Experiment 1: Logic Function Design using NAND gate

Debasish Panda 21D070021

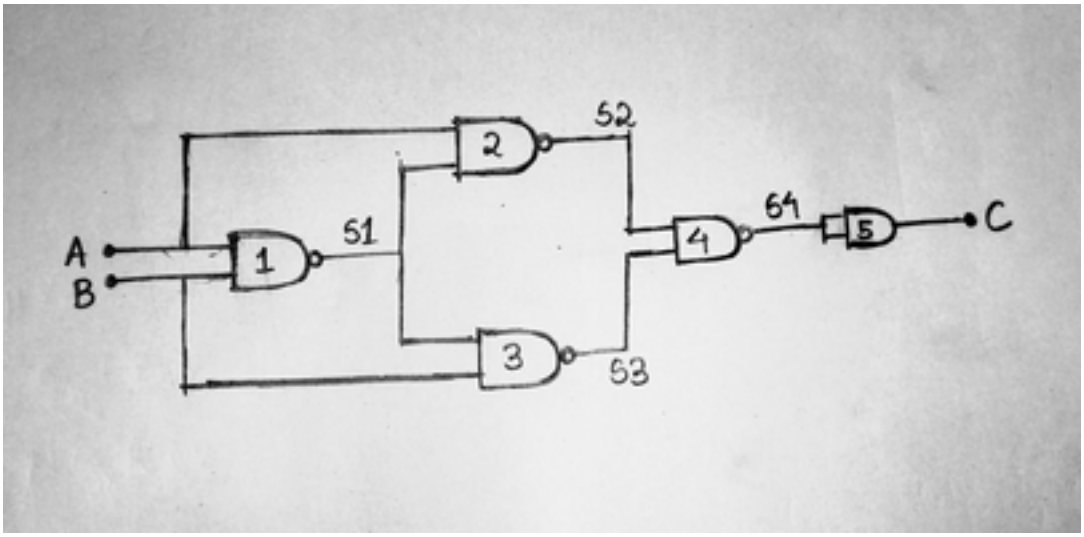
August 9, 2022

1 Overview of the experiment

The purpose of the experiment is to design a XNOR gate and a Full Subtractor using NAND gates as well as describe them in VHDL using Structural Modelling.
In order to perform the experiment, first a XNOR gate is designed using NAND gates only. A rough sketch of the digital circuit is drawn first. For designing the Full Subtractor, first a Half Subtractor and an OR gate is designed using NAND gates only. The appropriate code for Structural Modelling of these components is written using VHDL on Quartus Prime software.
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 tables for both the XNOR and the Full Subtractor. Attached alongwith are the pen-and-paper diagrams and pictures showing the simulation output of the two designs.

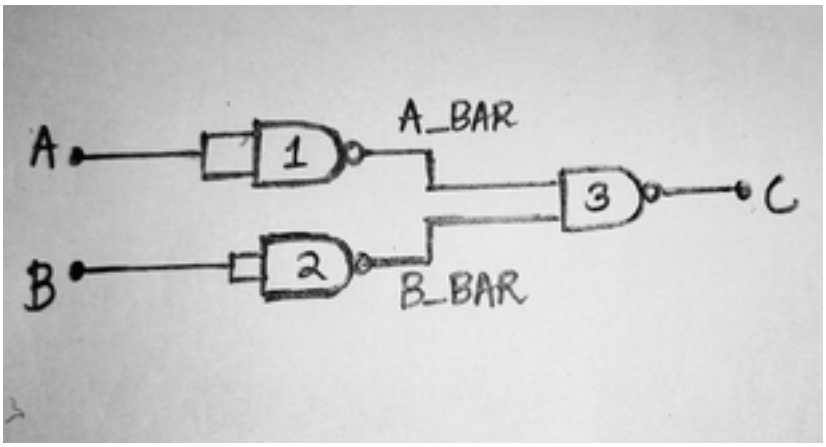
2 Experimental setup

First we need to design a XNOR gate using NAND gates. The XNOR gate thus designed can be easily converted to the XOR gate which will be further used in the Half- and Full-Subtractor. The pen-and-paper diagram of a XNOR gate is:



XNOR Gate using NAND Gates

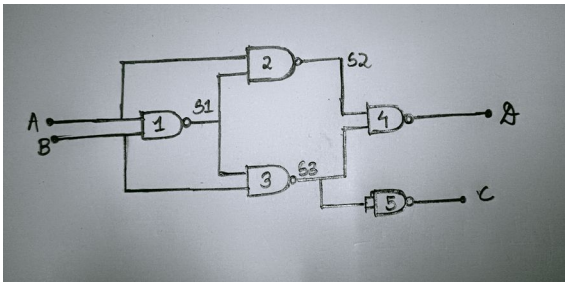
Additionally, we also need to design an OR gate using NAND gates which will be used while designing the Full Subtractor. The pen-and-paper diagram of the OR gate is:



OR Gate using NAND Gates

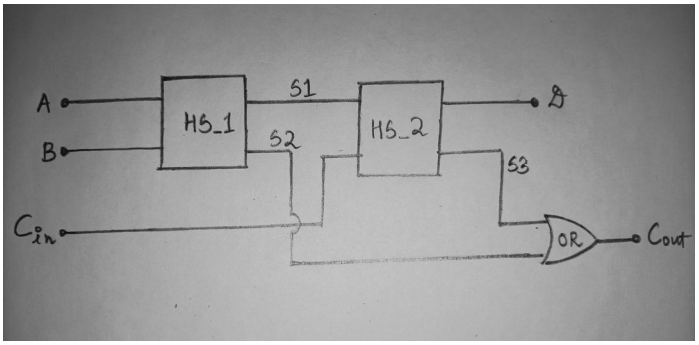
The function of a Half Subtractor is to take in two bits and subtract them, giving us a two-bit output of difference and carry. The difference bit

of the Half Subtractor is given by $A \oplus B$ and the borrow bit is given by $\bar{A}B$. The pen-and-paper diagram of the Half Subtractor is as follows:



Half Subtractor using NAND Gates

The Full Subtractor is similar in construction to the Full Adder, with the only change being made that the Half Adders are replaced by Half Subtractors. The function of the Full Subtractor is to take in three bits, subtract the last two bits from the MSB, and give us a two-bit output of difference and carry. The pen-and-paper diagram of the Full Subtractor using its constituent elements is as follows:



Full Subtractor using constituent elements

Now that we have the required circuits, we need to assign various component instances, architectures and signals to individual circuits 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:

2.1.1 XNOR using NAND Gates

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

library work;
use work.Gates.all;

entity XNOR_GATE is
 port (A, B: in std_logic; OUTPUT: out std_logic);
end entity XNOR_GATE;

architecture unique of XNOR_GATE is
 signal S1, S2, S3, S4: std_logic;

for all: NAND_2
use entity work.NAND_2(Equations);

begin
 NAND1: NAND_2 port map (A => A, B => B, Y => S1);
 NAND2: NAND_2 port map (A => A, B => S1, Y => S2);
 NAND3: NAND_2 port map (A => S1, B => B, Y => S3);
 NAND4: NAND_2 port map (A => S2, B => S3, Y => S4);
 NAND5: NAND_2 port map (A => S4, B => S4, Y => OUTPUT);
end unique;
```

In the code, first we have to declare which libraries we intend to use for our experiment. Following it are the entity declaration for the XNOR Gate and internal architecture of the gate. The interconnections are represented by signals, as per VHDL syntactical rules.

2.1.2 Full Subtractor using NAND Gates

```
library ieee;
```

```
use ieee.std_logic_1164.all;

library work;
use work.Gates.all;

entity XNOR_GATE is
 port (A, B: in std_logic; OUTPUT: out std_logic);
end entity XNOR_GATE;

architecture Unique of XNOR_GATE is
 signal S1, S2, S3, S4: std_logic;

for all: NAND_2
use entity work.NAND_2(Equations);

begin
 NAND1: NAND_2 port map (A => A, B => B, Y => S1);
 NAND2: NAND_2 port map (A => A, B => S1, Y => S2);
 NAND3: NAND_2 port map (A => S1, B => B, Y => S3);
 NAND4: NAND_2 port map (A => S2, B => S3, Y => OUTPUT);
 --NAND5: NAND_2 port map (A => S4, B => S4, Y => OUTPUT);
end Unique;

-------------------------------------------------------------------

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

entity NOR_GATE is
 port (A, B: in std_logic; OUTPUT: out std_logic);
end entity NOR_GATE;

architecture Struct of NOR_GATE is
 signal A_BAR, B_BAR, S : std_logic;

for all: NAND_2
```

```
use entity work.NAND_2(Equations);

begin
  NAND1: NAND_2 port map (A => A, B => A, Y => A_BAR);
  NAND2: NAND_2 port map (A => B, B => B, Y => B_BAR);
  NAND3: NAND_2 port map (A => A_BAR, B => B_BAR, Y => OUTPUT);
  --NAND4: NAND_2 port map (A => S, B => S, Y => OUTPUT);
end Struct;

-------------------------------------------------------------------

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

entity Half_Borrow is
 port (A, B: in std_logic; P: out std_logic);
end entity Half_Borrow;

architecture Easy of Half_Borrow is

signal A1, A2: std_logic;

for all: NAND_2
use entity work.NAND_2(Equations);

begin
 NAND1: NAND_2 port map (A => A, B => B, Y => A1);
 NAND2: NAND_2 port map (A => A1, B => B, Y => A2);
 NAND3: NAND_2 port map (A => A2, B => A2, Y => P);
end Easy;

-------------------------------------------------------------------

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

```
use work.Gates.all;

entity HS is
 port (A, B: in std_logic; T, C1: out std_logic);
end entity HS;

architecture Simple of HS is

component XNOR_GATE is
 port (A, B: in std_logic; OUTPUT: out std_logic);
end component XNOR_GATE;

component Half_Borrow is
 port (A, B: in std_logic; P: out std_logic);
end component Half_Borrow;

for all: XNOR_GATE
use entity work.XNOR_GATE(Unique);

for all: Half_Borrow
use entity work.Half_Borrow(Easy);

begin
  XNOR1: XNOR_GATE port map (A => A, B => B, OUTPUT => T);
  Half_Borrow1: Half_Borrow port map (A => A, B => B, P => C1);
end Simple;

--------------------------------------------------------------------

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

entity FS is
 port (A, B, C_i: in std_logic; S, C:out std_logic);
end entity FS;
```

```
architecture Complex of FS is

signal S1, S2, S3: std_logic;

component HS is
 port (A, B: in std_logic; T, C1: out std_logic);
end component HS;

component NOR_GATE is
 port (A, B: in std_logic; OUTPUT: out std_logic);
end component NOR_GATE;

for all: HS
use entity work.HS(Simple);

for all: NOR_GATE
use entity work.NOR_GATE(Struct);

begin
 HS1: HS port map (A => A, B => B, T => S1, C1 => S2);
 HS2: HS port map (A => S1, B => C_i, T => S, C1 => S3);
 NOR1: NOR_GATE port map (A => S2, B => S3, OUTPUT => C);
end Complex;
```

[**N.B.**: Although the names of some of the entities have been given as NOR_GATE and XNOR_GATE, they actually represent OR Gate and XOR Gate, respectively. This misrepresentation is due to some last-minute changes made in the original code.]

The VHDL code required for designing the Full Subtractor is quite lengthy since many of the components associated with the Full Subtractor have to be designed in here. The first section of the code deals with the declaration of a XOR Gate. The second part of the code designates an OR Gate while the third part of the code is for designing the logic gate for finding borrow in a Half Subtractor. The fourth part of the code deals with the design of a Half Subtractor using previously declared components. The final part of the code is the design of the Full Subtractor using all of these components.

3 Observations

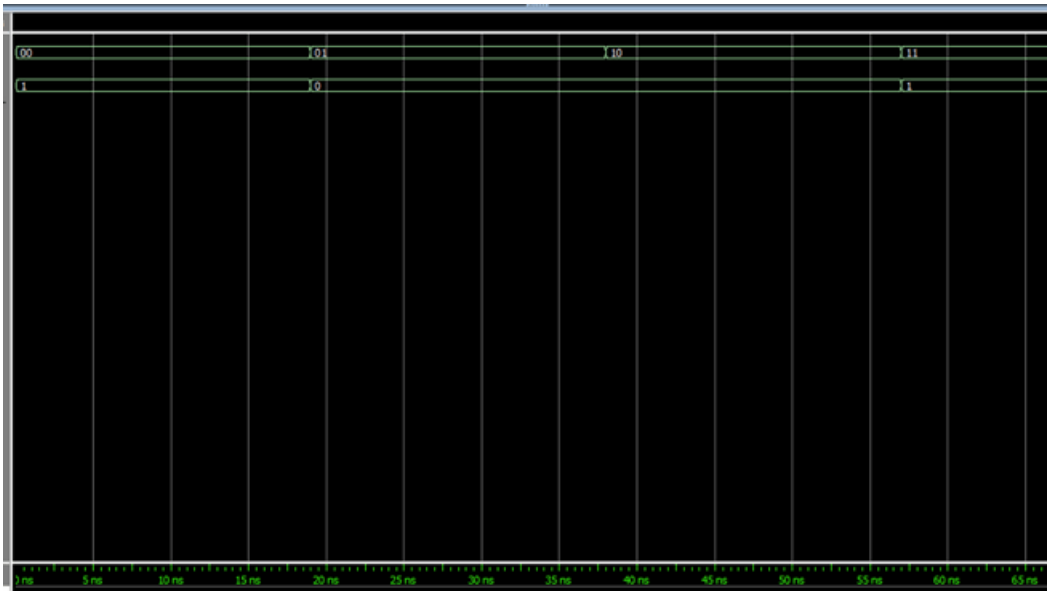
The experiment was conducted successfully, and the expected output was obtained through the RTL Simulation. The expected output of the XNOR Gate and the Full Subtractor is shown in the form of the truth table below, alongwith the final RTL Simulation results of the two designs.

A	B	C
0	0	1
0	1	0
1	0	0
1	1	1

Table 1: XNOR Gate truth table

A	B	C_{in}	D	C_{out}
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

Table 2: Full Subtractor truth table



RTL Simulation of XNOR Gate

RTL Simulation of Full Subtractor

4 References

[1] EE214 Github page

[2] Overleaf LaTeX tutorial for beginners