Sequential Logic Design using VHDL

Lab Exercises on December 4, 2020

Department of Electronics and Computer Engineering
Pulchowk Campus, Lalitpur

Ashlesh Pandey

PUL074BEX007

Contents

List of Figures

Listings

Problem 1

Problem 2

Problem 3

Problem 4

Problem 5

Problem 6

Problem 7

1 Introduction

1.1 Sequential Logic Circuits

Unlike combinational logic circuits that change state depending upon the actual signals being applied to their inputs at that time, sequential logic circuits have some form of inherent "memory" built in. This means that sequential logic circuits are able to take into account their previous input state as well as those actually present. Thus, the output state of a sequential logic circuit is a function of the three states, viz. the "present input", the "past input" and/or the "past output".

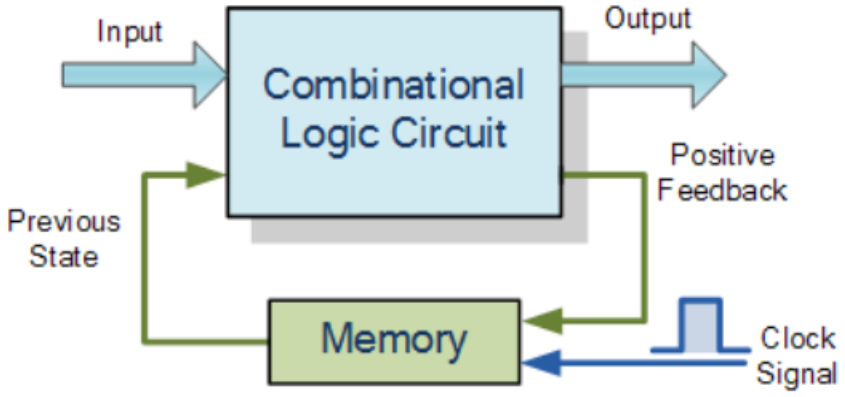


Figure 1: Sequential Logic Circuit

1.1.1 Synchronous Sequential Circuits

Synchronous sequential circuits use pulsed or level inputs and a clock input to drive the circuit (with restrictions on pulse width and circuit propagation).

1.1.2 Asynchronous Sequential Circuits

Asynchronous sequential circuits do not use a clock signal as synchronous circuits do, instead the circuit is driven by the pulses of the inputs.

1.2 VHDL Basics

VHDL stands for Very High-Speed Integration Circuit HDL (Hardware Description Language). It is an IEEE (Institute of Electrical and Electronics Engineers) standard hardware description language that is used to describe and simulate the behavior of complex digital circuits. VHDL also includes design management features, and features that allow precise modeling of events that occur over time.

2 Objectives

The primary objectives of this lab experiment is to understand programming concepts in VHDL for a Field Programmable Gate Array(FPGA). VHDL coding concepts for a FPGA will enable us to write codes capable of:

- Implementing sequential circuits.
- Implementing test benches to verify the working of sequential circuits.

3 Lab Experiment Environment

The lab experiments were performed virtually via simulation softwares. The basic usages of these tools will allow us to write, simulate and synthesize sequential circuts.

3.1 Xilinx ISE (Integrated Synthesis Environment) Design Suite

Xilinx ISE (Integrated Synthesis Environment) is a discontinued software tool from Xilinx for synthesis and analysis of HDL designs. It enables the developer to synthesize their designs, perform timing analysis, examine RTL diagrams, simulate a design's reaction to different stimuli, and configure the target device with the programmer. VHDL test benches are simulated on the ISE Simulator that provides a full-featured HDL simulator.

4 Lab Problems

Problem 1
Write a VHDL code to implement a JK flip-flop using a D flip-flop and the characteristic equation given in equation 1. The D flip-flop needs to be implemented using a structural architecture style using the circuit shown in figure below. The JK ip-op must be constructed using a component declaration for a D flip-flop. Write a VHDL test bench to verify the operation of the JK ip-op and provide waveforms.

The JK flip flop takes two input namely J and K that are used for setting, reseting, toggling or retaining states on the flip flop. For a D flip flop to be used as a JK flip flop, the following equation must be implemented,

$$D = JQ' + K'Q$$

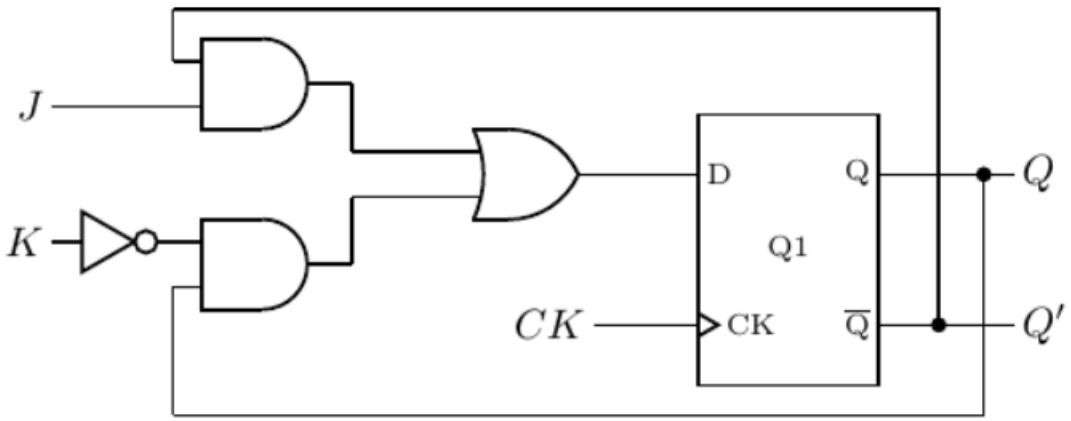


Figure 2: JK flip flop using D flip flop

combinational.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY combinational_1 IS PORT (
5          i1, i2, a3, a4 : IN STD_LOGIC;
6          o1 : OUT STD_LOGIC
7  );
8  END combinational_1;
9
10 ARCHITECTURE dataflow OF combinational_1 IS
11 BEGIN
12         o1 <= ((i1 AND a4) OR ((NOT i2) AND a3))
                  ;
13 END dataflow;
```

Listing 1: Combination circuit implementation: Dataflow model

nand_2.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY nand_2 IS
5          PORT (
6                  a, b : IN STD_LOGIC;
7                  o : OUT STD_LOGIC);
8  END nand_2;
9
10 ARCHITECTURE DATAFLOW OF nand_2 IS
11 BEGIN
12         o <= a NAND b;
13 END DATAFLOW;
```

Listing 2: 2-input NAND gate implementation: Dataflow model

nand_3.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY nand_3 IS
5          PORT (
6                  a, b, c : IN STD_LOGIC;
7                  o : OUT STD_LOGIC);
8  END nand_3;
9
10 ARCHITECTURE dataflow OF nand_3 IS
11 BEGIN
12         o <= NOT(a AND b AND c);
13 END dataflow;
```

Listing 3: 3-input NAND gate implementation: Dataflow model

dff.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY dff IS
5          PORT (
6                  CLK, DATA : IN STD_LOGIC;
7                  Q, QBAR : OUT STD_LOGIC);
8  END dff;
9
10 ARCHITECTURE structural OF dff IS
11         SIGNAL I1, I2, I3, I4 : STD_LOGIC;
12         SIGNAL I5 : STD_LOGIC := '0';
13         SIGNAL I6 : STD_LOGIC := '1';
14
15         COMPONENT nand_2
16                 PORT (
17                         a, b : IN STD_LOGIC;
18                         o : OUT STD_LOGIC
19                 );
20         END COMPONENT;
21
22         COMPONENT nand_3
23                 PORT (
24                         a, b, c : IN STD_LOGIC;
25                         o : OUT STD_LOGIC
26                 );
27         END COMPONENT;
28
29 BEGIN
30         Q <= I5;
31         QBAR <= I6;
32         O1 : nand_2 PORT MAP(a => I2, b => I4, o
                   => I1);
33         O2 : nand_2 PORT MAP(a => CLK, b =>
                   I1, o => I2);
34         O3 : nand_3 PORT MAP(a => I2, b =>
                   CLK, c => I4, o => I3);
35         O4 : nand_2 PORT MAP(a => I3, b =>
                   DATA, o => I4);
36         O5 : nand_2 PORT MAP(a => I2, b => I6, o
                   => I5);
37         O6 : nand_2 PORT MAP(a => I5, b => I3, o
                   => I6);
38 END structural;
```

Listing 4: D flipflop implementation: Structural model

jkff_st.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY jkff IS
5          PORT (
6                  CLOCK, J, K : IN STD_LOGIC;
7                  QT, QTBAR : INOUT STD_LOGIC);
8  END jkff;
9
10 ARCHITECTURE structural OF jkff IS
11         SIGNAL A1 : STD_LOGIC;
12         SIGNAL A5 : STD_LOGIC := '0';
13         SIGNAL A6 : STD_LOGIC := '1';
14
15         COMPONENT dff
16                 PORT (
17                         CLK, DATA : IN STD_LOGIC;
18                         Q, QBAR : INOUT
                               STD_LOGIC
19                 );
20         END COMPONENT;
21
22         COMPONENT combinational_1
23                 PORT (
24                         i1, i2, a3, a4 : IN
                                STD_LOGIC;
25                         o1 : OUT STD_LOGIC
26                 );
27         END COMPONENT;
28
29 BEGIN
30         QT <= A5;
31         QTBAR <= A6;
32         O1 : combinational_1 PORT MAP(i1 => J, i2
                => K, a3 => A5, a4 => A6, o1 => A1)
                ;
33         D1 : dff PORT MAP(CLK => CLOCK,
                DATA => A1, Q => A5, QBAR => A6)
                ;
34
35 END structural;
```

Listing 5: J-K flipflop implementation: Structural model

jkff_tb.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.NUMERIC_STD.ALL;
4
5  ENTITY jkff_tb IS
6  END jkff_tb;
7
8  ARCHITECTURE behavioral OF jkff_tb IS
9          COMPONENT jkff
10                 PORT (
11                         CLOCK : IN STD_LOGIC;
12                         J : IN STD_LOGIC;
13                         K : IN STD_LOGIC;
14                         QT : INOUT STD_LOGIC;
15                         QTBAR : INOUT
                                STD_LOGIC
16                 );
17         END COMPONENT;
18
19         SIGNAL clock : STD_LOGIC;
20         SIGNAL input_vector :
                STD_LOGIC_VECTOR (1 DOWNTO
                0) := "00";
21         SIGNAL out1 : STD_LOGIC := '0';
22         SIGNAL out2 : STD_LOGIC := '1';
23         CONSTANT clk_p : TIME := 100 ns;
24
25 BEGIN
26         uut : jkff PORT MAP(
27                 CLOCK => clock,
28                 J => input_vector(1),
29                 K => input_vector(0),
30                 QT => out1,
31                 QTBAR => out2
32         );
33
34         clkproc : PROCESS
35         BEGIN
36                 clock <= '0';
37                 WAIT FOR clk_p/2;
38                 clock <= '1';
39                 WAIT FOR clk_p/2;
40         END PROCESS clkproc;
41
42         stim_proc : PROCESS
43         BEGIN
44                 FOR index IN 0 TO 3 LOOP
45                         input_vector <=
                                STD_LOGIC_VECTOR
                                (to_unsigned(index, 2));
46                         WAIT FOR 100 ns;
47                 END LOOP;
48         END PROCESS;
49 END behavioral;
```

Listing 6: Testbench for all posible cases

Problem 2

Write a VHDL code to implement a T flip-flop using a D flip-flop and the characteristic equation. The D flip-flop needs to be implemented using a structural architecture style using the circuit shown in below figure. The T flip-flop must be constructed using a component declaration for a D flip-flop. Write a VHDL test bench to verify the operation of the T flip-flop and provide waveforms.

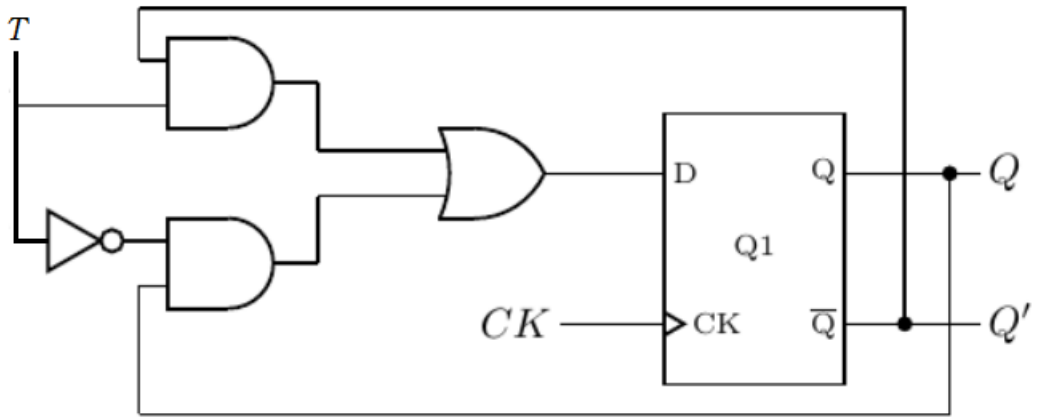


Figure 3: T flip flop using D flip flop

T flip flop, or also known as the toggle flip flop complements the output if the input T is high, else retains the state. For a D flip flop to be used as a T flip flop, the following equation must be implemented,

$$D = TQ' + T'Q$$

tff_st.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY tff IS
5      PORT (
6          CLOCK, T, TNOT: IN STD_LOGIC;
7          QT, QTBAR : INOUT STD_LOGIC);
8  END tff;
9
10 ARCHITECTURE structural OF tff IS
11     SIGNAL A1 : STD_LOGIC;
12     SIGNAL A5 : STD_LOGIC := '0';
13     SIGNAL A6 : STD_LOGIC := '1';
14
15     COMPONENT dff
16         PORT (
17             CLK, DATA : IN STD_LOGIC;
18             Q, QBAR : INOUT STD_LOGIC
19         );
20     END COMPONENT;
21
22     COMPONENT combinational_1
23         PORT (
24             i1, i2, a3, a4 : IN STD_LOGIC;
25             o1 : OUT STD_LOGIC
26         );
27     END COMPONENT;
28
29 BEGIN
30     QT <= A5;
31     QTBAR <= A6;
32     O1 : combinational_1 PORT MAP(i1 => T, i2 =>
           TNOT, a3 => A5, a4 => A6, o1 => A1);
33     D1 : dff PORT MAP(CLK => CLOCK, DATA
         => A1, Q => A5, QBAR => A6);
34
35 END structural;
```

Listing 7: T flipflop implementation: Structural model

tff_tb.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.NUMERIC_STD.ALL;
4
5  ENTITY tff_tb IS
6  END tff_tb;
7
8  ARCHITECTURE behavioral OF tff_tb IS
9          COMPONENT tff
10         PORT(
11                 CLOCK: IN STD_LOGIC;
12                 T: IN STD_LOGIC;
13                 TNOT: IN STD_LOGIC;
14                 QT: INOUT STD_LOGIC;
15                 QTBAR: INOUT STD_LOGIC
16         );
17         END COMPONENT;
18
19         SIGNAL clk: STD_LOGIC;
20         SIGNAL input: STD_LOGIC;
21         SIGNAL out1: STD_LOGIC:= '0';
22         SIGNAL out2: STD_LOGIC:= '1';
23         CONSTANT clk_p: time:= 100 ns;
24
25 BEGIN
26         uut: tff PORT MAP(
27                 CLOCK => clk,
28       T => input,
29       TNOT => not input,
30                 QT => out1,
31                 QTBAR => out2
32         );
33
34         clkproc: PROCESS
35         BEGIN
36                 clk <= '0';
37                 WAIT FOR clk_p/2;
38                 clk <= '1';
39                 WAIT FOR clk_p/2;
40         END PROCESS clkproc;
41
42         stim_proc: PROCESS
43         BEGIN
44                 if out1 = 'U' then
45                         input <= '0';
46                         WAIT FOR 100 ns;
47                 end if;
48
49                 input <= '0';
50                 WAIT FOR 100 ns;
51
52                 input <= '1';
53                 WAIT FOR 100 ns;
54         END PROCESS stim_proc;
55 END behavioral;
```

Listing 8: Testbench for all posible cases

Problem 3

Use VHDL to implement a 4-bit serial-in-serial-out (SISO) right-shift register as shown in figure below. Determine the output of the shift register after the input sequence 01010101 has been shifted eight times, starting with the MSB. Assume that the output of the shift register is reset initially to 0000. The shift register must be constructed with D flip-flops using a component declaration for a D flip-flop. Write a VHDL test bench to verify the operation of the 4-bit SISO and provide appropriate waveforms.

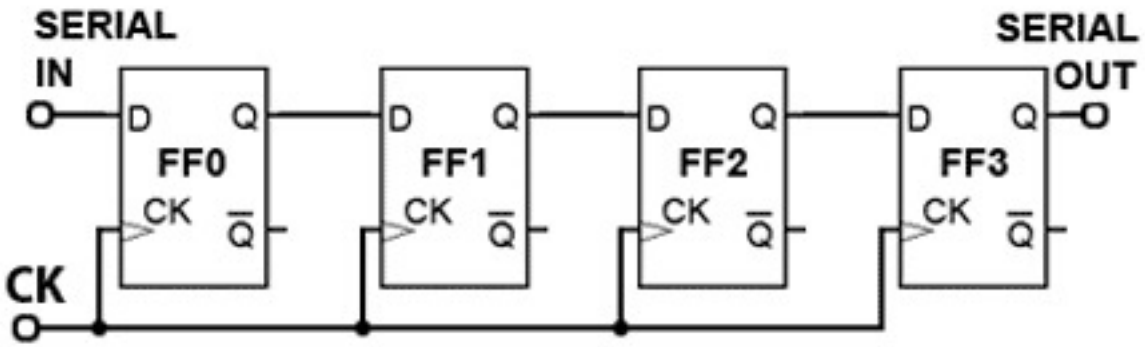


Figure 4: 4-bit serial-in-serial-out (SISO) right-shift register

siso_st.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY siso IS
5      PORT (
6          CLOCK, Din : IN STD_LOGIC;
7          QA, QB, QC : INOUT STD_LOGIC;
8          Dout, QTBAR : INOUT STD_LOGIC);
9  END siso;
10
11 ARCHITECTURE structural OF siso IS
12     SIGNAL Q0, Q0B, Q1, Q1B, Q2, Q2B, Q3, Q3B,
           A1 : STD_LOGIC;
13     SIGNAL A5 : STD_LOGIC := '0';
14     SIGNAL A6 : STD_LOGIC := '1';
15
16     COMPONENT dff
17         PORT (
18             CLK, DATA : IN STD_LOGIC;
19             Q, QBAR : INOUT STD_LOGIC
20         );
21     END COMPONENT;
22
23 BEGIN
24     Dout <= A5;
25     QTBAR <= A6;
26     QA <= Q0;
27     QB <= Q1;
28     QC <= Q2;
29     D1 : dff PORT MAP(CLK => CLOCK, DATA
           => Din, Q => Q0, QBAR => Q0B);
30     D2 : dff PORT MAP(CLK => CLOCK, DATA
           => Q0, Q => Q1, QBAR => Q1B);
31     D3 : dff PORT MAP(CLK => CLOCK, DATA
           => Q1, Q => Q2, QBAR => Q2B);
32     D4 : dff PORT MAP(CLK => CLOCK, DATA
           => Q2, Q => A5, QBAR => A6);
33
34 END structural;
```

Listing 9: 4-bit serial-in-serial-out right-shift register implementation: Structural model

siso_tb.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.NUMERIC_STD.ALL;
4
5  ENTITY siso_tb IS
6  END siso_tb;
7
8  ARCHITECTURE behavioral OF siso_tb IS
9      COMPONENT siso
10         PORT (
11             CLOCK : IN STD_LOGIC;
12             Din : IN STD_LOGIC;
13             QA, QB, QC : INOUT STD_LOGIC;
14             Dout : INOUT STD_LOGIC;
15             QTBAR : INOUT STD_LOGIC
16         );
17     END COMPONENT;
18
19     SIGNAL clk : STD_LOGIC;
20     SIGNAL input : STD_LOGIC;
21     SIGNAL input_seq : STD_LOGIC_VECTOR(7
           DOWNTO 0) := "10101010";
22     SIGNAL qa, qb, qc : STD_LOGIC;
23     SIGNAL out1 : STD_LOGIC := '1';
24     CONSTANT clk_p : TIME := 100 ns;
25
26 BEGIN
27     uut : siso PORT MAP(
28         CLOCK => clk,
29         QA => qa,
30         QB => qb,
31         QC => qc,
32         Din => input,
33         Dout => out1
34     );
35
36     clkproc : PROCESS
37     BEGIN
38         clk <= '0';
39         WAIT FOR clk_p/2;
40         clk <= '1';
41         WAIT FOR clk_p/2;
42     END PROCESS clkproc;
43
44     stim_proc : PROCESS
45     BEGIN
46         FOR index IN 1 TO 8 LOOP
47             input <= input_seq(7);
48             input_seq(7 DOWNTO 1) <=
49             input_seq(6 DOWNTO 0);
50             input_seq(0) <= '0';
51             WAIT FOR 100 ns;
52         END LOOP;
53         WAIT;
54     END PROCESS stim_proc;
55
56 END behavioral;
```

Listing 10: Testbench for all posible cases

Problem 4

Write a VHDL code to implement a 4-bit synchronous up-counter as shown in below figure. In a synchronous counter all flip-flops receive a common clock signal and change their states at the same time. The shift-register must be constructed with T flip-flops using a component declaration for a T flip-flop. Write a VHDL test bench to verify the operation of the 4-bit synchronous up-counter and provide the appropriate waveforms.

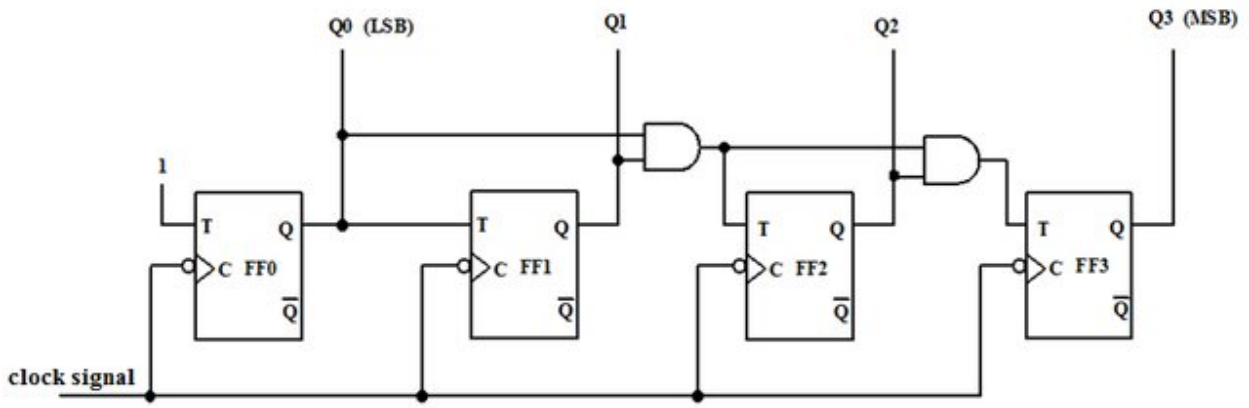


Figure 5: 4-bit synchronous up counter using T flip flop

tff_be.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY tff IS
5          PORT (
6                  CLOCK, RESET, T : IN STD_LOGIC;
7                  QT, QTBAR: OUT STD_LOGIC);
8  END tff;
9
10 ARCHITECTURE behavioral OF tff IS
11         SIGNAL output : STD_LOGIC;
12
13 BEGIN
14         PROCESS (RESET, CLOCK)
15         BEGIN
16                 IF RESET = '1' THEN
17                         output <= '0';
18                 ELSIF CLOCK'EVENT AND CLOCK
                           = '1' THEN
19                         IF T = '0' THEN
20                                 output <= output;
21                         ELSIF T = '1' THEN
22                                 output <= NOT
                                           output;
23                         ELSE
24                                 output <= 'U';
25                         END IF;
26                 END IF;
27         END PROCESS;
28         QT <= output;
29         QTBAR <= not output;
30 END behavioral;
```

Listing 11: T flipflop implementation: Behavioral model

syncup_st.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY syncup IS
5          PORT (
6                  clock, reset, input : IN STD_LOGIC;
7                  data : OUT STD_LOGIC_VECTOR
                          (3 DOWNTO 0));
8  END syncup;
9
10 ARCHITECTURE structural OF syncup IS
11         COMPONENT tff
12                 PORT (
13                         CLOCK, RESET, T : IN
                                STD_LOGIC;
14                         QT, QTBAR: OUT
                                STD_LOGIC
15                 );
16         END COMPONENT;
17
18         SIGNAL temp : STD_LOGIC_VECTOR(3
                DOWNTO 0) := "0000";
19         SIGNAL and_1, and_2 : STD_LOGIC;
20 BEGIN
21         and_1 <= temp(0) AND temp(1);
22         and_2 <= temp(2) AND and_1;
23
24         d0 : tff PORT MAP(CLOCK => clock,
                RESET => reset, T => '1', QT =>
                temp(0));
25
26         d1 : tff PORT MAP(CLOCK => clock,
                RESET => reset, T => temp(0), QT =>
                 temp(1));
27
28         d2 : tff PORT MAP(CLOCK => clock,
                RESET => reset, T => and_1, QT =>
                temp(2));
29
30         d3 : tff PORT MAP(CLOCK => clock,
                RESET => reset, T => and_2, QT =>
                temp(3));
31
32         data <= temp;
33
34 END structural;
```

Listing 12: 4-bit synchronous up counter implementation: Structural model

syncup_tb.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.NUMERIC_STD.ALL;
4  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  ENTITY syncup_tb IS
7  END syncup_tb;
8
9  ARCHITECTURE behavior OF syncup_tb IS
10     COMPONENT syncup
11         PORT (
12             clock : IN STD_LOGIC;
13             input : IN STD_LOGIC;
14             reset : IN STD_LOGIC;
15             data : OUT STD_LOGIC_VECTOR(3
                    DOWNTO 0));
16     END COMPONENT;
17
18     SIGNAL clock : STD_LOGIC := '0';
19     SIGNAL reset : STD_LOGIC := '1';
20     SIGNAL input : STD_LOGIC := '1';
21     SIGNAL data : STD_LOGIC_VECTOR(3
            DOWNTO 0);
22
23 BEGIN
24     uut : syncup PORT MAP(
25         clock => clock,
26         input => input,
27         reset => reset,
28         data => data
29     );
30
31     clk : PROCESS
32     BEGIN
33         WAIT FOR 5ns;
34         clock <= NOT clock;
35     END PROCESS clk;
36
37     main : PROCESS
38     BEGIN
39         WAIT FOR 7 ns;
40         reset <= '0';
41
42         WAIT FOR 20ns;
43         input <= '1';
44
45         WAIT;
46     END PROCESS main;
47
48 END behavior;
```

Listing 13: Testbench for all posible cases

Problem 5

Use VHDL to design an asynchronous decade counter as shown in below figure. The 10 states of a decade counter represent the BCD numbers from 0 to 9. Write the VHDL test bench to verify the operation of the decade counter and provide waveforms.

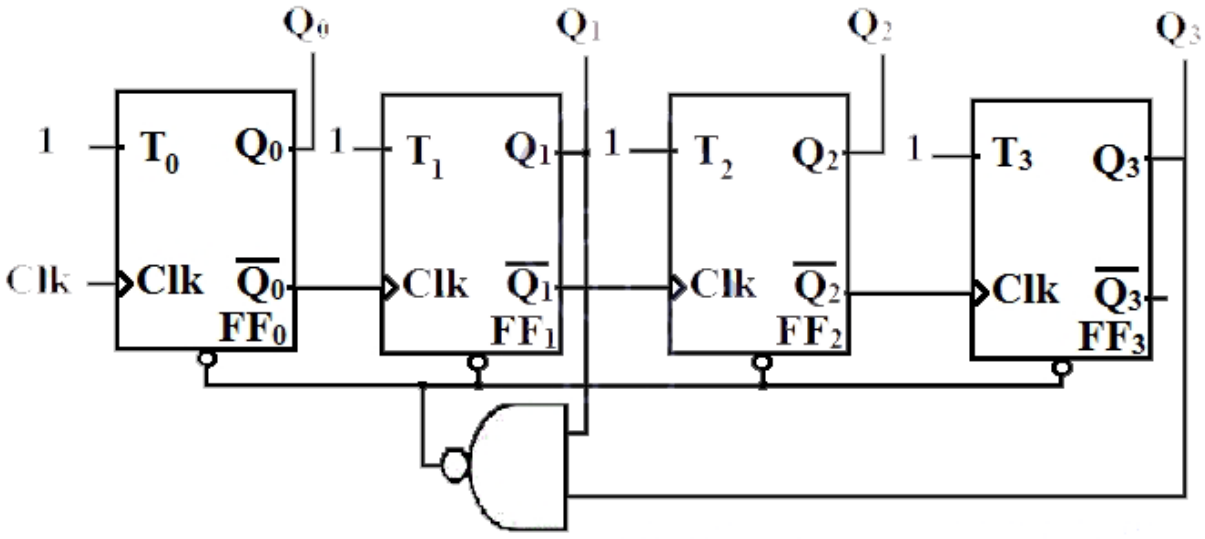


Figure 6: Asynchronous decade counter using T flip flop

asyncdec_st.vhd

```
1   LIBRARY IEEE;
2   USE IEEE.STD_LOGIC_1164.ALL;
3
4   ENTITY asyncdec IS
5       PORT (
6           clock, input : IN STD_LOGIC;
7           data : OUT STD_LOGIC_VECTOR(3
                DOWNTO 0));
8   END asyncdec;
9
10  ARCHITECTURE structural OF asyncdec IS
11      COMPONENT tff
12          PORT (
13              CLOCK, RESET, T : IN STD_LOGIC;
14              QT, QTBAR : OUT STD_LOGIC
15          );
16      END COMPONENT;
17
18      SIGNAL temp : STD_LOGIC_VECTOR(3
            DOWNTO 0) := "0000";
19      SIGNAL ntemp : STD_LOGIC_VECTOR(3
            DOWNTO 0) := "1111";
20      SIGNAL and_1 : STD_LOGIC := '1';
21  BEGIN
22      and_1 <= temp(1) AND temp(3);
23      d0 : tff PORT MAP(CLOCK => clock, RESET
            => and_1, T => input, QT => temp(0),
            QTBAR => ntemp(0));
24
25      d1 : tff PORT MAP(CLOCK => ntemp(0),
            RESET => and_1, T => input, QT => temp
            (1), QTBAR => ntemp(1));
26
27      d2 : tff PORT MAP(CLOCK => ntemp(1),
            RESET => and_1, T => input, QT => temp
            (2), QTBAR => ntemp(2));
28
29      d3 : tff PORT MAP(CLOCK => ntemp(2),
            RESET => and_1, T => input, QT => temp
            (3), QTBAR => ntemp(3));
30
31      data <= temp;
32
33  END structural;
```

Listing 14: 4-bit asynchronous decade counter implementation: Structural model

asyncdec_tb.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.NUMERIC_STD.ALL;
4  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  ENTITY asyncdec_tb IS
7  END asyncdec_tb;
8  ARCHITECTURE behavior OF asyncdec_tb IS
9          COMPONENT asyncdec
10                 PORT( clock : IN std_logic;
11 input : IN std_logic;
12                         data : out std_logic_vector(3
                                   downto 0) );
13         END COMPONENT;
14
15         SIGNAL clock : std_logic := '1';
16         SIGNAL input : std_logic;
17         SIGNAL data : std_logic_vector(3 downto 0) :
                 = "0000";
18
19         BEGIN
20         uut: asyncdec PORT MAP(
21                 clock => clock,
22                 input => input,
23                 data  => data
24         );
25
26         clk: PROCESS
27         BEGIN
28                 wait for 5ns;
29         clock <= not clock;
30         END PROCESS clk;
31
32         main: PROCESS
33         BEGIN
34         input <= '1';
35                 wait;
36         END PROCESS main;
37
38 END behavior;
```

Listing 15: Testbench for all posible cases

Problem 6

Use VHDL to design a four-bit Johnson counter as shown in below figure. The Johnson counter must be constructed using component declarations from the D flip-flops. Write a VHDL test bench to verify the operation of the Johnson counter and provide waveforms.

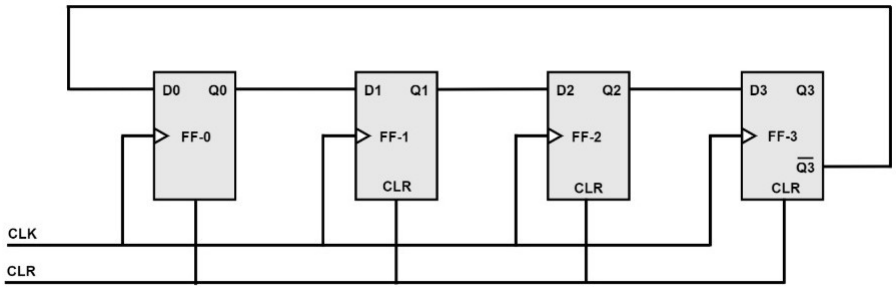


Figure 7: 4-bit johnson counter using D flip flop

dff_st.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.STD_LOGIC_ARITH.ALL;
4  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  ENTITY dff IS
7      PORT (
8          clock, reset, d : IN STD_LOGIC;
9          q, qnot: OUT STD_LOGIC);
10 END dff;
11
12 ARCHITECTURE structural OF dff IS
13 BEGIN
14     PROCESS (clock, d)
15     BEGIN
16         IF reset = '1' THEN
17                     q <= '0';
18                 ELSIF(clock'EVENT AND clock =
                       '1')
19             THEN
20             q <= d;
21                                 qnot <= not d;
22         END IF;
23     END PROCESS;
24
25 END structural;
```

Listing 16: D flipflop implementation: Structural model

john_st.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.STD_LOGIC_ARITH.ALL;
4  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  ENTITY johnson IS PORT (
7      clock : IN STD_LOGIC;
8      reset : IN STD_LOGIC;
9      data : OUT STD_LOGIC_VECTOR(3
           DOWNTO 0));
10 END johnson;
11
12 ARCHITECTURE structural OF johnson IS
13     COMPONENT dff
14         PORT (
15             clock : IN STD_LOGIC;
16             reset : IN STD_LOGIC;
17             d : IN STD_LOGIC;
18             q : OUT STD_LOGIC
19         );
20     END COMPONENT;
21
22     SIGNAL temp : STD_LOGIC_VECTOR(3
           DOWNTO 0) := "0000";
23     SIGNAL wir : STD_LOGIC := '0';
24 BEGIN
25     wir <= NOT temp(0);
26     d0 : dff PORT MAP(reset => reset, clock =>
           clock, d => wir, q => temp(3));
27
28     d1 : dff PORT MAP(reset => reset, clock =>
           clock, d => temp(3), q => temp(2));
29
30     d2 : dff PORT MAP(reset => reset, clock =>
           clock, d => temp(2), q => temp(1));
31
32     d3 : dff PORT MAP(reset => reset, clock =>
           clock, d => temp(1), q => temp(0));
33
34     data <= temp;
35
36 END structural;
```

Listing 17: 4-bit johnson counter implementation: Structural model

john_tb.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY johnson_tb IS
5  END johnson_tb;
6
7  ARCHITECTURE behavior OF johnson_tb IS
8      COMPONENT johnson
9          PORT (
10             clock : IN STD_LOGIC;
11             reset : IN STD_LOGIC;
12             data : OUT STD_LOGIC_VECTOR(3
                   DOWNTO 0));
13     END COMPONENT;
14
15     SIGNAL clock : STD_LOGIC := '0';
16     SIGNAL reset : STD_LOGIC := '1';
17     SIGNAL data : STD_LOGIC_VECTOR(3
           DOWNTO 0);
18
19 BEGIN
20     uut : johnson PORT MAP(
21             clock => clock,
22             reset => reset,
23             data => data
24     );
25
26     clk : PROCESS
27     BEGIN
28         WAIT FOR 5ns;
29         clock <= NOT clock;
30     END PROCESS clk;
31
32     main : PROCESS
33     BEGIN
34         reset <= '1';
35         WAIT FOR 20ns;
36
37         reset <= NOT reset;
38         WAIT;
39     END PROCESS main;
40
41 END behavior;
```

Listing 18: Testbench for all posible cases

Problem 7

Use VHDL to create a 2-bit BCD counter as shown in below figure. The BCD counter consists of two 1-bit BCD counters cascaded to form a 2-bit BCD counter. The 2-bit BCD counter counts from 00 to 99. The 2-bit BCD counter must be constructed using component declarations for the D flip-flops.

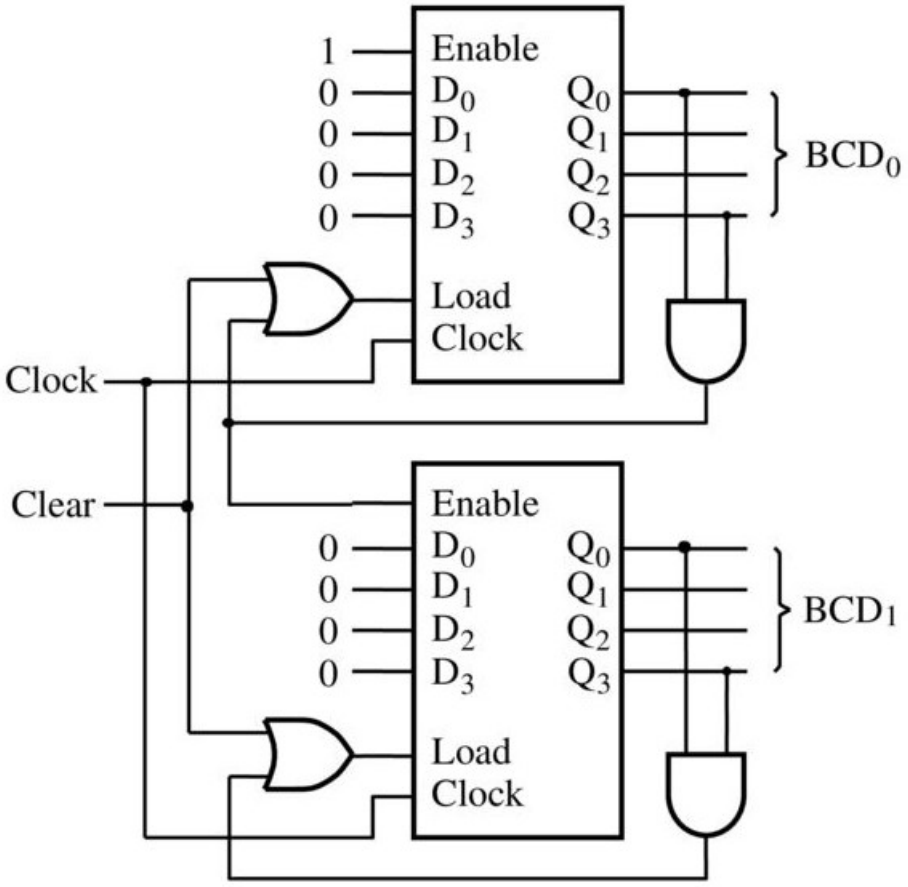


Figure 8: 2-bit BCD counter by cascading two 1-bit BCD counters

q7diff.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.STD_LOGIC_ARITH.ALL;
4  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  ENTITY dff IS
7      PORT (
8          enable : IN STD_LOGIC;
9          d : IN STD_LOGIC;
10         clock : IN STD_LOGIC;
11         reset : IN STD_LOGIC;
12         q : OUT STD_LOGIC;
13         qb : OUT STD_LOGIC
14     );
15 END dff;
16
```

```
17 ARCHITECTURE behavioral OF dff IS
18 BEGIN
19     PROCESS (reset, clock, enable)
20     BEGIN
21         IF (enable = '1') THEN
22             IF (reset = '1') THEN
23                 q <= '0';
24                 qb <= '1';
25             ELSIF (clock'EVENT AND clock = '1')
                   THEN
26                 q <= d;
27                 qb <= NOT d;
28             END IF;
29         END IF;
30     END PROCESS;
31 END behavioral;
```

Listing 19: D flipflop implementation with enable pin: Structural model

q7bcd_st.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3
4  ENTITY bit2dec IS
5      PORT (
6          clock, reset : IN STD_LOGIC;
7          output1 : OUT STD_LOGIC_VECTOR
8          (3 DOWNTO 0);
9          output2 : OUT STD_LOGIC_VECTOR
10         (3 DOWNTO 0));
11 END bit2dec;
12
13 ARCHITECTURE structural OF bit2dec IS
14     COMPONENT dff
15         PORT (
16             clock, enable, reset, d :
17             IN STD_LOGIC;
18             q, qb : OUT STD_LOGIC
19         );
20     END COMPONENT;
21
22     SIGNAL temp1 : STD_LOGIC_VECTOR (3
           DOWNTO 0) := "0000";
23     SIGNAL temp2 : STD_LOGIC_VECTOR (3
           DOWNTO 0) := "0101";
24     SIGNAL ntemp1 : STD_LOGIC_VECTOR (3
           DOWNTO 0) := "1111";
25     SIGNAL ntemp2 : STD_LOGIC_VECTOR (3
           DOWNTO 0) := "1010";
26     SIGNAL and_1 : STD_LOGIC := '1';
27     SIGNAL and_2 : STD_LOGIC := '1';
28
29 BEGIN
30     d0 : dff PORT MAP(clock => clock, enable =>
           '1', reset => and_1, d => ntemp1(0), q =>
           temp1(0), qb => ntemp1(0));
31     d1 : dff PORT MAP(clock => ntemp1(0), enable
           => '1', reset => and_1, d => ntemp1(1), q
           => temp1(1), qb => ntemp1(1));
32     d2 : dff PORT MAP(clock => ntemp1(1), enable
           => '1', reset => and_1, d => ntemp1(2), q
           => temp1(2), qb => ntemp1(2));
33     d3 : dff PORT MAP(clock => ntemp1(2), enable
           => '1', reset => and_1, d => ntemp1(3), q
           => temp1(3), qb => ntemp1(3));
34     and_1 <= temp1(1) AND temp1(3);
35     output1 <= temp1;
36     d4 : dff PORT MAP(clock => and_1, enable =>
           '1', reset => and_2, d => ntemp2(0), q =>
           temp2(0), qb => ntemp2(0));
37     d5 : dff PORT MAP(clock => ntemp2(0), enable
           => '1', reset => and_2, d => ntemp2(1), q
           => temp2(1), qb => ntemp2(1));
38     d6 : dff PORT MAP(clock => ntemp2(1), enable
           => '1', reset => and_2, d => ntemp2(2), q
           => temp2(2), qb => ntemp2(2));
39     d7 : dff PORT MAP(clock => ntemp2(2), enable
           => '1', reset => and_2, d => ntemp2(3), q
           => temp2(3), qb => ntemp2(3));
40     and_2 <= temp2(1) AND temp2(3);
41     output2 <= temp2;
42
43 END structural;
```

Listing 20: 2-bit BCD counter implementation: Structural model

q7bcd_tb.vhd

```
1  LIBRARY IEEE;
2  USE IEEE.STD_LOGIC_1164.ALL;
3  USE IEEE.NUMERIC_STD.ALL;
4  USE IEEE.STD_LOGIC_UNSIGNED.ALL;
5
6  ENTITY bit2dec_tb IS
7  END bit2dec_tb;
8  ARCHITECTURE behavioral OF bit2dec_tb IS
9          COMPONENT bit2dec
10                 PORT( clock, reset: IN STD_LOGIC;
11                       output1 : out std_logic_vector
                             (3 downto 0);
12                       output2 : out std_logic_vector
                             (3 downto 0)
13                 );
14         END COMPONENT;
15
16         SIGNAL clock : std_logic := '1';
17         SIGNAL reset : std_logic := '1';
18         SIGNAL out1 : std_logic_vector (3 downto 0) :
               = "0000";
19         SIGNAL out2 : std_logic_vector (3 downto 0) :
               = "0101";
20
21         BEGIN
22         uut: bit2dec PORT MAP(
23                 clock => clock,
24                 reset  => reset,
25                 output1 => out1,
26                 output2 => out2
27         );
28
29         clk: PROCESS
30         BEGIN
31                 wait for 5ns;
32                 clock <= not clock;
33         END PROCESS clk;
34
35         main: PROCESS
36         BEGIN
37                 wait for 5 ns;
38                 reset <= '0';
39                 wait;
40         END PROCESS main;
41
42 END behavioral;
```

Listing 21: Testbench for all posible cases

5 Observations

The observations for the ISim simulator are printed to individual .pdf fles using the print option available within the simulator. The time frames have been selected such that all the possible input cases are included at least once within the waveform. The expected outputs and the simulated waveform match leading us to a conclusion that the designed sequential circuits are functional. The RTL schematic for each question have been included with this report since it was also a major part of the observation. The RTL schematic shows the internal logic gates that are used starting from the top module.

Problem 1

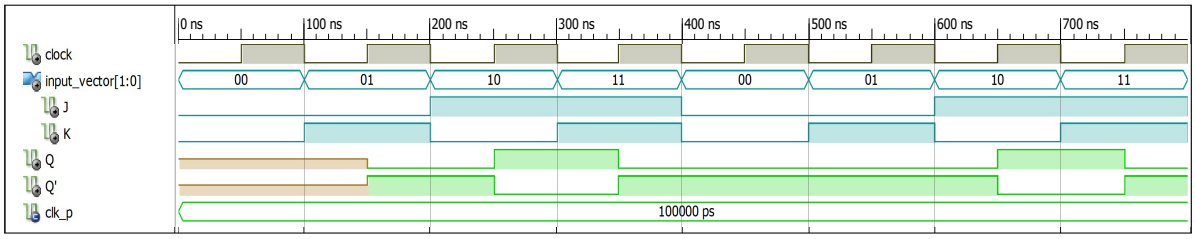


Figure 9: Observed waveform for Problem 1

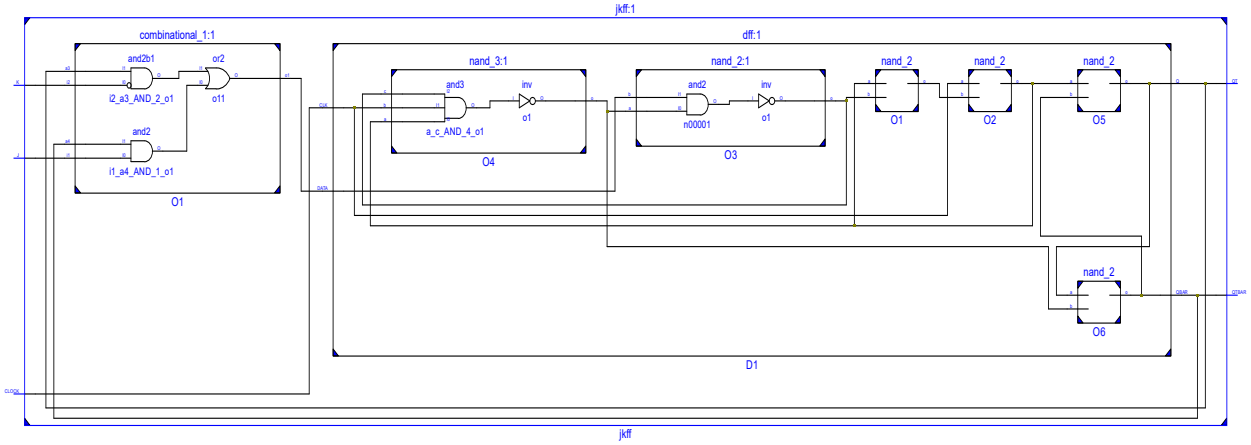


Figure 10: Observed RTL schematic for Problem 1

Problem 2

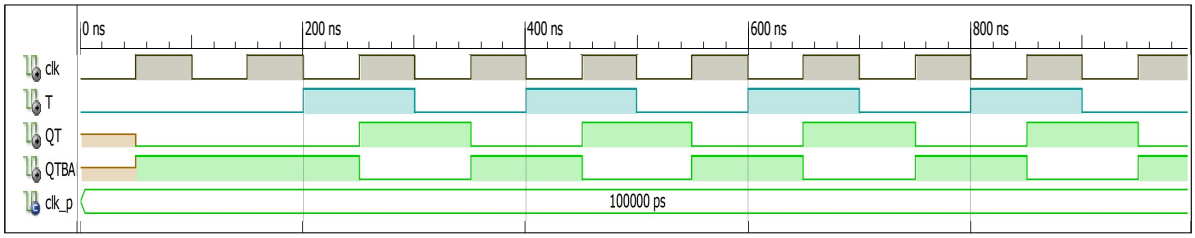


Figure 11: Observed waveform for Problem 2

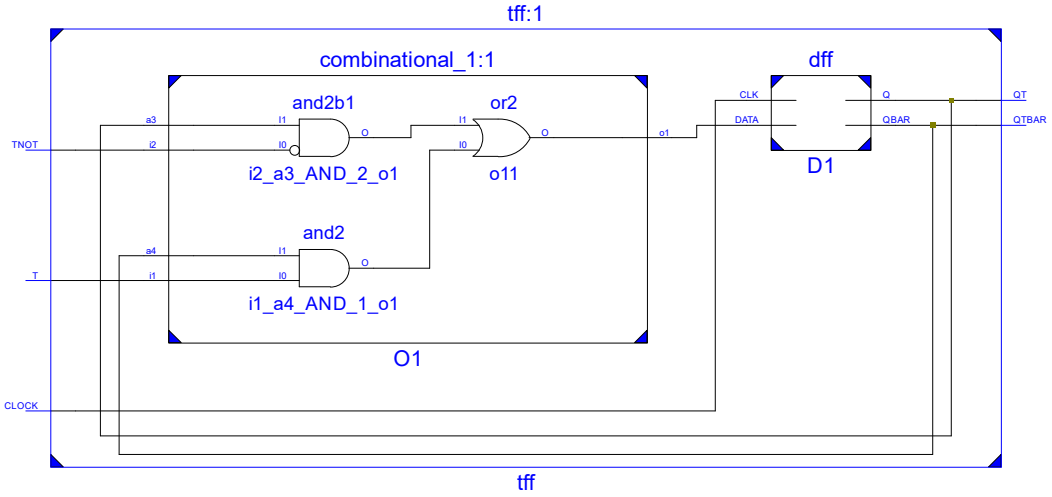


Figure 12: Observed RTL schematic for Problem 2

Problem 3

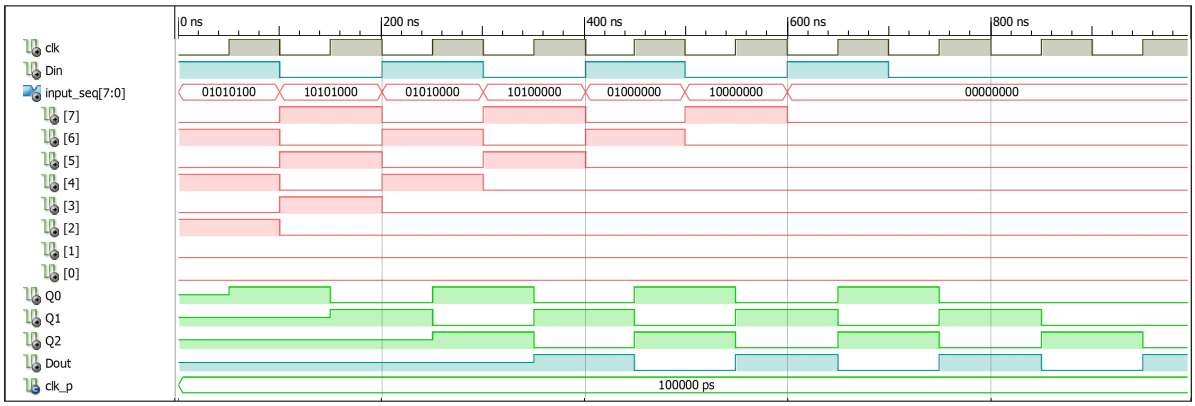


Figure 13: Observed waveform for Problem 3

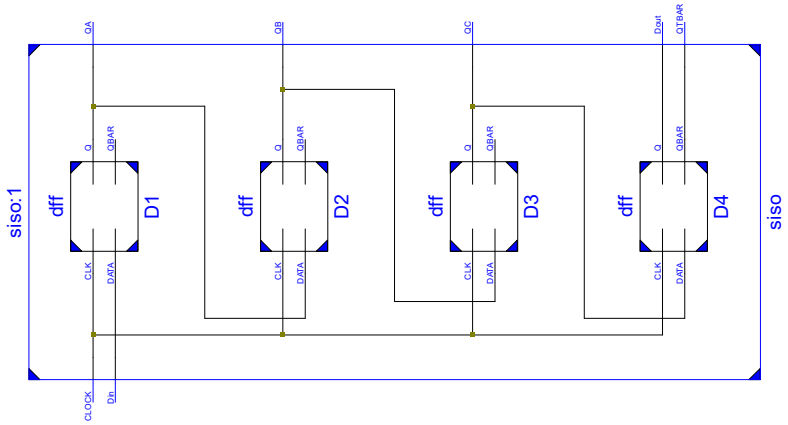
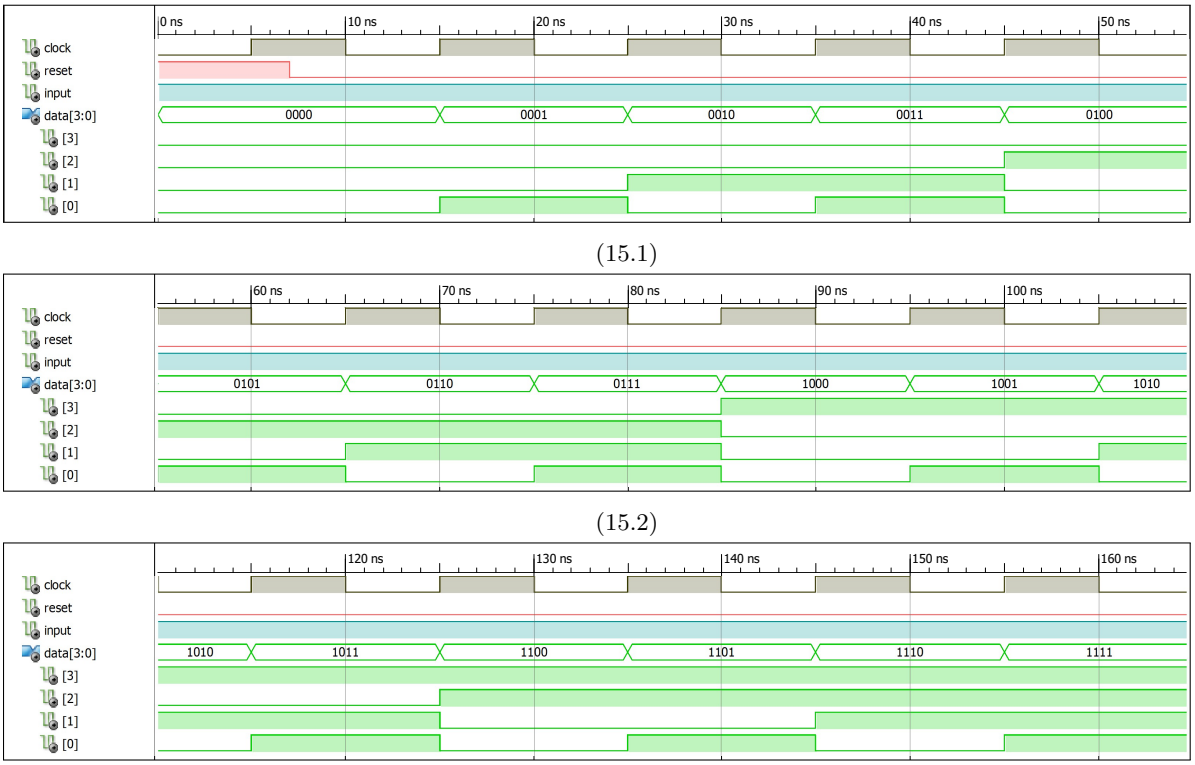


Figure 14: Observed RTL schematic for Problem 3

Problem 4



(15.1)

(15.2)

(15.3)

Figure 15: Observed waveform for Problem 4

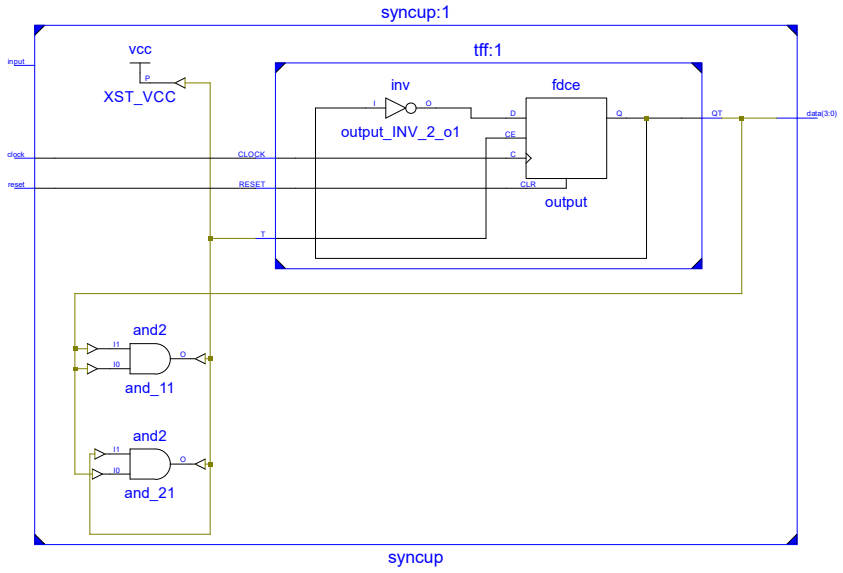
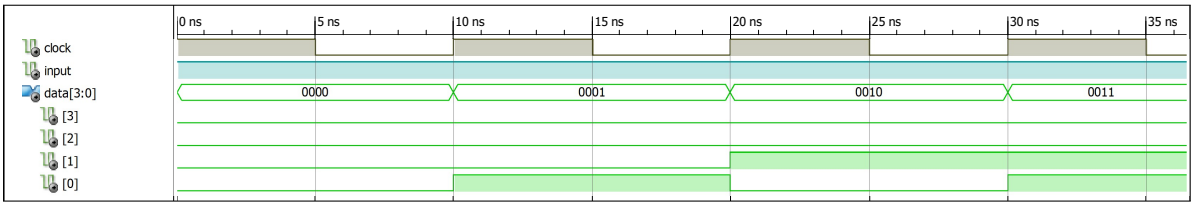


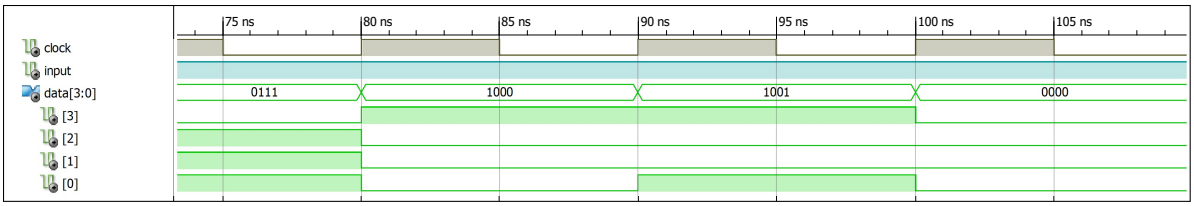
Figure 16: Observed RTL schematic for Problem 4

Problem 5



(17.1)

(17.2)



(17.3)

Figure 17: Observed waveform for Problem 5

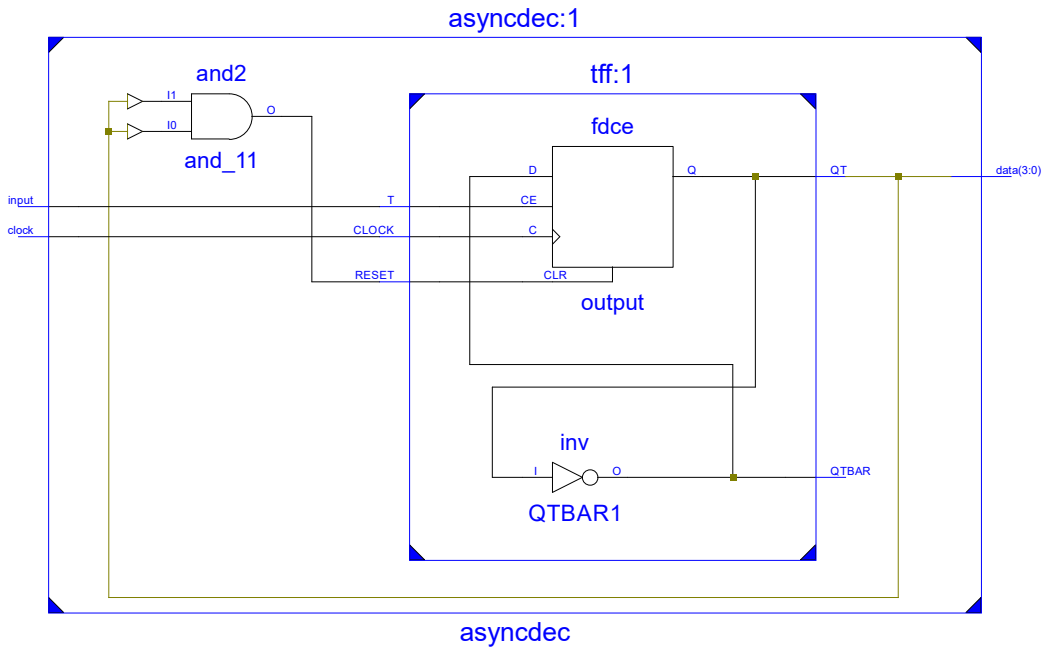
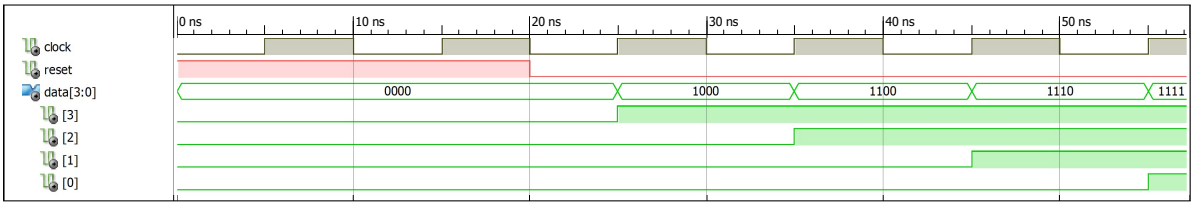
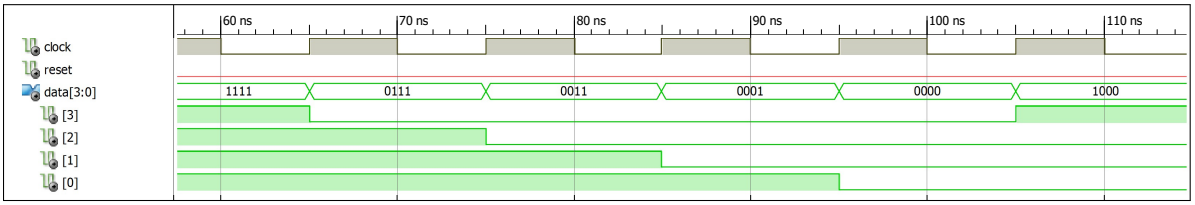


Figure 18: Observed RTL schematic for Problem 5

Problem 6



(19.1)



(19.2)

Figure 19: Observed waveform for Problem 6

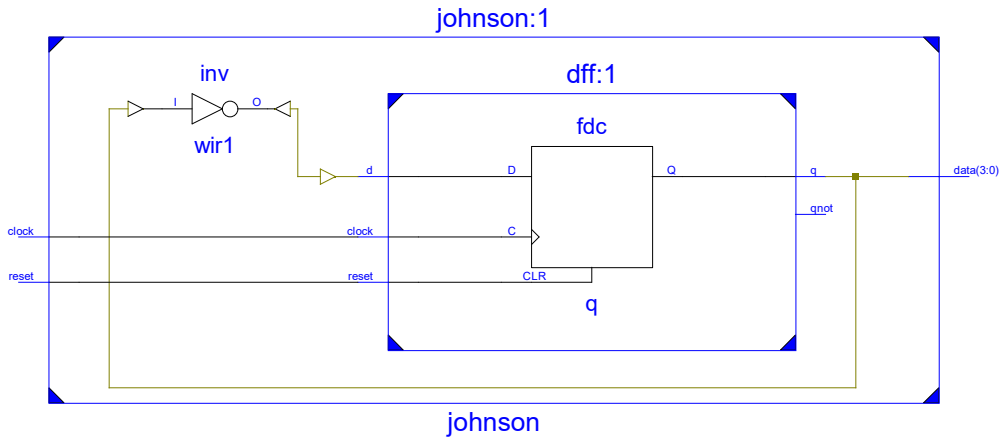
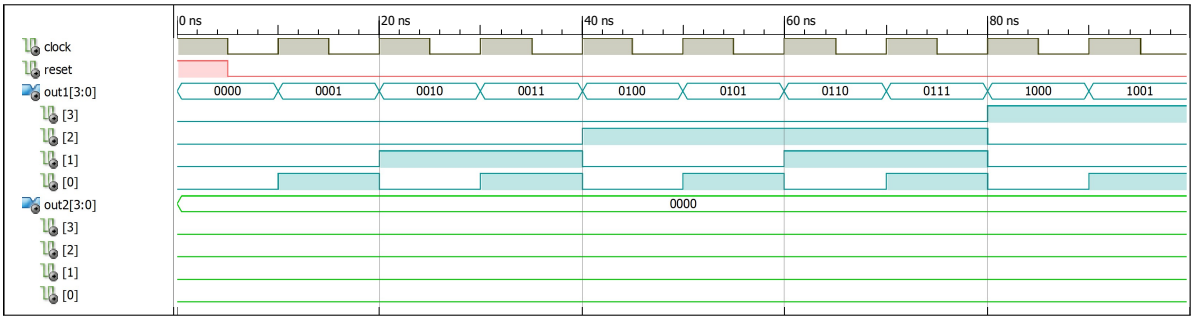


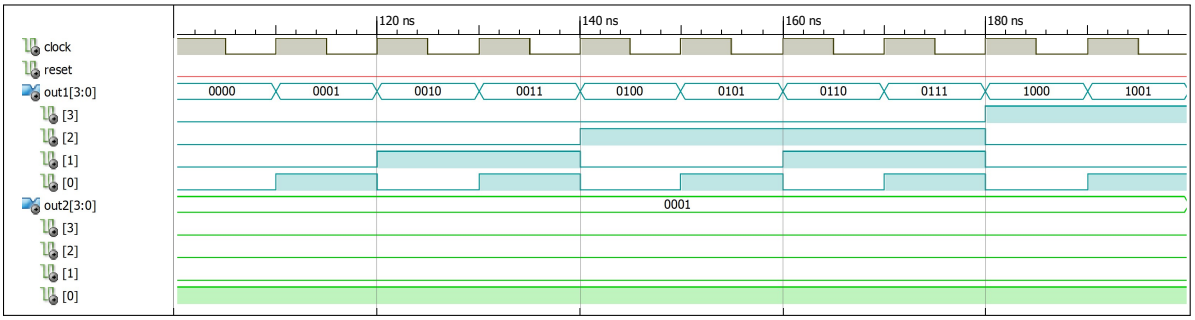
Figure 20: Observed RTL schematic for Problem 6

Problem 7

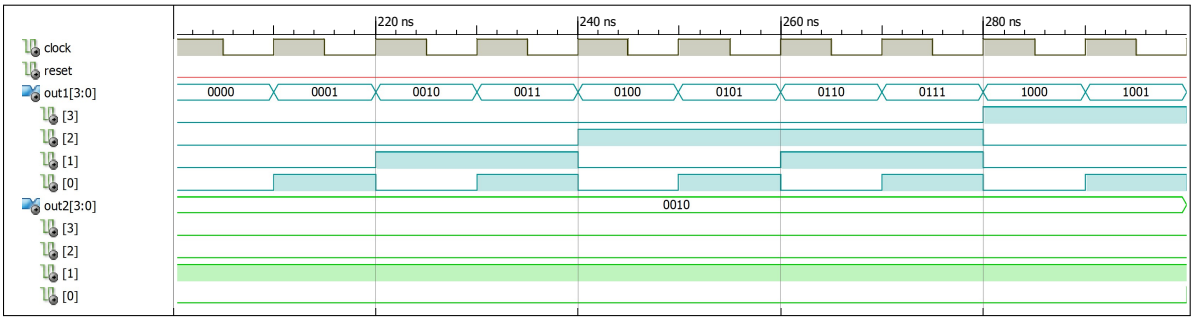


(21.1)

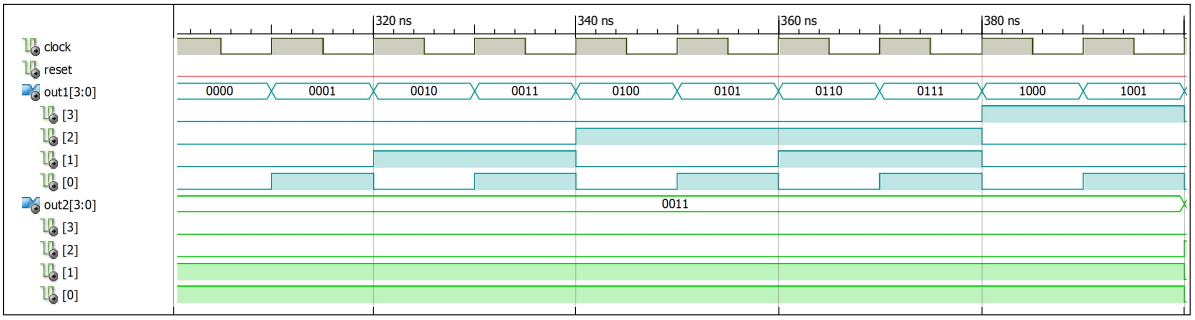
Figure 21: Observed waveform for Problem 7



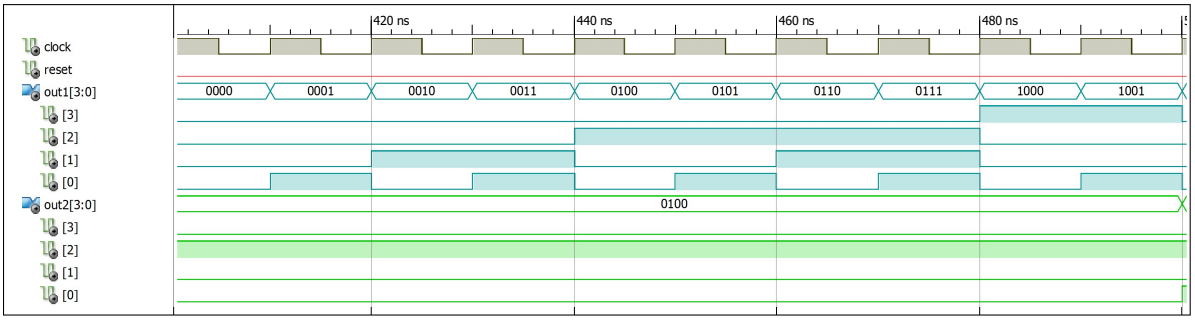
(21.2)



(21.3)

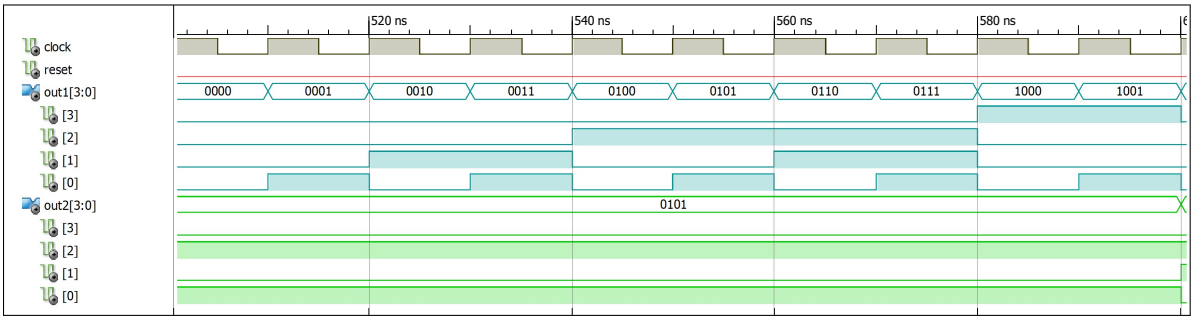


(21.4)

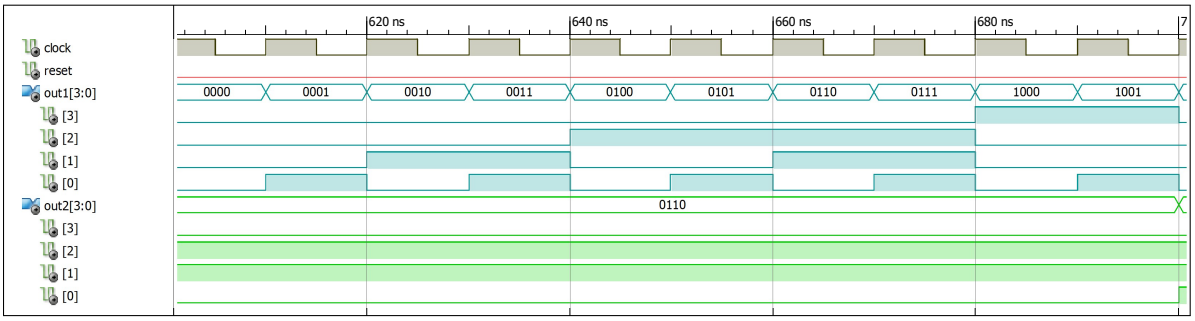


(21.5)

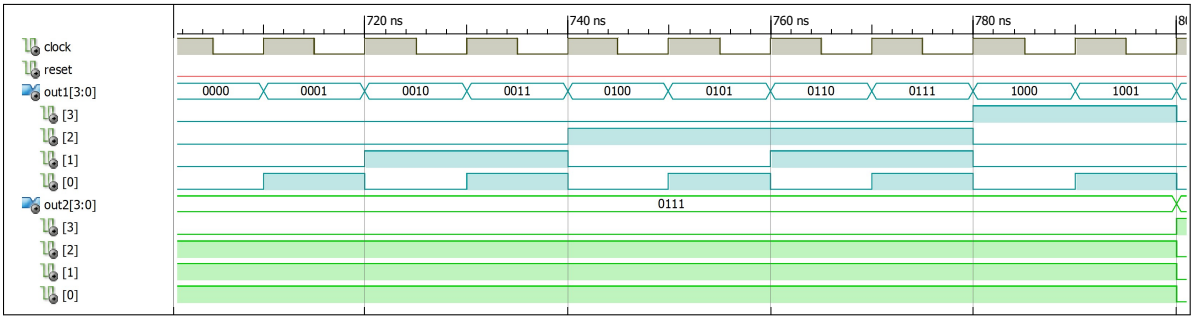
Figure 21: Observed waveform for Problem 7 (continued)



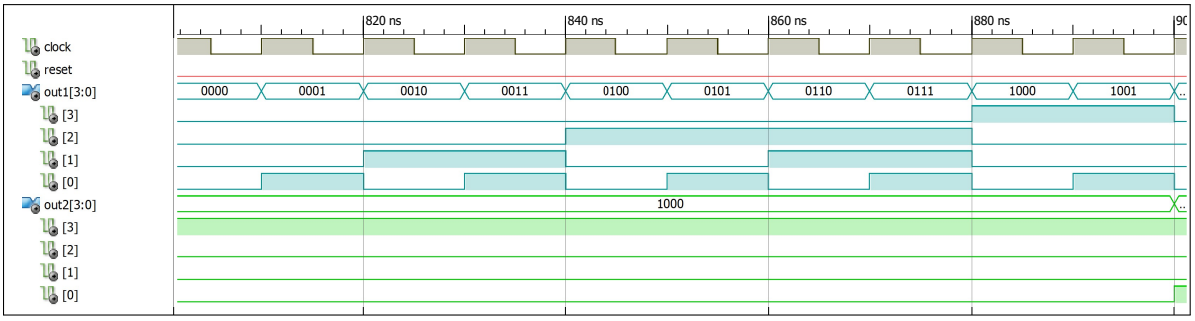
(21.6)



(21.7)

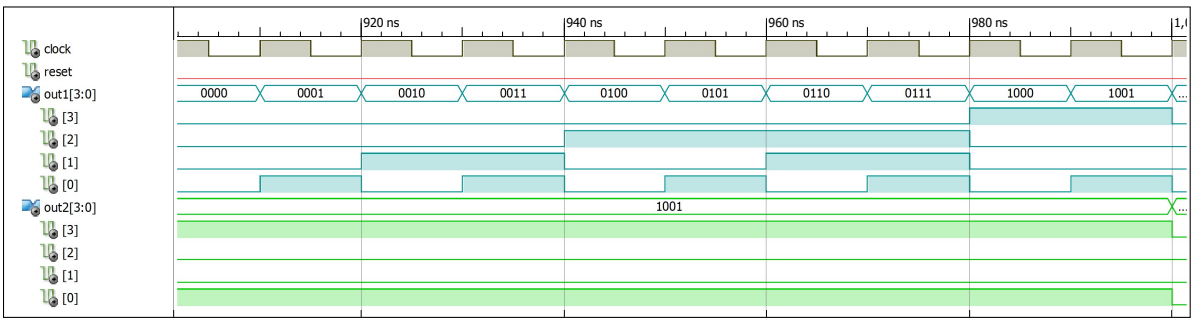


(21.8)



(21.9)

Figure 21: Observed waveform for Problem 7 (continued)



(21.10)

Figure 21: Observed waveform for Problem 7 (continued)

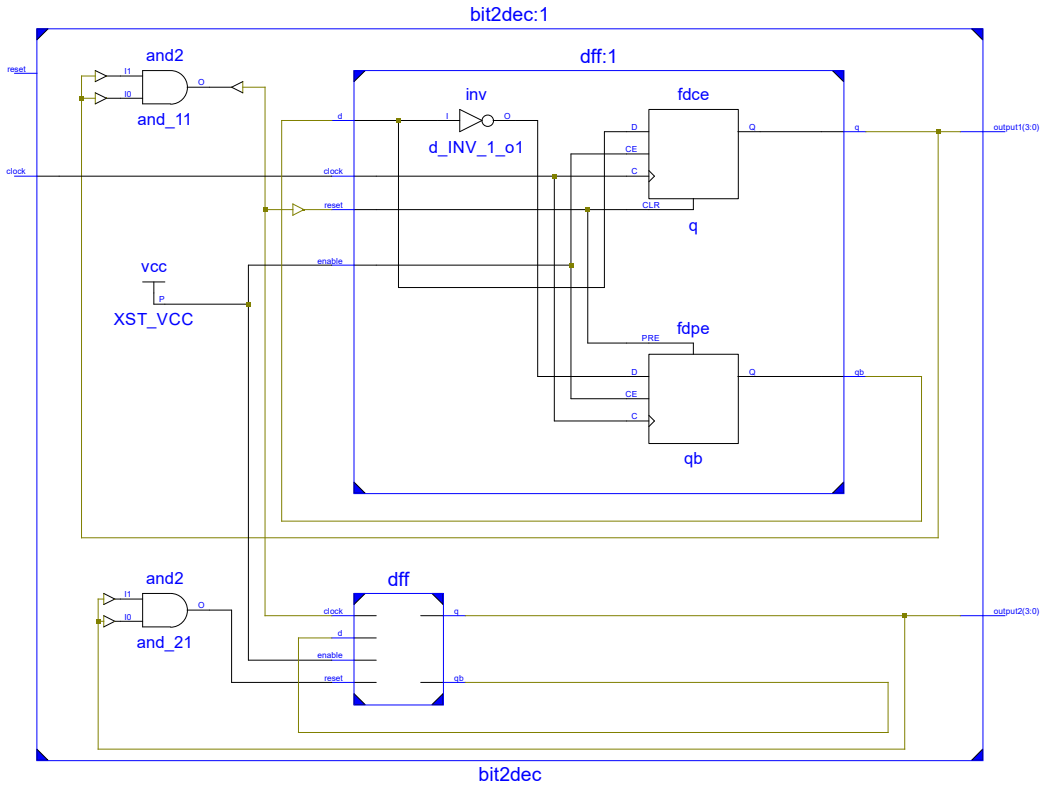


Figure 22: Observed RTL schematic for Problem 7

6 Discussion

In this lab experiment, various programs related to sequential logic circuits were written in VHDL. Problems related to JK and T flip flops using D flip flops and basic gates, SISO register, synchronous up, asynchronous decade, johnson and two bit BCD counters allowed us to be familiar with VHDL programming concepts that are involved while designing a FPGA. The waveforms are results of ISim, a full-featured HDL simulator which was initiated by test bench programs written in VHDL. The test bench codes have been included with this report as per the instructions. RTL schematics of the combinational circuits have been included with this report since it provides a clear idea of the various modules that are used while in a complex circuit.

Additional References

[1] *VHDL Reference Manual*. en. [Online]. Available: URL: https://www.ics.uci.edu/~jmoorkan/vhdlref/Synario%20VHDL%20Manual.pdf.

[2] D. Sharma. *An Introduction to VHDL*. en. [Online]. Available: URL: https://www.ee.iitb.ac.in/~smdp/DKStutorials/vhdl-overview.pdf.

[3] *ISE In-Depth Tutorial*. en. [Online]. Available: URL: https://www.xilinx.com/support/documentation/sw_manuals/xilinx13_3/ise_tutorial_ug695.pdf.