S. B. JAIN INSTITUTE OF TECHNOLOGY,MANAGEMENT & RESEARCH,NAGPUR.

Practical No. 08

Aim: Design and Simulation of Mealy and Moore machine to detect the sequence 1101 and verify it using Test Bench.

Name of Student	**: Aadesh R. Motghare**
Roll No.	**: 41(ET20065)**
Semester/Year	**: 6th Sem/3rd Year**
Academic Session	**: 2022-23**
Date of Performance	**:**
Date of Submission	**:**

AIM: Design and Simulation of Mealy and Moore machine to detect thesequences 1101 and verify it using Test Bench**.**

OBJECTIVE:

- To verify the functionality of Mealy and Moore machine which will detect the sequences 1101
- To develop logic for designing of Mealy and Moore machine which will detect the sequences 1101.
- Mealy and Moore Machine is used to design complex digital System like vending machines, washing machines.

SOFTWARE: - Xilinx ISE14.7.

THEORY:-

A sequence detector is good example of sequential logic circuit also called FSM (Finite State Machine). A 1101 sequence detector detects 1101 consecutive bits in a string of binary bits. It is good practice to draw the state diagram of the sequence of process that happens to capture understanding of the behavior of the circuit. In the design of 1101 sequence detector, the design of state diagram is the first step, which then is followed by the creation of state table, state transition table and finally the circuit itself and testing being the final step.

Readers are recommended to read the following posts which are related to this blog post. A detailed steps to construct a sequential circuit is provided in the blog post- How to design Sequence Detector in 10 easy steps. Also another post useful in sequential circuit is- how to design Moore sequential circuit in Simulink. And futhermore this post- how to

use Xilinx Schematic editor with example of sequence detector shows how xilinx software can be used to design and verify the sequence detector.

The state diagram of 1101 sequence detector is shown below-

State S1:

Beginning at state S1 when 0 is received it stays in the same state because it has nothing to remember and the output is 0 because the sequence 1101 is not detected. Only at the instant when 1101 sequence is detected the output is high, that is, 1. Also remember that the flip flops should be used when things are to be remembered by the circuit. When 1 arrives when in state S1, then it goes to next state S2 and it remembers that 1 was received which is part of the sequence 1101 which is to be detected.

StateS2:

When in state S2, when 1 arrives, since it is part of the sequence it goes to next state S3, meaning it remembers 1. When 0 is received it cannot go to next state S3(since 1 received has occupied the transition condition and because 0 is not part of the sequence and there is nothing to remember), and it cannot remain in the same state S2 because this would mean 010 indefinite loop while in state S2, therefore it goes back to the initial state S1. Consider 100 is received and machine remains in S2 when 0 is received, then because of 1 the state changes from S1 to S2, then 0 is received then the machine stays in S2 and when another 0 is received then it stays again in S2. But consider when 100 is received and machine goes back to S1, then when 1 is received it changes state from S1 to S2, when 0 is received then goes back to S1 and when another 0 is received it stays in S1.

StateS3:

When in state S3, when 0 is received then since it is part of the sequence 1101 it goes to new state S4 because the machine has to remember the new bit 0 as part of the sequence detection algorithm. When 1 is received it stays in the same state.

StateS4:

When in state S4, when 1 is received then since it is part of the sequence 1101 to be detected it goes to S2. And when 0 is received then it goes back to initial state S1. At this point the machine outputs 1.

LOGICAL DIAGRAM:-

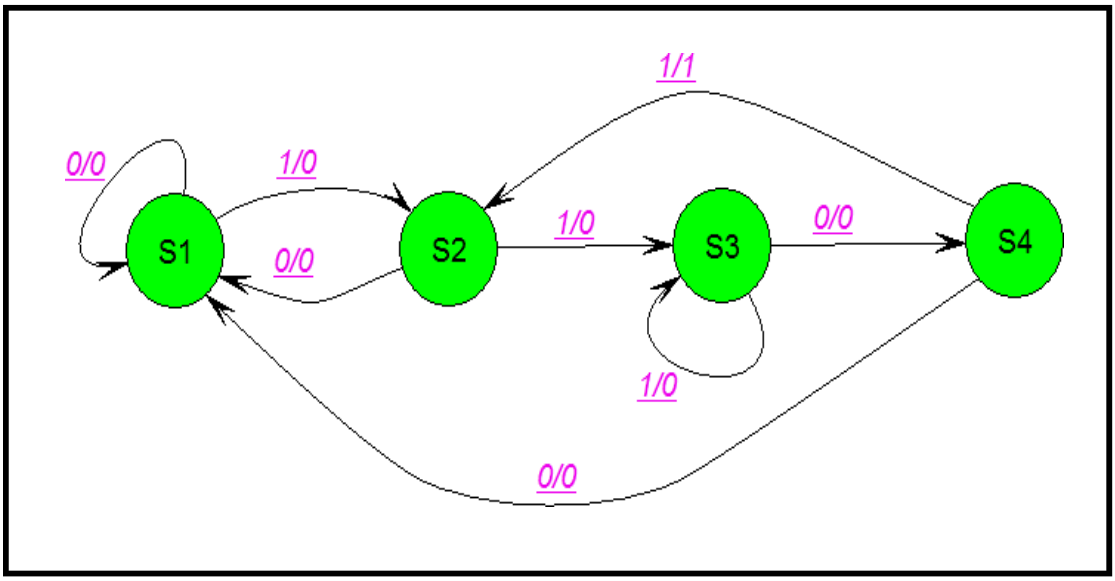


Fig 1. State Diagram of Mealy and Moore machine

TRUTH TABLE:-

Present State	Input	
	Next state , Output Input x=0	Next state , Output Input x=1
S_1	S_1, 0	S_2 , 0
S_2	S_1, 0	S_3, 0
S_3	S_4, 0	S_3, 0
S_4	S_1, 0	S_2, 1

Table 1. Transition table

VHDL CODE:-

STEPS FOR PROGRAM:-

Step 1.Library /Package Declaration : Involves declaration of all libraries and respective packages used in the design.

```
LIBRARY library_name;
USE library_name.package_name.all;
```

Step 2.Entity:

```
ENTITY entity_name is
        PORT(signal_name(s): mode signal_type;
                signal_name(s): mode signal_type;
                …);
end ENTITY entity_name;
```

Signals of the same mode and signal_type can be grouped on 1line.

MODE describes the direction data is transferred through port

- in – data flows into the port
- out – data flows out of port *only*
- buffer – data flows out of port *as well as read* internally.
- inout – bi-directional data flow into and out of port

SIGNAL_TYPE defines the data type for the signal(s)

- bit – single signals that can have logic values 0 and 1.
- bit_vector – bus signals(vector form of bit) that can have logic values 0 and 1.
- std_logic – part of std_logic_1164 package of IEEE library. Used to represent 2 value logical values i.e 0 and 1 as well as other values such as high impedance ,don't care and others as described below.

- std_logic_vector – bus signals (vector form of std_logic) but IEEE standard for simulation and synthesis note that all vectors must have a range specified example for a 4 bit bus: bit_vector (3 downto 0) or std_logic_vector (3 downto 0).

In order to use std_logic and std_logic_vector we must include the library and package usage declarations in the VHDL model before the entity statement as follows:

LIBRARY ieee;
USE ieee.std_logic_1164.all;

Values for std-logic:

U un-initialized (undefined logic value)
X forced unknown logic value
0 Logic low
1 Logic High
Z high impedance (tri-state)
W weak unknown
L weak 0
H weak 1
- don't care value (for synthesis minimization)

Step 3: Architecture Declaration

```
architecture architecture name of entity_name
 architecture_declarative_part;
begin
Statements;
 end architecture_name;
```

Here we should specify the entity name for which we are writing the architecture body. The architecture statements should be inside the begin and end keyword. Architecture declarative part may contain variables, constants, or component declaration.

Step 4: Simulate the VHDL code and remove the syntax errors if any.

Step 5: Write the testbench and verify the design .

CODE: Moore 1101 Sequence Detector

ISE Project Navigator (P.20131013) - C:\Users\ACER\OneDrive\Desktop\DSD practs\moore_1101_SD_pract\moore_1101_SD_pract.xise - [moore_1101.vhd]

File Edit View Project Source Process Tools Window Layout Help

Design

View: Implementation Simulation

Behavioral

Hierarchy

- moore_1101_SD_pract
- xc7a100t-3csg324
  - moore_1101_tb - Behavioral (moore_1101_tb.vh
    - DUT - moore_1101 - behavior (moore_1101.

No Processes Running

Processes: moore_1101_tb - Behavioral

- ISim Simulator
  - Behavioral Check Syntax
  - Simulate Behavioral Model

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

-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--use IEEE.NUMERIC_STD.ALL;
-- Moore 1101 Sequence Detector

entity moore_1101 is
    port (
        clk   : in std_logic;
        reset : in std_logic;
        input : in std_logic;
        output: out std_logic
    );
end entity moore_1101;
architecture behavior of moore_1101 is
    type state_type is (S0, S1, S2, S3, S4);
    signal current_state, next_state: state_type;
begin
    -- Moore FSM logic
    process (clk, reset)
    begin
        if reset = '1' then
            current_state <= S0;
        elsif rising_edge(clk) then
            current_state <= next_state;
        end if;
    end process;

    -- Next state and output logic
    process (current_state, input)
    begin
        case current_state is
            when S0 =>
                if input = '1' then
                    next_state <= S1;
                else
                    next_state <= S0;
                end if;
                output <= '0';
            when S1 =>
                if input = '1' then
                    next_state <= S2;
                else
                    next_state <= S0;
                end if;
                output <= '0';
            when S2 =>
                if input = '0' then
                    next_state <= S3;
                else
                    next_state <= S0;
                end if;
                output <= '0';
            when S3 =>
                if input = '1' then
                    next_state <= S4;
                else
                    next_state <= S0;
                end if;
                output <= '0';
            when S4 =>
                next_state <= S0;
                output <= '1';
        end case;
    end process;
end behavior;
```

Design

View: Implementation Simulation

Behavioral

Hierarchy

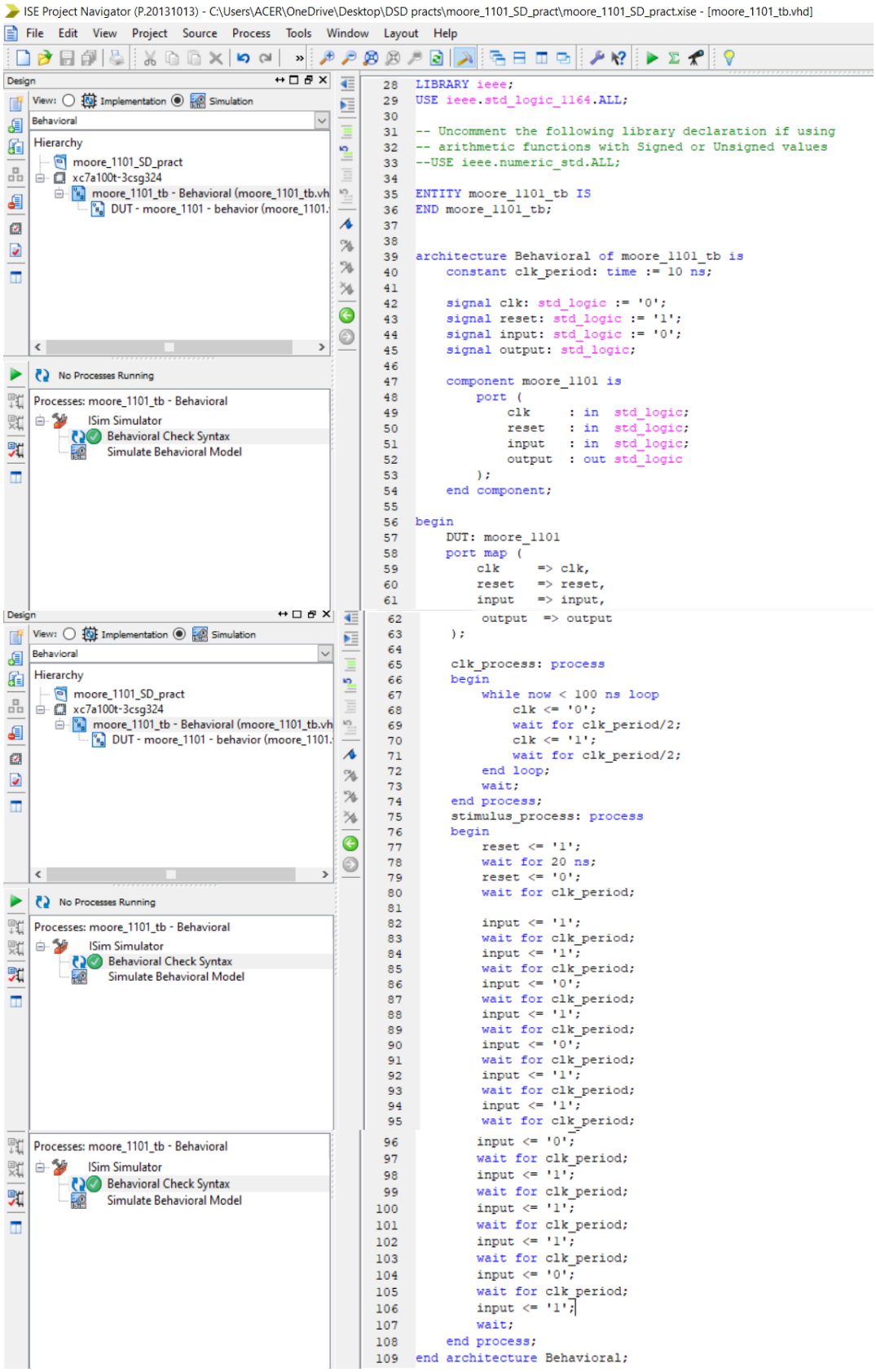
- moore_1101_SD_pract
- xc7a100t-3csg324
  - moore_1101_tb - Behavioral (moore_1101_tb.vh
    - DUT - moore_1101 - behavior (moore_1101.

No Processes Running

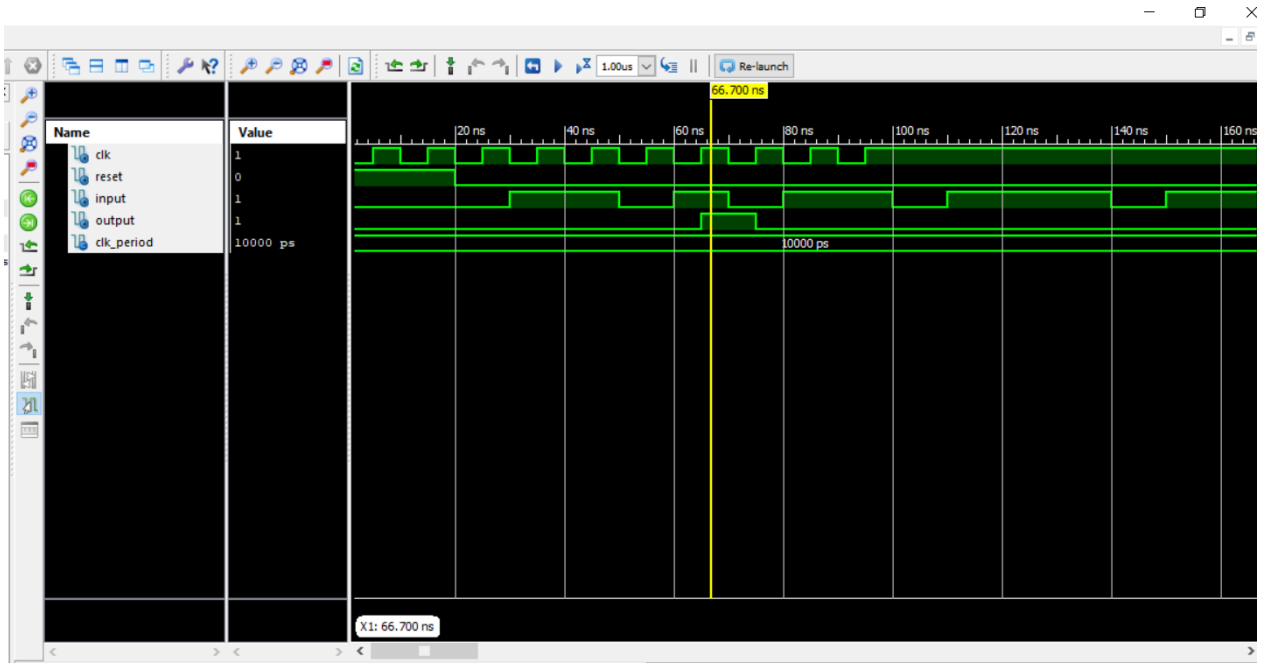
Processes: moore_1101_tb - Behavioral

- ISim Simulator
  - Behavioral Check Syntax
  - Simulate Behavioral Model

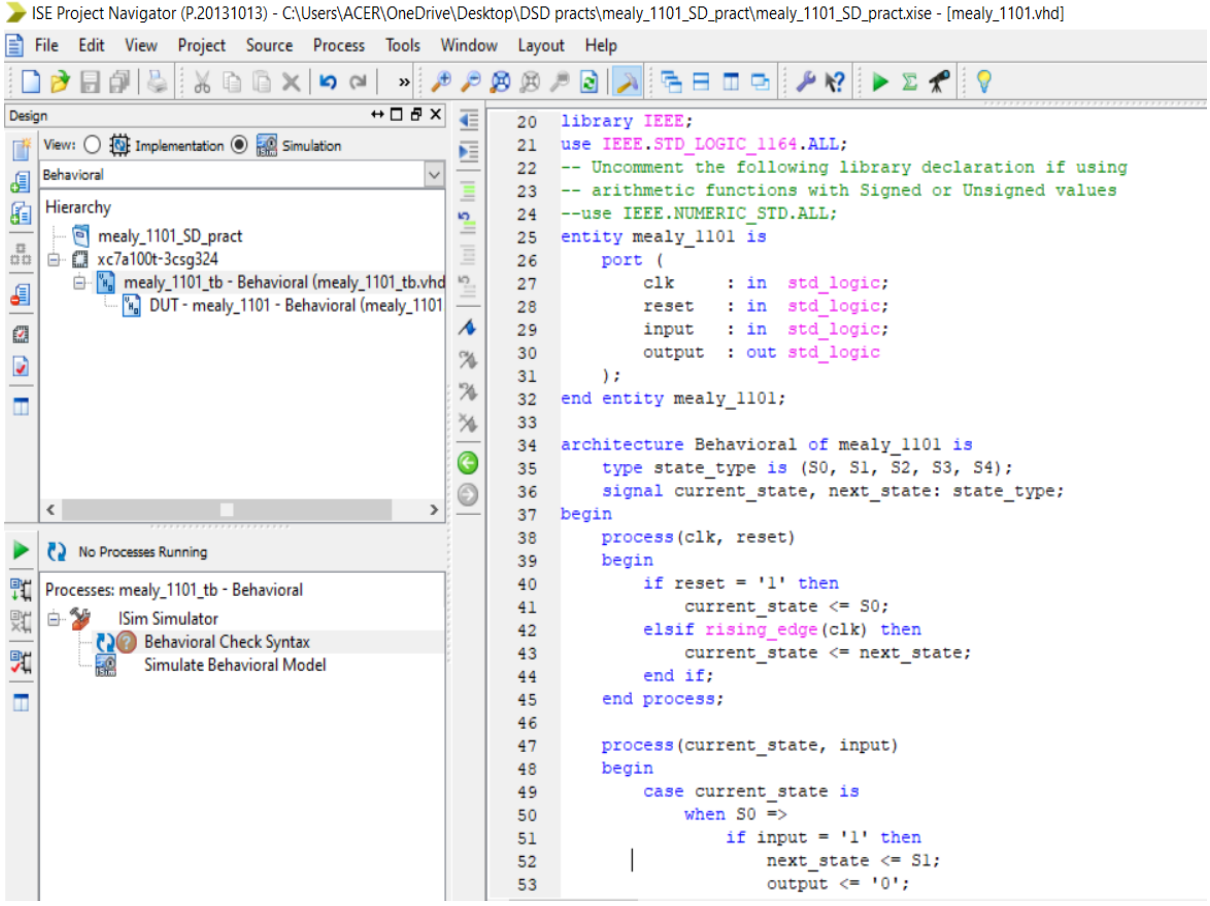
TESTBENCH:

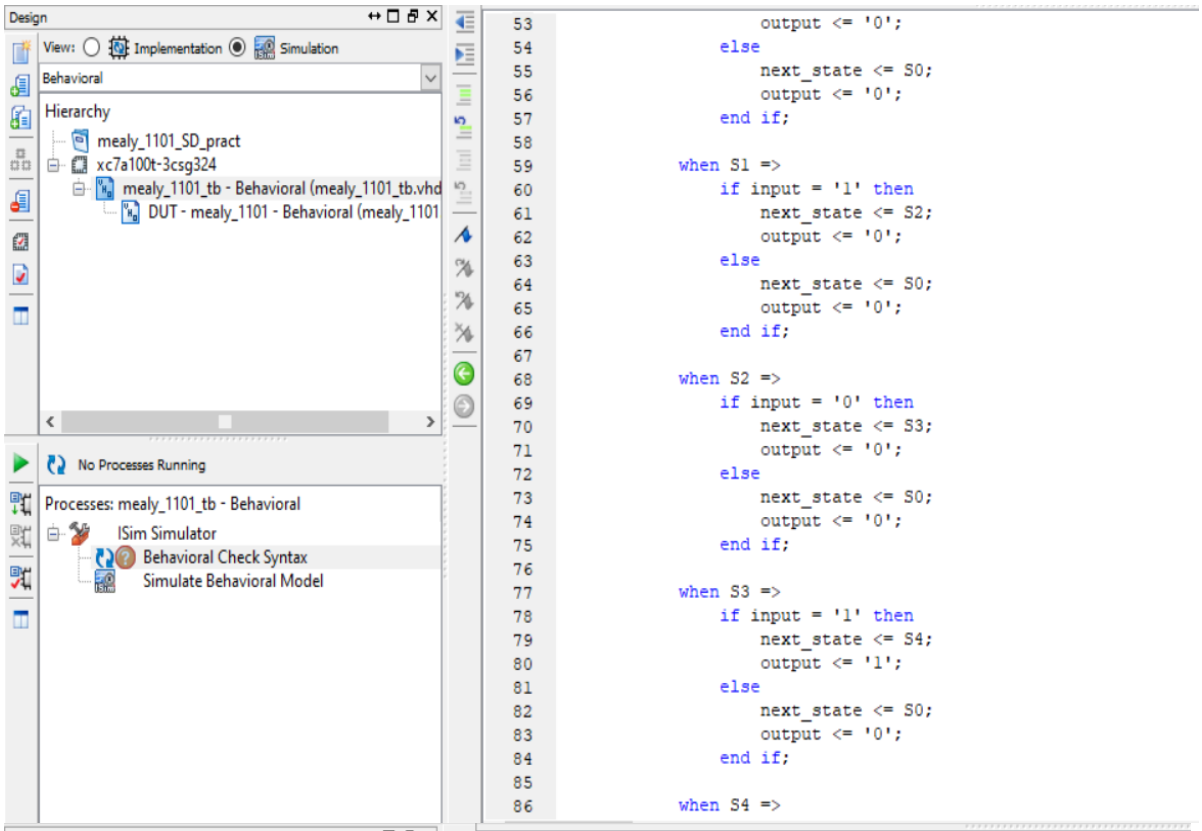

ISE Project Navigator (P.20131013) - C:\Users\ACER\OneDrive\Desktop\DSD practs\moore_1101_SD_pract\moore_1101_SD_pract.xise - [moore_1101_tb.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
moore_1101_SD_pract
xc7a100t-3csg324
moore_1101_tb - Behavioral (moore_1101_tb.vh
DUT - moore_1101 - behavior (moore_1101.
No Processes Running
Processes: moore_1101_tb - Behavioral
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;
ENTITY moore_1101_tb IS
END moore_1101_tb;
architecture Behavioral of moore_1101_tb is
constant clk_period: time := 10 ns;
signal clk: std_logic := '0';
signal reset: std_logic := '1';
signal input: std_logic := '0';
signal output: std_logic;
component moore_1101 is
port (
clk : in std_logic;
reset : in std_logic;
input : in std_logic;
output : out std_logic
);
end component;
begin
DUT: moore_1101
port map (
clk => clk,
reset => reset,
input => input,
output => output
);
clk_process: process
begin
while now < 100 ns loop
clk <= '0';
wait for clk_period/2;
clk <= '1';
wait for clk_period/2;
end loop;
wait;
end process;
stimulus_process: process
begin
reset <= '1';
wait for 20 ns;
reset <= '0';
wait for clk_period;
input <= '1';
wait for clk_period;
input <= '1';
wait for clk_period;
input <= '0';
wait for clk_period;
input <= '1';
wait for clk_period;
input <= '0';
wait for clk_period;
input <= '1';
wait for clk_period;
input <= '1';
wait for clk_period;
input <= '0';
wait for clk_period;
input <= '1';
wait for clk_period;
input <= '1';
wait for clk_period;
input <= '0';
wait for clk_period;
input <= '1';
wait;
end process;
end architecture Behavioral;

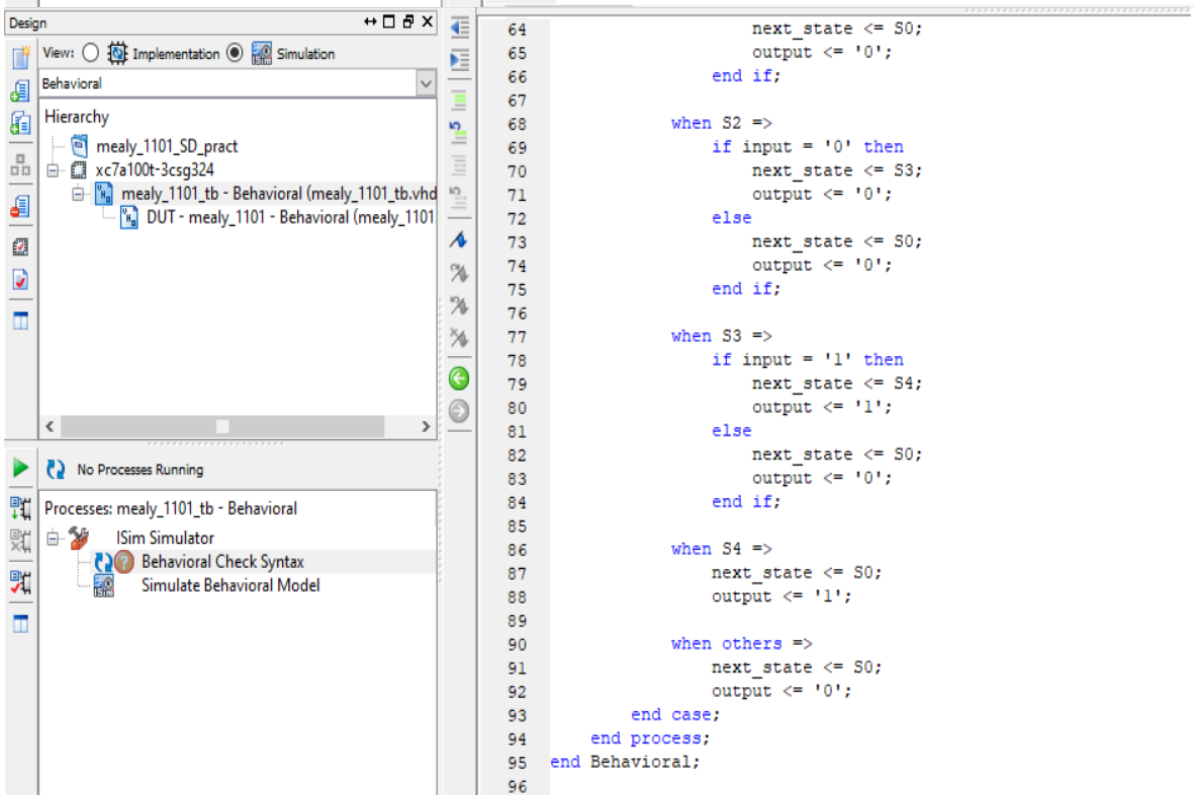
WAVEFORM:



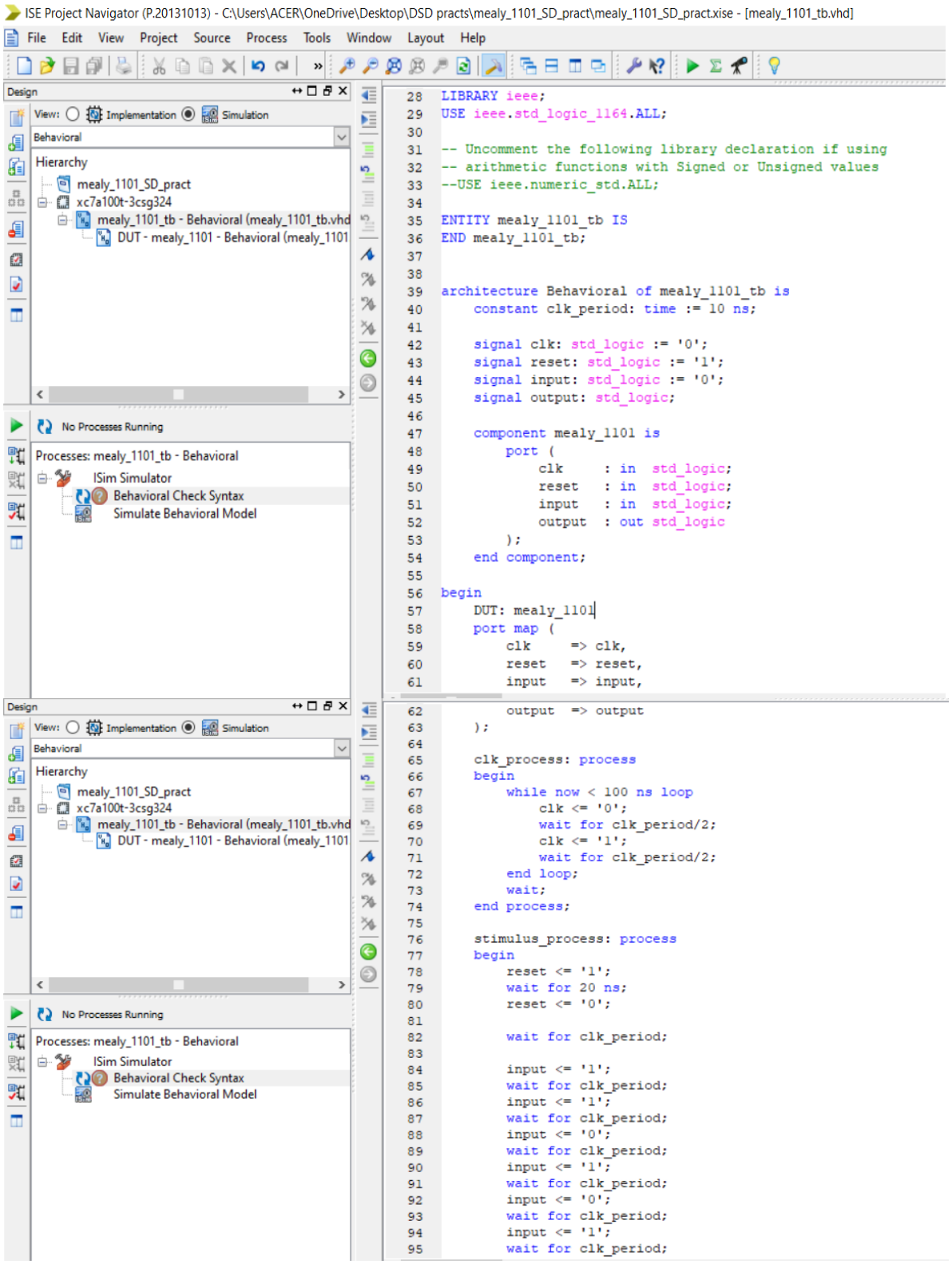
CODE: Mealy 1101 Sequence Detector

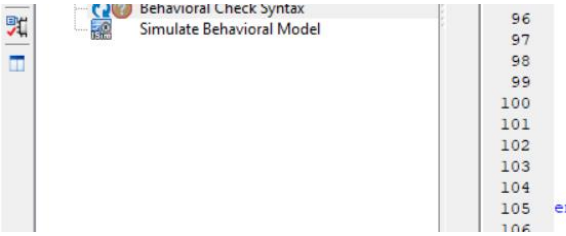



Design
View: Implementation Simulation
Behavioral
Hierarchy
mealy_1101_SD_pract
xc7a100t-3csg324
mealy_1101_tb - Behavioral (mealy_1101_tb.vhd
DUT - mealy_1101 - Behavioral (mealy_1101
No Processes Running
Processes: mealy_1101_tb - Behavioral
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
53 output <= '0';
54 else
55 next_state <= S0;
56 output <= '0';
57 end if;
58
59 when S1 =>
60 if input = '1' then
61 next_state <= S2;
62 output <= '0';
63 else
64 next_state <= S0;
65 output <= '0';
66 end if;
67
68 when S2 =>
69 if input = '0' then
70 next_state <= S3;
71 output <= '0';
72 else
73 next_state <= S0;
74 output <= '0';
75 end if;
76
77 when S3 =>
78 if input = '1' then
79 next_state <= S4;
80 output <= '1';
81 else
82 next_state <= S0;
83 output <= '0';
84 end if;
85
86 when S4 =>


Design
View: Implementation Simulation
Behavioral
Hierarchy
mealy_1101_SD_pract
xc7a100t-3csg324
mealy_1101_tb - Behavioral (mealy_1101_tb.vhd
DUT - mealy_1101 - Behavioral (mealy_1101
No Processes Running
Processes: mealy_1101_tb - Behavioral
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
64 next_state <= S0;
65 output <= '0';
66 end if;
67
68 when S2 =>
69 if input = '0' then
70 next_state <= S3;
71 output <= '0';
72 else
73 next_state <= S0;
74 output <= '0';
75 end if;
76
77 when S3 =>
78 if input = '1' then
79 next_state <= S4;
80 output <= '1';
81 else
82 next_state <= S0;
83 output <= '0';
84 end if;
85
86 when S4 =>
87 next_state <= S0;
88 output <= '1';
89
90 when others =>
91 next_state <= S0;
92 output <= '0';
93 end case;
94 end process;
95 end Behavioral;
96

TESTBENCH:


ISE Project Navigator (P.20131013) - C:\Users\ACER\OneDrive\Desktop\DSD practs\mealy_1101_SD_pract\mealy_1101_SD_pract.xise - [mealy_1101_tb.vhd]
File Edit View Project Source Process Tools Window Layout Help
Design
View: Implementation Simulation
Behavioral
Hierarchy
mealy_1101_SD_pract
xc7a100t-3csg324
mealy_1101_tb - Behavioral (mealy_1101_tb.vhd
DUT - mealy_1101 - Behavioral (mealy_1101
No Processes Running
Processes: mealy_1101_tb - Behavioral
ISim Simulator
Behavioral Check Syntax
Simulate Behavioral Model
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
-- Uncomment the following library declaration if using
-- arithmetic functions with Signed or Unsigned values
--USE ieee.numeric_std.ALL;
ENTITY mealy_1101_tb IS
END mealy_1101_tb;
architecture Behavioral of mealy_1101_tb is
constant clk_period: time := 10 ns;
signal clk: std_logic := '0';
signal reset: std_logic := '1';
signal input: std_logic := '0';
signal output: std_logic;
component mealy_1101 is
port (
clk : in std_logic;
reset : in std_logic;
input : in std_logic;
output : out std_logic
);
end component;
begin
DUT: mealy_1101
port map (
clk => clk,
reset => reset,
input => input,
output => output
);
clk_process: process
begin
while now < 100 ns loop
clk <= '0';
wait for clk_period/2;
clk <= '1';
wait for clk_period/2;
end loop;
wait;
end process;
stimulus_process: process
begin
reset <= '1';
wait for 20 ns;
reset <= '0';
wait for clk_period;
input <= '1';
wait for clk_period;
input <= '1';
wait for clk_period;
input <= '0';
wait for clk_period;
input <= '1';
wait for clk_period;
input <= '0';
wait for clk_period;
input <= '1';
wait for clk_period;

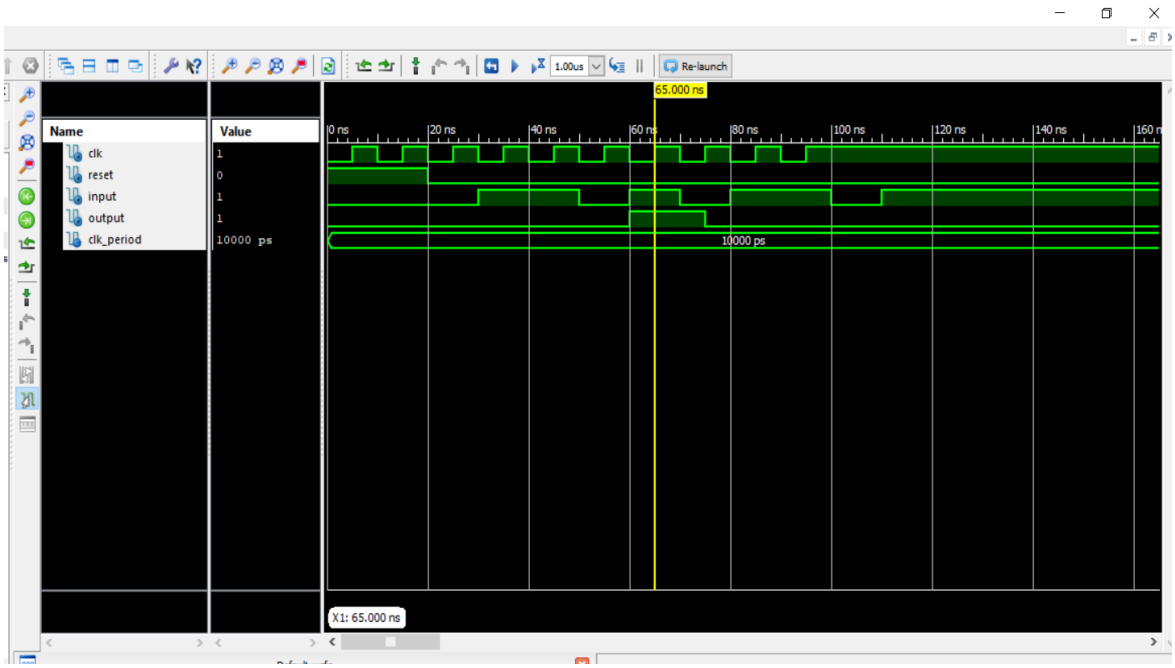



```
        input <= '1';
        wait for clk_period;
        input <= '0';
        wait for clk_period;
        input <= '1';

        wait;
    end process;

end Behavioral;
```

WAVEFORM:



RESULT:-

The VHDL code for Mealy and Moore machine which will detect the sequences 1101 isexecuted and desired output is obtained.

CONCLUSION: Here, I successfully design and check the moore and mealy sequence detector of 1101 sequence.

DISCUSSION & VIVA VOCE

1. Explain the working of Mealy and Moore.
2. How to design Mealy and Moore in VHDL
3. What are the applications of Mealy and Moore?

REFERENCE:

- VHDL Primer–J Bhasker –Pearson Education
- NPTEL Video Lecture link- https://youtu.be/FZAHhQ1v7B0?list=PL803563859BF7ED8C