Hacettepe University

ELE432 Advanced Digital Design

Experiment 1- ALU and FSM Design and Implementation

Mehmet Yücel Sarıtaş
21990451

Lecturer : Prof.Dr. Ali Ziya Alkar

Question 1)

Write a VHDL code that implements an ALU that does arithmetic operations (addition, subtraction, multiplication etc.) and logic operations (and, or, invert, xor etc.) with the use of 4 switches on the board

Implementation of ALU

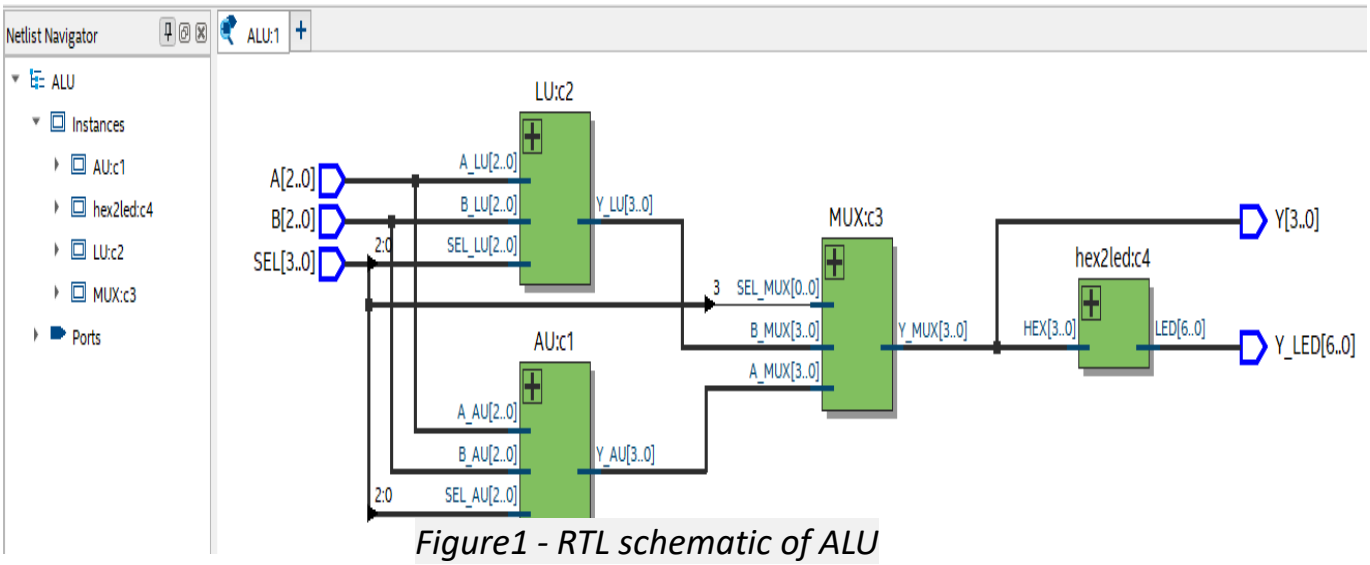


Figure1 - RTL schematic of ALU

Comment: In my design, I have used 3 components that make my ALU implementation more modular. These are as follows; AU (Arithmetic Unit), LU (Logic Unit), and Hex to seven-segment display converter. To see results in both binary and display formats I have ouputted 2 signals one of them is Y[3..0] other one is Y_LED[6..0].

VHDL Code of Design

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

entity ALU is
	port
	(
		A		: in unsigned (2 downto 0);
		B		: in unsigned (2 downto 0);
		SEL	: in unsigned (3 downto 0);
		Y		: out unsigned (3 downto 0);
		Y_LED : out std_logic_vector (6 downto 0)
	);
end ALU;

architecture rtl of ALU is

	component AU is
		port
		(
			A_AU		: in unsigned (2 downto 0);
			B_AU		: in unsigned (2 downto 0);
			SEL_AU	: in unsigned (2 downto 0);
			Y_AU	: out unsigned (3 downto 0)
		);
	end component;

	component LU is
		port
		(
			A_LU		: in unsigned (2 downto 0);
			B_LU		: in unsigned (2 downto 0);
			SEL_LU	: in unsigned (2 downto 0);
			Y_LU	: out unsigned (3 downto 0)
		);
	end component;
```

Code1 - VHDL of ALU design

```
component MUX is
		port
		(
			A_MUX		: in unsigned (3 downto 0);
			B_MUX		: in unsigned (3 downto 0);
			SEL_MUX		: in unsigned (0 downto 0);
			Y_MUX : out unsigned (3 downto 0)
		);
	end component;
	component hex2led IS
		PORT (
			HEX : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
			LED : OUT STD_LOGIC_VECTOR (6 DOWNTO 0)
		);
	end component;

	SIGNAL temp1: unsigned (3 downto 0);
	SIGNAL temp2: unsigned (3 downto 0);
	SIGNAL temp3: unsigned (3 downto 0);
begin
	c1: AU port map(A_AU => A, B_AU => B, SEL_AU => SEL(2 downto 0), Y_AU => temp1);
	c2: LU port map(A_LU => A, B_LU => B, SEL_LU => SEL(2 downto 0), Y_LU => temp2);
	c3: MUX port map(A_MUX => temp1, B_MUX => temp2, SEL_MUX => SEL(3 downto
3),Y_MUX => temp3);
	Y <= temp3;
	c4: hex2led port map(HEX => std_logic_vector(temp3), LED => Y_LED);
end rtl;
```

Code2 - VHDL of ALU design

VHDL Implementation of AU (Arithmetic Unit)

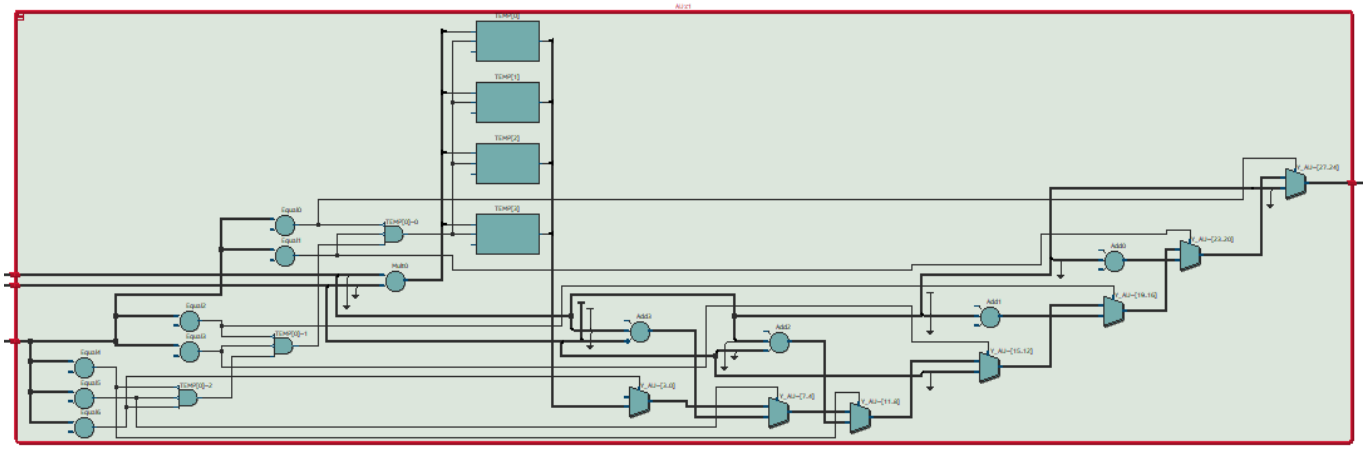


Figure2 - RTL schematic of AU

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

entity AU is
	port
	(
		A_AU		: in unsigned (2 downto 0);
		B_AU		: in unsigned (2 downto 0);
		SEL_AU		: in unsigned (2 downto 0);
		Y_AU	: out unsigned (3 downto 0)
	);
end entity;

architecture rtl of AU is
	SIGNAL TEMP: unsigned (7 DOWNTO 0);
	SIGNAL A_TEMP: unsigned (3 DOWNTO 0);
	SIGNAL B_TEMP: unsigned (3 DOWNTO 0);
begin
	process(A_AU,B_AU,SEL_AU) --a,b,SEL are  sensivity lists)
	begin
		A_TEMP <= "0" & A_AU;
		B_TEMP <= "0" & B_AU;
		-- add if "add_sub" is 1, else subtract
		if (SEL_AU = "000") then
			Y_AU <= A_TEMP;
		elsif (SEL_AU = "001") then
			Y_AU <= A_TEMP + 1;
		elsif (SEL_AU = "010") then
			Y_AU <= A_TEMP - 1;
		elsif (SEL_AU = "011") then
			Y_AU <= B_TEMP;
		elsif (SEL_AU = "100") then
		  Y_AU <= A_TEMP + B_TEMP;
		elsif (SEL_AU = "101") then
			Y_AU <= A_TEMP - B_TEMP;
		elsif (SEL_AU = "110") then
			TEMP <= A_TEMP*B_TEMP;
			Y_AU <= TEMP(3 DOWNTO 0);
		else
			Y_AU <= "0000";
		end if;
	end process;
end rtl;
```

Code3 - VHDL of AU design

VHDL Implementation of LU (Logic Unit)

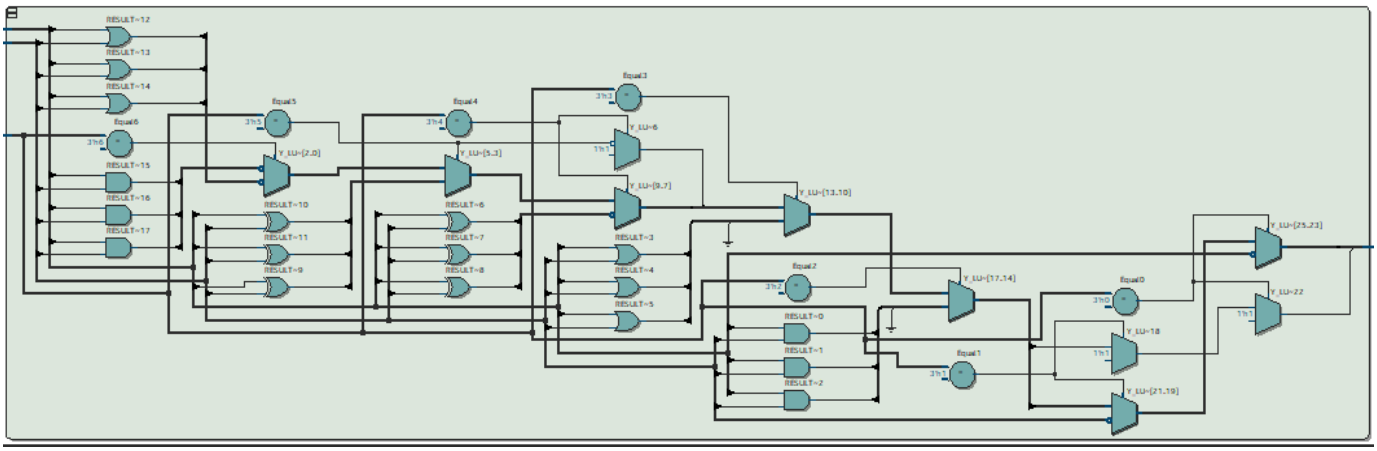


Figure3 - RTL schematic of LU

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

entity LU is

	port
	(
		A_LU		: in unsigned (2 downto 0);
		B_LU		: in unsigned (2 downto 0);
		SEL_LU		: in unsigned (2 downto 0);
		Y_LU	: out unsigned (3 downto 0)
	);

end entity;


architecture rtl of LU is
	SIGNAL TEMP: unsigned (7 DOWNTO 0);
	SIGNAL A_TEMP: unsigned (3 DOWNTO 0);
	SIGNAL B_TEMP: unsigned (3 DOWNTO 0);

begin

	process(A_LU,B_LU,SEL_LU) --a,b,SEL are  sensivity lists(when one of them changed processes triggered)
```

Code4 - VHDL of LU design

```
begin
                A_TEMP <= "0" & A_LU;
                B_TEMP <= "0" & B_LU;
                -- add if "add_sub" is 1, else subtract
                if (SEL_LU = "000") then
                        Y_LU <= NOT A_TEMP;
                elsif (SEL_LU = "001") then
                        Y_LU <= NOT B_TEMP;
                elsif (SEL_LU = "010") then
                        Y_LU <= A_TEMP AND B_TEMP;
                elsif (SEL_LU = "011") then
                        Y_LU <= A_TEMP OR B_TEMP;
                elsif (SEL_LU = "100") then
                        Y_LU <= A_TEMP XNOR B_TEMP;
                elsif (SEL_LU = "101") then
                        Y_LU <= A_TEMP XOR B_TEMP;
                elsif (SEL_LU = "110") then
                        Y_LU <= A_TEMP NOR B_TEMP;
                else
                        Y_LU <= A_TEMP NAND B_TEMP;
                end if;
        end process;
end rtl;
```

Code5 - VHDL of LU design

VHDL Implementation of MUX

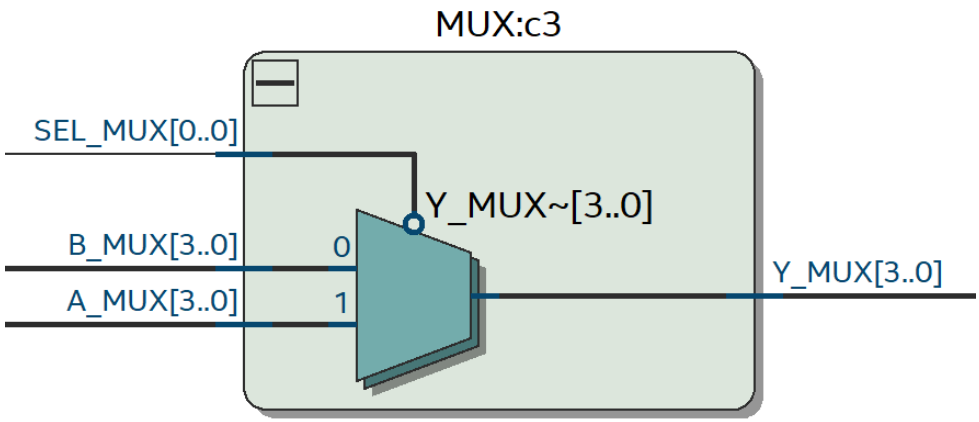


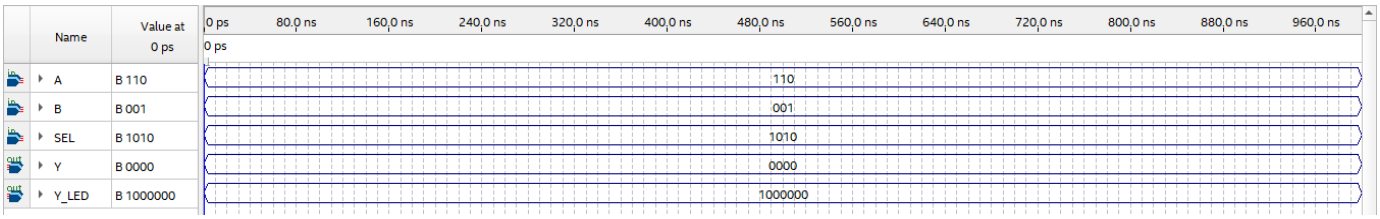
Figure4 - RTL schematic of MUX

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

entity MUX is

	port
	(
		A_MUX		: in unsigned (3 downto 0);
		B_MUX		: in unsigned (3 downto 0);
		SEL_MUX	: in unsigned (0 downto 0);
		Y_MUX		: out unsigned (3 downto 0)
	);

end entity;

architecture rtl of MUX is

begin

	process(A_MUX,B_MUX,SEL_MUX) --a,b,SEL are  sensivity
lists(when one of them changed processes triggered)

	begin

		if (SEL_MUX = "0") then
			Y_MUX <= A_MUX;
		else
			Y_MUX <= B_MUX;
		end if;

	end process;

end rtl;
```

Code6 - VHDL of MUX design

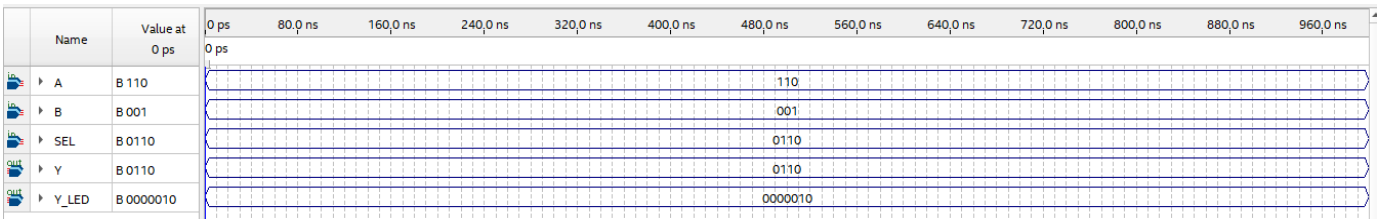
Simulation Results of ALU

A AND B (A = 110, B=001, SEL = 1010)



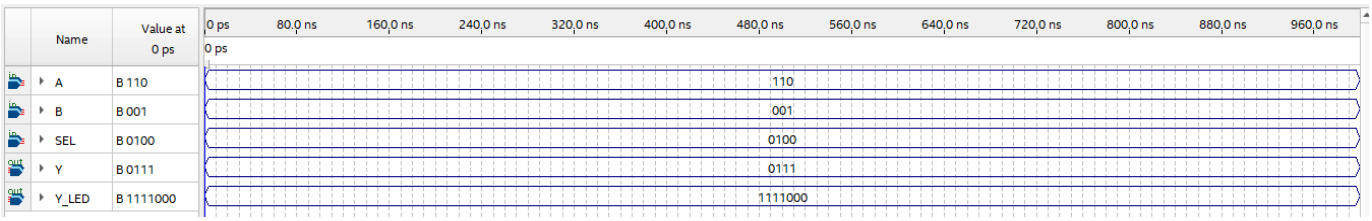
Simulation1- A and B

A * B (A = 110, B=001, SEL = 0110)



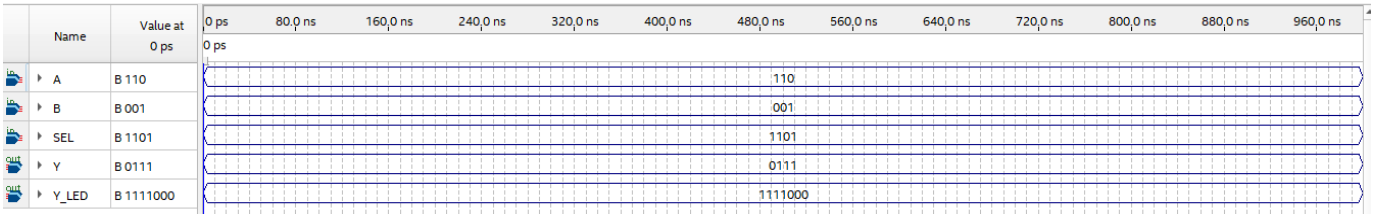
*Simulation2- A * B*

A + B (A = 110, B=001, SEL = 0100)



Simulation3- A + B

A XOR B (A = 110, B=001, SEL = 0100)



Simulation4- A xor B

Pin Planner of ALU Design

All Pins

Named: * Edit:

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate
A[2]	Input	PIN_AD12	3A	B3A_N0	2.5 V (default)		12mA (default)	
A[1]	Input	PIN_AD11	3A	B3A_N0	2.5 V (default)		12mA (default)	
A[0]	Input	PIN_AF10	3A	B3A_N0	2.5 V (default)		12mA (default)	
B[2]	Input	PIN_AF9	3A	B3A_N0	2.5 V (default)		12mA (default)	
B[1]	Input	PIN_AC12	3A	B3A_N0	2.5 V (default)		12mA (default)	
B[0]	Input	PIN_AB12	3A	B3A_N0	2.5 V (default)		12mA (default)	
SEL[3]	Input	PIN_AE12	3A	B3A_N0	2.5 V (default)		12mA (default)	
SEL[2]	Input	PIN_AD10	3A	B3A_N0	2.5 V (default)		12mA (default)	
SEL[1]	Input	PIN_AC9	3A	B3A_N0	2.5 V (default)		12mA (default)	
SEL[0]	Input	PIN_AE11	3A	B3A_N0	2.5 V (default)		12mA (default)	
Y[3]	Output	PIN_V18	4A	B4A_N0	2.5 V (default)		12mA (default)	1 (default)
Y[2]	Output	PIN_V17	4A	B4A_N0	2.5 V (default)		12mA (default)	1 (default)
Y[1]	Output	PIN_W16	4A	B4A_N0	2.5 V (default)		12mA (default)	1 (default)
Y[0]	Output	PIN_V16	4A	B4A_N0	2.5 V (default)		12mA (default)	1 (default)
Y_LED[6]	Output	PIN_AH28	5A	B5A_N0	2.5 V (default)		12mA (default)	1 (default)
Y_LED[5]	Output	PIN_AG28	5A	B5A_N0	2.5 V (default)		12mA (default)	1 (default)
Y_LED[4]	Output	PIN_AF28	5A	B5A_N0	2.5 V (default)		12mA (default)	1 (default)
Y_LED[3]	Output	PIN_AG27	5A	B5A_N0	2.5 V (default)		12mA (default)	1 (default)
Y_LED[2]	Output	PIN_AE28	5A	B5A_N0	2.5 V (default)		12mA (default)	1 (default)
Y_LED[1]	Output	PIN_AE27	5A	B5A_N0	2.5 V (default)		12mA (default)	1 (default)
Y_LED[0]	Output	PIN_AE26	5A	B5A_N0	2.5 V (default)		12mA (default)	1 (default)

I have assigned my pins as such, ”SW9-SW8-SW7-SW6” select the operation, ”SW5-SW4- SW3” inputs A, ”SW2-SW1-SW0” inputs B. ”LED3- LED2- LED1- LED0 ” outputs Y, ” HEX6- HEX5- HEX4- HEX3- HEX2- HEX1- HEX0” outputs Y_LED for seven segment display.

Question 2)

Write a VHDL code that implements a BCD counter.

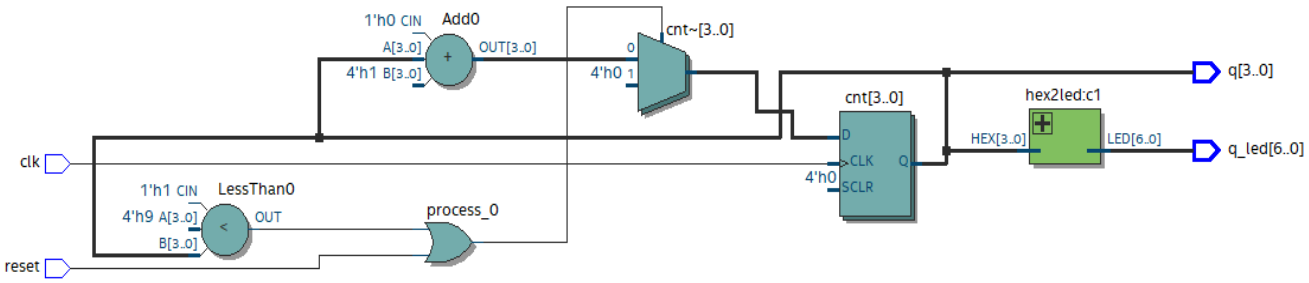


Figure5 - RTL schematic of BCD counter

VHDL Implementation of BCD Counter

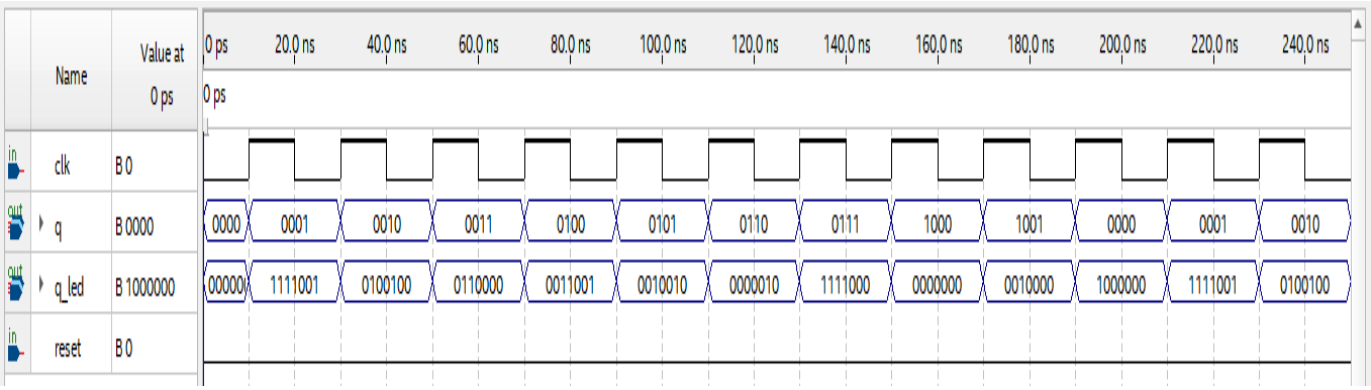
```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity BCD_Counter is
	generic
	(
		MIN_COUNT : natural := 0;
		MAX_COUNT : natural := 9);
	port
	(
		clk	: in std_logic;
		reset	: in std_logic;
		q		: out integer range MIN_COUNT to MAX_COUNT;
		q_led	: out std_logic_vector (6 DOWNTO 0));

end entity;

architecture rtl of BCD_Counter is

	component hex2led IS
		PORT (
			HEX : IN STD_LOGIC_VECTOR (3 DOWNTO 0);
			LED : OUT STD_LOGIC_VECTOR (6 DOWNTO 0)
		);
	END component;
```

Code7 - VHDL of BCD Counter

```
        signal cnt : integer range MIN_COUNT to MAX_COUNT;
        signal temp : std_logic_vector (3 downto 0);

begin
        temp <= std_logic_vector(to_unsigned(cnt, temp'length));
        c1: hex2led port map(HEX => temp, LED => q_led);

        process (clk)

        begin
                if (rising_edge(clk)) then

                        if reset = '1' or cnt >= 9 then
                                -- Reset the counter to 0
                                cnt <= 0;
                        else
                                -- Increment the counter if counting is enabled
                                cnt <= cnt + 1;
                        end if;

                end if;

                -- Output the current count
                q <= cnt;

        end process;

end rtl;
```

Code8 - VHDL of BCD Counter

Simulation Results of BCD Counter



Simulation5- BCD Counter

Pin Planner of BCD Counter

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate
clk	Input	PIN_AA15	3B	B3B_N0	2.5 V (default)		12mA (default)	
q[3]	Output	PIN_V18	4A	B4A_N0	2.5 V (default)		12mA (default)	1 (default)
q[2]	Output	PIN_V17	4A	B4A_N0	2.5 V (default)		12mA (default)	1 (default)
q[1]	Output	PIN_W16	4A	B4A_N0	2.5 V (default)		12mA (default)	1 (default)
q[0]	Output	PIN_V16	4A	B4A_N0	2.5 V (default)		12mA (default)	1 (default)
q_led[6]	Output	PIN_AH28	5A	B5A_N0	2.5 V (default)		12mA (default)	1 (default)
q_led[5]	Output	PIN_AG28	5A	B5A_N0	2.5 V (default)		12mA (default)	1 (default)
q_led[4]	Output	PIN_AF28	5A	B5A_N0	2.5 V (default)		12mA (default)	1 (default)
q_led[3]	Output	PIN_AG27	5A	B5A_N0	2.5 V (default)		12mA (default)	1 (default)
q_led[2]	Output	PIN_AE28	5A	B5A_N0	2.5 V (default)		12mA (default)	1 (default)
q_led[1]	Output	PIN_AE27	5A	B5A_N0	2.5 V (default)		12mA (default)	1 (default)
q_led[0]	Output	PIN_AE26	5A	B5A_N0	2.5 V (default)		12mA (default)	1 (default)
reset	Input	PIN_AA14	3B	B3B_N0	2.5 V (default)		12mA (default)	

Question 3)

Write a VHDL code that implements a sequence detector that detects the input ”101”. Overlaps must also be considered, that is, if a sequence of input ”01010101100” occurs, than the output should remain logic high (”1”) for three consecutive clock cycles.

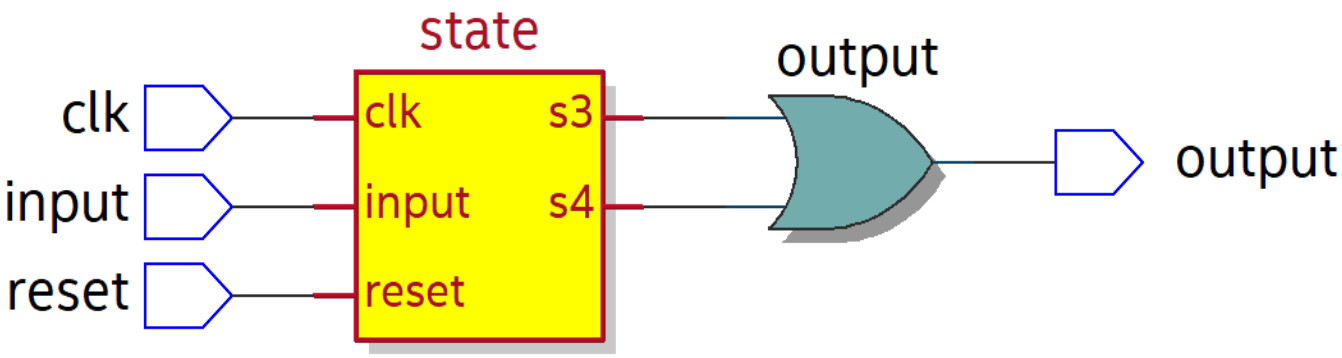


Figure6 - RTL schematic of Sequence Detector

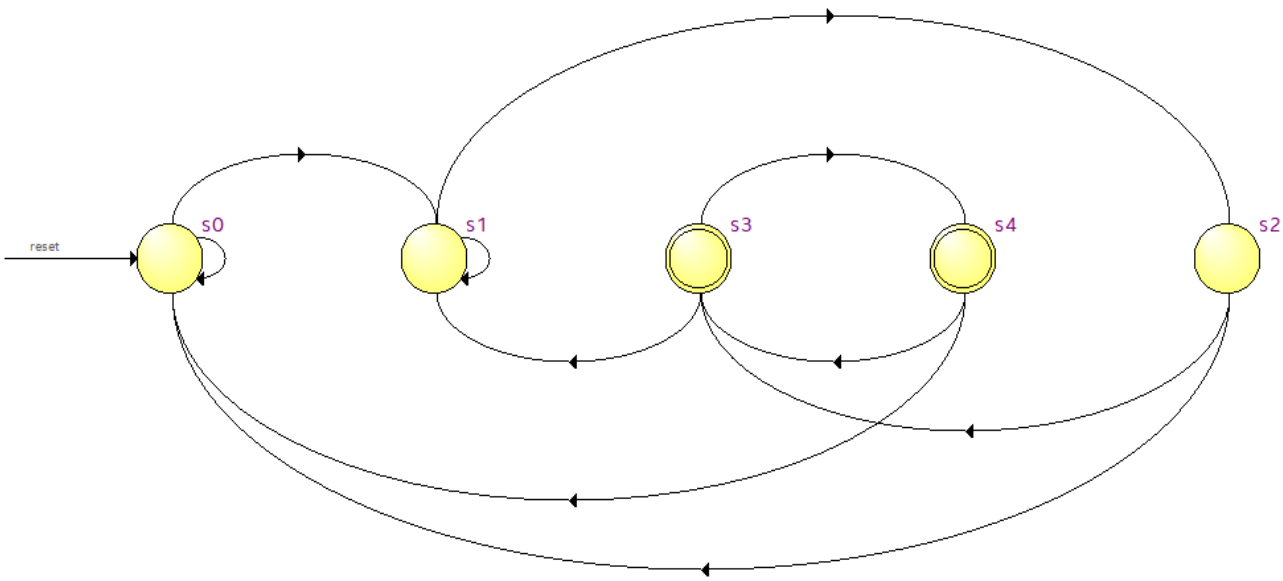


Figure7 – State Machine Viewer of Sequence Detector

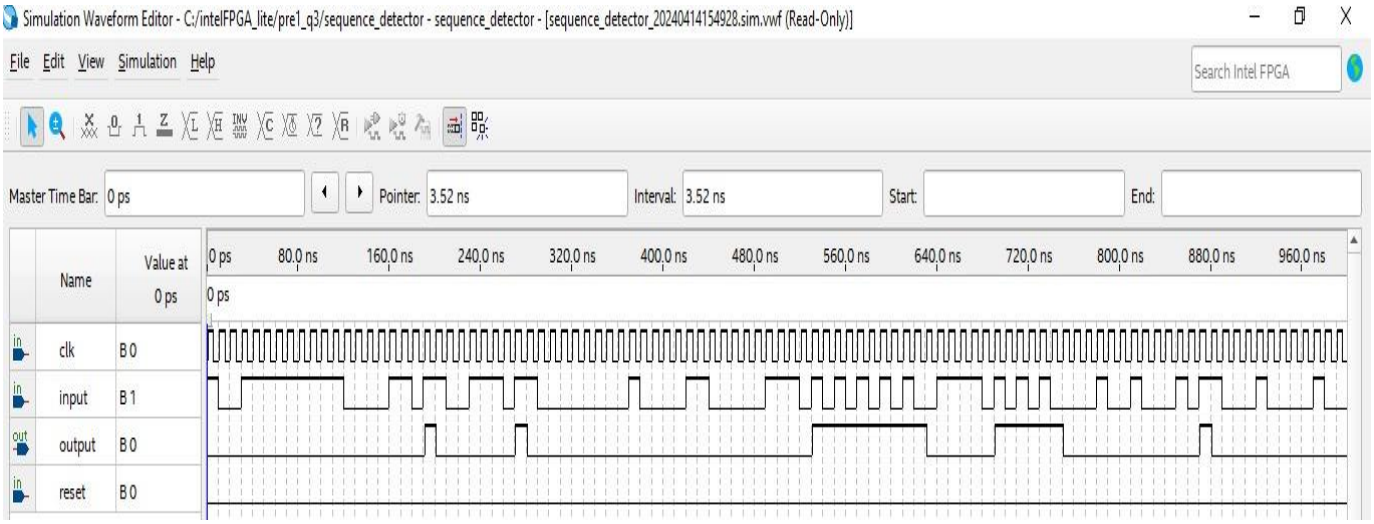
```
library ieee;
use ieee.std_logic_1164.all;

entity sequence_detector is

	port(
		clk		: in	std_logic;
		input	: in	std_logic;
		reset	: in	std_logic;
		output	: out	std_logic
	);

end entity;

architecture rtl of sequence_detector is

	-- Build an enumerated type for the state machine
	type state_type is (s0, s1, s2, s3, s4);

	-- Register to hold the current state
	signal state	: state_type;

begin

	-- Logic to advance to the next state
	process (clk, reset)
	begin

		if reset = '1' then
			state <= s0;

		elsif (rising_edge(clk)) then
			case state is
				when s0=>
					if input = '1' then
						state <= s1;
					else
						state <= s0;
					end if;
```

Code9 - VHDL of Sequence Detector

```
                        when s1=>
                                if input = '1' then
                                        state <= s1;
                                else
                                        state <= s2;
                                end if;
                        when s2=>
                                if input = '1' then
                                        state <= s3;
                                else
                                        state <= s0;
                                end if;
                        when s3 =>
                                if input = '1' then
                                        state <= s1;
                                else
                                        state <= s4;
                                end if;
                        when s4 =>
                                if input = '1' then
                                        state <= s3;
                                else
                                        state <= s0;
                                end if;
                end case;
        end if;
end process;

-- Output depends solely on the current state
process (state)
begin
        case state is
                when s0 =>
                        output <= '0';
                when s1 =>
                        output <= '0';
                when s2 =>
                        output <= '0';
                when s3 =>
                        output <= '1';
                when s4 =>
                        output <= '1';
        end case;
end process;
end rtl;
```

Code10 - VHDL of Sequence Detector

Simulation Results of Sequence Detector



Simulation6- Sequence Detector

Pin Planner of Sequence Detector Design

Node Name	Direction	Location	I/O Bank	VREF Group	I/O Standard	Reserved	Current Strength	Slew Rate
clk	Input	PIN_AA15	3B	B3B_N0	2.5 V (default)		12mA (default)	
input	Input	PIN_AA14	3B	B3B_N0	2.5 V (default)		12mA (default)	
output	Output	PIN_V16	4A	B4A_N0	2.5 V (default)		12mA (default)	1 (default)
reset	Input	PIN_W15	3B	B3B_N0	2.5 V (default)		12mA (default)	

Comment: I have assigned my pins as such, ”KEY0” for input, ”KEY1” for clock, and ”Key2” for reset. Have written my code with respec to the rising edge of the clock. Since my clock is linked to the ”KEY1” when I release the button It will check the input and go to the related state.