SEQUENCE DETECTOR USING MELAY AND MOORE MACHINE

Mini Project 2B Report

Submitted in partial fulfillment of the requirement of University of Mumbai

For the Degree of

(Electronics & Telecommunication Engineering)

By

Gopal Deeparam Chandora	**TU2F1920098**
Arvind Lalji Jaiswal	**TU2F1920071**
Ayush Pandey	**TU2F1920084**
Yash Arjun Shrivastav	**TU2F1920072**

Under Guidance of

Prof. Atul Kadam

Department of Electronics & Telecommunication Engineering

Terna Engineering College

Plot No. 12, Sector No. 22, Nerul, Navi Mumbai-400706

Affiliated to UNIVERSITY OF MUMBAI

2021-22

TERNA ENGINEERING COLLEGE, NERUL,

NAVI MUMBAI

Department of Electronic and Telecommunication Engineering

Academic Year 2021-22

CERTIFICATE

This is to certify that the mini project 2B entitles **“Sequence Detector using MELAY and MOORE MACHINE”** is a bonafide work of

Gopal Deeparam Chandora	**TU2F1920098**
Arvind Lalji Jaiswal	**TU2F1920071**
Ayush Pandey	**TU2F1920084**
Yash Arjun Shrivastav	**TU2F1920072**

submitted to the University of Mumbai in partial fulfillment of the requirement for the award of the Bachelor of Engineering (Electronic and Telecommunication Engineering).

Guide	**Head Of Department**	**Principal**
(Prof. Atul Kadam)	**(Dr. Jyothi Digge)**	**(Dr. L. K. Ragha)**

Project Report Approval

This Mini Project 2B Report – entitled **“Sequence Detector using MELAY and MOORE MACHINE”** by following students is approved for the degree of ***B.E. in "Electronics and Telecommunications Engineering".***

Submitted by:

Gopal Deeparam Chandora	**TU2F1920098**
Arvind Lalji Jaiswal	**TU2F1920071**
Ayush Pandey	**TU2F1920084**
Yash Arjun Shrivastav	**TU2F1920072**

Examiners Name & Signature:

1.--

2.---

Date: --------------------------------

Place: -------------------------------

Declaration

We declare that this written submission represents my ideas in my own words and where others' ideas or words have been included; I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Gopal Deeparam Chandora	**TU2F1920098**	___________________
Arvind Lalji Jaiswal	**TU2F1920071**	___________________
Ayush Pandey	**TU2F1920084**	___________________
Yash Arjun Shrivastav	**TU2F1920072**	___________________

Date: ______________________

Place: ______________________

Acknowledgement

We would like to express our sincere gratitude towards our guide **Prof. Atul Kadam**, Mini Project Coordinators **Prof. Vijaypal Yadav** for their help, guidance and encouragement, they provided during the project development. This work would have not been possible without their valuable time, patience and motivation. We thank them for making our stint thoroughly pleasant and enriching. It was great learning and an honor being their student.

We are deeply thankful to **Dr. Jyothi Digge (H.O.D: - Electronics and Telecommunications Department)** and entire team in the Electronics and Telecommunications Department. They supported us with scientific guidance, advice and encouragement, they were always helpful and enthusiastic and this inspired us in our work.

We take the privilege to express our sincere thanks to **Dr. L. K. Ragha** our principal for providing the encouragement and much support throughout our work.

Gopal Deeparam Chandora	**TU2F1920098**	___________________
Arvind Lalji Jaiswal	**TU2F1920071**	___________________
Ayush Pandey	**TU2F1920084**	___________________
Yash Arjun Shrivastav	**TU2F1920072**	___________________

Date: ____________________

Place: ____________________

Index

TABLE OF CONTENTS

1. Abstract

There is an enormous usage of sequence detectors in digital circuits as it is the basic function and it became essential in most of the digital systems counting ALU, microprocessors and DSP. Sequential circuit's works on a clock cycle which may be synchronous or asynchronous. Sequential circuits use current inputs and previous inputs by storing the information and putting back into the circuit on the next clock cycle. This paper presents the Sequence Detector in Verilog using Mealy and Moore Machines, which is a sequential state machine used to detect bits pattern in a binary string. The flip-flops help to detect the pattern in the given string. The Sequence Detector gives for some particular sequence of inputs and outputs, whenever the desired sequence has found. And this paper shows a great vision on the design analysis of sequence detector using Verilog. The delay (1.045ns) minimized. The proposed architecture of sequence detector is synthesized in Xilinx ISE14.7/ Xilinx Vivado.

Keyword: Finite state machine sequence detector, Verilog, VHDL, Xilinx Vivado, Reversible logic.

List of figures

List of Table

Chapter 1

Introduction

1.1 Motivation:

Consequently, testing is an indispensable part of system design and implementation; yet it has proved to be a formidable task for complex systems. This motivates us to study testing finite state machines to ensure the correct functioning of systems and to discover aspects of their behavior.

This report proposed a sequence detector which will generate an output logic high when the given sequence is detected. The detector is based on both Moore and Mealy machine principal. The output is shown on the waveform form with logic values. We have provided three input clk, rst, x and one output y. From the input sequence x the detector will extract the desire sequence.

1.2 Sequence Detector:

A sequence detector is a sequential state machine that takes an input string of bits and generates an output 1 whenever the target sequence has been detected.

1.3 Mealy Machine:

In the theory of computation, a Mealy machine is a finite-state machine whose output values are determined both by its current state and the current inputs. This is in contrast to a Moore machine, whose (Moore) output values are determined solely by its current state. A Mealy machine is a deterministic finite-state transducer: for each state and input, at most one transition is possible.

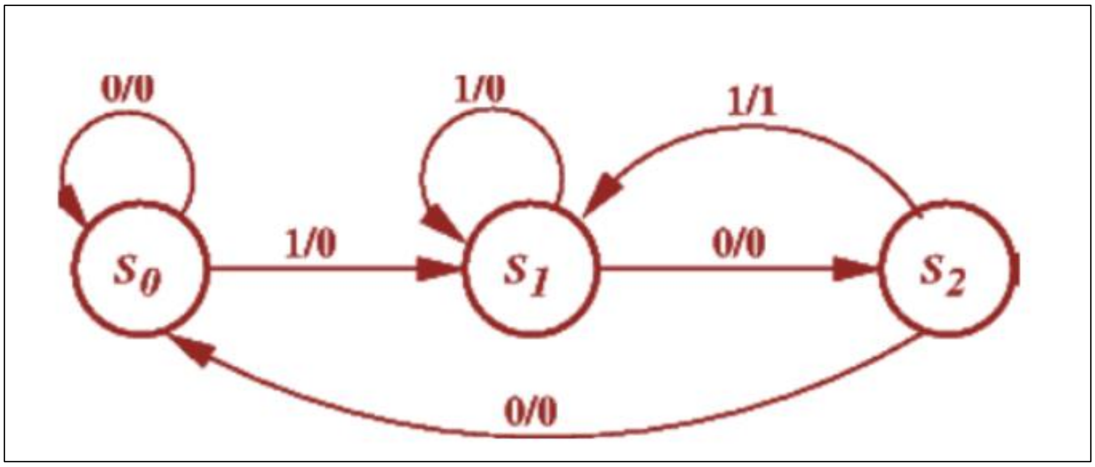

Figure 1.1 mealy state diagram
(source: www.allaboutfpga.com)

1.4 Moore's Machine:

In the theory of computation, a Moore machine is a finite-state machine whose current output values are determined only by its current state. This is in contrast to a Mealy machine, whose output values are determined both by its current state and by the values of its inputs. Like other finite state machines, in Moore machines, the input typically influences the next state. Thus, the input may indirectly influence subsequent outputs, but not the current or immediate output. The Moore machine is named after Edward F. Moore, who presented the concept in a 1956 paper, “Gedanken-experiments on Sequential Machines.

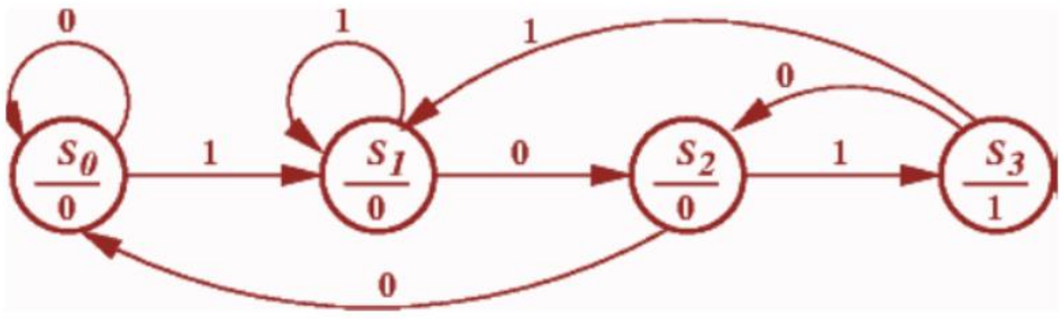

Figure 1.2 Moore State diagram

(source: www.allaboutfpga.com)

Mealy Machine	**Moore's Machine**
Output depends on present state as well as present input.	Output depends only upon present state.
If input changes, output also changes.	If input changes, output does change.
Less number of states are required.	More number of states are required.
There is more hardware requirement for circuit implementation.	There is less hardware requirement for circuit implementation.
They react faster to inputs.	They react slower to inputs(One clock cycle later).
Asynchronous output generation.	Synchronous output and state generation.
Output is placed on transitions.	Output is placed on states.
It is difficult to design.	Easy to design.

Table 1.1 Comparison between Mealy and Moore Machine

Chapter 2
Problem Statement

1.1 Statement:

To Design a sequence detector to detect 101 sequence from the given input sequence 010100 using Moore and Mealy principal, compared the results to find the optimum solution for this project.

1.2 Goals and Objective:

The main Goal is to complete semester project so we decide to make a sequence detector as our mini project work. The second goal is to gain practical knowledge on VLSI design circuits, Verilog and VHDL language using FPGA as a microcontroller.

The objective of the project is to develop a sequence detector for 101 sequence and compare the results of both principles and get the best solution for the sequence.

Chapter 3

Literature Survey

The exciting history of how finite automata became a branch of computer science illustrates its wide range of applications. The first people to consider the concept of a finite-state machine included a team of biologists, psychologists, mathematicians, engineers and some of the first computer scientists. They all shared a common interest: to model the human thought process, whether in the brain or in a computer. Warren McCulloch and Walter Pitts, two neurophysiologists, were the first to present a description of finite automata in 1943. Their paper, entitled, "A Logical Calculus Immanent in Nervous Activity", made significant contributions to the study of neural network theory, theory of automata, the theory of computation and cybernetics. Later, two computer scientists, G.H. Mealy and E.F. Moore, generalized the theory to much more powerful machines in separate papers, published in 1955-56. The finite-state machines, the Mealy machine and the Moore machine, are named in recognition of their work. While the Mealy machine determines its outputs through the current state and the input, the Moore machine's output is based upon the current state alone.

An automaton in which the state set Q contains only a finite number of elements is called a finite-state machine (FSM). FSMs are abstract machines, consisting of a set of states (set Q), set of input events (set I), a set of output events (set Z) and a state transition function. The state transition function takes the current state and an input event and returns the new set of output events and the next state. Therefore, it can be seen as a function which maps an ordered sequence of input events into a corresponding sequence, or set, of output events.

Finite-state machines are ideal computation models for a small amount of memory, and do not maintain memory. This mathematical model of a machine can only reach a finite number of states and transitions between these states. Its main application is in mathematical problem analysis. Finite-machines are also used for purposes aside from general computations, such as to recognize regular languages.

Chapter 4

Design and Implementation

4.1 Flowchart

4.1.1. Moore Machine

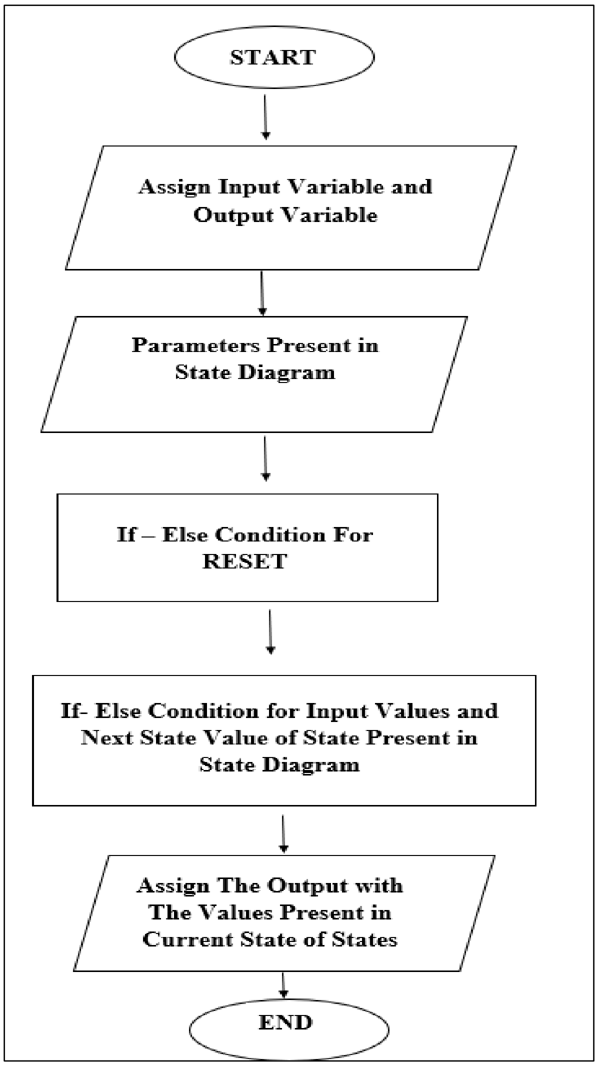
START
Assign Input Variable and Output Variable
Parameters Present in State Diagram
If – Else Condition For RESET
If- Else Condition for Input Values and Next State Value of State Present in State Diagram
Assign The Output with The Values Present in Current State of States
END

Figure 4.1 Flowchart of Moore's Machine

4.1.2 Mealy Machine

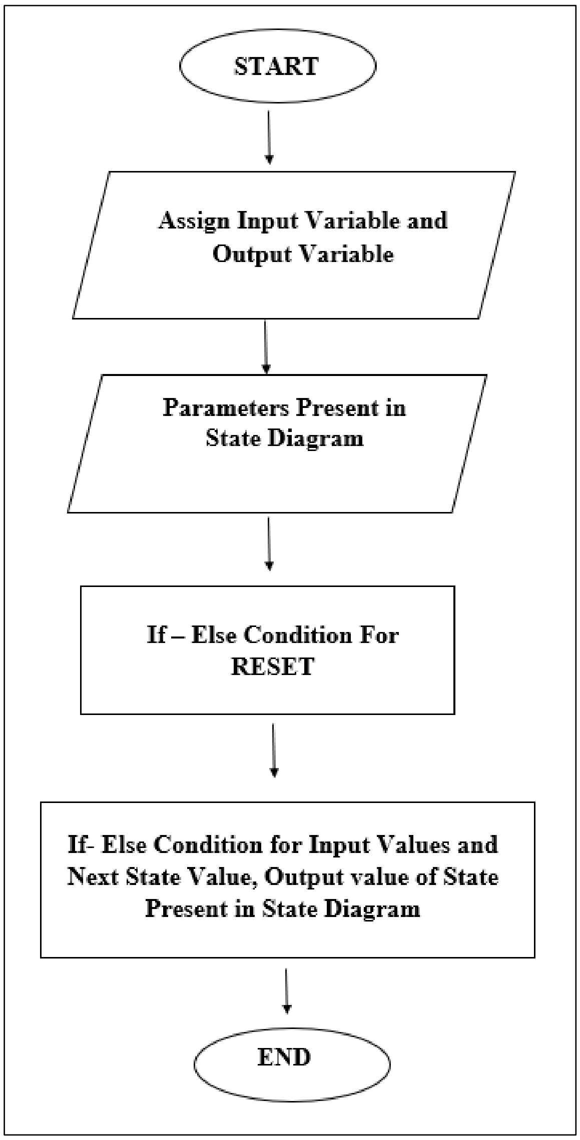
START
Assign Input Variable and Output Variable
Parameters Present in State Diagram
If – Else Condition For RESET
If- Else Condition for Input Values and Next State Value, Output value of State Present in State Diagram
END

Figure 4.2 Flowchart of Mealy Machine

4.1.3 Testbench

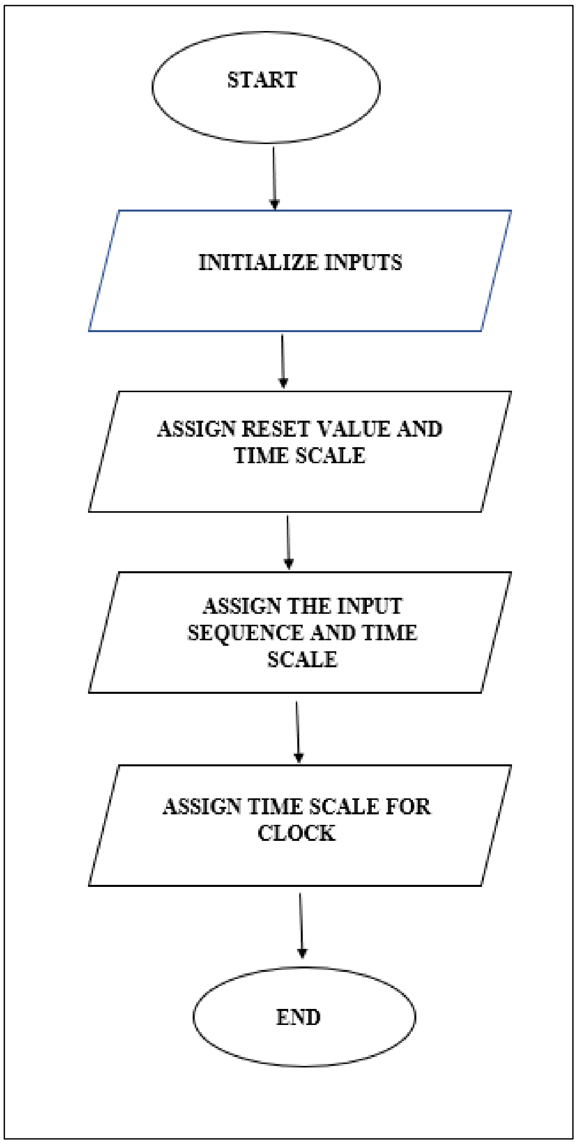
START
INITIALIZE INPUTS
ASSIGN RESET VALUE AND TIME SCALE
ASSIGN THE INPUT SEQUENCE AND TIME SCALE
ASSIGN TIME SCALE FOR CLOCK
END

Figure 4.3 Flowchart of Testbench code

4.2 Working:

Here we detect the sequence 101 by using both the machine Moore and mealy. The working of both the code in Xilinx Vivado are as follows -

4.2.1Working of Moore machine (overlapping) code to detect 101 sequence -

To write code we have to create and name a file by assigning code language, board, IC. The code start with a module depends on the file which we have created. Now, we have to give the inputs and outputs variables. We can write inputs and outputs in one line (separate with comma) or multiline also (separate with semi-colon).

In output we have to give register (reg) because we are writing code in behavioral. There are some parameters in the code like state in state diagram. We have to assign the parameters by giving parameters keywords followed by numbers of state present in state diagram also giving the values in the binary form. Now assigned the (reg) to current state as well as next state.

As we are writing the code in behavioral. So, always keywords followed by @ assigning the posedge to the clock and reset. To give the condition in behavioral, we have to start with begin and end with end. If reset == 1 then current state bits loop back to same state (i.e) S0 else the current state bits goes to next state. Now we have to write the code between current state and input bits.

As per the state diagram assign every state with the input's bits value and current state value and next state value to detect the 101 sequence. In S0 state it's also called as initial state if input==1 then next state bit value goes to S1 state else next state bit value loop back to same state. In S1 state if input==1 then next state bit value loop back to S1 state else next state bit value goes to S2 state. In S2 state if input==1 then next state bit value goes to S3 state else next state bit value goes to S0 state. In S3 state if input==1 then next state bit value goes to S1 state else next state bit value goes to S2 state, end the begin and case condition. There are some current state value in state diagram, we have to assign that value as the output. And end the module.

To detect the 101 sequence we have to give time scale for reset, clock and the inputs sequence bits as well as assigning the reset and input values. For these we required test bench, so we have to create and name one more file under the file which we have created earlier and give the time scale for reset, clock and input sequence and assign the input sequence and then end the module.

4.2.2 Working of Mealy machine (overlapping) code to detect 101 sequence -

For 101 sequence the number of state in mealy state diagram are three (i.e.) S0, S1, S2. The code for mealy machine is same as the code of Moore machine, some minor changes in mealy machine code are in parameters, condition for case of current state and input state, test bench. In parameters we have to assign only three state S0, S1, S2. The changes in condition for current state and inputs to detect 101 sequence are for S0 state if input=1 then next state goes to S1 and output will get 0 else next state goes to S0 and output will get 0. For S1 state if input==1 then next state goes to S1 state and output will get 0 else next state goes S2 state and output will get 0. For S2 state if input==1 then next state goes to S1 state and output will get 1 else next state goes to S0 state and output will get 0 end the case and module. The changes in test bench of mealy machine are time scale and it depends on the user to provide time scale for reset, clock, inputs sequence.

Chapter 5

Results

5.1 Mealy Machine:

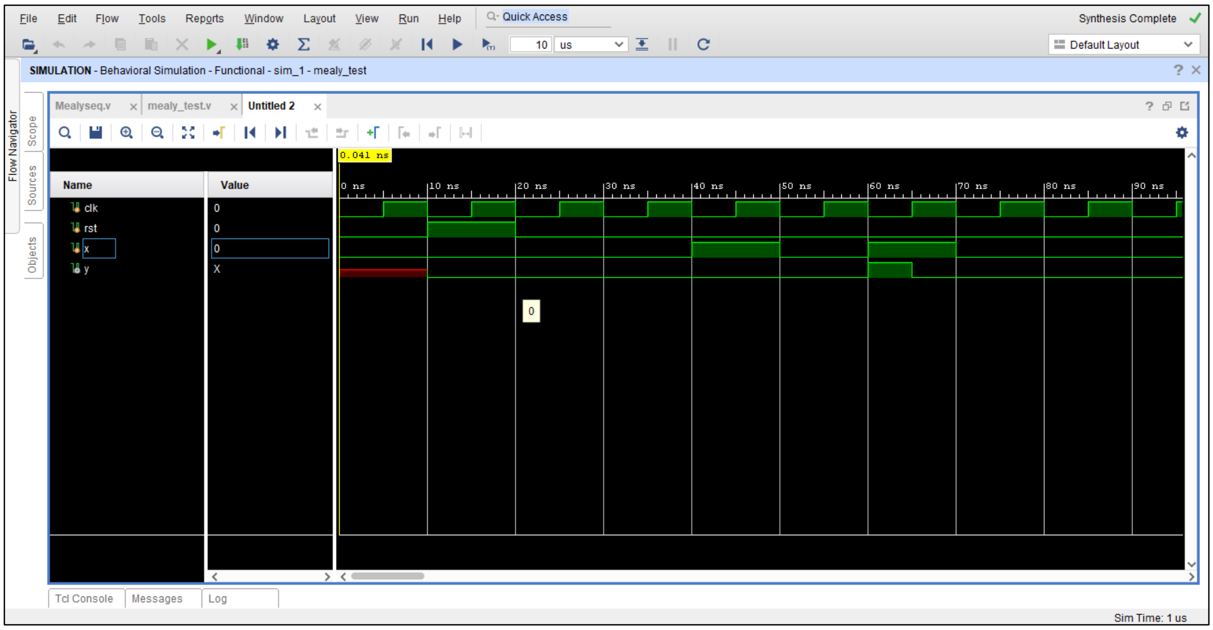

Figure 5.1 Result of Mealy Machine

The behavior of the Mealy FSM was described using Verilog HDL and simulated using Xilinx Vivado. The simulated results are shown in figure .. , the input string is 010100 and the output sequence is 000100. Here the input and output are changing with positive edge of clock pulse.

5.2 Moore's Machine:

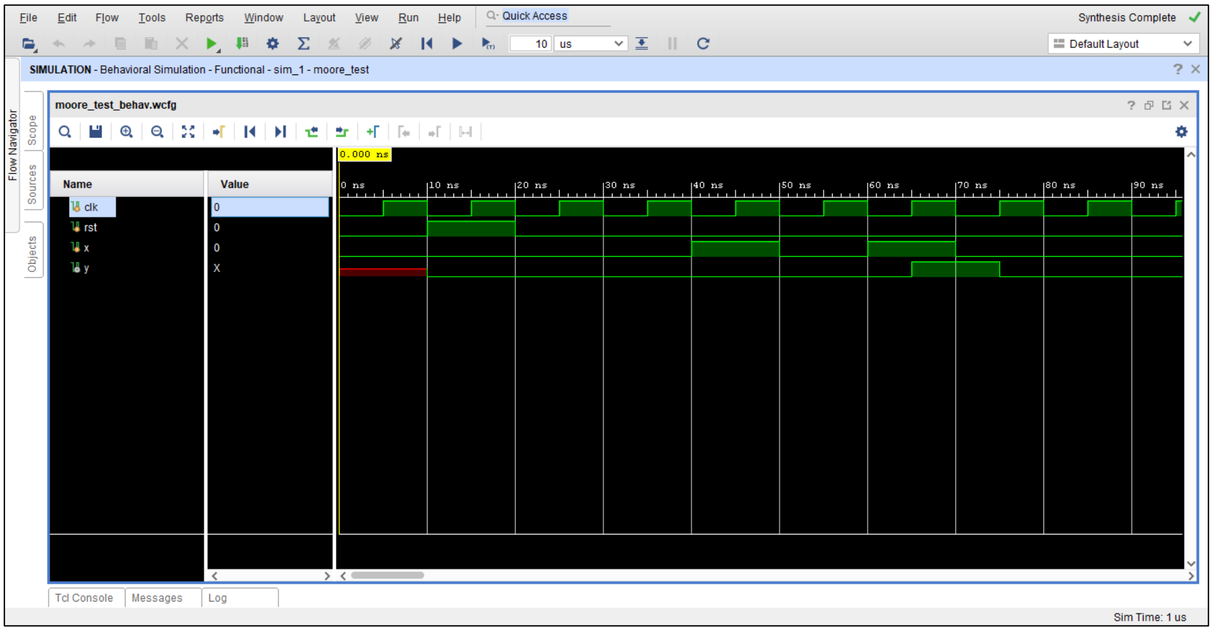
File Edit Flow Tools Reports Window Layout View Run Help
Quick Access
Synthesis Complete
Default Layout
SIMULATION - Behavioral Simulation - Functional - sim_1 - moore_test
moore_test_behav.wcfg
Name
Value
clk
rst
x
y
0.000 ns
Sim Time: 1 us

Figure 5.2 Result of Moore's Machine

The behavior of the Moore FSM was described using Verilog HDL and simulated using Xilinx Vivado. The simulated results are shown in figure .. , the input string is 010100 and the output sequence is 000100. Here the input and output are changing with positive edge of clock pulse.

5.3 **Application:**

- There are many practical scenarios where state diagrams helps in solve **tedious** questions .
- Like take the example of implementation of **Elevator** functionality using a state diagram.
- Each time you do a search (particularly a "pattern search") in your favorite editor/tool, the pattern is translated into some form of finite state machine, which does the matching.
- The lexical analysis part of your **compiler/interpreter** (yes, even your shell) is again a finite automaton which matches keywords and other tokens recognized by the language.

- Any **vending machine** is a finite automaton which takes in coins of different denominations and recognizes when the correct amount has been entered (OK, today's vending machines probably have a small CPU inside doing the adding, but the end result is the same).

Chapter 6

Conclusion and Future Scope

6.1 Conclusion

Hence, we have simulated 101 sequence detectors using Moore and mealy machine in Xilinx vivado software. The output has generated in waveform for the given inputs sequence.

• Mealy and Moore state machines are very important concepts in digital design.

• These state machine can be used in the design of mathematical algorithms.

• Mealy and Moore state machines can come in both simple (having one input and output) to complex (having many inputs and outputs) types.

6.2 Future Scope

In future, the number of states of the FSM can be increased for additional operations if required. Actuators like DC motor, stepper motor and solenoid valves can be interfaced with the FPGA to realize the real time operation and functioning of the washing machine control system.

References

- https://docs.google.com/document/d/13wEh59HTTCOkRCc-CLg7H5htm2oG_jjp/edit?usp=sharing&ouid=106883061211938960078&rtpof=true&sd=true
- http://www.warse.org/IJATCSE/static/pdf/file/ijatcse122912020.pdf
- https://www.slideshare.net/Munib6233/moore-and-mealy-machine
- https://www.xilinx.com/
- https://www.edaplayground.com/
- https://www.javatpoint.com/verilog
- https://www.geeksforgeeks.org/
- Sequence or Pattern Detector - You Tube
- https://study.com
- Design Sequence detector using mealy and moore machines
- https://youtu.be/_uaPyQDJ_PE
- https://www.intel.com/content/www/us/en/software/programmable/quartus-prime/download.html
- https://elearn.maven-silicon.com/vlsi-soc-design-using-verilog-hdl
- How to Download And Install Xilinx Vivado Design Suite? | Xilinx FPGA Programming Tutorials
- Xilinx Vivado Tutorial:1 (Basic Flow)
- https://www.fpgarelated.com/forums
- https://www.allaboutcircuits.com/forums/embedded/fpga-embedded/
- https://forum.digilent.com/forum/4-fpga/