Experiment 2: Vowel Detector
A submission Report

Hardik Dhansukhbhai Panchal
Roll Number 200070054
EE-214, WEL, IIT Bombay
August 19, 2021

Overview of the experiment:

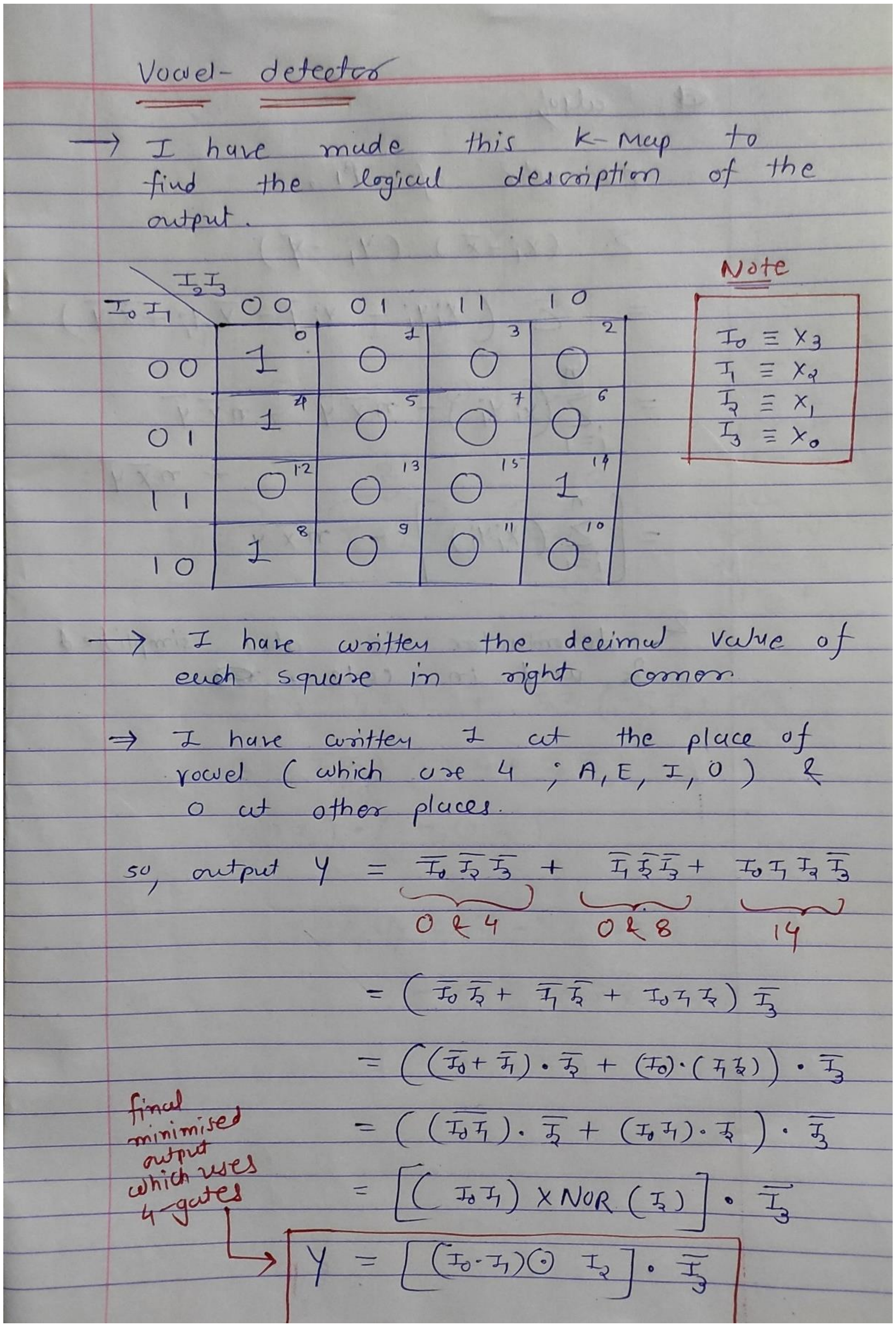
The purpose of this experiment was to get familiar with structural modelling concept used in VHDL codes. Means we have to instantiate components and use port map to connect those components.

We have to make a logic description of vowel detector. In that we have 16 alphabets and our code should identify the vowels and output 1 for that and 0 for other.

I have first made the DUT.vhdl file to take input and output data from TRACEFILE.txt and stored them as a vector. Modified the Testbench.vhdl according to need. Made K-map for this logic implementation and tried to minimise it and I succeed with making this whole implementation with only 4 gates.

My report has the RTL view of my design, the screenshots of RTL and Gate-Level simulations which contains their waveform window and transcript window. I have also specified the input and output format with MSB/LSB and wrote some test cases to elaborate. I have attached my DUT file code and main vowel detector file code also.

Approach to the experiment:



Vowel- detector

→ I have made this K- Map to find the logical description of the output.

$I_0 I_1$ \ $I_2 I_3$	00	01	11	10
00	1 [0]	0 [1]	0 [3]	0 [2]
01	1 [4]	0 [5]	0 [7]	0 [6]
11	0 [12]	0 [13]	0 [15]	1 [14]
10	1 [8]	0 [9]	0 [11]	0 [10]

Note

$I_0 \equiv X_3$
$I_1 \equiv X_2$
$I_2 \equiv X_1$
$I_3 \equiv X_0$

→ I have written the decimal value of each square in right corner

⇒ I have written 1 at the place of vowel (which are 4 ; A, E, I, O) & 0 at other places.

so, output $Y = \underbrace{\overline{I_0}\,\overline{I_2}\,\overline{I_3}}_{0 \,\&\, 4} + \underbrace{\overline{I_1}\,\overline{I_2}\,\overline{I_3}}_{0 \,\&\, 8} + \underbrace{I_0 I_1 I_2 \overline{I_3}}_{14}$

$= (\overline{I_0}\,\overline{I_2} + \overline{I_1}\,\overline{I_2} + I_0 I_1 I_2)\,\overline{I_3}$

$= ((\overline{I_0} + \overline{I_1}) \cdot \overline{I_2} + (I_0) \cdot (I_1 I_2)) \cdot \overline{I_3}$

$= ((\overline{I_0 I_1}) \cdot \overline{I_2} + (I_0 I_1) \cdot I_2) \cdot \overline{I_3}$

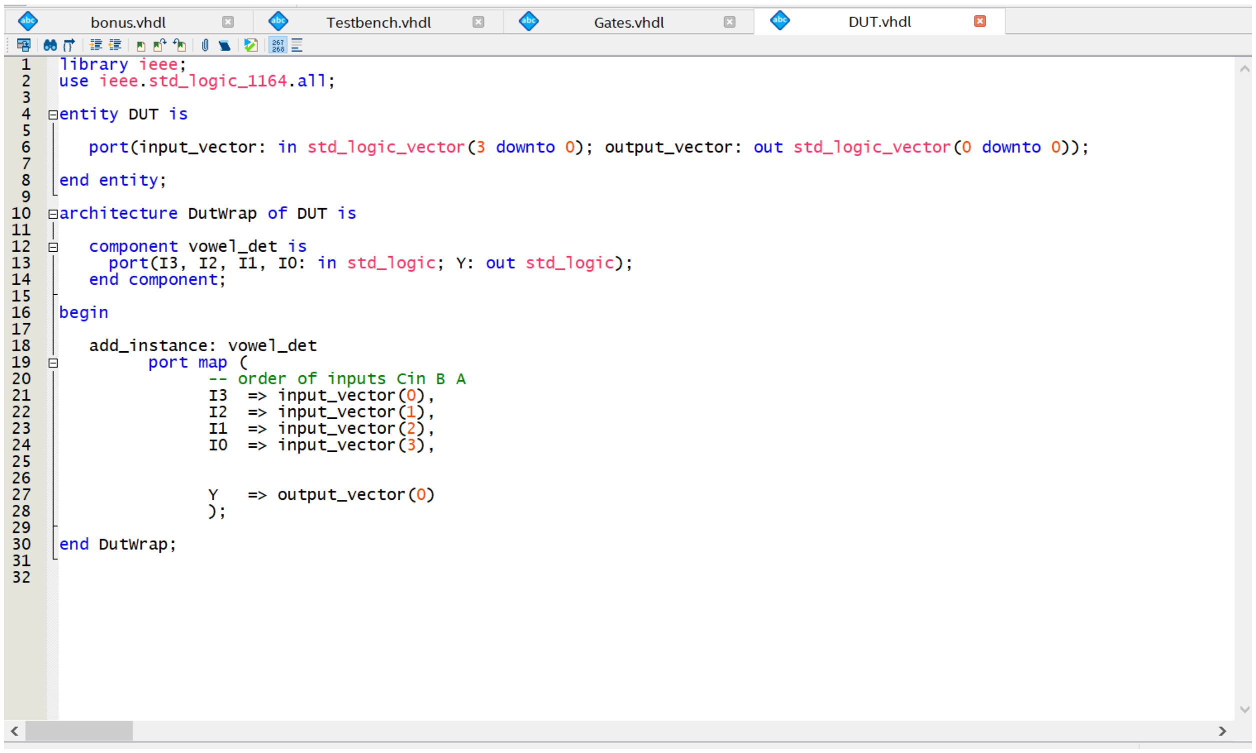
$= [(I_0 I_1)\ \mathrm{XNOR}\ (I_2)] \cdot \overline{I_3}$

final minimised output which uses 4 gates

$$\boxed{Y = [(I_0 \cdot I_1) \odot I_2] \cdot \overline{I_3}}$$

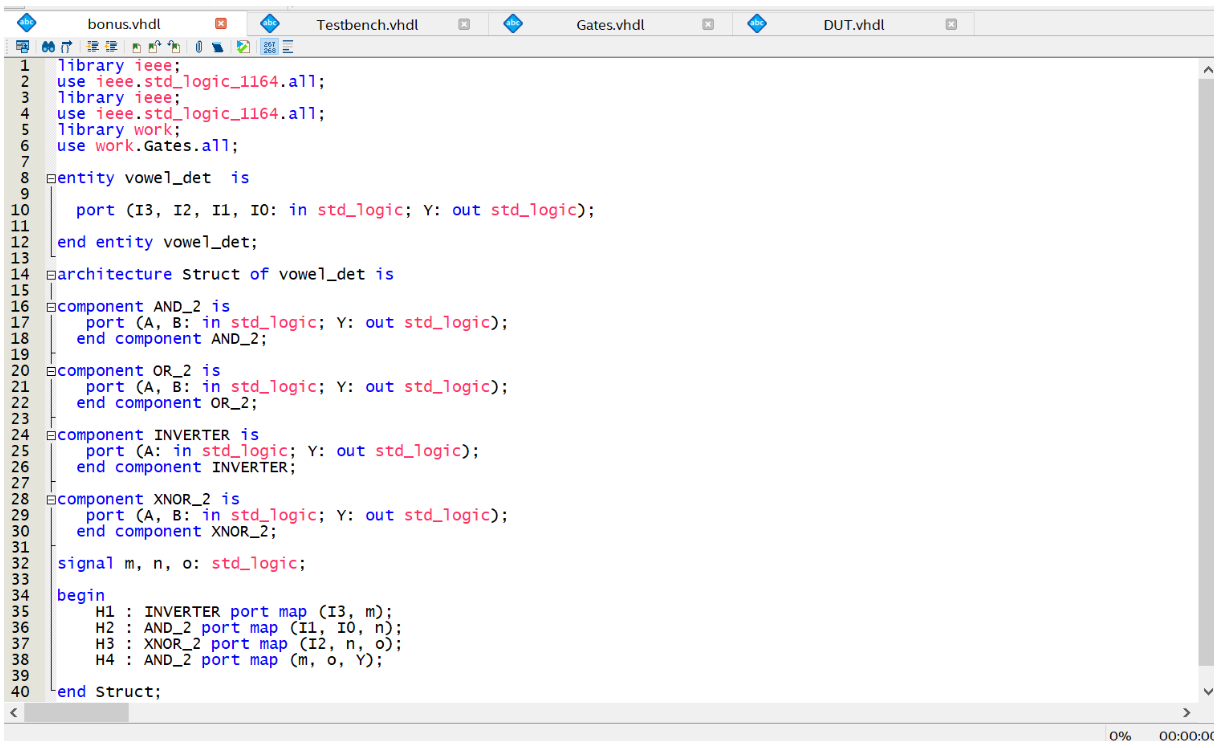
Design document and VHDL code if relevant:

Code of DUT.vhdl



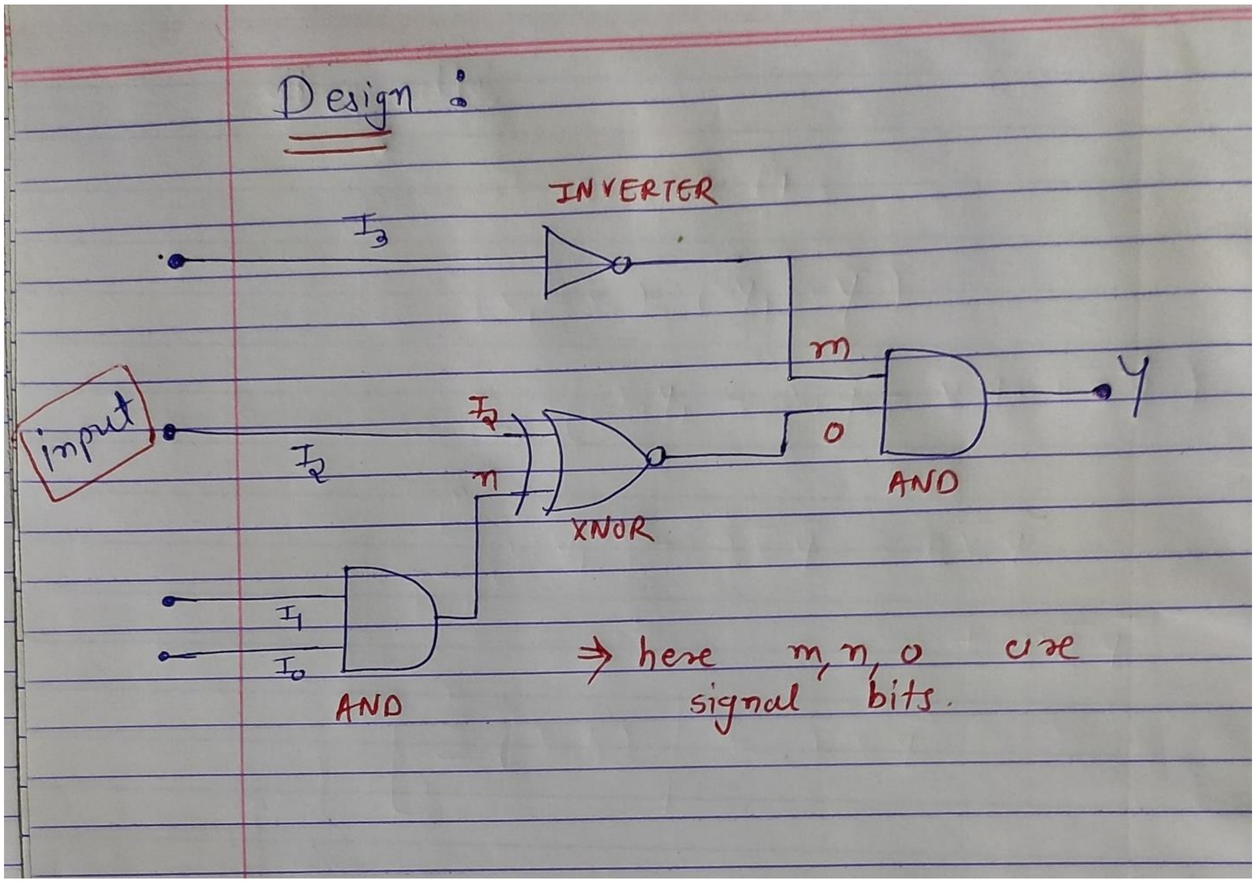
```vhdl
library ieee;
use ieee.std_logic_1164.all;

entity DUT is

   port(input_vector: in std_logic_vector(3 downto 0); output_vector: out std_logic_vector(0 downto 0));

end entity;

architecture DutWrap of DUT is

   component vowel_det is
     port(I3, I2, I1, I0: in std_logic; Y: out std_logic);
   end component;

begin

   add_instance: vowel_det
         port map (
               -- order of inputs Cin B A
               I3  => input_vector(0),
               I2  => input_vector(1),
               I1  => input_vector(2),
               I0  => input_vector(3),


               Y   => output_vector(0)
               );

end DutWrap;

```

Code of main file (bonus.vhdl)

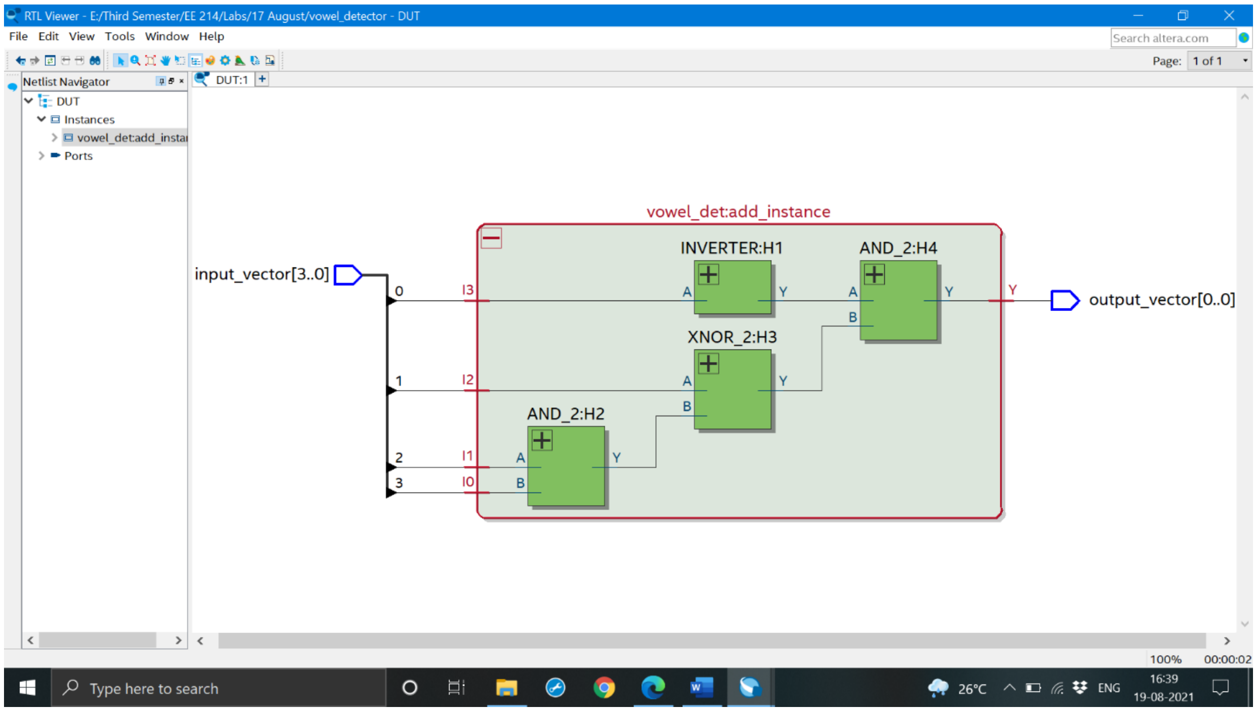


```vhdl
library ieee;
use ieee.std_logic_1164.all;
library ieee;
use ieee.std_logic_1164.all;
library work;
use work.Gates.all;

entity vowel_det  is

  port (I3, I2, I1, I0: in std_logic; Y: out std_logic);

end entity vowel_det;

architecture Struct of vowel_det is

component AND_2 is
   port (A, B: in std_logic; Y: out std_logic);
  end component AND_2;

component OR_2 is
   port (A, B: in std_logic; Y: out std_logic);
  end component OR_2;

component INVERTER is
   port (A: in std_logic; Y: out std_logic);
  end component INVERTER;

component XNOR_2 is
   port (A, B: in std_logic; Y: out std_logic);
  end component XNOR_2;

signal m, n, o: std_logic;

begin
    H1 : INVERTER port map (I3, m);
    H2 : AND_2 port map (I1, I0, n);
    H3 : XNOR_2 port map (I2, n, o);
    H4 : AND_2 port map (m, o, Y);

end Struct;
```

Design:


Design :
INVERTER
I_3
m
Y
input
I_2
I_2
n
o
XNOR
AND
I_1
I_0
AND
→ here m, n, o are signal bits.

RTL View:


RTL Viewer - E:/Third Semester/EE 214/Labs/17 August/vowel_detector - DUT
File Edit View Tools Window Help
Search altera.com
Page: 1 of 1
Netlist Navigator
DUT
Instances
vowel_det:add_insta
Ports
DUT:1
vowel_det:add_instance
input_vector[3..0]
0
1
2
3
I3
I2
I1
I0
INVERTER:H1
AND_2:H4
XNOR_2:H3
AND_2:H2
A
B
Y
output_vector[0..0]
100%
00:00:02
Type here to search
26°C
ENG
16:39
19-08-2021

DUT Input/Output Format:

Input in TRACEFILE.txt has 4 bits. The first one MSB was x3, second x2, third x1 and fourth x0. Output has 1 bit. It is Y.

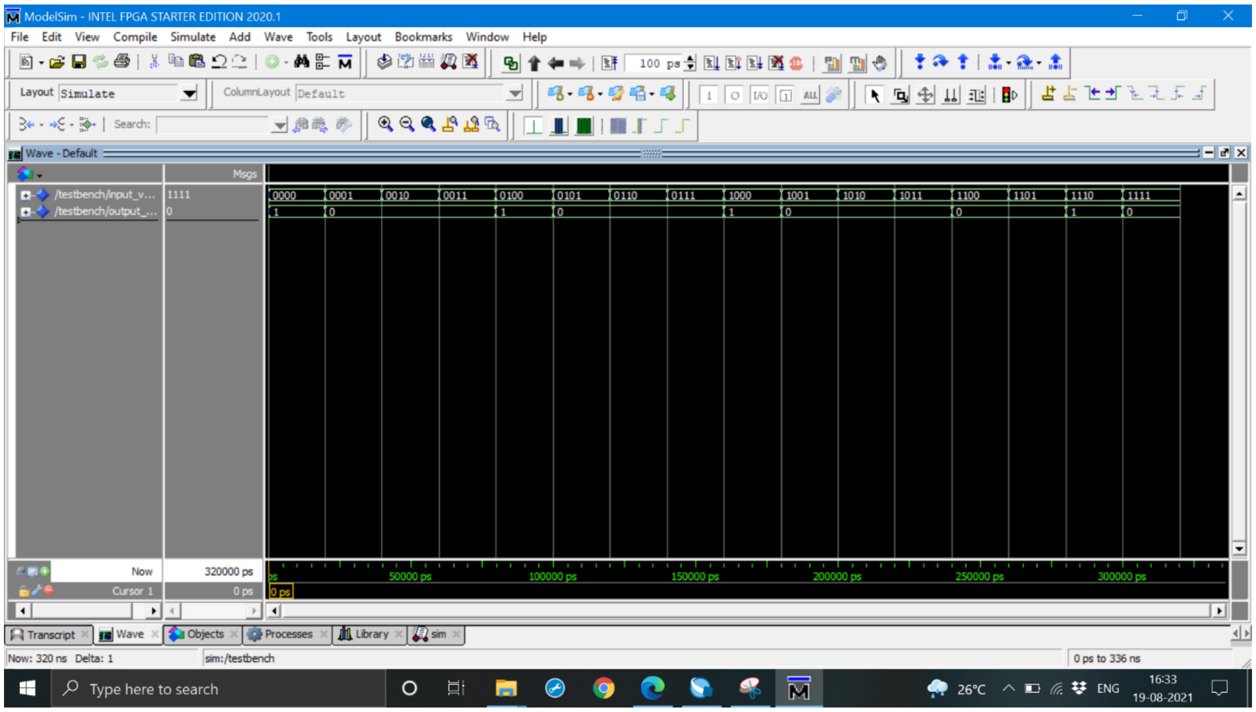
Format of TRACEFILE.txt:
<x3 x2 x1 x0> <Y> 1

Test Cases:

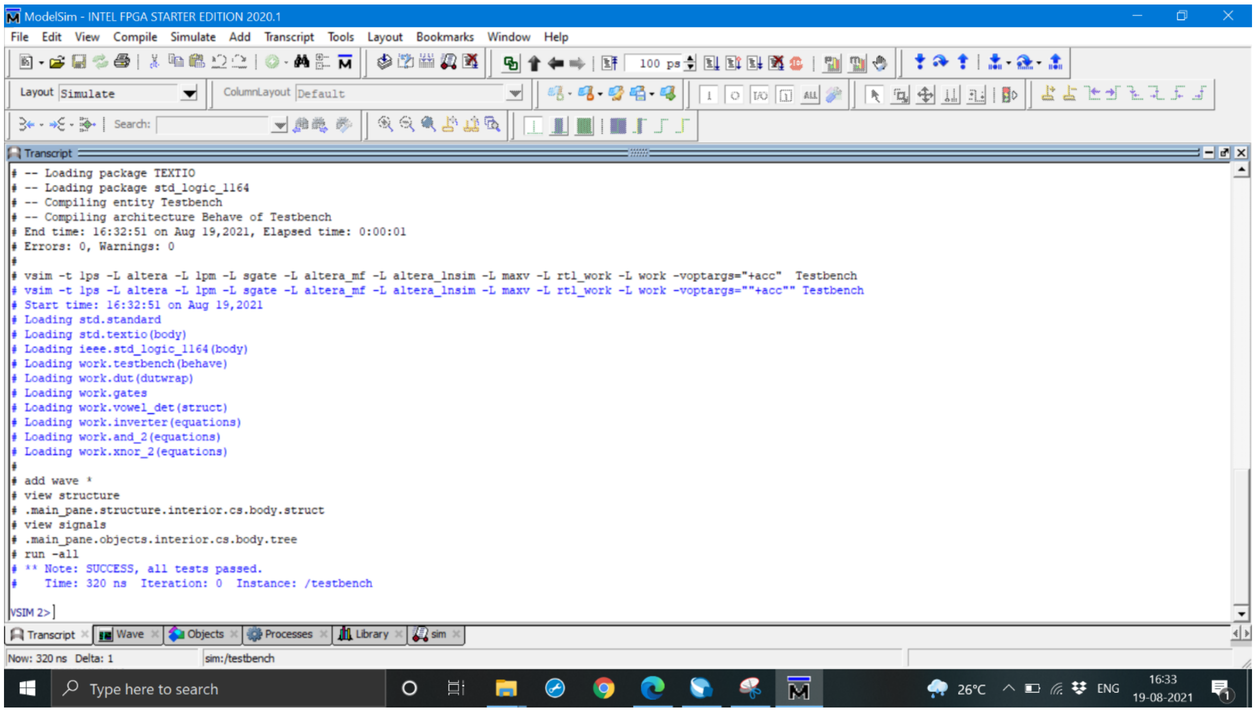
```
0000 1 1
0001 0 1
0010 0 1
0011 0 1
0100 1 1
0101 0 1
0110 0 1
0111 0 1
1000 1 1
1001 0 1
1010 0 1
1011 0 1
1100 0 1
1101 0 1
1110 1 1
1111 0 1
```

RTL Simulation:

RTL Simulation_Waveform



RTL Simulation_Transcript

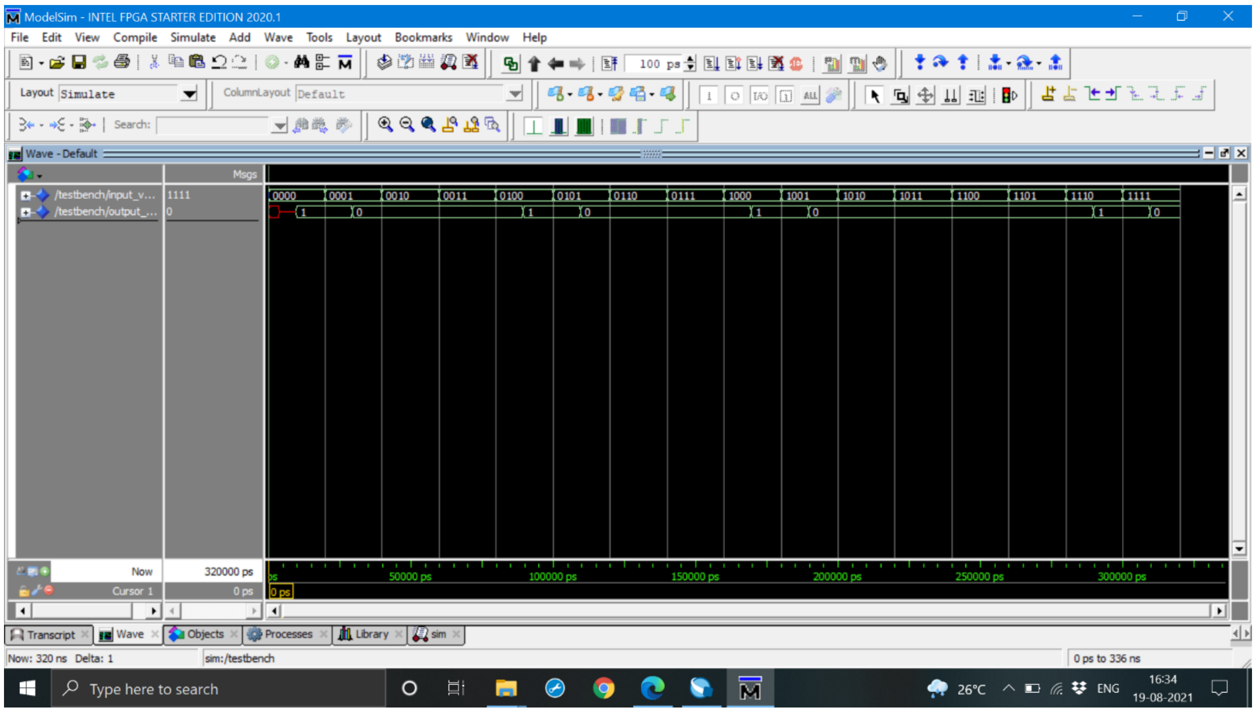


```
# -- Loading package TEXTIO
# -- Loading package std_logic_1164
# -- Compiling entity Testbench
# -- Compiling architecture Behave of Testbench
# End time: 16:32:51 on Aug 19,2021, Elapsed time: 0:00:01
# Errors: 0, Warnings: 0
#
# vsim -t 1ps -L altera -L lpm -L sgate -L altera_mf -L altera_lnsim -L maxv -L rtl_work -L work -voptargs="+acc"  Testbench
# vsim -t 1ps -L altera -L lpm -L sgate -L altera_mf -L altera_lnsim -L maxv -L rtl_work -L work -voptargs=""+acc"" Testbench
# Start time: 16:32:51 on Aug 19,2021
# Loading std.standard
# Loading std.textio(body)
# Loading ieee.std_logic_1164(body)
# Loading work.testbench(behave)
# Loading work.dut(dutwrap)
# Loading work.gates
# Loading work.vowel_det(struct)
# Loading work.inverter(equations)
# Loading work.and_2(equations)
# Loading work.xnor_2(equations)
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ** Note: SUCCESS, all tests passed.
#    Time: 320 ns  Iteration: 0  Instance: /testbench

VSIM 2>
```

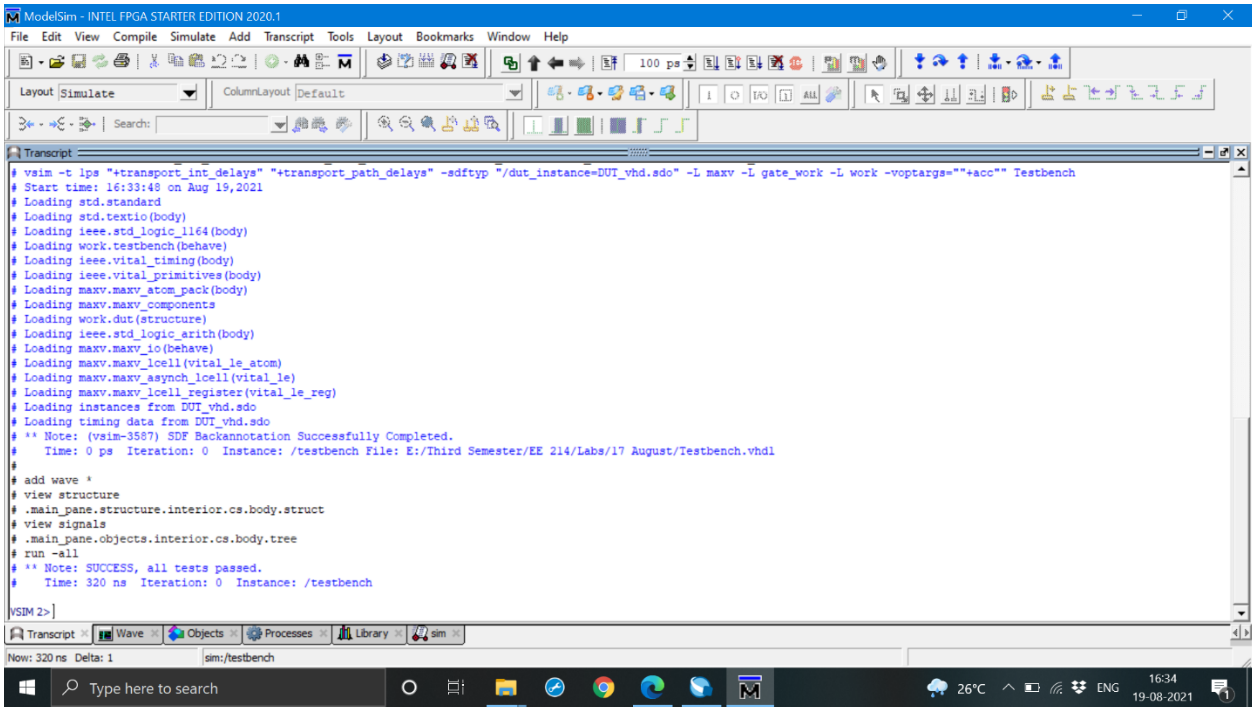
Now: 320 ns Delta: 1   sim:/testbench

Gate-level Simulation:

Gate Level Simulation_Waveform



Gate Level Simulation_Transcript



```
# vsim -t 1ps "+transport_int_delays" "+transport_path_delays" -sdftyp "/dut_instance=DUT_vhd.sdo" -L maxv -L gate_work -L work -voptargs=""+acc"" Testbench
# Start time: 16:33:48 on Aug 19,2021
# Loading std.standard
# Loading std.textio(body)
# Loading ieee.std_logic_1164(body)
# Loading work.testbench(behave)
# Loading ieee.vital_timing(body)
# Loading ieee.vital_primitives(body)
# Loading maxv.maxv_atom_pack(body)
# Loading maxv.maxv_components
# Loading work.dut(structure)
# Loading ieee.std_logic_arith(body)
# Loading maxv.maxv_io(behave)
# Loading maxv.maxv_lcell(vital_le_atom)
# Loading maxv.maxv_asynch_lcell(vital_le)
# Loading maxv.maxv_lcell_register(vital_le_reg)
# Loading instances from DUT_vhd.sdo
# Loading timing data from DUT_vhd.sdo
# ** Note: (vsim-3587) SDF Backannotation Successfully Completed.
#    Time: 0 ps  Iteration: 0  Instance: /testbench File: E:/Third Semester/EE 214/Labs/17 August/Testbench.vhdl
#
# add wave *
# view structure
# .main_pane.structure.interior.cs.body.struct
# view signals
# .main_pane.objects.interior.cs.body.tree
# run -all
# ** Note: SUCCESS, all tests passed.
#    Time: 320 ns  Iteration: 0  Instance: /testbench

VSIM 2>
```

Now: 320 ns Delta: 1   sim:/testbench

Krypton board*:

Map the logic circuit to the Krypton board and attach the images of the pin assignment and output observed on the board (switches/LEDs).

Observations*:

You must summarize your observations, either in words, using figures and/or tables.

References:

As a reference to theory of K-map part I have used Prof. BGF's slides of course EE 113, which he has taught us last year. It was very helpful to understand K-maps and their implementations. Prof. Dinesh Sharma's VHDL slides were also useful as a reference to write VHDL codes.