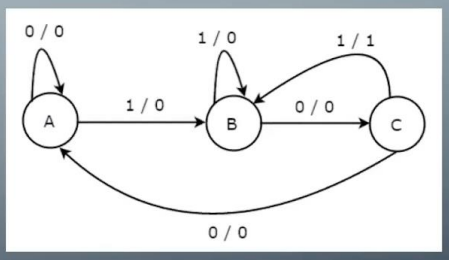
Sequential circuits

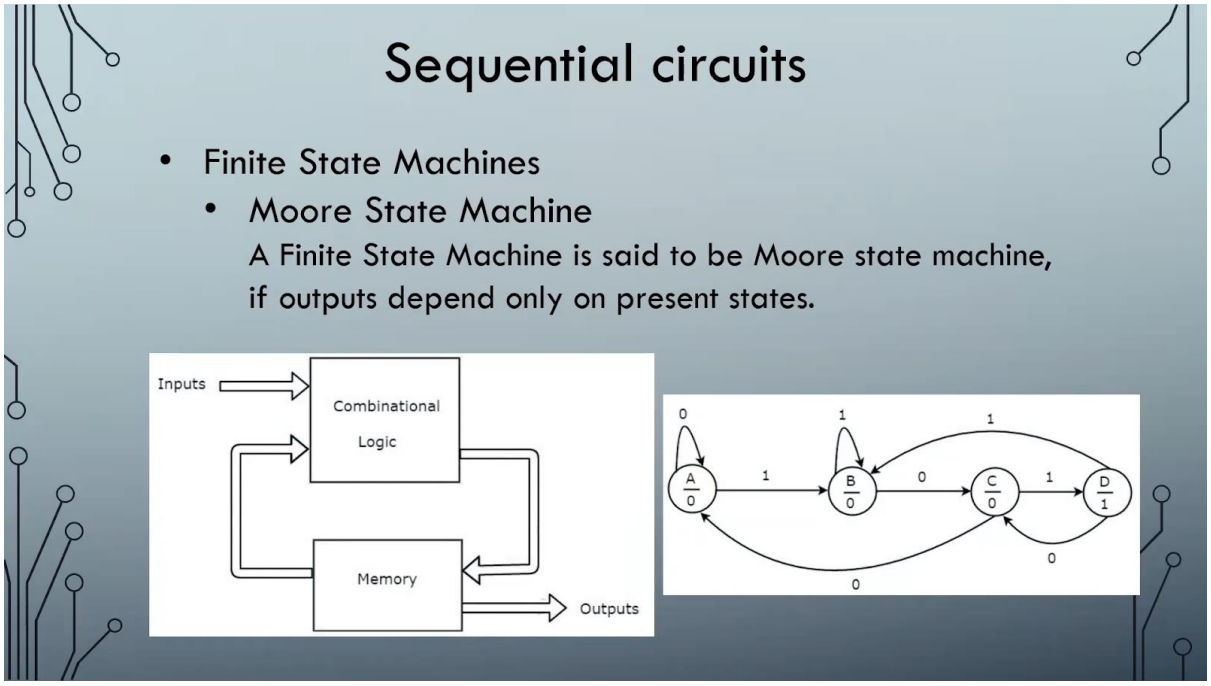
Slide Show
Resume Slide Show

- Finite State Machines
  - Mealy State Machine

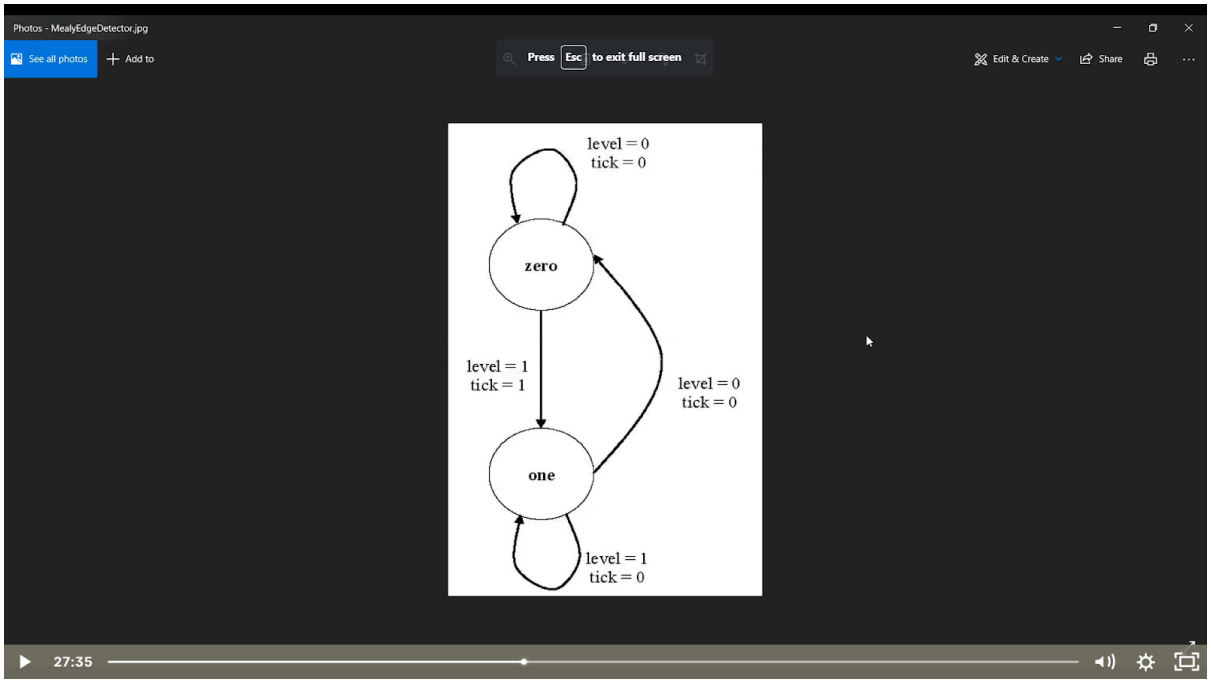
    A Finite State Machine is said to be Mealy state machine, if outputs depend on both present inputs & present states.



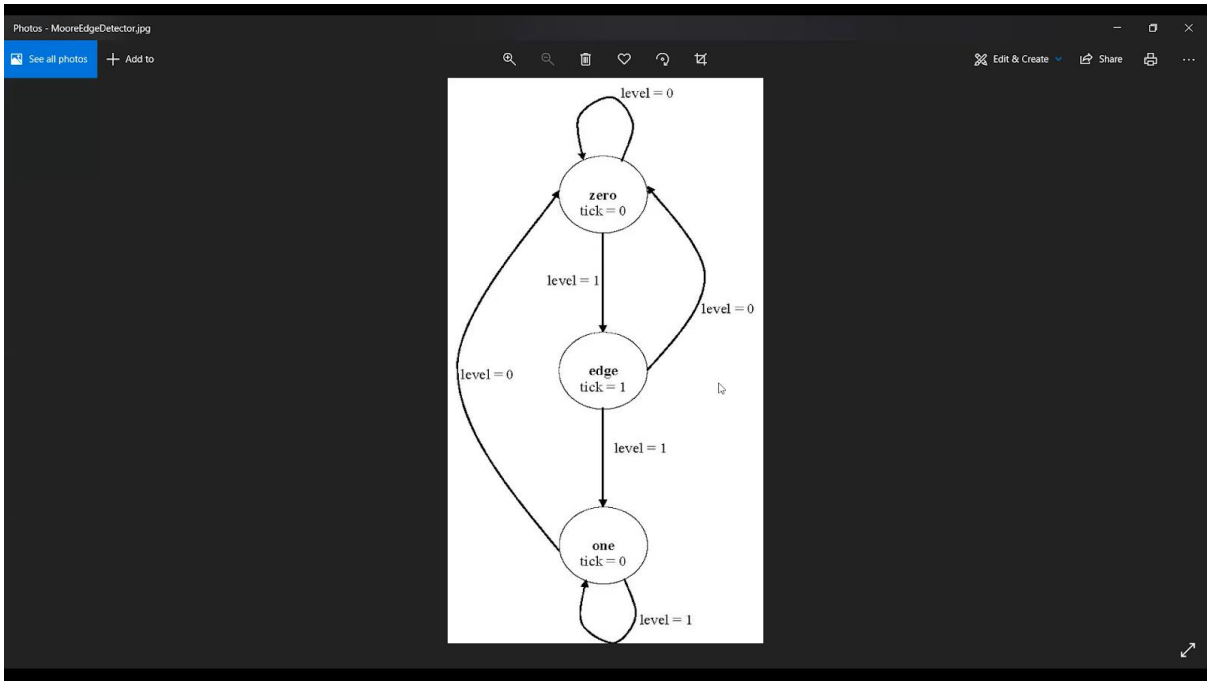




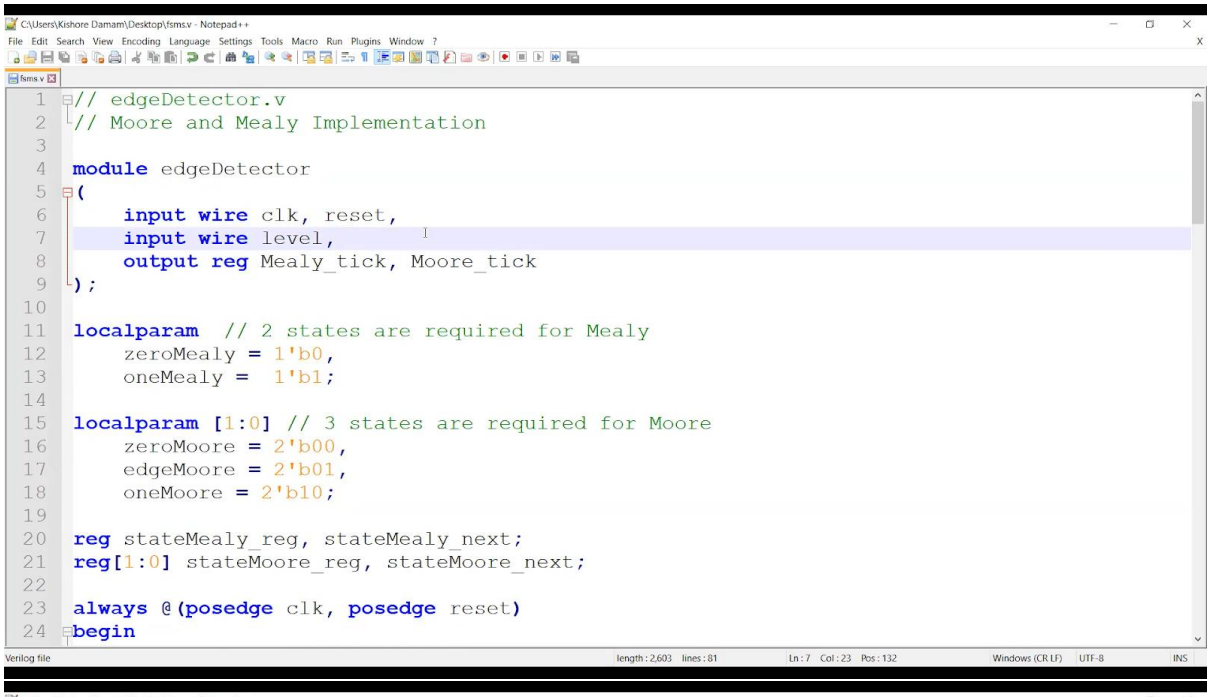
Mealy state machines



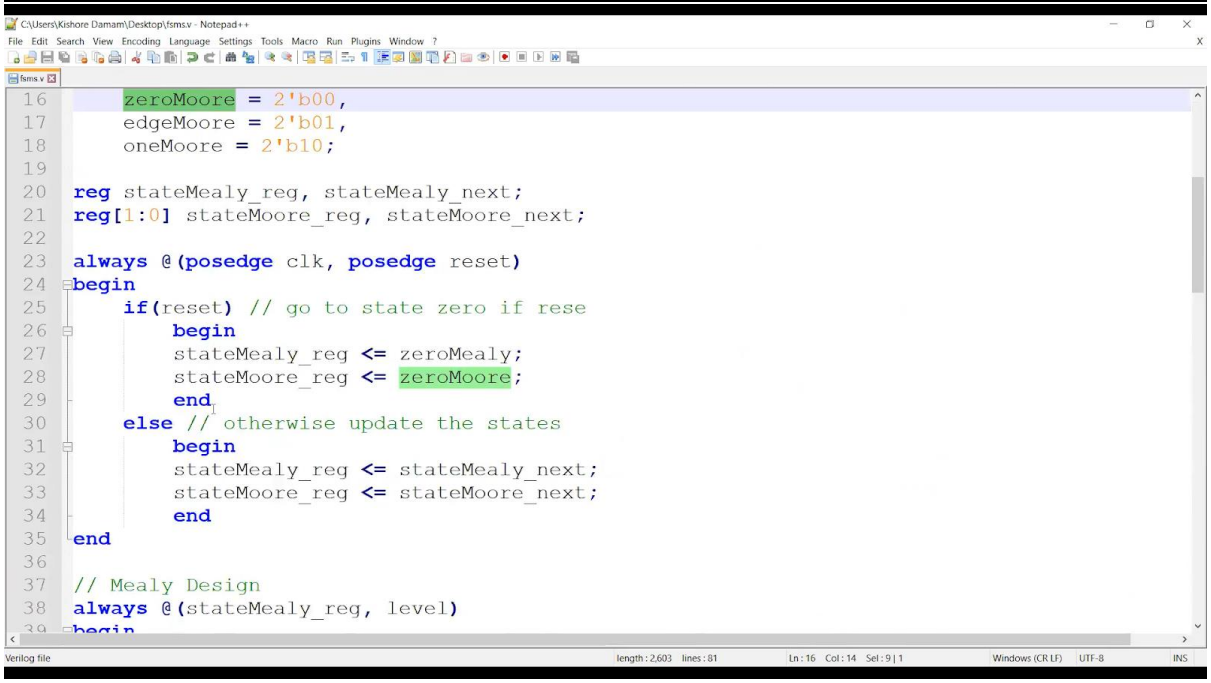
Moore state machines



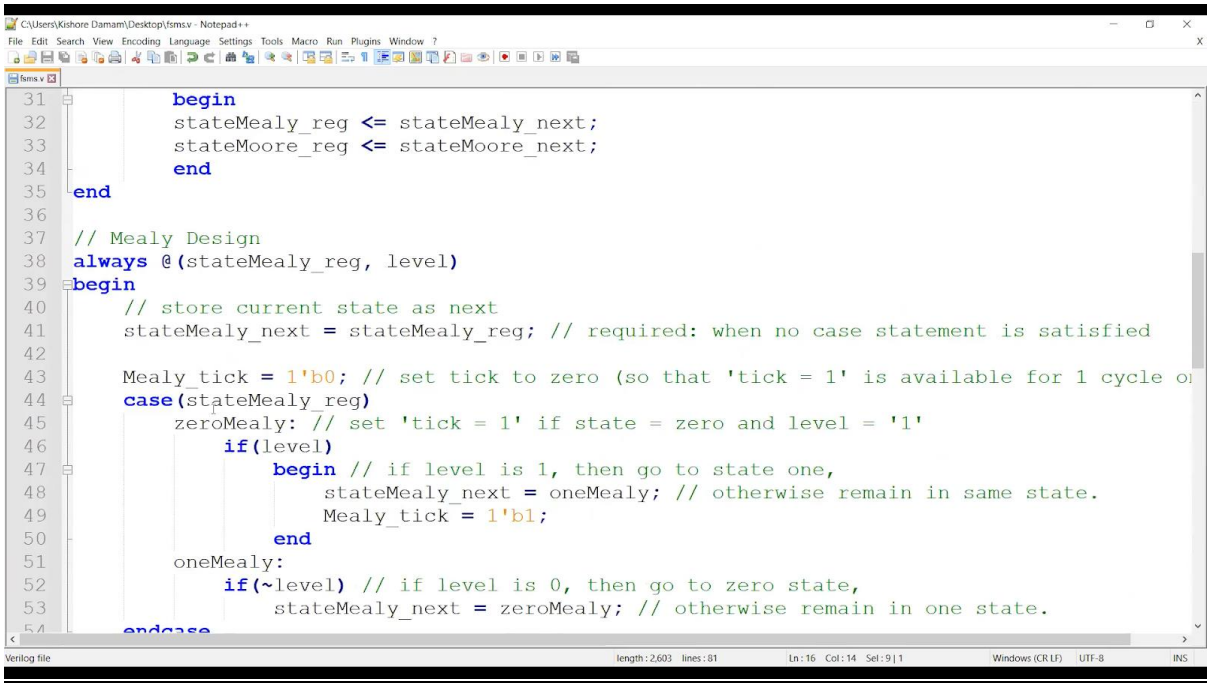
Step 3- Open notepad to write the Verilog code, a code to explain the basics of mealy and Moore state machine



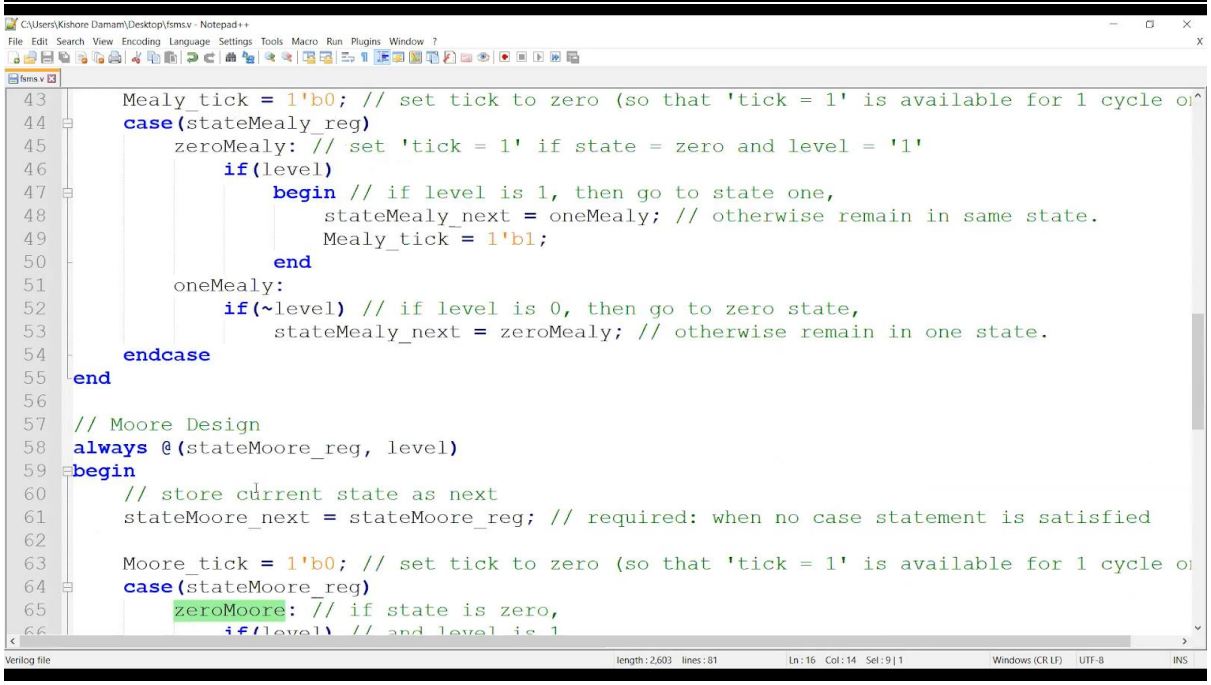
```
1  // edgeDetector.v
2  // Moore and Mealy Implementation
3
4  module edgeDetector
5  (
6      input wire clk, reset,
7      input wire level,
8      output reg Mealy_tick, Moore_tick
9  );
10
11 localparam  // 2 states are required for Mealy
12     zeroMealy = 1'b0,
13     oneMealy =  1'b1;
14
15 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
```

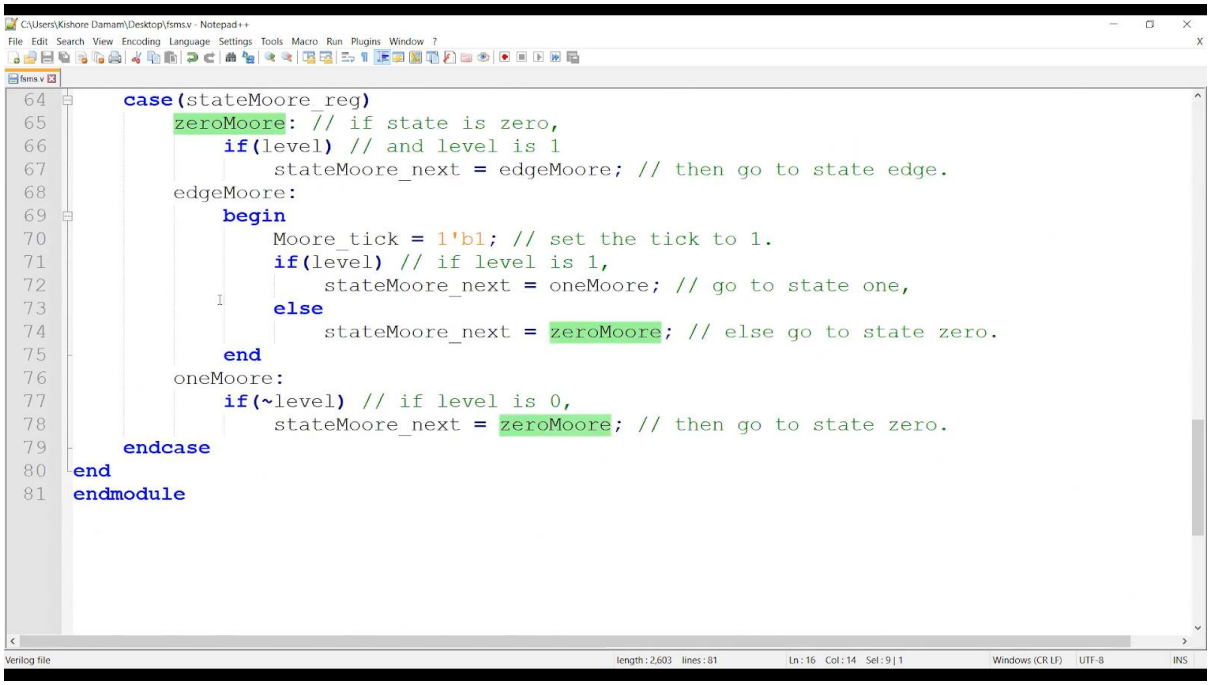
```
16     zeroMoore = 2'b00,
17     edgeMoore = 2'b01,
18     oneMoore = 2'b10;
19
20 reg stateMealy_reg, stateMealy_next;
21 reg[1:0] stateMoore_reg, stateMoore_next;
22
23 always @(posedge clk, posedge reset)
24 begin
25     if(reset) // go to state zero if rese
26         begin
27         stateMealy_reg <= 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
```



```
        begin
        stateMealy_reg <= stateMealy_next;
        stateMoore_reg <= stateMoore_next;
        end
end

// Mealy Design
always @(stateMealy_reg, level)
begin
    // store current state as next
    stateMealy_next = stateMealy_reg; // required: when no case statement is satisfied

    Mealy_tick = 1'b0; // set tick to zero (so that 'tick = 1' is available for 1 cycle o
    case(stateMealy_reg)
        zeroMealy: // set 'tick = 1' if state = zero and level = '1'
            if(level)
                begin // if level is 1, then go to state one,
                    stateMealy_next = oneMealy; // otherwise remain in same state.
                    Mealy_tick = 1'b1;
                end
        oneMealy:
            if(~level) // if level is 0, then go to zero state,
                stateMealy_next = zeroMealy; // otherwise remain in one state.
    endcase
```

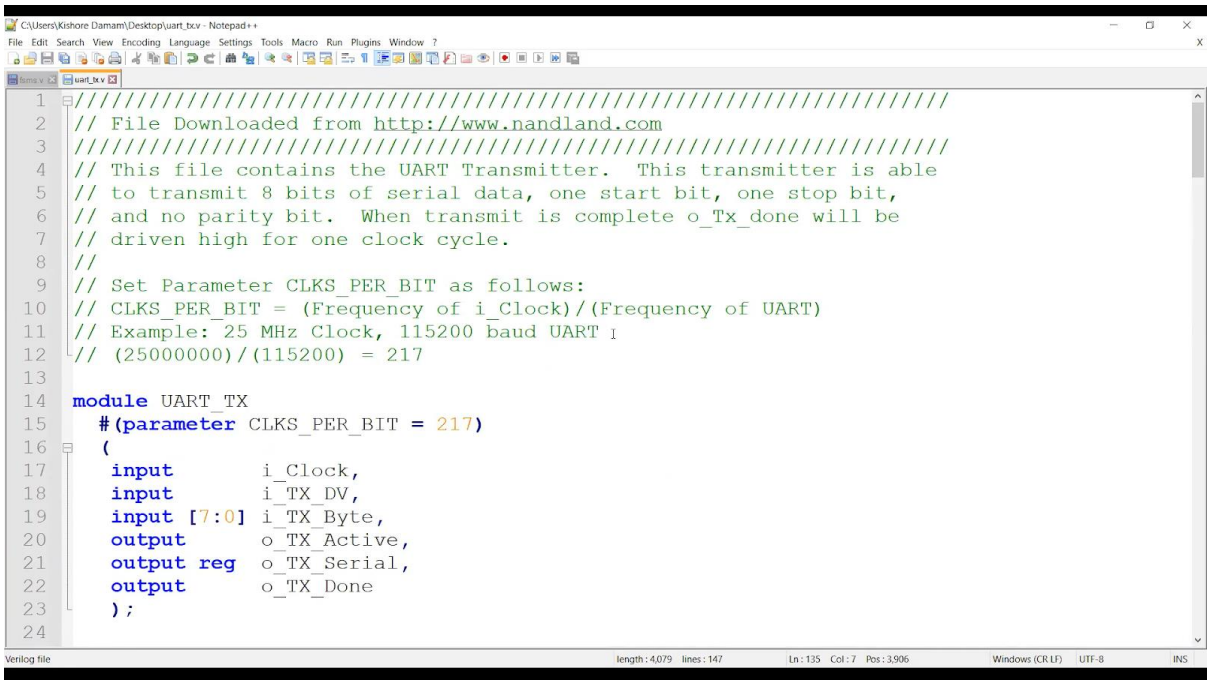



```
    Mealy_tick = 1'b0; // set tick to zero (so that 'tick = 1' is available for 1 cycle o
    case(stateMealy_reg)
        zeroMealy: // set 'tick = 1' if state = zero and level = '1'
            if(level)
                begin // if level is 1, then go to state one,
                    stateMealy_next = oneMealy; // otherwise remain in same state.
                    Mealy_tick = 1'b1;
                end
        oneMealy:
            if(~level) // if level is 0, then go to zero state,
                stateMealy_next = zeroMealy; // otherwise remain in one state.
    endcase
end

// Moore Design
always @(stateMoore_reg, level)
begin
    // store current state as next
    stateMoore_next = stateMoore_reg; // required: when no case statement is satisfied

    Moore_tick = 1'b0; // set tick to zero (so that 'tick = 1' is available for 1 cycle o
    case(stateMoore_reg)
        zeroMoore: // if state is zero,
```

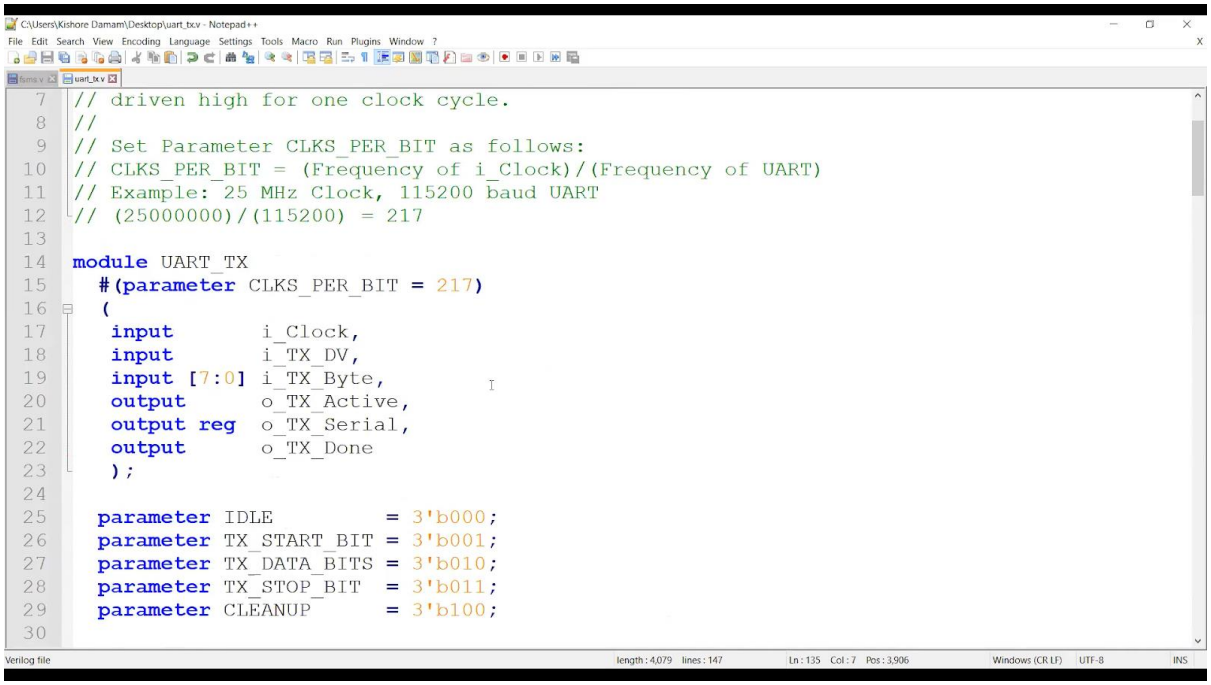


```
    case(stateMoore_reg)
        zeroMoore: // if state is zero,
            if(level) // and level is 1
                stateMoore_next = edgeMoore; // then go to state edge.
        edgeMoore:
            begin
                Moore_tick = 1'b1; // set the tick to 1.
                if(level) // if level is 1,
                    stateMoore_next = oneMoore; // go to state one,
                else
                    stateMoore_next = zeroMoore; // else go to state zero.
            end
        oneMoore:
            if(~level) // if level is 0,
                stateMoore_next = zeroMoore; // then go to state zero.
    endcase
end
endmodule
```

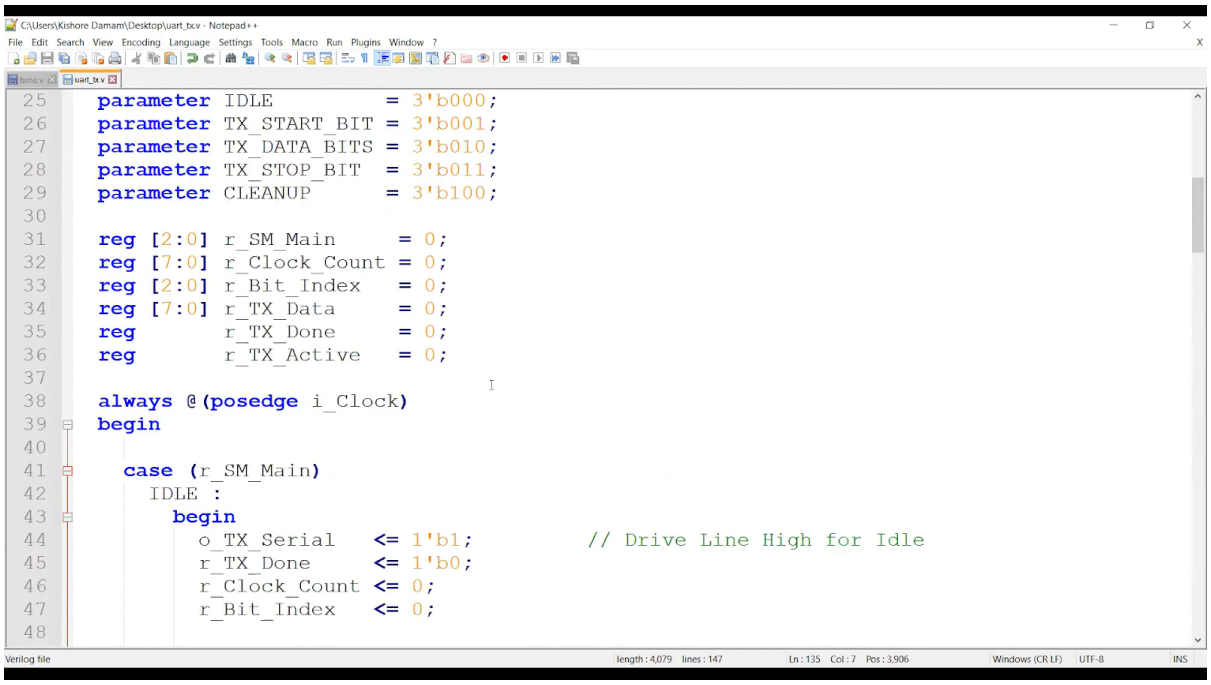
Open notepad to write the verilog code, a code to explain the UART module



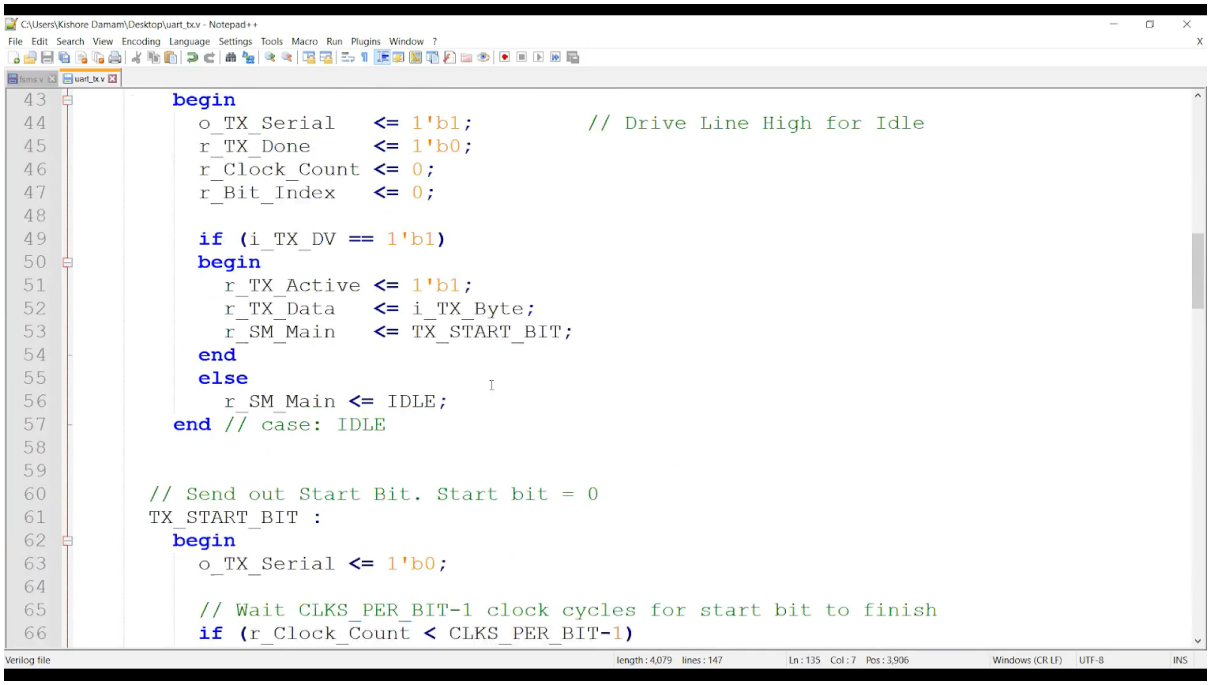
```
////////////////////////////////////////////////////////////////////////
// File Downloaded from http://www.nandland.com
////////////////////////////////////////////////////////////////////////
// This file contains the UART Transmitter.  This transmitter is able
// to transmit 8 bits of serial data, one start bit, one stop bit,
// and no parity bit.  When transmit is complete o_Tx_done will be
// driven high for one clock cycle.
//
// Set Parameter CLKS_PER_BIT as follows:
// CLKS_PER_BIT = (Frequency of i_Clock)/(Frequency of UART)
// Example: 25 MHz Clock, 115200 baud UART
// (25000000)/(115200) = 217

module UART_TX
  #(parameter CLKS_PER_BIT = 217)
  (
   input       i_Clock,
   input       i_TX_DV,
   input [7:0] i_TX_Byte,
   output      o_TX_Active,
   output reg  o_TX_Serial,
   output      o_TX_Done
   );
```



```
// driven high for one clock cycle.
//
// Set Parameter CLKS_PER_BIT as follows:
// CLKS_PER_BIT = (Frequency of i_Clock)/(Frequency of UART)
// Example: 25 MHz Clock, 115200 baud UART
// (25000000)/(115200) = 217

module UART_TX
  #(parameter CLKS_PER_BIT = 217)
  (
   input       i_Clock,
   input       i_TX_DV,
   input [7:0] i_TX_Byte,
   output      o_TX_Active,
   output reg  o_TX_Serial,
   output      o_TX_Done
   );

  parameter IDLE         = 3'b000;
  parameter TX_START_BIT = 3'b001;
  parameter TX_DATA_BITS = 3'b010;
  parameter TX_STOP_BIT  = 3'b011;
  parameter CLEANUP      = 3'b100;
```

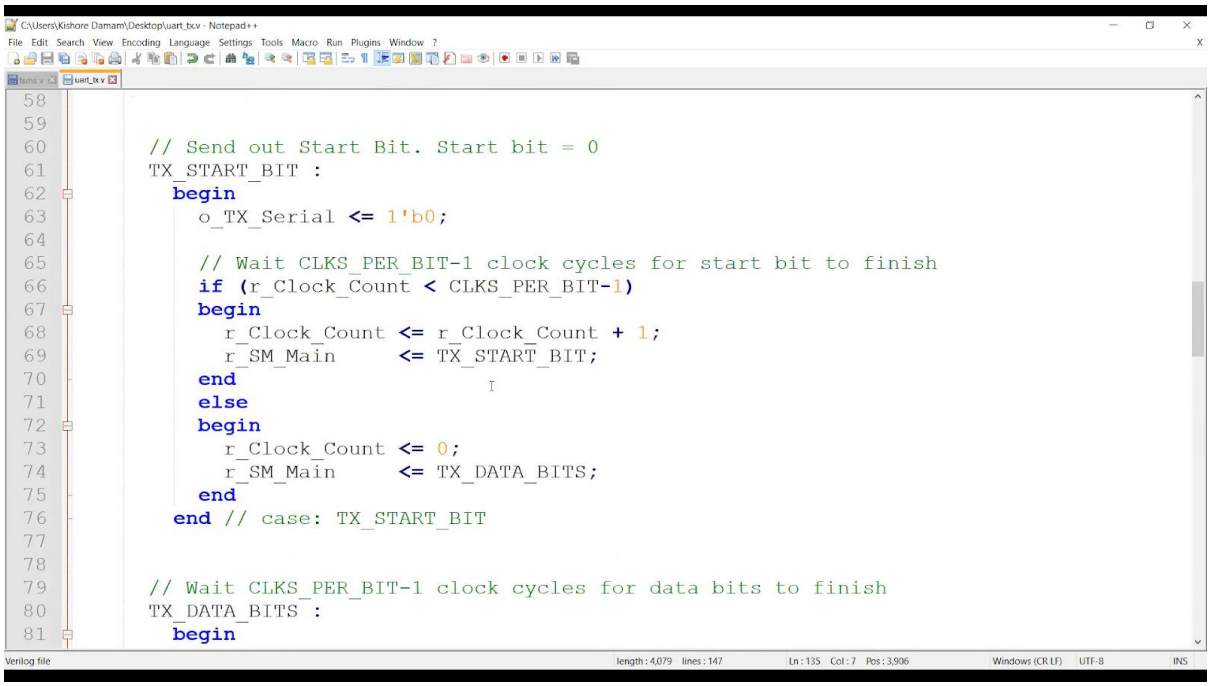


```
  parameter IDLE         = 3'b000;
  parameter TX_START_BIT = 3'b001;
  parameter TX_DATA_BITS = 3'b010;
  parameter TX_STOP_BIT  = 3'b011;
  parameter CLEANUP      = 3'b100;

  reg [2:0] r_SM_Main     = 0;
  reg [7:0] r_Clock_Count = 0;
  reg [2:0] r_Bit_Index   = 0;
  reg [7:0] r_TX_Data     = 0;
  reg       r_TX_Done     = 0;
  reg       r_TX_Active   = 0;

  always @(posedge i_Clock)
  begin

    case (r_SM_Main)
      IDLE :
        begin
          o_TX_Serial   <= 1'b1;         // Drive Line High for Idle
          r_TX_Done     <= 1'b0;
          r_Clock_Count <= 0;
          r_Bit_Index   <= 0;
```



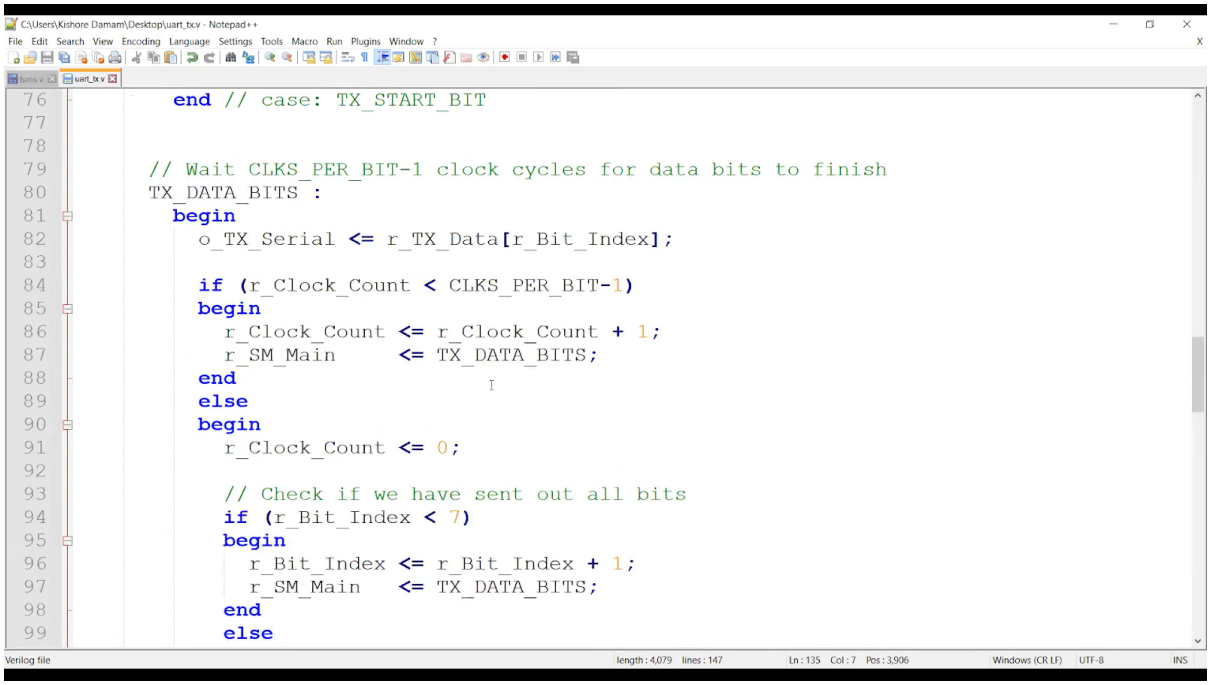
```
43        begin
44          o_TX_Serial    <= 1'b1;         // Drive Line High for Idle
45          r_TX_Done      <= 1'b0;
46          r_Clock_Count  <= 0;
47          r_Bit_Index    <= 0;
48
49          if (i_TX_DV == 1'b1)
50          begin
51            r_TX_Active <= 1'b1;
52            r_TX_Data   <= 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
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)
```

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

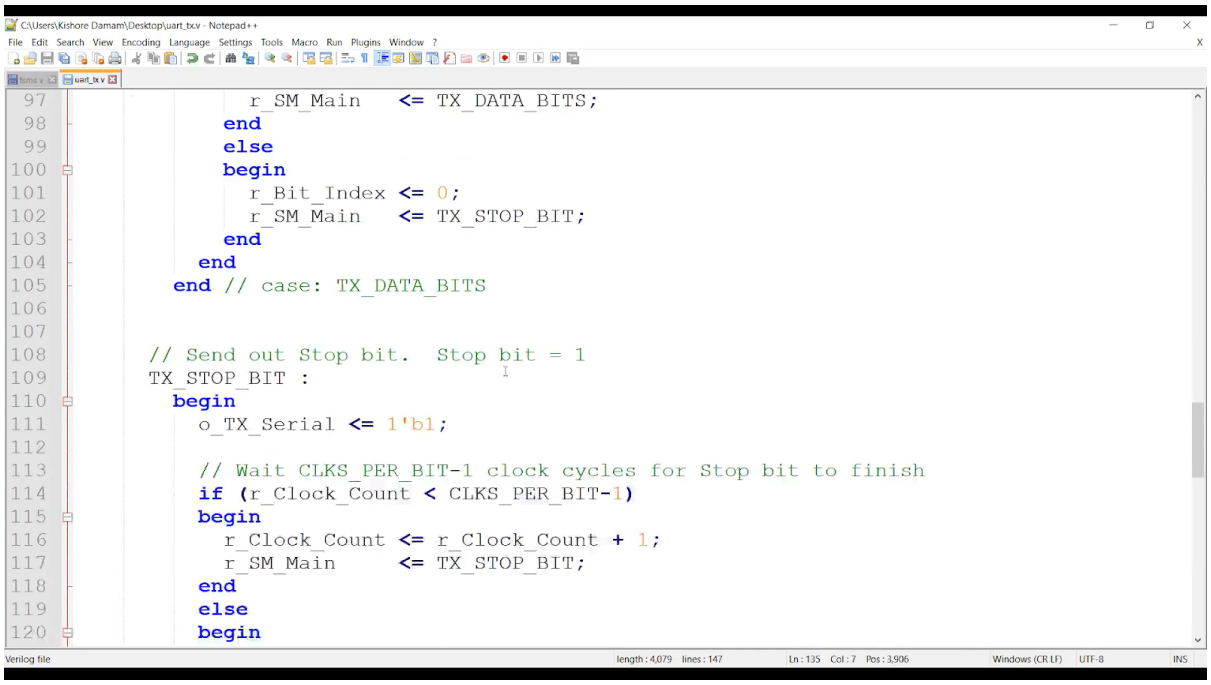


```
58
59
60      // Send out Start Bit. Start bit = 0
61      TX_START_BIT :
62        begin
63          o_TX_Serial <= 1'b0;
64
65          // Wait CLKS_PER_BIT-1 clock cycles for start bit to finish
66          if (r_Clock_Count < CLKS_PER_BIT-1)
67          begin
68            r_Clock_Count <= r_Clock_Count + 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
```

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



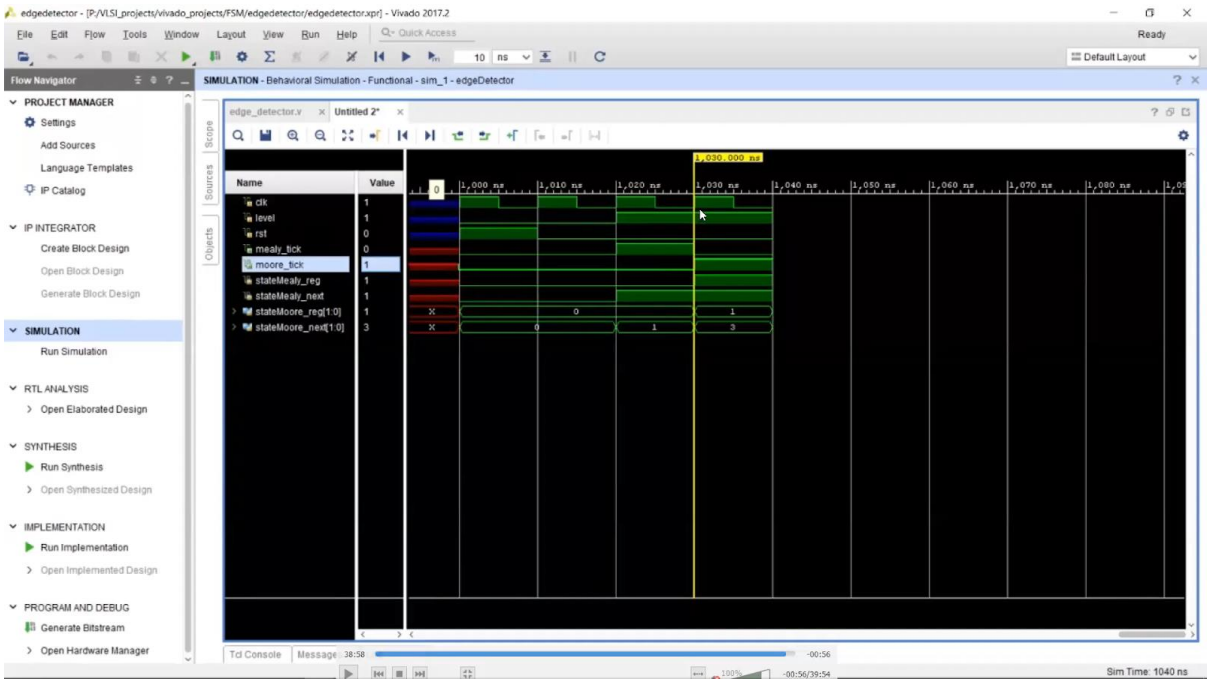
```
76          end // case: TX_START_BIT
77
78
79        // Wait CLKS_PER_BIT-1 clock cycles for data bits to finish
80        TX_DATA_BITS :
81          begin
82            o_TX_Serial <= r_TX_Data[r_Bit_Index];
83
84            if (r_Clock_Count < CLKS_PER_BIT-1)
85            begin
86              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
```



```
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
```

```
            r_TX_Active    <= 1'b0;
          end
        end // case: TX_STOP_BIT


      // Stay here 1 clock
      CLEANUP :
        begin
          r_TX_Done <= 1'b1;
          r_SM_Main <= IDLE;
        end


      default :
        r_SM_Main <= IDLE;

    endcase
  end

  assign o_TX_Active = r_TX_Active;
  assign o_TX_Done   = r_TX_Done;

endmodule
```

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

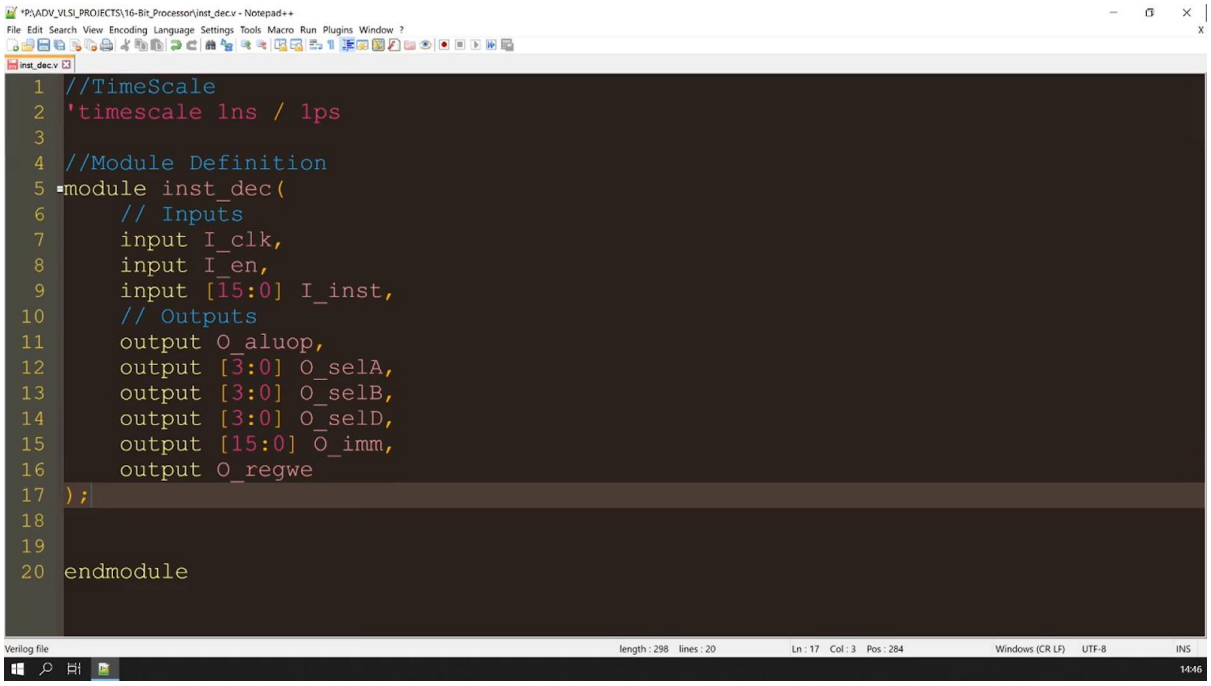
Software to download

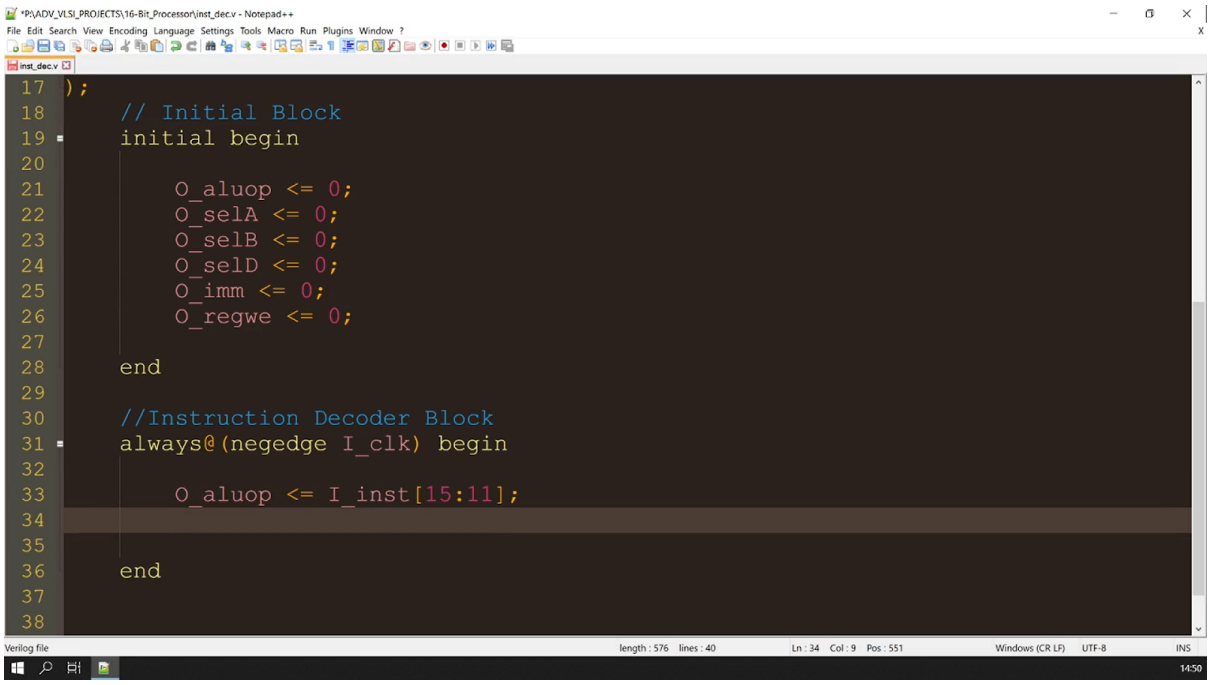
- Notepad++

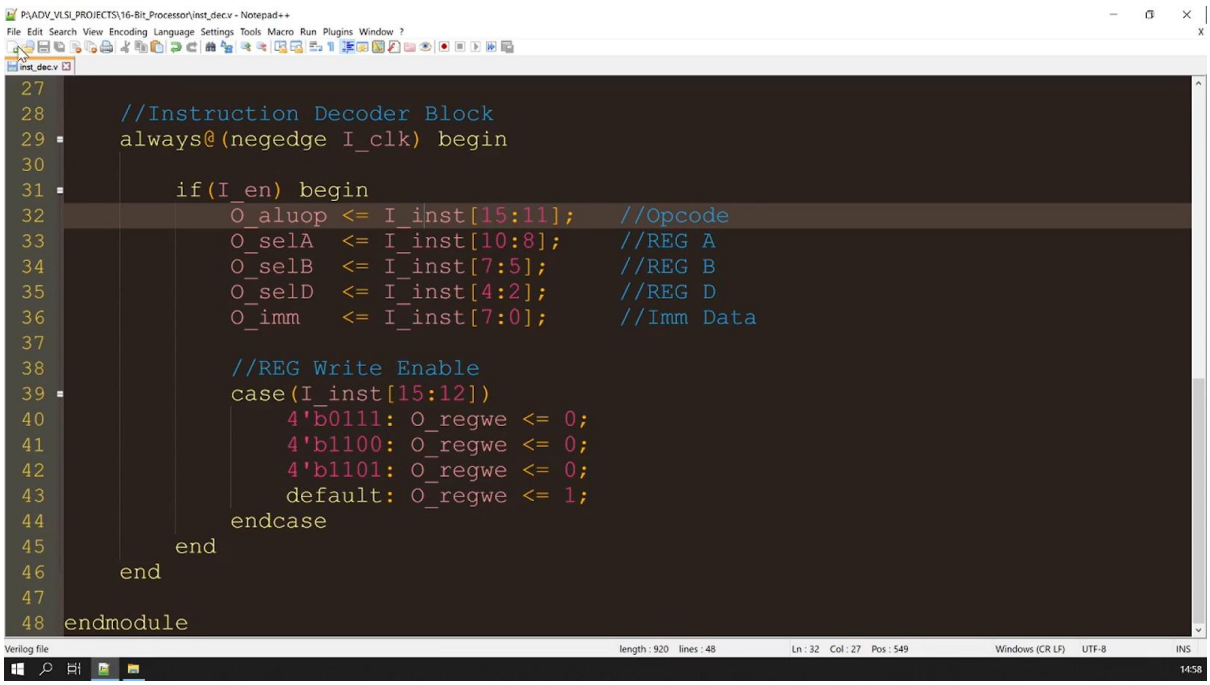
Step 1- Write the verilog program in notepad ++ .

DESIGN CODES

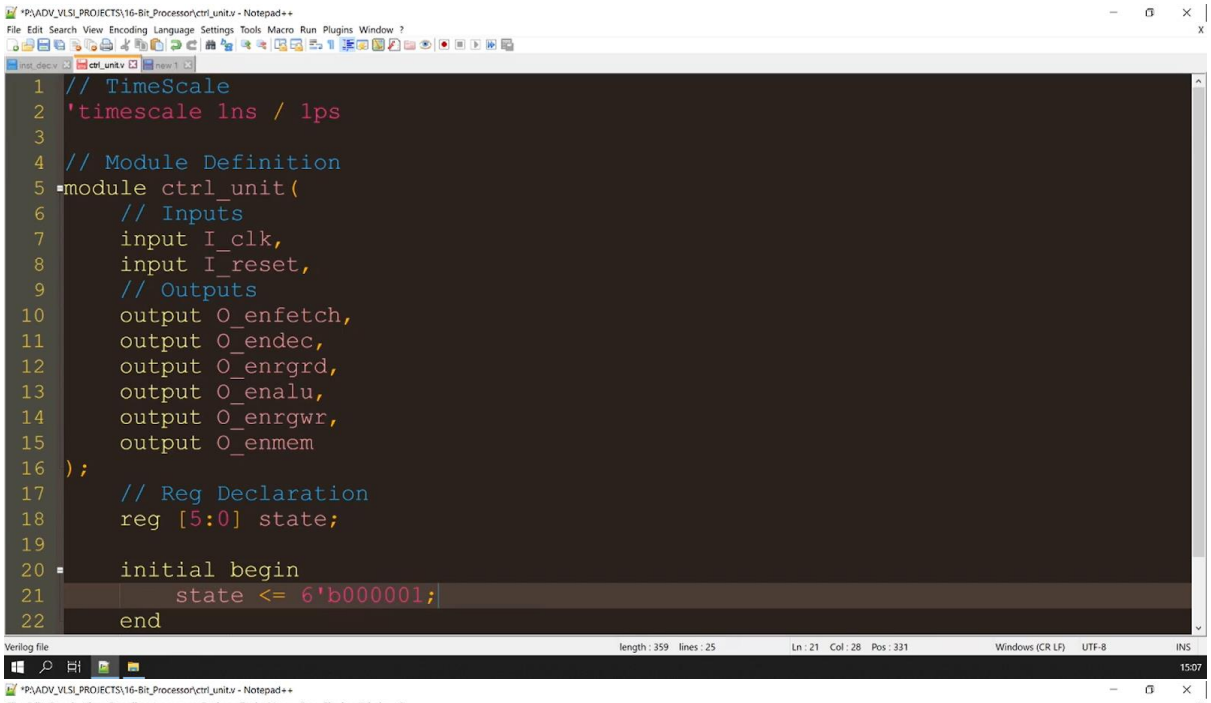
Module inst_dec

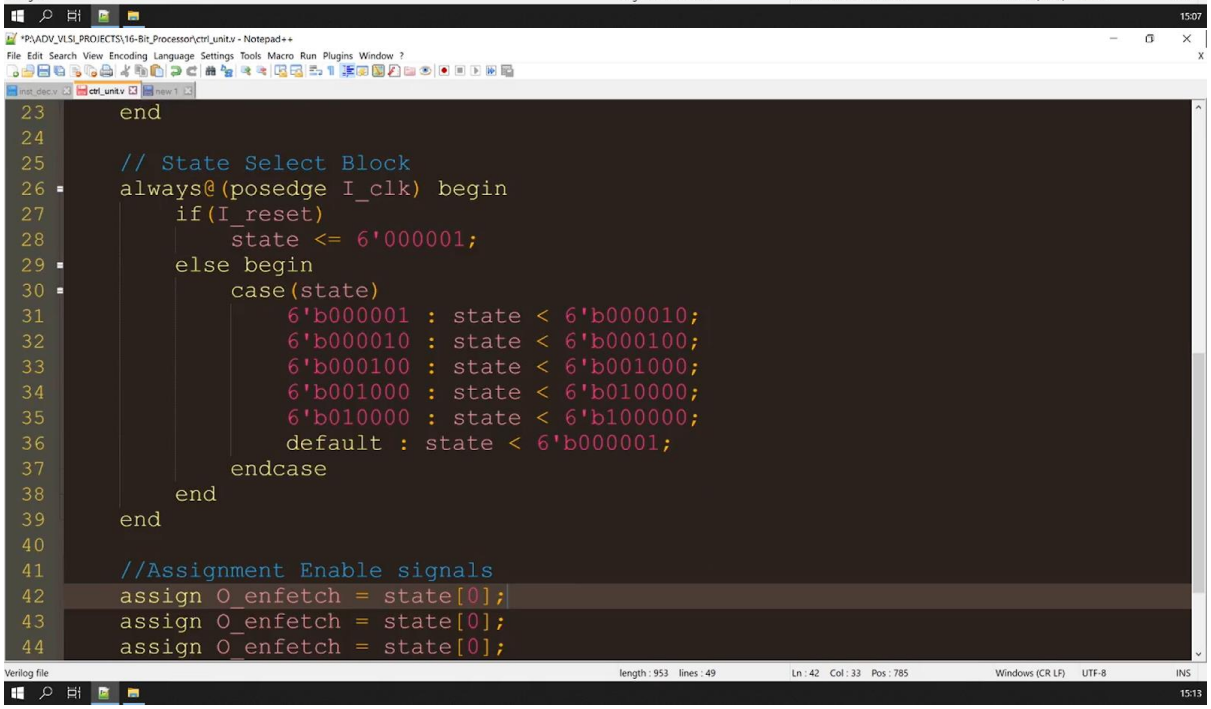

*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\inst_dec.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
inst_dec.v
1 //TimeScale
2 'timescale 1ns / 1ps
3
4 //Module Definition
5 module inst_dec(
6 // Inputs
7 input I_clk,
8 input I_en,
9 input [15:0] I_inst,
10 // 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
14:46

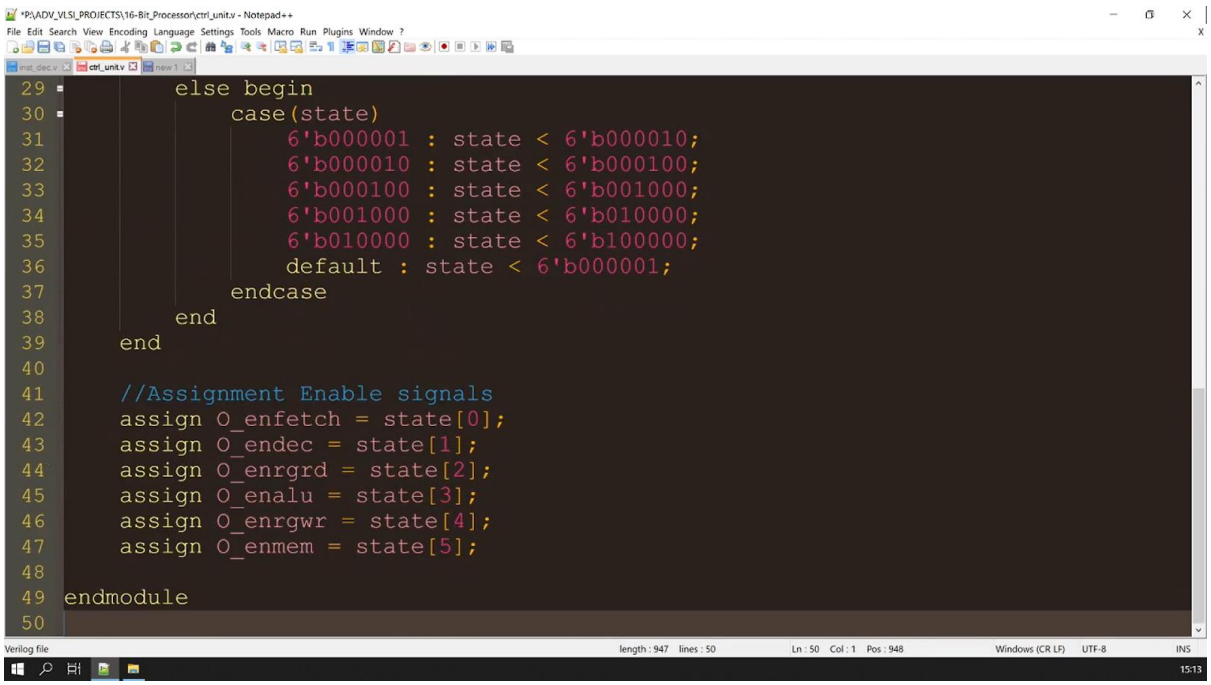

*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\inst_dec.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
inst_dec.v
17);
18 // Initial Block
19 initial begin
20
21 O_aluop <= 0;
22 O_selA <= 0;
23 O_selB <= 0;
24 O_selD <= 0;
25 O_imm <= 0;
26 O_regwe <= 0;
27
28 end
29
30 //Instruction Decoder Block
31 always@(negedge I_clk) begin
32
33 O_aluop <= I_inst[15:11];
34
35
36 end
37
38
Verilog file
length : 576 lines : 40
Ln : 34 Col : 9 Pos : 551
Windows (CR LF)
UTF-8
INS
14:50



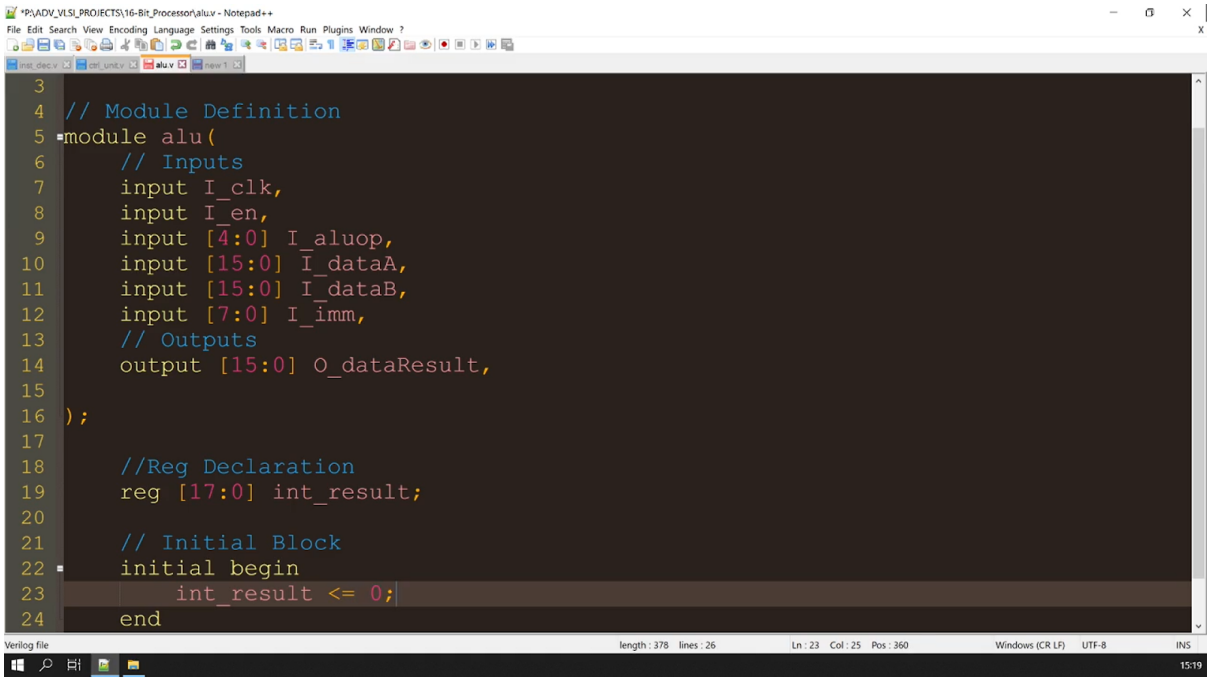
Module ctrl_unit

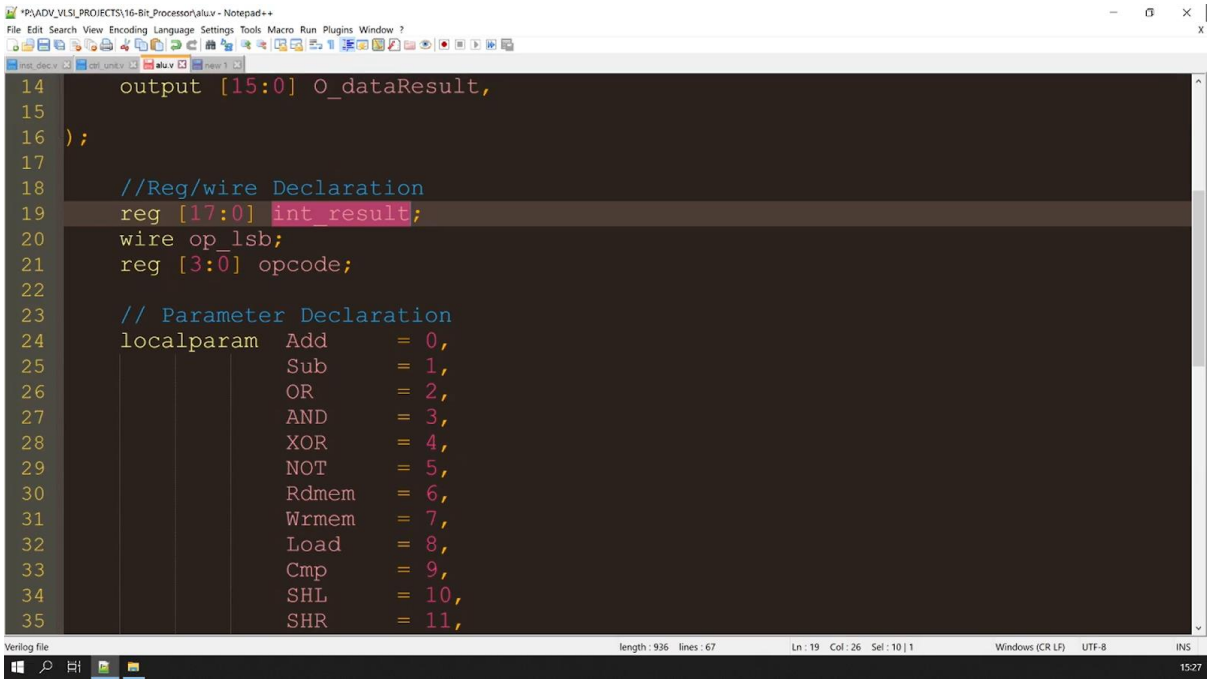

*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\ctrl_unit.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
inst_dec.v
ctrl_unit.v
new 1
// TimeScale
'timescale 1ns / 1ps
// Module Definition
module ctrl_unit(
// Inputs
input I_clk,
input I_reset,
// Outputs
output O_enfetch,
output O_endec,
output O_enrgrd,
output O_enalu,
output O_enrgwr,
output O_enmem
);
// Reg Declaration
reg [5:0] state;
initial begin
state <= 6'b000001;
end
Verilog file
length : 359 lines : 25
Ln : 21 Col : 28 Pos : 331
Windows (CR LF)
UTF-8
INS
15:07


*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\ctrl_unit.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
inst_dec.v
ctrl_unit.v
new 1
end
// State Select Block
always@(posedge I_clk) begin
if(I_reset)
state <= 6'000001;
else begin
case(state)
6'b000001 : state < 6'b000010;
6'b000010 : state < 6'b000100;
6'b000100 : state < 6'b001000;
6'b001000 : state < 6'b010000;
6'b010000 : state < 6'b100000;
default : state < 6'b000001;
endcase
end
end
//Assignment Enable signals
assign O_enfetch = state[0];
assign O_enfetch = state[0];
assign O_enfetch = state[0];
Verilog file
length : 953 lines : 49
Ln : 42 Col : 33 Pos : 785
Windows (CR LF)
UTF-8
INS
15:13

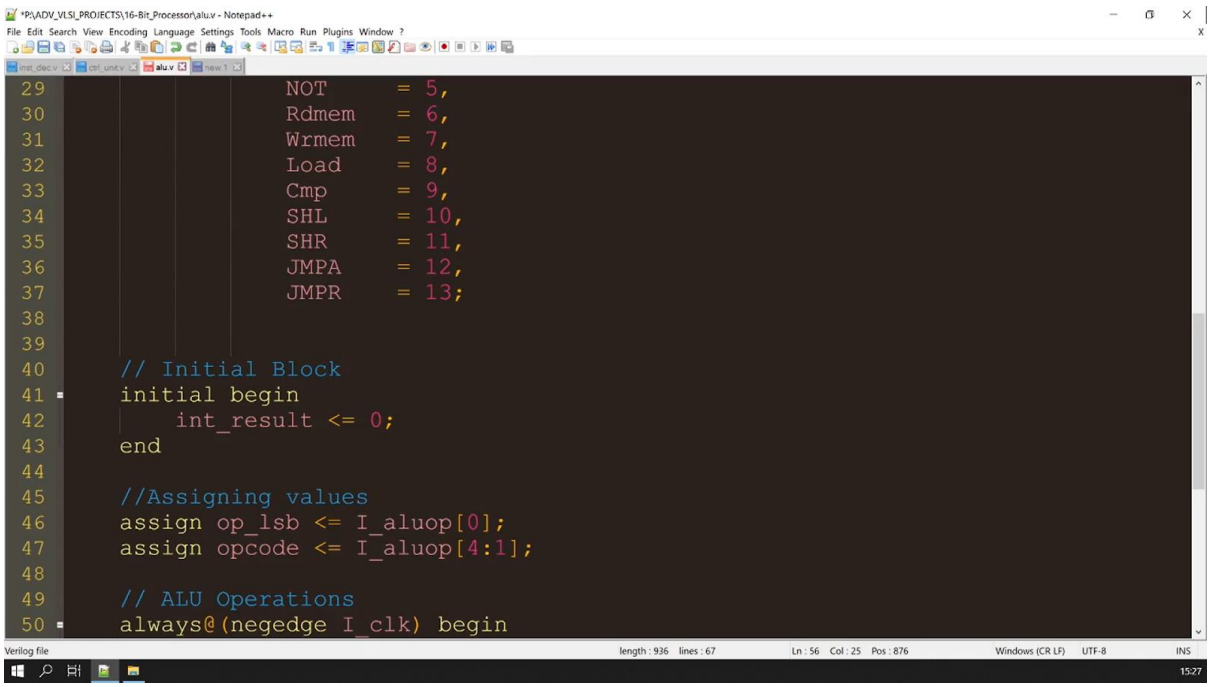

*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\ctrl_unit.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
inst_dec.v
ctrl_unit.v
new 1
else begin
case(state)
6'b000001 : state < 6'b000010;
6'b000010 : state < 6'b000100;
6'b000100 : state < 6'b001000;
6'b001000 : state < 6'b010000;
6'b010000 : state < 6'b100000;
default : state < 6'b000001;
endcase
end
end
//Assignment Enable signals
assign O_enfetch = state[0];
assign O_endec = state[1];
assign O_enrgrd = state[2];
assign O_enalu = state[3];
assign O_enrgwr = state[4];
assign O_enmem = state[5];
endmodule
Verilog file
length : 947 lines : 50
Ln : 50 Col : 1 Pos : 948
Windows (CR LF)
UTF-8
INS
15:13

Module alu

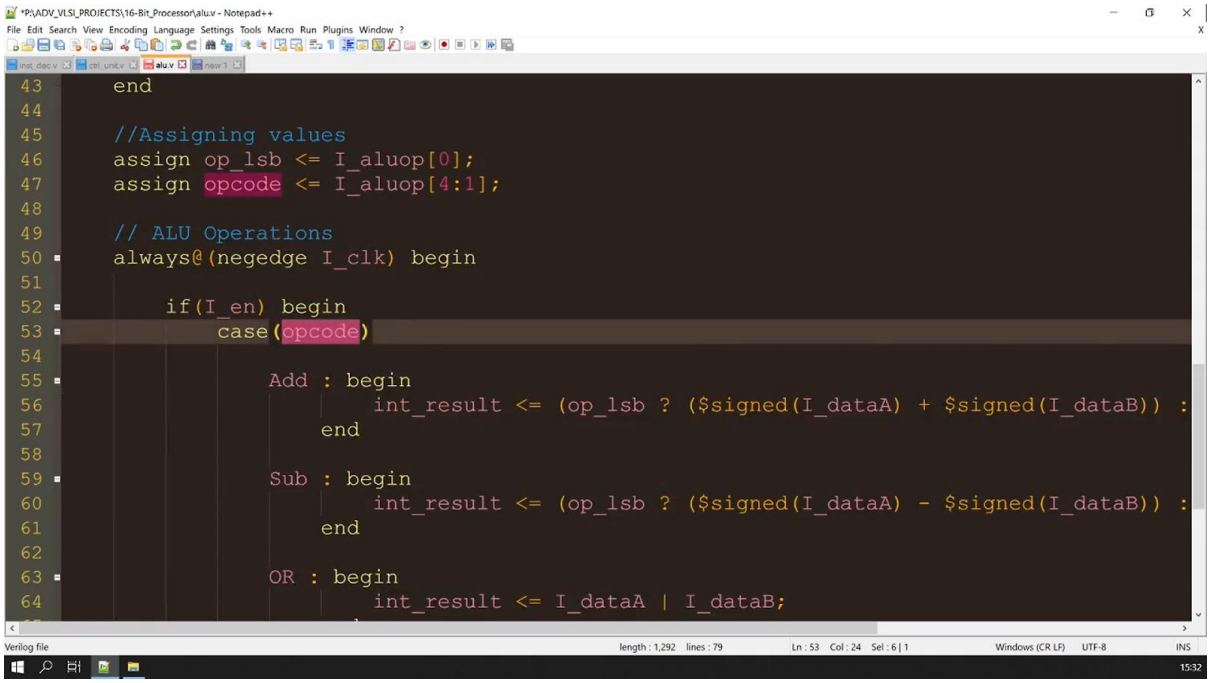

*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\alu.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
inst_dec.v
ctrl_unit.v
alu.v
new 1
// Module Definition
module alu(
// Inputs
input I_clk,
input I_en,
input [4:0] I_aluop,
input [15:0] I_dataA,
input [15:0] I_dataB,
input [7:0] I_imm,
// Outputs
output [15:0] O_dataResult,
);
//Reg Declaration
reg [17:0] int_result;
// Initial Block
initial begin
int_result <= 0;
end
Verilog file
length : 378 lines : 26
Ln : 23 Col : 25 Pos : 360
Windows (CR LF)
UTF-8
INS
15:19

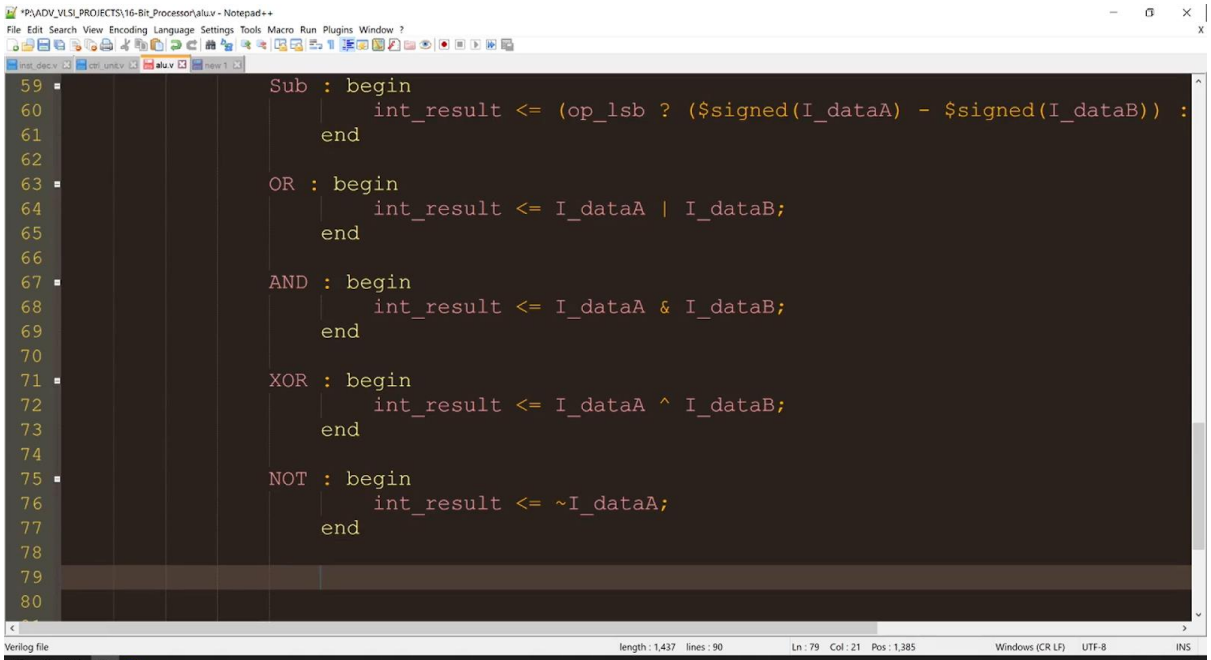


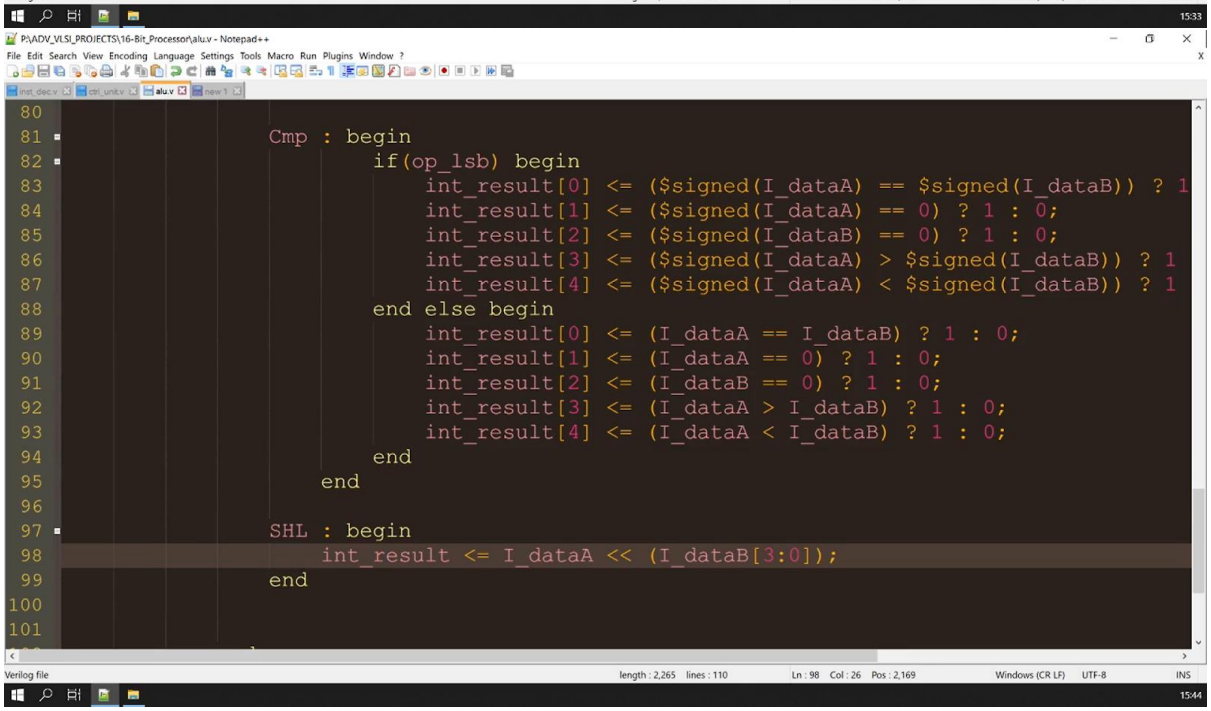
```
14      output [15:0] O_dataResult,
15
16  );
17
18      //Reg/wire Declaration
19      reg [17:0] int_result;
20      wire op_lsb;
21      reg [3:0] opcode;
22
23      // Parameter Declaration
24      localparam  Add     = 0,
25                  Sub     = 1,
26                  OR      = 2,
27                  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,
```

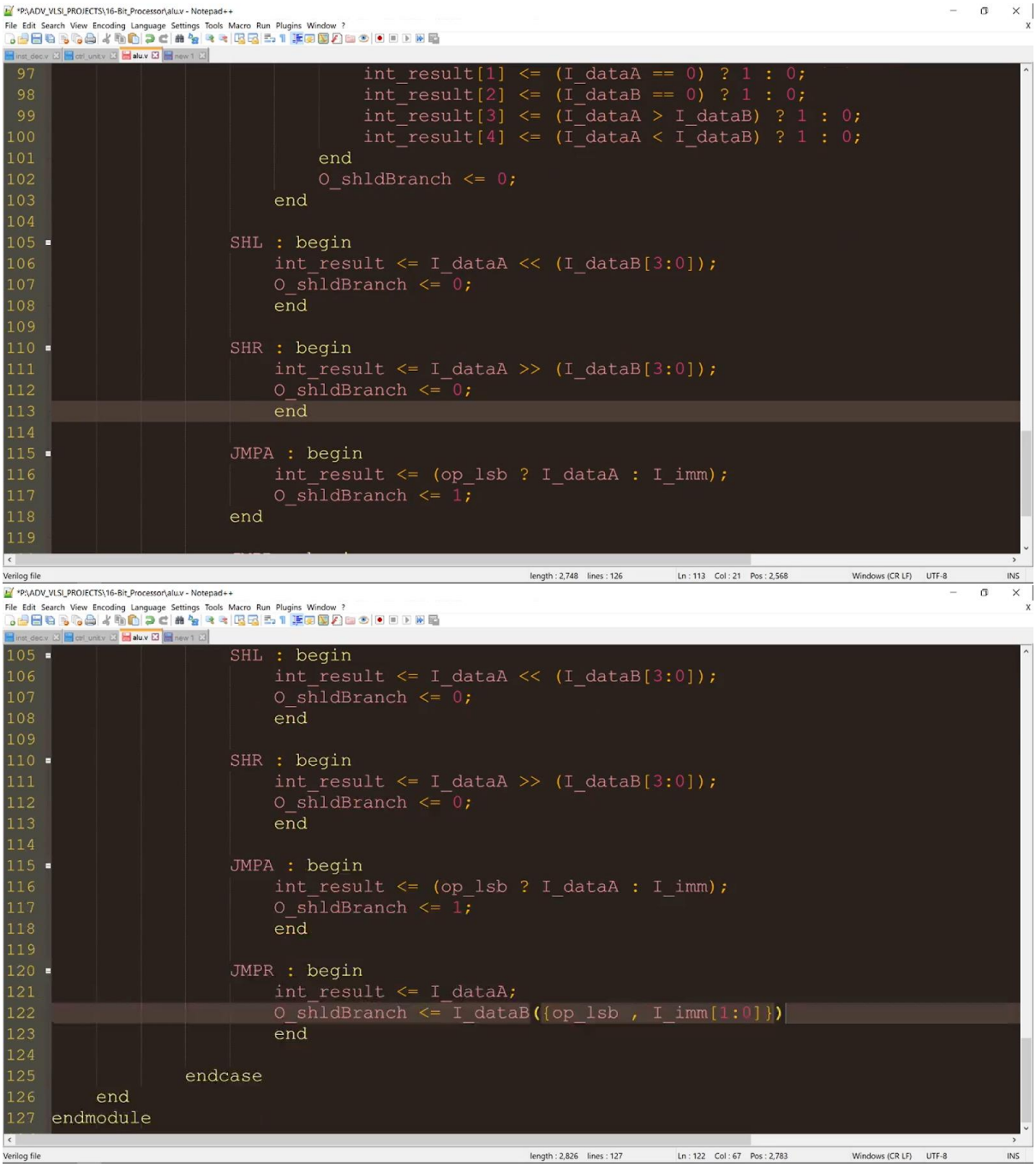


```
29                  NOT     = 5,
30                  Rdmem   = 6,
31                  Wrmem   = 7,
32                  Load    = 8,
33                  Cmp     = 9,
34                  SHL     = 10,
35                  SHR     = 11,
36                  JMPA    = 12,
37                  JMPR    = 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
```


*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
ctrl_unit.v
alu.v
new 1
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
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;
Verilog file
length : 1,292 lines : 79
Ln : 53 Col : 24 Sel : 6 | 1
Windows (CR LF)
UTF-8
INS
15:32


*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
ctrl_unit.v
alu.v
new 1
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 end
66
67 AND : begin
68 int_result <= I_dataA & I_dataB;
69 end
70
71 XOR : begin
72 int_result <= I_dataA ^ I_dataB;
73 end
74
75 NOT : begin
76 int_result <= ~I_dataA;
77 end
78
79
80
Verilog file
length : 1,437 lines : 90
Ln : 79 Col : 21 Pos : 1,385
Windows (CR LF) UTF-8 INS
15:33


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
ctrl_unit.v
alu.v
new 1
80
81 Cmp : begin
82 if(op_lsb) begin
83 int_result[0] <= ($signed(I_dataA) == $signed(I_dataB)) ? 1
84 int_result[1] <= ($signed(I_dataA) == 0) ? 1 : 0;
85 int_result[2] <= ($signed(I_dataB) == 0) ? 1 : 0;
86 int_result[3] <= ($signed(I_dataA) > $signed(I_dataB)) ? 1
87 int_result[4] <= ($signed(I_dataA) < $signed(I_dataB)) ? 1
88 end else begin
89 int_result[0] <= (I_dataA == I_dataB) ? 1 : 0;
90 int_result[1] <= (I_dataA == 0) ? 1 : 0;
91 int_result[2] <= (I_dataB == 0) ? 1 : 0;
92 int_result[3] <= (I_dataA > I_dataB) ? 1 : 0;
93 int_result[4] <= (I_dataA < I_dataB) ? 1 : 0;
94 end
95 end
96
97 SHL : begin
98 int_result <= I_dataA << (I_dataB[3:0]);
99 end
100
101
Verilog file
length : 2,265 lines : 110
Ln : 98 Col : 26 Pos : 2,169
Windows (CR LF) UTF-8 INS
15:44

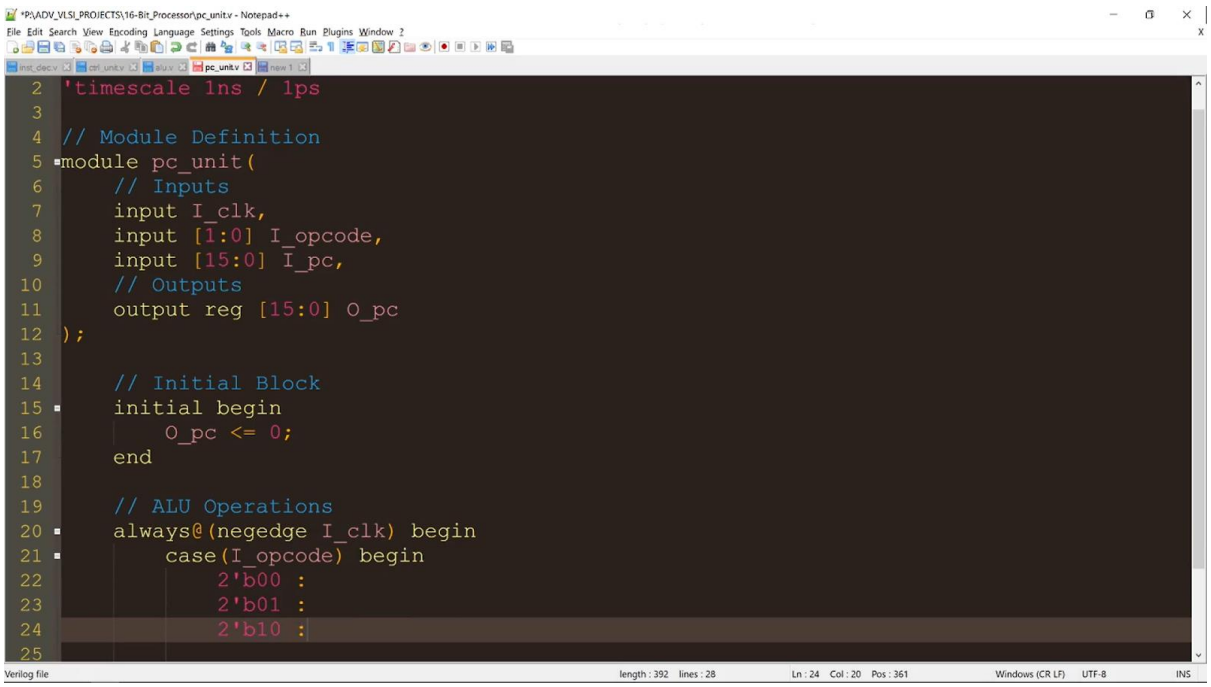


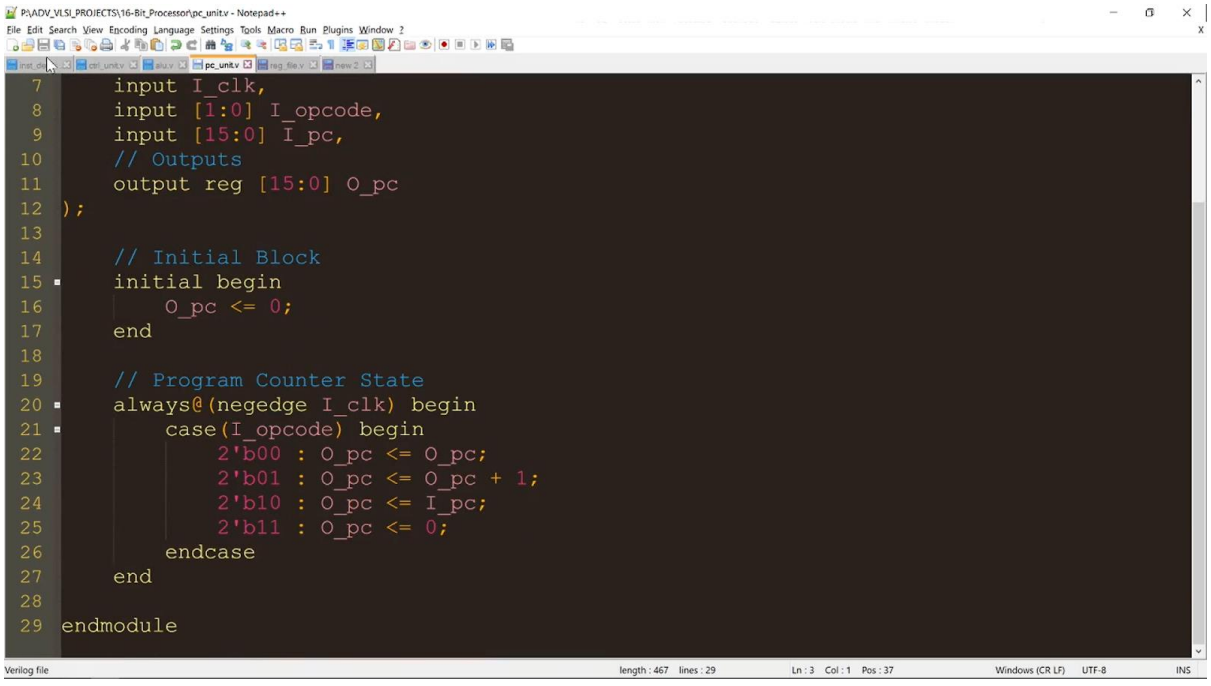
```
                                int_result[1] <= (I_dataA == 0) ? 1 : 0;
                                int_result[2] <= (I_dataB == 0) ? 1 : 0;
                                int_result[3] <= (I_dataA > I_dataB) ? 1 : 0;
                                int_result[4] <= (I_dataA < I_dataB) ? 1 : 0;
                            end
                            O_shldBranch <= 0;
                        end

                    SHL : begin
                        int_result <= I_dataA << (I_dataB[3:0]);
                        O_shldBranch <= 0;
                        end

                    SHR : begin
                        int_result <= I_dataA >> (I_dataB[3:0]);
                        O_shldBranch <= 0;
                        end

                    JMPA : begin
                        int_result <= (op_lsb ? I_dataA : I_imm);
                        O_shldBranch <= 1;
                    end

                    SHL : begin
                        int_result <= I_dataA << (I_dataB[3:0]);
                        O_shldBranch <= 0;
                        end

                    SHR : begin
                        int_result <= I_dataA >> (I_dataB[3:0]);
                        O_shldBranch <= 0;
                        end

                    JMPA : begin
                        int_result <= (op_lsb ? I_dataA : I_imm);
                        O_shldBranch <= 1;
                        end

                    JMPR : begin
                        int_result <= I_dataA;
                        O_shldBranch <= I_dataB({op_lsb , I_imm[1:0]})
                        end

                endcase
        end
    endmodule
```

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

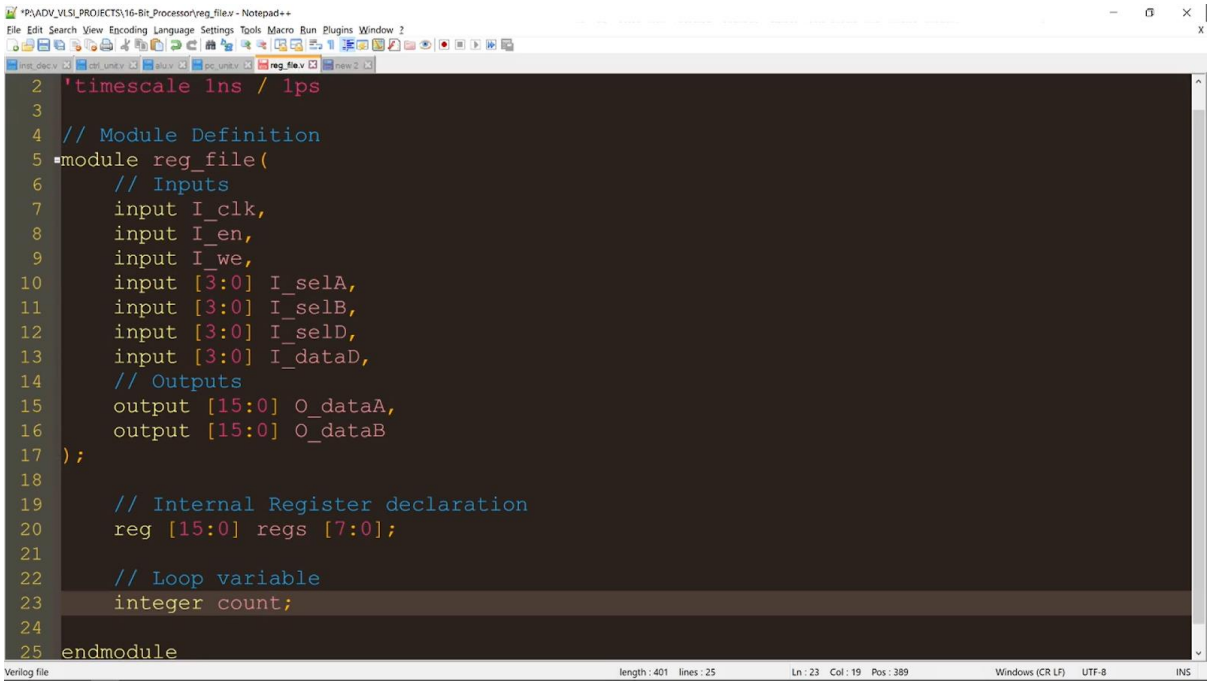
O_shldBranch <= 0;

Module pc_unit

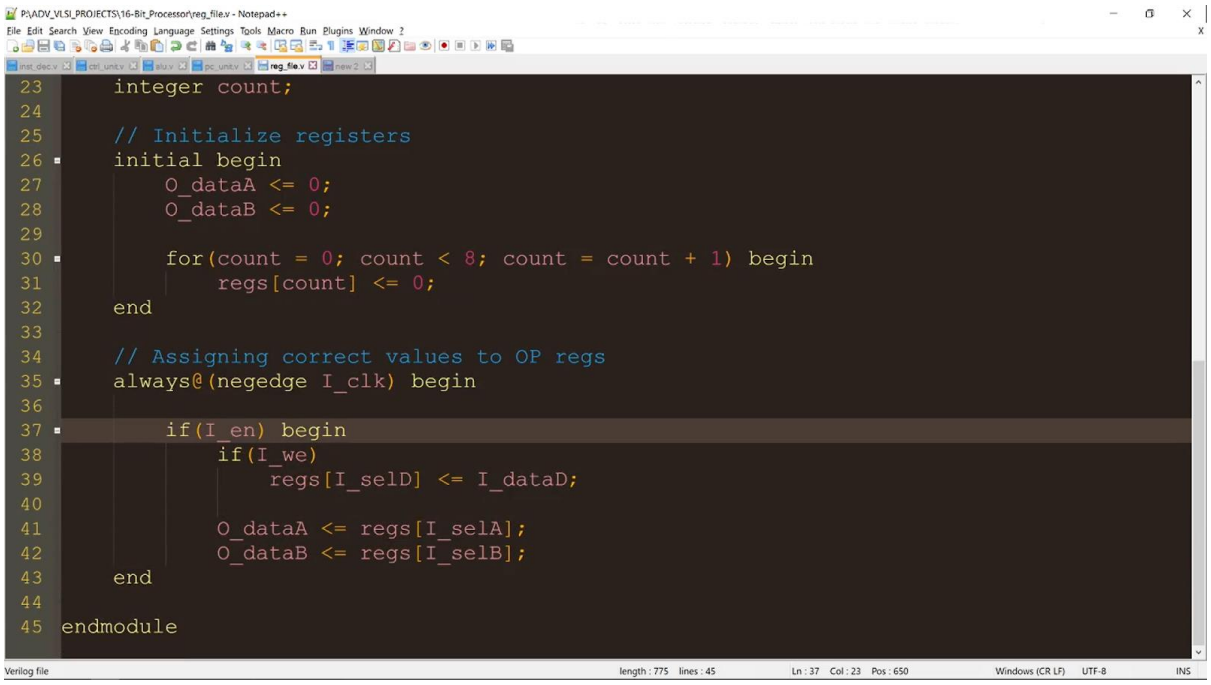




Module reg_file

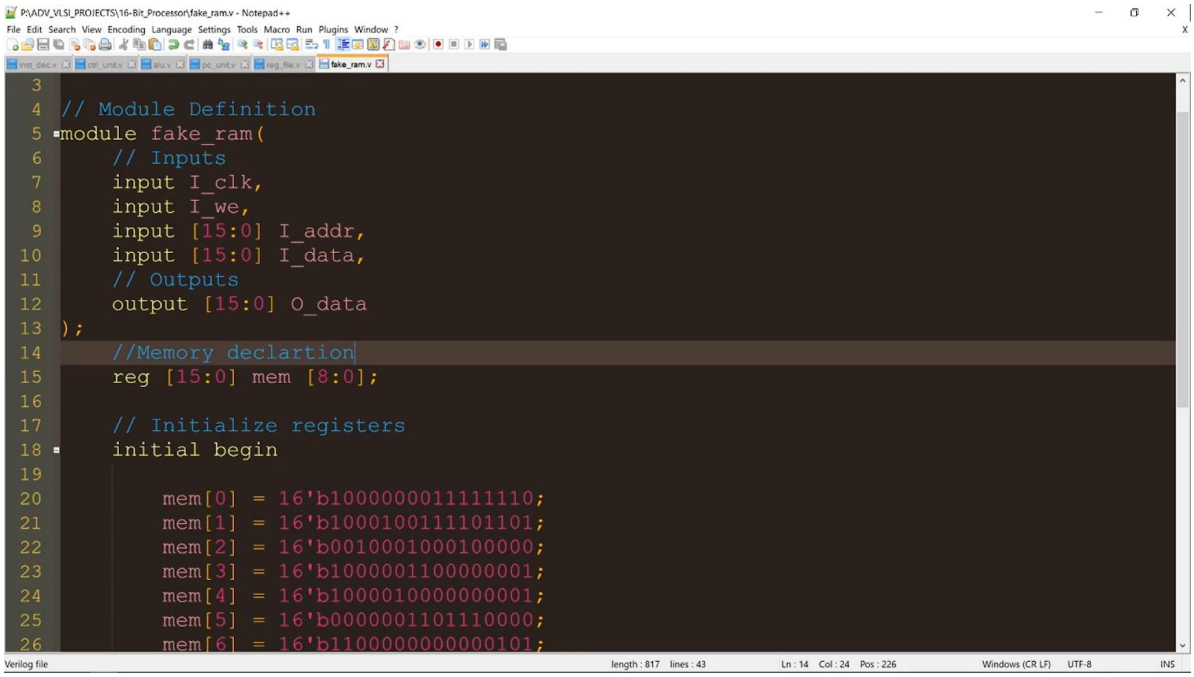


```
'timescale 1ns / 1ps

// Module Definition
module reg_file(
    // Inputs
    input I_clk,
    input I_en,
    input I_we,
    input [3:0] I_selA,
    input [3:0] I_selB,
    input [3:0] I_selD,
    input [3:0] I_dataD,
    // Outputs
    output [15:0] O_dataA,
    output [15:0] O_dataB
);

    // Internal Register declaration
    reg [15:0] regs [7:0];

    // Loop variable
    integer count;

endmodule
```

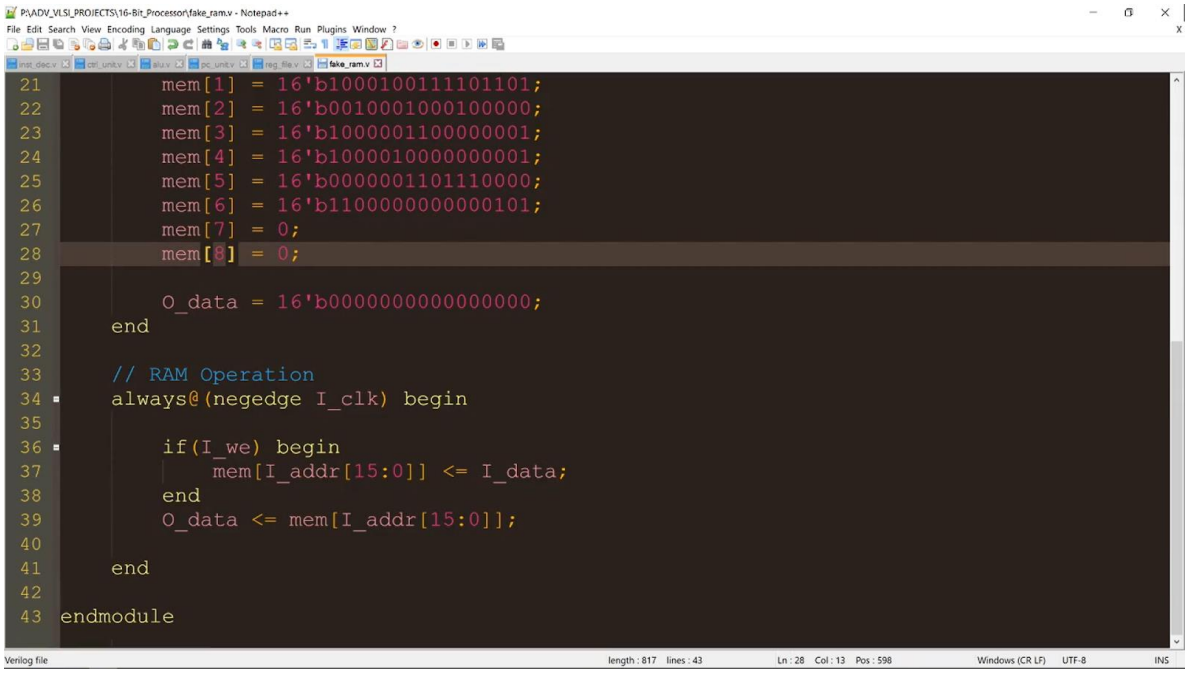


```
    integer count;

    // Initialize registers
    initial begin
        O_dataA <= 0;
        O_dataB <= 0;

        for(count = 0; count < 8; count = count + 1) begin
            regs[count] <= 0;
    end

    // Assigning correct values to OP regs
    always@(negedge I_clk) begin

        if(I_en) begin
            if(I_we)
                regs[I_selD] <= I_dataD;

            O_dataA <= regs[I_selA];
            O_dataB <= regs[I_selB];
    end

endmodule
```

Module fake_ram



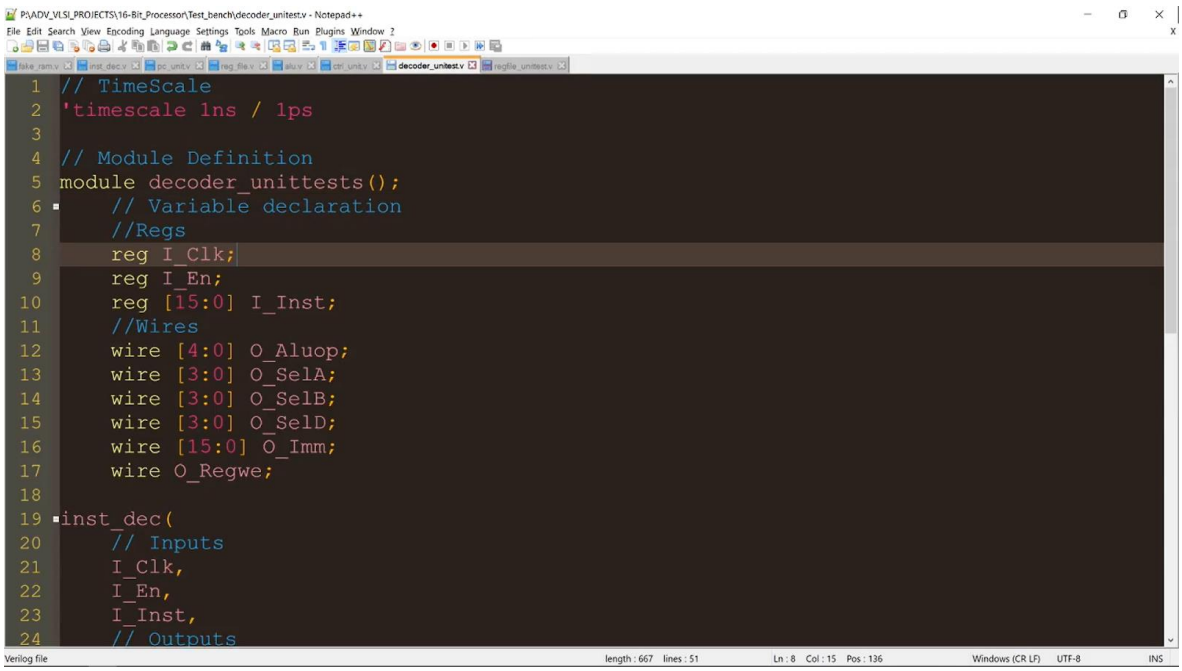
```
// Module Definition
module fake_ram(
    // Inputs
    input I_clk,
    input I_we,
    input [15:0] I_addr,
    input [15:0] I_data,
    // Outputs
    output [15:0] O_data
);
    //Memory declartion
    reg [15:0] mem [8:0];

    // Initialize registers
    initial begin

        mem[0] = 16'b1000000011111110;
        mem[1] = 16'b1000100111101101;
        mem[2] = 16'b0010001000100000;
        mem[3] = 16'b1000001100000001;
        mem[4] = 16'b1000010000000001;
        mem[5] = 16'b0000001101110000;
        mem[6] = 16'b1100000000000101;
```



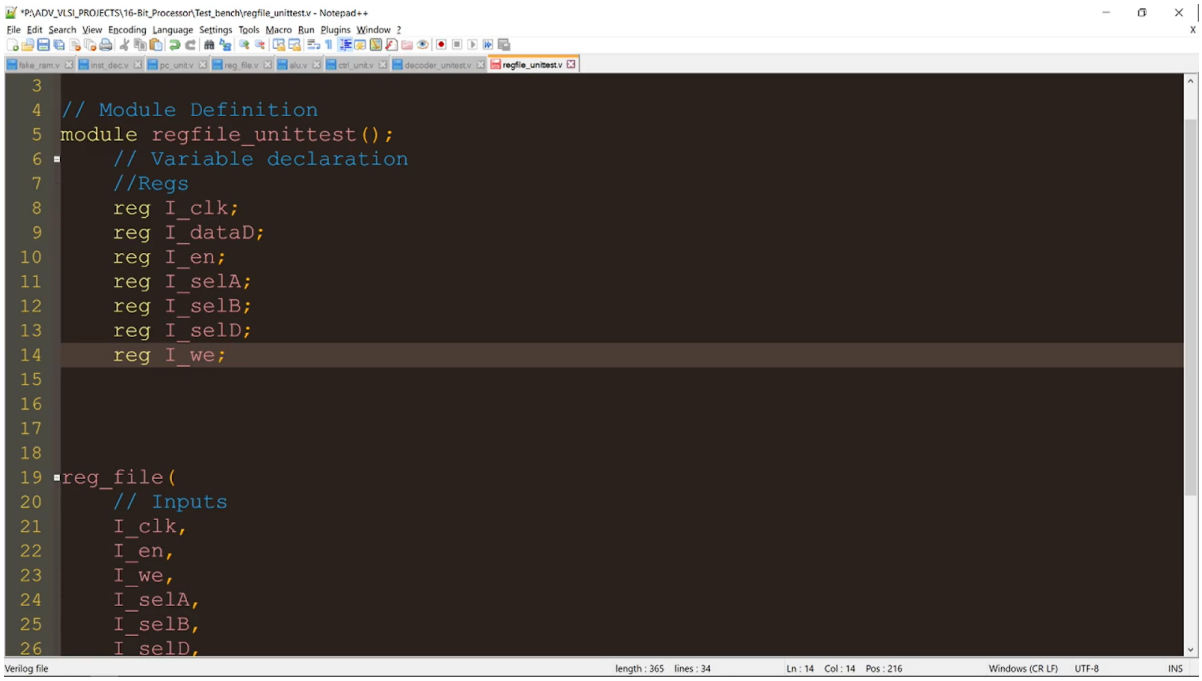
```
        mem[1] = 16'b1000100111101101;
        mem[2] = 16'b0010001000100000;
        mem[3] = 16'b1000001100000001;
        mem[4] = 16'b1000010000000001;
        mem[5] = 16'b0000001101110000;
        mem[6] = 16'b1100000000000101;
        mem[7] = 0;
        mem[8] = 0;

        O_data = 16'b0000000000000000;
    end

    // RAM Operation
    always@(negedge I_clk) begin

        if(I_we) begin
            mem[I_addr[15:0]] <= I_data;
        end
        O_data <= mem[I_addr[15:0]];

    end

endmodule
```

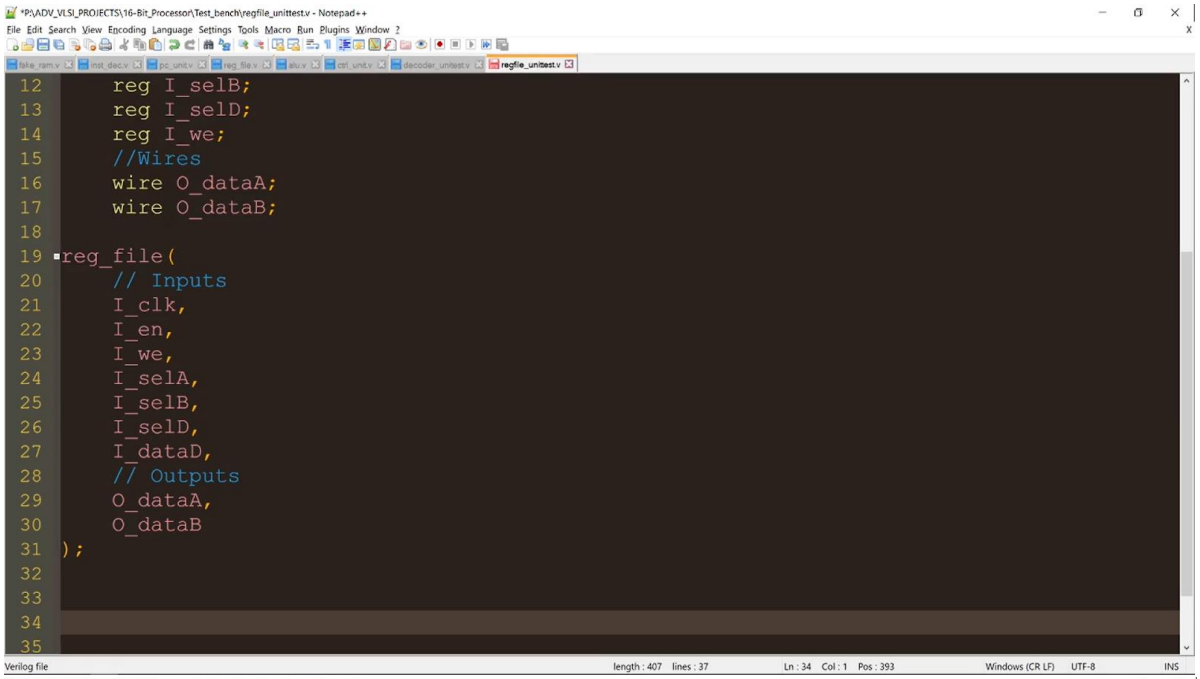
TESTING CODES

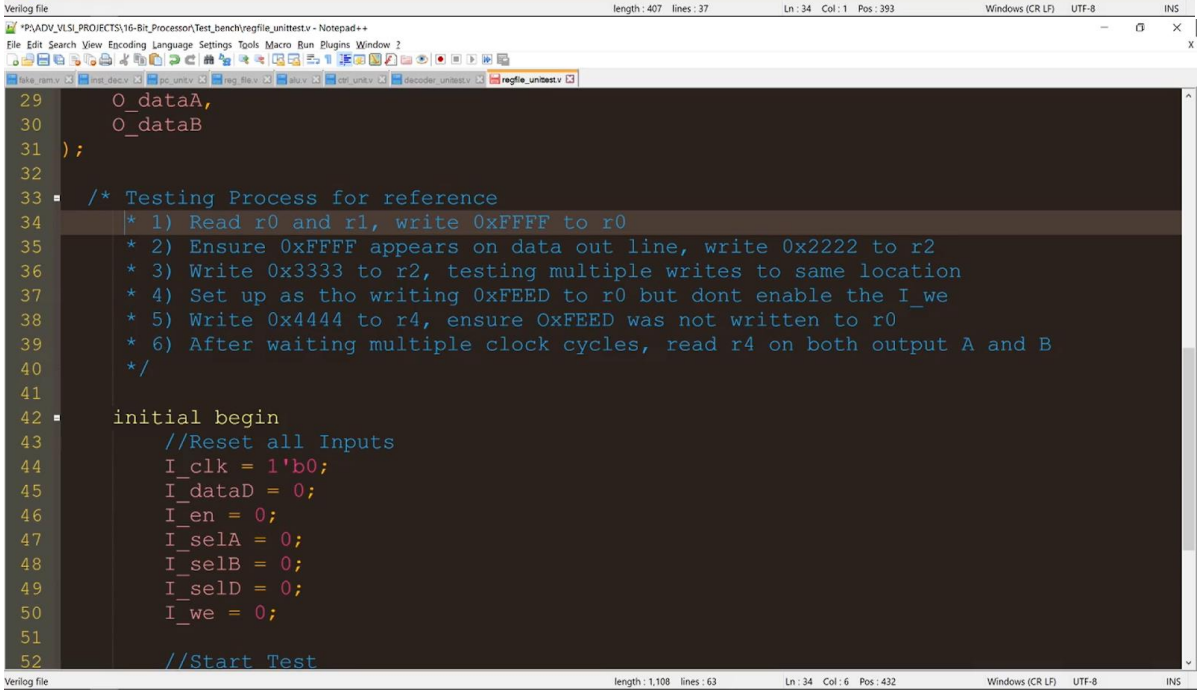
Module decoder unit_test

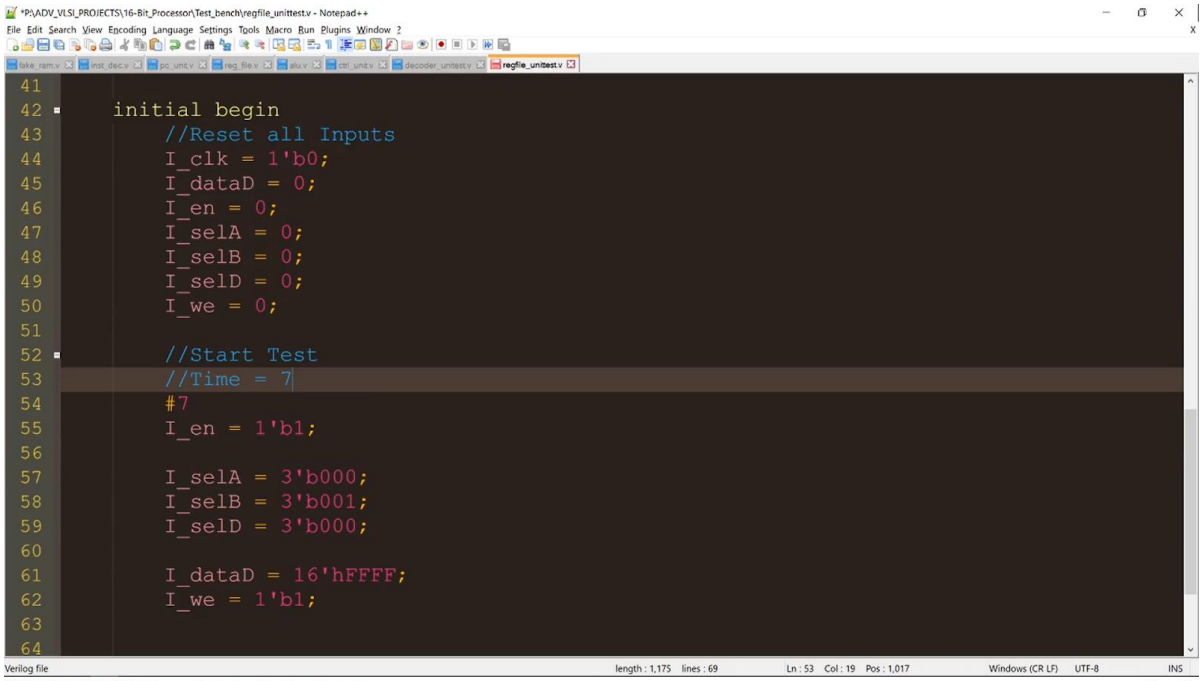


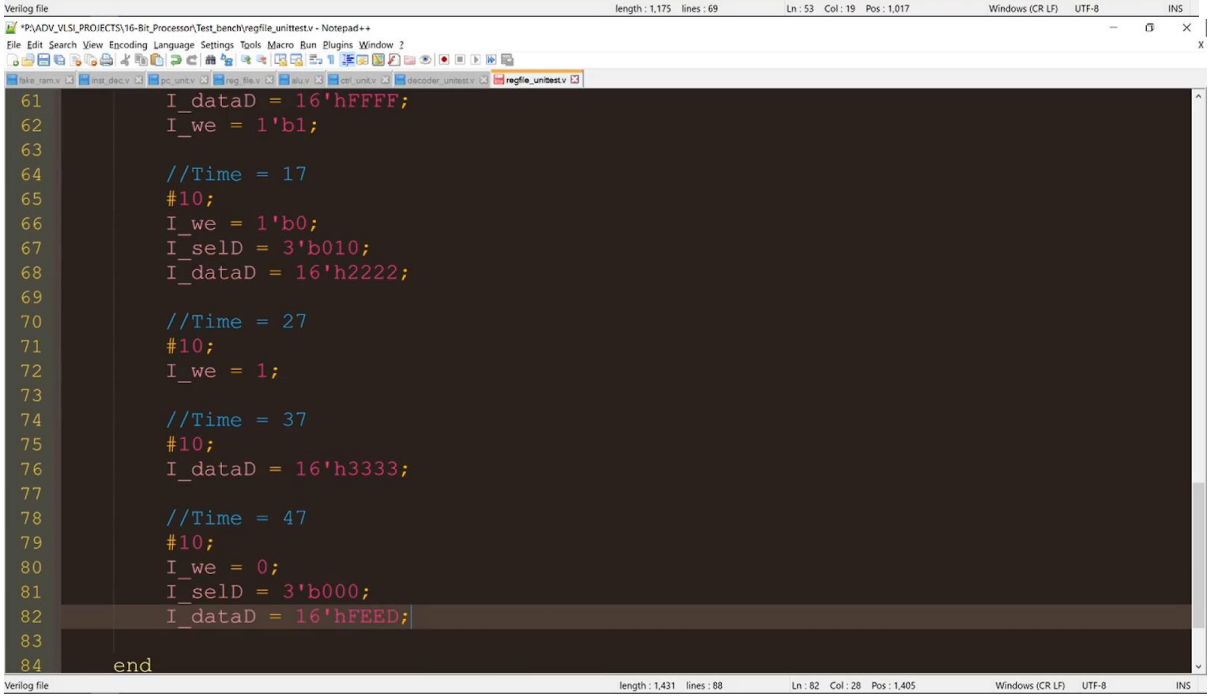
Module regfile unit_test

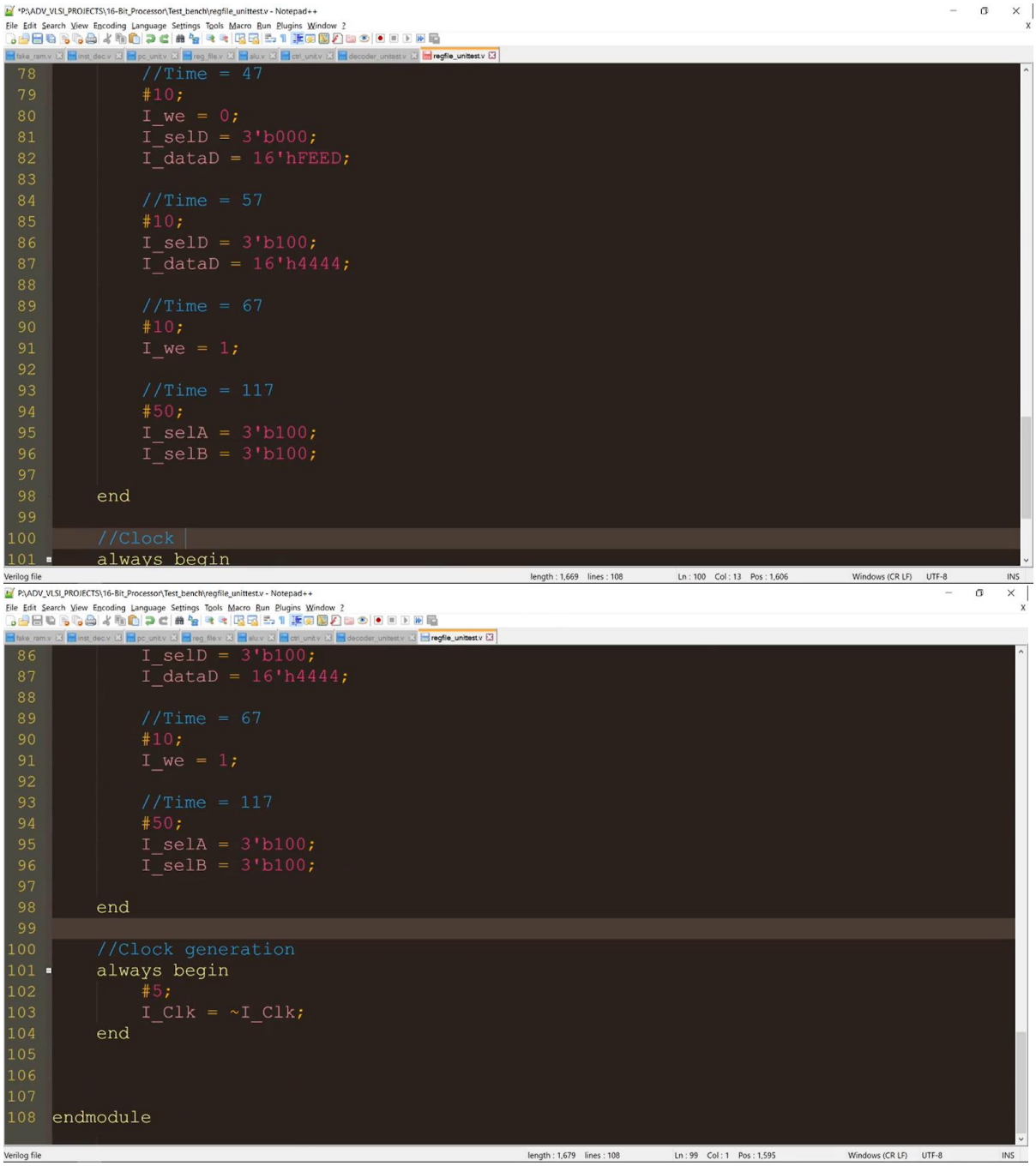



*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\regfile_unittest.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
fake_ram.v inst_dec.v pc_unit.v reg_file.v alu.v ctrl_unit.v decoder_unittest.v regfile_unittest.v
reg I_selB;
reg I_selD;
reg I_we;
//Wires
wire O_dataA;
wire O_dataB;
reg_file(
// Inputs
I_clk,
I_en,
I_we,
I_selA,
I_selB,
I_selD,
I_dataD,
// Outputs
O_dataA,
O_dataB
);
Verilog file
length : 407 lines : 37
Ln : 34 Col : 1 Pos : 393
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 ?
fake_ram.v inst_dec.v pc_unit.v reg_file.v alu.v ctrl_unit.v decoder_unittest.v regfile_unittest.v
O_dataA,
O_dataB
);
/* Testing Process for reference
* 1) Read r0 and r1, write 0xFFFF to r0
* 2) Ensure 0xFFFF appears on data out line, write 0x2222 to r2
* 3) Write 0x3333 to r2, testing multiple writes to same location
* 4) Set up as tho writing 0xFEED to r0 but dont enable the I_we
* 5) Write 0x4444 to r4, ensure 0xFEED was not written to r0
* 6) After waiting multiple clock cycles, read r4 on both output A and B
*/
initial begin
//Reset all Inputs
I_clk = 1'b0;
I_dataD = 0;
I_en = 0;
I_selA = 0;
I_selB = 0;
I_selD = 0;
I_we = 0;
//Start Test
Verilog file
length : 1,108 lines : 63
Ln : 34 Col : 6 Pos : 432
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 ?
fake_ram.v inst_dec.v pc_unit.v reg_file.v alu.v ctrl_unit.v decoder_unittest.v regfile_unittest.v
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
Verilog file
length : 1,175 lines : 69
Ln : 53 Col : 19 Pos : 1,017
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 ?
fake_ram.v inst_dec.v pc_unit.v reg_file.v alu.v ctrl_unit.v decoder_unittest.v regfile_unittest.v
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

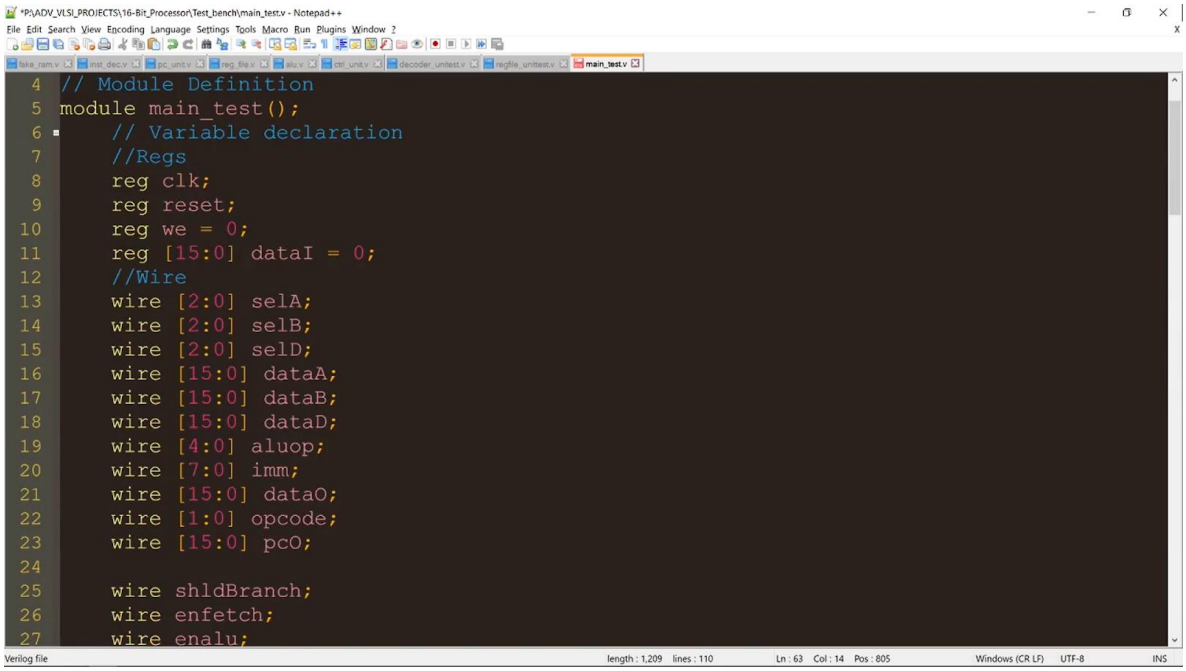


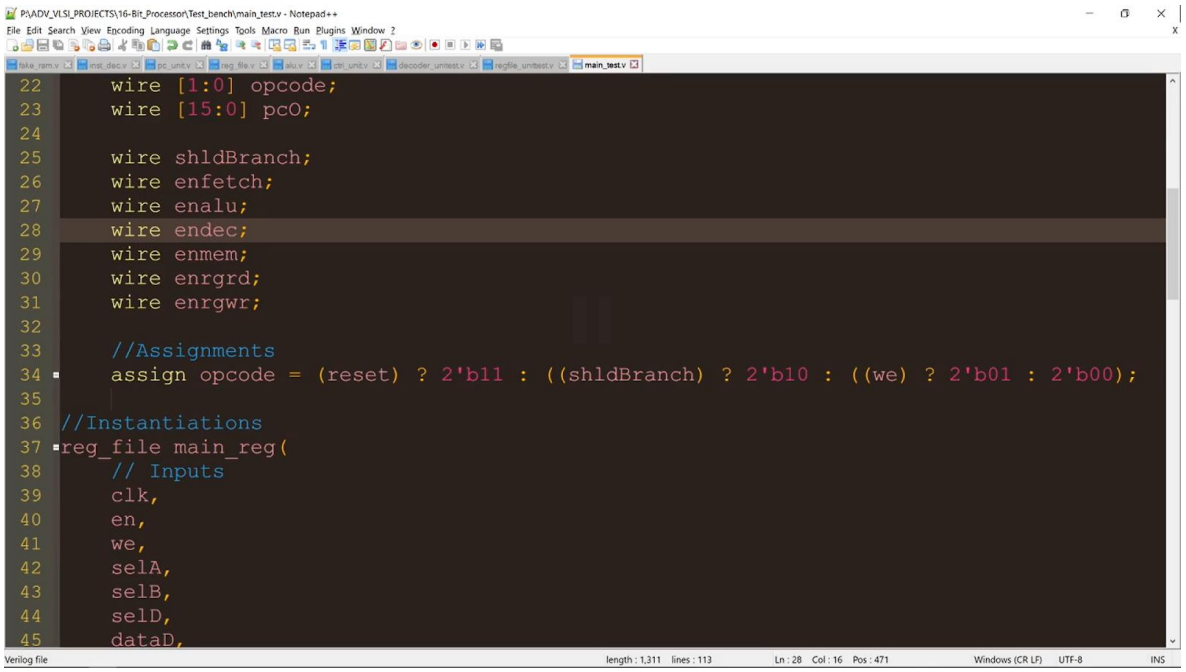
```
        //Time = 47
        #10;
        I_we = 0;
        I_selD = 3'b000;
        I_dataD = 16'hFEED;

        //Time = 57
        #10;
        I_selD = 3'b100;
        I_dataD = 16'h4444;

        //Time = 67
        #10;
        I_we = 1;

        //Time = 117
        #50;
        I_selA = 3'b100;
        I_selB = 3'b100;

    end

    //Clock
    always begin
```

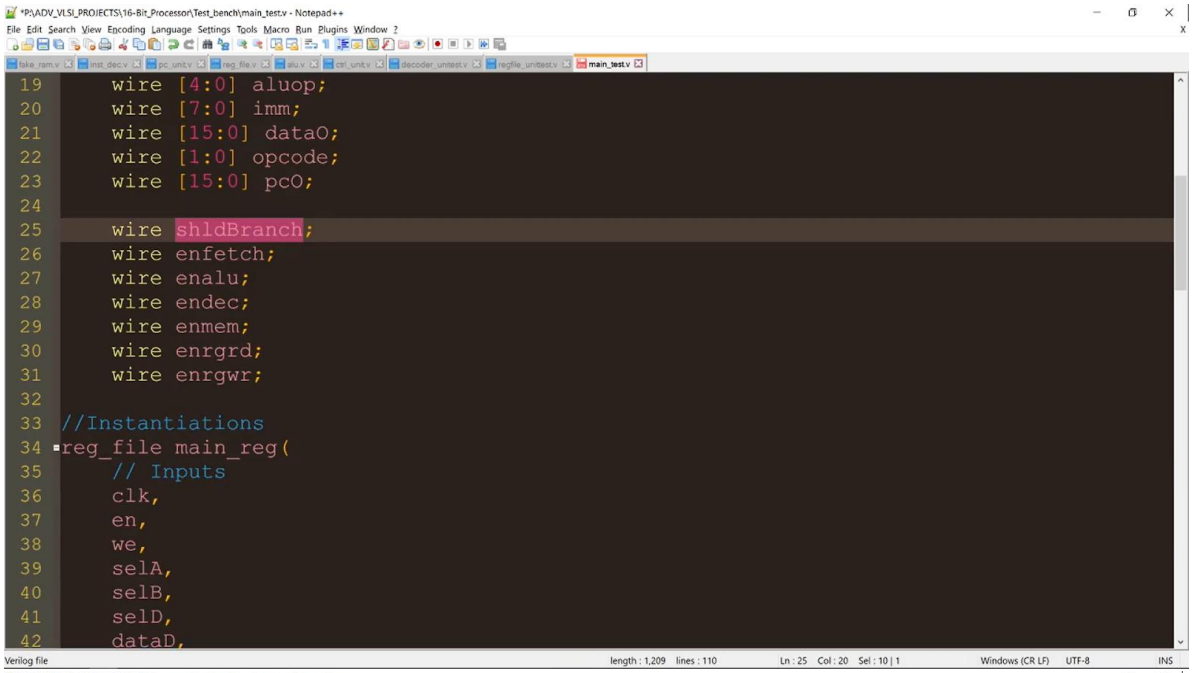


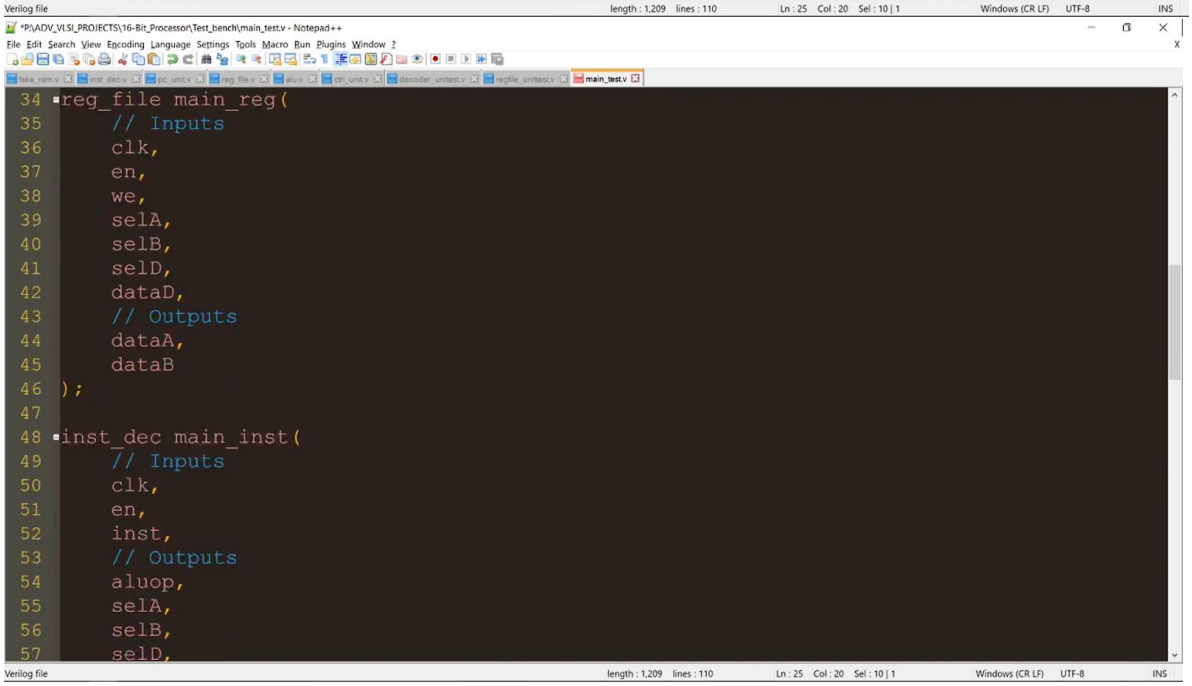
```
        I_selD = 3'b100;
        I_dataD = 16'h4444;

        //Time = 67
        #10;
        I_we = 1;

        //Time = 117
        #50;
        I_selA = 3'b100;
        I_selB = 3'b100;

    end

    //Clock generation
    always begin
        #5;
        I_Clk = ~I_Clk;
    end


endmodule
```

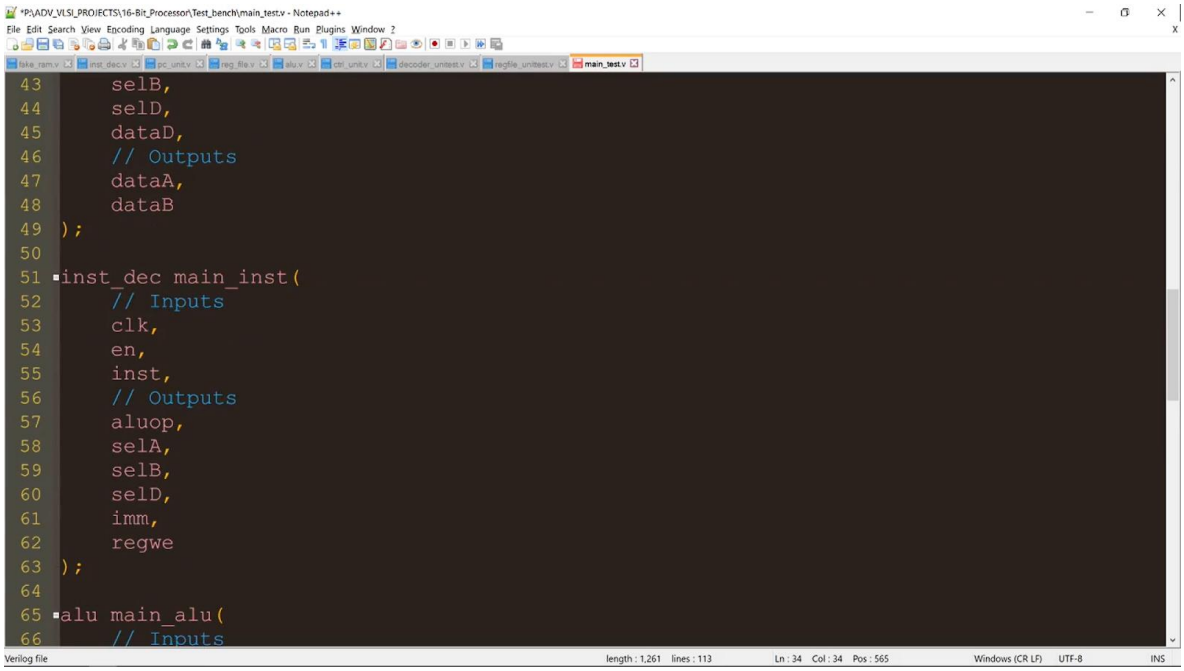
Module regfile main_test

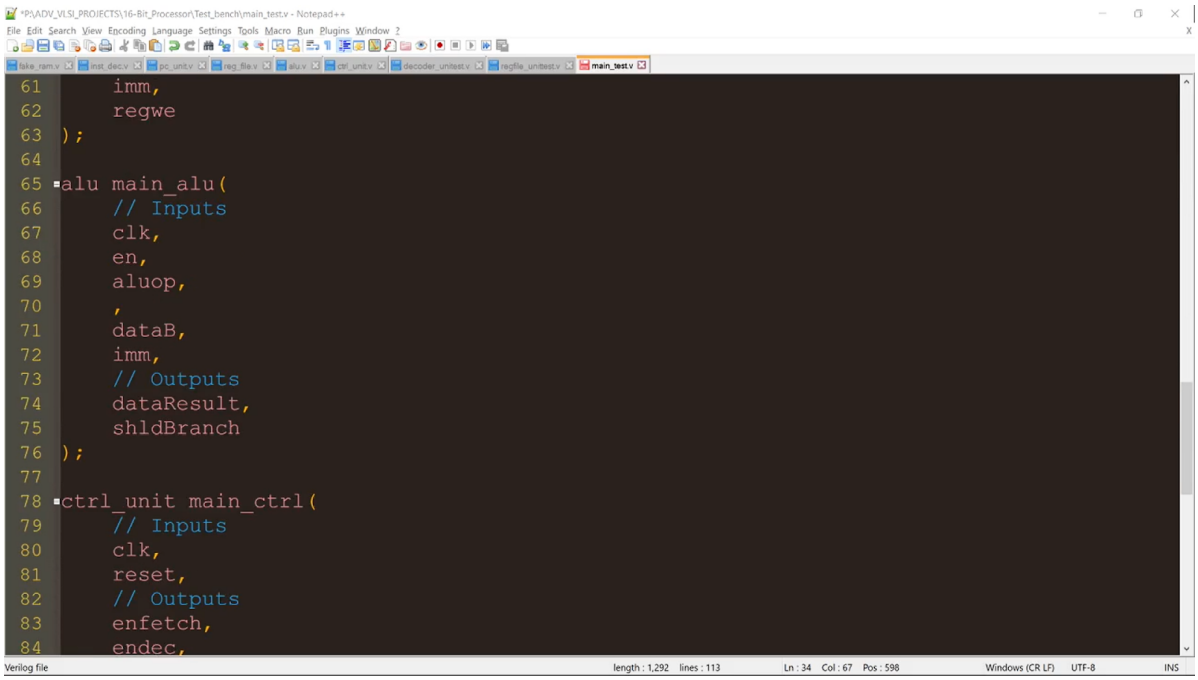

*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_test.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
fake_ram.v inst_dec.v pc_unit.v reg_file.v alu.v ctrl_unit.v decoder_unittest.v regfile_unittest.v main_test.v
4 // Module Definition
5 module main_test();
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;
Verilog file
length : 1,209 lines : 110
Ln : 63 Col : 14 Pos : 805
Windows (CR LF) UTF-8 INS

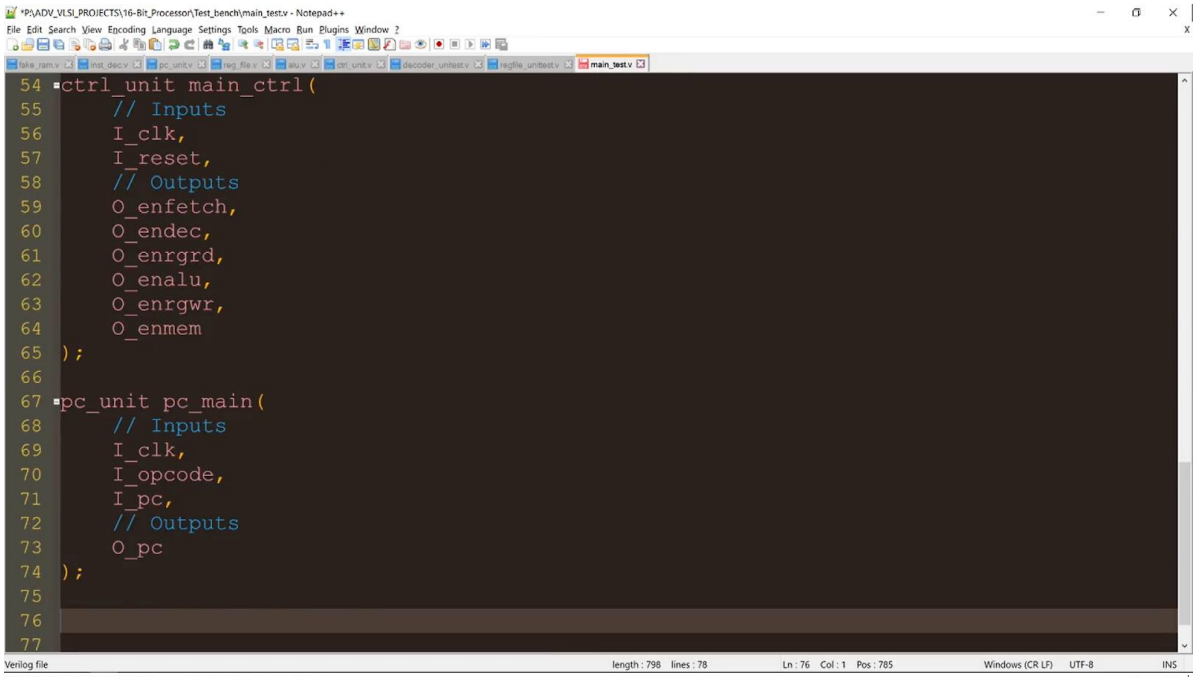

P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_test.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
fake_ram.v inst_dec.v pc_unit.v reg_file.v alu.v ctrl_unit.v decoder_unittest.v regfile_unittest.v main_test.v
22 wire [1:0] opcode;
23 wire [15:0] pcO;
24
25 wire shldBranch;
26 wire enfetch;
27 wire enalu;
28 wire endec;
29 wire enmem;
30 wire enrgrd;
31 wire enrgwr;
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 : 1,311 lines : 113
Ln : 28 Col : 16 Pos : 471
Windows (CR LF) UTF-8 INS

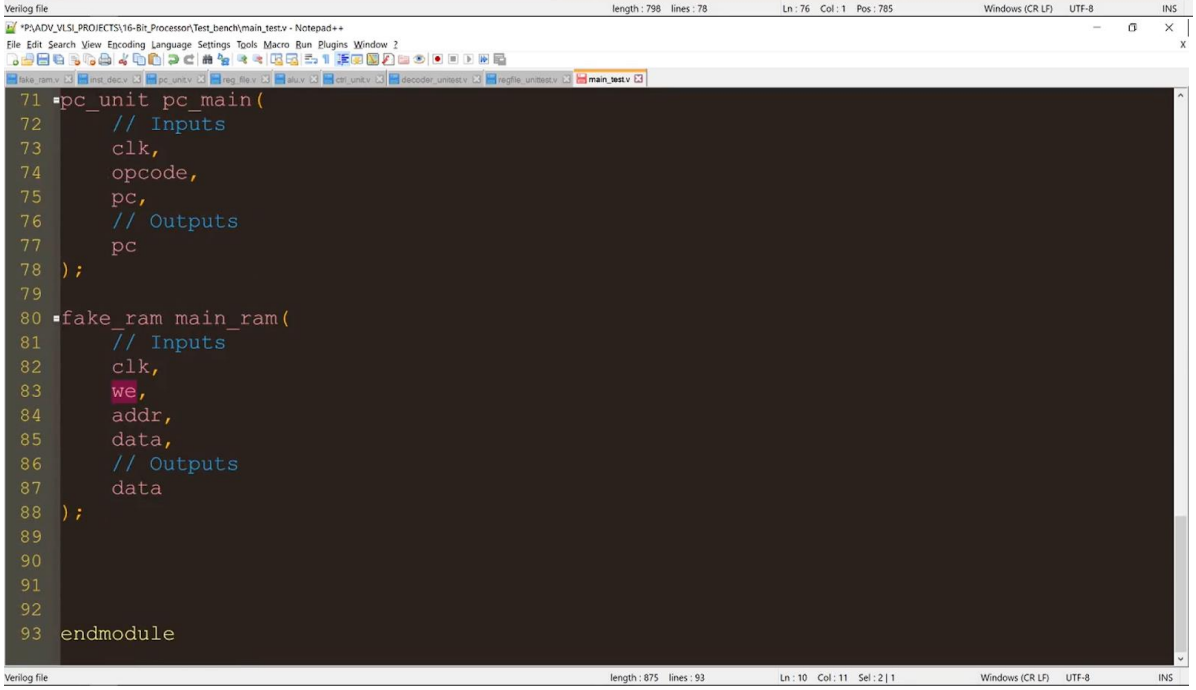

*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_test.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
fake_ram.v inst_dec.v pc_unit.v reg_file.v alu.v ctrl_unit.v decoder_unitest.v regfile_unitest.v main_test.v
19 wire [4:0] aluop;
20 wire [7:0] imm;
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;
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


*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_test.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
fake_ram.v inst_dec.v pc_unit.v reg_file.v alu.v ctrl_unit.v decoder_unitest.v regfile_unitest.v main_test.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,
Verilog file
length : 1,209 lines : 110
Ln : 25 Col : 20 Sel : 10 | 1
Windows (CR LF) UTF-8 INS


*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_test.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
fake_ram.v inst_dec.v pc_unit.v reg_file.v alu.v ctrl_unit.v decoder_unitest.v regfile_unitest.v main_test.v
selB,
selD,
dataD,
// Outputs
dataA,
dataB
);
inst_dec main_inst(
// Inputs
clk,
en,
inst,
// Outputs
aluop,
selA,
selB,
selD,
imm,
regwe
);
alu main_alu(
// 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_test.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
fake_ram.v inst_dec.v pc_unit.v reg_file.v alu.v ctrl_unit.v decoder_unitest.v regfile_unitest.v main_test.v
imm,
regwe
);
alu main_alu(
// Inputs
clk,
en,
aluop,
,
dataB,
imm,
// Outputs
dataResult,
shldBranch
);
ctrl_unit main_ctrl(
// Inputs
clk,
reset,
// Outputs
enfetch,
endec,
Verilog file
length : 1,292 lines : 113
Ln : 34 Col : 67 Pos : 598
Windows (CR LF) UTF-8 INS


*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_test.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
fake_ram.v inst_dec.v pc_unit.v reg_file.v alu.v ctrl_unit.v decoder_unittest.v regfile_unittest.v main_test.v
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

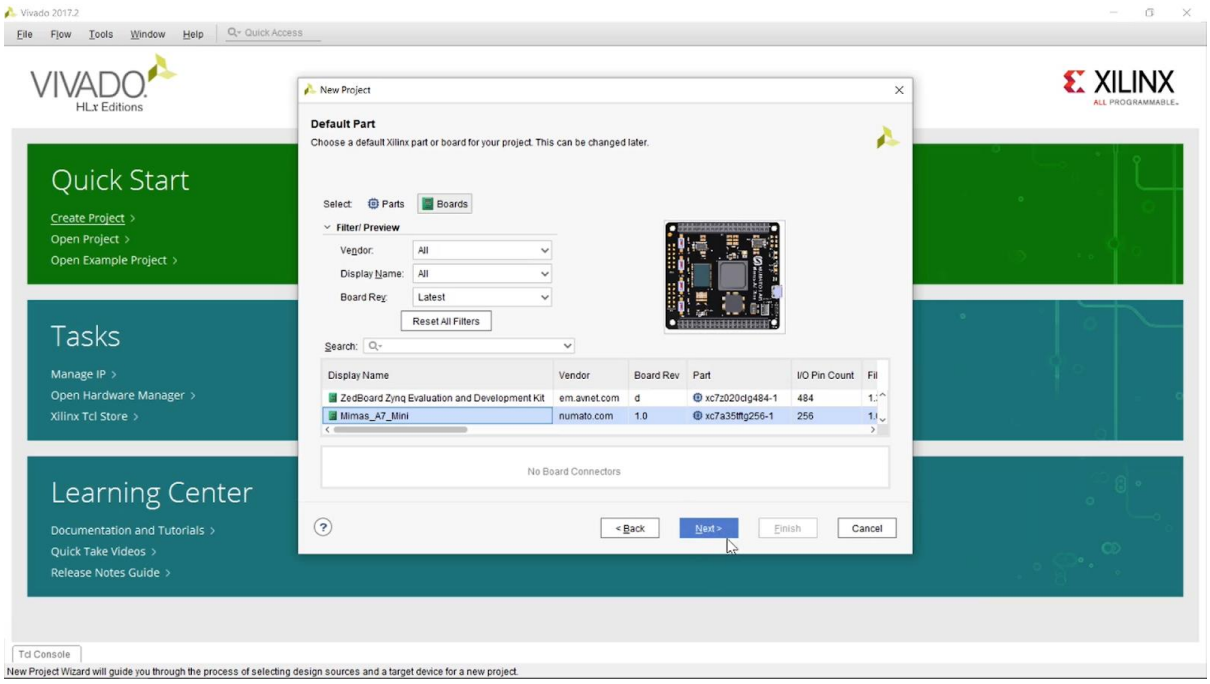

*P:\ADV_VLSI_PROJECTS\16-Bit_Processor\Test_bench\main_test.v - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
fake_ram.v inst_dec.v pc_unit.v reg_file.v alu.v ctrl_unit.v decoder_unittest.v regfile_unittest.v main_test.v
71 pc_unit pc_main(
72 // Inputs
73 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

```
    addr,
    dataI,
    // Outputs
    dataO
);

    initial begin

        clk = 0;
        reset = 1;
        #20
        reset = 0;

    end

    //Clock generation
    always begin
        #5;
        I_Clk = ~I_Clk;
    end


endmodule
```

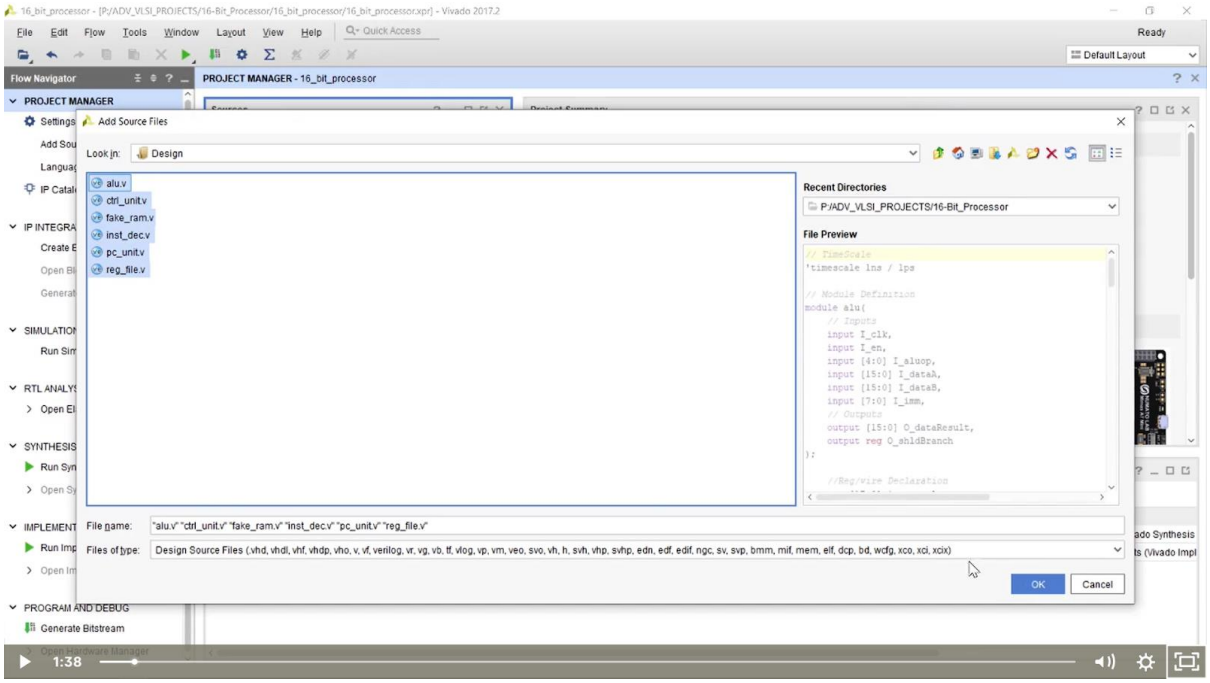
Time to simulate!

Create a project with a name.

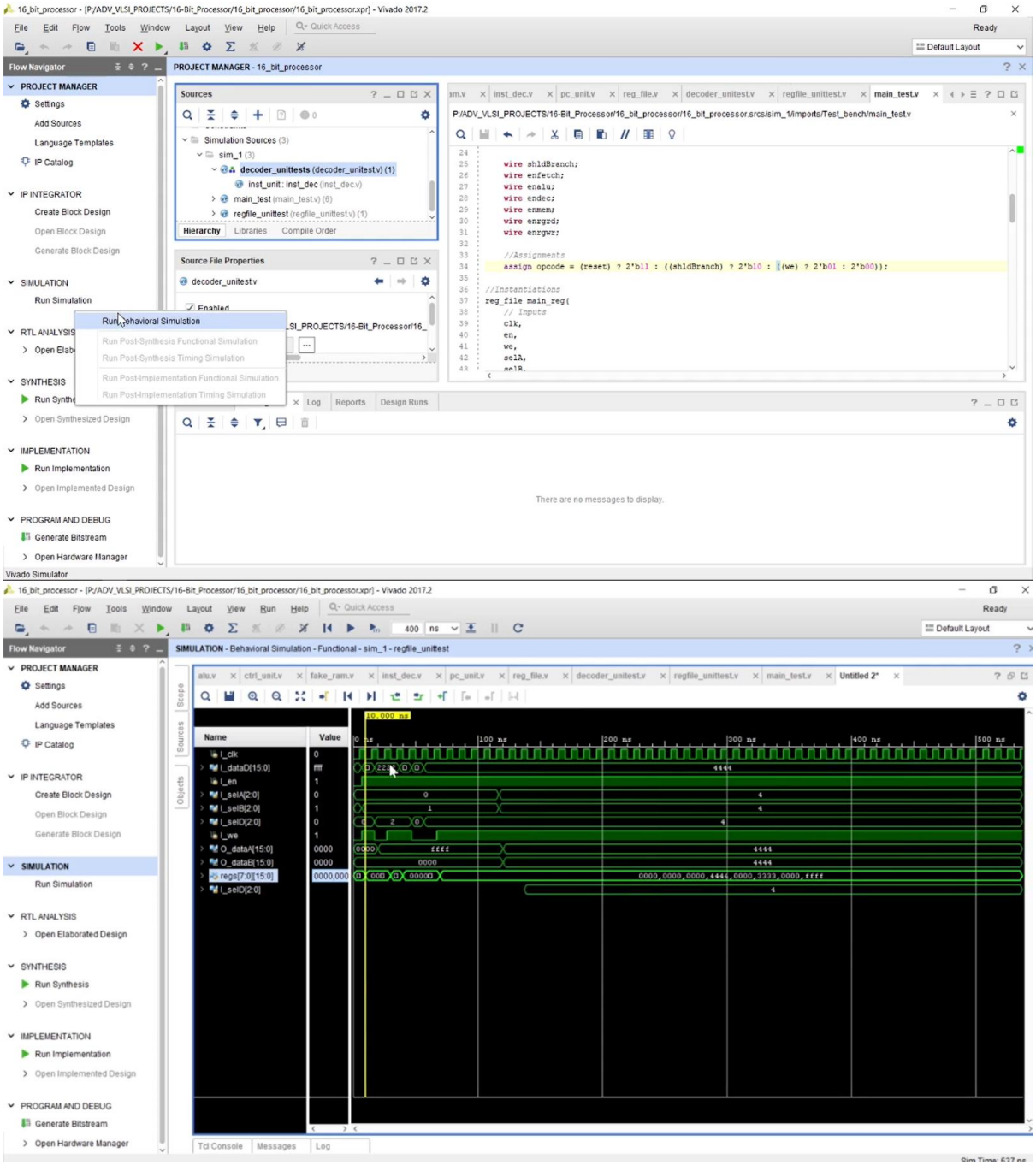
Chose the board, Mimas_A7_Mini



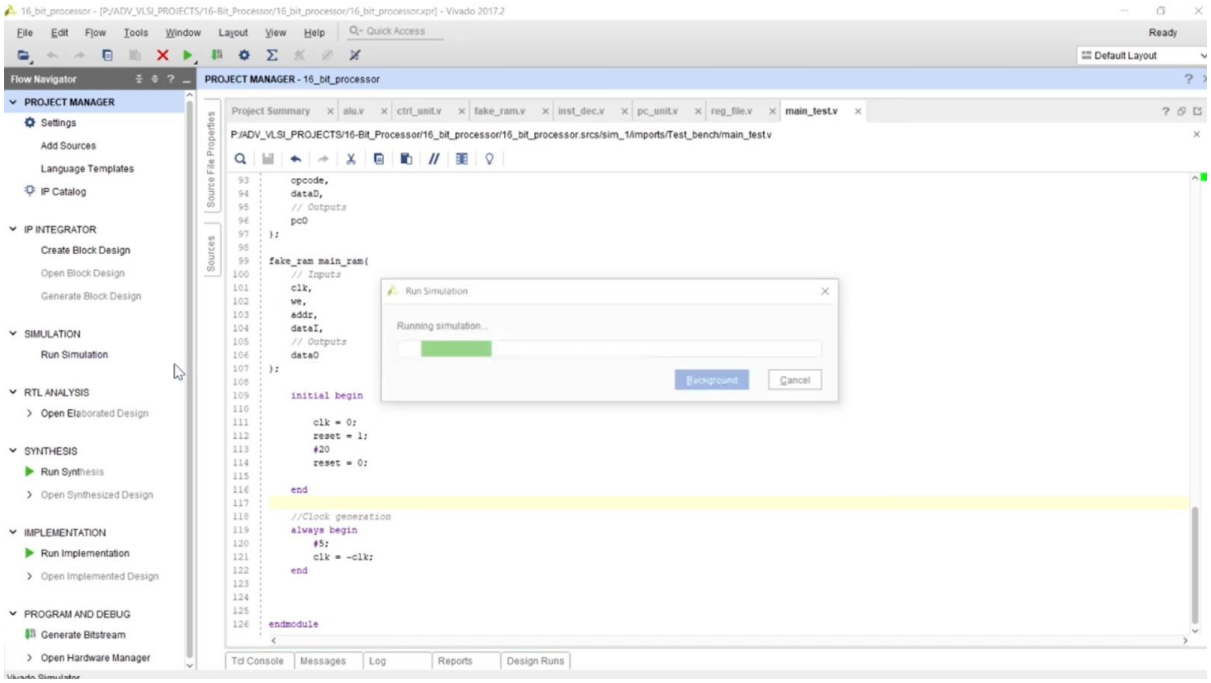
Import all the verilog files

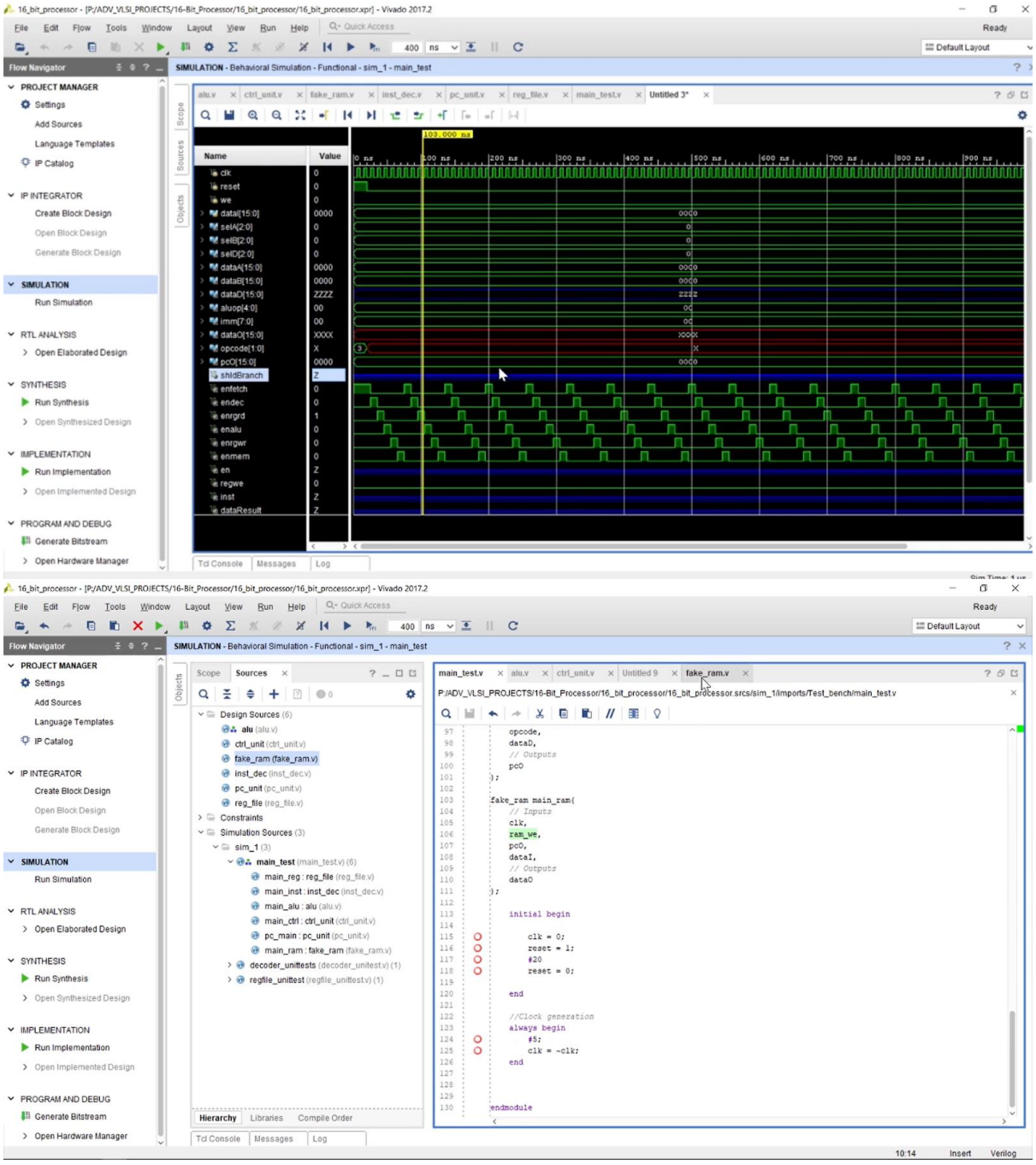


After successfully compiling the errors, run Behavorial simulation model of our instruction decoder.

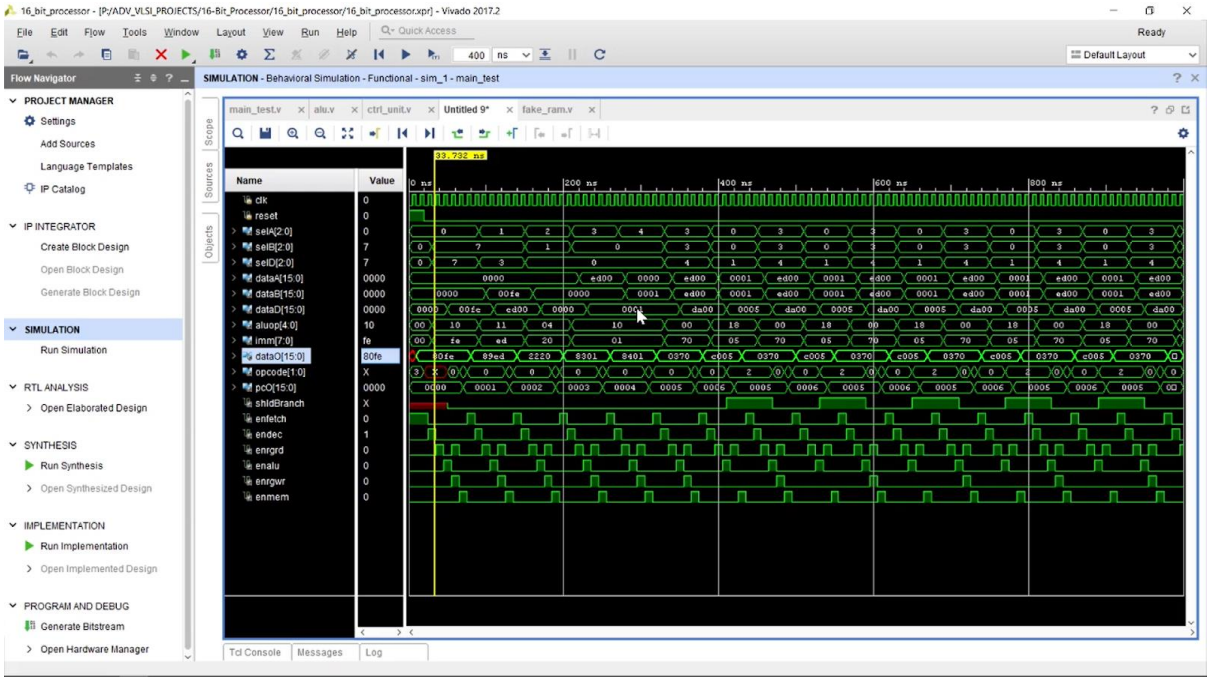


Run the main test bench simulation

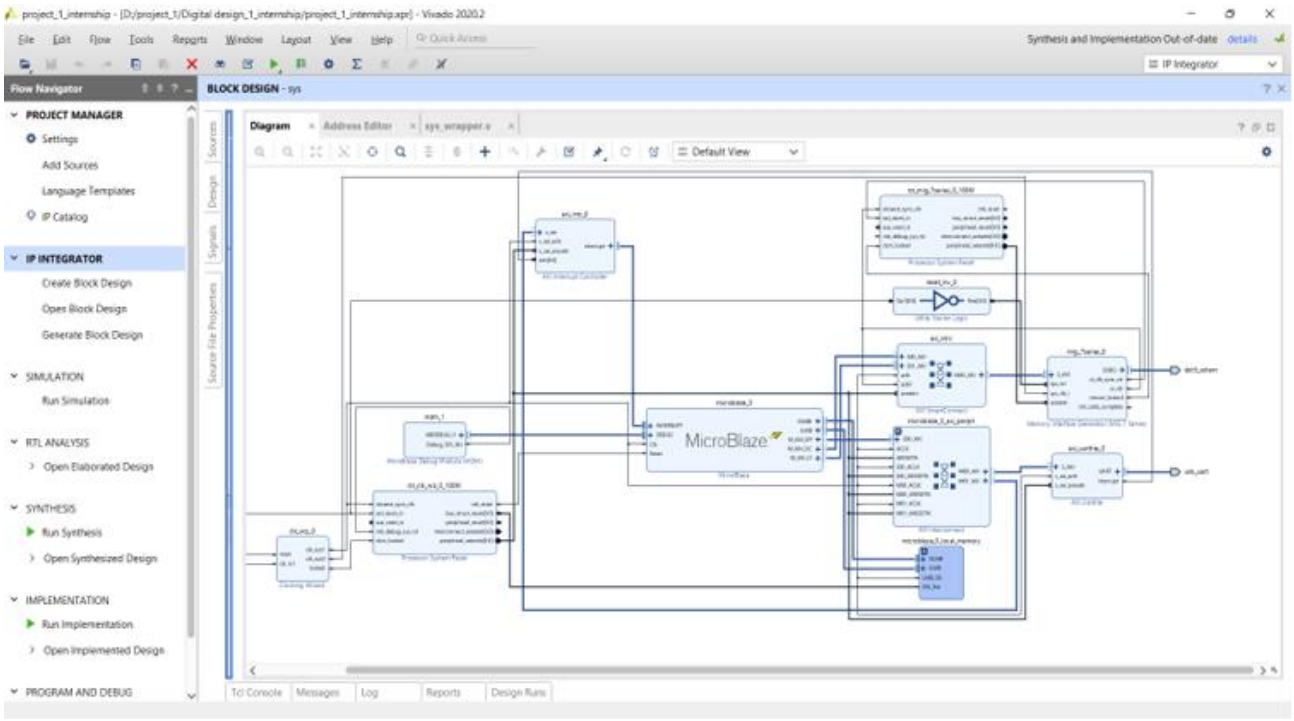

16_bit_processor - [P:/ADV_VLSI_PROJECTS/16-Bit_Processor/16_bit_processor/16_bit_processor.xpr] - Vivado 2017.2
File Edit Flow Tools Window Layout View Help
Quick Access
Ready
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
PROJECT MANAGER - 16_bit_processor
Project Summary
alu.v
ctrl_unit.v
fake_ram.v
inst_dec.v
pc_unit.v
reg_file.v
main_test.v
P:/ADV_VLSI_PROJECTS/16-Bit_Processor/16_bit_processor.srcs/sim_1/imports/Test_bench/main_test.v
Source File Properties
Sources
93 opcode,
94 dataD,
95 // Outputs
96 pc0
97);
98
99 fake_ram main_ram(
100 // Inputs
101 clk,
102 we,
103 addr,
104 dataI,
105 // Outputs
106 data0
107);
108
109 initial begin
110
111 clk = 0;
112 reset = 1;
113 #20
114 reset = 0;
115
116 end
117
118 //Clock generation
119 always begin
120 #5;
121 clk = ~clk;
122 end
123
124
125
126 endmodule
Run Simulation
Running simulation...
Background
Cancel
Tcl Console
Messages
Log
Reports
Design Runs


16_bit_processor - [P:/ADV_VLSI_PROJECTS/16-Bit_Processor/16_bit_processor/16_bit_processor.xpr] - Vivado 2017.2
File Edit Flow Tools Window Layout View Run Help
Quick Access
Ready
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
SIMULATION - Behavioral Simulation - Functional - sim_1 - main_test
alu.v ctrl_unit.v fake_ram.v inst_dec.v pc_unit.v reg_file.v main_test.v Untitled 3*
Name
Value
clk
reset
we
dataI[15:0]
selA[2:0]
selB[2:0]
selD[2:0]
dataA[15:0]
dataB[15:0]
dataD[15:0]
aluop[4:0]
imm[7:0]
dataO[15:0]
opcode[1:0]
pcO[15:0]
shldBranch
enfetch
endec
enrgrd
enalu
enrgwr
enmem
en
regwe
inst
dataResult
Tcl Console Messages Log

16_bit_processor - [P:/ADV_VLSI_PROJECTS/16-Bit_Processor/16_bit_processor/16_bit_processor.xpr] - Vivado 2017.2
File Edit Flow Tools Window Layout View Run Help
Quick Access
Ready
Default Layout
Flow Navigator
PROJECT MANAGER
Settings
Add Sources
Language Templates
IP Catalog
IP INTEGRATOR
Create Block Design
Open Block Design
Generate Block Design
SIMULATION
Run Simulation
RTL ANALYSIS
Open Elaborated Design
SYNTHESIS
Run Synthesis
Open Synthesized Design
IMPLEMENTATION
Run Implementation
Open Implemented Design
PROGRAM AND DEBUG
Generate Bitstream
Open Hardware Manager
SIMULATION - Behavioral Simulation - Functional - sim_1 - main_test
Scope Sources
Design Sources (6)
alu (alu.v)
ctrl_unit (ctrl_unit.v)
fake_ram (fake_ram.v)
inst_dec (inst_dec.v)
pc_unit (pc_unit.v)
reg_file (reg_file.v)
Constraints
Simulation Sources (3)
sim_1 (3)
main_test (main_test.v) (6)
main_reg : reg_file (reg_file.v)
main_inst : inst_dec (inst_dec.v)
main_alu : alu (alu.v)
main_ctrl : ctrl_unit (ctrl_unit.v)
pc_main : pc_unit (pc_unit.v)
main_ram : fake_ram (fake_ram.v)
decoder_unittests (decoder_unitest.v) (1)
regfile_unittest (regfile_unittest.v) (1)
Hierarchy Libraries Compile Order
main_test.v alu.v ctrl_unit.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
opcode,
dataD,
// Outputs
pcO
);
fake_ram main_ram(
// Inputs
clk,
ram_we,
pcO,
dataI,
// Outputs
dataO
);
initial begin
clk = 0;
reset = 1;
#20
reset = 0;
end
//Clock generation
always begin
#5;
clk = ~clk;
end
endmodule
Tcl Console Messages Log
10:14 Insert Verilog



5 Digital Design



https://drive.google.com/drive/folders/1j1CQQcNiZkKO7DBlcB5UgeO1-nQJEAIK?usp=sharing