INSTITUTE OF ENGINNERING , 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

List of Figures

Problem 1

Problem 2

Problem 3

Problem 4

Problem 5

Problem 6

Problem 7

List of VHDLCodes

Problem 1

Problem 2

Problem 3

Problem 4

SOP

POS

Problem 5

Problem 6

Problem 7

1 Title

Combinational Logic Design Using VHDL

2 Objective

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

- 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 cableSoftware:

Software:

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

4 Introduction

VHDL stands for Very High Speed Integrated Circuit (VHSIC) Hardware DescriptiveLanguage. 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 has4 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.**

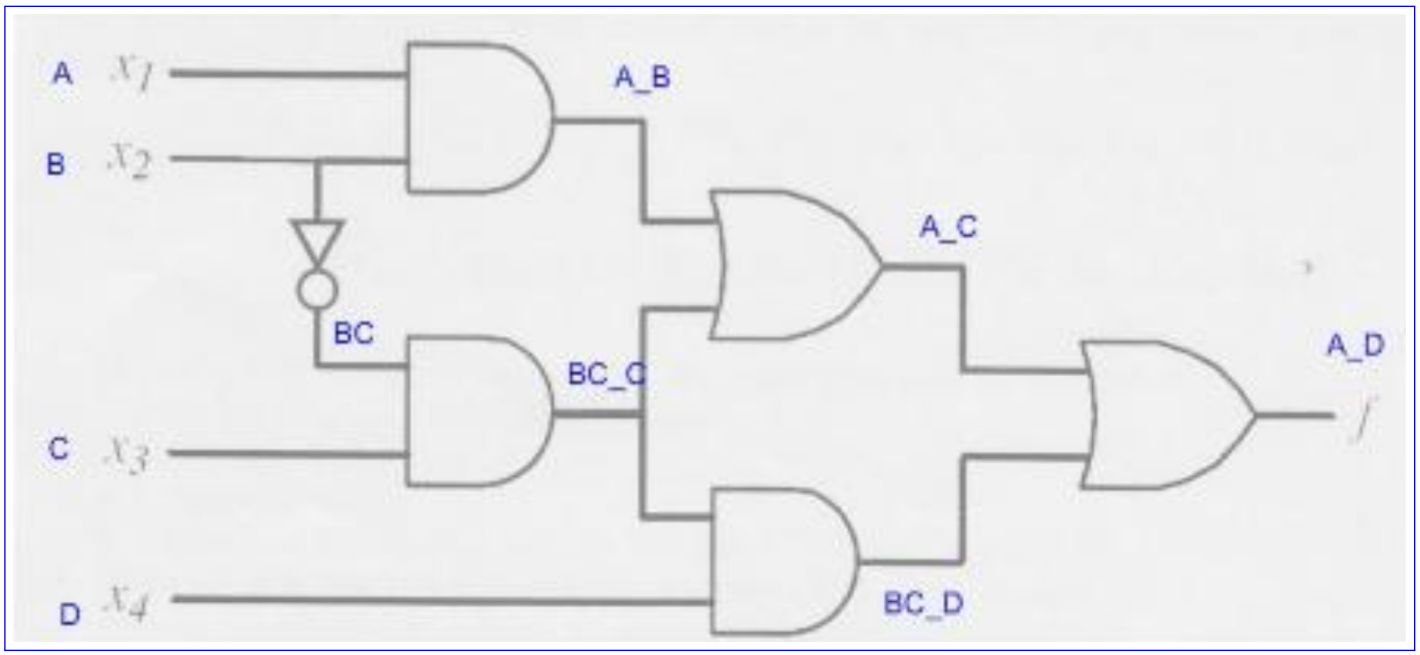


Figure 1: Combinational Circuit for

⇓ q1-df.vhd ⇓

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q1 IS PORT (
   A, B, C, D : IN STD_LOGIC;
   F : OUT STD_LOGIC
);
END q1;

ARCHITECTURE dataflow OF q1 IS
BEGIN
   F <= (A AND B) OR (NOT B AND C);
END dataflow;
```

Code 1: Dataflow model

⇓ and1.vhd ⇓

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY and1 IS PORT (
   i1, i2 : IN STD_LOGIC;
   o1 : OUT STD_LOGIC
);
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 behavorial 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 behavorial;
```

Code 5: Behavorial 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 (
```

```
      i1, i2 : IN STD_LOGIC;
      o1 : OUT STD_LOGIC
      );
   END COMPONENT;

   COMPONENT or1 IS PORT (
      i1, i2 : IN STD_LOGIC;
      o1 : OUT STD_LOGIC
      );
   END COMPONENT;


BEGIN
   C1 : and1 PORT MAP(i1 => A, i2 => B, o1 => F1);
   C2 : andnot PORT MAP(i1 => B, i2 => C, o1 => F2);
   C3 : or1 PORT MAP(i1 => F1, i2 => F2, o1 => F);
END structural;
```

Code 6: Structural model

⇓ q1-tb.vhd ⇓

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

ENTITY q1_tb IS
END q1_tb;

ARCHITECTURE behavioral OF q1_tb IS
   -- component declaaration for the Unit Under Test (UUT)
   COMPONENT q1
      PORT (
         A : IN STD_LOGIC;
         B : IN STD_LOGIC;
         C : IN STD_LOGIC;
         D : IN STD_LOGIC;
         F : OUT STD_LOGIC
      );
   END COMPONENT;

   SIGNAL input_vector : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0000";
   SIGNAL output : STD_LOGIC;


BEGIN
   --INSTANTIATE the unit under test
   uut : q1 PORT MAP(
      A => input_vector(3),
      B => input_vector(2),
      C => input_vector(1),
      D => input_vector(0),
      F => output
   );

   --stimulus process
   stim_proc : PROCESS

   BEGIN
      FOR index IN 0 TO 15 LOOP
         input_vector <= STD_LOGIC_VECTOR(to_unsigned(index, 4));
         WAIT FOR 50 ns;
      END LOOP;
   END PROCESS;
END behavioral;
```

Code 7: Testbench for all cases

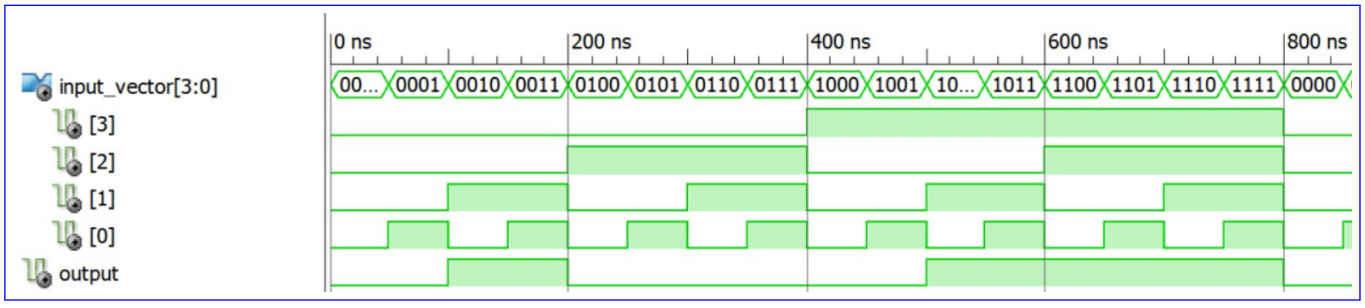


Figure 2: Simulation Waveform

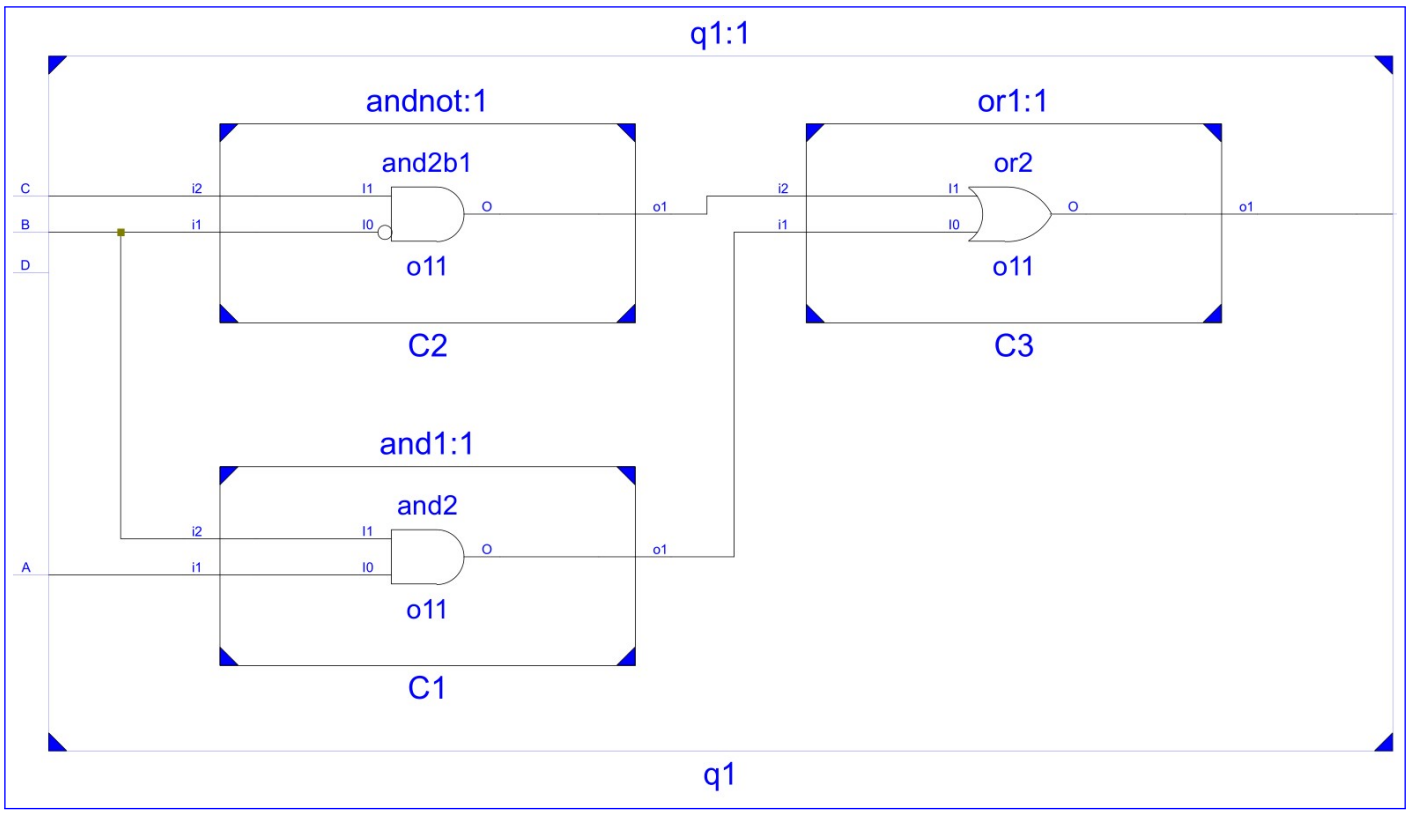


Figure 3: RTL Schematic

5.2 Question -2

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

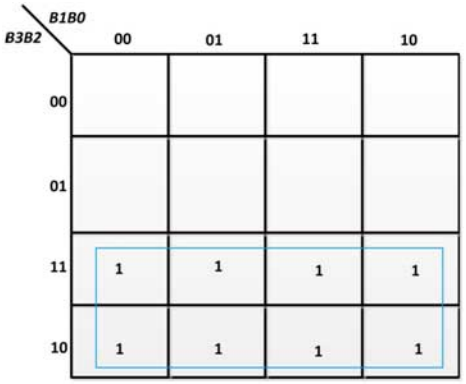
1. **Use Karnaugh maps to simplify the output function.**
2. **Provide the following architectural styles:**
   - **Dateflow style**
   - **Behavioral style**
   - **Structural style using only NOR gates**
3. **Write a VHDL test bench to verify the operation of the logic circuit.**
4. **Provide a simulation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

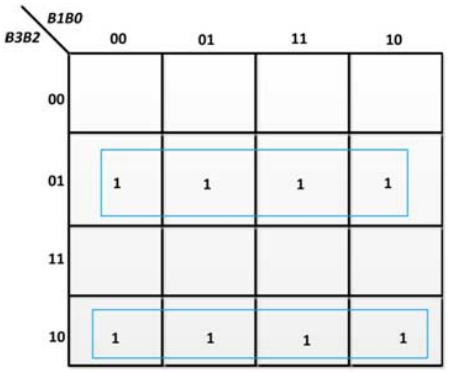
⇓ q2-df.vhd ⇓

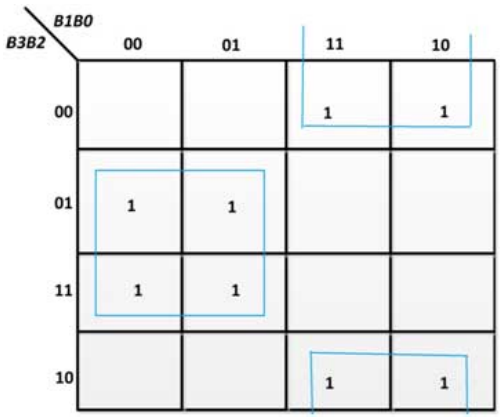
```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q2 IS PORT (
   X3, X2, X1, X0 : IN STD_LOGIC;
   Y3, Y2, Y1, Y0 : OUT STD_LOGIC
);
END q2;

ARCHITECTURE dataflow OF q2 IS

BEGIN
   Y3 <= X3;
   Y2 <= X2 XOR X3;
   Y1 <= X1 XOR X2;
   Y0 <= X0 XOR X1;
END dataflow;
```

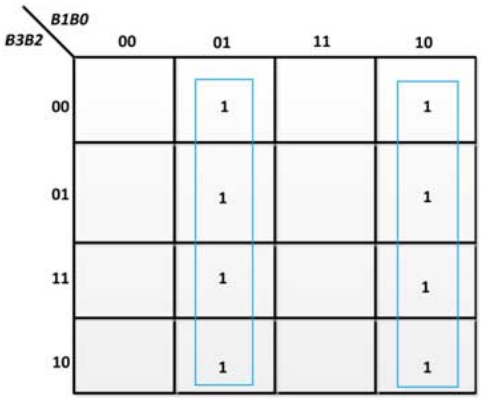
Code 8: Dataflow model

BINARY INPUT				GRAY CODE INPUT			
B3	B2	B1	B0	G3	G2	G1	G0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0









G3 = B3 G2 = B3 ⊕ B2 G1 = B2 ⊕ B1 G0 = B1 ⊕ B0

Y3 = X3 Y2 = X3 ⊕ X2 Y1 = X2 ⊕ X1 Y0 = X1 ⊕ X0

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

⇓ **q2-be.vhd** ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q2 IS PORT (
   X3, X2, X1, X0 : IN STD_LOGIC;
   Y3, Y2, Y1, Y0 : OUT STD_LOGIC
);
END q2;

ARCHITECTURE behavorial OF q2 IS

BEGIN
   example : PROCESS (X3, X2, X1, X0)

   BEGIN
      Y3 <= X3;
      Y2 <= X2 XOR X3;
      Y1 <= X1 XOR X2;
      Y0 <= X0 XOR X1;
   END PROCESS example;

END behavorial;
```

Code 9: Behavioral model

⇓ **xor-nor.vhd** ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY xor_nor IS PORT (
   a, b : IN STD_LOGIC;
   o : OUT STD_LOGIC
);
END xor_nor;

ARCHITECTURE behavorial OF xor_nor IS
   SIGNAL nota, notb, xnorab : STD_LOGIC;

BEGIN
   xor_nor : PROCESS (a, b, nota, notb, xnorab
    )

   BEGIN
      nota <= a NOR a;
      notb <= b NOR b;
      xnorab <= (a NOR notb) NOR (nota NOR b);
      o <= xnorab NOR xnorab;

   END PROCESS xor_nor;

END behavorial;
```

Code 10: XOR gate

⇓ **q2-st.vhd** ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q2 IS PORT (
   X3, X2, X1, X0 : IN STD_LOGIC;
   Y3, Y2, Y1, Y0 : OUT STD_LOGIC
);
END q2;

ARCHITECTURE structural OF q2 IS

   COMPONENT xor_nor IS PORT (
      a, b : IN STD_LOGIC;
      o : OUT STD_LOGIC
      );
   END COMPONENT;

BEGIN
   C0 : xor_nor PORT MAP(a => X3, b => '0', o
    => Y3);
   C1 : xor_nor PORT MAP(a => X3, b => X2, o
    => Y2);
   C2 : xor_nor PORT MAP(a => X2, b => X1, o
    => Y1);
   C3 : xor_nor PORT MAP(a => X1, b => X0, o
    => Y0);

END structural;
```

Code 11: Structural model

⇓ **q2-tb.vhd** ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

ENTITY q2_tb IS
END q2_tb;

ARCHITECTURE behavioral OF q2_tb IS

   -- component declaaration for the Unit
    Under Test (UUT)
   COMPONENT q2
      PORT (
         X3 : IN STD_LOGIC;
         X2 : IN STD_LOGIC;
         X1 : IN STD_LOGIC;
         X0 : IN STD_LOGIC;
```

```
        Y3 : OUT STD_LOGIC;
        Y2 : OUT STD_LOGIC;
        Y1 : OUT STD_LOGIC;
        Y0 : OUT STD_LOGIC
    );
  END COMPONENT;

  SIGNAL input_vector : STD_LOGIC_VECTOR(3
   DOWNTO 0) := "0000";
  SIGNAL output_vector : STD_LOGIC_VECTOR(3
   DOWNTO 0) := "0000";

BEGIN
  --INSTANTIATE the unit under test
  uut : q2 PORT MAP(
    X3 => input_vector(3),
    X2 => input_vector(2),
    X1 => input_vector(1),
    X0 => input_vector(0),
    Y3 => output_vector(3),
    Y2 => output_vector(2),
    Y1 => output_vector(1),
    Y0 => output_vector(0)
  );

  --stimulus process
  stim_proc : PROCESS

  BEGIN
    FOR index IN 0 TO 15 LOOP
      input_vector <= STD_LOGIC_VECTOR(
   to_unsigned(index, 4));
      WAIT FOR 50 ns;
    END LOOP;
  END PROCESS;
END behavioral;
```

Code 12: Testbench for all cases

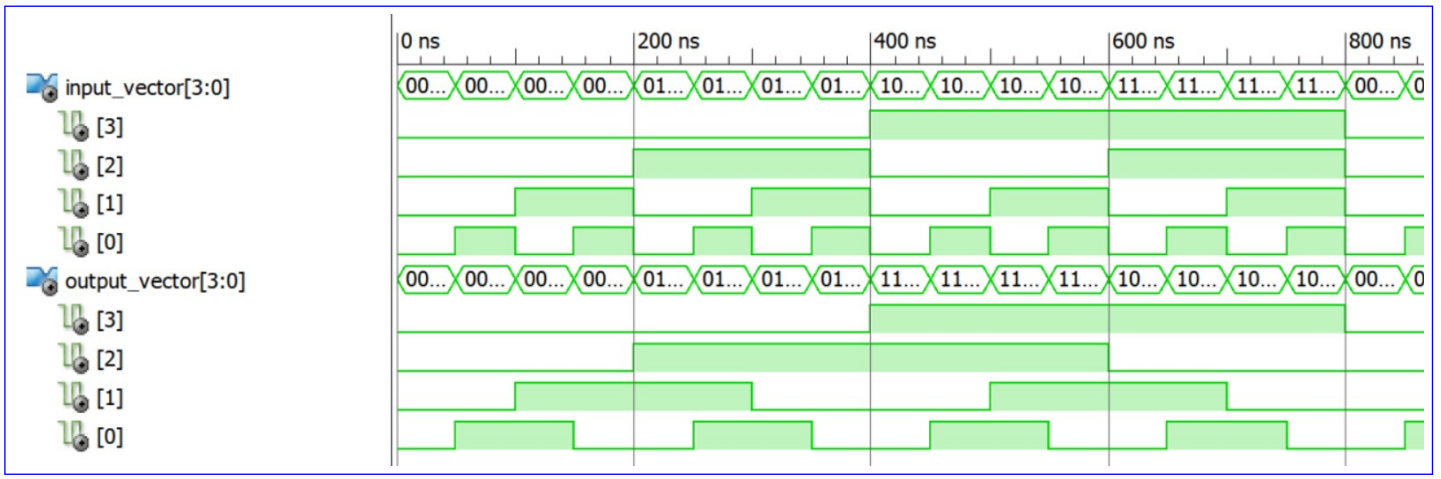


Figure 5: Simulation Waveform

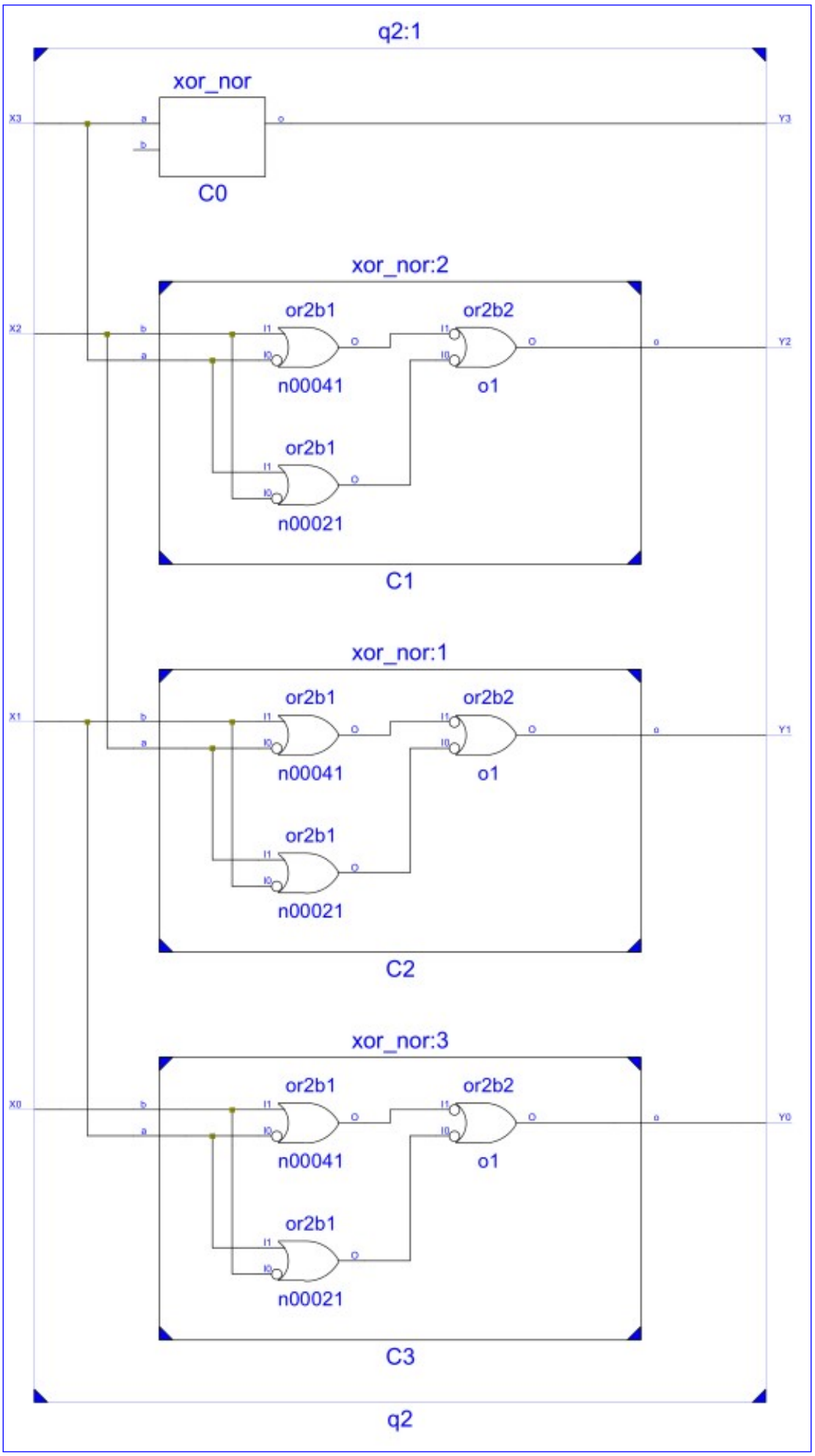


Figure 6: RTL Schematic

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:**
   - **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.**

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 Probelm 3

X2 X1 / **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 ⇓

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q3 IS PORT (
   X1, X2, X3 : IN STD_LOGIC;
   F : OUT STD_LOGIC
);
END q3;

ARCHITECTURE dataflow OF q3 IS
BEGIN
   F <= (X1 AND X2 AND NOT X3) OR (X3 AND (X1
    XOR X2));
END dataflow;
```

Code 13: Dataflow model

⇓ q3-be.vhd ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q3 IS PORT (
   X1, X2, X3 : IN STD_LOGIC;
   F : OUT STD_LOGIC
);
END q3;

ARCHITECTURE behavioral OF q3 IS
   SIGNAL A1, A2, A3 : STD_LOGIC;

BEGIN

   PROCESS (X1, X2, X3, A1, A2, A3)
   BEGIN
      A1 <= X1 AND X2 AND NOT X3;
      A2 <= X1 XOR X2;
      A3 <= A2 AND X3;
      F <= A1 OR A3;
   END PROCESS;
END behavioral;
```

Code 14: Behavorial model

⇓ q3-not.vhd ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q3_not IS
   PORT (
      a : IN STD_LOGIC;
      o : OUT STD_LOGIC);
END q3_not;

ARCHITECTURE behavorial OF q3_not IS

BEGIN
   q3_not : PROCESS (a)

   BEGIN
      o <= a NOR a;
   END PROCESS q3_not;

END behavorial;
```

Code 15: NOT gate using only NOR

⇓ q3-or.vhd ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q3_or IS PORT (
   a, b, c : IN STD_LOGIC;
   o : OUT STD_LOGIC
);
END q3_or;

ARCHITECTURE behavorial OF q3_or IS
   SIGNAL G1, G2, G3 : STD_LOGIC;

BEGIN
   q3_or : PROCESS (a, b, c, G1, G2, G3)

   BEGIN
      G1 <= a NOR b;
      G2 <= G1 NOR G1;
      G3 <= G2 NOR c;
      o <= G3 NOR G3;
   END PROCESS q3_or;

END behavorial;
```

Code 16: 3-input OR gate using only NOR

⇓ q3-and.vhd ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q3_and IS PORT (
   a, b, c: IN STD_LOGIC;
   o : OUT STD_LOGIC
);
END q3_and;

ARCHITECTURE behavorial OF q3_and IS
   SIGNAL F1, F2, F3, F4, F5 : STD_LOGIC;
BEGIN
   q3_and : PROCESS (a, b, c, F1, F2, F3, F4,
    F5)
   BEGIN
      F1 <= a NOR a;
      F2 <= b NOR b;
      F3 <= c NOR c;
      F4 <= F1 NOR F2;
      F5 <= F4 NOR F4;
      o <= F5 NOR F3;

   END PROCESS q3_and;
END behavorial;
```

Code 17: 3-input AND gate using only NOR

⇓ q3-st.vhd ⇓

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q3 IS PORT (
   X1, X2, X3 : IN STD_LOGIC;
   F : OUT STD_LOGIC
);
END q3;

ARCHITECTURE structural OF q3 IS
   SIGNAL A1, A2, A3, A4, A5, A6 : STD_LOGIC;

   COMPONENT q3_and IS PORT (
      a, b, c : IN STD_LOGIC;
      o : OUT STD_LOGIC
      );
   END COMPONENT;

   COMPONENT q3_or IS PORT (
      a, b, c : IN STD_LOGIC;
      o : OUT STD_LOGIC
      );
   END COMPONENT;

   COMPONENT q3_not IS PORT (
      a : IN STD_LOGIC;
      o : OUT STD_LOGIC
      );
   END COMPONENT;

BEGIN
   N1 : q3_not PORT MAP(a => X1, o => A1);
   N2 : q3_not PORT MAP(a => X2, o => A2);
   N3 : q3_not PORT MAP(a => X3, o => A3);
   N4 : q3_and PORT MAP(a => A1, b => X2, c =>
     X3, o =>  A4);
   N5 : q3_and PORT MAP(a => A2, b => X1, c =>
     X3, o =>  A5);
   N6 : q3_and PORT MAP(a => A3, b => X1, c =>
     X2, o =>  A6);
   N7 : q3_or PORT MAP(a => A4, b => A5, c =>
    A6, o => F);
END structural;
```

Code 18: Structural model

⇓ q3-tb.vhd ⇓

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

ENTITY q3_tb IS
END q3_tb;

ARCHITECTURE behavioral OF q3_tb IS
   COMPONENT q3
      PORT (
         X1, X2, X3 : IN STD_LOGIC;
         F : OUT STD_LOGIC
      );
   END COMPONENT;

   SIGNAL input : STD_LOGIC_VECTOR(2 DOWNTO 0)
     := "000";
   SIGNAL output : STD_LOGIC := '0';

BEGIN
   uut : q3 PORT MAP(
      X3 => input(2),
      X2 => input(1),
      X1 => input(0),
      F => output
   );

   stim_proc : PROCESS

   BEGIN
      FOR index IN 0 TO 7 LOOP
         input <= STD_LOGIC_VECTOR(TO_UNSIGNED
    (index, 3));
         WAIT FOR 50 ns;
      END LOOP;
   END PROCESS;

END behavioral;
```

Code 19: Testbench for all cases

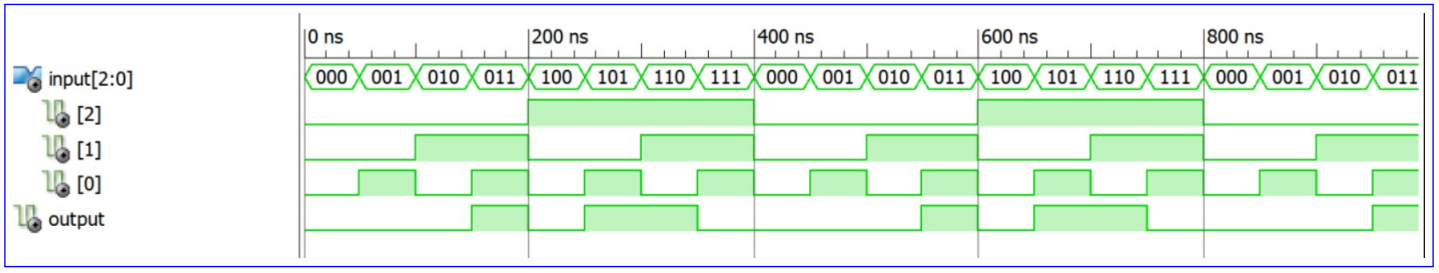


Figure 7: Simulation Waveform for Problem 3

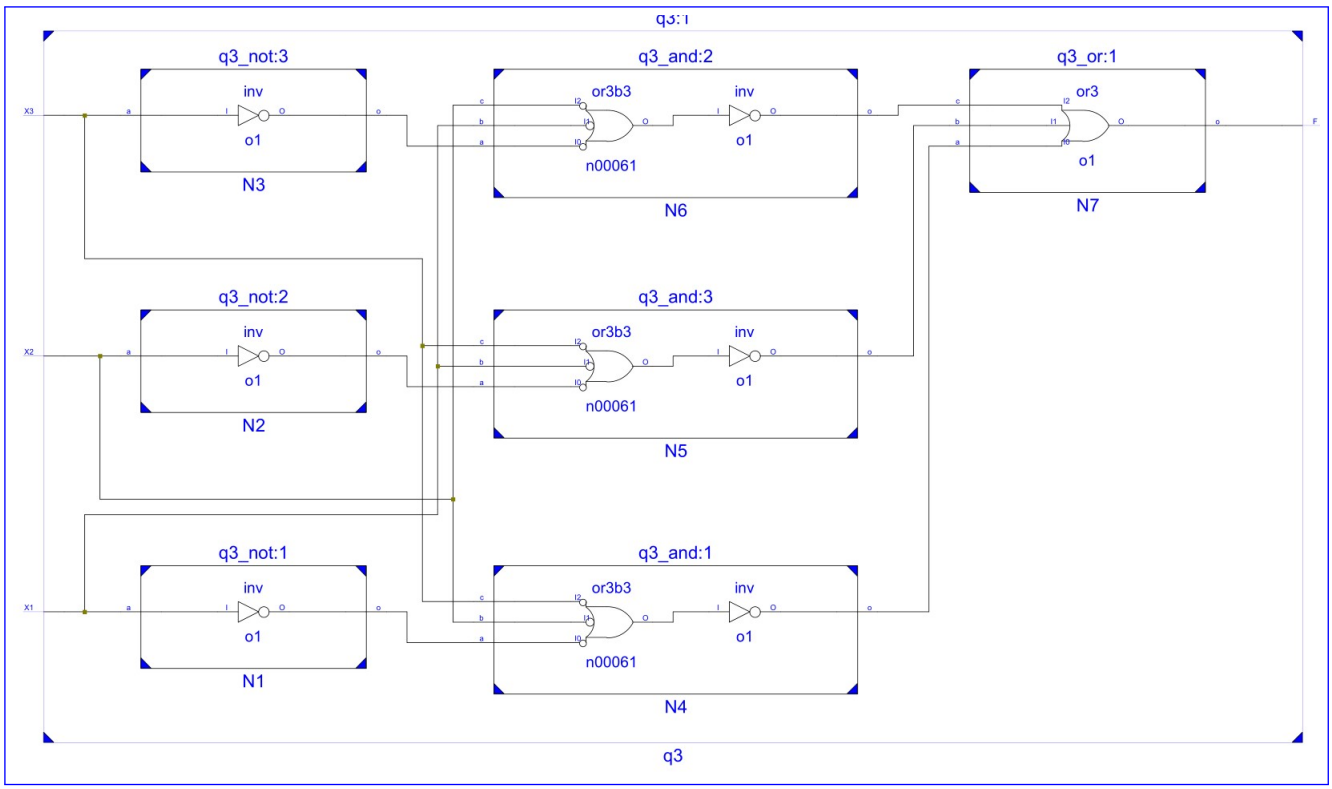


Figure 8: RTL Schematic for Problem 3

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:**
   - **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.**

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

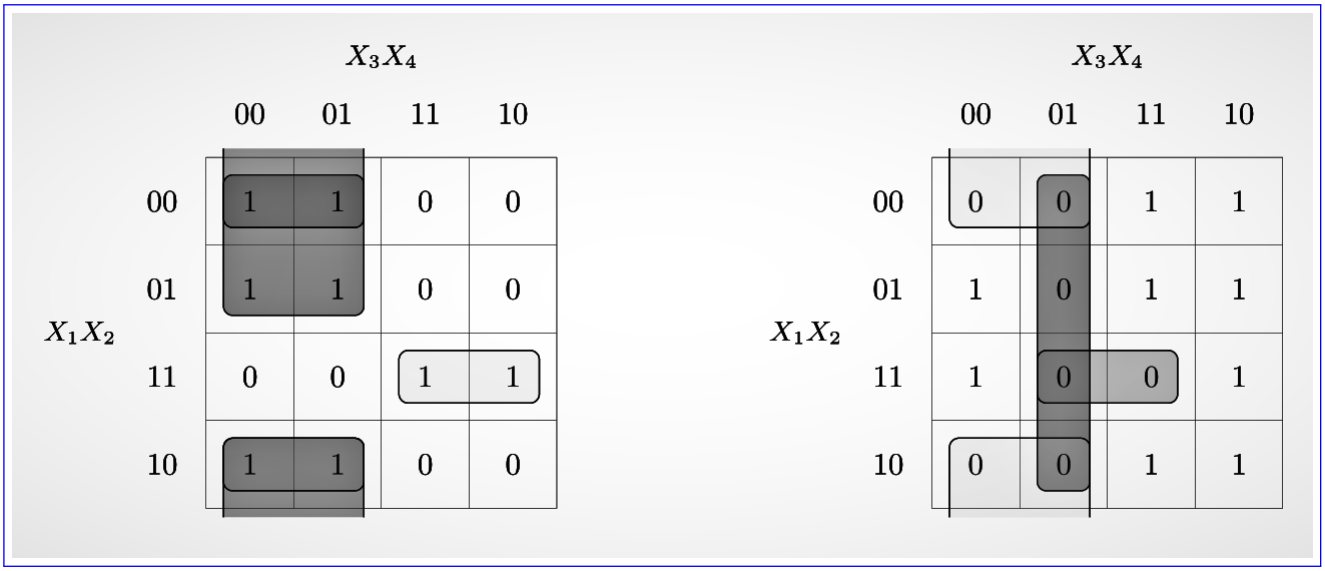


Figure 10: SOP and POS K-map reduction F1 and F2

$$F_1 = X_1'X_3' + X_1X_2X_3 + X_2'X_3'$$
$$F_2 = (X_3 + X_4)(X_1' + X_2' + X_4')(X_2 + X_3)$$

⇓ **q4sop-df.vhd** ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q4_sop IS PORT (
   X1, X2, X3, X4 : IN STD_LOGIC;
   Y : OUT STD_LOGIC
);
END q4_sop;

ARCHITECTURE dataflow OF q4_sop IS

BEGIN
   Y <= (NOT X1 AND NOT X3) OR (NOT X2 AND NOT
     X3) OR (X1 AND X2 AND X3);
END dataflow;
```

Code 20: Dataflow model SOP

⇓ q4sop-be.vhd ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q4_sop IS PORT (
   X1, X2, X3, X4 : IN STD_LOGIC;
   Y : OUT STD_LOGIC
);
END q4_sop;

ARCHITECTURE behavorial OF q4_sop IS
   SIGNAL Y1, Y2, Y3 : STD_LOGIC;

BEGIN

   PROCESS (X1, X2, X3, X4, Y1, Y2, Y3)

   BEGIN
      Y1 <= NOT X1 AND NOT X3;
      Y2 <= NOT X2 AND NOT X3;
      Y3 <= X1 AND X2 AND X3;
      Y <= Y1 OR Y2 OR Y3;

   END PROCESS;

END behavorial;
```

Code 21: Behavorial model SOP

⇓ q4-not.vhd ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q4_not IS
   PORT (
      a : IN STD_LOGIC;
      o : OUT STD_LOGIC);
END q4_not;

ARCHITECTURE behavorial OF q4_not IS

BEGIN
   q4_not : PROCESS (a)

   BEGIN
      o <= a NOR a;
   END PROCESS q4_not;

END behavorial;
```

Code 22: NOT gate implementation using only NOR

⇓ q4-or.vhd ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q4_or IS PORT (
   a, b, c : IN STD_LOGIC;
   o : OUT STD_LOGIC
);
END q4_or;

ARCHITECTURE behavorial OF q4_or IS
   SIGNAL G1, G2, G3 : STD_LOGIC;

BEGIN
   q4_or : PROCESS (a, b, c, G1, G2, G3)

   BEGIN
      G1 <= a NOR b;
      G2 <= G1 NOR G1;
      G3 <= G2 NOR c;
      o <= G3 NOR G3;
   END PROCESS q4_or;

END behavorial;
```

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

⇓ q4-and.vhd ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q4_and IS PORT (
   a, b, c : IN STD_LOGIC;
   o : OUT STD_LOGIC
);
END q4_and;

ARCHITECTURE behavorial OF q4_and IS
   SIGNAL F1, F2, F3, F4, F5 : STD_LOGIC;
BEGIN
   q4_and : PROCESS (a, b, c, F1, F2, F3, F4, F5)
   BEGIN
      F1 <= a NOR a;
      F2 <= b NOR b;
      F3 <= c NOR c;
      F4 <= F1 NOR F2;
      F5 <= F4 NOR F4;
      o <= F5 NOR F3;

   END PROCESS q4_and;
END behavorial;
```

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

⇓ q4sop-st.vhd ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q4_sop IS PORT (
   X1, X2, X3, X4 : IN STD_LOGIC;
   Y : OUT STD_LOGIC
);
END q4_sop;

ARCHITECTURE structural OF q4_sop IS
   SIGNAL A1, A2, A3, A4, A5, A6 : STD_LOGIC;

   COMPONENT q4_and IS PORT (
      a, b, c : IN STD_LOGIC;
      o : OUT STD_LOGIC
      );
   END COMPONENT;

   COMPONENT q4_or IS PORT (
      a, b, c : IN STD_LOGIC;
      o : OUT STD_LOGIC
      );
   END COMPONENT;

   COMPONENT q4_not IS PORT (
      a : IN STD_LOGIC;
      o : OUT STD_LOGIC
      );
   END COMPONENT;

BEGIN
   N1 : q4_not PORT MAP(a => X1, o => A1);
   N2 : q4_not PORT MAP(a => X2, o => A2);
   N3 : q4_not PORT MAP(a => X3, o => A3);
   N4 : q4_and PORT MAP(a => A1, b => A3, c => '1', o => A4);
   N5 : q4_and PORT MAP(a => A2, b => A3, c => '1', o => A5);
   N6 : q4_and PORT MAP(a => X1, b => X2, c => X3, o => A6);
   N7 : q4_or PORT MAP(a => A4, b => A5, c => A6, o => Y);
END structural;
```

Code 25: Structural model SOP

⇓ q4sop-tb.vhd ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

ENTITY q4_sop_tb IS
END q4_sop_tb;

ARCHITECTURE behavioral OF q4_sop_tb IS
   -- component declaaration for the Unit Under Test (UUT)
   COMPONENT q4_sop
      PORT (
         X1 : IN STD_LOGIC;
         X2 : IN STD_LOGIC;
         X3 : IN STD_LOGIC;
         X4 : IN STD_LOGIC;
         Y : OUT STD_LOGIC
      );
   END COMPONENT;

   SIGNAL input_vector : STD_LOGIC_VECTOR(3 DOWNTO 0) := "0000";
   SIGNAL output : STD_LOGIC := '0';

BEGIN
   --INSTANTIATE the unit under test
   uut : q4_sop PORT MAP(
      X1 => input_vector(3),
      X2 => input_vector(2),
      X3 => input_vector(1),
      X4 => input_vector(0),
      Y => output
   );

   --stimulus process
   stim_proc : PROCESS

   BEGIN
      FOR index IN 0 TO 15 LOOP
         input_vector <= STD_LOGIC_VECTOR(to_unsigned(index, 4));
         WAIT FOR 50 ns;
      END LOOP;
   END PROCESS;
END behavioral;
```

Code 26: Testbench for all posible cases SOP

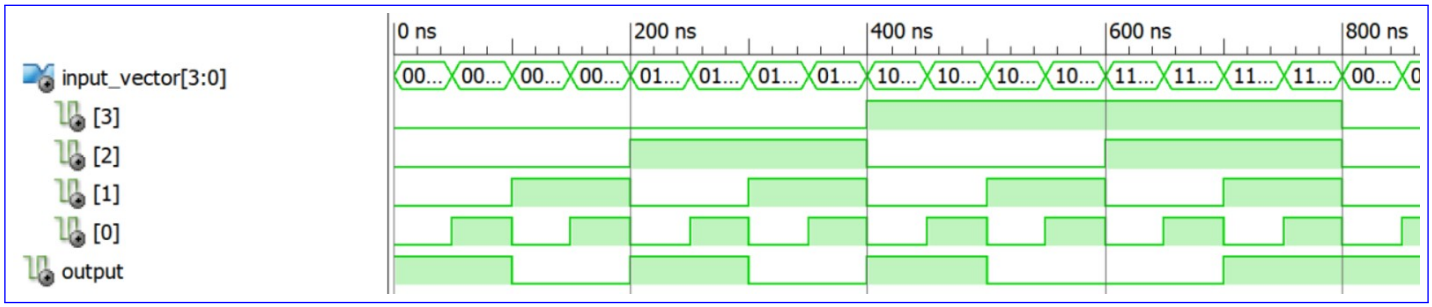


Figure 11: Simulation Waveform SOP

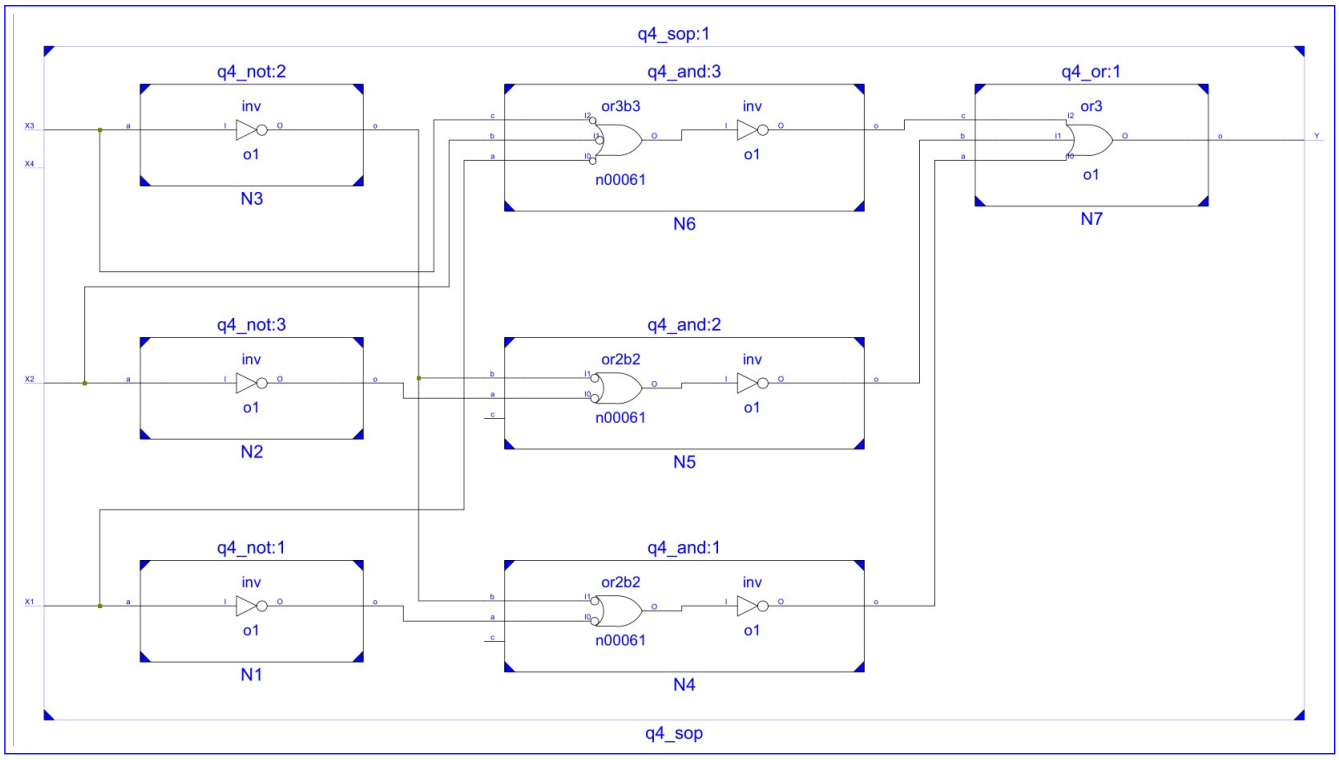


Figure 12: RTL Schematic SOP

⇓ **q4pos-df.vhd** ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q4_pos IS PORT (
   X1, X2, X3, X4 : IN STD_LOGIC;
   Y : OUT STD_LOGIC
);
END q4_pos;

ARCHITECTURE dataflow OF q4_pos IS

BEGIN
   Y <= (X3 OR NOT X4) AND (X2 OR X3) AND (NOT
     X1 OR NOT X2 OR NOT X4);
END dataflow;
```

Code 27: Dataflow model POS

⇓ **q4pos-be.vhd** ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q4_pos IS PORT (
   X1, X2, X3, X4 : IN STD_LOGIC;
   Y : OUT STD_LOGIC
);
END q4_pos;

ARCHITECTURE behavorial OF q4_pos IS
   SIGNAL Y1, Y2, Y3 : STD_LOGIC;

BEGIN

   PROCESS (X1, X2, X3, X4, Y1, Y2, Y3)

   BEGIN
      Y1 <= X3 OR NOT X4;
      Y2 <= X2 OR X3;
      Y3 <= NOT X1 OR NOT X2 OR NOT X4;
      Y <= Y1 AND Y2 AND Y3;

   END PROCESS;

END behavorial;
```

Code 28: Behavorial model POS

⇓ q4pos-st.vhd ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q4_pos IS PORT (
   X1, X2, X3, X4 : IN STD_LOGIC;
   Y : OUT STD_LOGIC
);
END q4_pos;

ARCHITECTURE structural OF q4_pos IS
   SIGNAL A1, A2, A3, A4, A5, A6 : STD_LOGIC;

   COMPONENT q4_and IS PORT (
      a, b, c : IN STD_LOGIC;
      o : OUT STD_LOGIC
      );
   END COMPONENT;

   COMPONENT q4_or IS PORT (
      a, b, c : IN STD_LOGIC;
      o : OUT STD_LOGIC
      );
   END COMPONENT;

   COMPONENT q4_not IS PORT (
      a : IN STD_LOGIC;
      o : OUT STD_LOGIC
      );
   END COMPONENT;

BEGIN
   N1 : q4_not PORT MAP(a => X1, o => A1);
   N2 : q4_not PORT MAP(a => X2, o => A2);
   N3 : q4_not PORT MAP(a => X4, o => A3);
   N4 : q4_or PORT MAP(a => A1, b => A2, c =>
    A3, o => A4);
   N5 : q4_or PORT MAP(a => X3, b => A3, c =>
    '0', o => A5);
   N6 : q4_or PORT MAP(a => X2, b => X3, c =>
    '0', o => A6);
   N7 : q4_and PORT MAP(a => A4, b => A5, c =>
     A6, o => Y);
END structural;
```

Code 29: Structural model POS

⇓ q4pos-tb.vhd ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

ENTITY q4_pos_tb IS
END q4_pos_tb;

ARCHITECTURE behavioral OF q4_pos_tb IS
   -- component declaaration for the Unit
    Under Test (UUT)
   COMPONENT q4_pos
      PORT (
         X1 : IN STD_LOGIC;
         X2 : IN STD_LOGIC;
         X3 : IN STD_LOGIC;
         X4 : IN STD_LOGIC;
         Y : OUT STD_LOGIC
      );
   END COMPONENT;

   SIGNAL input_vector : STD_LOGIC_VECTOR(3
    DOWNTO 0) := "0000";
   SIGNAL output : STD_LOGIC := '0';

BEGIN
   --INSTANTIATE the unit under test
   uut : q4_pos PORT MAP(
      X1 => input_vector(3),
      X2 => input_vector(2),
      X3 => input_vector(1),
      X4 => input_vector(0),
      Y => output
   );

   --stimulus process
   stim_proc : PROCESS

   BEGIN
      FOR index IN 0 TO 15 LOOP
         input_vector <= STD_LOGIC_VECTOR(
    to_unsigned(index, 4));
         WAIT FOR 50 ns;
      END LOOP;
   END PROCESS;
END behavioral;
```

Code 30: Testbench for all posible cases POS

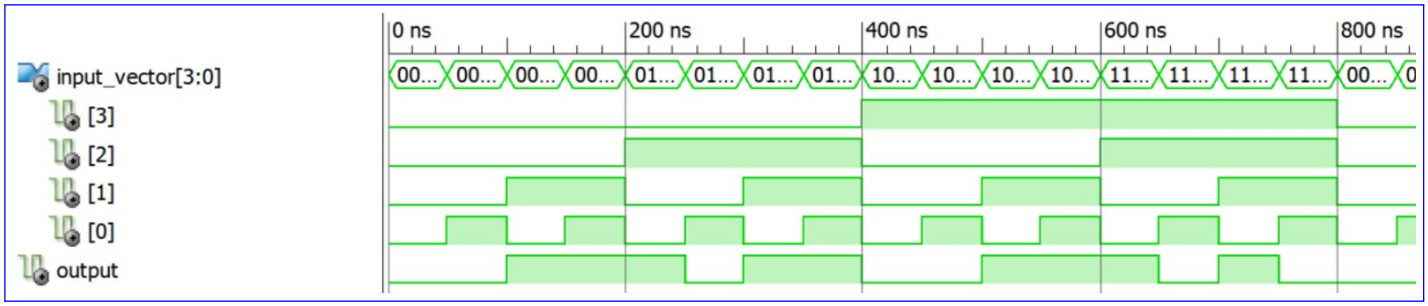


Figure 13: Simulation Waveform POS

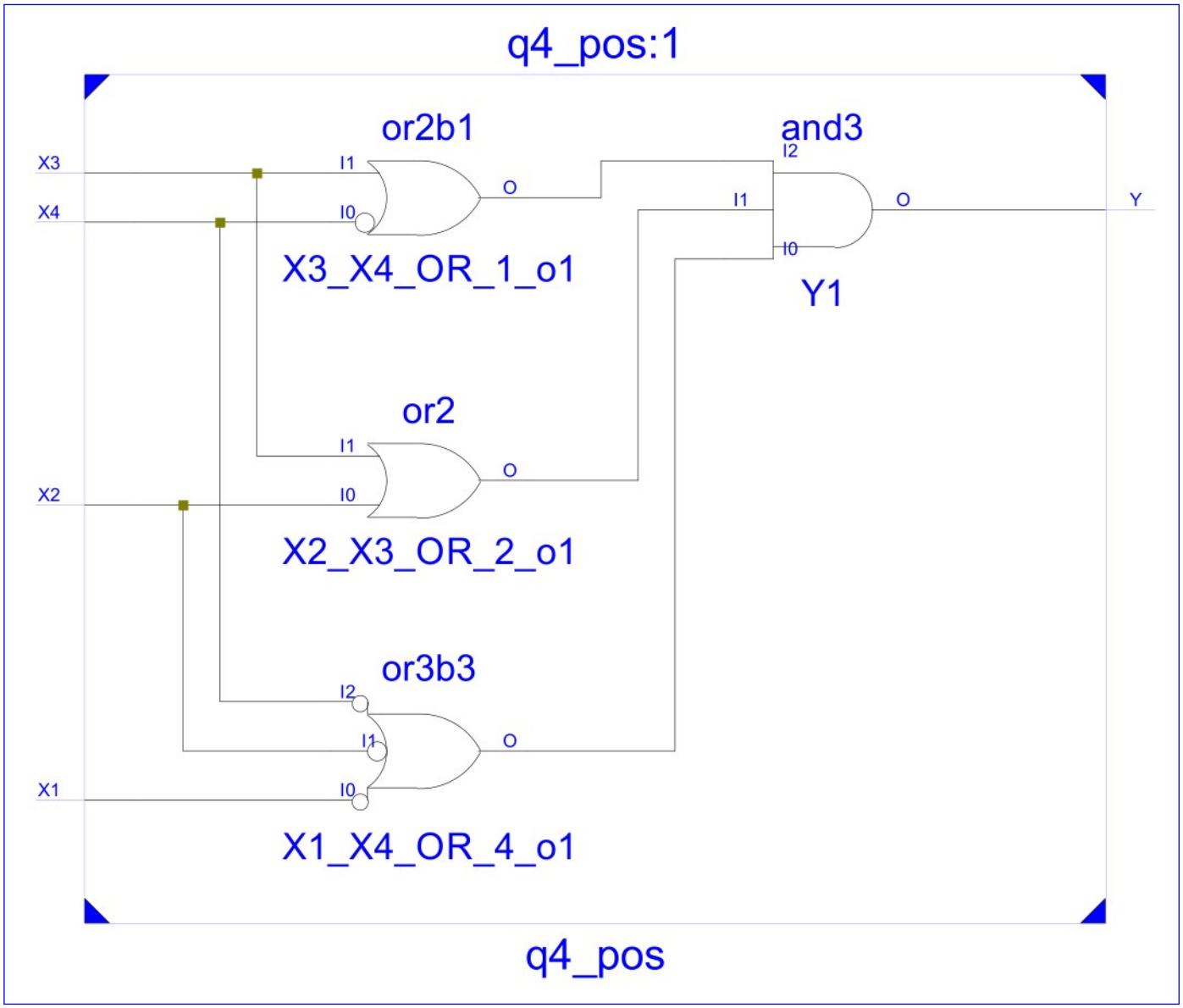


Figure 14: RTL Schematic POS

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 ⇓

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q5_mux2_1 IS PORT (
   S, X1, X2 : IN STD_LOGIC;
   Y : OUT STD_LOGIC
);
END q5_mux2_1;
```

```

ARCHITECTURE dataflow OF q5_mux2_1 IS
BEGIN
   Y <= '1' WHEN ((NOT S AND X1) OR (S AND X2)
    ) = '1' ELSE
      '0';
END dataflow;
```

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

⇓ q5-if-then-else.vhd ⇓

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q5_mux2_1 IS PORT (
   S, X1, X2 : IN STD_LOGIC;
   Y : OUT STD_LOGIC
);
END q5_mux2_1;

ARCHITECTURE behavioral OF q5_mux2_1 IS
BEGIN
```

```

   PROCESS (S, X1, X2)
   BEGIN
      IF ((NOT S AND X1) OR (S AND X2)) = '1'
    THEN
         Y <= '1';
      ELSE
         Y <= '0';
      END IF;
   END PROCESS;
END behavioral;
```

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

⇓ q5-tb.vhd ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

ENTITY q5_mux2_1_tb IS
END q5_mux2_1_tb;

ARCHITECTURE behavioral OF q5_mux2_1_tb IS
   COMPONENT q5_mux2_1
      PORT (
         S : IN STD_LOGIC;
         X1 : IN STD_LOGIC;
         X2 : IN STD_LOGIC;
         Y : OUT STD_LOGIC
      );
   END COMPONENT;

   SIGNAL input : STD_LOGIC_VECTOR(2 DOWNTO 0)
     := "000";
   SIGNAL output : STD_LOGIC;

BEGIN
   uut : q5_mux2_1 PORT MAP(
      S => input(2),
      X2 => input(1),
      X1 => input(0),
      Y => output
   );

   stim_proc : PROCESS

   BEGIN
      FOR index IN 0 TO 7 LOOP
         input <= STD_LOGIC_VECTOR(TO_UNSIGNED
    (index, 3));
         WAIT FOR 50 ns;
      END LOOP;
   END PROCESS;

END behavioral;
```

Code 33: Testbench for all possible cases

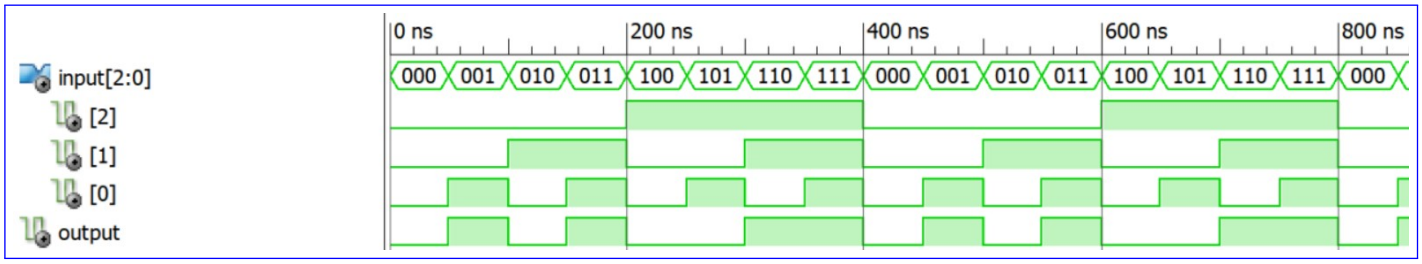


Figure 16: Simulation Waveform

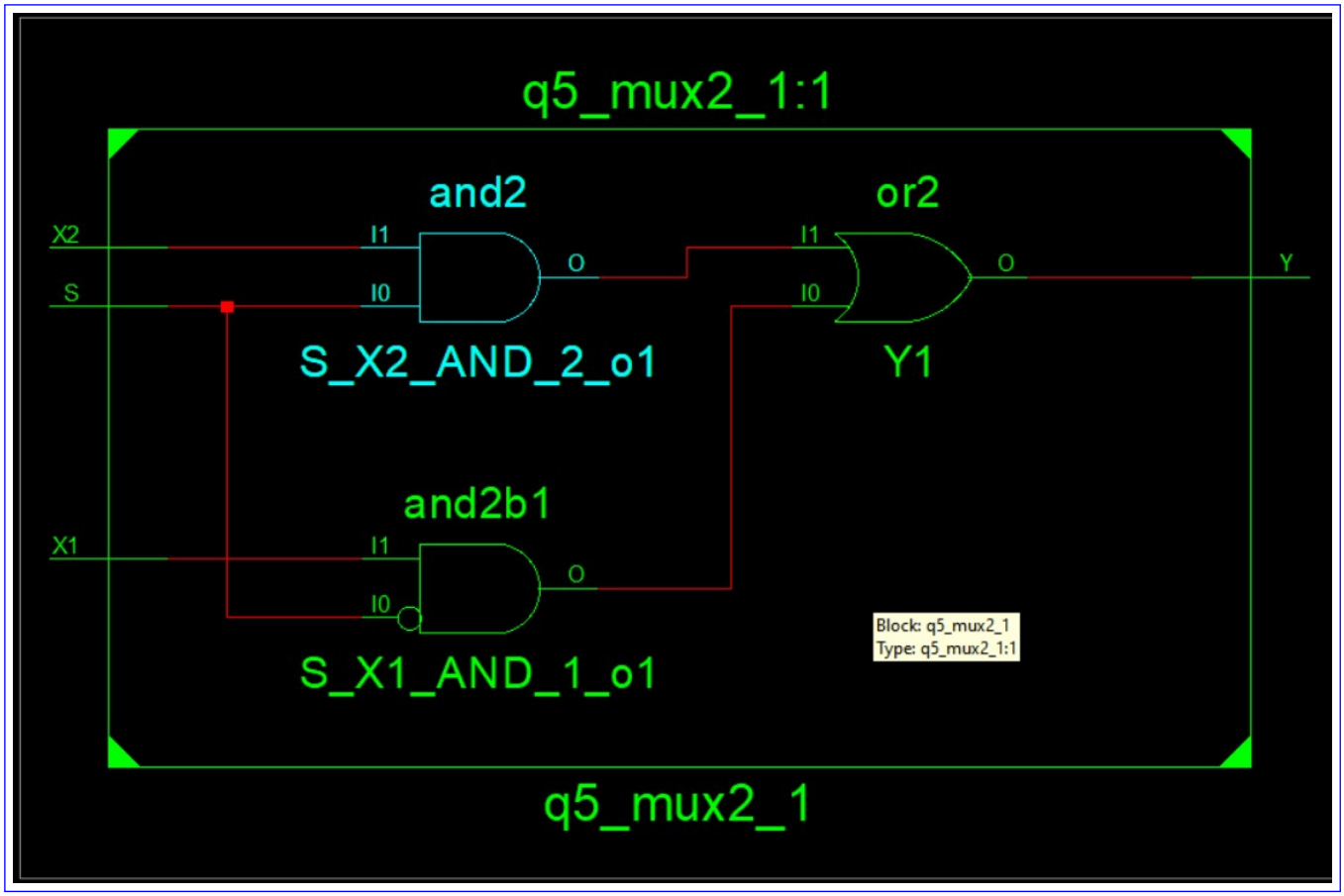


Figure 17: RTL Schematic

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.**

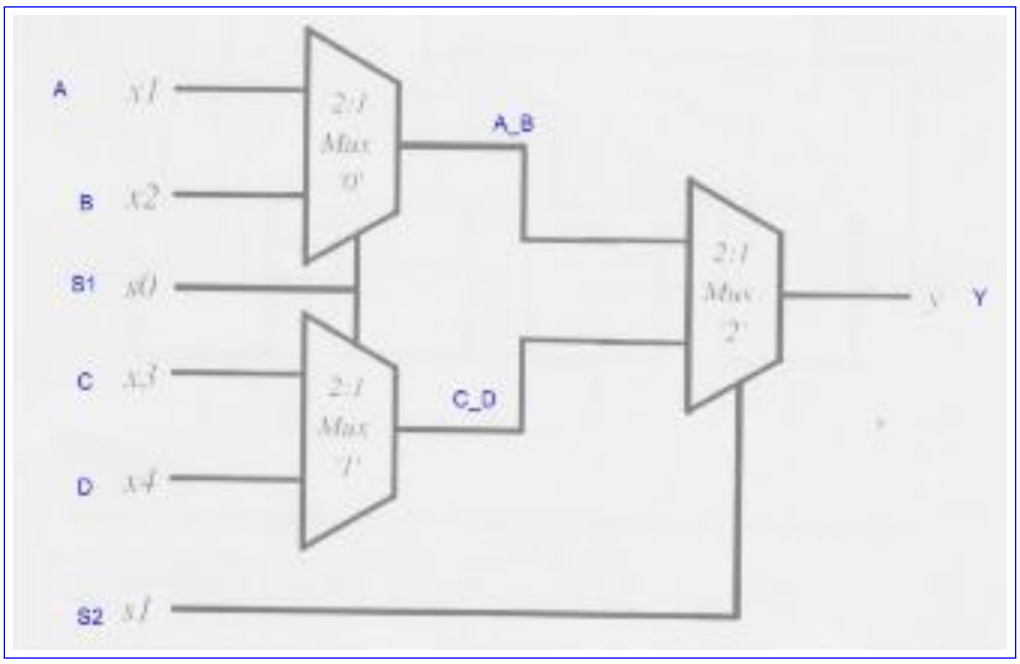


Figure 18: MUX using three 2:1 MUX as basic building blocks

⇓ q5-when-else.vhd ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q5_mux2_1 IS PORT (
   S, X1, X2 : IN STD_LOGIC;
   Y : OUT STD_LOGIC
);
END q5_mux2_1;

ARCHITECTURE dataflow OF q5_mux2_1 IS
BEGIN
   Y <= '1' WHEN ((NOT S AND X1) OR (S AND X2)
    ) = '1' ELSE
      '0';
END dataflow;
```

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

⇓ q6-st.vhd ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q6_mux4_1 IS PORT (
   X1, X2, X3, X4, S0, S1 : IN STD_LOGIC;
   Y : OUT STD_LOGIC
);
END q6_mux4_1;

ARCHITECTURE structural OF q6_mux4_1 IS
   SIGNAL F1, F2 : STD_LOGIC;

   COMPONENT q5_mux2_1 IS PORT (
      X1, X2, S : IN STD_LOGIC;
      Y : OUT STD_LOGIC
      );
   END COMPONENT;

BEGIN
   M0 : q5_mux2_1 PORT MAP(X1 => X1, X2 => X2,
     S => S0, Y => F1);
   M1 : q5_mux2_1 PORT MAP(X1 => X3, X2 => X4,
     S => S0, Y => F2);
   M2 : q5_mux2_1 PORT MAP(X1 => F1, X2 => F2,
     S => S1, Y => Y);
END structural;
```

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

⇓ q6-tb.vhd ⇓

```vhdl
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

ENTITY q6_mux4_1_tb IS
END q6_mux4_1_tb;

ARCHITECTURE behavioral OF q6_mux4_1_tb IS
   -- component declaaration for the Unit
    Under Test (UUT)
   COMPONENT q6_mux4_1
      PORT (
         X1, X2, X3, X4, S0, S1 : IN STD_LOGIC
    ;
         Y : OUT STD_LOGIC
      );
   END COMPONENT;

   SIGNAL select_vector : STD_LOGIC_VECTOR(1
    DOWNTO 0) := "00";
   SIGNAL input_vector : STD_LOGIC_VECTOR(3
    DOWNTO 0) := "0000";
   SIGNAL output : STD_LOGIC := '0';

BEGIN
   --INSTANTIATE the unit under test
   uut : q6_mux4_1 PORT MAP(
      S0 => select_vector(0),
      S1 => select_vector(1),
      X1 => input_vector(3),
      X2 => input_vector(2),
      X3 => input_vector(1),
      X4 => input_vector(0),
      Y => output
   );

   --stimulus process
   stim_proc : PROCESS

   BEGIN
      FOR selector IN 0 TO 3 LOOP
         select_vector <= STD_LOGIC_VECTOR(
    to_unsigned(selector, 2));
         FOR index IN 0 TO 15 LOOP
         input_vector <= STD_LOGIC_VECTOR(
    to_unsigned(index, 4));
            WAIT FOR 40 ns;
         END LOOP;
      END LOOP;
   END PROCESS;
END behavioral;
```

Code 36: Testbench for all possible cases

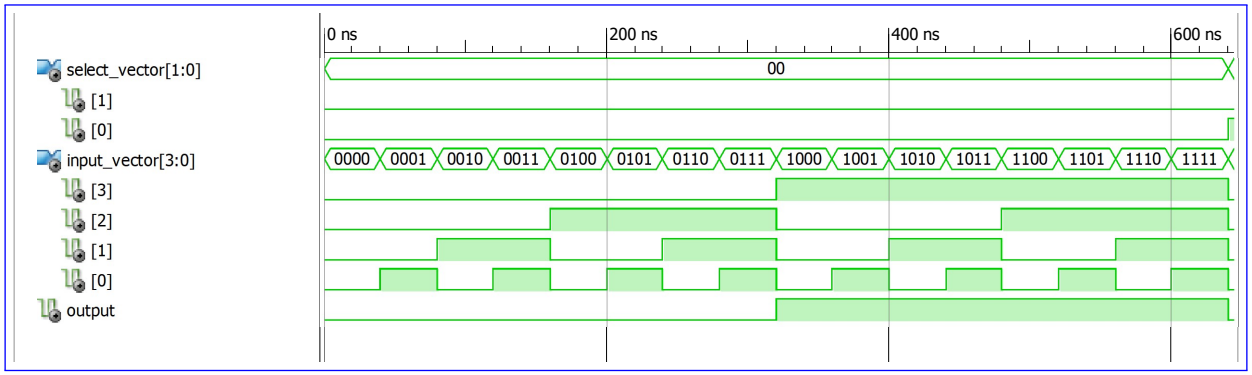


Figure 19: Simulation Waveform 0 ns to 700 ns

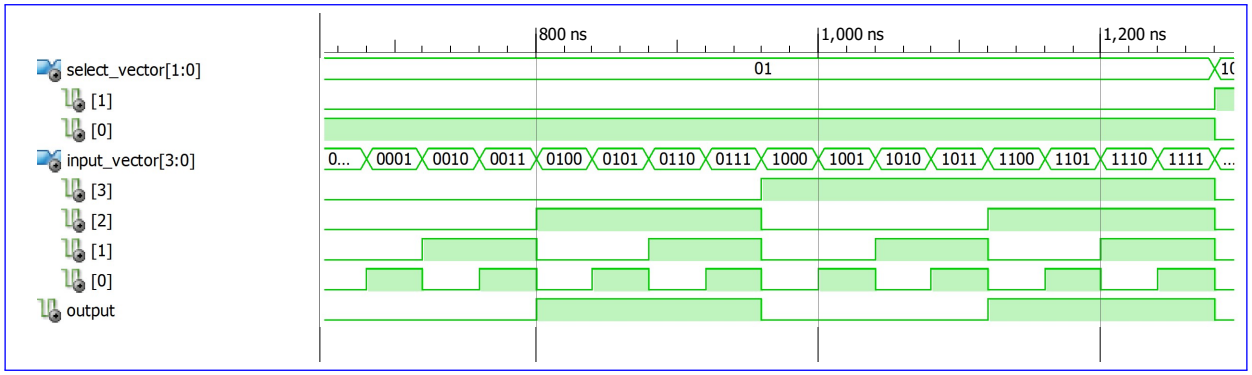


Figure 20: Simulation Waveform 700 ns to 1300 ns

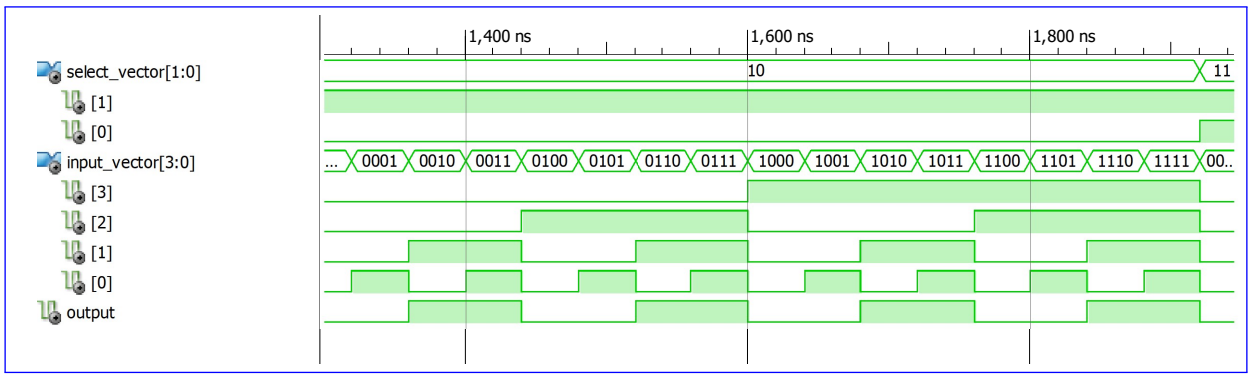


Figure 21: Simulation Waveform 1300 ns to 1900 ns

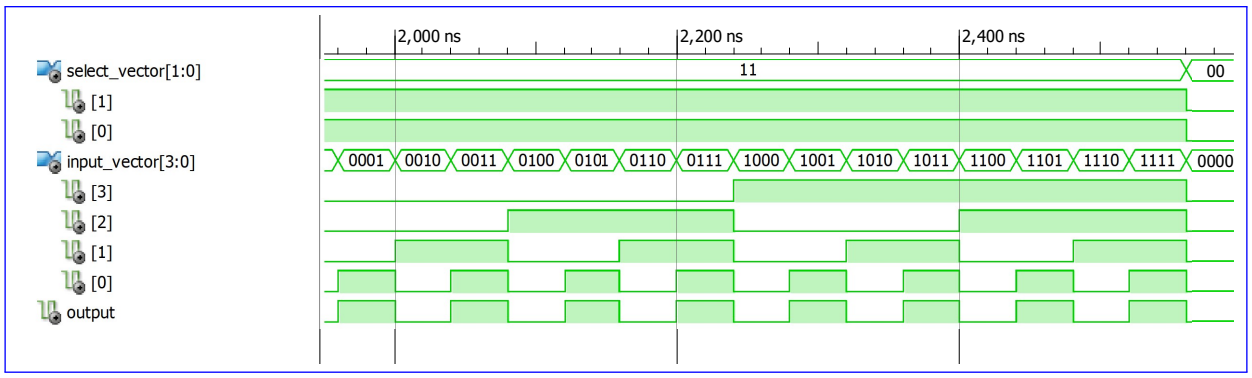


Figure 22: Simulation Waveform 1900 ns to 2600 ns

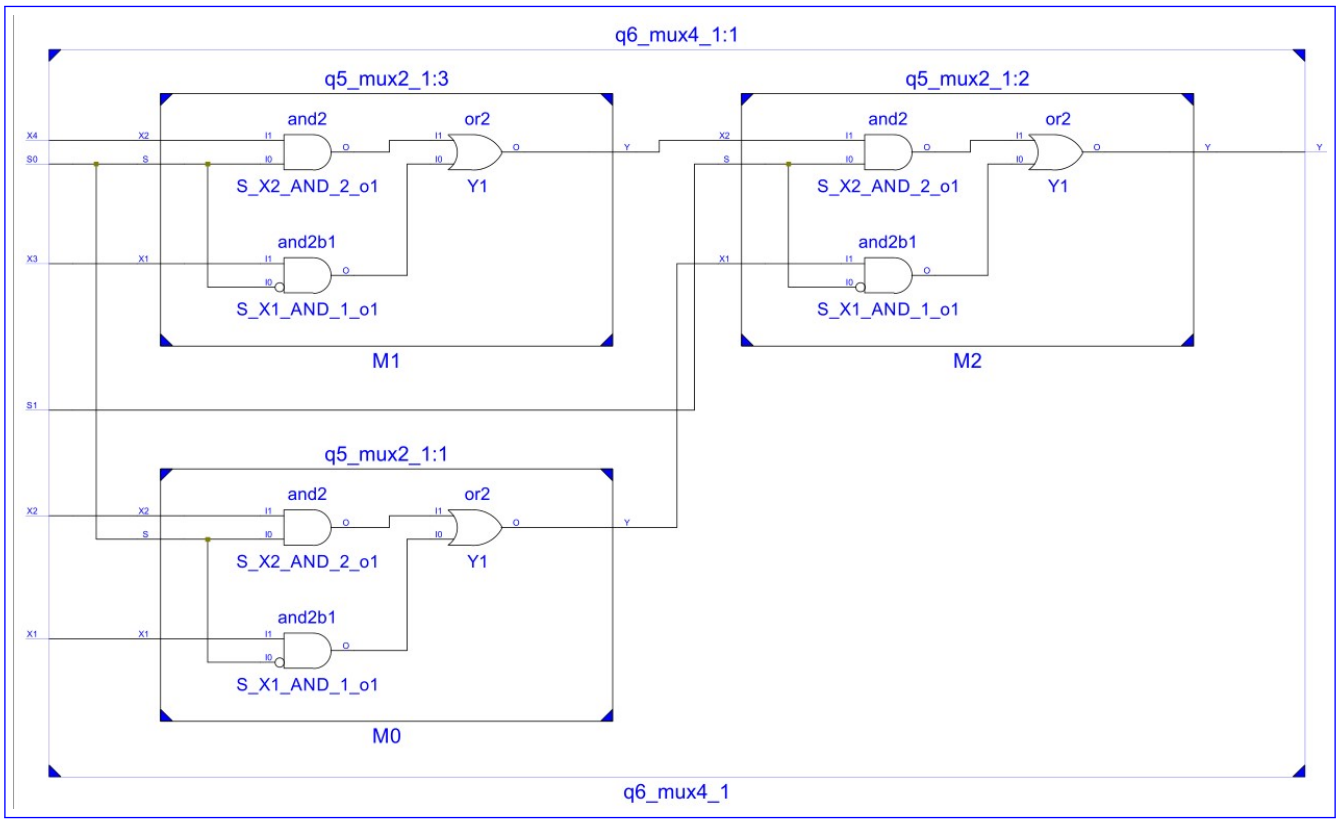


Figure 23: RTL Schematic

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 ⇓

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q7_xor IS PORT (
   i1, i2 : IN STD_LOGIC;
   o1 : OUT STD_LOGIC
);
END q7_xor;

ARCHITECTURE dataflow OF q7_xor IS
BEGIN
   o1 <= i1 XOR i2;
END dataflow;
```

Code 37: XOR gate implementation

⇓ full-adder.vhd ⇓

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY full_adder IS PORT (
   i1, i2, cin : IN STD_LOGIC;
   sum, cout : OUT STD_LOGIC
);
END full_adder;

ARCHITECTURE dataflow OF full_adder IS
BEGIN
   sum <= i1 XOR i2 XOR cin;
   cout <= (i1 AND i2) OR ((i1 XOR i2) AND cin
    );
END dataflow;
```

Code 38: Full adder implementation

⇓ q7-st.vhd ⇓

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;

ENTITY q7_add_sub IS PORT (
   X3, X2, X1, X0, Y3, Y2, Y1, Y0, A_S : IN
    STD_LOGIC;
   S4, S3, S2, S1, S0 : OUT STD_LOGIC
);
END q7_add_sub;

ARCHITECTURE structural OF q7_add_sub IS
   SIGNAL F0, F1, F2, F3, C1, C2, C3 :
    STD_LOGIC;

   COMPONENT q7_xor IS PORT (
      i1, i2 : IN STD_LOGIC;
      o1 : OUT STD_LOGIC
      );
   END COMPONENT;

   COMPONENT full_adder IS PORT (
      i1, i2, cin : IN STD_LOGIC;
      sum, cout : OUT STD_LOGIC
      );
   END COMPONENT;

BEGIN
   A0 : q7_xor PORT MAP(i1 => A_S, i2 => Y0,
    o1 => F0);
   A1 : full_adder PORT MAP(i1 => X0, i2 => F0
    , cin => A_S, sum => S0, cout => C1);

   A2 : q7_xor PORT MAP(i1 => A_S, i2 => Y1,
    o1 => F1);
```

```
    A3 : full_adder PORT MAP(i1 => X1, i2 => F1
     , cin => C1, sum => S1, cout => C2);

    A4 : q7_xor PORT MAP(i1 => A_S, i2 => Y2,
     o1 => F2);
    A5 : full_adder PORT MAP(i1 => X2, i2 => F2
     , cin => C2, sum => S2, cout => C3);

    A6 : q7_xor PORT MAP(i1 => A_S, i2 => Y3,
     o1 => F3);
    A7 : full_adder PORT MAP(i1 => X3, i2 => F3
     , cin => C3, sum => S3, cout => S4);
END structural;
```

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

⇓ q7-tb.vhd ⇓

```
LIBRARY IEEE;
USE IEEE.STD_LOGIC_1164.ALL;
USE IEEE.NUMERIC_STD.ALL;

ENTITY q7_add_sub_tb IS
END q7_add_sub_tb;

ARCHITECTURE behavioral OF q7_add_sub_tb IS
    -- component declaaration for the Unit
     Under Test (UUT)
    COMPONENT q7_add_sub
       PORT (
          X3, X2, X1, X0, Y3, Y2, Y1, Y0, A_S :
      IN STD_LOGIC;
          S4, S3, S2, S1, S0 : OUT STD_LOGIC
       );
    END COMPONENT;

    SIGNAL addsub : STD_LOGIC := '1';
    --Change addsub to '0' for 4-bit full-adder
    SIGNAL input_vector1 : STD_LOGIC_VECTOR(3
     DOWNTO 0) := "0000";
    SIGNAL input_vector2 : STD_LOGIC_VECTOR(3
     DOWNTO 0) := "0000";
    SIGNAL output_vector : STD_LOGIC_VECTOR(4
     DOWNTO 0) := "00000";

BEGIN
    --INSTANTIATE the unit under test
    uut : q7_add_sub PORT MAP(
       A_S => addsub,
       X3 => input_vector1(3),
       X2 => input_vector1(2),
       X1 => input_vector1(1),
       X0 => input_vector1(0),

       Y3 => input_vector2(3),
       Y2 => input_vector2(2),
       Y1 => input_vector2(1),
       Y0 => input_vector2(0),

       S4 => output_vector(4),
       S3 => output_vector(3),
       S2 => output_vector(2),
       S1 => output_vector(1),
       S0 => output_vector(0)
    );

    --stimulus process
    stim_proc : PROCESS

    BEGIN
       FOR index1 IN 0 TO 15 LOOP
          input_vector1 <= STD_LOGIC_VECTOR(
     to_unsigned(index1, 4));
          FOR index2 IN 0 TO 15 LOOP
          input_vector2 <= STD_LOGIC_VECTOR(
     to_unsigned(index2, 4));
             WAIT FOR 50 ns;
          END LOOP;
       END LOOP;
    END PROCESS;
END behavioral;
```

Code 40: Testbench for all possible cases

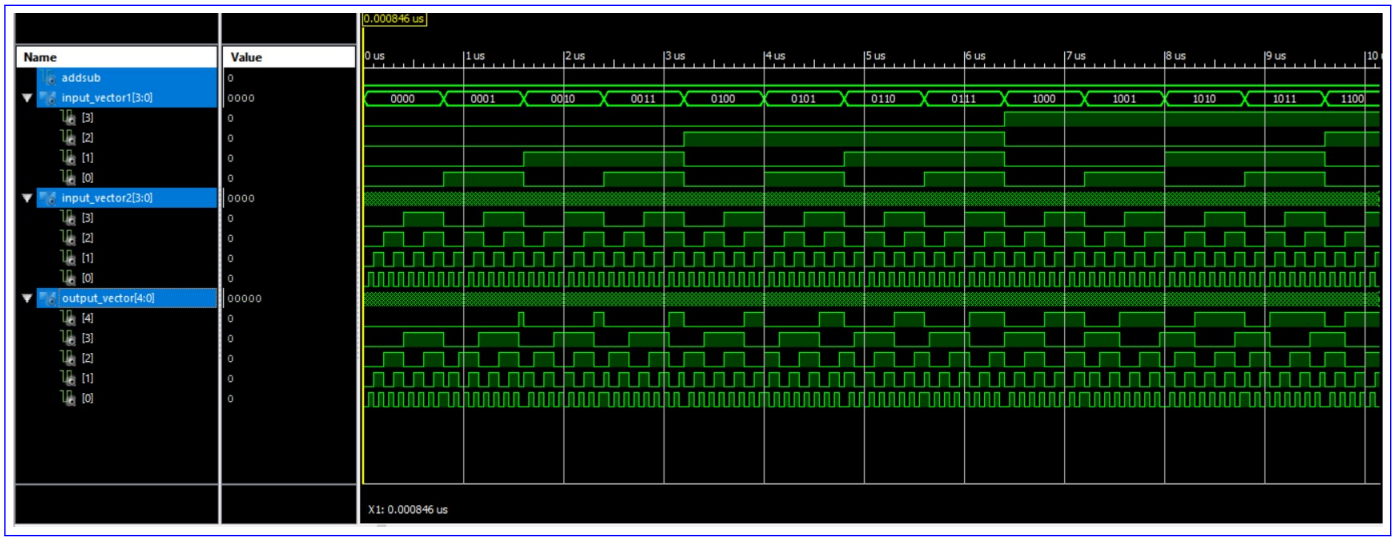


Figure 24: Simulation Waveform 4 bit Adder

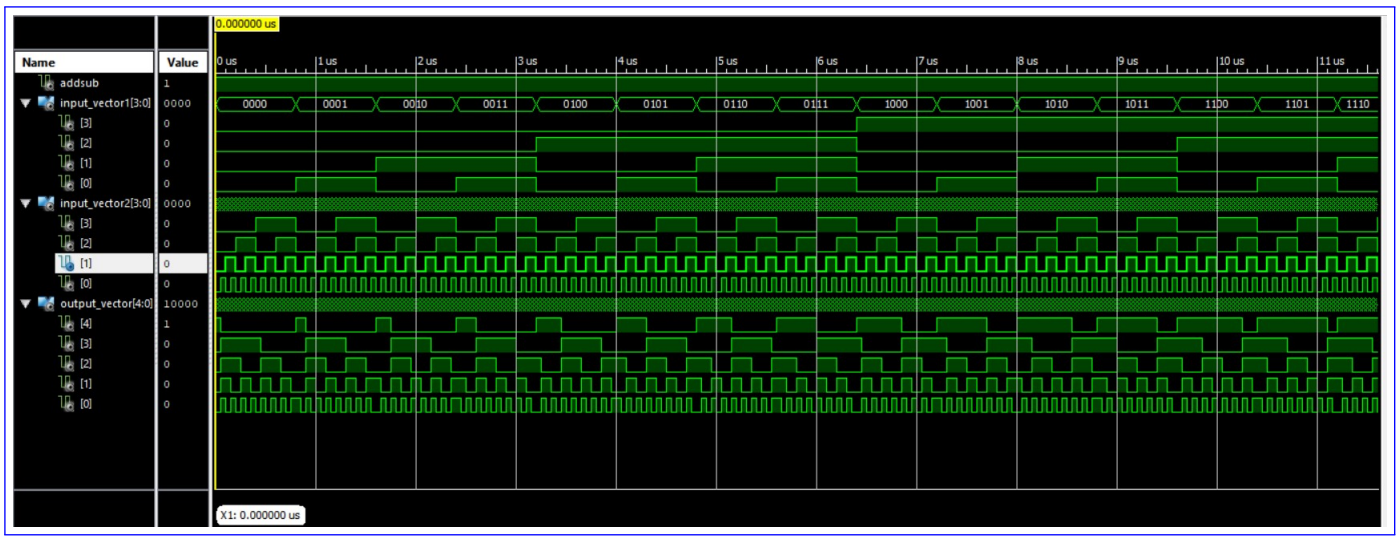


Figure 25: Simulation Waveform for 4 bit Subtractor

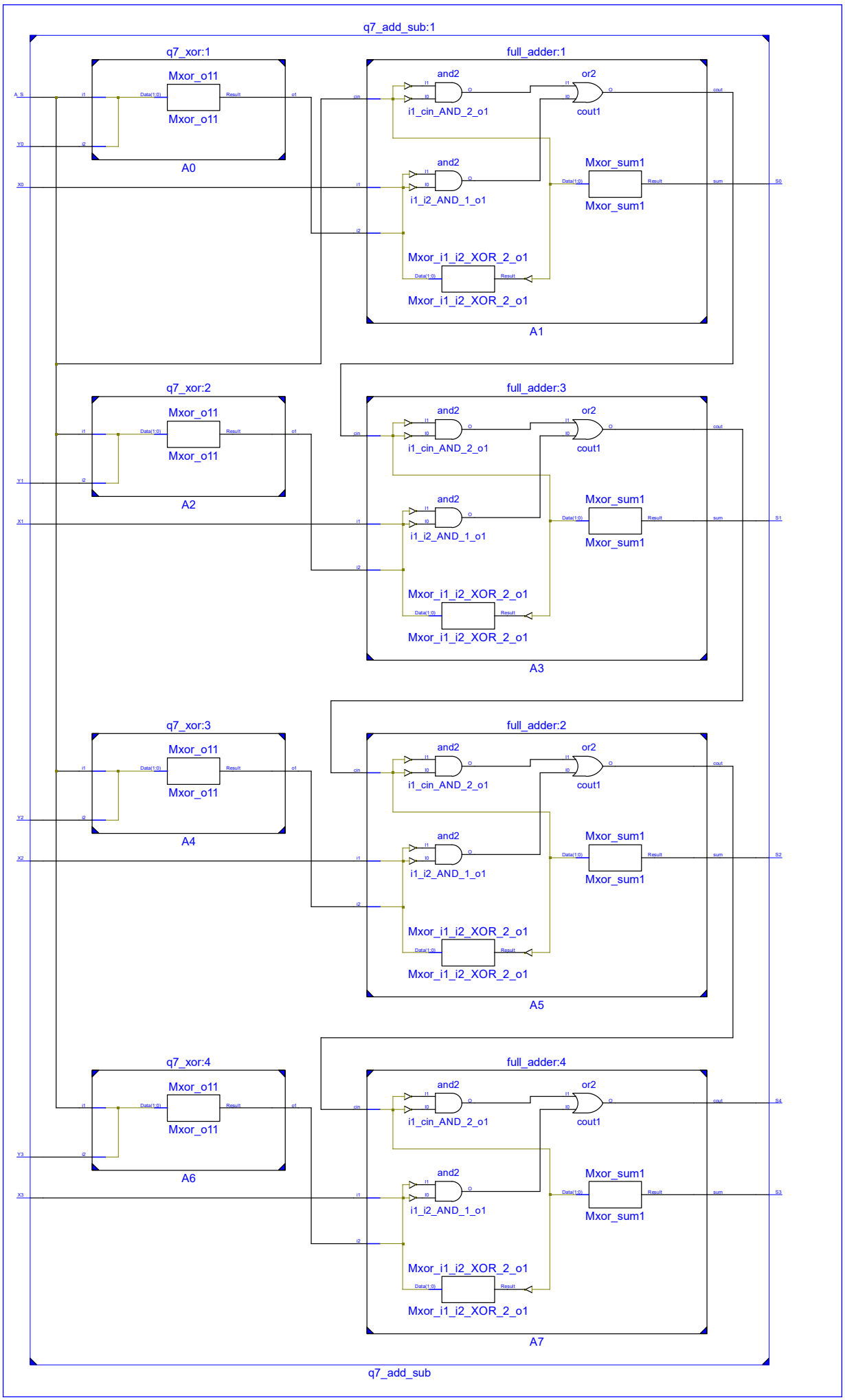


Figure 26: RTL Schematic

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.