Introduction to Logic Design
EEF205E

Homework 5

Rüzgar Erik
040240783

Istanbul Technical University
Faculty of Electrical and Electronics Engineering

January 11, 2025

Question 1

Recalling the JK flip-flop equations we can derive $A(t+1)$ and $B(t+1)$ as follows:

$$Q(t+1) = J \cdot \overline{Q} + \overline{K} \cdot Q \tag{1}$$

For the A flip-flop: $J_A = x$ and $K_A = b$

$$A(t+1) = x \cdot \overline{A} + \overline{b} \cdot A \tag{2}$$

For the B flip-flop: $J_B = x$ and $K_B = \overline{a}$

$$B(t+1) = x \cdot \overline{B} + a \cdot B \tag{3}$$

Given that there are 2 flip-flops there are $2^2 = 4$ states and we can write the state transition table as follows:

Table 1: State Transition Table

Present State			Next State	
A	B	x	A(t+1)	B(t+1)
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	1	0
1	0	0	1	0
1	0	1	1	1
1	1	0	0	1
1	1	1	0	1

We can draw the state diagram as follows:

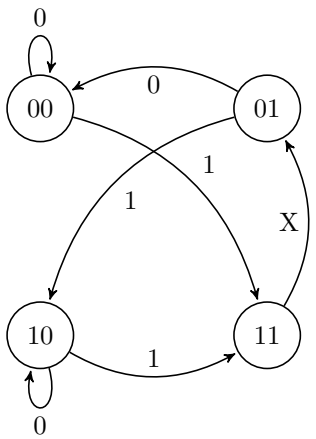

Figure 1: State Diagram

Question 2

The logic diagram for the state machine can be drawn as follows:

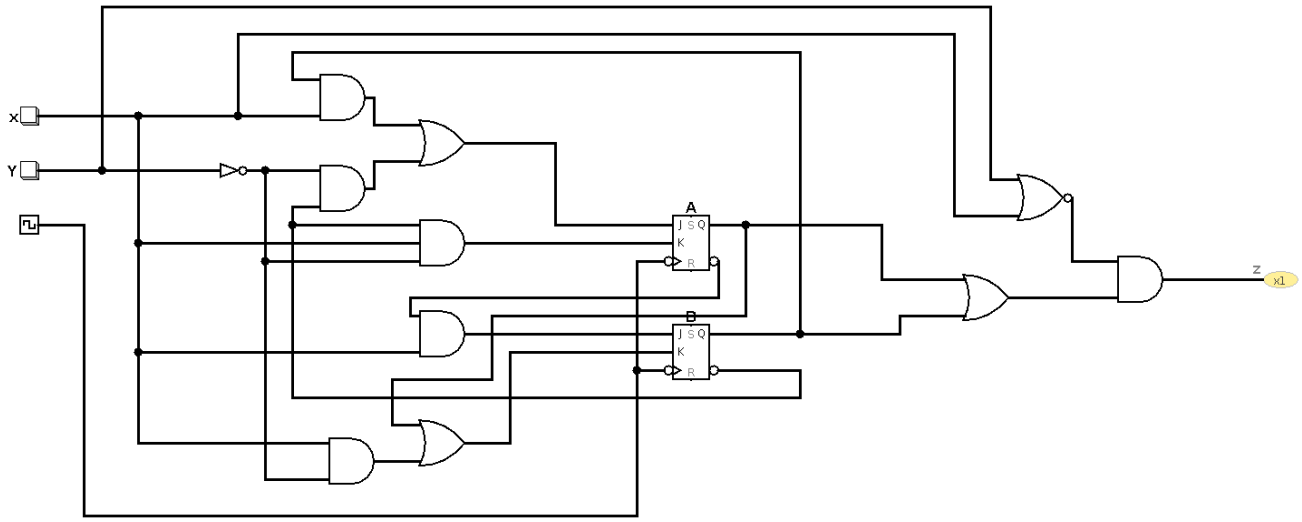

Figure 2: Logic Diagram for the State Machine

The state table can be written as follows:

Figure 3: State Table

Present State				**Next State**		
A	**B**	**x**	**y**	**A***	**B***	**z**
0	0	0	0	1	0	0
0	0	0	1	0	0	0
0	0	1	0	1	1	0
0	0	1	1	0	1	0
0	1	0	0	0	1	1
0	1	0	1	0	1	0
0	1	1	0	1	0	0
0	1	1	1	1	1	0
1	0	0	0	1	0	1
1	0	0	1	1	0	0
1	0	1	0	0	0	0
1	0	1	1	1	0	0
1	1	0	0	1	0	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0
1	1	1	1	1	0	0

Recalling the JK flip-flop equations we can derive $A(t+1)$ and $B(t+1)$ as follows:

$$Q(t+1) = J \cdot \overline{Q} + \overline{K} \cdot Q \tag{4}$$

For the A flip-flop:

$$A^* = (b \cdot x + b' \cdot y') \cdot A' + (b + x' + y) \cdot A \tag{5}$$

$$B^* = (a' \cdot x) \cdot B' + [a' \cdot (x' + y)] \cdot B \tag{6}$$

Question 3

Part A

The state table for the requested circuit can be written as follows:

Table 2: State Table

Present State			Next State	
A	B	x	A(t+1)	B(t+1)
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	1
1	0	0	1	0
1	0	1	0	0
1	1	0	1	1
1	1	1	1	0

To design the circuit with D Flip Flops We need its excitation table:

Table 3: D Flip-Flop Excitation Table

Present State	Next State	D Input
0	0	0
0	1	1
1	0	0
1	1	1

It can be seen that the equation for the D flip-flop is $D = Q(t+1)$ so we can take the next state values from the state table and write the D values as follows:

Table 4: State Table

Present State			Next State		D Inputs	
A	B	x	A(t+1)	B(t+1)	D_A	D_B
0	0	0	0	0	0	0
0	0	1	0	1	0	1
0	1	0	0	1	0	1
0	1	1	1	1	1	1
1	0	0	1	0	1	0
1	0	1	0	0	0	0
1	1	0	1	1	1	1
1	1	1	1	0	1	0

We can find equations for D_A and D_B with the help of kmaps:

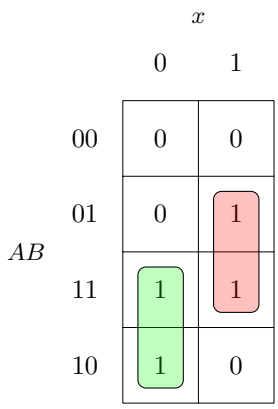

Figure 4: K-Map for D_A

The simplified equation for D_A is:

$$D_A = (\overline{x_{in}} \cdot A) + (x_{in} \cdot B) \tag{7}$$

We can draw the K-Map for D_B as follows:

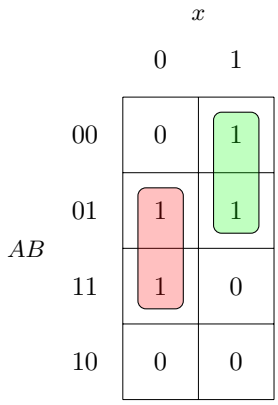

Figure 5: K-Map for D_B

The simplified equation for D_B is:

$$D_B = (\overline{x_{in}} \cdot B) + (x_{in} \cdot \overline{A}) \tag{8}$$

All of the equations can be implemented in a circuit as follows:

$$D_A = (\overline{x_{in}} \cdot A) + (x_{in} \cdot B) \tag{9}$$

$$D_B = (\overline{x_{in}} \cdot B) + (x_{in} \cdot \overline{A}) \tag{10}$$

Part B

The state table for the requested circuit can be written as follows:

Table 5: State Table

Present State			Next State	
A	B	x	A(t+1)	B(t+1)
0	0	0	0	0
0	0	1	1	1
0	1	0	0	1
0	1	1	1	0
1	0	0	1	0
1	0	1	0	0
1	1	0	1	1
1	1	1	0	1

Again using the principle of D flip-flops we can add the D inputs to the state table as follows:

Table 6: State Table

Present State			Next State		D Inputs	
A	B	x	A(t+1)	B(t+1)	D_A	D_B
0	0	0	0	0	0	0
0	0	1	1	1	1	1
0	1	0	0	1	0	1
0	1	1	1	0	1	0
1	0	0	1	0	1	0
1	0	1	0	0	0	0
1	1	0	1	1	1	1
1	1	1	0	1	0	1

We can find equations for D_A and D_B with the help of kmaps:

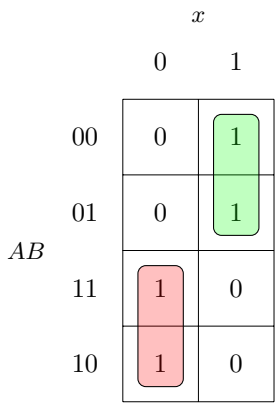

Figure 6: K-Map for D_A

The simplified equation for D_A is:

$$D_A = (\overline{x_{in}} \cdot A) + (x_{in} \cdot \overline{A}) \tag{11}$$

This shows an XOR gate is needed for the implementation of D_A.

$$D_A = x_{in} \oplus A \tag{12}$$

We can draw the K-Map for D_B as follows:

AB \ x	0	1
00	0	1
01	1	0
11	1	1
10	0	0

Figure 7: K-Map for D_B

The simplified equation for D_B is:

$$D_B = (\overline{x_{in}} \cdot B) + (x_{in} \cdot \overline{B} \cdot \overline{A}) + (x_{in} \cdot A \cdot B) \tag{13}$$

Also it can be simplified with XNOR gates as follows:

$$D_B = (\overline{x_{in}} \cdot B) + (x_{in} \cdot (A \odot B)) \tag{14}$$

So the circuit can be implemented as follows:

$$D_A = x_{in} \oplus A \tag{15}$$

$$D_B = (\overline{x_{in}} \cdot B) + (x_{in} \cdot (A \odot B)) \tag{16}$$

Question 4

For a serial 2s complementer, the circuit can be implemented as follows:
S_0 : We have not seen a 1 yet copy the input to the output

S_1 : We have seen a 1, invert the output.

This is for LSB sent first approach.

The finite state machine can be implemented as follows:

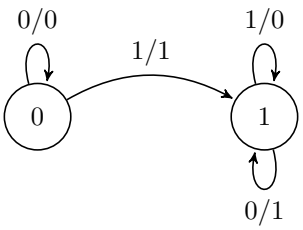

Figure 8: State Diagram

The state table can be written as follows:

Table 7: State Table

Present State		Next State	
Q	x	Q(t+1)	y
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	0

We are using D latch for this implementation. The D input can be written as follows:
It can be seen from the table that the D input it:

$$D = Q + x \tag{17}$$

And it can be seen that the y is the xor of the Q and x:

$$y = Q \oplus x \tag{18}$$

The circuit can be implemented as follows:

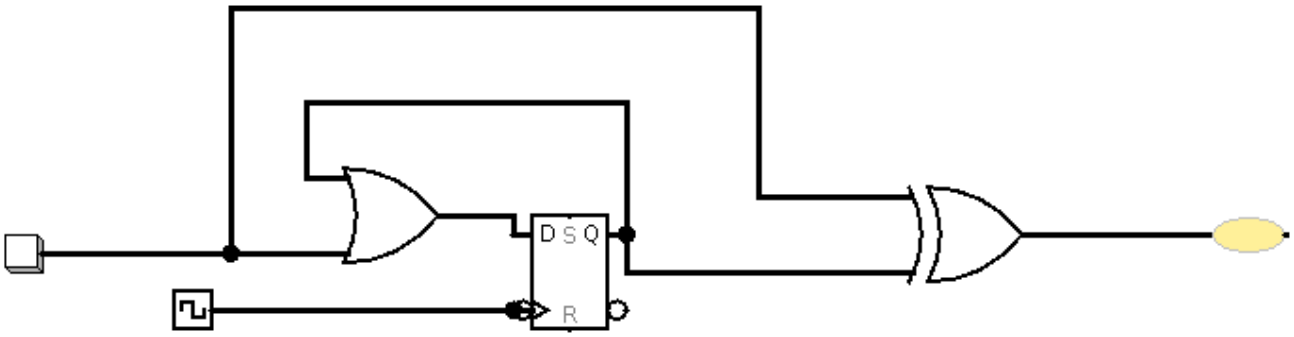

Figure 9: Logic Diagram for the State Machine

Question 5

The requested state transition table is given in Table 8.

Table 8: State Transition Table for Sequential Circuit

Current State		Inputs		Next State	
A	B	E	F	A*	B*
0	0	0	0	0	0
0	0	0	1	0	0
0	0	1	0	1	1
0	0	1	1	0	1
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	0
0	1	1	1	1	0
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	0	1
1	0	1	1	1	1
1	1	0	0	1	1
1	1	0	1	1	1
1	1	1	0	1	0
1	1	1	1	0	0

The state machine diagram can be drawn as follows:

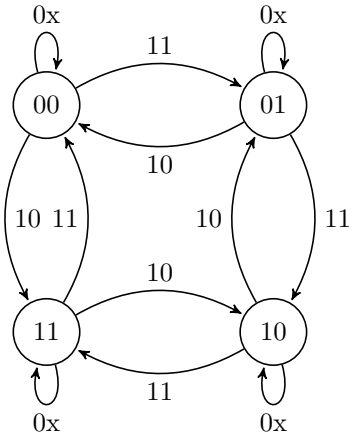

Figure 10: State Diagram

The E signal allows signal to change if its 0 F signal does not matter and the state does not change.

F = 1 counts up the state and F = 0 counts down the state.

The JK flip-flop equations can be written as follows:

$$Q(t+1) = J \cdot \overline{Q} + \overline{K} \cdot Q \tag{19}$$

And the excitation table for the JK flip-flop can be written as follows:

Table 9: JK Flip-Flop Excitation Table

Present State	Next State	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

We can add 4 columns for J_A, J_B, K_A, K_B

Table 10: State Transition Table for Sequential Circuit

Current State		Inputs		Next State		FF Inputs			
A	B	E	F	A*	B*	J_A	K_A	J_B	K_B
0	0	0	0	0	0	0	X	0	X
0	0	0	1	0	0	0	X	0	X
0	0	1	0	1	1	1	X	1	X
0	0	1	1	0	1	0	X	1	X
0	1	0	0	0	1	0	X	X	0
0	1	0	1	0	1	0	X	X	0
0	1	1	0	0	0	0	X	X	1
0	1	1	1	1	0	1	X	X	1
1	0	0	0	1	0	X	0	0	X
1	0	0	1	1	0	X	0	0	X
1	0	1	0	0	1	X	1	1	X
1	0	1	1	1	1	X	0	1	X
1	1	0	0	1	1	X	0	X	0
1	1	0	1	1	1	X	0	X	0
1	1	1	0	1	0	X	0	X	1
1	1	1	1	0	0	X	1	X	1

For finding the equations we can use kmaps:

AB \ EF	00	01	11	10
00	0	0	0	1
01	0	0	1	0
11	X	X	X	X
10	X	X	X	X

Figure 11: K-map for J_A

AB \ EF	00	01	11	10
00	X	X	X	X
01	X	X	X	X
11	0	0	1	0
10	0	0	0	1

Figure 12: K-map for K_A

AB \ EF	00	01	11	10
00	0	0	1	1
01	X	X	X	X
11	X	X	X	X
10	0	0	1	1

Figure 13: K-map for J_B

AB \ EF	00	01	11	10
00	X	X	X	X
01	0	0	1	1
11	0	0	1	1
10	X	X	X	X

Figure 14: K-map for K_B

The final equations for the flip flops can be obtained from kmaps as follows:

$$J_A = A'B \cdot (BF + B'F') \tag{20}$$

$$K_A = AE \cdot (BF + B'F') \tag{21}$$

$$J_B = E \tag{22}$$

$$K_B = E \tag{23}$$

Part 2

Question 1

The Vhdl code for the requested circuit can be written as follows:

Listing 1: JK_LATCH.vhd

```
1 library IEEE;
2
3 use IEEE.STD_LOGIC_1164.ALL;
4
```

```
entity JK_Latch is
    Port ( J : in  STD_LOGIC;
           K : in  STD_LOGIC;
           EN: in STD_LOGIC;
           Q : out  STD_LOGIC;
           QN : out  STD_LOGIC);

end JK_Latch;


architecture Behavioral of JK_Latch is

    signal state : STD_LOGIC := '0';

begin
    process (J, K, EN)
    begin
        if (EN = '1') then
            if (J = '0' and K = '0') then
                state <= state;
            elsif (J = '0' and K = '1') then
                state <= '0';
            elsif (J = '1' and K = '0') then
                state <= '1';
            elsif (J = '1' and K = '1') then
                state <= not state;
            end if;
        end if;
    end process;

    Q <= state;
    QN <= not state;

end Behavioral;
```

The testbench for the requested circuit can be written as follows:

Listing 2: JK_LATCH_tb.vhd

```
library IEEE;

use IEEE.STD_LOGIC_1164.ALL;

entity JK_Latch_tb is
end JK_Latch_tb;

architecture Behavioral of JK_Latch_tb is

    component JK_Latch
        Port ( J : in  STD_LOGIC;
               K : in  STD_LOGIC;
               EN: in STD_LOGIC;
               Q : out  STD_LOGIC;
               QN : out  STD_LOGIC);
    end component;

    signal JTB : STD_LOGIC := '0';
    signal KTB : STD_LOGIC := '0';
    signal ENTB : STD_LOGIC := '0';
    signal QTB : STD_LOGIC;
    signal QNTB : STD_LOGIC;

```

```
24 begin
25
26     UUT: JK_Latch port map (J => JTB, K => KTB, EN => ENTB, Q => QTB, Q
27
28     stim_proc: process
29     begin
30
31         -- Initial state
32
33         JTB <= '0';
34         KTB <= '0';
35         ENTB <= '0';
36         wait for 100 ns;
37
38
39         -- No change expected
40         ENTB <= '1';
41         JTB <= '0';
42         KTB <= '0';
43         wait for 100 ns;
44
45         -- Q = 1
46         ENTB <= '1';
47         JTB <= '1';
48         KTB <= '0';
49         wait for 100 ns;
50
51         -- Q = 0
52         ENTB <= '1';
53         JTB <= '0';
54         KTB <= '1';
55         wait for 100 ns;
56
57
58         -- Toggle
59
60         ENTB <= '1';
61         JTB <= '1';
62         KTB <= '1';
63         wait for 100 ns;
64
65         wait;
66     end process;
67
68 end Behavioral;
```

The RTL schematic for the requested circuit can be seen in Figure 15.

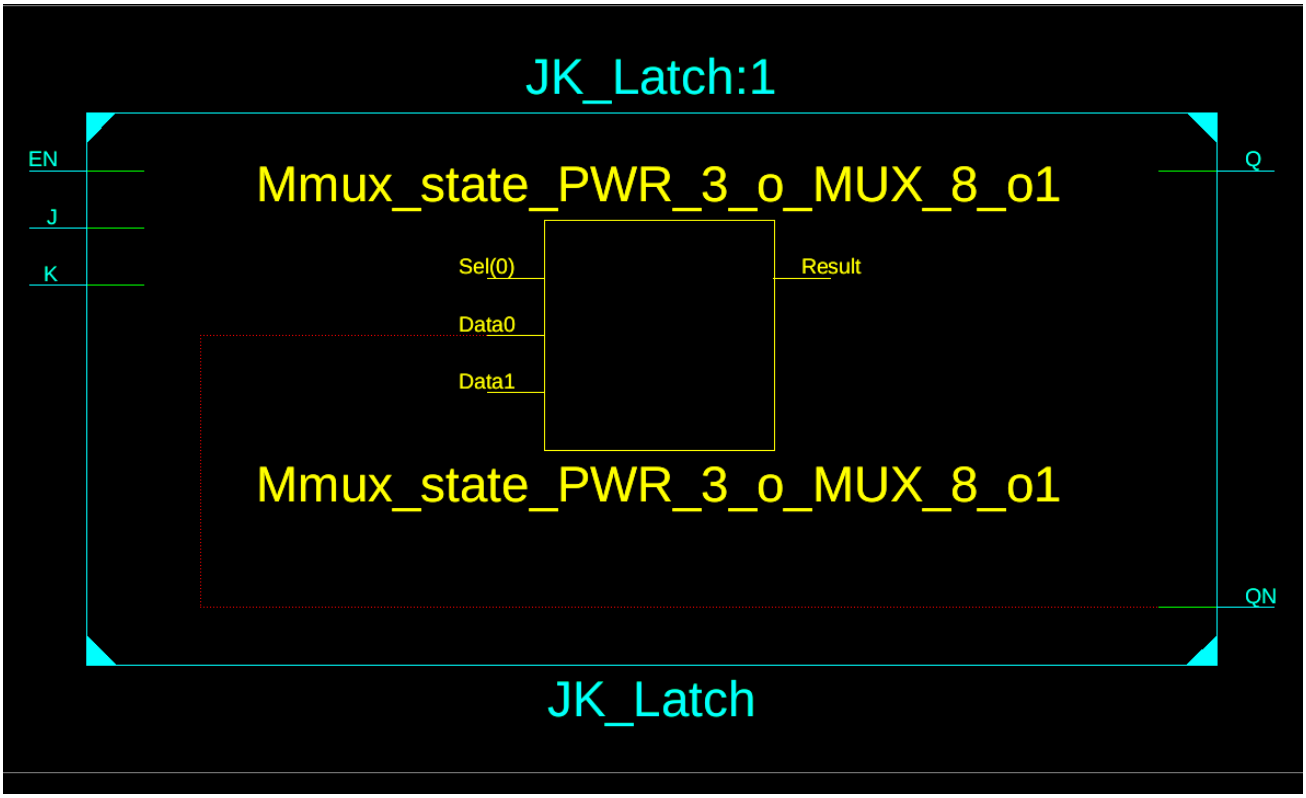

Figure 15: RTL Schematic for JK Latch

The simulation results can be seen in Figure 16.

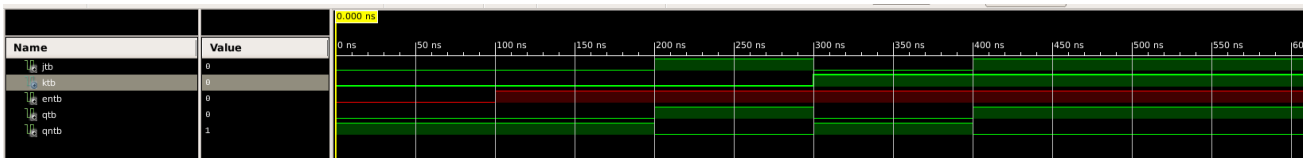

Figure 16: Simulation Results for JK Latch

Question 2

The Vhdl code for the requested circuit can be written as follows:

Listing 3: JK_FF.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity JK_FF is
    Port (
        J : in STD_LOGIC;
        K : in STD_LOGIC;
        CLK : in STD_LOGIC;
        Q : out STD_LOGIC;
        QN : out STD_LOGIC
    );
end JK_FF;

architecture Behavioral of JK_FF is
    signal state : STD_LOGIC := '0';
begin

```

```
18      process(CLK)
19      begin
20          if rising_edge(CLK) then
21              if J = '0' and K = '0' then
22                  state <= state;
23              elsif J = '0' and K = '1' then
24                  state <= '0';
25              elsif J = '1' and K = '0' then
26                  state <= '1';
27              elsif J = '1' and K = '1' then
28                  state <= not state;
29              end if;
30          end if;
31      end process;
32
33      Q <= state;
34      QN <= not state;
35
36  end Behavioral;
```

The testbench for the requested circuit can be written as follows:

Listing 4: JK_FF_tb.vhd

```
1   library IEEE;
2   use IEEE.STD_LOGIC_1164.ALL;
3
4   entity JK_FF_TB is
5   end JK_FF_TB;
6
7   architecture Behavioral of JK_FF_TB is
8
9       component JK_FF
10          Port ( J : in  STD_LOGIC;
11                 K : in  STD_LOGIC;
12                 CLK: in STD_LOGIC;
13                 Q : out  STD_LOGIC;
14                 QN : out  STD_LOGIC);
15      end component;
16
17      signal JTB : STD_LOGIC := '0';
18      signal KTB : STD_LOGIC := '0';
19      signal CLKTB : STD_LOGIC := '0';
20      signal QTB : STD_LOGIC;
21      signal QNTB : STD_LOGIC;
22
23  begin
24
25      UUT: JK_FF port map (J => JTB, K => KTB, CLK => CLKTB, Q => QTB, QN
26
27      clock : process
28      begin
29          CLKTB <= '0';
30          wait for 5 ns;
31          CLKTB <= '1';
32          wait for 5 ns;
33      end process;
34
35      stim_proc: process
36      begin
37          JTB <= '0';
38          KTB <= '0';
```

```
39          wait for 100 ns;
40
41          JTB <= '0';
42          KTB <= '0';
43          wait for 100 ns;
44
45          JTB <= '1';
46          KTB <= '0';
47          wait for 100 ns;
48
49          JTB <= '0';
50          KTB <= '1';
51          wait for 100 ns;
52
53          JTB <= '1';
54          KTB <= '1';
55          wait for 100 ns;
56
57          wait;
58      end process;
59
60  end Behavioral;
```

The RTL schematic for the requested circuit can be seen in Figure 17.

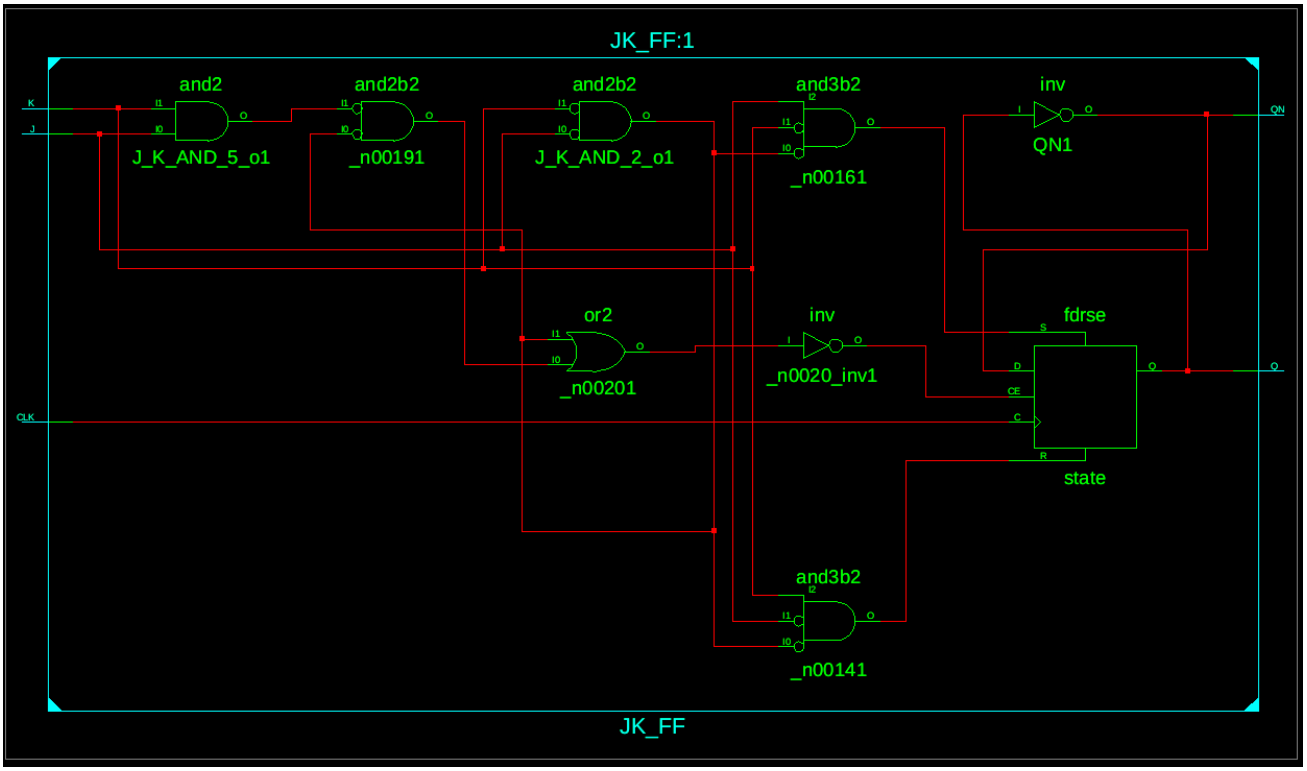

Figure 17: RTL Schematic for JK Flip Flop

The simulation results can be seen in Figure 18.

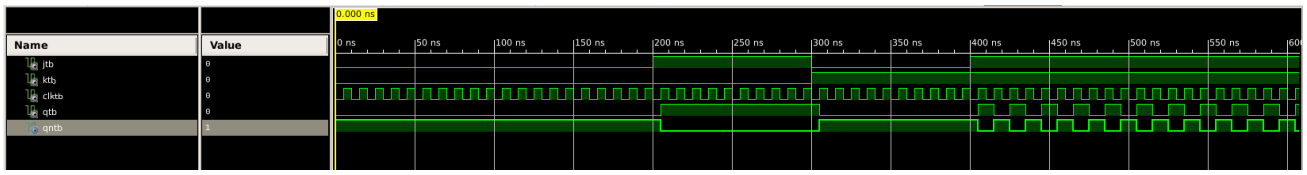

Figure 18: Simulation Results for JK Flip Flop

Question 3

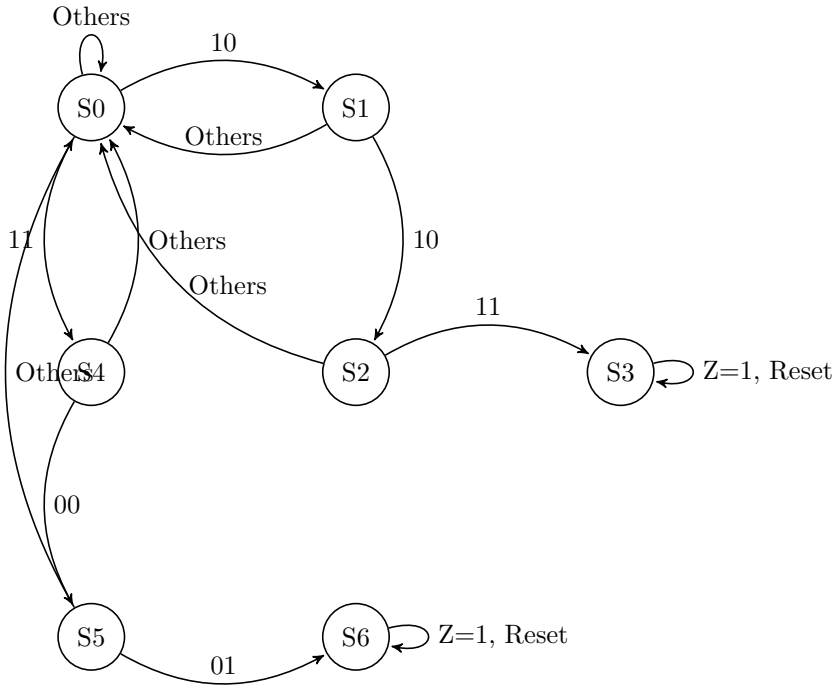

Figure 19: State Diagram for Pattern Detector

1. **S0**: Initial state (waiting for a valid start of a pattern).
2. **S1**: ‘10‘ detected (beginning of Pattern 1).
3. **S2**: ‘10-10‘ detected (middle of Pattern 1).
4. **S3**: ‘10-10-11‘ detected (Pattern 1 complete, $Z = 1$).
5. **S4**: ‘11‘ detected (beginning of Pattern 2).
6. **S5**: ‘11-00‘ detected (middle of Pattern 2).
7. **S6**: ‘11-00-01‘ detected (Pattern 2 complete, $Z = 1$).

The VHDL code for the requested circuit can be written as follows:

Listing 5: Pattern_Detector.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Pattern_Detector is
    Port (
        CLK : in STD_LOGIC;
        DATA_IN : in STD_LOGIC_VECTOR(1 downto 0);
        Z : out STD_LOGIC
    );
end Pattern_Detector;

architecture Behavioral of Pattern_Detector is
    type state_type is (S0, S1, S2, S3, S4, S5, S6);
    signal STATE_CURRENT, STATE_NEXT : state_type := S0;
    signal Z_internal : STD_LOGIC := '0';
begin
    process (CLK)
    begin
        if rising_edge(CLK) then
            STATE_CURRENT <= STATE_NEXT;
```

```
21              Z <= Z_internal;
22          end if;
23      end process;
24
25      process (STATE_CURRENT, DATA_IN)
26      begin
27          STATE_NEXT <= S0;
28          Z_internal <= '0';
29
30          case STATE_CURRENT is
31              when S0 =>
32                  if DATA_IN = "10" then
33                      STATE_NEXT <= S1;
34                  elsif DATA_IN = "11" then
35                      STATE_NEXT <= S4;
36                  end if;
37
38              when S1 =>
39                  if DATA_IN = "10" then
40                      STATE_NEXT <= S2;
41                  else
42                      STATE_NEXT <= S0;
43                  end if;
44
45              when S2 =>
46                  if DATA_IN = "11" then
47                      STATE_NEXT <= S3;
48                      Z_internal <= '1';
49                  else
50                      STATE_NEXT <= S0;
51                  end if;
52
53              when S3 =>
54                  STATE_NEXT <= S0;
55
56              when S4 =>
57                  if DATA_IN = "00" then
58                      STATE_NEXT <= S5;
59                  else
60                      STATE_NEXT <= S0;
61                  end if;
62
63              when S5 =>
64                  if DATA_IN = "01" then
65                      STATE_NEXT <= S6;
66                      Z_internal <= '1';
67                  else
68                      STATE_NEXT <= S0;
69                  end if;
70
71              when S6 =>
72                  STATE_NEXT <= S0;
73
74              when others =>
75                  STATE_NEXT <= S0;
76          end case;
77      end process;
78  end Behavioral;
```

The testbench for the requested circuit can be written as follows:

Listing 6: Pattern_Detector_tb.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity Pattern_Detector_tb is
end Pattern_Detector_tb;

architecture Behavioral of Pattern_Detector_tb is
    signal CLK : STD_LOGIC := '0';
    signal DATA_IN : STD_LOGIC_VECTOR(1 downto 0) := "00";
    signal Z : STD_LOGIC;

    component Pattern_Detector is
        Port (
            CLK : in STD_LOGIC;
            DATA_IN : in STD_LOGIC_VECTOR(1 downto 0);
            Z : out STD_LOGIC
        );
    end component;

begin
    uut: Pattern_Detector
        port map (
            CLK => CLK,
            DATA_IN => DATA_IN,
            Z => Z
        );

    clock : process
    begin
        while true loop
            CLK <= '0';
            wait for 5 ns;
            CLK <= '1';
            wait for 5 ns;
        end loop;
    end process;

    process
    begin
        wait for 10 ns;
        DATA_IN <= "00";
        wait for 10 ns;
        DATA_IN <= "10";
        wait for 10 ns;
        DATA_IN <= "10";
        wait for 10 ns;
        DATA_IN <= "11";
        wait for 10 ns;
        DATA_IN <= "01";
        wait for 10 ns;
        DATA_IN <= "10";
        wait for 10 ns;
        DATA_IN <= "11";
        wait for 10 ns;
        DATA_IN <= "00";
        wait for 10 ns;
        DATA_IN <= "01";
        wait for 10 ns;
        DATA_IN <= "00";
        wait for 10 ns;
```

```
61          DATA_IN <= "10";
62          wait for 10 ns;
63          DATA_IN <= "00";
64          wait for 10 ns;
65          DATA_IN <= "10";
66          wait for 10 ns;
67          DATA_IN <= "10";
68          wait for 10 ns;
69          DATA_IN <= "11";
70          wait for 10 ns;
71          DATA_IN <= "10";
72          wait for 10 ns;
73          DATA_IN <= "10";
74          wait for 10 ns;
75          DATA_IN <= "11";
76          wait for 10 ns;
77          wait;
78      end process;
79
80  end Behavioral;
```

The RTL schematic for the requested circuit can be seen in Figure 20.

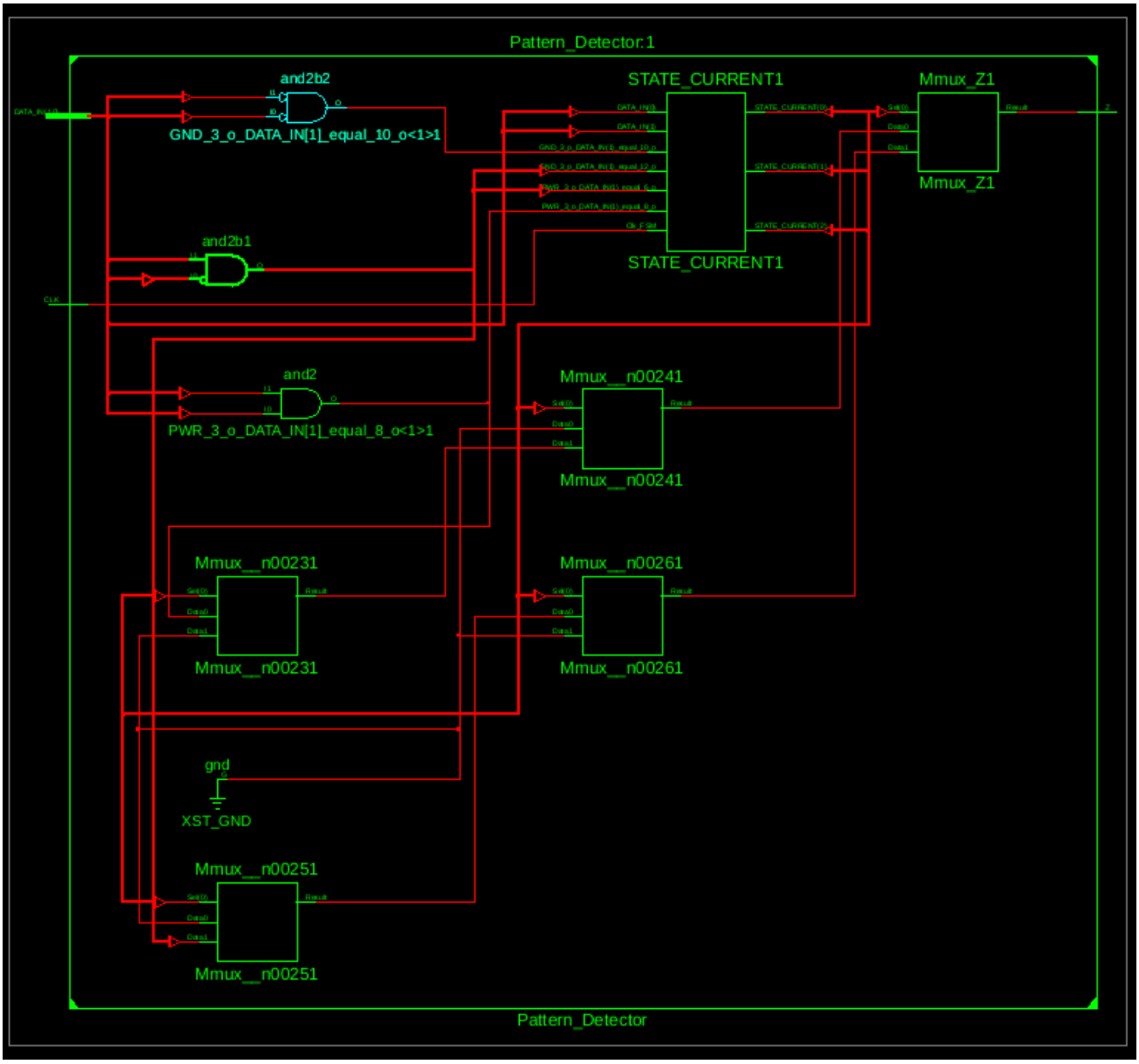

Figure 20: RTL Schematic for Pattern Detector

The simulation results can be seen in Figure 21.

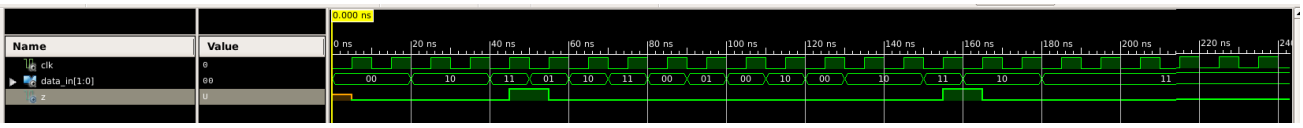

Figure 21: Simulation Results for Pattern Detector

Question 4

Assigning the states as follows:

Table 11: State and Assignment Table

State	$Q_2Q_1Q_0$
S_0	000
S_1	001
S_2	010
S_3	011
S_4	100
S_5	101
S_6	110

Using the excitation table of JK flip-flops, we can find the JK inputs for each state transition as follows:

Table 12: JK Flip-Flop Excitation Table

Present State	Next State	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Current $(Q_2Q_1Q_0)$	**Input** (A_1A_0)	**Next** $(Q_2^+Q_1^+Q_0^+)$	**Z**	(J_2, K_2)	(J_1, K_1)	(J_0, K_0)
000 $(S0)$	10	001 $(S1)$	0	$(0, X)$	$(0, X)$	$(1, X)$
000 $(S0)$	11	100 $(S4)$	0	$(1, X)$	$(0, X)$	$(0, X)$
000 $(S0)$	Others	000 $(S0)$	0	$(0, X)$	$(0, X)$	$(0, X)$
001 $(S1)$	10	010 $(S2)$	0	$(0, X)$	$(1, X)$	$(X, 1)$
001 $(S1)$	Others	000 $(S0)$	0	$(0, X)$	$(0, X)$	$(X, 1)$
010 $(S2)$	11	011 $(S3)$	1	$(0, X)$	$(X, 0)$	$(1, X)$
010 $(S2)$	Others	000 $(S0)$	0	$(0, X)$	$(X, 1)$	$(0, X)$
011 $(S3)$	Any	000 $(S0)$	0	$(0, X)$	$(X, 1)$	$(X, 1)$
100 $(S4)$	00	101 $(S5)$	0	$(X, 0)$	$(0, X)$	$(1, X)$
100 $(S4)$	Others	000 $(S0)$	0	$(X, 1)$	$(0, X)$	$(0, X)$
101 $(S5)$	01	110 $(S6)$	1	$(X, 0)$	$(1, X)$	$(X, 1)$
101 $(S5)$	Others	000 $(S0)$	0	$(X, 1)$	$(0, X)$	$(X, 1)$
110 $(S6)$	Any	000 $(S0)$	0	$(X, 1)$	$(X, 1)$	$(0, X)$

Table 13: State Transition Table with JK Inputs

The expressions were simplified and the results are

$$
\begin{aligned}
J_2 &= \overline{Q_2}\,\overline{Q_1}\,\overline{Q_0}\,A_1\,A_0,\\
K_2 &= (Q_2Q_1) + (Q_2\,\overline{Q_1}\,\overline{Q_0}\,(A_1 + A_0)) + (Q_2\,\overline{Q_1}\,Q_0\,(A_1 + \overline{A_0})),\\
J_1 &= \overline{Q_1}\,Q_0\,\big(\overline{Q_2}\,A_1\,\overline{A_0} + Q_2\,\overline{A_1}\,A_0\big),\\
K_1 &= \overline{Q_2}\,Q_1\,\overline{Q_0}\,(\overline{A_1} + \overline{A_0}) + \overline{Q_2}\,Q_1\,Q_0 + Q_2\,Q_1\,\overline{Q_0},\\
J_0 &= \overline{Q_2}\,\overline{Q_1}\,\overline{Q_0}\,A_1\,\overline{A_0} + \overline{Q_2}\,Q_1\,\overline{Q_0}\,A_1\,A_0 + Q_2\,\overline{Q_1}\,\overline{Q_0}\,\overline{A_1}\,\overline{A_0},\\
K_0 &= \overline{Q_2}\,\overline{Q_1}\,Q_0\,(\overline{A_1} + A_0) + \overline{Q_2}\,Q_1\,Q_0 + Q_2\,\overline{Q_1}\,Q_0,\\
Z &= \overline{Q_2}\,Q_1\,\overline{Q_0}\,A_1\,A_0 + Q_2\,\overline{Q_1}\,Q_0\,\overline{A_1}\,A_0.
\end{aligned}
$$

The vhdl code for the requested circuit can be written as follows:

Listing 7: Pattern_Detector_JK_FF.vhd

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity sequence_detector is
    Port (
        clk     : in  STD_LOGIC;
        A1      : in  STD_LOGIC;
        A0      : in  STD_LOGIC;
        Z       : out STD_LOGIC
    );
end sequence_detector;

architecture Behavioral of sequence_detector is
    component JK_FF
        Port (
            J      : in  STD_LOGIC;
            K      : in  STD_LOGIC;
            CLK    : in  STD_LOGIC;
            Q      : out STD_LOGIC;
            QN     : out STD_LOGIC
        );
    end component;

    signal Q2, Q1, Q0 : STD_LOGIC;
    signal QN2, QN1, QN0 : STD_LOGIC;
    signal J2, K2, J1, K1, J0, K0 : STD_LOGIC;

begin
    J2 <= (not Q2) and (not Q1) and (not Q0) and A1 and A0;

    K2 <= (Q2 and Q1) or
          (Q2 and (not Q1) and (not Q0) and (A1 or A0)) or
          (Q2 and (not Q1) and Q0 and (A1 or (not A0)));

    J1 <= (not Q1) and Q0 and
          ((not Q2 and A1 and (not A0)) or (Q2 and (not A1) and A0));

    K1 <= ((not Q2) and Q1 and (not Q0) and ((not A1) or (not A0))) or
          ((not Q2) and Q1 and Q0) or
          (Q2 and Q1 and (not Q0));

    J0 <= ((not Q2) and (not Q1) and (not Q0) and A1 and (not A0)) or
          ((not Q2) and Q1 and (not Q0) and A1 and A0) or
          (Q2 and (not Q1) and (not Q0) and (not A1) and (not A0));

    K0 <= ((not Q2) and (not Q1) and Q0 and ((not A1) or A0)) or
          ((not Q2) and Q1 and Q0) or
          (Q2 and (not Q1) and Q0);

    Z <= ((not Q2) and Q1 and (not Q0) and A1 and A0) or
         (Q2 and (not Q1) and Q0 and (not A1) and A0);

    FF2: JK_FF port map (
        J => J2,
        K => K2,
        CLK => clk,
        Q => Q2,
        QN => QN2
    );

```

```
61      FF1: JK_FF port map (
62          J => J1,
63          K => K1,
64          CLK => clk,
65          Q => Q1,
66          QN => QN1
67      );
68
69      FF0: JK_FF port map (
70          J => J0,
71          K => K0,
72          CLK => clk,
73          Q => Q0,
74          QN => QN0
75      );
76
77  end Behavioral;
```

The testbench for the requested circuit can be written as follows:

Listing 8: Pattern_Detector_JK_FF_tb.vhd

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3
4  entity sequence_detector_tb is
5  end sequence_detector_tb;
6
7  architecture Behavioral of sequence_detector_tb is
8      signal CLK : STD_LOGIC := '0';
9      signal A1  : STD_LOGIC := '0';
10     signal A0  : STD_LOGIC := '0';
11     signal Z   : STD_LOGIC;
12
13     component sequence_detector is
14         Port (
15             clk : in  STD_LOGIC;
16             A1  : in  STD_LOGIC;
17             A0  : in  STD_LOGIC;
18             Z   : out STD_LOGIC
19         );
20     end component;
21
22 begin
23     uut: sequence_detector
24         port map (
25             clk => CLK,
26             A1  => A1,
27             A0  => A0,
28             Z   => Z
29         );
30
31     clock : process
32     begin
33         while true loop
34             CLK <= '0';
35             wait for 10 ns;
36             CLK <= '1';
37             wait for 10 ns;
38         end loop;
39     end process;
40
```

```
41      stimulus : process
42      begin
43          wait for 20 ns;
44          A1 <= '1'; A0 <= '0'; wait for 20 ns;
45          A1 <= '1'; A0 <= '0'; wait for 20 ns;
46          A1 <= '1'; A0 <= '1'; wait for 20 ns;
47          A1 <= '1'; A0 <= '1'; wait for 20 ns;
48          A1 <= '0'; A0 <= '0'; wait for 20 ns;
49          A1 <= '0'; A0 <= '1'; wait for 20 ns;
50          A1 <= '0'; A0 <= '0'; wait for 20 ns;
51          A1 <= '1'; A0 <= '0'; wait for 20 ns;
52          A1 <= '1'; A0 <= '0'; wait for 20 ns;
53          A1 <= '1'; A0 <= '1'; wait for 20 ns;
54          A1 <= '1'; A0 <= '0'; wait for 20 ns;
55          A1 <= '1'; A0 <= '0'; wait for 20 ns;
56          A1 <= '1'; A0 <= '1'; wait for 20 ns;
57          wait;
58      end process;
59
60  end Behavioral;
```

The RTL schematic for the requested circuit can be seen in Figure 22.

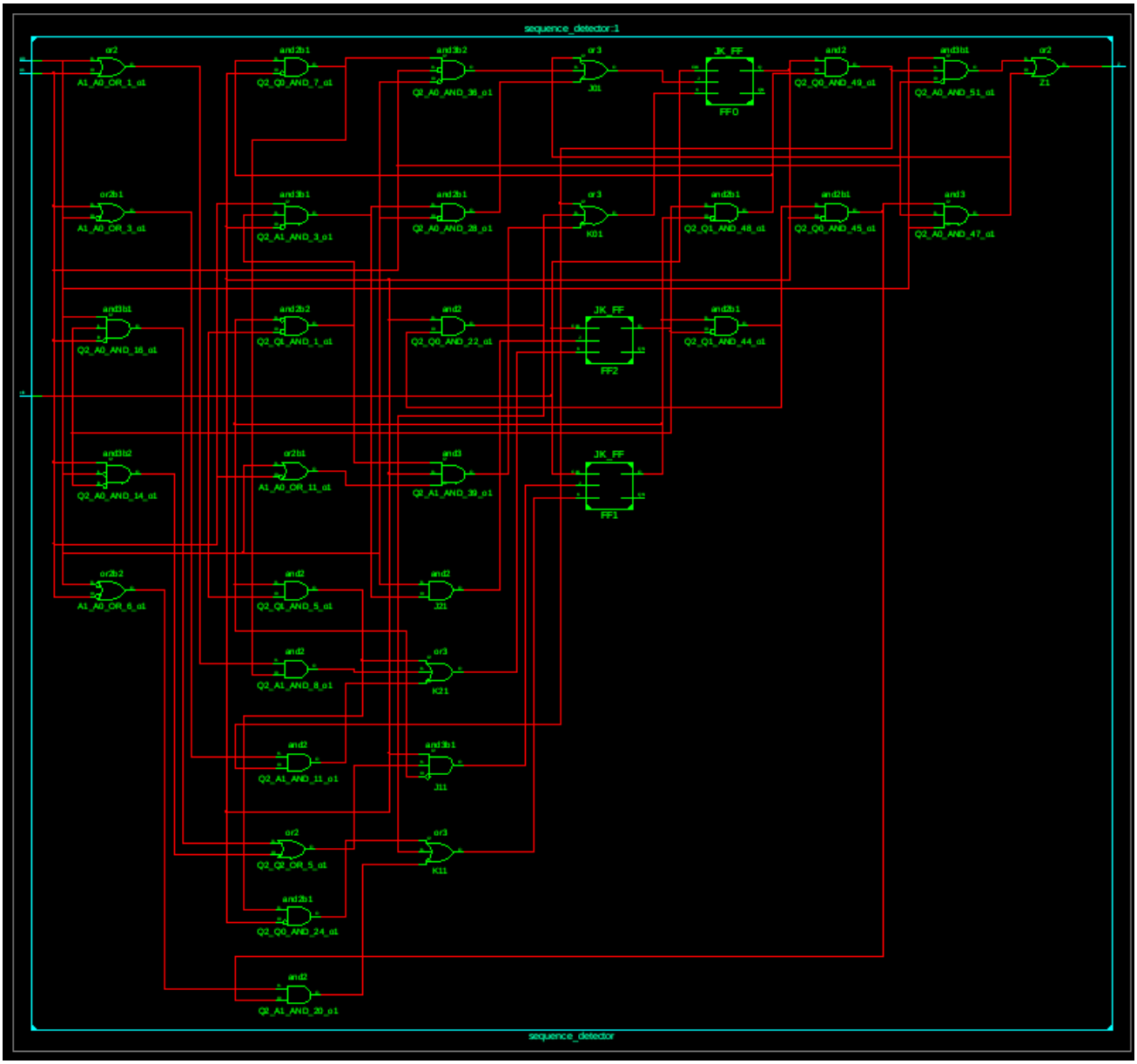

Figure 22: RTL Schematic for Pattern Detector with JK Flip-Flops

The simulation results can be seen in Figure 23.

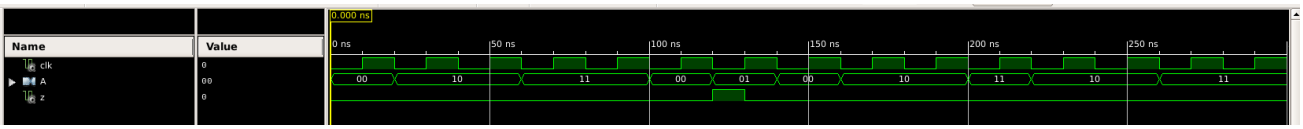

Figure 23: Simulation Results for Pattern Detector with JK Flip-Flops