Course Name:	**Hardware Description Language Lab (2UXL401)**	**Semester:**	**IV**
Date of Performance:	**20 / 04 / 2021**	**Batch No:**	**B2**
Faculty Name:	**Prof. Bhargavi Kaslikar**	**Roll No:**	**1912060**
Faculty Sign & Date:		**Grade/Marks:**	

Experiment No: 6

Title: FSM implementation: Sequence detection

Aim and Objective of the Experiment:

Write a VHDL code for implementing a Moore type, non-overlapping sequence detector which detects "10101" sequence and gives output high. Write a test bench to verify your results.

To study FSM implementation in VHDL and to understand use of test bench for simulation.

COs to be achieved:

CO 2: Test a VHDL code and verify the circuit model.
CO 3: Synthesize and Implement the designed circuits on CPLD/ FPGA.

Work to be done

Upload VHDL codes for sequence generator FSM. Also upload test bench and simulation for the same.

Post Lab Subjective/Objective type Questions:

Upload Answer of following question before coming to next laboratory.

Q1. What changes will you make in VHDL code if the same sequence generator is mealy overlapping type?

Q2. Examine the following VHDL code and complete the following entity Problem
Port (X, CLK : in
bit; Z1, Z2 : out
bit);
end Problem;
architecture Table of Problem is
signal State, Nextstate: integer range 0 to 3
:=0; begin

```
        process(State,
        X) begin
        case State
        is when 0
        =>
        if X = „0“ then Z1 <= „1“; Z2 <= „0“; Nextstate < =
        0; else Z1 < =“0“; Z2 <=“0“;Nextstate < = 1; end if;

        when 1 =>
if X = „0“ then Z1 <=„1“; Z2 <= „1“; Nextstate < = 1;
else Z1 < =„0“; Z2 <=“1“; Nextstate < = 2; end if;

when 2 =>
if X = „0“ then Z1 <=„0“; Z2 <= „1“; Nextstate < = 2;
else Z1 < =„0“; Z2 <=“1“; Nextstate < = 3; end if;

when 3 =>
if X = „0“ then Z1 <=„0“; Z2 <= „0“; Nextstate < = 0;
else Z1 < =„1“; Z2 <=“0“; Nextstate < = 1; end if;  end
case;
end process;

process(CLK)
begin
if CLK“event and CLK =“1“ then
State <= Nextstate;
end if;
end process;
end Table;
```

(a) **Draw a block diagram of the circuit implemented by this code**
(b) Write the state table that is implemented by this code
(c) Write the type of state machine

Sequence given: 10011 - Overlapping

Mealy Machine;

Main Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity seq_detect_naik is
	port(
		clk : in std_logic;
		reset : in std_logic;
		x : in std_logic;
		output : out std_logic
	);
end seq_detect_naik;
-- 10011 Overlapping Mealy
architecture seq_detect_naik_arch of seq_detect_naik is
	type state is (rst, A, B, C, D);
	signal present_state, next_state : state;
	begin
	synchronous_process: process (clk)
		begin
			if rising_edge(clk) then
				if (reset = '1') then
					present_state <= rst;
				else
					present_state <= next_state;
				end if;
			end if;
	end process;

	output_decoder : process(present_state, x)
	begin
		--next_state <= rst;
		output <= '0';
		case (present_state) is
		when rst =>
			if (x = '1') then
				next_state <= A;
			else
				next_state <= rst;
```

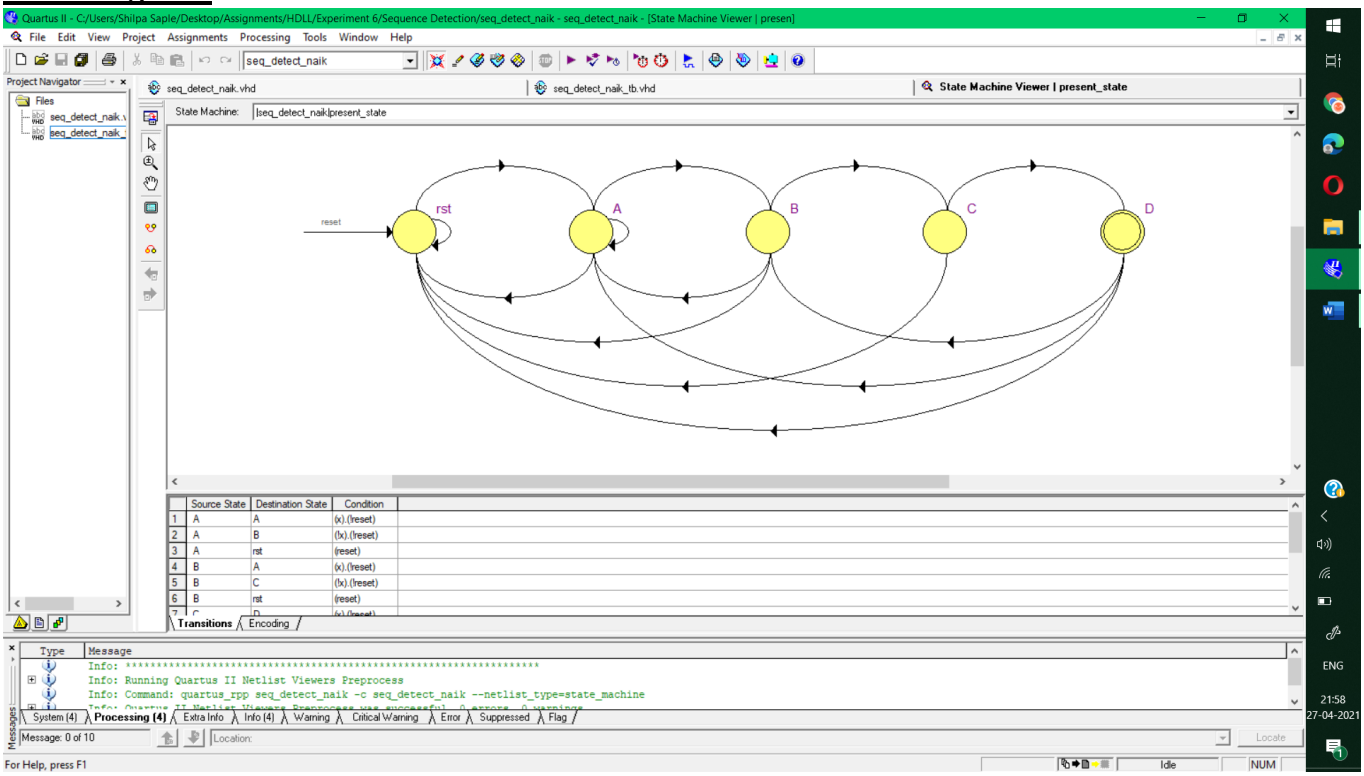
```
                end if;

        when A =>
                if (x = '1') then
                        next_state <= A;
                else
                        next_state <= B;
                end if;

        when B =>
                if (x = '1') then
                        next_state <= A;
                else
                        next_state <= C;
                end if;

        when C =>
                if (x= '1') then
                        next_state <= D;
                else
                        next_state <= rst;
                end if;

        when D =>
                output <= '1';
                if (x= '1') then
                        next_state <= A;
                        --output <= '1';
                else
                        next_state <= B;
                end if;
        --when others => next_state <= rst;
        end case;
        end process;


end seq_detect_naik_arch;
```

Testbench:

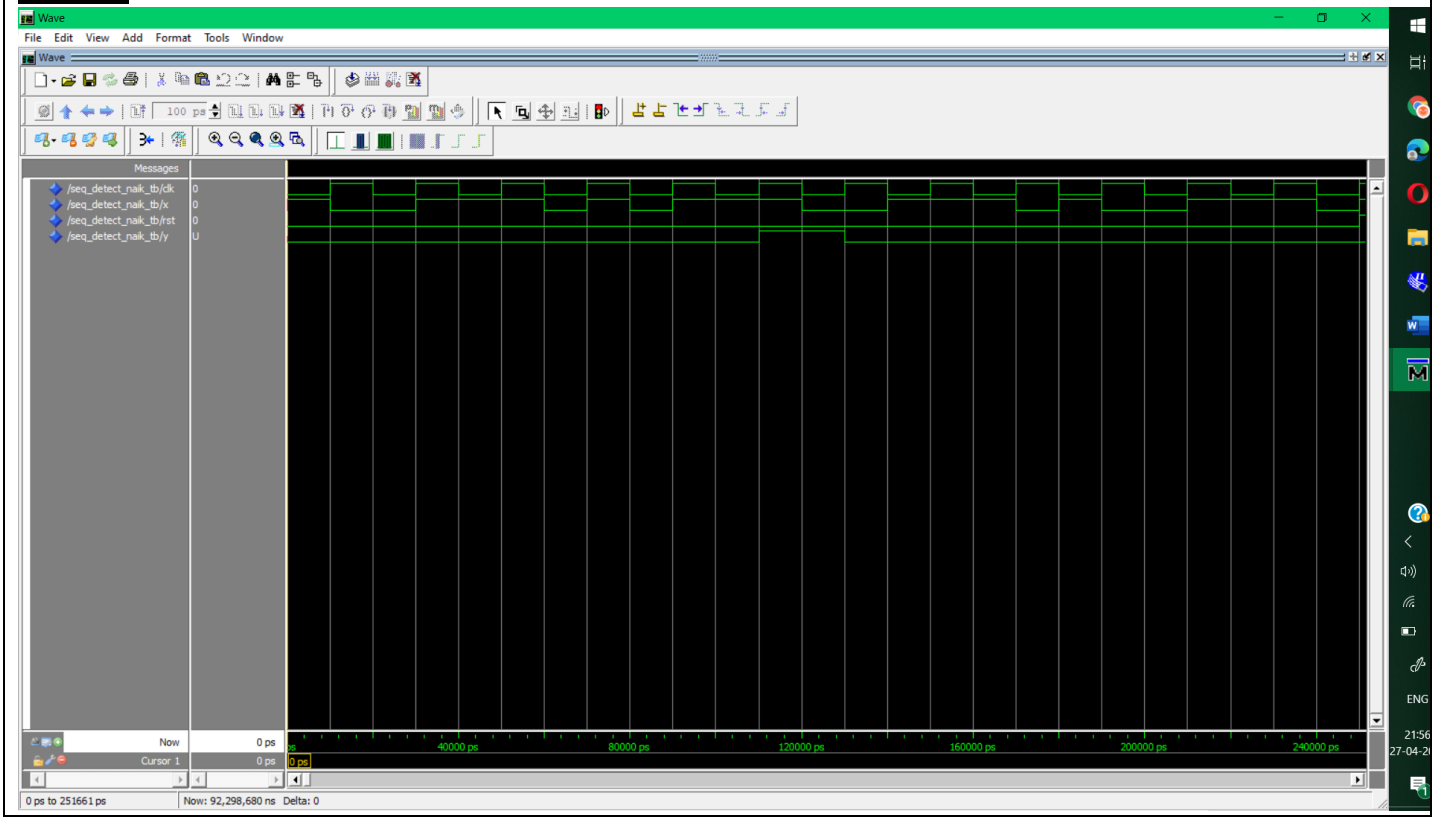
```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY seq_detect_naik_tb IS
END seq_detect_naik_tb;
ARCHITECTURE seq_detect_naik_tb_arch OF seq_detect_naik_tb IS

        COMPONENT seq_detect_naik
                port(
                        clk : in std_logic;
                        reset : in std_logic;
                        x : in std_logic;
                        output : out std_logic
                );
        END COMPONENT;

signal clk : std_logic := '0';
signal x : std_logic := '0';
signal rst : std_logic := '0';
signal y : std_logic;

        BEGIN
        uut: seq_detect_naik PORT MAP(clk,rst,x,y);

        process begin
                        clk <= '0';
                        wait for 10ns;
                        clk <= '1';
                        wait for 10ns;
        end process;
        process begin
                        rst <= '0';
                        wait for 250ns;
                        rst <= '1';
                        wait for 150ns;
        end process;
        process begin


                x <= '1';
                wait for 10ns;
```

```
                x <= '0';
                wait for 10ns;
                x <= '0';
                wait for 10ns;
                x <= '1';
                wait for 10ns;
                x <= '1';
                wait for 10ns;
                x <= '1';
                wait for 10ns;
                x <= '0';
                wait for 10ns;
                x <= '1';
                wait for 10ns;

                x <= '0';
                wait for 10ns;
                x <= '1';
                wait for 10ns;
                x <= '1';
                wait for 10ns;
                x <= '1';
                wait for 10ns;
                x <= '0';
                wait for 10ns;
                x <= '1';
                x <= '0';
                wait for 10ns;
                x <= '1';
                wait for 10ns;
                x <= '1';
                wait for 10ns;
                x <= '1';
                wait for 10ns;
                x <= '0';
                wait for 10ns;
                x <= '1';

        end process;
end seq_detect_naik_tb_arch;
```

State Diagram:

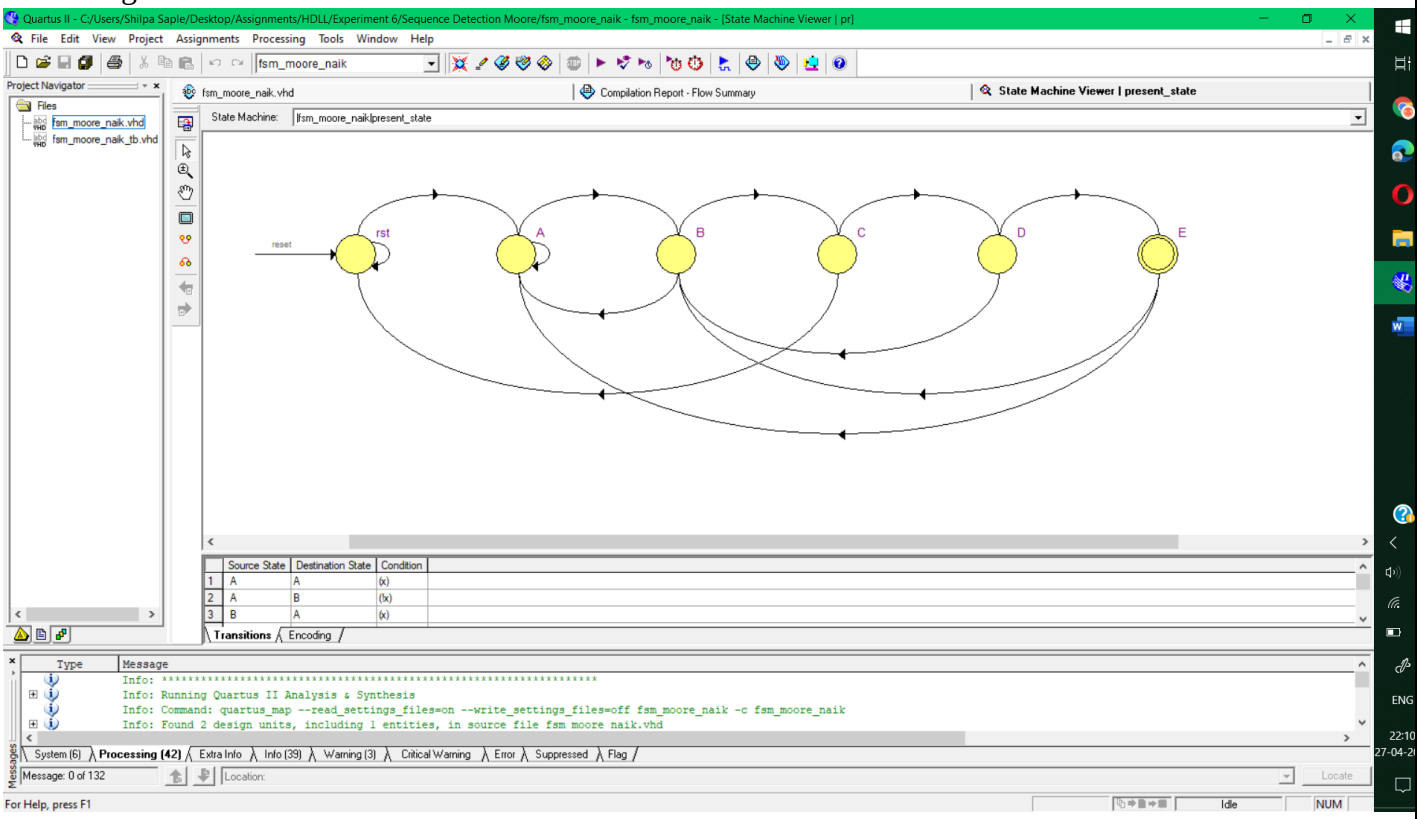


Output:



Moore Machine:

State Diagram:



Main Code:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity fsm_moore_naik is
	port(
		clk : in std_logic;
		reset : in std_logic;
		x : in std_logic;
		output : out std_logic
	);
end fsm_moore_naik;
-- 10011 Overlapping Moore
architecture fsm_moore_naik_arch of fsm_moore_naik is
	type state is (rst, A, B, C, D, E);
	signal present_state, next_state : state;
	begin
	synchronous_process: process (clk)
		begin
```

```
                if rising_edge(clk) then
                        if (reset = '1') then
                                present_state <= rst;
                        else
                                present_state <= next_state;
                        end if;
                end if;
end process;

output_decoder : process(present_state, x)
begin
        --next_state <= rst;
        case (present_state) is
        when rst =>
                if (x = '1') then
                        next_state <= A;
                else
                        next_state <= rst;
                end if;

        when A =>
                if (x = '1') then
                        next_state <= A;
                else
                        next_state <= B;
                end if;

        when B =>
                if (x = '1') then
                        next_state <= A;
                else
                        next_state <= C;
                end if;

        when C =>
                if (x= '1') then
                        next_state <= D;
                else
                        next_state <= rst;
                end if;
```

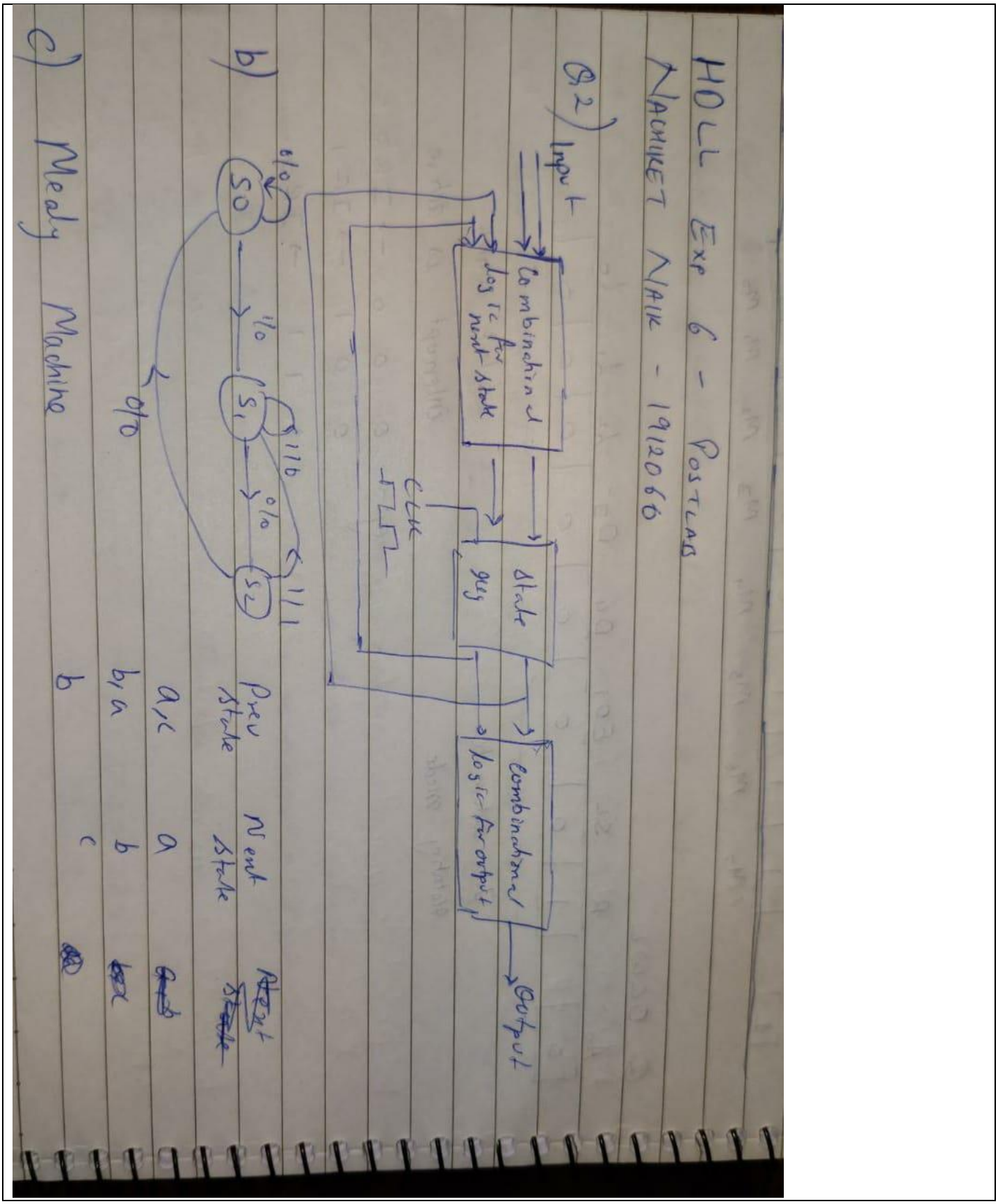
```
                when D =>
                        if (x= '1') then
                                next_state <= E;
                        else
                                next_state <= B;
                        end if;

                when E =>
                        if (x= '1') then
                                next_state <= A;
                        else
                                next_state <= B;
                        end if;
                when others => next_state <= rst;
                end case;
                end process;

        process(present_state)
                begin
                        case (present_state) is
                                when A => output <= '0';
                                when B => output <= '0';
                                when C => output <= '0';
                                when D => output <= '0';
                                when E => output <= '1';
                                when others => output <= '0';
                        end case;
        end process;

end fsm_moore_naik_arch;
```

Testbench:

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
ENTITY fsm_moore_naik_tb IS
END fsm_moore_naik_tb;
ARCHITECTURE fsm_moore_naik_tb_arch OF fsm_moore_naik_tb IS

        COMPONENT fsm_moore_naik_tb
```

```
            port(
                clk : in std_logic;
                reset : in std_logic;
                x : in std_logic;
                output : out std_logic
            );
        END COMPONENT;

signal clk : std_logic := '0';
signal x : std_logic := '0';
signal rst : std_logic := '0';
signal output : std_logic;

        BEGIN
        uut: fsm_moore_naik_tb PORT MAP(clk,rst,x,output);

        process begin
                clk <= '0';
                wait for 5ns;
                clk <= '1';
                wait for 5ns;
        end process;
        process begin
                rst <= '0';
                wait for 250ns;
                rst <= '1';
                wait for 50ns;
        end process;
        process begin


            x <= '1';
            wait for 10ns;
            x <= '0';
            wait for 10ns;
            x <= '0';
            wait for 10ns;
            x <= '1';
            wait for 10ns;
            x <= '1';
            wait for 10ns;
```

```
                x <= '1';
                wait for 10ns;
                x <= '0';
                wait for 10ns;
                x <= '1';
                wait for 10ns;

                x <= '0';
                wait for 10ns;
                x <= '1';
                wait for 10ns;
                x <= '1';
                wait for 10ns;
                x <= '1';
                wait for 10ns;
                x <= '0';
                wait for 10ns;
                x <= '1';
                x <= '0';
                wait for 10ns;
                x <= '1';
                wait for 10ns;
                x <= '1';
                wait for 10ns;
                x <= '1';
                wait for 10ns;
                x <= '0';
                wait for 10ns;
                x <= '1';

        end process;
end fsm_moore_naik_tb_arch;
```


HDL Exp 6 - Postlab
Nachiket Naik - 1912066
Q2) Input
Combinational
logic for next state
State
reg
Combinational
logic for output
Output
clk
rst
b)
S0
S1
S2
1/0
0/0
0/0
1/0
1/1
Prev State
Next State
a/c
b/a
b/a
a
b
c
c) Mealy Machine

Conclusion:

Thus, in this experiment we studied about implementing FSM (sequence detector) in VHDL. We created a non-overlapping sequence detector using Mealy and Moore Machine.

Signature of faculty in-charge with Date: