LAB 6 Submission

Submitted by: Prateek Chandan

Roll: 120050042

FSM Sequence Detector

Aim: Design and simulate a sequence detector module to detect following pattern in an input bit sequence using Xilinx ISE. "101101101"

- Use of state machine is required for design
- Module will have two inputs – Serial_Input and Input_valid
- Module will have one output – Sequence_detect
- To detect a sequence, module will take input from "Serial_Input" signal, only at the transition of the clock and Input_valid signal is High.
- Sequence_detect signal must go high only when it detects the sequence

Procedure:

1. Draw the state diagram for determining the string from serial input

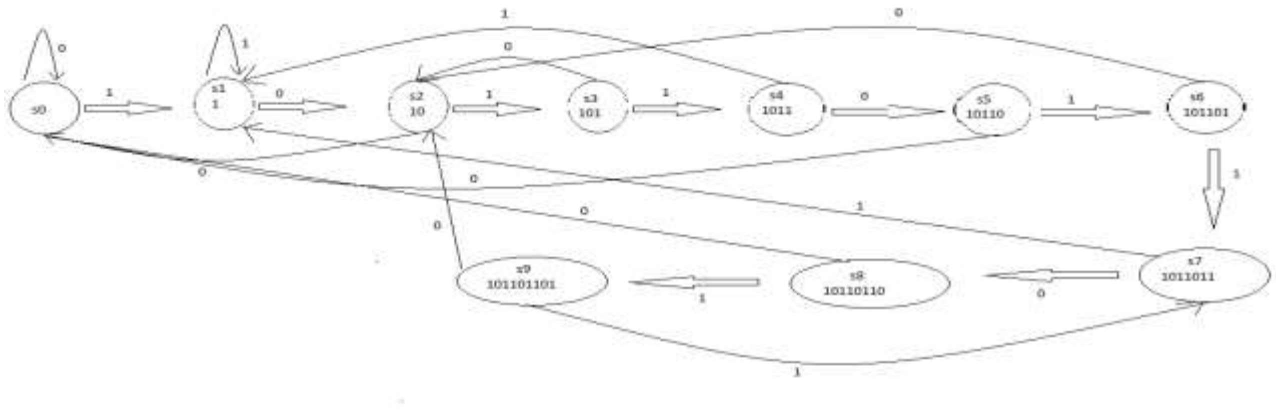

Figure 2: State diagram for string 101101101 overlapping case

2. The input to circuit was serial_input , input_valid and clock
3. Code the states and the transition function in VHDL
4. Implemented the test bench for detecting the sequence
5. Set up the clock period to be 10ns
6. Taken the test string to check to be "101001101011011011011101011011011010101011011011011"
7. Ran a loop over the test string to create the timing diagram for the test string. (Manually the string contains 6 times the given string including the overlapping"
8. Ran the simulation for 2000ns which was the total time to loop over the test string
9. Obtained the timing diagram
10. The same is done for Non-Overlapping case

OVERLAPPING CASE:

Timing Diagram:

FSM sequence detector:

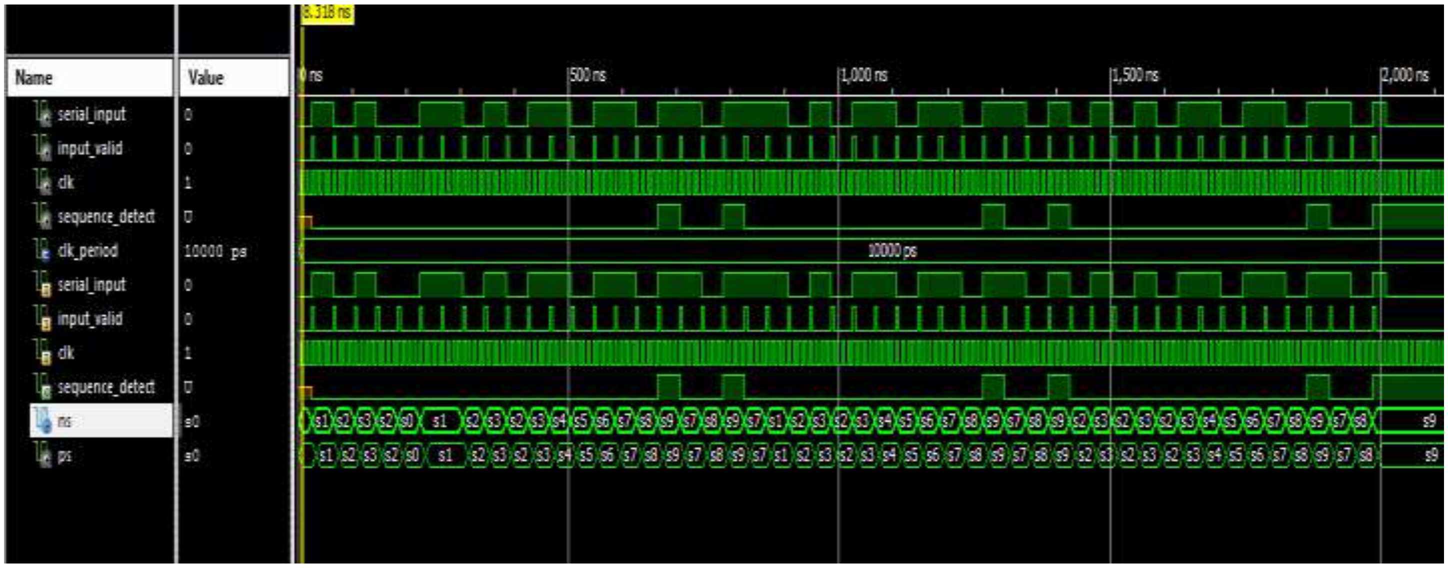

Figure 2: Timing Diagram for FSM Sequence Detector

Code:

```
--------------------------------------------------------------------------------
-- 120050042
-- Prateek Chandan
-- FSM Sequence Detector for overlapping case
--------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity fsm is
    Port ( serial_input : in  STD_LOGIC;
           input_valid : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           sequence_detect : out  STD_LOGIC);
end fsm;

architecture Behavioral of fsm is
Type state_type is (s0,s1,s2,s3,s4,s5,s6,s7,s8,s9);
signal nS,pS:state_type:=s0;
begin

         SEQ:process(clk)
         Begin
                  if(rising_edge(clk)) then
```

```
        PS <= NS;
            if(input_valid='1') then
                sequence_detect <= '0';
                case PS is
                when s0 =>
                    if(serial_input='1') then
                        NS <= s1;
                    else
                        NS <= s0;
                    end if;
                when s1 =>
                    if(serial_input='1') then
                        NS <= s1;
                    else
                        NS <= s2;
                    end if;
                when s2 =>
                    if(serial_input='1') then
                        NS <= s3;
                    else
                        NS <= s0;
                    end if;
                when s3 =>
                    if(serial_input='1') then
                        NS <= s4;
                    else
                        NS <= s2;
                    end if;
                when s4 =>
                    if(serial_input='1') then
                        NS <= s1;
                    else
                        NS <= s5;
                    end if;
                when s5 =>
                    if(serial_input='1') then
                        NS <= s6;
                    else
                        NS <= s0;
                    end if;
                when s6 =>
                    if(serial_input='1') then
                        NS <= s7;
                    else
                        NS <= s2;
                    end if;
                when s7 =>
                    if(serial_input='1') then
                        NS <= s1;
                    else
                        NS <= s8;
                    end if;
                when s8 =>
                    if(serial_input='1') then
                        NS <= s9;
                        sequence_detect <= '1';
                    else
                        NS <= s0;
                    end if;
                when s9 =>
                    if(serial_input='1') then
                        NS <= s7;
                    else
                        NS <= s2;
                    end if;
                end case;
            end if;
        end if;
    end process SEQ;
end Behavioral;
```

NON OVERLAPPING CASE:

<u>Timing Diagram:</u>

FSM sequence detector:

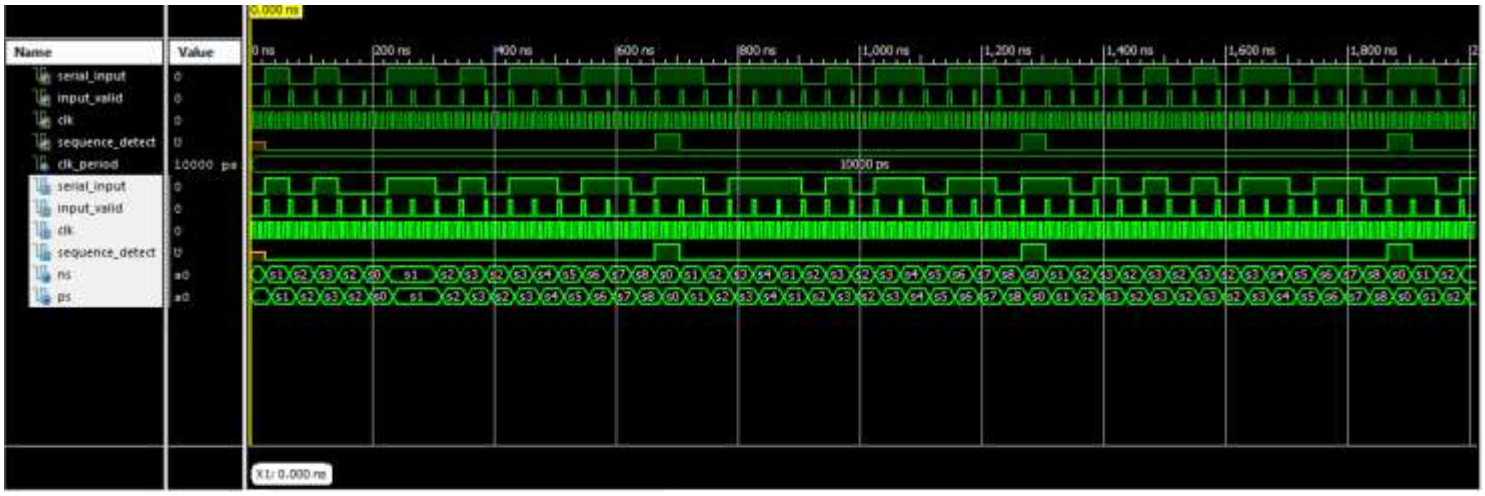

Figure 3: Timing Diagram for FSM Sequence Detector Non-Overlapping

Code:

```
-------------------------------------------------------------------------------------
-- 120050042
-- Prateek Chandan
-- FSM Sequence Detector for Non-Overlapping case
-------------------------------------------------------------------------------------
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
entity fsm is
    Port ( serial_input : in  STD_LOGIC;
           input_valid : in  STD_LOGIC;
           clk : in  STD_LOGIC;
           sequence_detect : out  STD_LOGIC);
end fsm;

architecture Behavioral of fsm is
Type state_type is (s0,s1,s2,s3,s4,s5,s6,s7,s8,s9);
signal nS,pS:state_type:=s0;
begin

          SEQ:process(clk)
          Begin
                    if(rising_edge(clk)) then
                    PS <= NS;
                              if(input_valid='1') then
                                        sequence_detect <= '0';
                                        case PS is
                                        when s0 =>
                                                  if(serial_input='1') then
                                                            NS <= s1;
                                                  else
                                                            NS <= s0;
                                                  end if;
                                        when s1 =>
                                                  if(serial_input='1') then
                                                            NS <= s1;
                                                  else
                                                            NS <= s2;
                                                  end if;
                                        when s2 =>
                                                  if(serial_input='1') then
                                                            NS <= s3;
                                                  else
                                                            NS <= s0;
                                        end if;
                                when s3 =>
                                        if(serial_input='1') then
                                                NS <= s4;
                                        else
                                                NS <= s2;
                                        end if;
                                when s4 =>
                                        if(serial_input='1') then
                                                NS <= s1;
                                        else
                                                NS <= s5;
                                        end if;
                                when s5 =>
                                        if(serial_input='1') then
                                                NS <= s6;
                                        else
                                                NS <= s0;
                                        end if;
                                when s6 =>
                                        if(serial_input='1') then
                                                NS <= s7;
                                        else
                                                NS <= s2;
                                        end if;
                                when s7 =>
                                        if(serial_input='1') then
                                                NS <= s1;
                                        else
                                                NS <= s8;
                                        end if;
                                when s8 =>
                                        if(serial_input='1') then
                                                NS <= s0;
                                                sequence_detect <= '1';
                                        else
                                                NS <= s0;
                                        end if;
                                end case;
                        end if;
                end if;
        end process SEQ;
end Behavioral;
```

<u>Inference</u>:

- Learnt How to implement FSM Sequence Detector using VHDL
- Learnt take serial input by synchronizing it with the clock
- Understood the concepts of states
- Understood how to code the state diagrams in VHDL
- Leant to run the simulation for serial input of string and test the code and data