Experiment 3: Fibonacci Number Detector

Debasish Panda 21D070021

August 22, 2022

1 Overview of the experiment

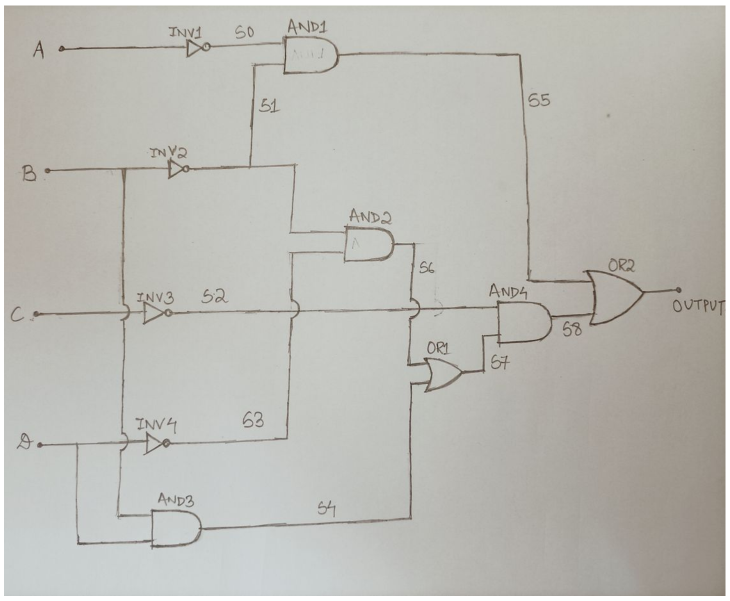
The purpose of the experiment is to design a Fibonacci Number Detector using fundamental logic gates as well as describe them in VHDL using Structural Modelling. Following a successful RTL simulation, we have to demonstrate the same on Xen10 FPGA Board.
In order to perform the experiment, first the Detector is designed using AND, OR and INVERTER gates. A rough sketch of the digital circuit is drawn first. The appropriate code for Structural Modelling of these components is written using VHDL on Quartus Prime software. Next, we have to demonstrate the Detector practically by using the Xen10 FPGA board. In order to run the VHDL code on the board, we also need to determine the input and output pins correctly as per the specifications of the circuit board.
In this report first we discuss the setup required for the experiment and the approach used by me to solve the assignment. Next, I put forward the observations of the experiment in the form of truth table for the Detector. Attached alongwith are the pen-and-paper diagram and picture showing the simulation output of the design.

2 Experimental setup

First we need to design the Fibonacci Detector using logic gates. In order to do so,we need to solve for its logical expression using K-maps. On doing so, one obtains the required logical expression as:

$$\text{OUTPUT} = \bar{A}\bar{B} + B\bar{C}D + \bar{B}\bar{C}\bar{D}$$

The pen-and-paper diagram of the Fibonacci Number Detector is:



Fibonacci Number Detector

Now that we have the required circuit, we need to assign various component instances, architecture and signals as per VHDL syntax. The required code for the same is discussed in the next section.

2.1 Design Documentation and VHDL Code

The VHDL code for designing the entity-architecture pairs associated with the digital circuit is as follows:

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

entity Fibonacci_Detector is
```

```
 port (A, B, C, D: in std_logic; OUTPUT: out std_logic);
end entity Fibonacci_Detector;

architecture Struct of Fibonacci_Detector is
 signal S0, S1, S2, S3, S4, S5, S6, S7, S8: std_logic;

for all: AND_2
use entity work.AND_2(Equations);

for all: OR_2
use entity work.OR_2(Equations);

for all: INVERTER
use entity work.INVERTER(Equations);

begin
  INV1: INVERTER port map (A => A, Y => S0);
  INV2: INVERTER port map (A => B, Y => S1);
  INV3: INVERTER port map (A => C, Y => S2);
  INV4: INVERTER port map (A => D, Y => S3);
  AND1: AND_2 port map (A => S0, B => S1, Y => S5);
  AND2: AND_2 port map (A => S1, B => S3, Y => S6);
  AND3: AND_2 port map (A => B, B => D, Y => S4);
  AND4: AND_2 port map (A => S2, B => S7, Y => S8);
  OR1: OR_2 port map (A => S6, B => S4, Y => S7);
  OR2: OR_2 port map (A => S5, B => S8, Y => OUTPUT);
end Struct;
```

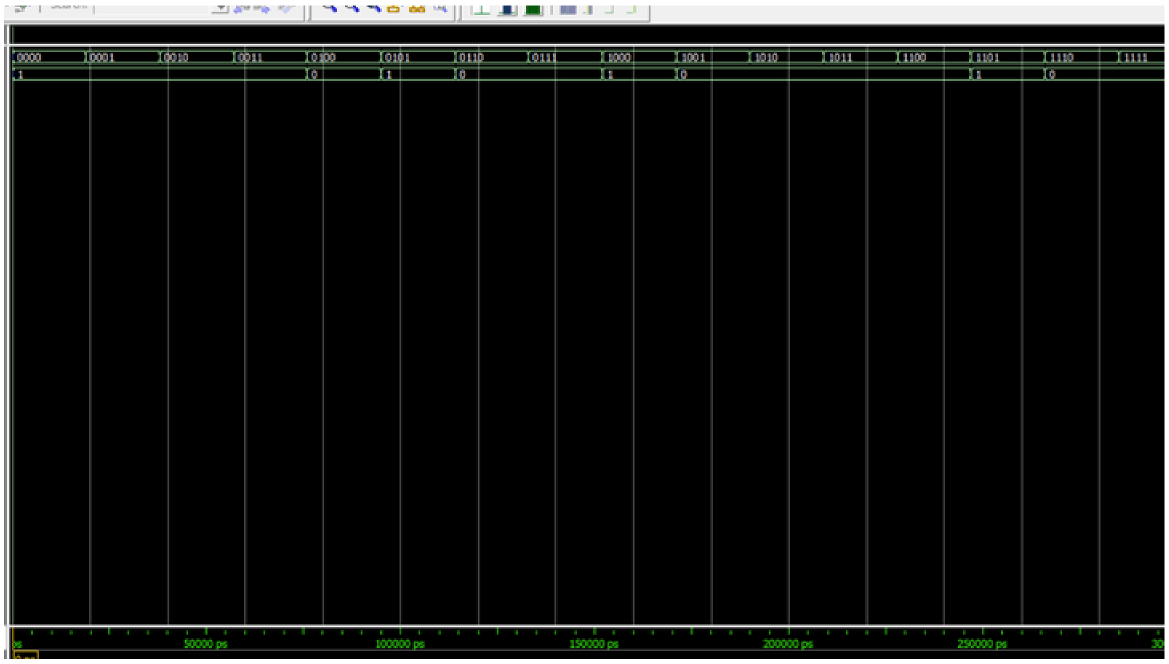
3 Observations

The experiment was conducted successfully, and the expected output was obtained through the RTL Simulation. The expected output of the Detector is shown in the form of the truth table below, alongwith the final RTL Simulation result of the design.

A	B	C	D	Y
0	0	0	0	1
0	0	0	1	1
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	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	0
1	1	1	1	0

Table 1: Fibonacci Number Detector truth table

0



RTL Simulation of Fibonacci Number Detector

Following this, we have to do the pin planning correctly and convert our VHDL file to a .svf file, which is executable by the Xen10 board.

4 References

[1] EE214 Github page

[2] Overleaf LaTeX tutorial for beginners

[3] Introduction to Xen10 Board