**Mealy Sequential Circuit**

**CENG 3151**

Submitted by

Michael Lanford

1816337

Computer Engineering

University of Houston-Clear Lake

Houston, Texas 77058

February 13, 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**…………………………………………………………………………………....8**

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

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

**Abstract**

# A Mealy sequential circuit is a circuit where the output depends both on the present state of the circuit along with the input. In this lab, we will be using Xilinx Vivado to build a Mealy sequential circuit that will accept some input and produce some output while satisfying the condition: the output will be 1 for any input sequence ending in 1010, provided that the sequence 001 has occurred at least once. The major results of this experiment will be a waveform that shows the correct output for each input, which will reflect our state table in the prelab.

# Introduction

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

1. **Requirements**

Design a Mealy sequential circuit that can begin with output 0 and then become 1 when the sequence 1010 occurs, but only after the sequence 001 has already occurred. The circuit has a single input labeled X along with a single output labeled Y. The figure of this circuit can be seen below:

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 Mealy Sequential circuit.

State graph:

1/0

0/0

0/0

1/0

1/0

0/0

0/0

S4

0

S3

0

S2

0

S1

0

S0

0

1/0

1/0

0/0

0/0

1/0

S6

0

S5

1

0/1

1/0

State table:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Next State | | Output | |
| Present State | X=0 | X=1 | X=0 | X=1 |
| S0 | S1 | S0 | 0 | 0 |
| S1 | S2 | S0 | 0 | 0 |
| S2 | S2 | S3 | 0 | 0 |
| S3 | S3 | S4 | 0 | 0 |
| S4 | S5 | S4 | 0 | 0 |
| S5 | S3 | S6 | 0 | 0 |
| S6 | S5 | S4 | 1 | 0 |

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 Lab3Design is

Port ( X : in STD\_LOGIC;

Reset : in STD\_LOGIC;

clk : in STD\_LOGIC;

Y : out STD\_LOGIC);

end Lab3Design;

architecture Behavioral of Lab3Design is

type statetype is(S0, S1, S2, S3, S4, S5, S6);-- 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 <= S2; -- If 0 move to S2

else

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

end if;

when S2 => --This is where we check if we received '001' before checking for '1010'

if(X = '0') then

next\_state <= S2;--If we did not get '001', we still check for '001' until we satisfy this condition.

else

next\_state <= S3;--If we did get '001', now we check for '1010'

end if;

when S3 =>--This is where we actually begin checking for '1010' pattern.

if(X = '0') then

next\_state <= S3; -- If 0 move to S3

else

next\_state <= S4; -- If not 0 move to S4

end if;

when S4 =>

if(X = '0') then

next\_state <= S5; -- If 0 move to S5

else

next\_state <= S4; -- If not 0 move to S4

end if;

when S5 =>

if(X = '0') then

next\_state <= S3; -- If 0 move to S3

else

next\_state <= S6; -- If not 0 move to S6

end if;

when S6 =>

if(X = '0') then

next\_state <= S5; -- If 0 move to S5

else

next\_state <= S4; -- If not 0 move to S4

end if;

when others =>

next\_state <= S0;

end case;

end process;

Y <= '1' when (current\_state = S6 and X = '0') else '0';--Since we have two potential outputs, we should see 1 only when reaching S6 and when x = 0

end Behavioral;

**4.2 Schematics**

**Diagram, schematic

Description automatically generated**

**Figure 2:** Mealy Sequential Circuit Diagram

**4.3 Testbench**

library IEEE;

use IEEE.STD\_LOGIC\_1164.ALL;

entity Lab3Sim is

-- Port ( );

end Lab3Sim;

architecture Behavioral of Lab3Sim is-- Instantiate the test component

component Lab3Design 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: Lab3Design 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 (10100101010100111) for testing

begin

Reset <= '1';

wait for Clk\_period;

Reset <= '0';

X <= '1'; -- Beginning of provided input table, expect 0

wait for Clk\_period;

X <= '0';-- Expect output of 0

wait for Clk\_period;

X <= '1';-- Expect output of 0

wait for Clk\_period;

X <= '0';-- Expect output of 0

wait for Clk\_period;

X <= '0';-- Expect output of 0

wait for Clk\_period;

X <= '1';-- Expect output of 0

wait for Clk\_period;

X <= '0';-- Expect output of 0

wait for Clk\_period;

X <= '1';-- Expect output of 0

wait for Clk\_period;

X <= '0';-- Expect an output of 1 here

wait for Clk\_period;

X <= '1';-- Expect an output of 0

wait for Clk\_period;

X <= '0';-- Expect an output of 1 again here

wait for Clk\_period;

X <= '1';-- Expect output of 0

wait for Clk\_period;

X <= '0';-- Expect output of 0

wait for Clk\_period;

X <= '0';-- Expect output of 0

wait for Clk\_period;

X <= '1';-- Expect output of 0

wait for Clk\_period;

X <= '1';-- Expect output of 0

wait for Clk\_period;

X <= '1';-- Expect output of 0

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 001 occurred and was followed by the sequence 1010, the output became 1 and then returned to 0 until the next 1010 sequence occurred.

A screenshot of a computer

Description automatically generated

**Figure 3:** Combinational Circuit Waveform

# Conclusion

In this lab, we were able to successfully code a Mealy 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.

# 