



INSTITUTE OF ENGINEERING , CENTRAL CAMPUS,PULCHOWK

EMBEDDED SYSTEM

LAB #4

---

## Combinational Logic Design Using VHDL

---

**Submitted BY:**

Amrit Prasad Phuyal

Roll: PULL074BEX004

**Submitted To:**

Department of Electronics and  
Computer Engineering

December 4, 2020

Contents

1	Title	1
2	Objective	1
3	Requirement	1
4	Introduction	1
	4.0.1 Features of VHDL . . . . .	1
5	LAB Problems	2
	5.1 Question -1 . . . . .	2
	5.2 Question -2 . . . . .	5
	5.3 Question -3 . . . . .	10
	5.4 Question -4 . . . . .	13
	5.5 Question -5 . . . . .	19
	5.6 Question -6 . . . . .	22
	5.7 Question -7 . . . . .	26
6	Conclusion	30

## List of Figures

---

### Problem 1

---

1	Combinational Circuit for . . . . .	2
2	Simulation Waveform . . . . .	4
3	RTL Schematic . . . . .	5

---

### Problem 2

---

4	BCD to Gray Code K map and Solution for . . . . .	6
5	Simulation Waveform . . . . .	8
6	RTL Schematic . . . . .	9

---

### Problem 3

---

7	Simulation Waveform for Problem 3 . . . . .	12
8	RTL Schematic for Problem 3 . . . . .	13

---

### Problem 4

---

9	Truth table Problem 4 . . . . .	14
10	SOP and POS K-map reduction F1 and F2 . . . . .	14
11	Simulation Waveform SOP . . . . .	17
12	RTL Schematic SOP . . . . .	17
13	Simulation Waveform POS . . . . .	19
14	RTL Schematic POS . . . . .	19

---

### Problem 5

---

15	Truth Table 2:1 multiplexer . . . . .	20
16	Simulation Waveform . . . . .	21
17	RTL Schematic . . . . .	22

---

### Problem 6

---

18	MUX using three 2:1 MUX as basic building blocks . . . . .	23
19	Simulation Waveform 0 ns to 700 ns . . . . .	24
20	Simulation Waveform 700 ns to 1300 ns . . . . .	24
21	Simulation Waveform 1300 ns to 1900 ns . . . . .	25
22	Simulation Waveform 1900 ns to 2600 ns . . . . .	25
23	RTL Schematic . . . . .	25

---

**Problem 7**

---

24	Simulation Waveform 4 bit Adder . . . . .	27
25	Simulation Waveform for 4 bit Subtractor . . . . .	28
26	RTL Schematic . . . . .	29

## List of VHDL Codes

---

### Problem 1

---

1	Dataflow model . . . . .	2
2	AND gate . . . . .	3
3	OR gate . . . . .	3
4	AND with one inverted variable . . . . .	3
5	Behavioral model . . . . .	3
6	Structural model . . . . .	4
7	Testbench for all cases . . . . .	4
8	Dataflow model . . . . .	5

---

### Problem 2

---

9	Behavioral model . . . . .	7
10	XOR gate . . . . .	7
11	Structural model . . . . .	7
12	Testbench for all cases . . . . .	8

---

### Problem 3

---

13	Dataflow model . . . . .	10
14	Behavioral model . . . . .	11
15	NOT gate using only NOR . . . . .	11
16	3-input OR gate using only NOR . . . . .	11
17	3-input AND gate using only NOR . . . . .	12
18	Structural model . . . . .	12
19	Testbench for all cases . . . . .	12

---

### Problem 4

---

#### SOP

20	Dataflow model SOP . . . . .	15
21	Behavioral model SOP . . . . .	15
22	NOT gate implementation using only NOR . . . . .	15
23	3-input OR gate implementation using only NOR . . . . .	15
24	3-input AND gate implementation using only NOR . . . . .	16
25	Structural model SOP . . . . .	16
26	Testbench for all possible cases SOP . . . . .	17

#### POS

27	Dataflow model POS . . . . .	17
28	Behavioral model POS . . . . .	18
29	Structural model POS . . . . .	18
30	Testbench for all possible cases POS . . . . .	19

---

**Problem 5**


---

31	2:1 MUX implementation using WHEN-ELSE statement . . . . .	20
32	2:1 MUX implementation using IF-THEN-ELSE statement . . . . .	20
33	Testbench for all possible cases . . . . .	21

---

**Problem 6**


---

34	2:1 MUX implementation using WHEN-ELSE statement . . . . .	23
35	4:1 MUX implementation using three 2:1 MUX: Structural model . . . . .	23
36	Testbench for all possible cases . . . . .	24

---

**Problem 7**


---

37	XOR gate implementation . . . . .	26
38	Full adder implementation . . . . .	26
39	4-bit adder/subtractor implementation using four 1-bit full-adder: Structural model . . . . .	27
40	Testbench for all possible cases . . . . .	27

# 1 Title

Combinational Logic Design Using VHDL

## 2 Objective

To enable us to write VHDL code for a Field Programmable Gate Array (FPGA) capable of:

- Implementing combinational circuits
- Implementing test benches to verify the working of combinational circuits

## 3 Requirement

**Hardware:**

- Spartan-3E or Spartan-3AN FPGA starter kit
- Power cable and Data cable

**Software:**

- Xilinx ISE (Integrated Synthesis Environment) Design Suite
- iMPACT configuration tool

## 4 Introduction

VHDL stands for Very High Speed Integrated Circuit (VHSIC) Hardware Descriptive Language. It is one of the programming languages which is used to model the digital circuits by using different style of modelling such as dataflow, behavioral and structural. It is an event driven language, it means whenever an event occurs on signals in VHDL, it triggers the execution of a statement. It allows both concurrent as well as sequential modelling. It is case-insensitive and is a strongly typed language, that is, it does not support implicit conversion between data types. It supports code reusability and code sharing via packages and user defined libraries. In VHDL, an entity is used to describe a hardware module. An entity can be described using,

- Entity declaration
- Architecture
- Configuration
- Package Declaration
- Package Body

### 4.0.1 Features of VHDL

- It is a hardware descriptive language used for design entry and simulation of digital circuits.
- It is an event-driven language: i.e. whenever an event occurs on signals in VHDL, it triggers the execution of a statement.
- It allows both concurrent as well as sequential modelling.
- It gives the flexibility to define data types that are specific to user needs apart from predefined types.
- It supports code reusability and code sharing via packages and user defined libraries.
- It is case-insensitive i.e. it does not differentiate between lowercase and uppercase letters.
- It is strongly typed language i.e. it does not support implicit conversion between data types.

## 5 LAB Problems

### 5.1 Question -1

Write VHDL code to implement the logic circuit shown in below figure, which has 4 inputs (x1, x2, x3 and x4) and one output (f).

- Provide the following architectural styles
  1. Dataflow Style
  2. Behavioral Style
  3. Structural Style
- Write a VHDL test bench to verify the operation of the logic circuit.
- Provide a simulation waveform depicting all possible input cases.

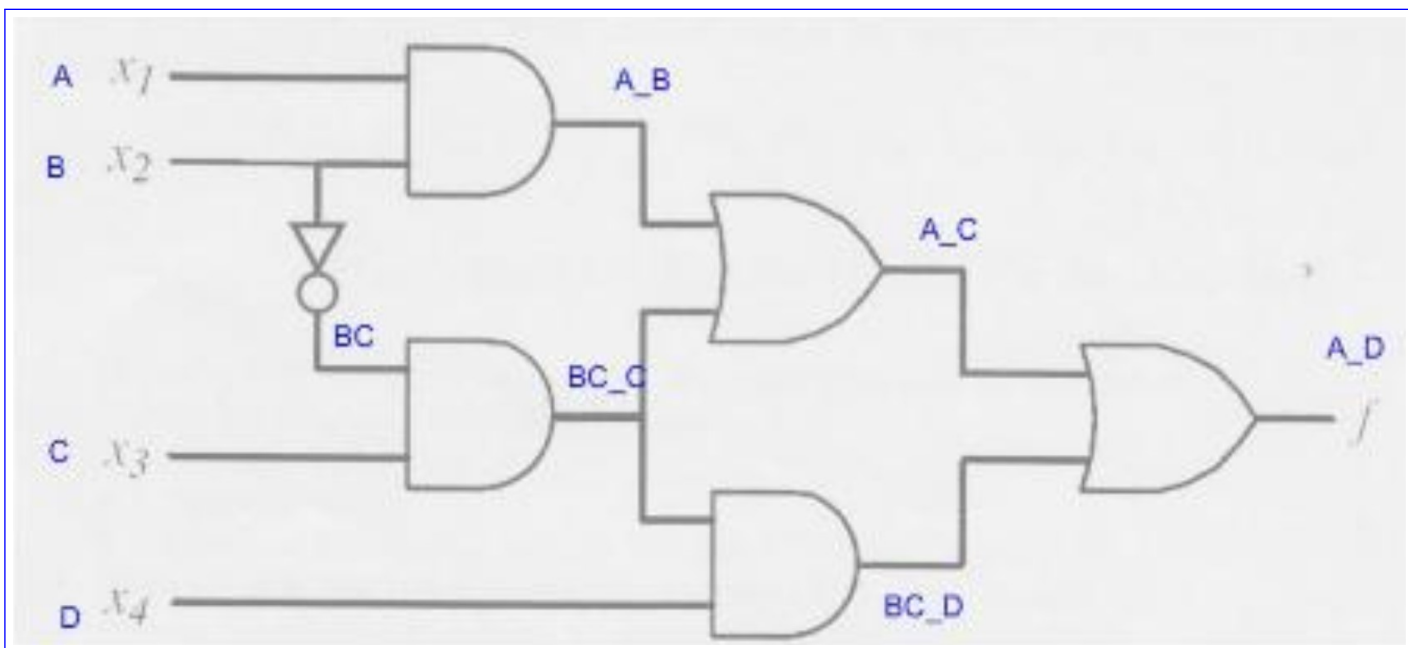


Figure 1: Combinational Circuit for

↓ q1-df.vhd ↓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY q1 IS PORT (
5     A, B, C, D : IN STD_LOGIC;
6     F : OUT STD_LOGIC
7 );
8
9
10 ARCHITECTURE dataflow OF q1 IS
11 BEGIN
12     F <= (A AND B) OR (NOT B AND C);
13 END dataflow;
```

Code 1: Dataflow model

↓ and1.vhd ↓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY and1 IS PORT (
5     i1, i2 : IN STD_LOGIC;
6     o1 : OUT STD_LOGIC
7 );
8 END and1;
```



```

9
10 ARCHITECTURE dataflow OF and1 IS
11 BEGIN
12     o1 <= i1 AND i2;
13 END dataflow;

```

Code 2: AND gate

---



---

⇓ or1.vhd ⇓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY or1 IS PORT (
5     i1,i2: IN STD_LOGIC;
6     o1: OUT STD_LOGIC
7 );
8 END or1;
9
10 ARCHITECTURE dataflow OF or1 IS
11 BEGIN
12     o1 <= i1 OR i2;
13 END dataflow;

```

Code 3: OR gate

---



---

⇓ andnot.vhd ⇓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY andnot IS PORT (
5     i1, i2 : IN STD_LOGIC;
6     o1 : OUT STD_LOGIC
7 );
8 END andnot;
9
10 ARCHITECTURE dataflow OF andnot IS
11 BEGIN
12     o1 <= NOT i1 AND i2;
13 END dataflow;

```

Code 4: AND with one inverted variable

---



---

⇓ q1-be.vhd ⇓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY q1 IS PORT (
5     A, B, C, D : IN STD_LOGIC;
6     F : OUT STD_LOGIC
7 );
8 END q1;
9
10 ARCHITECTURE behavioral OF q1 IS
11     SIGNAL F1, F2, F3, F4 : STD_LOGIC;
12
13 BEGIN
14     be_proc : PROCESS (A, B, C, D, F1, F2, F3, F4)
15
16     BEGIN
17         F1 <= A AND B;
18         F2 <= NOT B AND C;
19         F <= F1 OR F2;
20     END PROCESS be_proc;
21
22 END behavioral;

```

Code 5: Behavioral model

---



---

⇓ q1-st.vhd ⇓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY q1 IS PORT (
5     A, B, C, D : IN STD_LOGIC;
6     F : OUT STD_LOGIC
7 );
8 END q1;
9
10 ARCHITECTURE structural OF q1 IS
11     SIGNAL F1, F2 : STD_LOGIC;
12
13     COMPONENT and1 IS PORT (
14         i1, i2 : IN STD_LOGIC;
15         o1 : OUT STD_LOGIC
16     );
17 END COMPONENT;
18 COMPONENT andnot IS PORT (

```

```

19     i1, i2 : IN STD_LOGIC;
20     o1 : OUT STD_LOGIC
21 );
22 END COMPONENT;
23
24 COMPONENT or1 IS PORT (
25     i1, i2 : IN STD_LOGIC;
26     o1 : OUT STD_LOGIC
27 );
28 END COMPONENT;
29
30 BEGIN
31     C1 : and1 PORT MAP(i1 => A, i2 => B, o1 =>
32         F1);
33     C2 : andnot PORT MAP(i1 => B, i2 => C, o1
34         => F2);
35     C3 : or1 PORT MAP(i1 => F1, i2 => F2, o1 =>
36         F);
37 END structural;

```

Code 6: Structural model

↓ q1-tb.vhd ↓

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.NUMERIC_STD.ALL;
4
5  ENTITY q1_tb IS
6  END q1_tb;
7
8  ARCHITECTURE behavioral OF q1_tb IS
9      -- component declaration for the Unit
10     Under Test (UUT)
11     COMPONENT q1
12     PORT (
13         A : IN STD_LOGIC;
14         B : IN STD_LOGIC;
15         C : IN STD_LOGIC;
16         D : IN STD_LOGIC;
17         F : OUT STD_LOGIC
18     );
19     END COMPONENT;
20     SIGNAL input_vector : STD_LOGIC_VECTOR(3
21         DOWNT0 0) := "0000";
22     SIGNAL output : STD_LOGIC;
23
24 BEGIN
25     --INSTANTIATE the unit under test
26     uut : q1 PORT MAP(
27         A => input_vector(3),
28         B => input_vector(2),
29         C => input_vector(1),
30         D => input_vector(0),
31         F => output
32     );
33
34     --stimulus process
35     stim_proc : PROCESS
36     BEGIN
37         FOR index IN 0 TO 15 LOOP
38             input_vector <= STD_LOGIC_VECTOR(
39                 to_unsigned(index, 4));
40             WAIT FOR 50 ns;
41         END LOOP;
42     END PROCESS;
43 END behavioral;

```

Code 7: Testbench for all cases

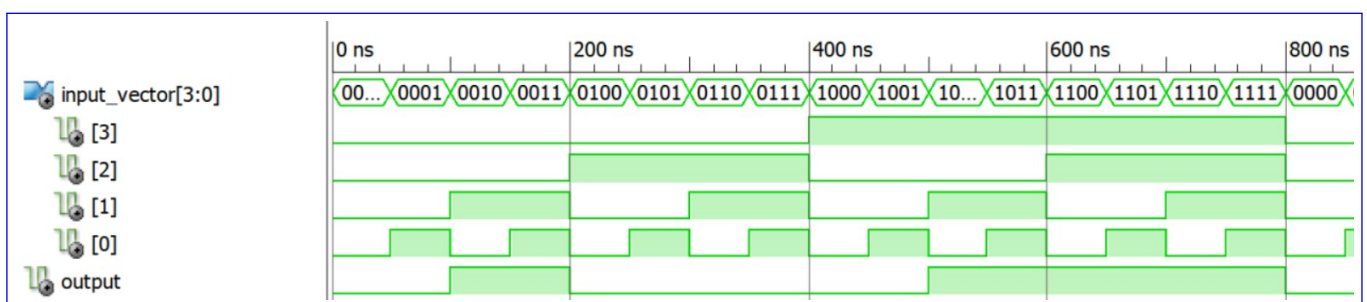


Figure 2: Simulation Waveform

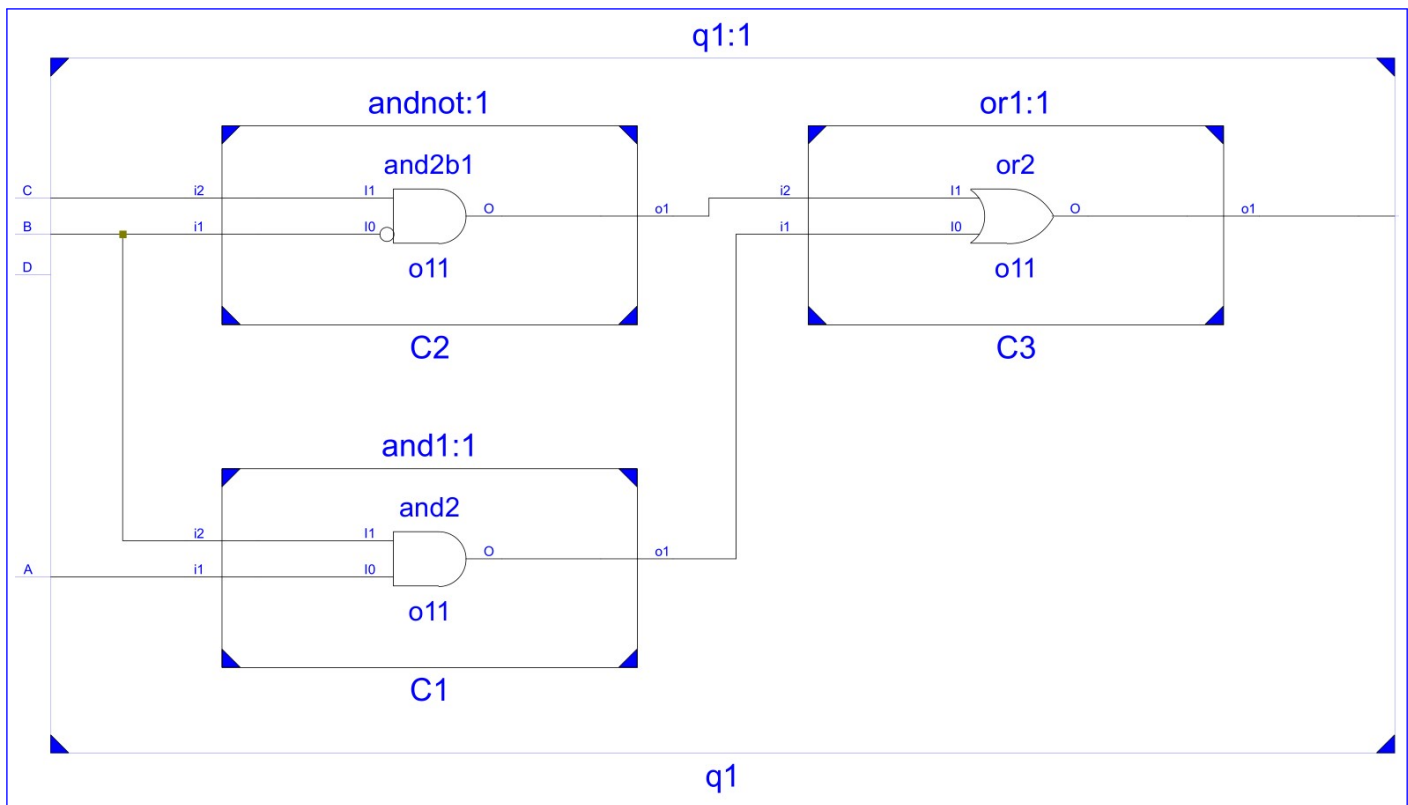


Figure 3: RTL Schematic

## 5.2 Question -2

Write VHDL code to design a logic circuit that implements the truth table of BCD-to-Gray code converter.

1. Use Karnaugh maps to simplify the output function.
2. Provide the following architectural styles:
  - Dateflow style
  - Behavioral style
  - Structural style using only NOR gates
3. Write a VHDL test bench to verify the operation of the logic circuit.
4. Provide a simulation waveform depicting all possible input cases.

↓ q2-df.vhd ↓

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY q2 IS PORT (
5      X3, X2, X1, X0 : IN STD_LOGIC;
6      Y3, Y2, Y1, Y0 : OUT STD_LOGIC
7  );
8  END q2;
9

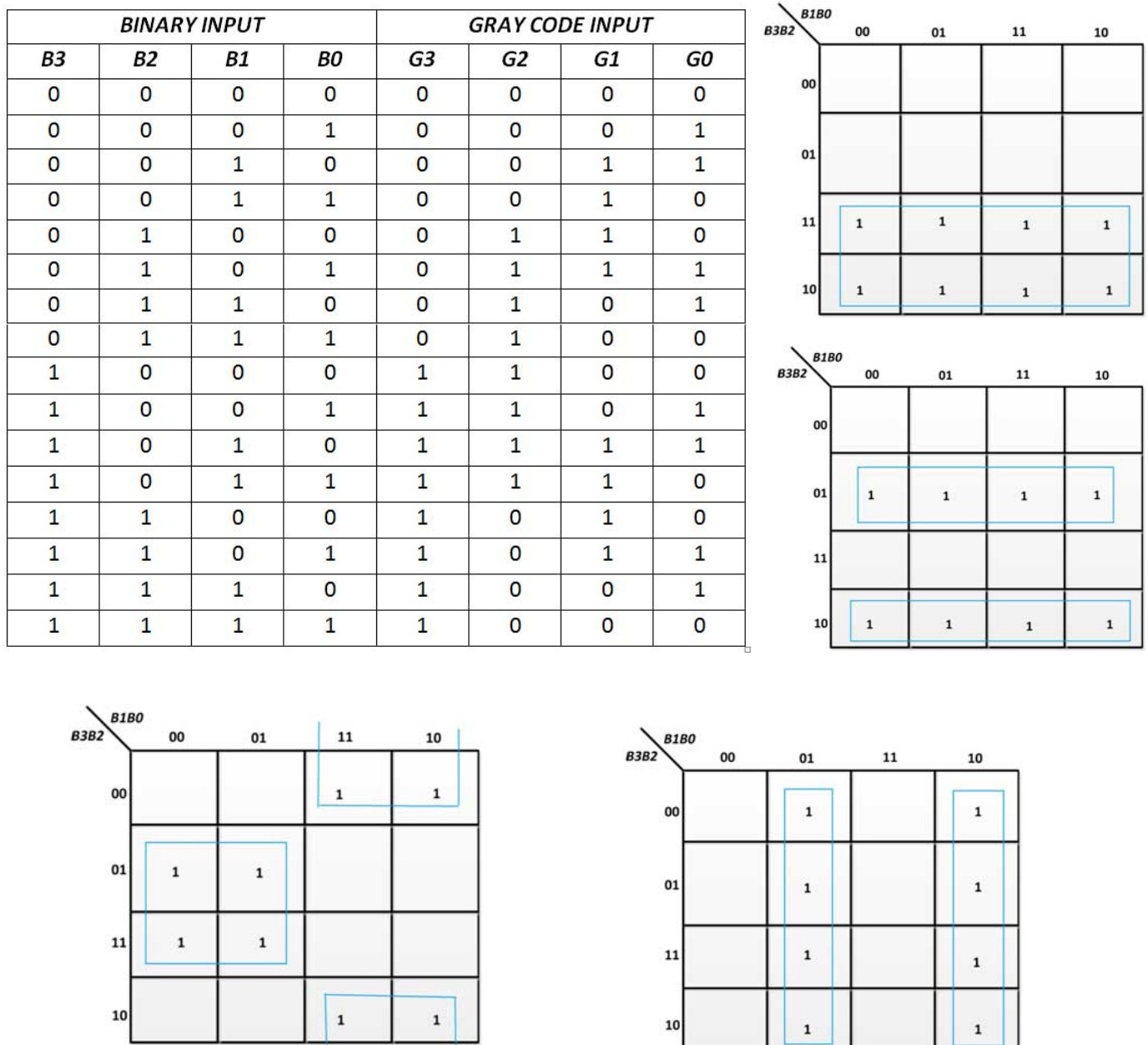
```

```

10 ARCHITECTURE dataflow OF q2 IS
11
12 BEGIN
13     Y3 <= X3;
14     Y2 <= X2 XOR X3;
15     Y1 <= X1 XOR X2;
16     Y0 <= X0 XOR X1;
17 END dataflow;

```

Code 8: Dataflow model



$$G3 = B3 \quad G2 = B3 \oplus B2 \quad G1 = B2 \oplus B1 \quad G0 = B1 \oplus B0$$

$$Y3 = X3 \quad Y2 = X3 \oplus X2 \quad Y1 = X2 \oplus X1 \quad Y0 = X1 \oplus X0$$

Figure 4: BCD to Gray Code K map and Solution for

⇓ q2-be.vhd ⇓

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY q2 IS PORT (
5      X3, X2, X1, X0 : IN STD_LOGIC;
6      Y3, Y2, Y1, Y0 : OUT STD_LOGIC
7  );
8  END q2;
9
10 ARCHITECTURE behavioral OF q2 IS
11
12 BEGIN
13     example : PROCESS (X3, X2, X1, X0)
14
15     BEGIN
16         Y3 <= X3;
17         Y2 <= X2 XOR X3;
18         Y1 <= X1 XOR X2;
19         Y0 <= X0 XOR X1;
20     END PROCESS example;
21
22 END behavioral;

```

Code 9: Behavioral model

↓ xor-nor.vhd ↓

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY xor_nor IS PORT (
5      a, b : IN STD_LOGIC;
6      o : OUT STD_LOGIC
7  );
8  END xor_nor;
9
10 ARCHITECTURE behavioral OF xor_nor IS
11     SIGNAL nota, notb, xnorab : STD_LOGIC;
12
13 BEGIN
14     xor_nor : PROCESS (a, b, nota, notb, xnorab)
15
16     BEGIN
17         nota <= a NOR a;
18         notb <= b NOR b;
19         xnorab <= (a NOR notb) NOR (nota NOR b);
20         o <= xnorab NOR xnorab;
21
22     END PROCESS xor_nor;
23
24 END behavioral;

```

Code 10: XOR gate

↓ q2-st.vhd ↓

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY q2 IS PORT (
5      X3, X2, X1, X0 : IN STD_LOGIC;
6      Y3, Y2, Y1, Y0 : OUT STD_LOGIC
7  );
8  END q2;
9
10 ARCHITECTURE structural OF q2 IS
11
12     COMPONENT xor_nor IS PORT (
13         a, b : IN STD_LOGIC;
14         o : OUT STD_LOGIC
15     );
16     END COMPONENT;
17
18 BEGIN
19     C0 : xor_nor PORT MAP(a => X3, b => '0', o
20     => Y3);
21     C1 : xor_nor PORT MAP(a => X3, b => X2, o
22     => Y2);
23     C2 : xor_nor PORT MAP(a => X2, b => X1, o
24     => Y1);
25     C3 : xor_nor PORT MAP(a => X1, b => X0, o
26     => Y0);
27
28 END structural;

```

Code 11: Structural model

↓ q2-tb.vhd ↓

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.NUMERIC_STD.ALL;
4
5  ENTITY q2_tb IS
6  END q2_tb;
7
8  ARCHITECTURE behavioral OF q2_tb IS
9
10     -- component declaration for the Unit
11     -- Under Test (UUT)
12     COMPONENT q2
13     PORT (
14         X3 : IN STD_LOGIC;
15         X2 : IN STD_LOGIC;
16         X1 : IN STD_LOGIC;
17         X0 : IN STD_LOGIC;

```

```

16         Y3 : OUT STD_LOGIC;
17         Y2 : OUT STD_LOGIC;
18         Y1 : OUT STD_LOGIC;
19         Y0 : OUT STD_LOGIC
20     );
21 END COMPONENT;
22
23 SIGNAL input_vector : STD_LOGIC_VECTOR(3
24     DOWNTO 0) := "0000";
25 SIGNAL output_vector : STD_LOGIC_VECTOR(3
26     DOWNTO 0) := "0000";
27 BEGIN
28     --INSTANTIATE the unit under test
29     uut : q2 PORT MAP(
30         X3 => input_vector(3),
31         X2 => input_vector(2),
32         X1 => input_vector(1),
33         X0 => input_vector(0),
34
35         Y3 => output_vector(3),
36         Y2 => output_vector(2),
37         Y1 => output_vector(1),
38         Y0 => output_vector(0)
39     );
40
41 --stimulus process
42 stim_proc : PROCESS
43 BEGIN
44     FOR index IN 0 TO 15 LOOP
45         input_vector <= STD_LOGIC_VECTOR(
46             to_unsigned(index, 4));
47         WAIT FOR 50 ns;
48     END LOOP;
49 END PROCESS;
50 END behavioral;

```

Code 12: Testbench for all cases

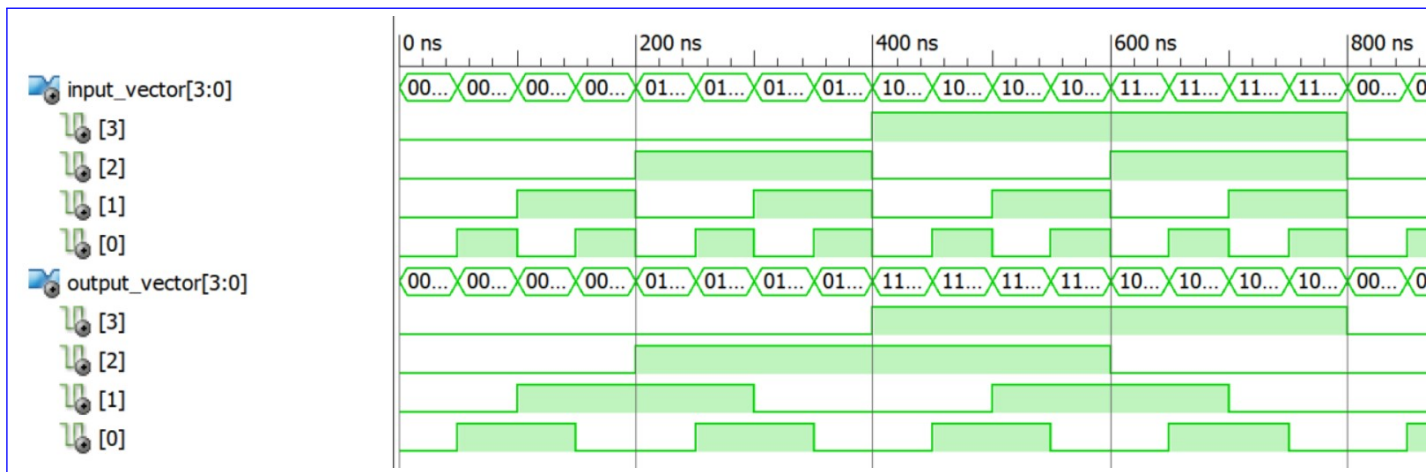


Figure 5: Simulation Waveform

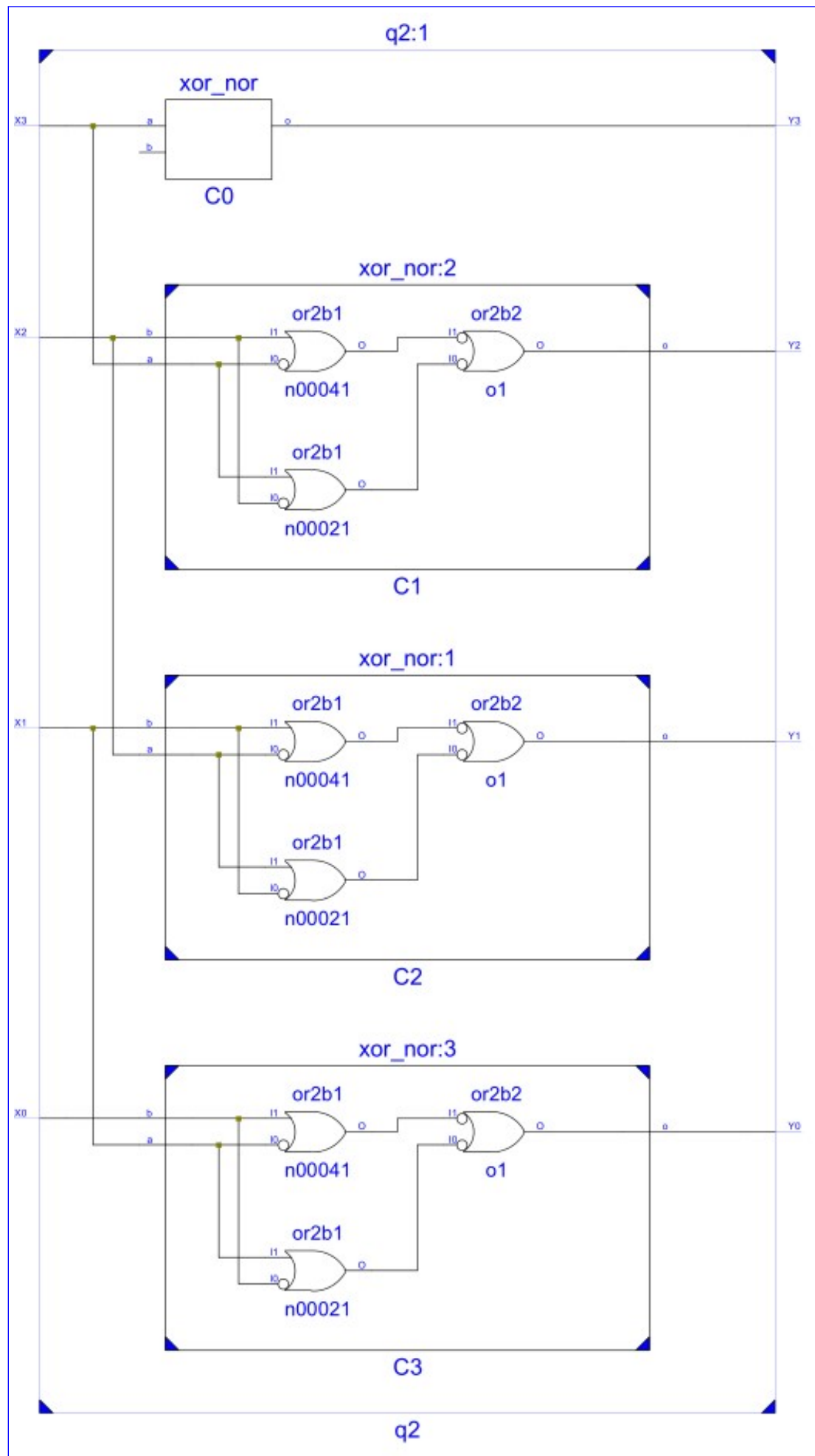


Figure 6: RTL Schematic

### 5.3 Question -3

Write VHDL code to implement the logic function (F) with the three input variables x1, x2, and x3. The function (F) is equal to 1 if and only if two variables are equal to 1; otherwise, it is equal to zero.

1. Draw a truth table for the function (F), and use Karnaugh maps to simplify
2. Provide the following architectural styles:
  - Dataflow style
  - Behavioral style
  - Structural style using only NOR gates
3. Write a VHDL test bench to verify the operation of the logic circuit.
4. Provide a simulation waveform depicting all possible input cases.

Input			Output
X3	X2	X1	F
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

Table 1: Truth table Problem 3

X1 \ X2	X1			
	00	01	11	11
0	0	0	1	0
1	0	1	0	1

Table 2: K-map solution

Expression Obtained is:

$$F = x_1x_2'x_3 + x_1'x_2x_3 + x_1x_2x_3'$$

$$= ((x_1' + x_2 + x_3')(x_1 + x_2' + x_3')(x_1' + x_2' + x_3'))$$

⇓ q3-df.vhd ⇓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY q3 IS PORT (
5     X1, X2, X3 : IN STD_LOGIC;
6     F : OUT STD_LOGIC
7 );
8
9
10 ARCHITECTURE dataflow OF q3 IS
11 BEGIN
12     F <= (X1 AND X2 AND NOT X3) OR (X3 AND (X1
13         XOR X2));
14 END dataflow;
```

Code 13: Dataflow model

⇓ q3-be.vhd ⇓



```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY q3 IS PORT (
5     X1, X2, X3 : IN STD_LOGIC;
6     F : OUT STD_LOGIC
7 );
8 END q3;
9
10 ARCHITECTURE behavioral OF q3 IS
11     SIGNAL A1, A2, A3 : STD_LOGIC;
12
13 BEGIN
14
15     PROCESS (X1, X2, X3, A1, A2, A3)
16     BEGIN
17         A1 <= X1 AND X2 AND NOT X3;
18         A2 <= X1 XOR X2;
19         A3 <= A2 AND X3;
20         F <= A1 OR A3;
21     END PROCESS;
22 END behavioral;

```

Code 14: Behavioral model

---



---

⇓ q3-not.vhd ⇓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY q3_not IS
5     PORT (
6         a : IN STD_LOGIC;
7         o : OUT STD_LOGIC);
8 END q3_not;
9
10 ARCHITECTURE behavioral OF q3_not IS
11
12 BEGIN
13     q3_not : PROCESS (a)
14
15     BEGIN
16         o <= a NOR a;
17     END PROCESS q3_not;
18
19 END behavioral;

```

Code 15: NOT gate using only NOR

---



---

⇓ q3-or.vhd ⇓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY q3_or IS PORT (
5     a, b, c : IN STD_LOGIC;
6     o : OUT STD_LOGIC
7 );
8 END q3_or;
9
10 ARCHITECTURE behavioral OF q3_or IS
11     SIGNAL G1, G2, G3 : STD_LOGIC;
12
13 BEGIN
14     q3_or : PROCESS (a, b, c, G1, G2, G3)
15
16     BEGIN
17         G1 <= a NOR b;
18         G2 <= G1 NOR G1;
19         G3 <= G2 NOR c;
20         o <= G3 NOR G3;
21     END PROCESS q3_or;
22
23 END behavioral;

```

Code 16: 3-input OR gate using only NOR

---



---

⇓ q3-and.vhd ⇓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY q3_and IS PORT (
5     a, b, c : IN STD_LOGIC;
6     o : OUT STD_LOGIC
7 );
8 END q3_and;
9
10 ARCHITECTURE behavioral OF q3_and IS
11     SIGNAL F1, F2, F3, F4, F5 : STD_LOGIC;
12 BEGIN
13     q3_and : PROCESS (a, b, c, F1, F2, F3, F4, F5)
14     BEGIN
15         F1 <= a NOR a;
16         F2 <= b NOR b;
17         F3 <= c NOR c;
18         F4 <= F1 NOR F2;
19         F5 <= F4 NOR F4;
20         o <= F5 NOR F3;
21     END PROCESS q3_and;
22
23 END behavioral;

```

## Code 17: 3-input AND gate using only NOR

↓ q3-st.vhd ↓

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY q3 IS PORT (
5      X1, X2, X3 : IN STD_LOGIC;
6      F : OUT STD_LOGIC
7  );
8  END q3;
9
10 ARCHITECTURE structural OF q3 IS
11     SIGNAL A1, A2, A3, A4, A5, A6 : STD_LOGIC;
12
13     COMPONENT q3_and IS PORT (
14         a, b, c : IN STD_LOGIC;
15         o : OUT STD_LOGIC
16     );
17     END COMPONENT;
18
19     COMPONENT q3_or IS PORT (
20         a, b, c : IN STD_LOGIC;
21         o : OUT STD_LOGIC
22     );
23
24     END COMPONENT;
25
26     COMPONENT q3_not IS PORT (
27         a : IN STD_LOGIC;
28         o : OUT STD_LOGIC
29     );
30     END COMPONENT;
31 BEGIN
32     N1 : q3_not PORT MAP(a => X1, o => A1);
33     N2 : q3_not PORT MAP(a => X2, o => A2);
34     N3 : q3_not PORT MAP(a => X3, o => A3);
35     N4 : q3_and PORT MAP(a => A1, b => X2, c =>
36         X3, o => A4);
37     N5 : q3_and PORT MAP(a => A2, b => X1, c =>
38         X3, o => A5);
39     N6 : q3_and PORT MAP(a => A3, b => X1, c =>
40         X2, o => A6);
41     N7 : q3_or PORT MAP(a => A4, b => A5, c =>
42         A6, o => F);
43 END structural;

```

## Code 18: Structural model

↓ q3-tb.vhd ↓

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.NUMERIC_STD.ALL;
4
5  ENTITY q3_tb IS
6  END q3_tb;
7
8  ARCHITECTURE behavioral OF q3_tb IS
9      COMPONENT q3
10         PORT (
11             X1, X2, X3 : IN STD_LOGIC;
12             F : OUT STD_LOGIC
13         );
14     END COMPONENT;
15
16     SIGNAL input : STD_LOGIC_VECTOR(2 DOWNTO 0)
17         := "000";
18     SIGNAL output : STD_LOGIC := '0';
19 BEGIN
20     uut : q3 PORT MAP(
21         X3 => input(2),
22         X2 => input(1),
23         X1 => input(0),
24         F => output
25     );
26
27     stim_proc : PROCESS
28
29     BEGIN
30         FOR index IN 0 TO 7 LOOP
31             input <= STD_LOGIC_VECTOR(TO_UNSIGNED
32                 (index, 3));
33             WAIT FOR 50 ns;
34         END LOOP;
35     END PROCESS;
36 END behavioral;

```

## Code 19: Testbench for all cases

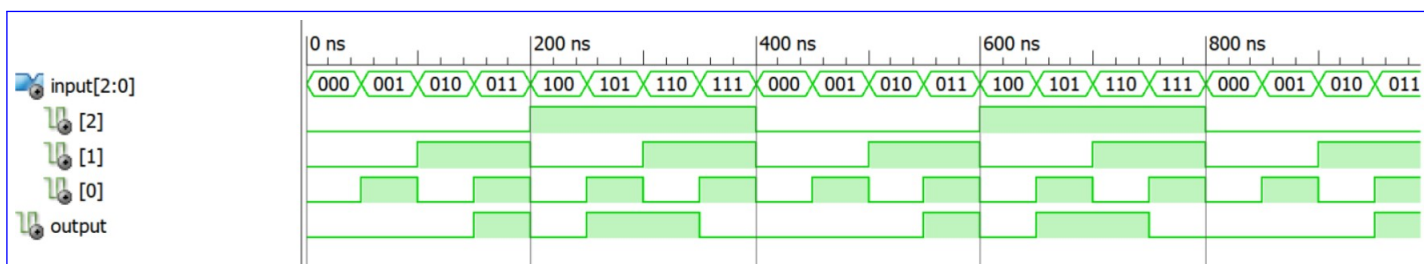


Figure 7: Simulation Waveform for Problem 3

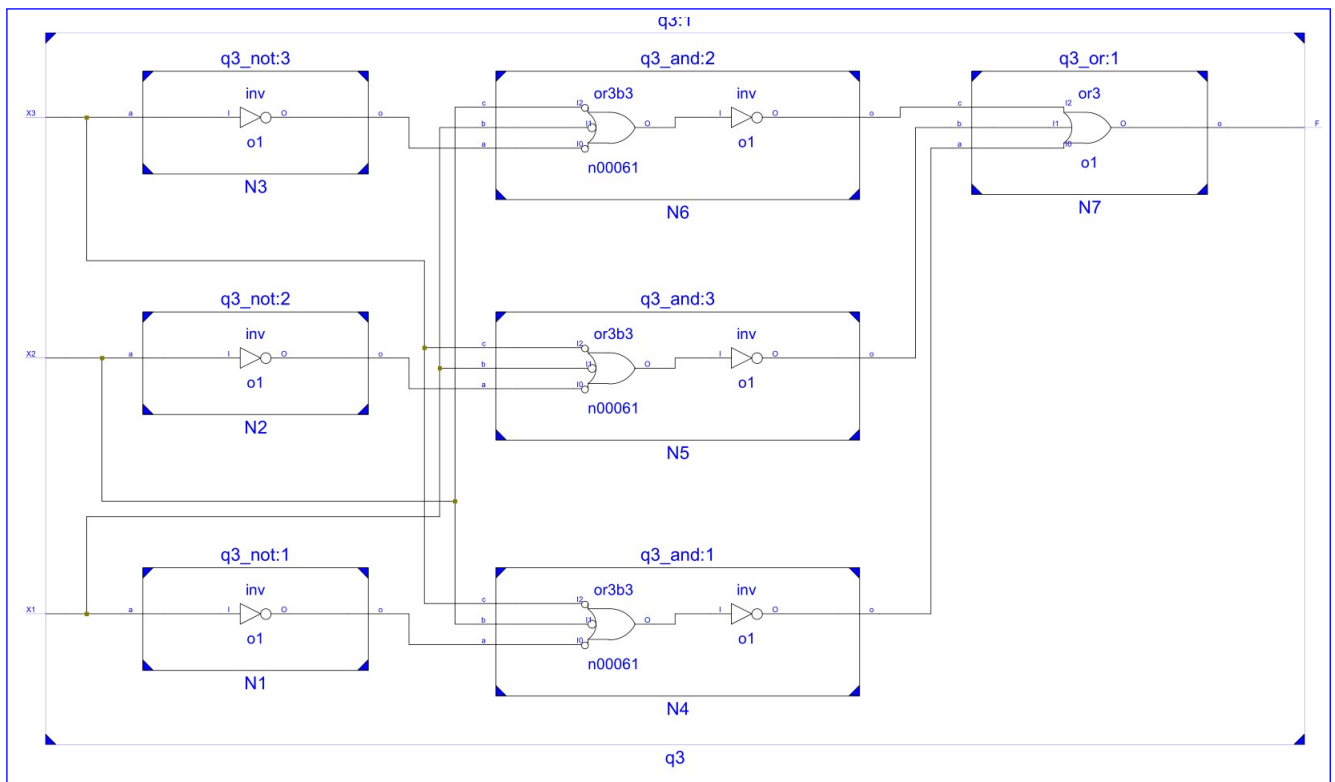


Figure 8: RTL Schematic for Problem 3

#### 5.4 Question -4

Write VHDL code to implement the implicit sum of products (SOP) and product of sums (POS) logic functions.

$$F(x_1, x_2, x_3, x_4) = \sum(m_0, m_1, m_4, m_5, m_8, m_9, m_{14}, m_{15})$$

$$F(x_1, x_2, x_3, x_4) = \Pi(M_0, M_1, M_5, M_8, M_9, M_{13}, M_{15})$$

1. Draw a truth table for the function (F), and use Karnaugh maps to simplify
2. Provide the following architectural styles:
  - Dataflow style
  - Behavioral style
  - Structural style using only NOR gates
3. Write a VHDL test bench to verify the operation of the logic circuit.
4. Provide a simulation waveform depicting all possible input cases.

Input				Output	
X1	X2	X3	X4	F1	F2
0	0	0	0	1	0
0	0	0	1	1	0
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	1	1
0	1	0	1	1	0
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	0	1
1	0	1	1	0	1
1	1	0	0	0	1
1	1	0	1	0	0
1	1	1	0	1	1
1	1	1	1	1	0

Figure 9: Truth table Problem 4

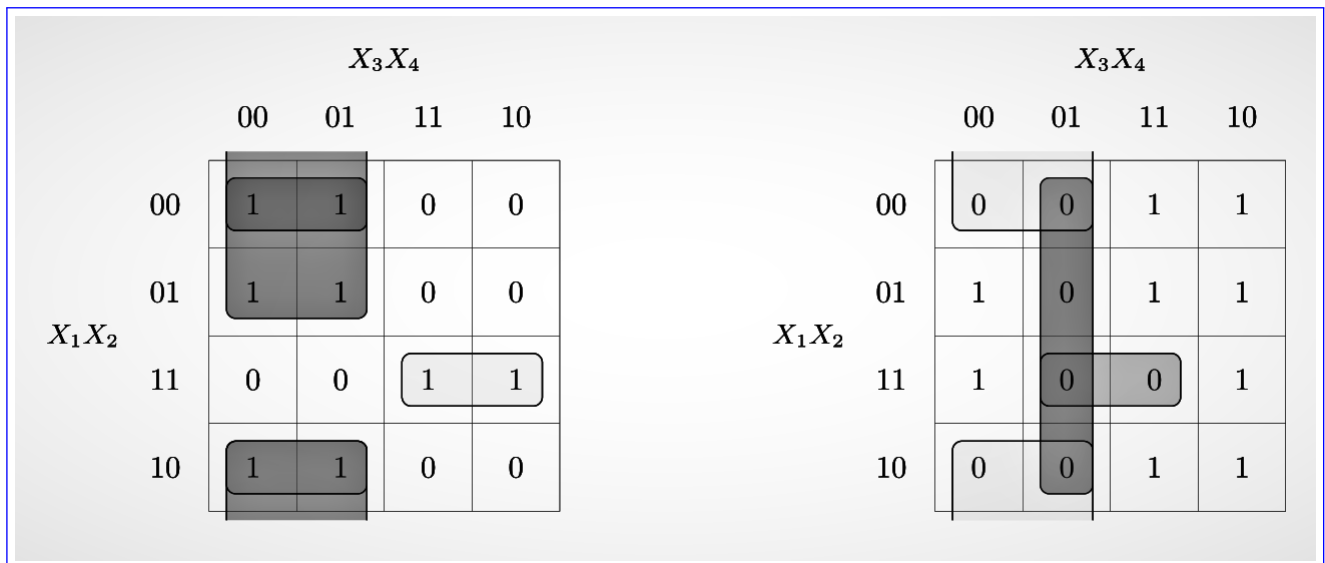


Figure 10: SOP and POS K-map reduction F1 and F2

$$F_1 = X'_1X'_3 + X_1X_2X_3 + X'_2X'_3$$

$$F_2 = (X_3 + X_4)(X'_1 + X'_2 + X'_4)(X_2 + X_3)$$

↓ q4sop-df.vhd ↓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY q4_sop IS PORT (
5     X1, X2, X3, X4 : IN STD_LOGIC;
6     Y : OUT STD_LOGIC
7 );
8 END q4_sop;

```

```

9
10 ARCHITECTURE dataflow OF q4_sop IS
11
12 BEGIN
13     Y <= (NOT X1 AND NOT X3) OR (NOT X2 AND NOT
14           X3) OR (X1 AND X2 AND X3);
15 END dataflow;

```

Code 20: Dataflow model SOP

⇓ q4sop-be.vhd ⇓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY q4_sop IS PORT (
5     X1, X2, X3, X4 : IN STD_LOGIC;
6     Y : OUT STD_LOGIC
7 );
8 END q4_sop;
9
10 ARCHITECTURE behavioral OF q4_sop IS
11     SIGNAL Y1, Y2, Y3 : STD_LOGIC;
12
13 BEGIN

```

```

14
15     PROCESS (X1, X2, X3, X4, Y1, Y2, Y3)
16
17     BEGIN
18         Y1 <= NOT X1 AND NOT X3;
19         Y2 <= NOT X2 AND NOT X3;
20         Y3 <= X1 AND X2 AND X3;
21         Y <= Y1 OR Y2 OR Y3;
22
23     END PROCESS;
24
25 END behavioral;

```

Code 21: Behavioral model SOP

⇓ q4-not.vhd ⇓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY q4_not IS
5     PORT (
6         a : IN STD_LOGIC;
7         o : OUT STD_LOGIC);
8 END q4_not;
9
10 ARCHITECTURE behavioral OF q4_not IS

```

```

11
12 BEGIN
13     q4_not : PROCESS (a)
14
15     BEGIN
16         o <= a NOR a;
17     END PROCESS q4_not;
18
19 END behavioral;

```

Code 22: NOT gate implementation using only NOR

⇓ q4-or.vhd ⇓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY q4_or IS PORT (
5     a, b, c : IN STD_LOGIC;
6     o : OUT STD_LOGIC
7 );
8 END q4_or;
9
10 ARCHITECTURE behavioral OF q4_or IS
11     SIGNAL G1, G2, G3 : STD_LOGIC;
12

```

```

13 BEGIN
14     q4_or : PROCESS (a, b, c, G1, G2, G3)
15
16     BEGIN
17         G1 <= a NOR b;
18         G2 <= G1 NOR G1;
19         G3 <= G2 NOR c;
20         o <= G3 NOR G3;
21     END PROCESS q4_or;
22
23 END behavioral;

```

Code 23: 3-input OR gate implementation using only NOR

⇓ q4-and.vhd ⇓

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY q4_and IS PORT (
5      a, b, c : IN STD_LOGIC;
6      o : OUT STD_LOGIC
7  );
8  END q4_and;
9
10 ARCHITECTURE behavioral OF q4_and IS
11     SIGNAL F1, F2, F3, F4, F5 : STD_LOGIC;
12 BEGIN
13     q4_and : PROCESS (a, b, c, F1, F2, F3, F4, F5)
14     BEGIN
15         F1 <= a NOR a;
16         F2 <= b NOR b;
17         F3 <= c NOR c;
18         F4 <= F1 NOR F2;
19         F5 <= F4 NOR F4;
20         o <= F5 NOR F3;
21     END PROCESS q4_and;
22 END behavioral;

```

Code 24: 3-input AND gate implementation using only NOR

⇓ q4sop-st.vhd ⇓

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY q4_sop IS PORT (
5      X1, X2, X3, X4 : IN STD_LOGIC;
6      Y : OUT STD_LOGIC
7  );
8  END q4_sop;
9
10 ARCHITECTURE structural OF q4_sop IS
11     SIGNAL A1, A2, A3, A4, A5, A6 : STD_LOGIC;
12
13     COMPONENT q4_and IS PORT (
14         a, b, c : IN STD_LOGIC;
15         o : OUT STD_LOGIC
16     );
17     END COMPONENT;
18
19     COMPONENT q4_or IS PORT (
20         a, b, c : IN STD_LOGIC;
21         o : OUT STD_LOGIC
22     );
23     END COMPONENT;
24
25     COMPONENT q4_not IS PORT (
26         a : IN STD_LOGIC;
27         o : OUT STD_LOGIC
28     );
29     END COMPONENT;
30
31 BEGIN
32     N1 : q4_not PORT MAP(a => X1, o => A1);
33     N2 : q4_not PORT MAP(a => X2, o => A2);
34     N3 : q4_not PORT MAP(a => X3, o => A3);
35     N4 : q4_and PORT MAP(a => A1, b => A3, c =>
36         '1', o => A4);
37     N5 : q4_and PORT MAP(a => A2, b => A3, c =>
38         '1', o => A5);
39     N6 : q4_and PORT MAP(a => X1, b => X2, c =>
40         X3, o => A6);
41     N7 : q4_or PORT MAP(a => A4, b => A5, c =>
42         A6, o => Y);
43 END structural;

```

Code 25: Structural model SOP

⇓ q4sop-tb.vhd ⇓

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.NUMERIC_STD.ALL;
4
5  ENTITY q4_sop_tb IS
6  END q4_sop_tb;
7
8  ARCHITECTURE behavioral OF q4_sop_tb IS
9      -- component declaaration for the Unit
10     Under Test (UUT)
11     COMPONENT q4_sop
12     PORT (
13         X1 : IN STD_LOGIC;
14         X2 : IN STD_LOGIC;
15         X3 : IN STD_LOGIC;
16         X4 : IN STD_LOGIC;
17         Y : OUT STD_LOGIC
18     );
19     END COMPONENT;
20
21     SIGNAL input_vector : STD_LOGIC_VECTOR(3
22         DOWNT0 0) := "0000";
23     SIGNAL output : STD_LOGIC := '0';
24
25 BEGIN
26     --INSTANTIATE the unit under test
27     uut : q4_sop PORT MAP(
28         X1 => input_vector(3),
29         X2 => input_vector(2),
30         X3 => input_vector(1),
31         X4 => input_vector(0),
32         Y => output
33     );
34
35     --stimulus process
36     stim_proc : PROCESS
37     BEGIN
38         FOR index IN 0 TO 15 LOOP
39             input_vector <= STD_LOGIC_VECTOR(
40                 to_unsigned(index, 4));
41             WAIT FOR 50 ns;
42         END LOOP;
43     END PROCESS;
44 END behavioral;

```

Code 26: Testbench for all possible cases SOP

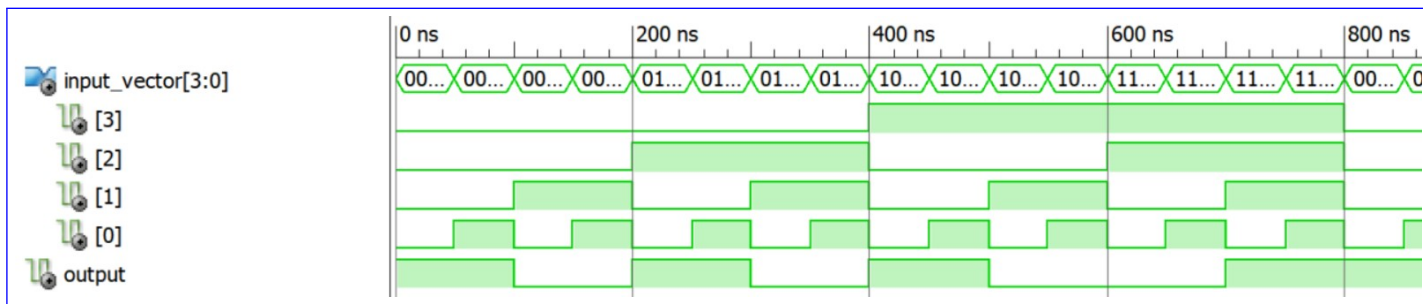


Figure 11: Simulation Waveform SOP

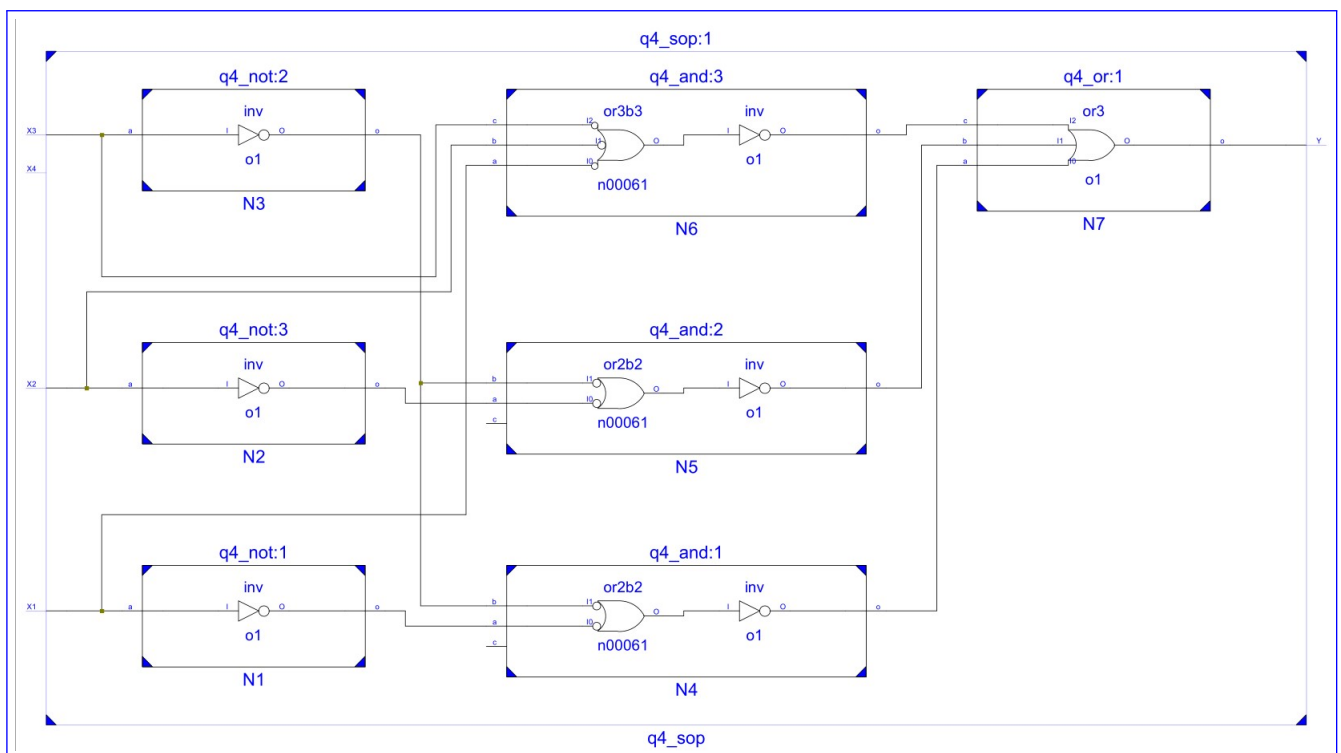


Figure 12: RTL Schematic SOP

⇓ q4pos-df.vhd ⇓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY q4_pos IS PORT (
5     X1, X2, X3, X4 : IN STD_LOGIC;
6     Y : OUT STD_LOGIC
7 );
8 END q4_pos;

```

```

9
10 ARCHITECTURE dataflow OF q4_pos IS
11
12 BEGIN
13     Y <= (X3 OR NOT X4) AND (X2 OR X3) AND (NOT
14         X1 OR NOT X2 OR NOT X4);
15 END dataflow;

```

Code 27: Dataflow model POS

⇓ q4pos-be.vhd ⇓

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY q4_pos IS PORT (
5      X1, X2, X3, X4 : IN STD_LOGIC;
6      Y : OUT STD_LOGIC
7  );
8  END q4_pos;
9
10 ARCHITECTURE behavioral OF q4_pos IS
11     SIGNAL Y1, Y2, Y3 : STD_LOGIC;
12
13 BEGIN
14
15     PROCESS (X1, X2, X3, X4, Y1, Y2, Y3)
16
17     BEGIN
18         Y1 <= X3 OR NOT X4;
19         Y2 <= X2 OR X3;
20         Y3 <= NOT X1 OR NOT X2 OR NOT X4;
21         Y <= Y1 AND Y2 AND Y3;
22
23     END PROCESS;
24
25 END behavioral;

```

Code 28: Behavioral model POS

⇓ q4pos-st.vhd ⇓

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY q4_pos IS PORT (
5      X1, X2, X3, X4 : IN STD_LOGIC;
6      Y : OUT STD_LOGIC
7  );
8  END q4_pos;
9
10 ARCHITECTURE structural OF q4_pos IS
11     SIGNAL A1, A2, A3, A4, A5, A6 : STD_LOGIC;
12
13     COMPONENT q4_and IS PORT (
14         a, b, c : IN STD_LOGIC;
15         o : OUT STD_LOGIC
16     );
17     END COMPONENT;
18
19     COMPONENT q4_or IS PORT (
20         a, b, c : IN STD_LOGIC;
21         o : OUT STD_LOGIC
22     );
23     END COMPONENT;
24
25     COMPONENT q4_not IS PORT (
26         a : IN STD_LOGIC;
27         o : OUT STD_LOGIC
28     );
29     END COMPONENT;
30
31 BEGIN
32     N1 : q4_not PORT MAP(a => X1, o => A1);
33     N2 : q4_not PORT MAP(a => X2, o => A2);
34     N3 : q4_not PORT MAP(a => X4, o => A3);
35     N4 : q4_or PORT MAP(a => A1, b => A2, c =>
36         A3, o => A4);
37     N5 : q4_or PORT MAP(a => X3, b => A3, c =>
38         '0', o => A5);
39     N6 : q4_or PORT MAP(a => X2, b => X3, c =>
40         '0', o => A6);
41     N7 : q4_and PORT MAP(a => A4, b => A5, c =>
42         A6, o => Y);
43 END structural;

```

Code 29: Structural model POS

⇓ q4pos-tb.vhd ⇓

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.NUMERIC_STD.ALL;
4
5  ENTITY q4_pos_tb IS
6  END q4_pos_tb;
7
8  ARCHITECTURE behavioral OF q4_pos_tb IS
9      -- component declaaraation for the Unit
10      Under Test (UUT)
11      COMPONENT q4_pos
12          PORT (
13              X1 : IN STD_LOGIC;
14              X2 : IN STD_LOGIC;
15              X3 : IN STD_LOGIC;
16              X4 : IN STD_LOGIC;
17              Y : OUT STD_LOGIC
18          );
19      END COMPONENT;
20
21      SIGNAL input_vector : STD_LOGIC_VECTOR(3
22          DOWNT0 0) := "0000";
23      SIGNAL output : STD_LOGIC := '0';
24
25      --INSTANTIATE the unit under test
26      uut : q4_pos PORT MAP(
27          X1 => input_vector(3),
28          X2 => input_vector(2),
29          X3 => input_vector(1),
30          X4 => input_vector(0),
31          Y => output
32      );
33
34      --stimulus process
35      stim_proc : PROCESS
36
37      BEGIN
38          FOR index IN 0 TO 15 LOOP
39              input_vector <= STD_LOGIC_VECTOR(
40                  to_unsigned(index, 4));
41              WAIT FOR 50 ns;
42          END LOOP;
43      END PROCESS;
44
45 END behavioral;

```



Code 30: Testbench for all possible cases POS

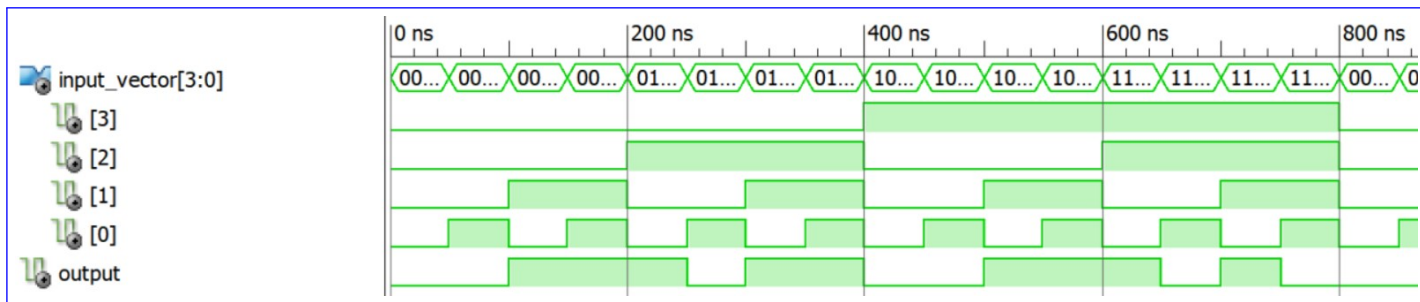


Figure 13: Simulation Waveform POS

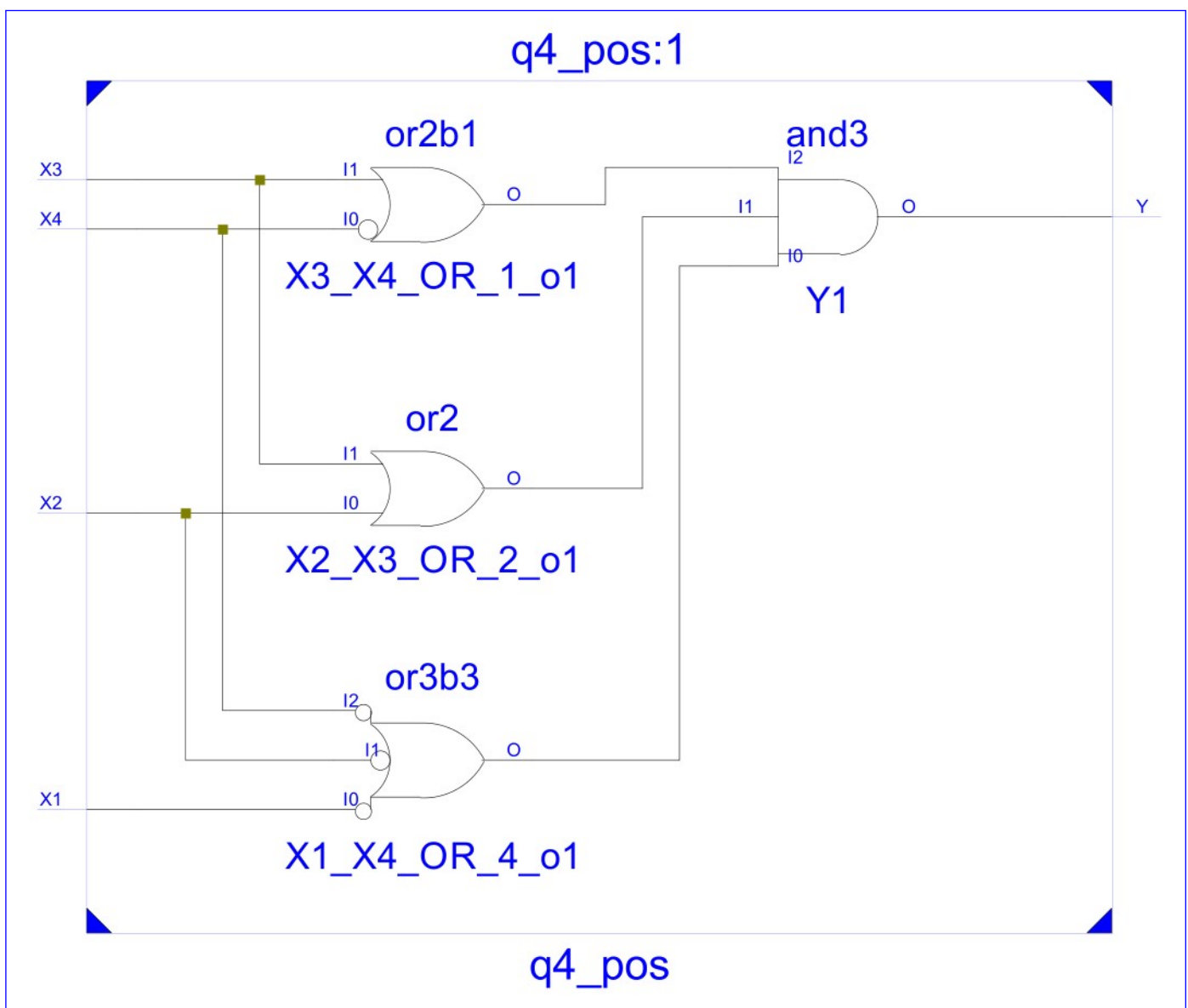


Figure 14: RTL Schematic POS

### 5.5 Question -5

Write VHDL code to implement a 2:1 MUX having inputs x1 and x2, select line s and output y.

1. Provide the following architectural implementations:
  - Using WHEN-ELSE statement
  - IF-THEN-ELSE statement
2. Write a VHDL test bench to verify the operation of the 2:1 MUX.
3. Provide a simulation waveform depicting all possible input cases.

Select	Input		Output
S	X2	X1	Y
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

Figure 15: Truth Table 2:1 multiplexer

The required Boolean Equation is ,  $Y = SX_2 + S'X_1$

⇓ q5-when-else.vhd ⇓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY q5_mux2_1 IS PORT (
5     S, X1, X2 : IN STD_LOGIC;
6     Y : OUT STD_LOGIC
7 );
8 END q5_mux2_1;
9
10 ARCHITECTURE dataflow OF q5_mux2_1 IS
11 BEGIN
12     Y <= '1' WHEN ((NOT S AND X1) OR (S AND X2))
13         = '1' ELSE
14         '0';
15 END dataflow;

```

Code 31: 2:1 MUX implementation using WHEN-ELSE statement

⇓ q5-if-then-else.vhd ⇓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY q5_mux2_1 IS PORT (
5     S, X1, X2 : IN STD_LOGIC;
6     Y : OUT STD_LOGIC
7 );
8 END q5_mux2_1;
9
10 ARCHITECTURE behavioral OF q5_mux2_1 IS
11 BEGIN
12     PROCESS (S, X1, X2)
13     BEGIN
14         IF ((NOT S AND X1) OR (S AND X2)) = '1'
15         THEN
16             Y <= '1';
17         ELSE
18             Y <= '0';
19         END IF;
20     END PROCESS;
21 END behavioral;

```

Code 32: 2:1 MUX implementation using IF-THEN-ELSE statement

↓ q5-tb.vhd ↓

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.NUMERIC_STD.ALL;
4
5  ENTITY q5_mux2_1_tb IS
6  END q5_mux2_1_tb;
7
8  ARCHITECTURE behavioral OF q5_mux2_1_tb IS
9      COMPONENT q5_mux2_1
10         PORT (
11             S : IN STD_LOGIC;
12             X1 : IN STD_LOGIC;
13             X2 : IN STD_LOGIC;
14             Y : OUT STD_LOGIC
15         );
16     END COMPONENT;
17
18     SIGNAL input : STD_LOGIC_VECTOR(2 DOWNTO 0)
19         := "000";
20     SIGNAL output : STD_LOGIC;
21
22 BEGIN
23     uut : q5_mux2_1 PORT MAP (
24         S => input(2),
25         X2 => input(1),
26         X1 => input(0),
27         Y => output
28     );
29
30     stim_proc : PROCESS
31     BEGIN
32         FOR index IN 0 TO 7 LOOP
33             input <= STD_LOGIC_VECTOR(TO_UNSIGNED
34                 (index, 3));
35             WAIT FOR 50 ns;
36         END LOOP;
37     END PROCESS;
38 END behavioral;

```

Code 33: Testbench for all possible cases

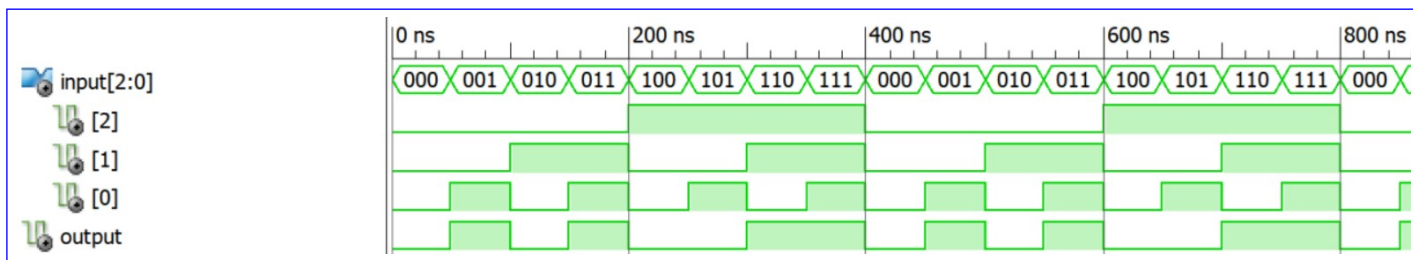


Figure 16: Simulation Waveform

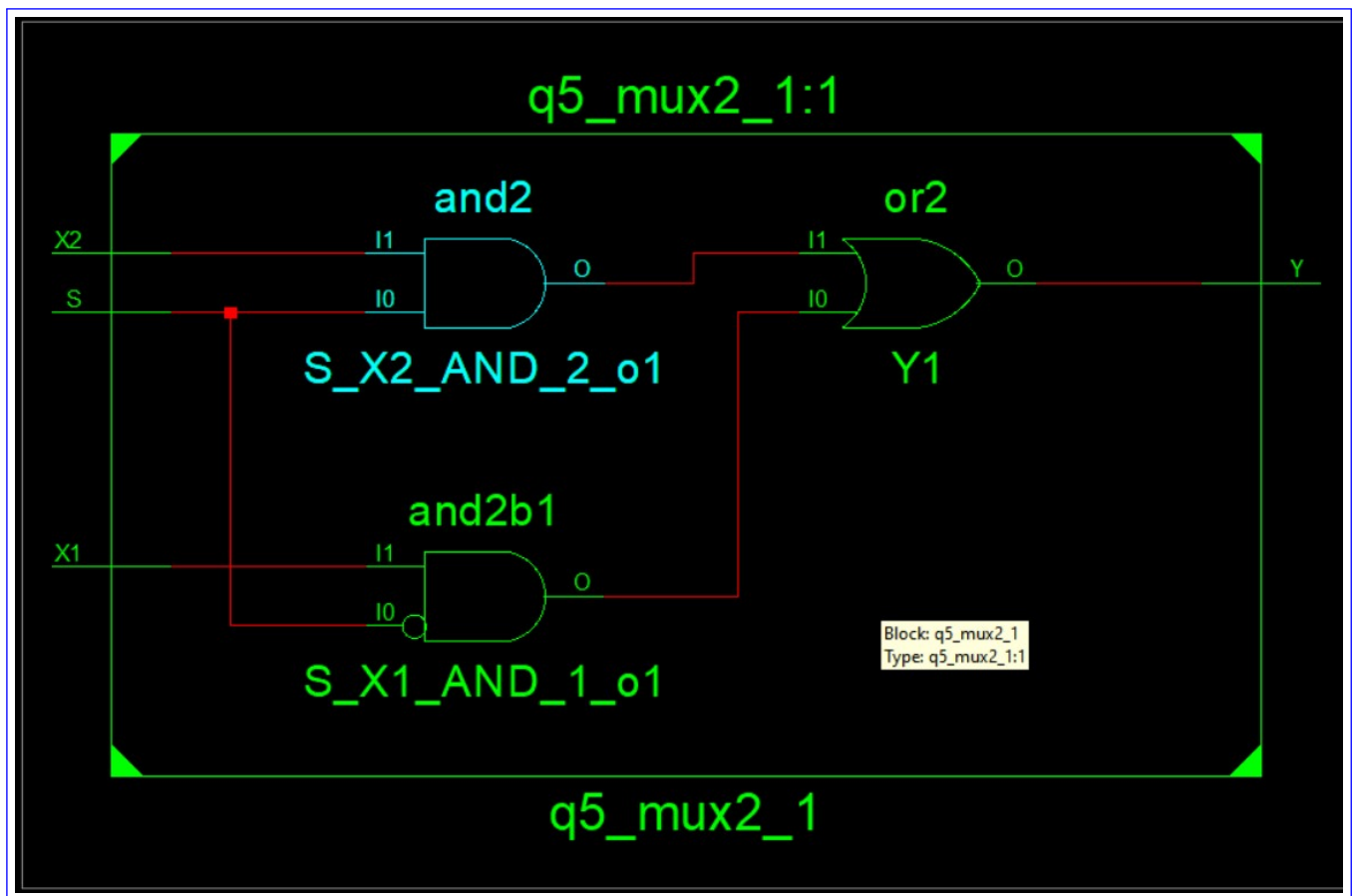


Figure 17: RTL Schematic

### 5.6 Question -6

Write VHDL code to implement a 4:1 MUX having inputs x1, x2, x3 and x4, select lines s1, s0 and output y using three 2:1 multiplexers as the basic building blocks.

1. Use a hierarchical design approach:
  - (a) Create component definitions in separate (.vhd) files. Use either Dataflow or Behavioral or Structural design styles.
  - (b) Use Structural design style for the 4:1 MUX architecture:
    - i. Make use of 2:1 MUX component declaration.
    - ii. Make use of 2:1 MUX component instantiation.
2. Write a VHDL test bench to verify the operation of the 2:1 MUX.
3. Provide a simulation waveform depicting all possible input cases.

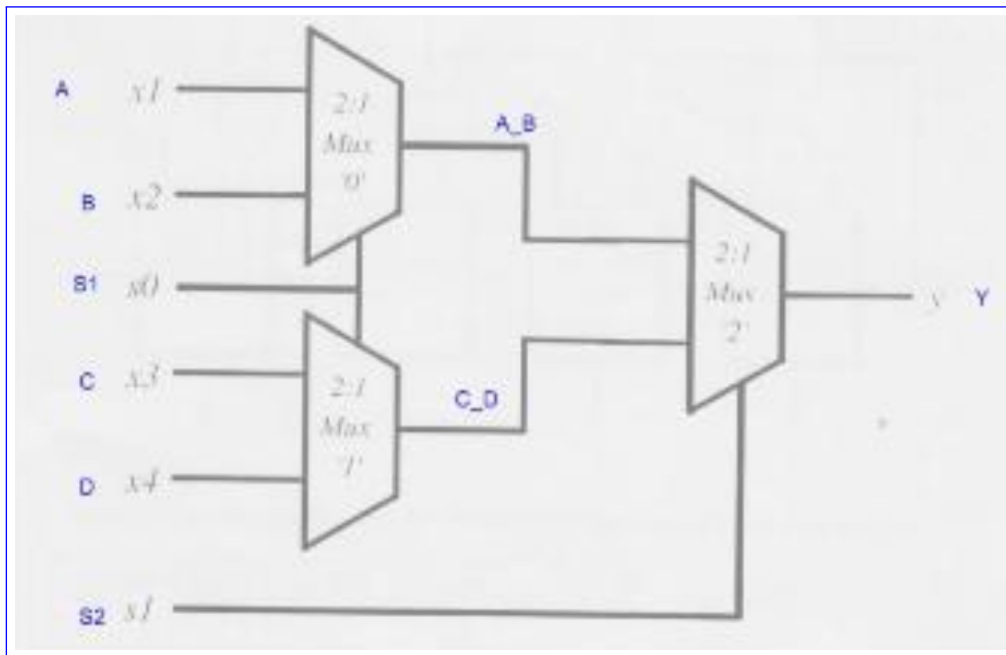


Figure 18: MUX using three 2:1 MUX as basic building blocks

↓ q5-when-else.vhd ↓

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY q5_mux2_1 IS PORT (
5      S, X1, X2 : IN STD_LOGIC;
6      Y : OUT STD_LOGIC
7  );
8  END q5_mux2_1;
9
10 ARCHITECTURE dataflow OF q5_mux2_1 IS
11 BEGIN
12     Y <= '1' WHEN ((NOT S AND X1) OR (S AND X2))
13         = '1' ELSE
14         '0';
15 END dataflow;

```

Code 34: 2:1 MUX implementation using WHEN-ELSE statement

↓ q6-st.vhd ↓

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY q6_mux4_1 IS PORT (
5      X1, X2, X3, X4, S0, S1 : IN STD_LOGIC;
6      Y : OUT STD_LOGIC
7  );
8  END q6_mux4_1;
9
10 ARCHITECTURE structural OF q6_mux4_1 IS
11     SIGNAL F1, F2 : STD_LOGIC;
12
13     COMPONENT q5_mux2_1 IS PORT (
14         X1, X2, S : IN STD_LOGIC;
15         Y : OUT STD_LOGIC
16     );
17     END COMPONENT;
18
19 BEGIN
20     M0 : q5_mux2_1 PORT MAP(X1 => X1, X2 => X2,
21                             S => S0, Y => F1);
22     M1 : q5_mux2_1 PORT MAP(X1 => X3, X2 => X4,
23                             S => S0, Y => F2);
24     M2 : q5_mux2_1 PORT MAP(X1 => F1, X2 => F2,
25                             S => S1, Y => Y);
26 END structural;

```

Code 35: 4:1 MUX implementation using three 2:1 MUX: Structural model

↓ q6-tb.vhd ↓

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.NUMERIC_STD.ALL;
4
5  ENTITY q6_mux4_1_tb IS
6  END q6_mux4_1_tb;
7
8  ARCHITECTURE behavioral OF q6_mux4_1_tb IS
9      -- component declaration for the Unit
10     Under Test (UUT)
11     COMPONENT q6_mux4_1
12     PORT (
13         X1, X2, X3, X4, S0, S1 : IN STD_LOGIC
14     );
15     Y : OUT STD_LOGIC
16     );
17 END COMPONENT;
18
19 SIGNAL select_vector : STD_LOGIC_VECTOR(1
20 DOWNT0 0) := "00";
21 SIGNAL input_vector : STD_LOGIC_VECTOR(3
22 DOWNT0 0) := "0000";
23 SIGNAL output : STD_LOGIC := '0';
24
25 BEGIN
26     --INSTANTIATE the unit under test
27
28     uut : q6_mux4_1 PORT MAP(
29         S0 => select_vector(0),
30         S1 => select_vector(1),
31         X1 => input_vector(3),
32         X2 => input_vector(2),
33         X3 => input_vector(1),
34         X4 => input_vector(0),
35         Y => output
36     );
37
38     --stimulus process
39     stim_proc : PROCESS
40     BEGIN
41         FOR selector IN 0 TO 3 LOOP
42             select_vector <= STD_LOGIC_VECTOR(
43                 to_unsigned(selector, 2));
44             FOR index IN 0 TO 15 LOOP
45                 input_vector <= STD_LOGIC_VECTOR(
46                     to_unsigned(index, 4));
47                 WAIT FOR 40 ns;
48             END LOOP;
49         END LOOP;
50     END PROCESS;
51 END behavioral;

```

Code 36: Testbench for all possible cases

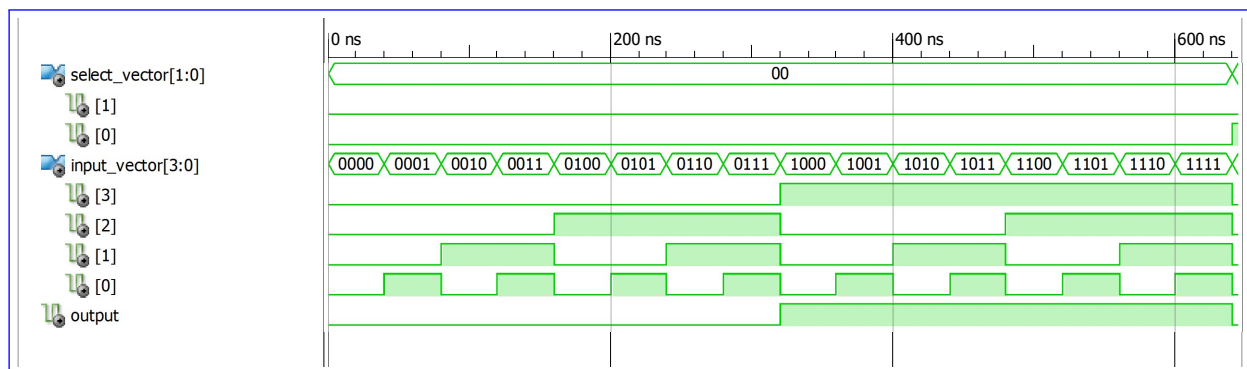


Figure 19: Simulation Waveform 0 ns to 700 ns

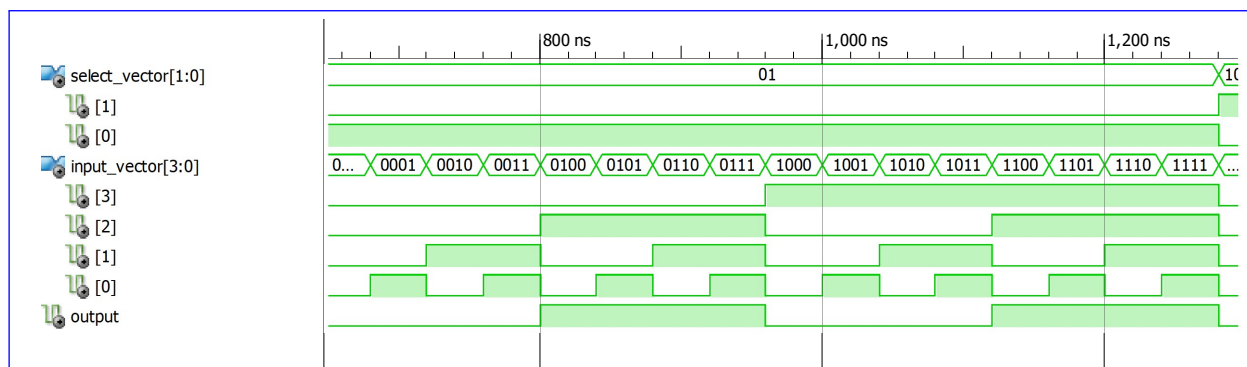


Figure 20: Simulation Waveform 700 ns to 1300 ns

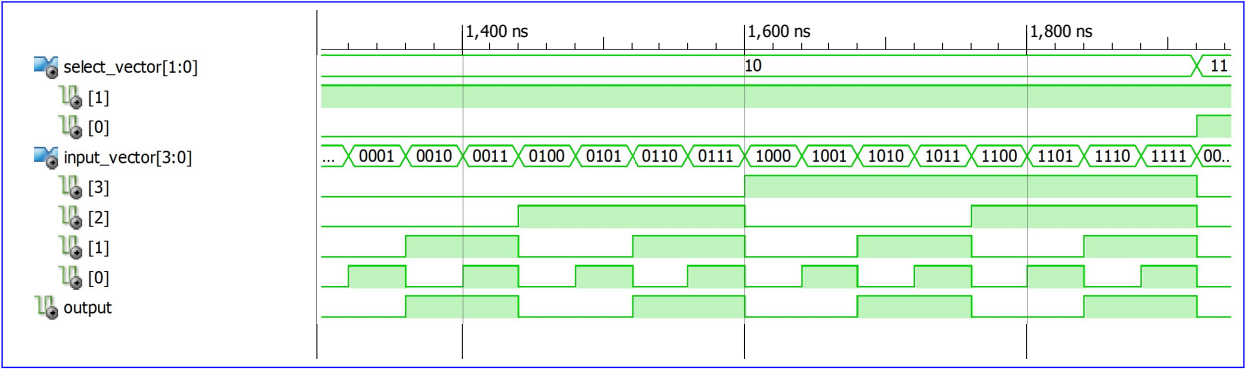


Figure 21: Simulation Waveform 1300 ns to 1900 ns

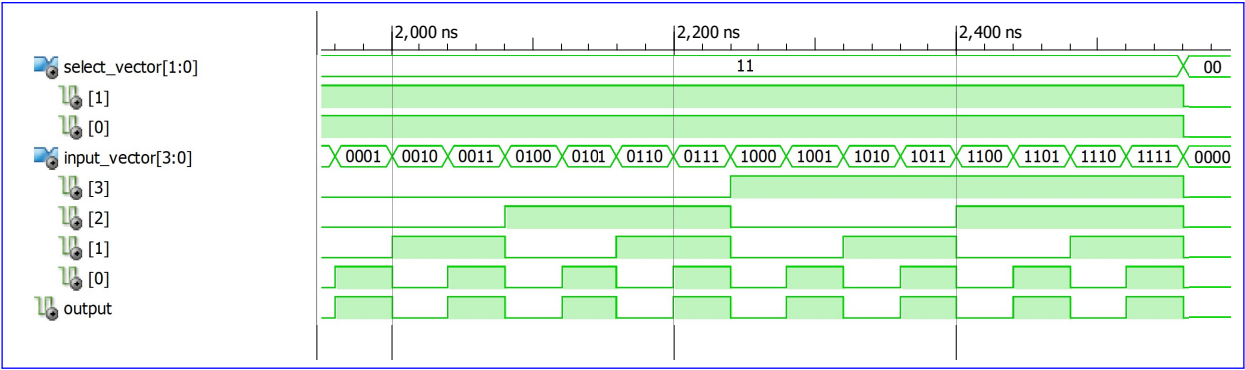


Figure 22: Simulation Waveform 1900 ns to 2600 ns

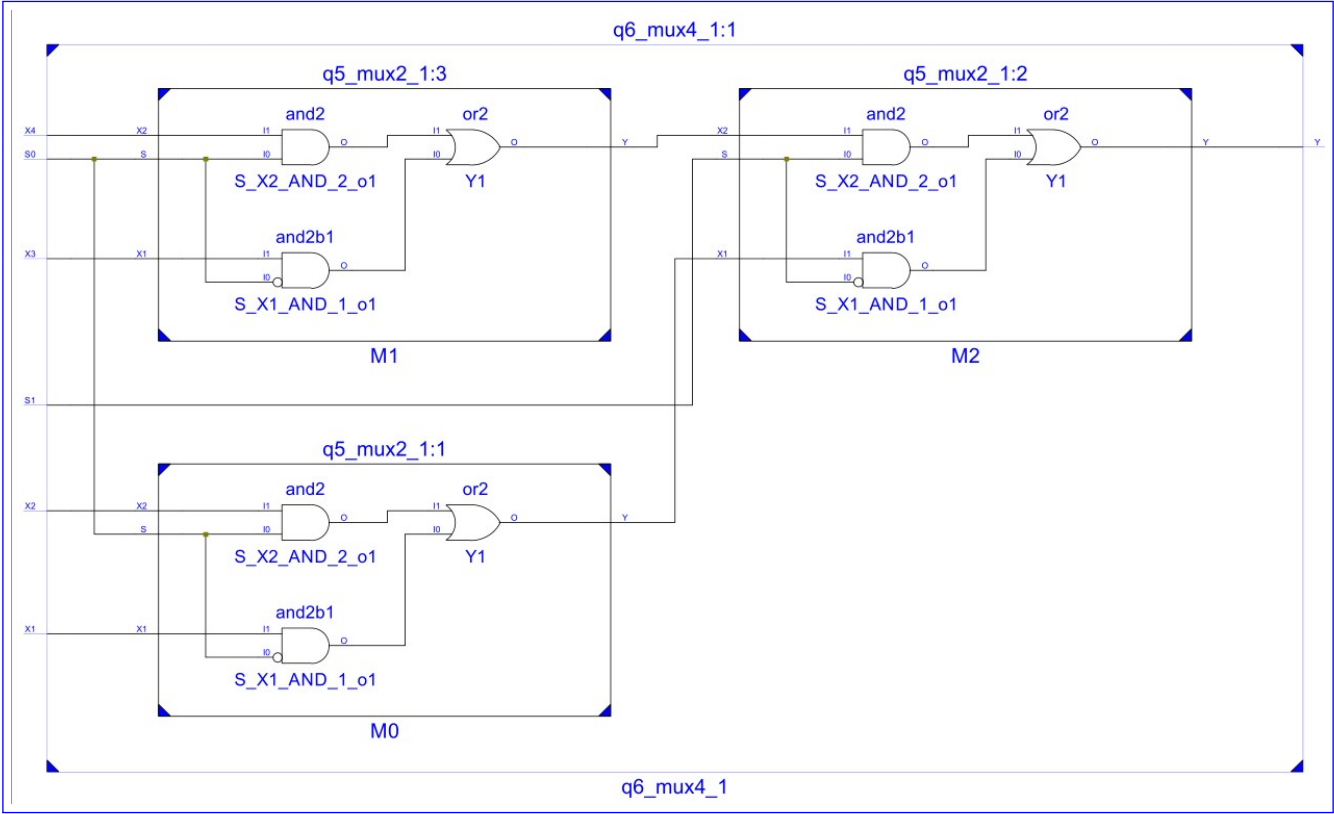


Figure 23: RTL Schematic

## 5.7 Question -7

Write VHDL code to implement a 4-bit adder/subtractor using four 1-bit full adders.

1. Use a Structural architecture style with hierarchical design approach:
  - (a) Use 1-bit adder as the basic building block
  - (b) Implement the 4-bit adder/subtractor using four 1-bit full adders.
2. Write a VHDL test bench to verify the operation of the 4-bit adder/subtractor.
3. Provide a simulation waveform depicting all possible input cases.

The required Boolean equation is :

$$S = X \oplus Y \oplus C_{in}$$

$$C_{out} = XY + (X \oplus Y)C_{in}$$

⇓ q7-xor.vhd ⇓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY q7_xor IS PORT (
5     i1, i2 : IN STD_LOGIC;
6     o1 : OUT STD_LOGIC
7 );
8
9
10 ARCHITECTURE dataflow OF q7_xor IS
11 BEGIN
12     o1 <= i1 XOR i2;
13 END dataflow;
```

Code 37: XOR gate implementation

⇓ full-adder.vhd ⇓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY full_adder IS PORT (
5     i1, i2, cin : IN STD_LOGIC;
6     sum, cout : OUT STD_LOGIC
7 );
8 END full_adder;
9
10 ARCHITECTURE dataflow OF full_adder IS
11 BEGIN
12     sum <= i1 XOR i2 XOR cin;
13     cout <= (i1 AND i2) OR ((i1 XOR i2) AND cin);
14 END dataflow;
```

Code 38: Full adder implementation

⇓ q7-st.vhd ⇓

```

1 LIBRARY IEEE;
2 USE IEEE.STD_LOGIC_1164.ALL;
3
4 ENTITY q7_add_sub IS PORT (
5     X3, X2, X1, X0, Y3, Y2, Y1, Y0, A_S : IN
6     STD_LOGIC;
7     S4, S3, S2, S1, S0 : OUT STD_LOGIC
8 );
9 END q7_add_sub;
10
11 ARCHITECTURE structural OF q7_add_sub IS
12     SIGNAL F0, F1, F2, F3, C1, C2, C3 :
13     STD_LOGIC;
14
15     COMPONENT q7_xor IS PORT (
16         i1, i2 : IN STD_LOGIC;
17         o1 : OUT STD_LOGIC
18     );
19     END COMPONENT;
20
21     COMPONENT full_adder IS PORT (
22         i1, i2, cin : IN STD_LOGIC;
23         sum, cout : OUT STD_LOGIC
24     );
25     END COMPONENT;
26
27 BEGIN
28     A0 : q7_xor PORT MAP(i1 => A_S, i2 => Y0,
29         o1 => F0);
30     A1 : full_adder PORT MAP(i1 => X0, i2 => F0,
31         cin => A_S, sum => S0, cout => C1);
32     A2 : q7_xor PORT MAP(i1 => A_S, i2 => Y1,
33         o1 => F1);
34     A3 : full_adder PORT MAP(i1 => X1, i2 => F1,
35         cin => C1, sum => S1, cout => C2);
36     A4 : q7_xor PORT MAP(i1 => A_S, i2 => Y2,
37         o1 => F2);
38     A5 : full_adder PORT MAP(i1 => X2, i2 => F2,
39         cin => C2, sum => S2, cout => C3);
40     A6 : q7_xor PORT MAP(i1 => A_S, i2 => Y3,
41         o1 => F3);
42     A7 : full_adder PORT MAP(i1 => X3, i2 => F3,
43         cin => C3, sum => S3, cout => C4);
44     A8 : q7_xor PORT MAP(i1 => A_S, i2 => Y0,
45         o1 => F0);
46     A9 : full_adder PORT MAP(i1 => X0, i2 => F0,
47         cin => A_S, sum => S0, cout => C1);
48     A10 : q7_xor PORT MAP(i1 => A_S, i2 => Y1,
49         o1 => F1);
50     A11 : full_adder PORT MAP(i1 => X1, i2 => F1,
51         cin => C1, sum => S1, cout => C2);
52     A12 : q7_xor PORT MAP(i1 => A_S, i2 => Y2,
53         o1 => F2);
54     A13 : full_adder PORT MAP(i1 => X2, i2 => F2,
55         cin => C2, sum => S2, cout => C3);
56     A14 : q7_xor PORT MAP(i1 => A_S, i2 => Y3,
57         o1 => F3);
58     A15 : full_adder PORT MAP(i1 => X3, i2 => F3,
59         cin => C3, sum => S3, cout => C4);
60     A16 : q7_xor PORT MAP(i1 => A_S, i2 => Y4,
61         o1 => F4);
62     A17 : full_adder PORT MAP(i1 => X4, i2 => F4,
63         cin => C4, sum => S4, cout => C5);
64     A18 : q7_xor PORT MAP(i1 => A_S, i2 => Y5,
65         o1 => F5);
66     A19 : full_adder PORT MAP(i1 => X5, i2 => F5,
67         cin => C5, sum => S5, cout => C6);
68     A20 : q7_xor PORT MAP(i1 => A_S, i2 => Y6,
69         o1 => F6);
70     A21 : full_adder PORT MAP(i1 => X6, i2 => F6,
71         cin => C6, sum => S6, cout => C7);
72     A22 : q7_xor PORT MAP(i1 => A_S, i2 => Y7,
73         o1 => F7);
74     A23 : full_adder PORT MAP(i1 => X7, i2 => F7,
75         cin => C7, sum => S7, cout => C8);
76     A24 : q7_xor PORT MAP(i1 => A_S, i2 => Y8,
77         o1 => F8);
78     A25 : full_adder PORT MAP(i1 => X8, i2 => F8,
79         cin => C8, sum => S8, cout => C9);
80     A26 : q7_xor PORT MAP(i1 => A_S, i2 => Y9,
81         o1 => F9);
82     A27 : full_adder PORT MAP(i1 => X9, i2 => F9,
83         cin => C9, sum => S9, cout => C10);
84     A28 : q7_xor PORT MAP(i1 => A_S, i2 => Y10,
85         o1 => F10);
86     A29 : full_adder PORT MAP(i1 => X10, i2 => F10,
87         cin => C10, sum => S10, cout => C11);
88     A30 : q7_xor PORT MAP(i1 => A_S, i2 => Y11,
89         o1 => F11);
90     A31 : full_adder PORT MAP(i1 => X11, i2 => F11,
91         cin => C11, sum => S11, cout => C12);
92     A32 : q7_xor PORT MAP(i1 => A_S, i2 => Y12,
93         o1 => F12);
94     A33 : full_adder PORT MAP(i1 => X12, i2 => F12,
95         cin => C12, sum => S12, cout => C13);
96     A34 : q7_xor PORT MAP(i1 => A_S, i2 => Y13,
97         o1 => F13);
98     A35 : full_adder PORT MAP(i1 => X13, i2 => F13,
99         cin => C13, sum => S13, cout => C14);
100     A36 : q7_xor PORT MAP(i1 => A_S, i2 => Y14,
101         o1 => F14);
102     A37 : full_adder PORT MAP(i1 => X14, i2 => F14,
103         cin => C14, sum => S14, cout => C15);
104     A38 : q7_xor PORT MAP(i1 => A_S, i2 => Y15,
105         o1 => F15);
106     A39 : full_adder PORT MAP(i1 => X15, i2 => F15,
107         cin => C15, sum => S15, cout => C16);
108     A40 : q7_xor PORT MAP(i1 => A_S, i2 => Y16,
109         o1 => F16);
110     A41 : full_adder PORT MAP(i1 => X16, i2 => F16,
111         cin => C16, sum => S16, cout => C17);
112     A42 : q7_xor PORT MAP(i1 => A_S, i2 => Y17,
113         o1 => F17);
114     A43 : full_adder PORT MAP(i1 => X17, i2 => F17,
115         cin => C17, sum => S17, cout => C18);
116     A44 : q7_xor PORT MAP(i1 => A_S, i2 => Y18,
117         o1 => F18);
118     A45 : full_adder PORT MAP(i1 => X18, i2 => F18,
119         cin => C18, sum => S18, cout => C19);
120     A46 : q7_xor PORT MAP(i1 => A_S, i2 => Y19,
121         o1 => F19);
122     A47 : full_adder PORT MAP(i1 => X19, i2 => F19,
123         cin => C19, sum => S19, cout => C20);
124     A48 : q7_xor PORT MAP(i1 => A_S, i2 => Y20,
125         o1 => F20);
126     A49 : full_adder PORT MAP(i1 => X20, i2 => F20,
127         cin => C20, sum => S20, cout => C21);
128     A50 : q7_xor PORT MAP(i1 => A_S, i2 => Y21,
129         o1 => F21);
130     A51 : full_adder PORT MAP(i1 => X21, i2 => F21,
131         cin => C21, sum => S21, cout => C22);
132     A52 : q7_xor PORT MAP(i1 => A_S, i2 => Y22,
133         o1 => F22);
134     A53 : full_adder PORT MAP(i1 => X22, i2 => F22,
135         cin => C22, sum => S22, cout => C23);
136     A54 : q7_xor PORT MAP(i1 => A_S, i2 => Y23,
137         o1 => F23);
138     A55 : full_adder PORT MAP(i1 => X23, i2 => F23,
139         cin => C23, sum => S23, cout => C24);
140     A56 : q7_xor PORT MAP(i1 => A_S, i2 => Y24,
141         o1 => F24);
142     A57 : full_adder PORT MAP(i1 => X24, i2 => F24,
143         cin => C24, sum => S24, cout => C25);
144     A58 : q7_xor PORT MAP(i1 => A_S, i2 => Y25,
145         o1 => F25);
146     A59 : full_adder PORT MAP(i1 => X25, i2 => F25,
147         cin => C25, sum => S25, cout => C26);
148     A60 : q7_xor PORT MAP(i1 => A_S, i2 => Y26,
149         o1 => F26);
150     A61 : full_adder PORT MAP(i1 => X26, i2 => F26,
151         cin => C26, sum => S26, cout => C27);
152     A62 : q7_xor PORT MAP(i1 => A_S, i2 => Y27,
153         o1 => F27);
154     A63 : full_adder PORT MAP(i1 => X27, i2 => F27,
155         cin => C27, sum => S27, cout => C28);
156     A64 : q7_xor PORT MAP(i1 => A_S, i2 => Y28,
157         o1 => F28);
158     A65 : full_adder PORT MAP(i1 => X28, i2 => F28,
159         cin => C28, sum => S28, cout => C29);
160     A66 : q7_xor PORT MAP(i1 => A_S, i2 => Y29,
161         o1 => F29);
162     A67 : full_adder PORT MAP(i1 => X29, i2 => F29,
163         cin => C29, sum => S29, cout => C30);
164     A68 : q7_xor PORT MAP(i1 => A_S, i2 => Y30,
165         o1 => F30);
166     A69 : full_adder PORT MAP(i1 => X30, i2 => F30,
167         cin => C30, sum => S30, cout => C31);
168     A70 : q7_xor PORT MAP(i1 => A_S, i2 => Y31,
169         o1 => F31);
170     A71 : full_adder PORT MAP(i1 => X31, i2 => F31,
171         cin => C31, sum => S31, cout => C32);
172     A72 : q7_xor PORT MAP(i1 => A_S, i2 => Y32,
173         o1 => F32);
174     A73 : full_adder PORT MAP(i1 => X32, i2 => F32,
175         cin => C32, sum => S32, cout => C33);
176     A74 : q7_xor PORT MAP(i1 => A_S, i2 => Y33,
177         o1 => F33);
178     A75 : full_adder PORT MAP(i1 => X33, i2 => F33,
179         cin => C33, sum => S33, cout => C34);
180     A76 : q7_xor PORT MAP(i1 => A_S, i2 => Y34,
181         o1 => F34);
182     A77 : full_adder PORT MAP(i1 => X34, i2 => F34,
183         cin => C34, sum => S34, cout => C35);
184     A78 : q7_xor PORT MAP(i1 => A_S, i2 => Y35,
185         o1 => F35);
186     A79 : full_adder PORT MAP(i1 => X35, i2 => F35,
187         cin => C35, sum => S35, cout => C36);
188     A80 : q7_xor PORT MAP(i1 => A_S, i2 => Y36,
189         o1 => F36);
190     A81 : full_adder PORT MAP(i1 => X36, i2 => F36,
191         cin => C36, sum => S36, cout => C37);
192     A82 : q7_xor PORT MAP(i1 => A_S, i2 => Y37,
193         o1 => F37);
194     A83 : full_adder PORT MAP(i1 => X37, i2 => F37,
195         cin => C37, sum => S37, cout => C38);
196     A84 : q7_xor PORT MAP(i1 => A_S, i2 => Y38,
197         o1 => F38);
198     A85 : full_adder PORT MAP(i1 => X38, i2 => F38,
199         cin => C38, sum => S38, cout => C39);
200     A86 : q7_xor PORT MAP(i1 => A_S, i2 => Y39,
201         o1 => F39);
202     A87 : full_adder PORT MAP(i1 => X39, i2 => F39,
203         cin => C39, sum => S39, cout => C40);
204     A88 : q7_xor PORT MAP(i1 => A_S, i2 => Y40,
205         o1 => F40);
206     A89 : full_adder PORT MAP(i1 => X40, i2 => F40,
207         cin => C40, sum => S40, cout => C41);
208     A90 : q7_xor PORT MAP(i1 => A_S, i2 => Y41,
209         o1 => F41);
210     A91 : full_adder PORT MAP(i1 => X41, i2 => F41,
211         cin => C41, sum => S41, cout => C42);
212     A92 : q7_xor PORT MAP(i1 => A_S, i2 => Y42,
213         o1 => F42);
214     A93 : full_adder PORT MAP(i1 => X42, i2 => F42,
215         cin => C42, sum => S42, cout => C43);
216     A94 : q7_xor PORT MAP(i1 => A_S, i2 => Y43,
217         o1 => F43);
218     A95 : full_adder PORT MAP(i1 => X43, i2 => F43,
219         cin => C43, sum => S43, cout => C44);
220     A96 : q7_xor PORT MAP(i1 => A_S, i2 => Y44,
221         o1 => F44);
222     A97 : full_adder PORT MAP(i1 => X44, i2 => F44,
223         cin => C44, sum => S44, cout => C45);
224     A98 : q7_xor PORT MAP(i1 => A_S, i2 => Y45,
225         o1 => F45);
226     A99 : full_adder PORT MAP(i1 => X45, i2 => F45,
227         cin => C45, sum => S45, cout => C46);
228     A100 : q7_xor PORT MAP(i1 => A_S, i2 => Y46,
229         o1 => F46);
230     A101 : full_adder PORT MAP(i1 => X46, i2 => F46,
231         cin => C46, sum => S46, cout => C47);
232     A102 : q7_xor PORT MAP(i1 => A_S, i2 => Y47,
233         o1 => F47);
234     A103 : full_adder PORT MAP(i1 => X47, i2 => F47,
235         cin => C47, sum => S47, cout => C48);
236     A104 : q7_xor PORT MAP(i1 => A_S, i2 => Y48,
237         o1 => F48);
238     A105 : full_adder PORT MAP(i1 => X48, i2 => F48,
239         cin => C48, sum => S48, cout => C49);
240     A106 : q7_xor PORT MAP(i1 => A_S, i2 => Y49,
241         o1 => F49);
242     A107 : full_adder PORT MAP(i1 => X49, i2 => F49,
243         cin => C49, sum => S49, cout => C50);
244     A108 : q7_xor PORT MAP(i1 => A_S, i2 => Y50,
245         o1 => F50);
246     A109 : full_adder PORT MAP(i1 => X50, i2 => F50,
247         cin => C50, sum => S50, cout => C51);
248     A110 : q7_xor PORT MAP(i1 => A_S, i2 => Y51,
249         o1 => F51);
250     A111 : full_adder PORT MAP(i1 => X51, i2 => F51,
251         cin => C51, sum => S51, cout => C52);
252     A112 : q7_xor PORT MAP(i1 => A_S, i2 => Y52,
253         o1 => F52);
254     A113 : full_adder PORT MAP(i1 => X52, i2 => F52,
255         cin => C52, sum => S52, cout => C53);
256     A114 : q7_xor PORT MAP(i1 => A_S, i2 => Y53,
257         o1 => F53);
258     A115 : full_adder PORT MAP(i1 => X53, i2 => F53,
259         cin => C53, sum => S53, cout => C54);
260     A116 : q7_xor PORT MAP(i1 => A_S, i2 => Y54,
261         o1 => F54);
262     A117 : full_adder PORT MAP(i1 => X54, i2 => F54,
263         cin => C54, sum => S54, cout => C55);
264     A118 : q7_xor PORT MAP(i1 => A_S, i2 => Y55,
265         o1 => F55);
266     A119 : full_adder PORT MAP(i1 => X55, i2 => F55,
267         cin => C55, sum => S55, cout => C56);
268     A120 : q7_xor PORT MAP(i1 => A_S, i2 => Y56,
269         o1 => F56);
270     A121 : full_adder PORT MAP(i1 => X56, i2 => F56,
271         cin => C56, sum => S56, cout => C57);
272     A122 : q7_xor PORT MAP(i1 => A_S, i2 => Y57,
273         o1 => F57);
274     A123 : full_adder PORT MAP(i1 => X57, i2 => F57,
275         cin => C57, sum => S57, cout => C58);
276     A124 : q7_xor PORT MAP(i1 => A_S, i2 => Y58,
277         o1 => F58);
278     A125 : full_adder PORT MAP(i1 => X58, i2 => F58,
279         cin => C58, sum => S58, cout => C59);
280     A126 : q7_xor PORT MAP(i1 => A_S, i2 => Y59,
281         o1 => F59);
282     A127 : full_adder PORT MAP(i1 => X59, i2 => F59,
283         cin => C59, sum => S59, cout => C60);
284     A128 : q7_xor PORT MAP(i1 => A_S, i2 => Y60,
285         o1 => F60);
286     A129 : full_adder PORT MAP(i1 => X60, i2 => F60,
287         cin => C60, sum => S60, cout => C61);
288     A130 : q7_xor PORT MAP(i1 => A_S, i2 => Y61,
289         o1 => F61);
290     A131 : full_adder PORT MAP(i1 => X61, i2 => F61,
291         cin => C61, sum => S61, cout => C62);
292     A132 : q7_xor PORT MAP(i1 => A_S, i2 => Y62,
293         o1 => F62);
294     A133 : full_adder PORT MAP(i1 => X62, i2 => F62,
295         cin => C62, sum => S62, cout => C63);
296     A134 : q7_xor PORT MAP(i1 => A_S, i2 => Y63,
297         o1 => F63);
298     A135 : full_adder PORT MAP(i1 => X63, i2 => F63,
299         cin => C63, sum => S63, cout => C64);
300     A136 : q7_xor PORT MAP(i1 => A_S, i2 => Y64,
301         o1 => F64);
302     A137 : full_adder PORT MAP(i1 => X64, i2 => F64,
303         cin => C64, sum => S64, cout => C65);
304     A138 : q7_xor PORT MAP(i1 => A_S, i2 => Y65,
305         o1 => F65);
306     A139 : full_adder PORT MAP(i1 => X65, i2 => F65,
307         cin => C65, sum => S65, cout => C66);
308     A140 : q7_xor PORT MAP(i1 => A_S, i2 => Y66,
309         o1 => F66);
310     A141 : full_adder PORT MAP(i1 => X66, i2 => F66,
311         cin => C66, sum => S66, cout => C67);
312     A142 : q7_xor PORT MAP(i1 => A_S, i2 => Y67,
313         o1 => F67);
314     A143 : full_adder PORT MAP(i1 => X67, i2 => F67,
315         cin => C67, sum => S67, cout => C68);
316     A144 : q7_xor PORT MAP(i1 => A_S, i2 => Y68,
317         o1 => F68);
318     A145 : full_adder PORT MAP(i1 => X68, i2 => F68,
319         cin => C68, sum => S68, cout => C69);
320     A146 : q7_xor PORT MAP(i1 => A_S, i2 => Y69,
321         o1 => F69);
322     A147 : full_adder PORT MAP(i1 => X69, i2 => F69,
323         cin => C69, sum => S69, cout => C70);
324     A148 : q7_xor PORT MAP(i1 => A_S, i2 => Y70,
325         o1 => F70);
326     A149 : full_adder PORT MAP(i1 => X70, i2 => F70,
327         cin => C70, sum => S70, cout => C71);
328     A150 : q7_xor PORT MAP(i1 => A_S, i2 => Y71,
329         o1 => F71);
330     A151 : full_adder PORT MAP(i1 => X71, i2 => F71,
331         cin => C71, sum => S71, cout => C72);
332     A152 : q7_xor PORT MAP(i1 => A_S, i2 => Y72,
333         o1 => F72);
334     A153 : full_adder PORT MAP(i1 => X72, i2 => F72,
335         cin => C72, sum => S72, cout => C73);
336     A154 : q7_xor PORT MAP(i1 => A_S, i2 => Y73,
337         o1 => F73);
338     A155 : full_adder PORT MAP(i1 => X73, i2 => F73,
339         cin => C73, sum => S73, cout => C74);
340     A156 : q7_xor PORT MAP(i1 => A_S, i2 => Y74,
341         o1 => F74);
342     A157 : full_adder PORT MAP(i1 => X74, i2 => F74,
343         cin => C74, sum => S74, cout => C75);
344     A158 : q7_xor PORT MAP(i1 => A_S, i2 => Y75,
345         o1 => F75);
346     A159 : full_adder PORT MAP(i1 => X75, i2 => F75,
347         cin => C75, sum => S75, cout => C76);
348     A160 : q7_xor PORT MAP(i1 => A_S, i2 => Y76,
349         o1 => F76);
350     A161 : full_adder PORT MAP(i1 => X76, i2 => F76,
351         cin => C76, sum => S76, cout => C77);
352     A162 : q7_xor PORT MAP(i1 => A_S, i2 => Y77,
353         o1 => F77);
354     A163 : full_adder PORT MAP(i1 => X77, i2 => F77,
355         cin => C77, sum => S77, cout => C78);
356     A164 : q7_xor PORT MAP(i1 => A_S, i2 => Y78,
357         o1 => F78);
358     A165 : full_adder PORT MAP(i1 => X78, i2 => F78,
359         cin => C78, sum => S78, cout => C79);
360     A166 : q7_xor PORT MAP(i1 => A_S, i2 => Y79,
361         o1 => F79);
362     A167 : full_adder PORT MAP(i1 => X79, i2 => F79,
363         cin => C79, sum => S79, cout => C80);
364     A168 : q7_xor PORT MAP(i1 => A_S, i2 => Y80,
365         o1 => F80);
366     A169 : full_adder PORT MAP(i1 => X80, i2 => F80,
367         cin => C80, sum => S80, cout => C81);
368     A170 : q7_xor PORT MAP(i1 => A_S, i2 => Y81,
369         o1 => F81);
370     A171 : full_adder PORT MAP(i1 => X81, i2 => F81,
371         cin => C81, sum => S81, cout => C82);
372     A172 : q7_xor PORT MAP(i1 => A_S, i2 => Y82,
373         o1 => F82);
374     A173 : full_adder PORT MAP(i1 => X82, i2 => F82,
375         cin => C82, sum => S82, cout => C83);
376     A174 : q7_xor PORT MAP(i1 => A_S, i2 => Y83,
377         o1 => F83);
378     A175 : full_adder PORT MAP(i1 => X83, i2 => F83,
379         cin => C83, sum => S83, cout => C84);
380     A176 : q7_xor PORT MAP(i1 => A_S, i2 => Y84,
381         o1 => F84);
382     A177 : full_adder PORT MAP(i1 => X84, i2 => F84,
383         cin => C84, sum => S84, cout => C85);
384     A178 : q7_xor PORT MAP(i1 => A_S, i2 => Y85,
385         o1 => F85);
386     A179 : full_adder PORT MAP(i1 => X85, i2 => F85,
387         cin => C85, sum => S85, cout => C86);
388     A180 : q7_xor PORT MAP(i1 => A_S, i2 => Y86,
389         o1 => F86);
390     A181 : full_adder PORT MAP(i1 => X86, i2 => F86,
391         cin => C86, sum => S86, cout => C87);
392     A182 : q7_xor PORT MAP(i1 => A_S, i2 => Y87,
393         o1 => F87);
394     A183 : full_adder PORT MAP(i1 => X87, i2 => F87,
395         cin => C87, sum => S87, cout => C88);
396     A184 : q7_xor PORT MAP(i1 => A_S, i2 => Y88,
397         o1 => F88);
398     A185 : full_adder PORT MAP(i1 => X88, i2 => F88,
399         cin => C88, sum => S88, cout => C89);
400     A186 : q7_xor PORT MAP(i1 => A_S, i2 => Y89,
401         o1 => F89);
402     A187 : full_adder PORT MAP(i1 => X89, i2 => F89,
403         cin => C89, sum => S89, cout => C90);
404     A188 : q7_xor PORT MAP(i1 => A_S, i2 => Y90,
405         o1 => F90);
406     A189 : full_adder PORT MAP(i1 => X90, i2 => F90,
407         cin => C90, sum => S90, cout => C91);
408     A190 : q7_xor PORT MAP(i1 => A_S, i2 => Y91,
409         o1 => F91);
410     A191 : full_adder PORT MAP(i1 => X91, i2 => F91,
411         cin => C91, sum => S91, cout => C92);
412     A192 : q7_xor PORT MAP(i1 => A_S, i2 => Y92,
413         o1 => F92);
414     A193 : full_adder PORT MAP(i1 => X92, i2 => F92,
415         cin => C92, sum => S92, cout => C93);
416     A194 : q7_xor PORT MAP(i1 => A_S, i2 => Y93,
417         o1 => F93);
418     A195 : full_adder PORT MAP(i1 => X93, i2 => F93,
419         cin => C93, sum => S93, cout => C94);
420     A196 : q7_xor PORT MAP(i1 => A_S, i2 => Y94,
421         o1 => F94);
422     A197 : full_adder PORT MAP(i1 => X94, i2 => F94,
423         cin => C94, sum => S94, cout => C95);
424     A198 : q7_xor PORT MAP(i1 => A_S, i2 => Y95,
425         o1 => F95);
426     A199 : full_adder PORT MAP(i1 => X95, i2 => F95,
427         cin => C95, sum => S95, cout => C96);
428     A200 : q7_xor PORT MAP(i1 => A_S, i2 => Y96,
429         o1 => F96);
430     A201 : full_adder PORT MAP(i1 => X96, i2 => F96,
431         cin => C96, sum => S96, cout => C97);
432     A202 : q7_xor PORT MAP(i1 => A_S, i2 => Y97,
433         o1 => F97);
434     A203 : full_adder PORT MAP(i1 => X97, i2 => F97,
435         cin => C97, sum => S97, cout => C98);
436     A204 : q7_xor PORT MAP(i1 => A_S, i2 => Y98,
437         o1 => F98);
438     A205 : full_adder PORT MAP(i1 => X98, i2 => F98,
439         cin => C98, sum => S98, cout => C99);
440     A206 : q7_xor PORT MAP(i1 => A_S, i2 => Y99,
441         o1 => F99);
442     A207 : full_adder PORT MAP(i1 => X99, i2 => F99,
443         cin => C99, sum => S99, cout => C100);
444     A208 : q7_xor PORT MAP(i1 => A_S, i2 => Y100,
445         o1 => F100);
446     A209 : full_adder PORT MAP(i1 => X100, i2 => F100,
447         cin => C100, sum => S100, cout => C101);
448     A210 : q7_xor PORT MAP(i
```



```

30  A3 : full_adder PORT MAP(i1 => X1, i2 => F1
    , cin => C1, sum => S1, cout => C2);
31
32  A4 : q7_xor PORT MAP(i1 => A_S, i2 => Y2,
    o1 => F2);
33  A5 : full_adder PORT MAP(i1 => X2, i2 => F2
    , cin => C2, sum => S2, cout => C3);
34
35  A6 : q7_xor PORT MAP(i1 => A_S, i2 => Y3,
    o1 => F3);
36  A7 : full_adder PORT MAP(i1 => X3, i2 => F3
    , cin => C3, sum => S3, cout => S4);
37  END structural;

```

Code 39: 4-bit adder/subtractor implementation using four 1-bit full-adder: Structural model

↓ q7-tb.vhd ↓

```

1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.NUMERIC_STD.ALL;
4
5  ENTITY q7_add_sub_tb IS
6  END q7_add_sub_tb;
7
8  ARCHITECTURE behavioral OF q7_add_sub_tb IS
9      -- component declaration for the Unit
10     Under Test (UUT)
11     COMPONENT q7_add_sub
12     PORT (
13         X3, X2, X1, X0, Y3, Y2, Y1, Y0, A_S :
14         IN STD_LOGIC;
15         S4, S3, S2, S1, S0 : OUT STD_LOGIC
16     );
17 END COMPONENT;
18
19 SIGNAL addsub : STD_LOGIC := '1';
20 --Change addsub to '0' for 4-bit full-adder
21 SIGNAL input_vector1 : STD_LOGIC_VECTOR(3
22     DOWNT0 0) := "0000";
23 SIGNAL input_vector2 : STD_LOGIC_VECTOR(3
24     DOWNT0 0) := "0000";
25 SIGNAL output_vector : STD_LOGIC_VECTOR(4
26     DOWNT0 0) := "00000";
27
28 BEGIN
29     --INSTANTIATE the unit under test
30     uut : q7_add_sub PORT MAP(
31         A_S => addsub,
32         X3 => input_vector1(3),
33         X2 => input_vector1(2),
34         X1 => input_vector1(1),
35         X0 => input_vector1(0),
36         Y3 => input_vector2(3),
37         Y2 => input_vector2(2),
38         Y1 => input_vector2(1),
39         Y0 => input_vector2(0),
40         S4 => output_vector(4),
41         S3 => output_vector(3),
42         S2 => output_vector(2),
43         S1 => output_vector(1),
44         S0 => output_vector(0)
45     );
46
47     --stimulus process
48     stim_proc : PROCESS
49     BEGIN
50         FOR index1 IN 0 TO 15 LOOP
51             input_vector1 <= STD_LOGIC_VECTOR(
52                 to_unsigned(index1, 4));
53             FOR index2 IN 0 TO 15 LOOP
54                 input_vector2 <= STD_LOGIC_VECTOR(
55                     to_unsigned(index2, 4));
56                 WAIT FOR 50 ns;
57             END LOOP;
58         END LOOP;
59     END PROCESS;
60 END behavioral;

```

Code 40: Testbench for all possible cases

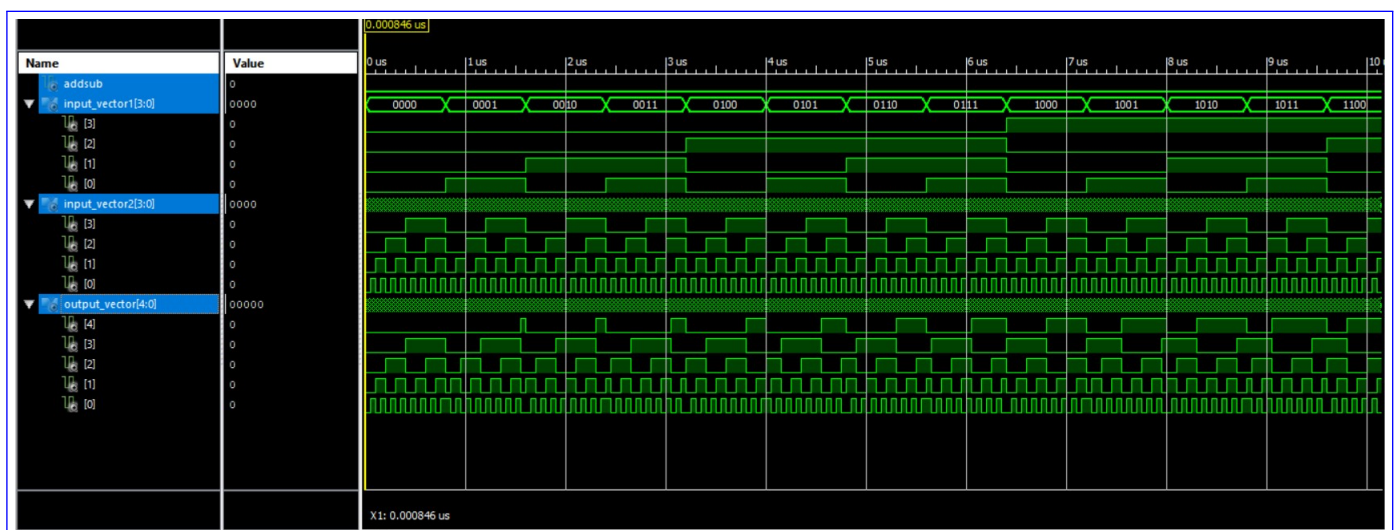


Figure 24: Simulation Waveform 4 bit Adder

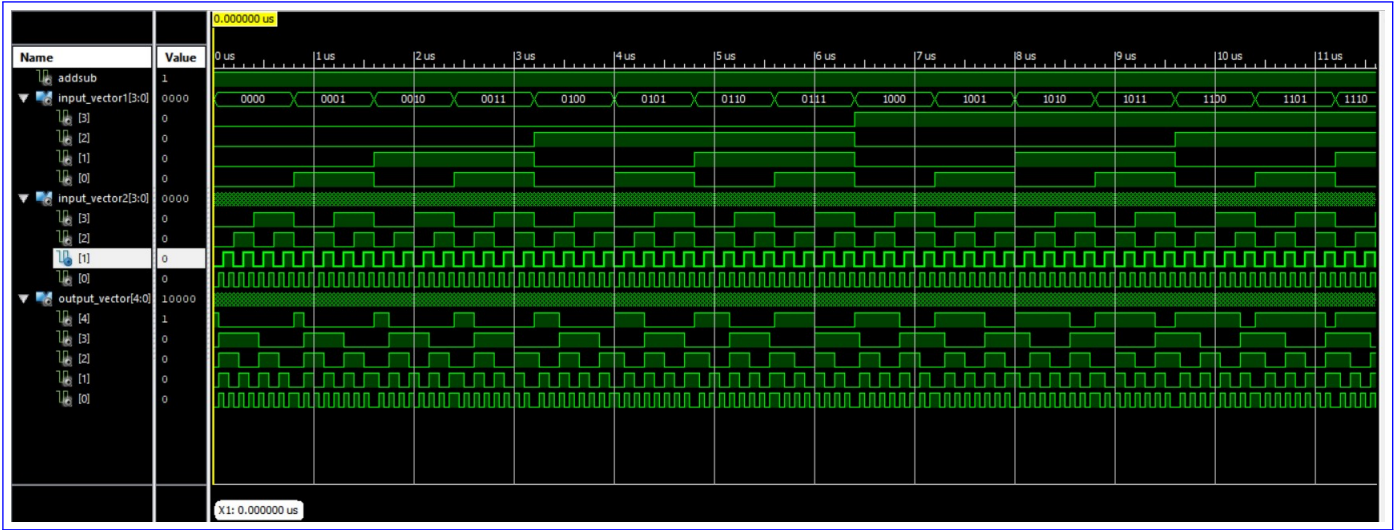


Figure 25: Simulation Waveform for 4 bit Subtractor

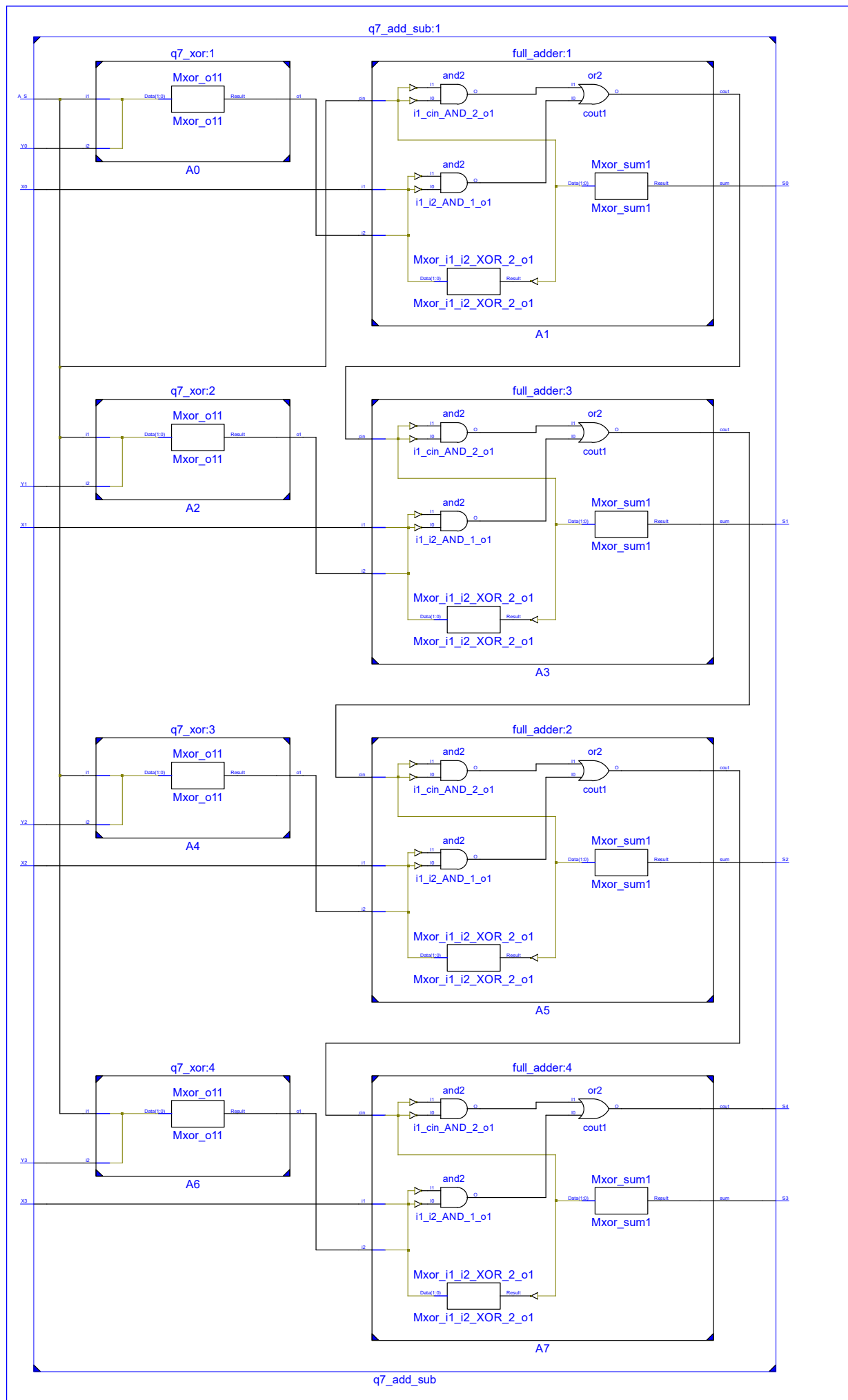


Figure 26: RTL Schematic

## 6 Conclusion

In this Lab we programs various circuits in VHDL using Xilinx. We also familiarize ourselves with different architectural model like dataflow, behavioral and structural. Different waveforms and RTL schematic are included in this report.