**Moore Sequential Circuit**

**CENG 3151**

Submitted by

Michael Lanford

1816337

Computer Engineering

University of Houston-Clear Lake

Houston, Texas 77058

February 6, 2023

# Table of Contents

**1** [**Abstract 3**](#_Toc126060766)

[**2 Introduction 3**](#_Toc126060767)

**3 Requirements……………………………………………………………………….……......3**

**4 Prelab………………………………………………………………………………..……….3**

**5 Implementation……………………………………………………….…………………......4**

5.1 Design Code / Design Diagrams**……………………………………………………………5**

5.2 Schematics**……………………………………………………………………………….....7**

5.3 Testbench**…………………………………………………………………………………....7**

5.4 Waveform / Results**………………………………………………………………………..10**

[6 Conclusion **11**](#_Toc126060768)

**Abstract**

# A Moore sequential circuit is a circuit where the output depends solely on the current state of the circuit. In this lab, we will be using Xilinx Vivado to build a Moore sequential circuit that will accept some input and produce some output while satisfying the condition: the output will remain 1 or 0 until the input sequence 011 occurs, then the output will switch to the other number and remain that number until the sequence occurs again. The major results of this experiment will be a waveform that shows the correct output for each input, which will reflect our state table and state graph in the prelab.

# Introduction

For this project, we will be using Xilinx Vivado to build and test a Moore sequential circuit that will accept some input and produce some output.

1. **Requirements**

Design a Moore sequential circuit that can begin with output 0 then can become and remain 1 when the input sequence 011 occurs and continue to switch every time the sequence occurs again. The circuit has a single input labeled X along with a single output labeled Y. The figure of this circuit can be seen below:

Diagram, timeline

Description automatically generated

**Figure 1:** Diagram for the circuit to be designed.

1. **Prelab**

For this prelab, we were required to draw the state graph and state table for a Moore Sequential circuit.

1

0

S0

0

State graph:

1

S5

1

S1

0

0

1

0

0

1

0

S4

1

S2

0

0

1

S3

1

1

State table:

|  |  |  |  |
| --- | --- | --- | --- |
|  | Next State | |  |
| Present State | X=0 | X=1 | Output |
| S0 | S1 | S0 | 0 |
| S1 | S1 | S2 | 0 |
| S2 | S1 | S3 | 0 |
| S3 | S4 | S3 | 1 |
| S4 | S4 | S5 | 1 |
| S5 | S4 | S0 | 1 |

1. **Implementation**

The first step of implementation was to create a new project in Xilinx Vivado. After setting up the project with the correct options, we added a design source file and added the necessary inputs and outputs to it. We then coded the clock and reset conditionals along with each conditional for the states found in the prelab, then created a simulation file. We then added in the component instantiation, interface signal declarations, instance port map, the clock process, and the test cases to the simulation file and tested the waveform for it.

**4.1 Design Code / Design Diagrams**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

-- Input and Output declarations

entity Lab2Design is

Port ( X : in STD\_LOGIC;

Reset : in STD\_LOGIC;

Clk : in STD\_LOGIC;

Y : out STD\_LOGIC);

end Lab2Design;

architecture Behavioral of Lab2Design is

type statetype is(S0,S1,S2,S3,S4,S5);-- Defined state type

signal current\_state, next\_state: statetype; -- Declared states

begin

process(Clk, Reset) is begin

if(Reset ='1') then-- If reset is done, return to beginning state 0

current\_state <=S0;

elsif (rising\_edge(Clk)) then-- If reset is not done, continue going to the next state

current\_state <= next\_state;

end if;

end process;

process(X, current\_state) is begin-- Process to update to the next state / Replicates the state graph from the pre lab

case current\_state is

when S0 =>

if(X='0') then

next\_state <= S1;-- If 0 move to S1

else

next\_state <= S0; -- If not 0 move to S0

end if;

when S1 =>

if(X='0') then

next\_state <= S1; -- If 0 move to S1

else

next\_state <= S2; -- If not 0 move to S2

end if;

when S2 =>

if(X='0') then

next\_state <= S1; -- If 0 move to S1

else

next\_state <= S3; -- If not 0 move to S3

end if;

when S3 =>

if(X='0') then

next\_state <= S4; -- If 0 move to S4

else

next\_state <= S3; -- If not 0 move to S3

end if;

when S4 =>

if(X='0') then

next\_state <= S4; -- If 0 move to S4

else

next\_state <= S5; -- If not 0 move to S5

end if;

when S5 =>

if(X='0') then

next\_state <= S4; -- If 0 move to S4

else

next\_state <= S0; -- If not 0 move to S0

end if;

end case;

end process;

--update output: output is 1 at s3, s4, and s5

Y <= '1' when current\_state = S3 or current\_state = S4 or current\_state = S5 else '0';

end Behavioral;

**4.2 Schematics**

**Diagram, schematic

Description automatically generated**

**Figure 2:** Moore Sequential Circuit Diagram

**4.3 Testbench**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Lab2Sim is

-- Port ( );

end Lab2Sim;

architecture Behavioral of Lab2Sim is-- Instantiate the test component

component Lab2Design is

Port ( X : in STD\_LOGIC;

Reset : in STD\_LOGIC;

Clk : in STD\_LOGIC;

Y : out STD\_LOGIC);

end component;

signal X, Reset, Clk : std\_logic;-- Declare signals

signal Y : std\_logic;

constant Clk\_period : time := 10 ns;

begin

uut: Lab2Design PORT MAP (X, Reset, Clk, Y);-- Port maps

Clk\_process :process-- Clock process

begin

Clk <= '0';

wait for Clk\_period/2;

Clk <= '1';

wait for Clk\_period/2;

end process;

process-- Process used to enter the inputs given from the prelab (01011010110100111) for testing

begin

Reset <= '1';

wait for Clk\_period;

Reset <= '0';

X <= '0';

wait for Clk\_period;

X <= '1';

wait for Clk\_period;

X <= '0';

wait for Clk\_period;

X <= '1';

wait for Clk\_period;

X <= '1';

wait for Clk\_period;

X <= '0';

wait for Clk\_period;

X <= '1';

wait for Clk\_period;

X <= '0';

wait for Clk\_period;

X <= '1';

wait for Clk\_period;

X <= '1';

wait for Clk\_period;

X <= '0';

wait for Clk\_period;

X <= '1';

wait for Clk\_period;

X <= '0';

wait for Clk\_period;

X <= '0';

wait for Clk\_period;

X <= '1';

wait for Clk\_period;

X <= '1';

wait for Clk\_period;

X <= '1';

wait;

end process;

end Behavioral;

**4.4 Waveform / Results**

The waveform below shows that the code we made above in the Testbench and Design Code / Design Diagrams sections was able to produce correct results for each of our inputs that we were given in the lab instructions. When the input sequence 011 showed up, the output become 1 and stayed 1 until it showed up again and then returned to 0.

Graphical user interface

Description automatically generated

**Figure 3:** Moore Sequential Circuit Waveform

# Conclusion

In this lab, we were able to successfully code a Moore sequential circuit in Xilinx Vivado by using the state table and state graph from our prelab to create it. These programs were made to be able to simulate the circuit and take in the inputs given to us in the lab instructions and produce the correct output. The result of this simulation is shown in our waveform where you can see that for every X input we entered, the correct Y output appeared.

# 