

# **LAB 3: PRIME NUMBER AND INTRODUCTION TO XENON BOARD**

ARYA AGARWAL  
Roll Number - 210070012

18 August 2022

## **1. Overview of the Experiment / Assignment:**

- In this experiment, first we were introduced to Xenon board.
- Then using Quartus and Xenon board given to us we have to design a system that detects a prime number.
- After writing the vhdl code, we have to test our logic circuit using xenon board.

## **2. Experiment Setup or Approach to the Assignment:**

- By observing the Truth table (given below) for detecting a prime number, we try to formulate the possible circuit which is drawn below.

N3	N2	N1	N0	P
0	0	0	0	0
0	0	0	1	0
0	0	1	0	1
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

- Therefore, the possible figure of circuit is given in Fig 1

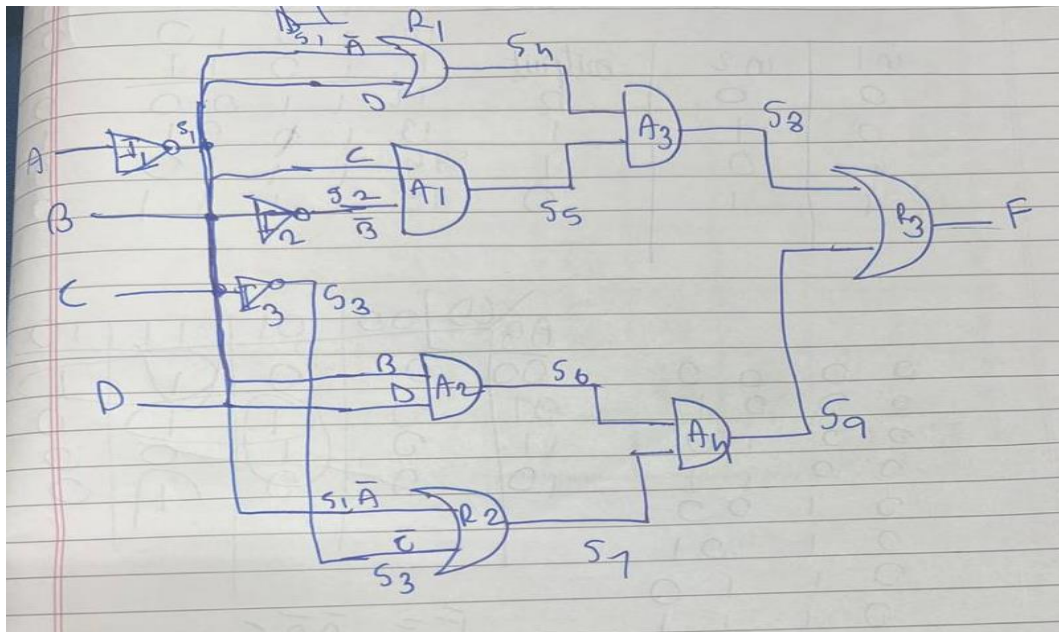


Fig 1

## 2.1 Design Code and Documentation:

Now once we have drawn the circuits on paper, its time to describe the circuit using code in vhdl language in Quartus. The dut, gates files will almost be same except some minor changes in both case.

The code for the main file is written below which is used to describe the circuit.

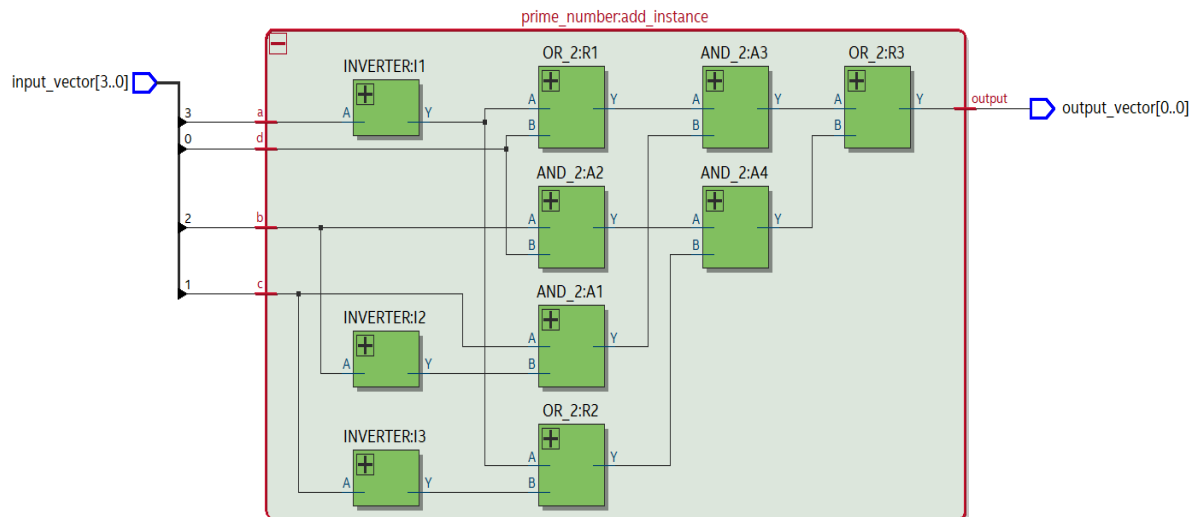
You can refer to the pen paper design for instance and signal names.

```

1  library ieee;
2  use ieee.std_logic_1164.all;
3
4  library work;
5  use work.Gates.all;
6
7  entity prime_number is
8  port (a,b,c,d:in std_logic;
9        output:out std_logic);
10 end entity prime_number;
11
12 architecture struct of prime_number is
13   signal s1,s2,s3,s4,s5,s6,s7,s8,s9:std_logic;
14 begin
15   I1 : INVERTER
16     port map(a=>a,y=>s1);
17   I2 : INVERTER
18     port map(a=>b,y=>s2);
19   I3 : INVERTER
20     port map(a=>c,y=>s3);
21   A1 : AND_2
22     port map(a=>c,b=>s2,y=>s5);
23   A2 : AND_2
24     port map(a=>b,b=>d,y=>s6);
25   A3 : AND_2
26     port map(a=>s4,b=>s5,y=>s8);
27   A4 : AND_2
28     port map(a=>s6,b=>s7,y=>s9);
29   R1 : OR_2
30     PORT map(a=>s1,b=>d,y=>s4);
31   R2 : OR_2
32     PORT map(a=>s1,b=>s3,y=>s7);
33   R3 : OR_2
34     PORT map(a=>s8,b=>s9,y=>output);
35 end struct;
36
37
38

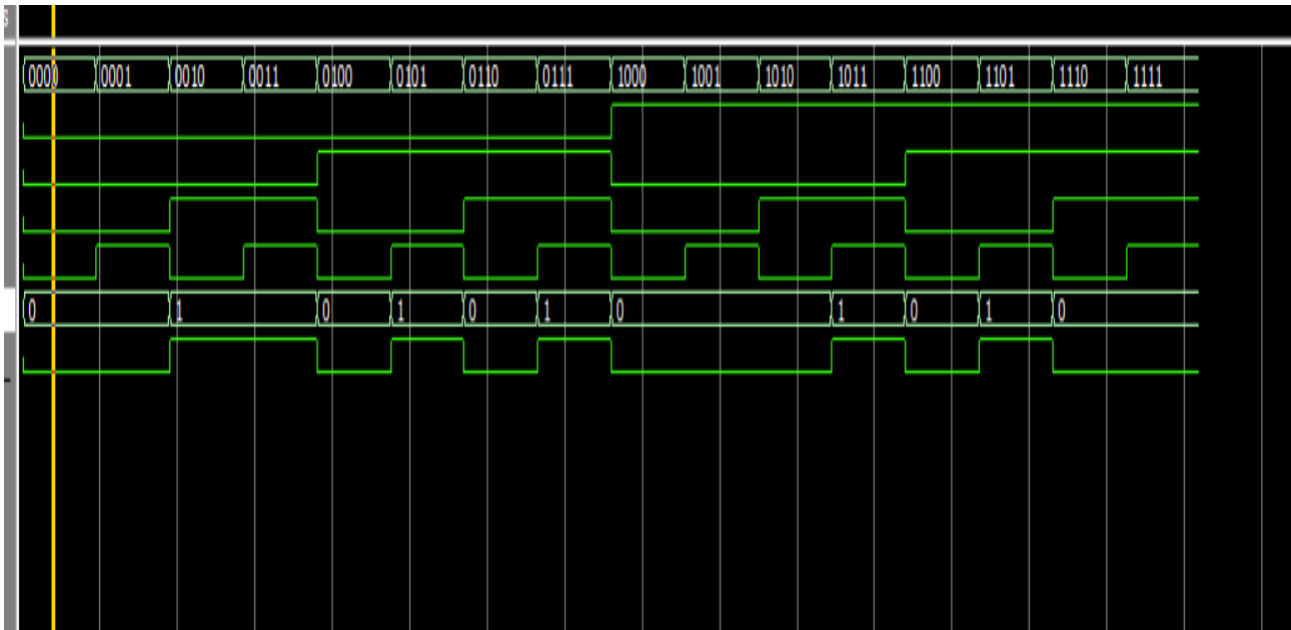
```

## RTL VIEW



## 3. Observations:

After you run the analysis of code with no error. The next step is to run the simulation. For this we again use the Modelsim - Altera Software. We also need the Testbench and the tracefile (already provided) to run the simulation. The simulation is shown below.



## TRANSCRIPT(all test cases passed successfully)

```

M MODELSIM - INTEL FPGA STARTER EDITION 2020.1
File Edit View Compile Simulate Add Transcript Tools Layout Bookmarks Window Help
ColumnLayout Default
Transcript
# -- Compiling architecture struct of prime_number
# End time: 16:14:37 on Aug 18, 2022, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
#
# vcom -93 -work work (C:/Users/ARYA AGARWAL/Desktop/LAB3/Testbench.vhdl)
# Model Technology ModelSim - Intel FPGA Edition vcom 2020.1 Compiler 2020.02 Feb 28 2020
# Start time: 16:14:37 on Aug 18, 2022
# vcom -reportprogress 300 -93 -work work C:/Users/ARYA AGARWAL/Desktop/LAB3/Testbench.vhdl
# -- Loading package STANDARD
# -- Loading package TEXTIO
# -- Loading package std_logic_1164
# -- Loading package Testbench
# -- Compiling architecture Behave of Testbench
# End time: 16:14:37 on Aug 18, 2022, Elapsed time: 0:00:00
# Errors: 0, Warnings: 0
#
# vsim -t lps -L altera -L lpm -L sgate -L altera_mf -L altera_lnsim -L fiftyfivenm -L rtl_work -L work -voptargs="+acc" Testbench
# vsim -t lps -L altera -L lpm -L sgate -L altera_mf -L altera_lnsim -L fiftyfivenm -L rtl_work -L work -voptargs="+acc" Testbench
# Start time: 16:14:37 on Aug 18, 2022
# Loading std.standard
# Loading std.textio(body)
# Loading ieee.std_logic_1164(body)
# Loading work.testbench(behavior)
# Loading work.dut(dutwrap)
# Loading work.gates
# Loading work.prime_number(struct)
# Loading work.inverter(equations)
# Loading work.and_2(equations)
# Loading work.or_2(equations)
#
# add wave *
# view structure
# .main_panel.structure.interior.cs.body.struct
# view signals
# .main_panel.objects.interior.cs.body.tree
# run -all
# ** Note: SUCCESS, all tests passed.
# Time: 304 ns Iteration: 0 Instance: /testbench
VSIOM 2>

```

## XENON BOARD

After running the code successfully, we have to connect our design to Xenon by creating svf file of our code. First, we will open PIN PLANNER from tools ,then we will assign the each input vector a particular switch and output vector a particular LEDs. We have to refer to Xenon User manual for numbering of the switches and LEDs. Before Using the Xenon Board, it should be checked thoroughly. We have to perform all the tests before we use it. Once the Testing is done we can throw our svf file in our Xen10 board using UrJTAG.

After doing we can check all the testcases manually.

Here is my Xenon board for a particular case. You can see the LED is ON.

