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

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.

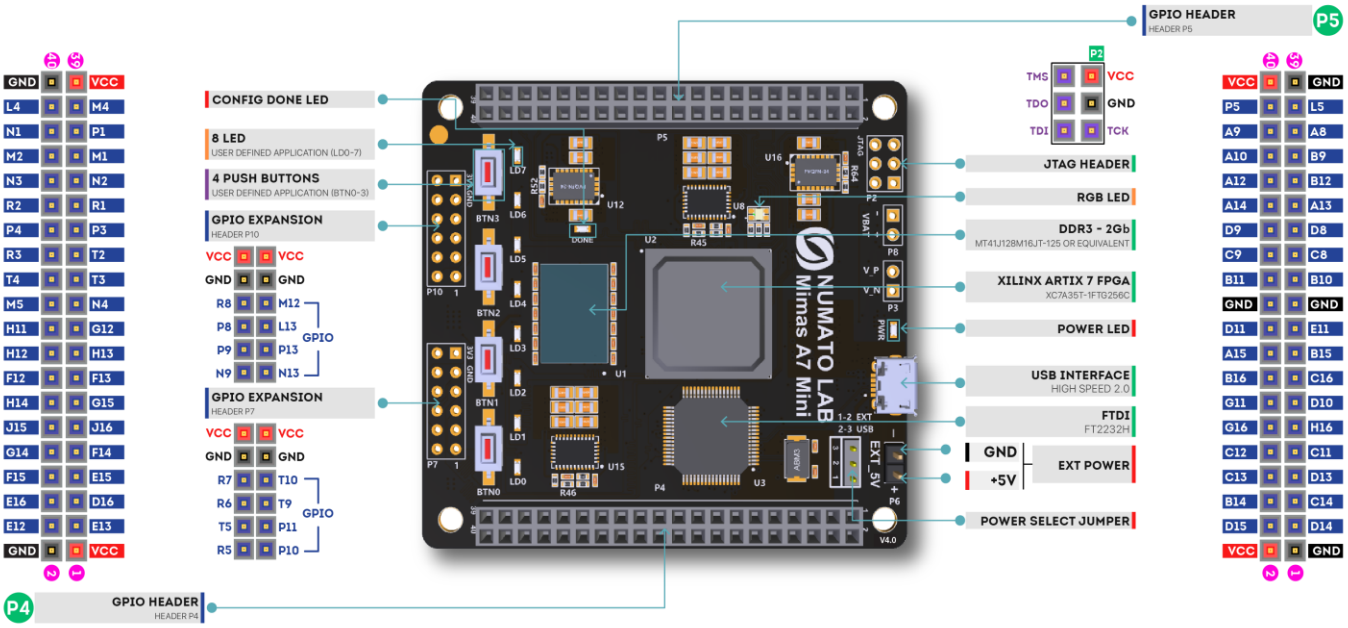
Features,

→ 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

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

Software to download

Notepad++

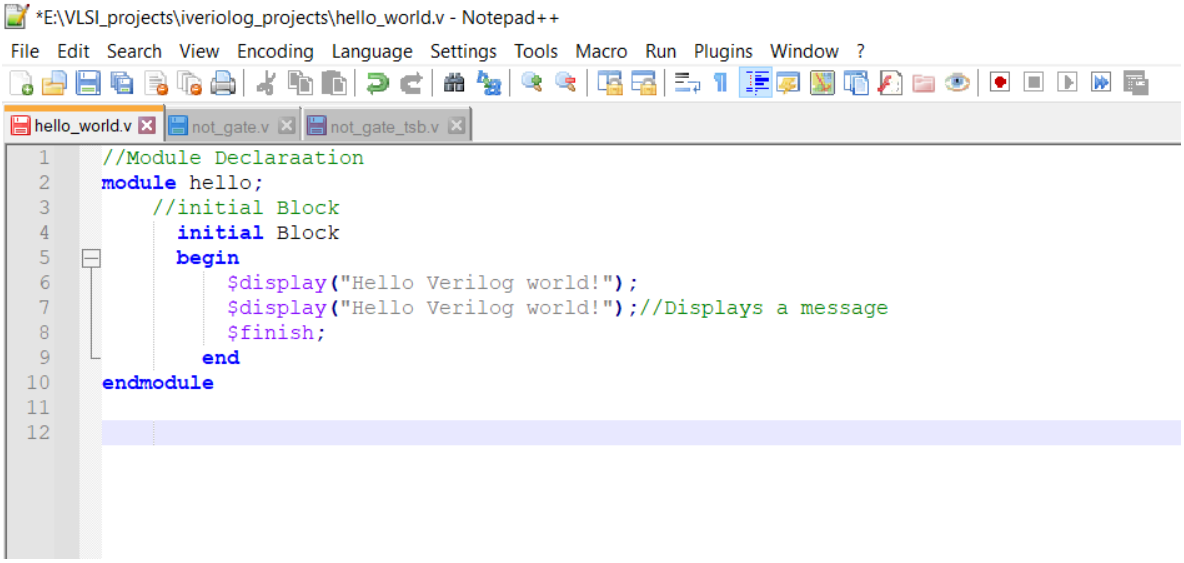
Step 1- Write the verilog program in notepad ++

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

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

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 ?

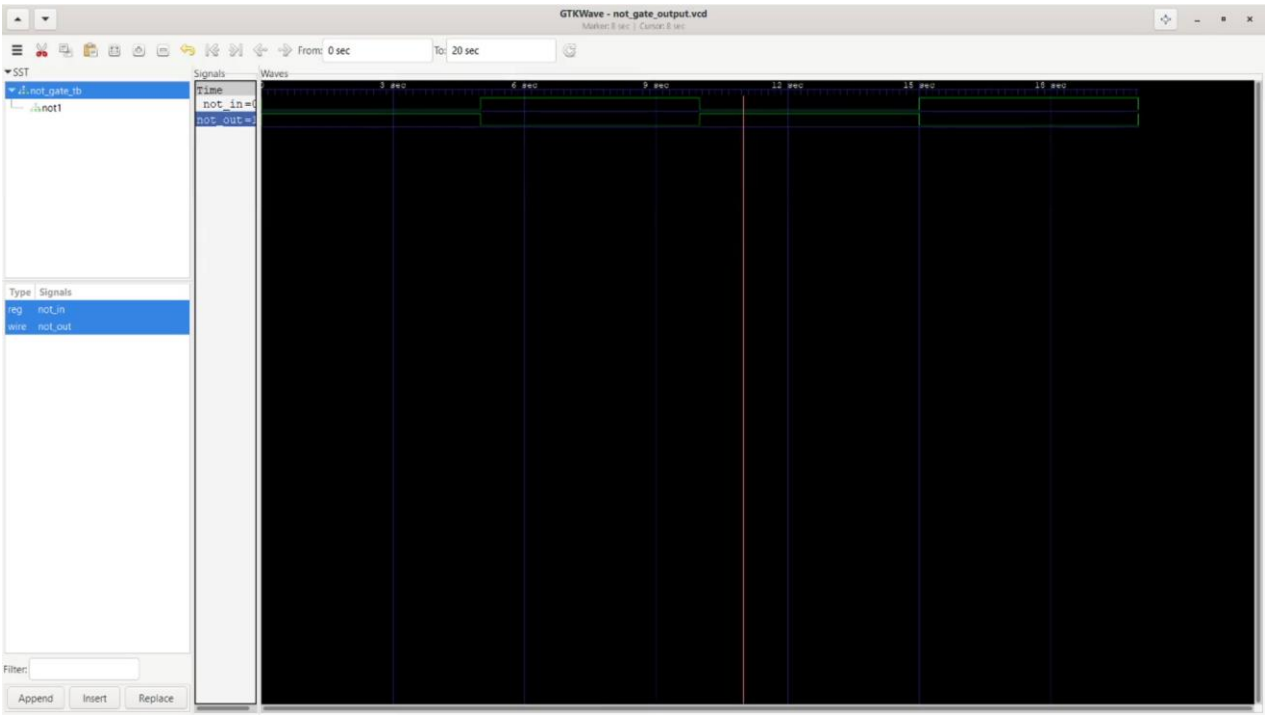
hello_world.v | not_gate.v | not_gate_tsb.v

```
1  //Test Bench for not_gate
2  module not_gate_tb;
3
4      //wire/reg Declarations
5      reg not_in;         //input
6      wire not_out;       //output
7
8      //Instantiation
9      not_gate not1(not_in, not_out);
10
11     //Initial block
12     initial
13     begin
14
15     $dumpfile("not_gate_output.vcd");
16     $dumpvars(0,not_gate_tsb);
17
18     not_in = 1'b0;
19     #5 not_in = 1'b1;
20     #5 not_in = 1'b0;
21     #5 not_in = 1'b1;
22     #5 not_in = 1'b0;
23     end
```

COMMAND PROMPT

```
C:\Windows\System32\cmd.exe - gtkwave  not_gate_output.vcd

P:\VLSI_projects\iverilog_projects\hello_world>iverilog -o hello_out1 hello_world.v

P:\VLSI_projects\iverilog_projects\hello_world>vvp hello_out1
Hello Verilog world!!
Welcome to VLSI Projects!
hello_world.v:8: $finish called at 0 (1s)

P:\VLSI_projects\iverilog_projects\hello_world>cd ../

P:\VLSI_projects\iverilog_projects>cd not_gate

P:\VLSI_projects\iverilog_projects\not_gate>iverilog -o not_gate_output not_gate_tb.v not_gate.v

P:\VLSI_projects\iverilog_projects\not_gate>vvp not_gate_output
VCD info: dumpfile not_gate_output.vcd opened for output.

P:\VLSI_projects\iverilog_projects\not_gate>gtkwave

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

GTKWAVE | Use the -h, --help command line flags to display help.
WM Destroy

P:\VLSI_projects\iverilog_projects\not_gate>gtkwave not_gate_output.vcd

GTKWave Analyzer v3.3.108 (w)1999-2020 BSI

[0] start time.
[20] end time.
```

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



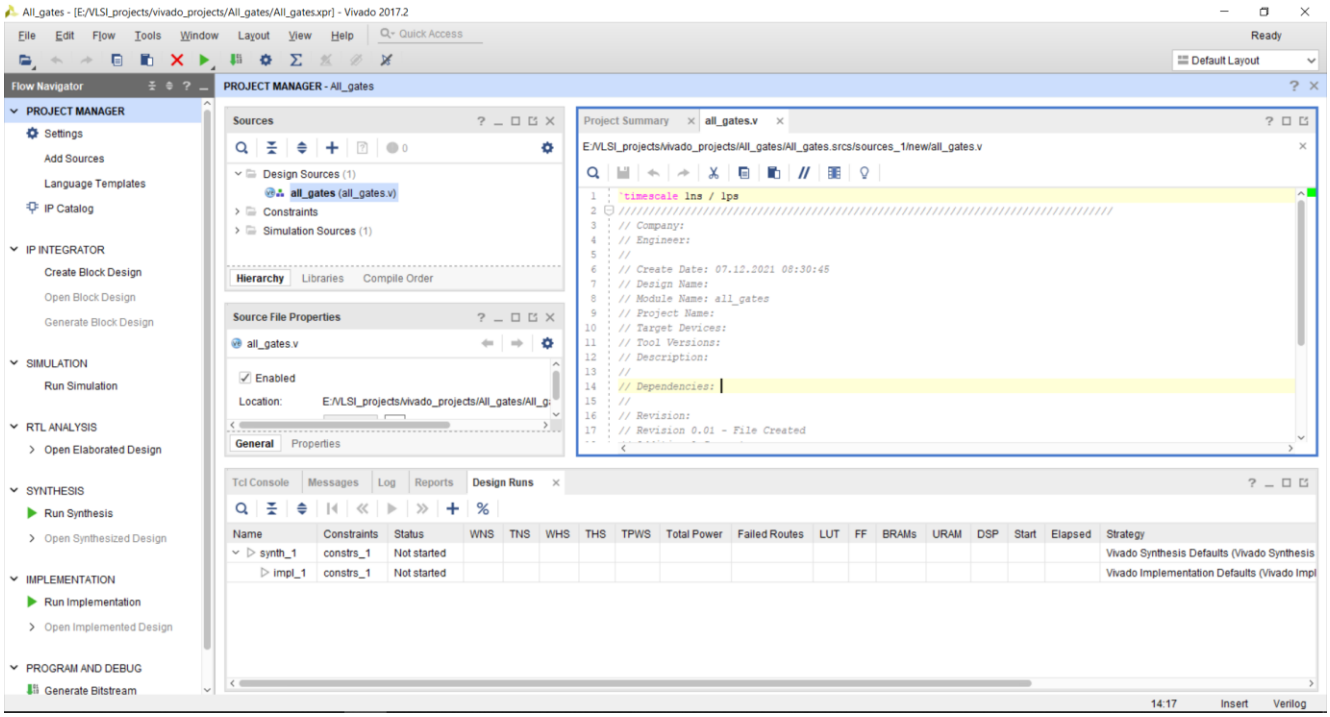
NOW WITH ALL BASIC GATES

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

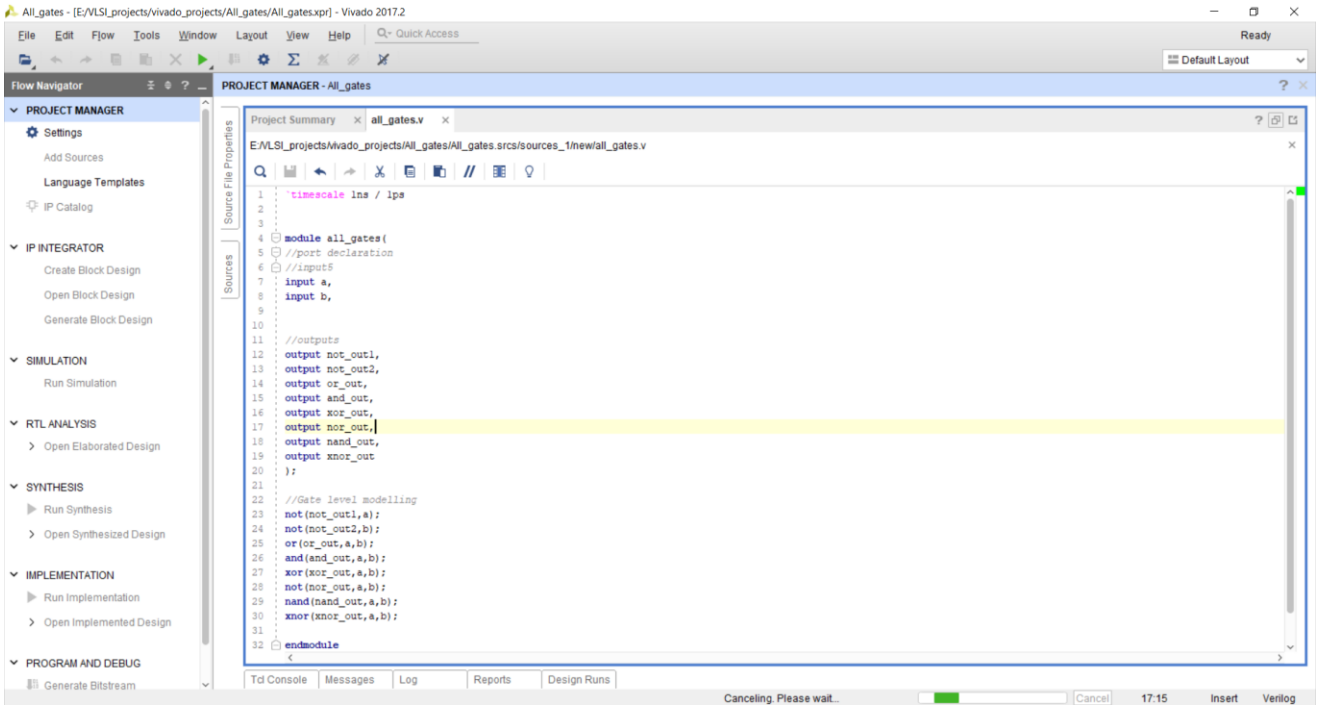
numato (Numato Lab) (github.com)

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.

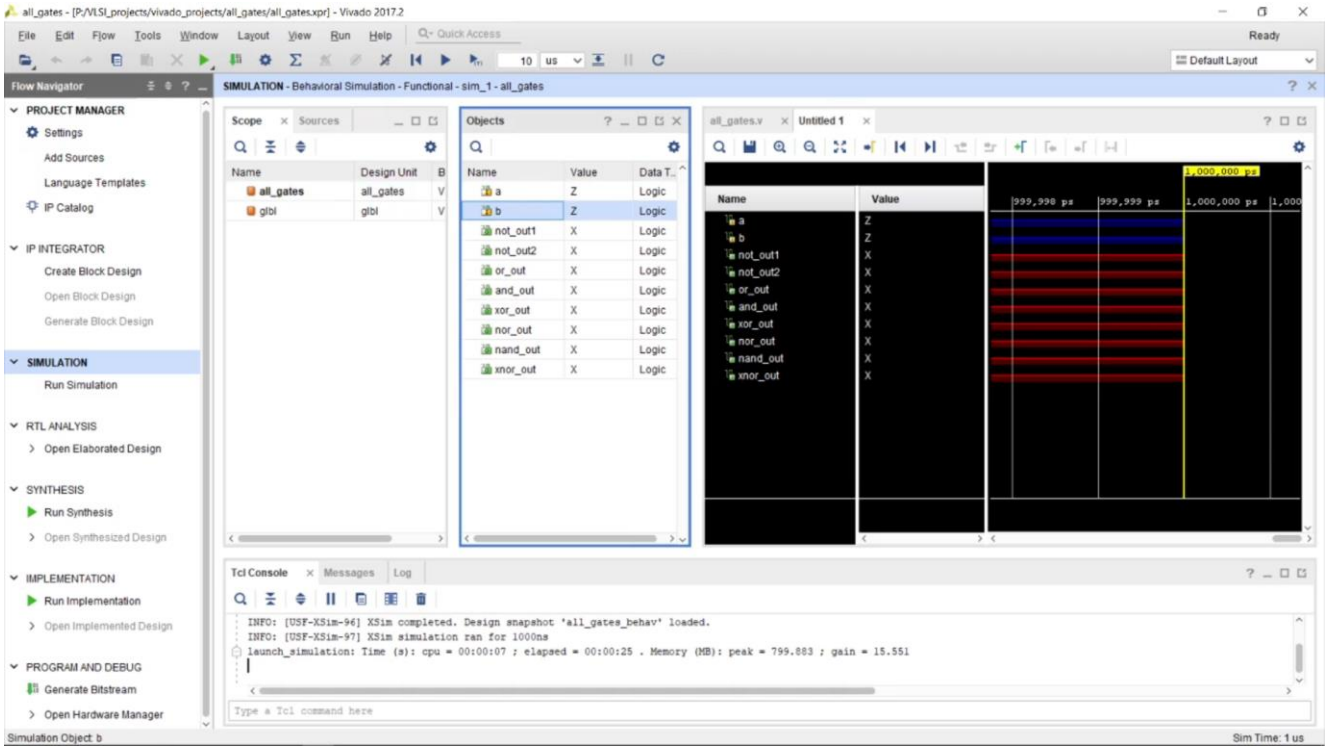
After that we have to open the vivado software where we have to create a new project and save in a directory, and after that through the process we have to select the board on it, that is Mimas A7 mini.



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



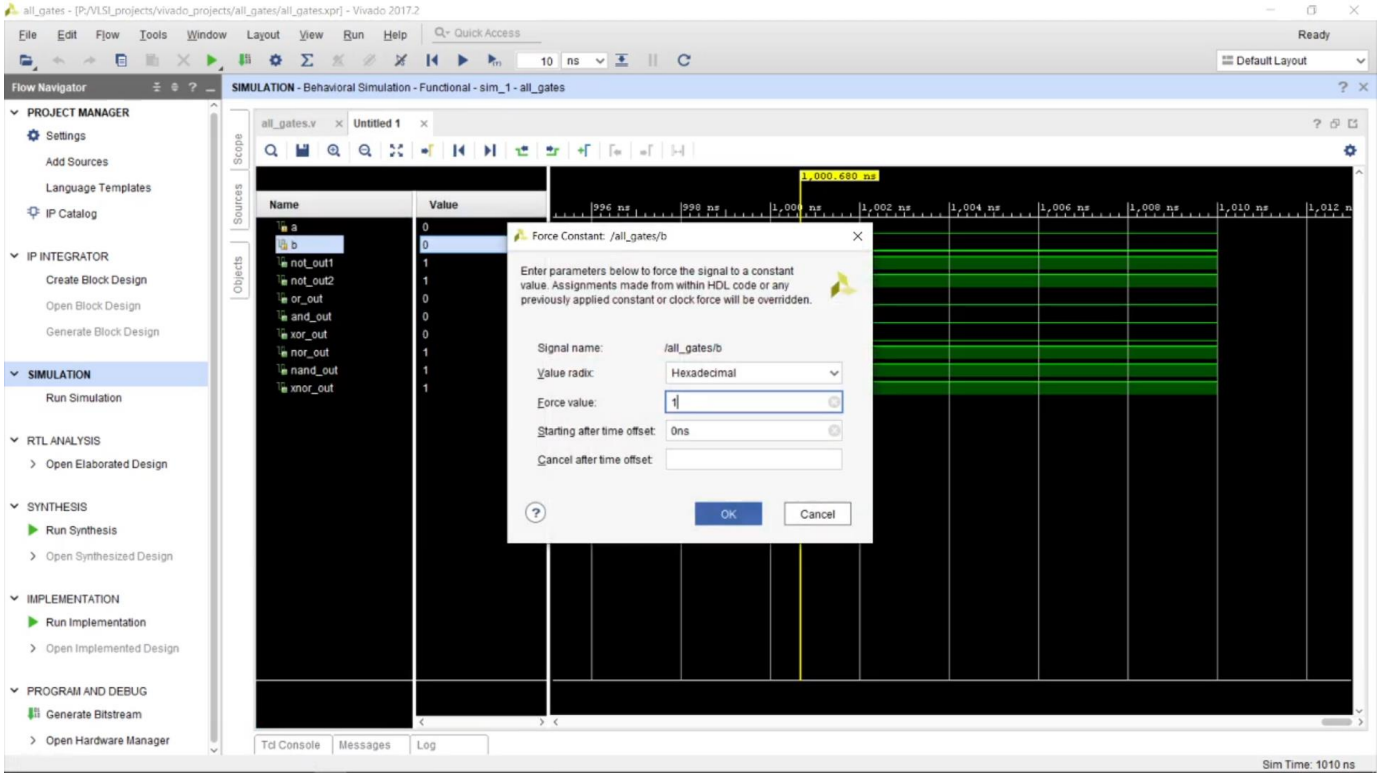
After that we have to run simulation to our program



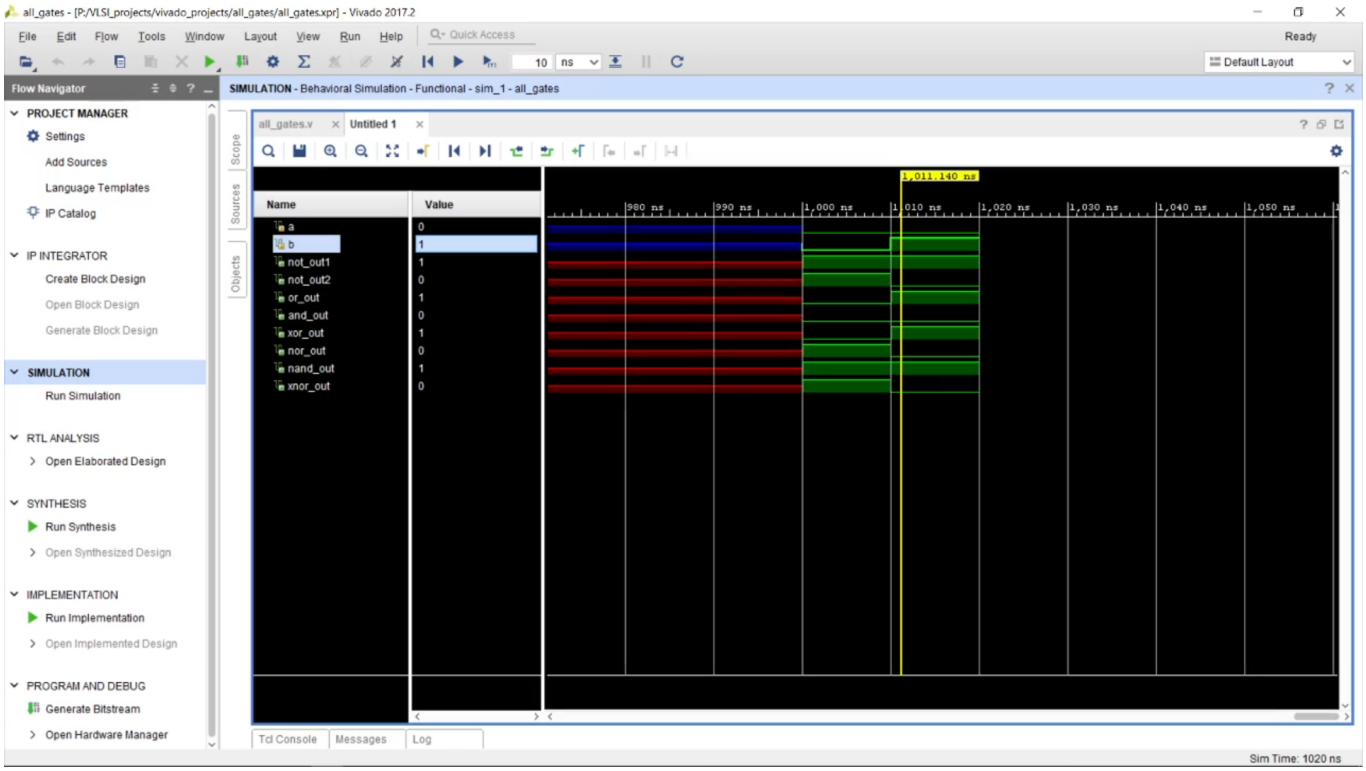
As we can see the inputs and the outputs.

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

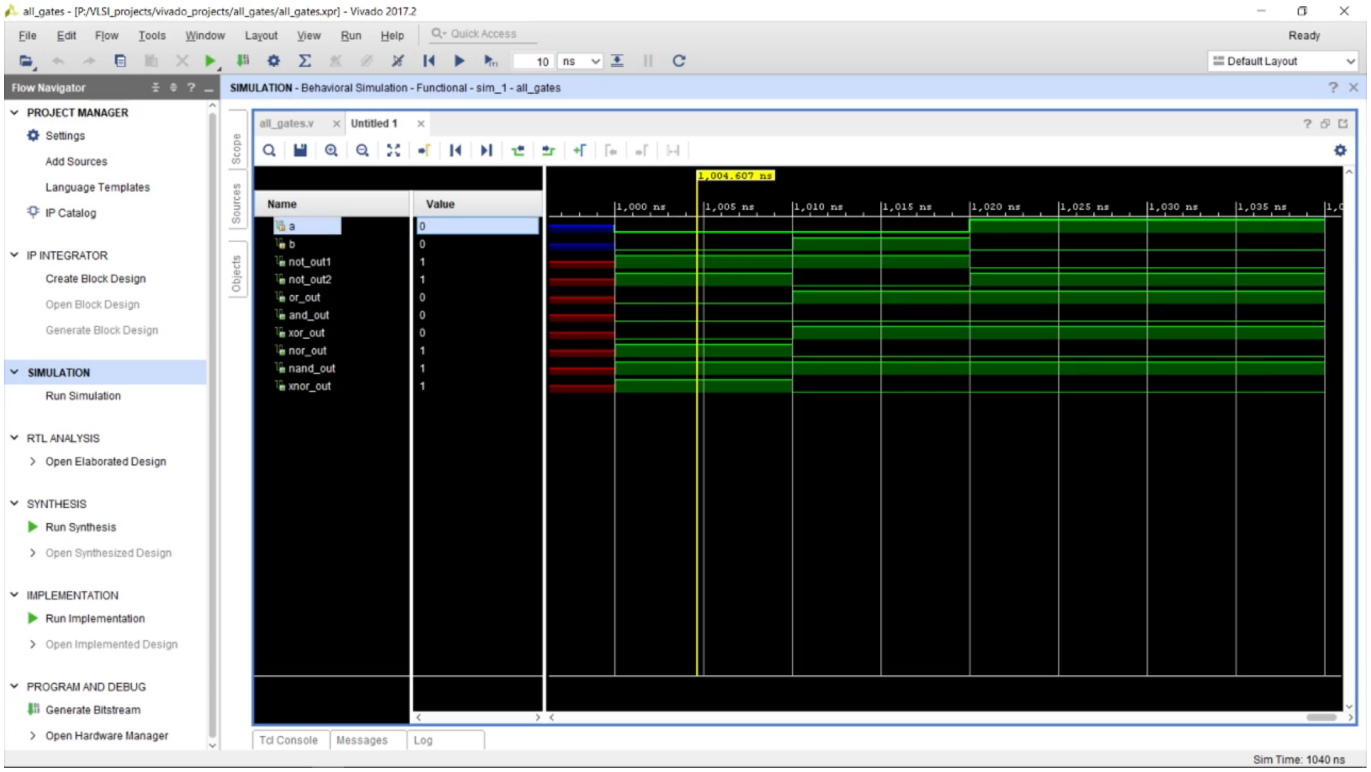
We can choose any type of value as hexadecimal or decimal as we are only going to deal with 0 and 1, Force value for a will be 0 and b will be 1, with 10 ns .



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.



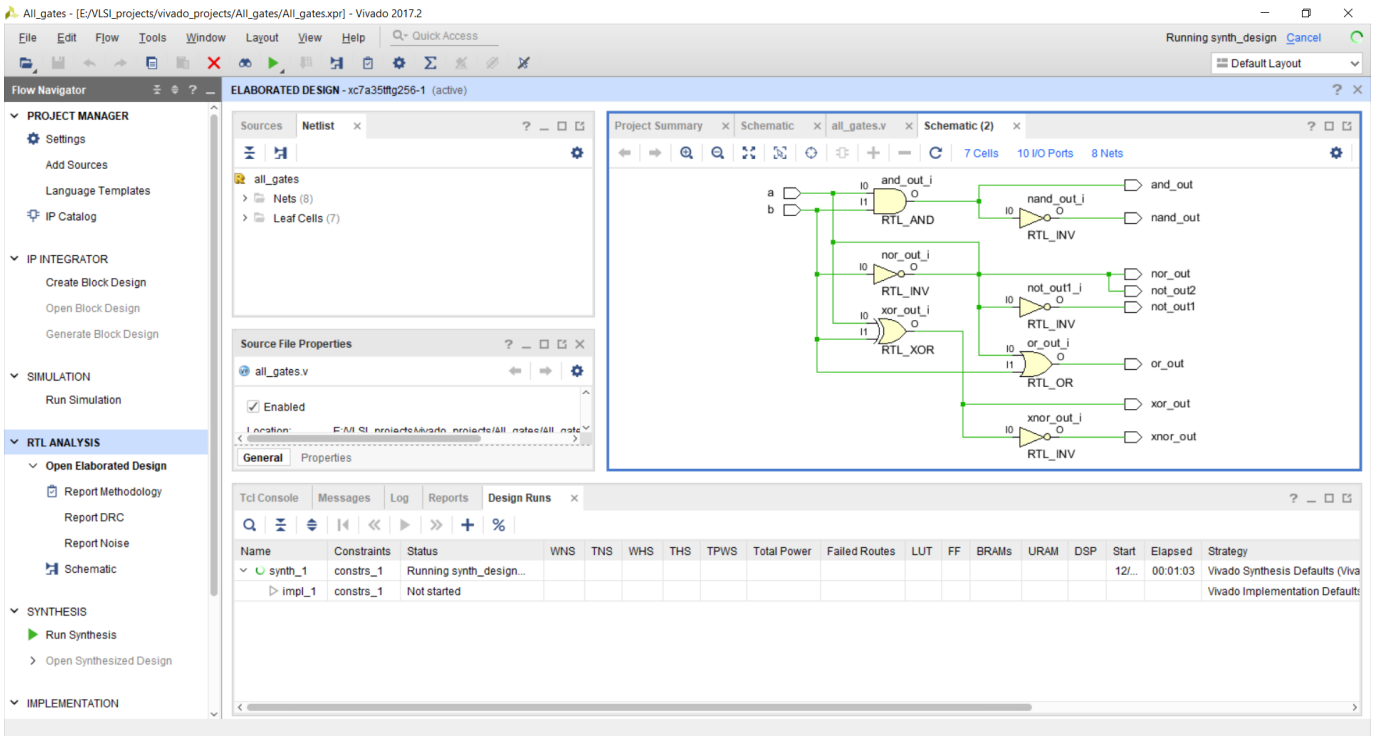
With respect to the values of a and b, and or not xor nor xnot nand output will change its value eventually

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

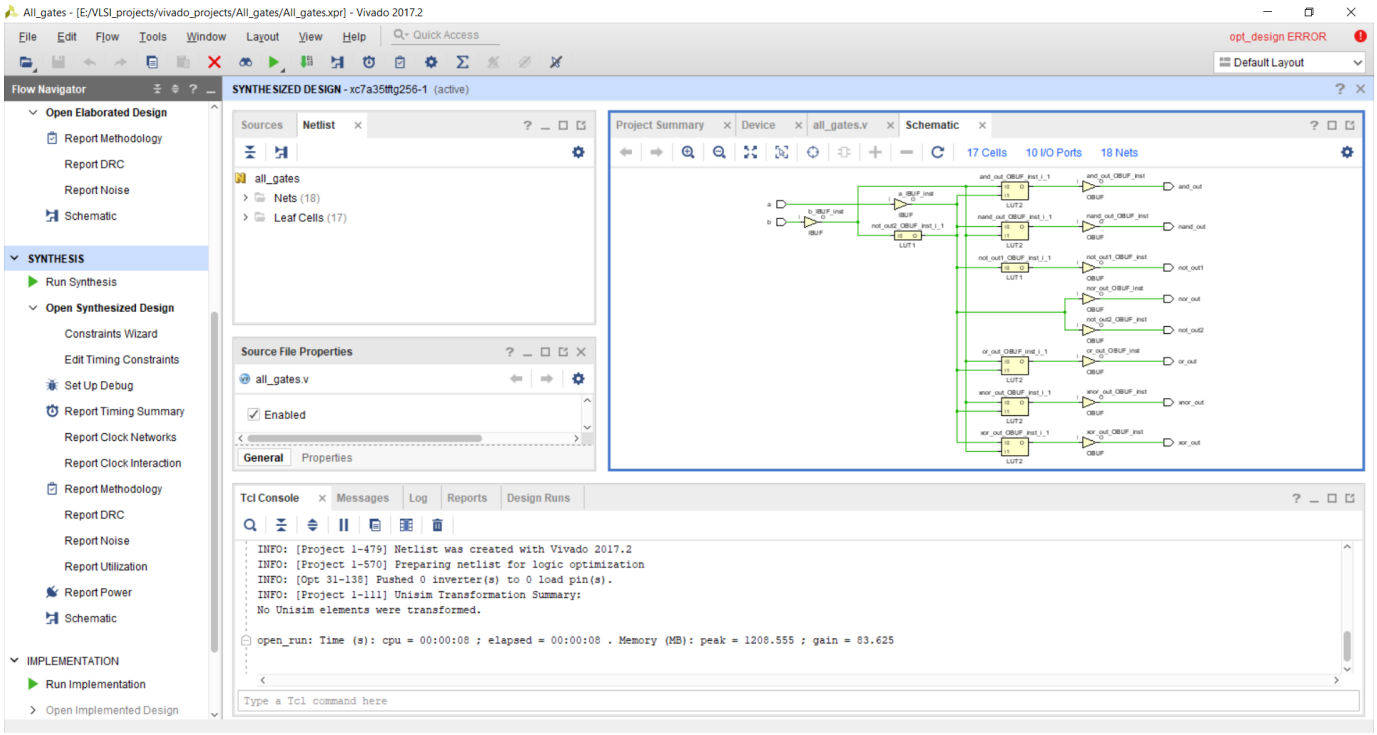
To implement it we first have to synthesis is by run synthesis , we can also elaborate the synthesis, under RTL analysis by choosing Schematic.

The RTL analysis is based on our Verilog code.

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



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

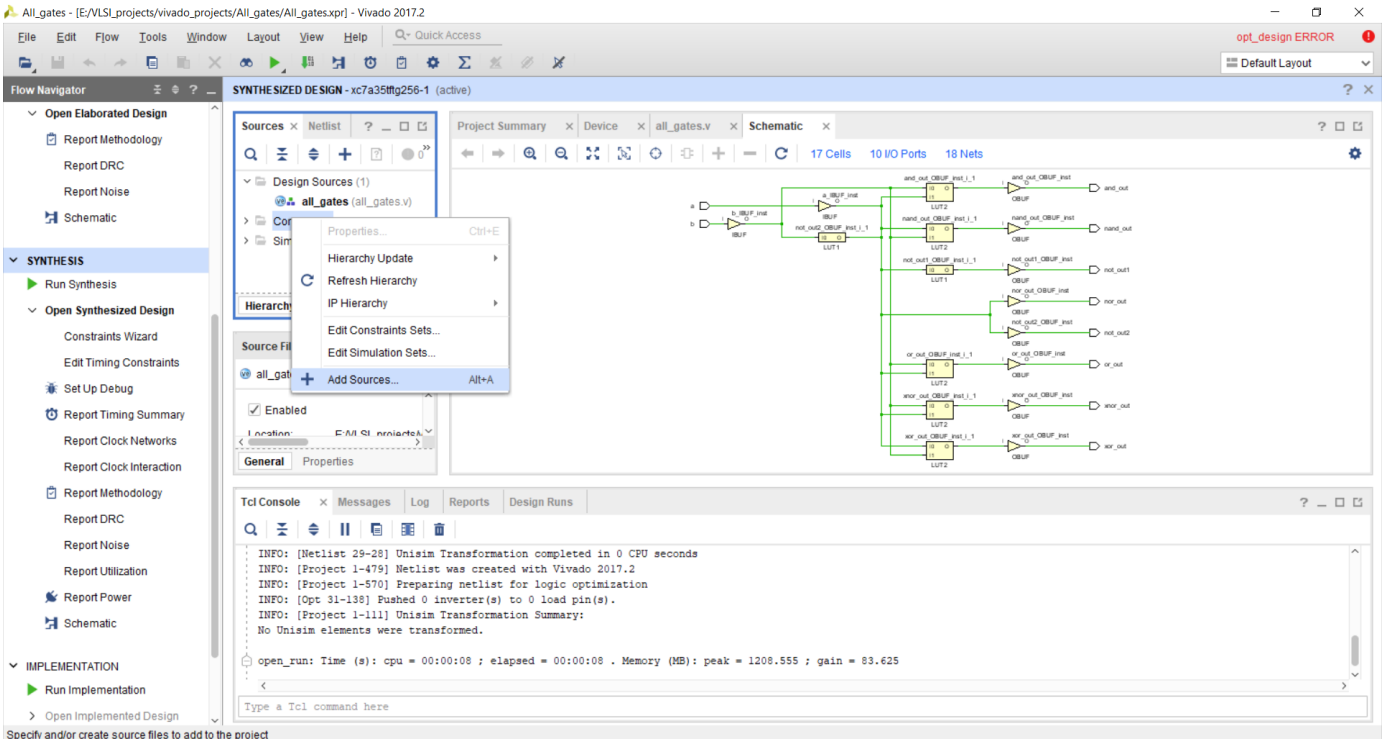
Mimasa7Mini - Notepad

File Edit Format View Help

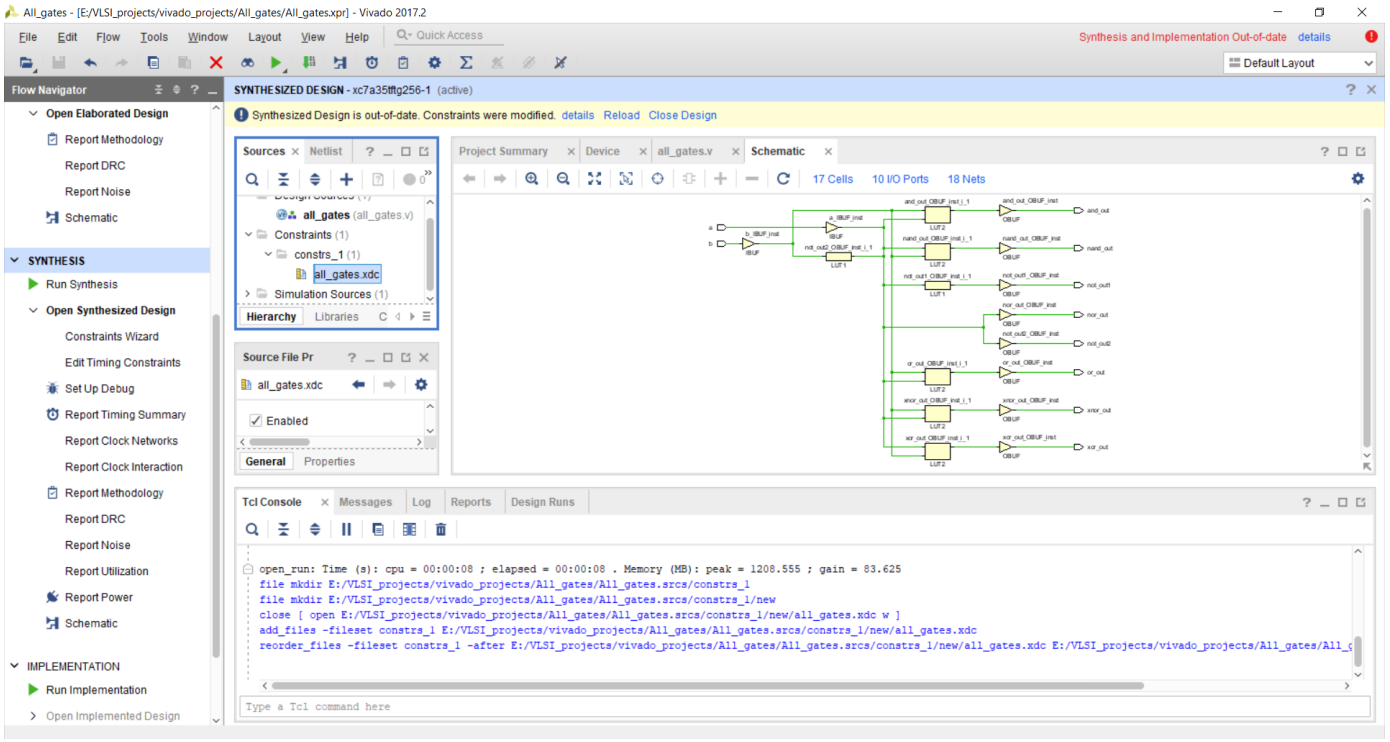
```
##############################################################################################################
#                                                QSPI - FLASH                                                #
##############################################################################################################
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
```

Ln 112, Col 1 | 100% | Windows (CRLF) | UTF-8

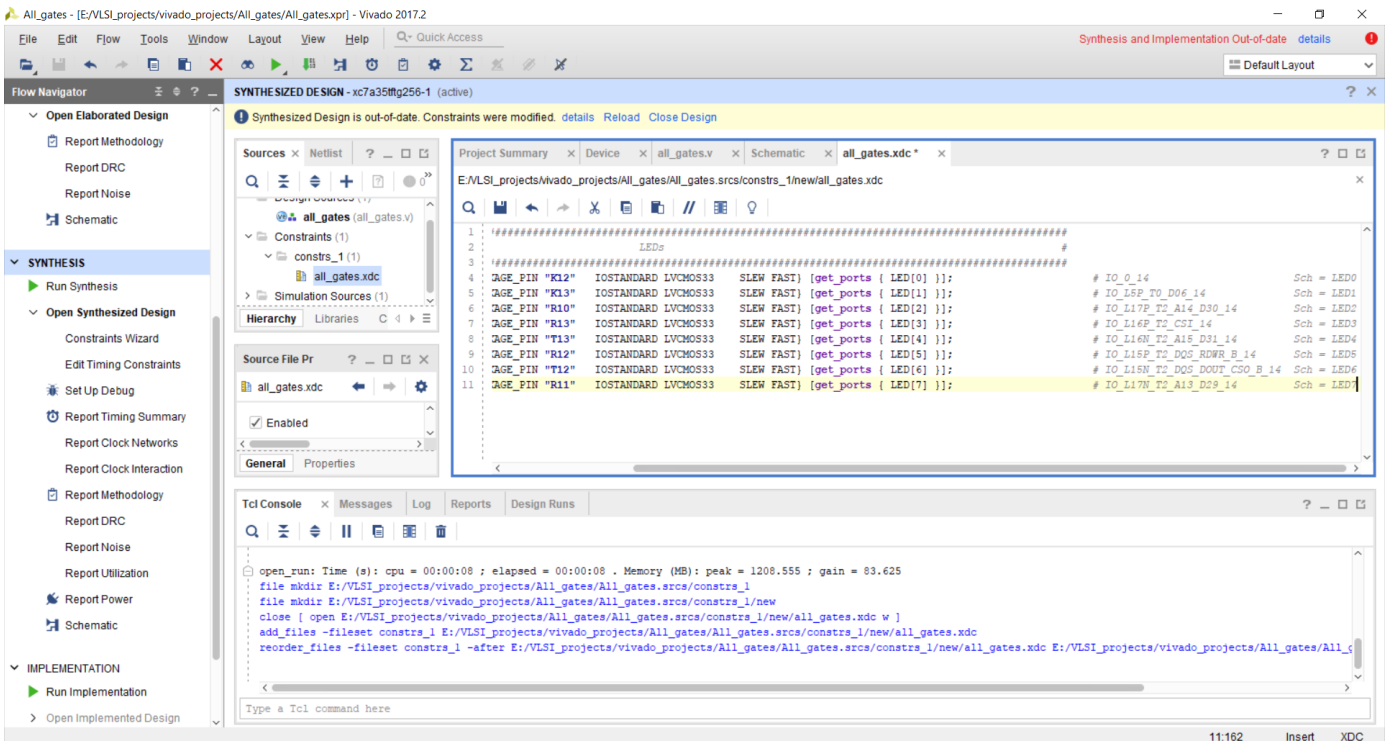
Now let's add the XDC file to vivado by adding sources to constrains, add or create constraints, then create a file name as "all_gates.xdc"



There by it is added!

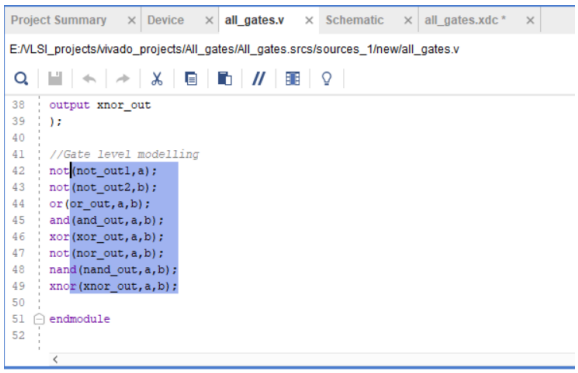


Ok so now it is time to get the xdc file which we download, copy the xdc file for the needed requirements, for us it is Leds and pushbutton and we will paste it here in the constraints file as shown below.

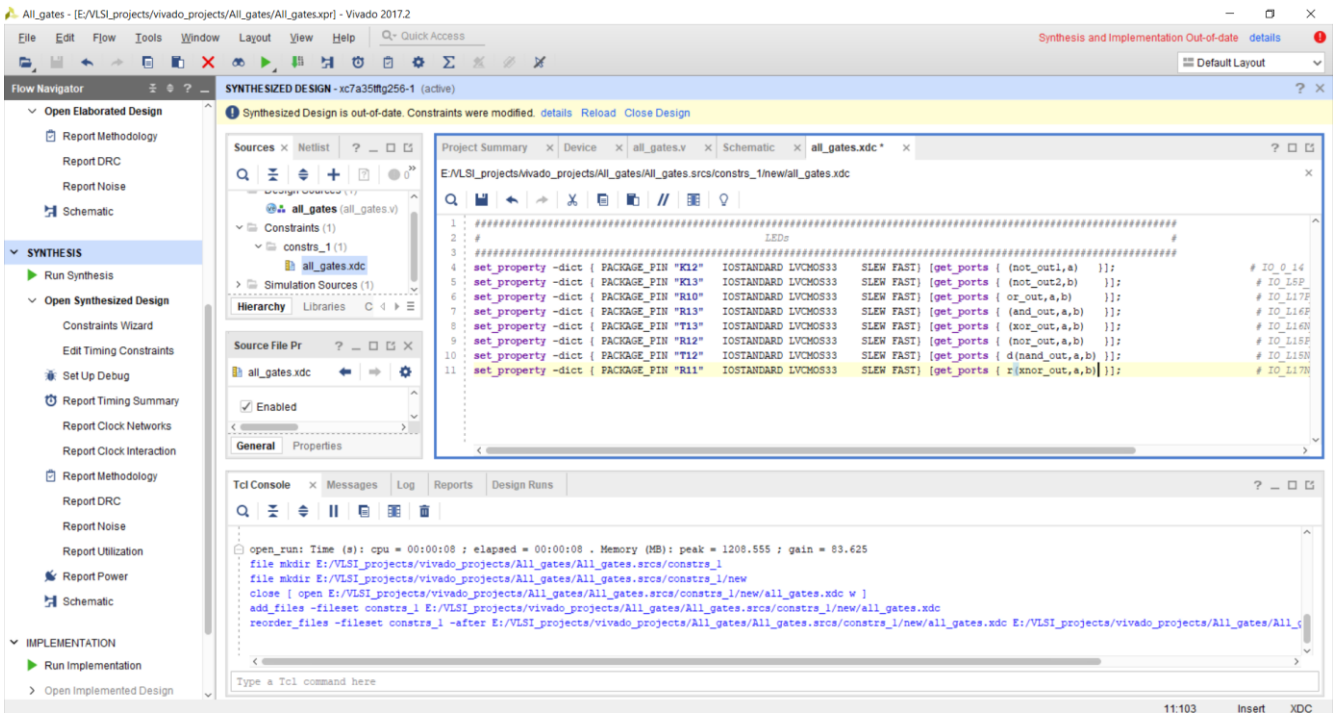


Now we have to change the identifiers in the LED pins so all we will do will take the output from our Verilog code and paste it here as shown below

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.

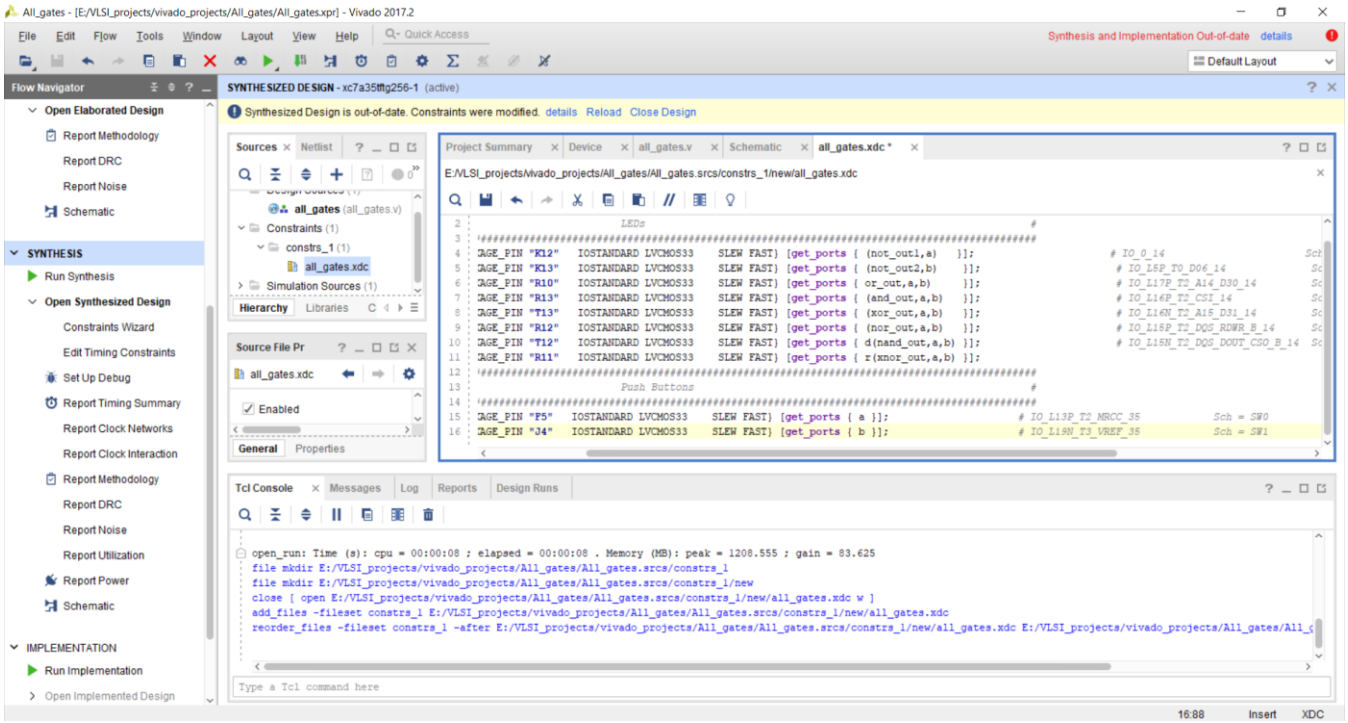
Mimasa7Mini - Notepad

File Edit Format View Help

```
##############################################################################################################
#                                          QSPI - FLASH                                                      #
##############################################################################################################
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
```

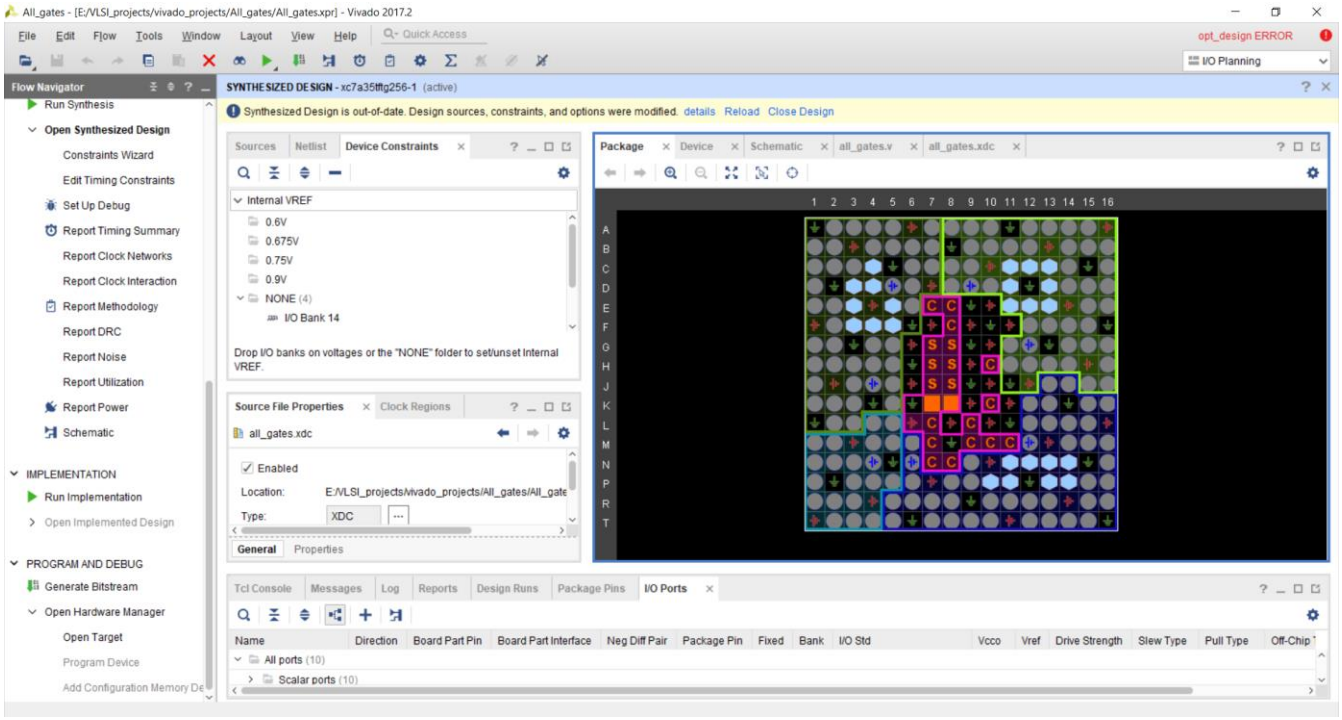
Ln 97, Col 162 | 100% | Windows (CRLF) | UTF-8

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

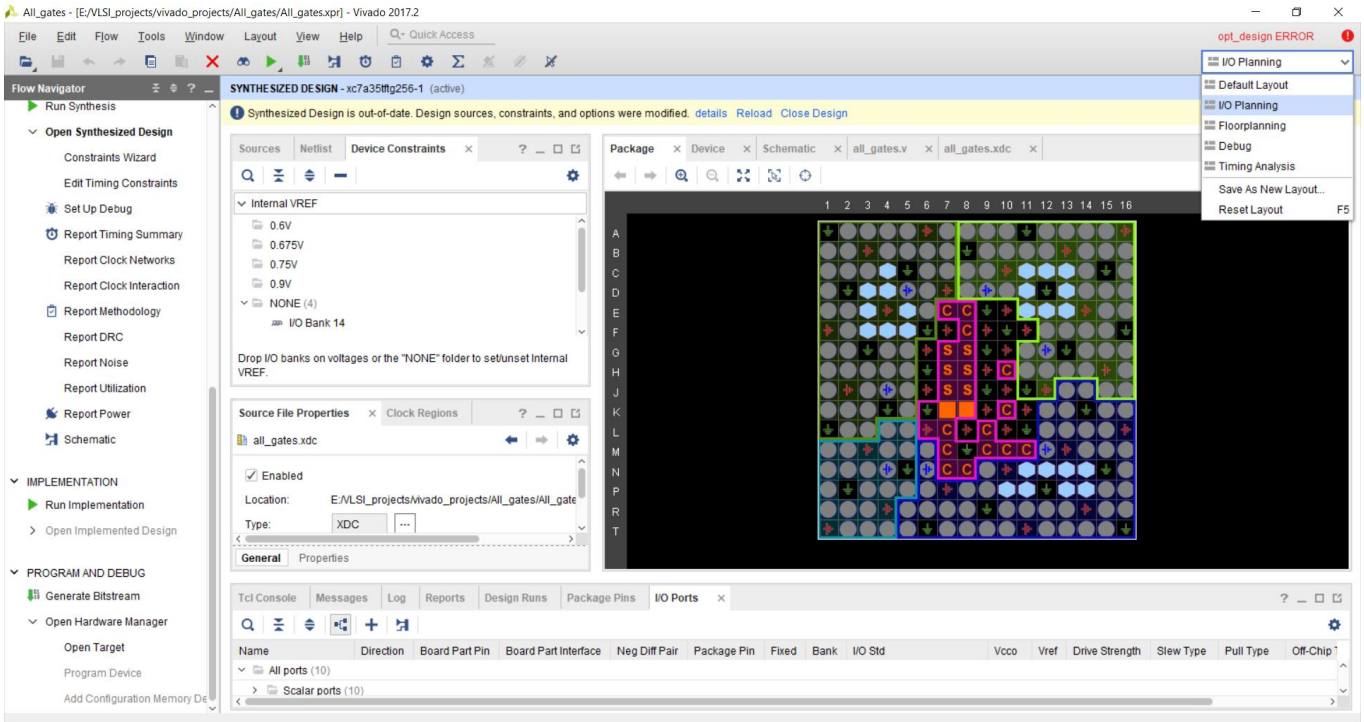


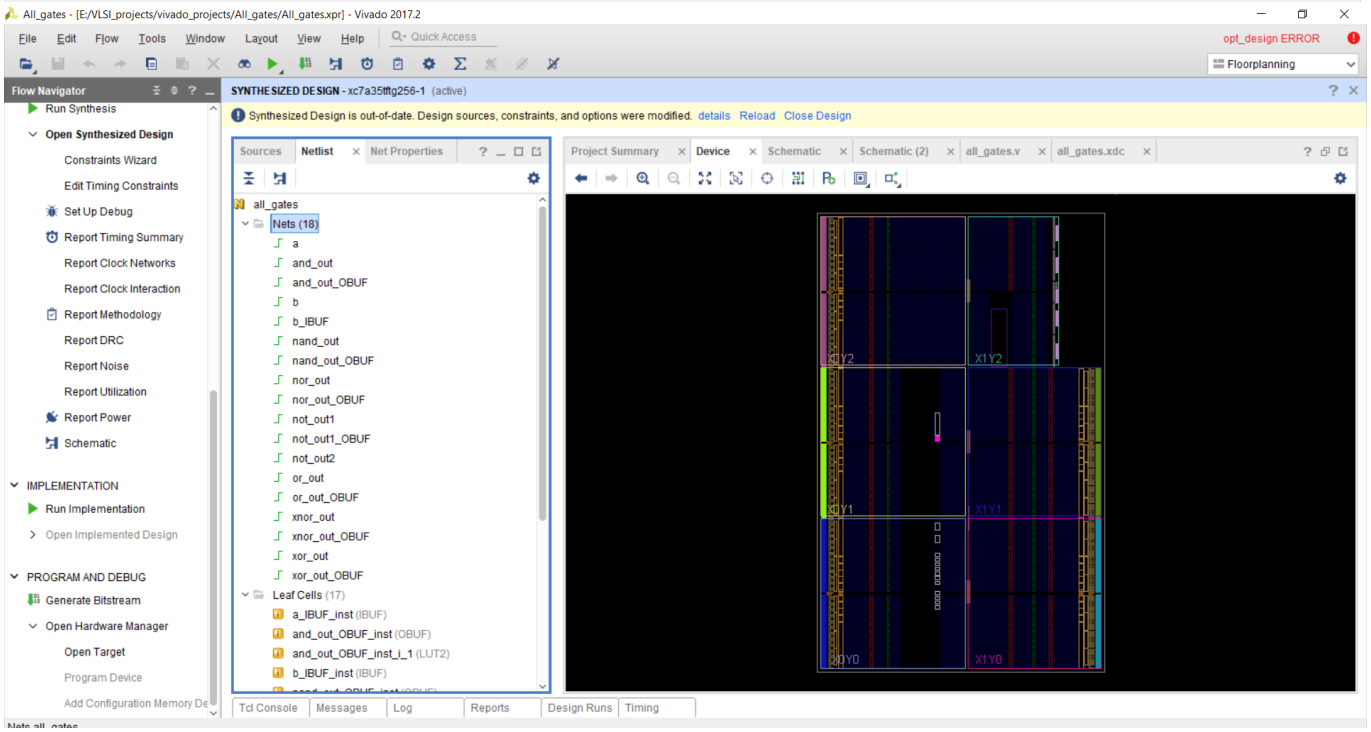
After that we under program and debug we have to choose generate bitstream

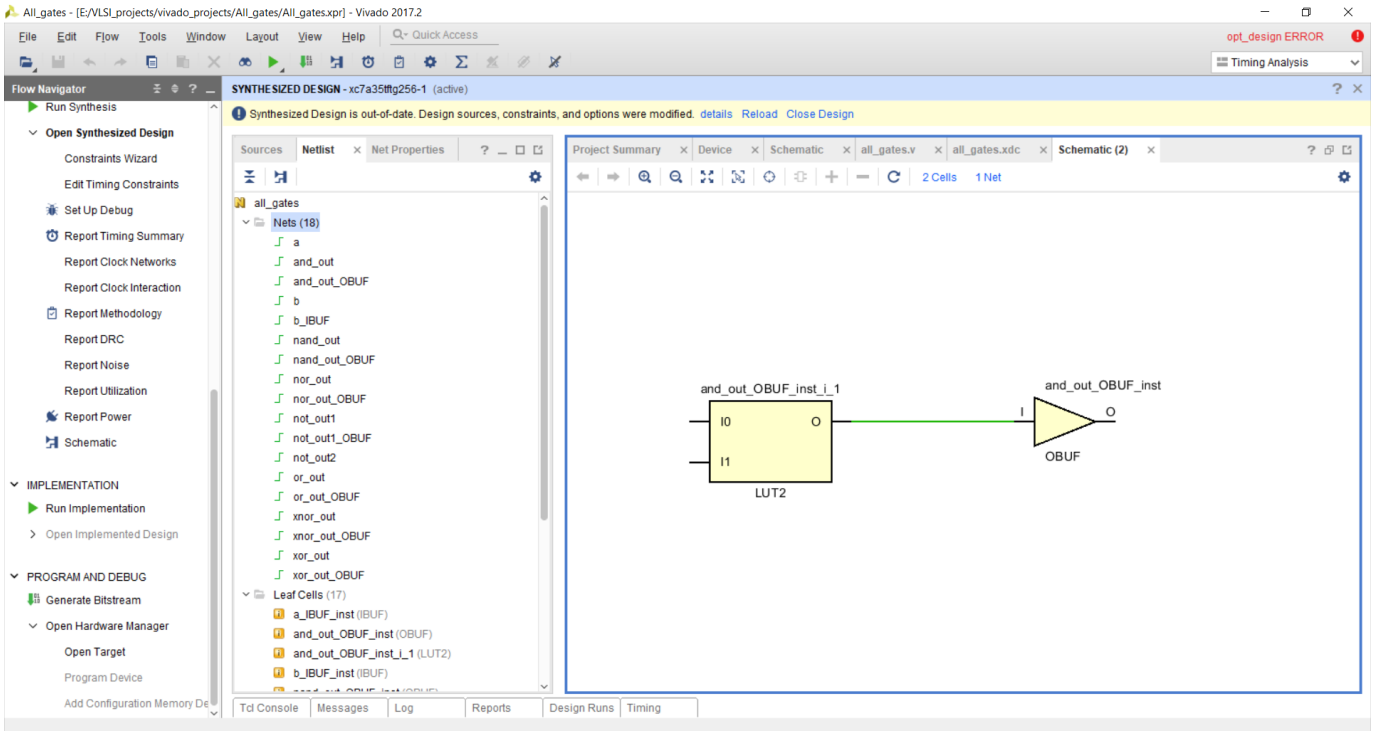
After running, I/O planning is what we can see.



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.v all_gates.xdc
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
A B C D E F G H J K L M N P R T
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)


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.v all_gates.xdc
X0Y2 X1Y2
X0Y1 X1Y1
X0Y0 X1Y0
Tcl Console Messages Log Reports Design Runs Timing



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.