# Project Based Learning Report For B. Tech Sem – VI

**Department of**

# Electronics & Communication Engineering

## Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune – 411043

**Academic Year: 2023-24**

**Project Based Learning Report**

**On,**

**Title of the topic:**

**To model sequence detector '1011' with Mealy machine using VHDL and Xilinx simulator**

Submitted in the partial fulfillment of the requirements For the Project based learning in,

( **VLSI Design Technology**)

Electronics & Communication Engineering

By

**PRN Name of student**

**2114110475 Naveen Kumar**

**2114110472 Praveen Kumar**

**2114110487 Pratyush manjul**

Under the guidance of Course In-charge Prof. Dr. Meena Chavan

Department of

Electronics Communication Engineering

Bharati Vidyapeeth (Deemed to be University) College of Engineering,

Pune – 411043

**Academic Year: 2023-24**

**Bharati Vidyapeeth (Deemed to be University) College of Engineering, Pune – 411043**

**DEPARTMENT OF**

**ELECTRONICS & COMMUNICATION ENGINEERING**

**CERTIFICATE**

Certified that the Project Based Learning report entitled, “To model sequence detector '1011' with Mealy machine using VHDL and Xilinx simulator**”** is work done by,

**PRN NAME OF STUDENT**

**2114110475 NAVEEN KUMAR**

**2114110472 PRAVEEN KUMAR**

**2114110487 PRATYUSH MANJUL**

in partial fulfillment of the requirements for the award of credits for Project Based Learning (PBL) in **VLSI Design Technology** of Bachelor of Technology Semester VI, in Electronics and Communication Engineering.

**Date : 22/3/2024**

**Prof. Dr. Meena Chavan Dr. Arundhati A. Shinde**

**Course In-charge Head of department**

**INDEX:**

|  |  |  |
| --- | --- | --- |
| **Sr no.** | **Content** | **Page no.** |
| 1. | PROBLEM STATEMENT | 1 |
| 2. | INTRODUCTION | 2 – 3 |
| 3. | MEALY MACHINE CONCEPT | 4 – 5 |
| 4. | KEY COMPONENTS | 6 |
| 5. | STATE DIAGRAM | 7 |
| 6. | SOFTWARE USED FOR SIMULATION | 8 |
| 7. | SCHEMATIC DIAGRAM | 9 |
| 8. | SIMULATION | 10 – 12 |
| 9. | CONCLUSION | 13 |
| 10. | REFERENCE | 14 |

**PROBLEM STATEMENT**

**Design a model sequence detector that detect '1011' with Mealy machine using VHDL and Xilinx simulator. We need to design a sequence detector having high speed, critical responsiveness and where asynchronous state changes are acceptable.**

**Requirement of mealy machines:**

**Fewer States: Mealy machines generally have fewer states compared to Moore machines.**

**Output Dependency: In a Mealy machine, the output changes based on both the current input and the present state. This dynamic behavior can be advantageous for certain applications.**

**Faster Response: Since the output depends on the input, Mealy machines can respond more quickly to changes in the environment.**

**Asynchronous State Changes: The state can change asynchronously due to input transitions, which can impact predictability.**

**Less Hardware: In hardware implementations, Mealy machines require less hardware in their circuits.**

1

**INTRODUCTION**

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. In a Mealy machine, output depends on the present state and the external input (x). Hence, in the diagram, the output is written outside the states, along with inputs.

Sequence detector is of two types: 

* Overlapping
* Non-Overlapping

Overlapping sequence detector:

In an overlapping sequence detector, the last bit of one sequence becomes the first bit of the next sequence. However, in a non-overlapping sequence detector, the last bit of one sequence does not become the first bit of the next sequence. In this post, we’ll discuss the design procedure for non-overlapping 101 Mealy sequence detectors.

Non-Overlapping Sequence Detector:

In contrast, a non-overlapping sequence detector does not allow overlap. It resets itself to the start state when the target sequence has been detected. For instance, after detecting the initial sequence “1101,” the detector resets and begins searching for the initial “1” of the next sequence. The last bit of one sequence does not become the first bit of the next sequence in non-overlapping detectors.

Mealy machine

A Mealy machine is a type of finite state machine (FSM) that produces an output based on both its current state and the external input. When designing a Mealy machine, we can create sequence detectors that recognize specific input sequences.

2

A Mealy machine is a machine in which output symbol depends upon the present input symbol and present state of the machine. In the Mealy machine, the output is represented with each input symbol for each state separated by /. The Mealy machine can be described by 6 tuples (Q, q0, ∑, O, δ, λ') where

1. Q: finite set of states
2. q0: initial state of machine
3. ∑: finite set of input alphabet
4. O: output alphabet
5. δ: transition function where Q × ∑ → Q
6. λ': output function where Q × ∑ →O

In mealy machine:

Outputs depend on the current state and input.

Might require fewer states as outputs are tied to transitions.

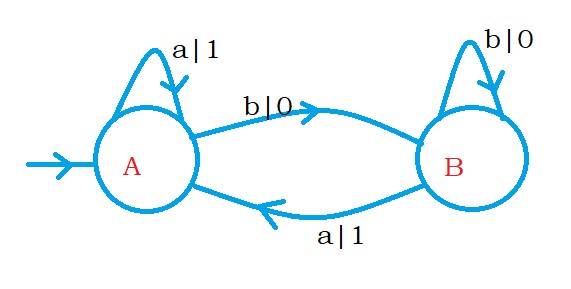
Faster response to input changes due to immediate output updates.

Can be more complex due to combined state-input cases.

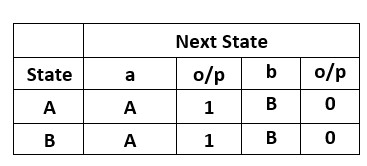
3

**MEALY MACHINE CONCEPT**

Mealy Machine: The mealy machine is a finite state machine with an output value on each transition. The below diagram shows the simple mealy machine.



We have two states, A and B. On state A, if we see the input symbol a, we move to state A and display output 1. We define output on the transition. Not on the state. On state A if we see the input symbol b, we move to state B and display output 0. The below diagram shows the transition table for the mealy machine.



Take an input string abbba. Suppose we process the input string abbba. The mealy machine will display the output 10001. We start from the initial state A, take the first input symbol a, move to state A, and display 1. Take the next input symbol, b and move to state B and display 0. Similarly, we process the remaining input symbols.

4

Example: write a mealy machine to convert a binary number to its 2’s complement.

Logic: Take a binary number 10100.

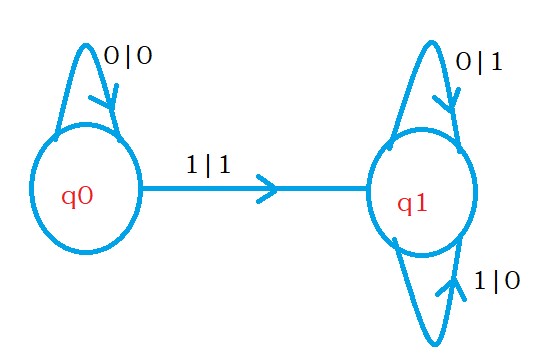
The 2’s complement of 10100 is 01100.

We move from right to left on the binary number.

We keep the binary values the same until we find the first 1.

After finding the first one, we change the bits from 0 to 1 and 1 to 0.

The below diagram shows the mealy machine to convert binary to its 2’s complement form.



On state q0, if we find the input symbol zero, we move to state q0 and display zero. On state q0, if we find the input symbol one, we move to state q1 and display 1. We are moving to state q1 when we find the first input symbol 1. We use the state q1 to change the bits 1 to 0 and 0 to 1. On state q1, if we see input symbol 0, we display 1. Similarly, for input symbol one, we display 0.

5

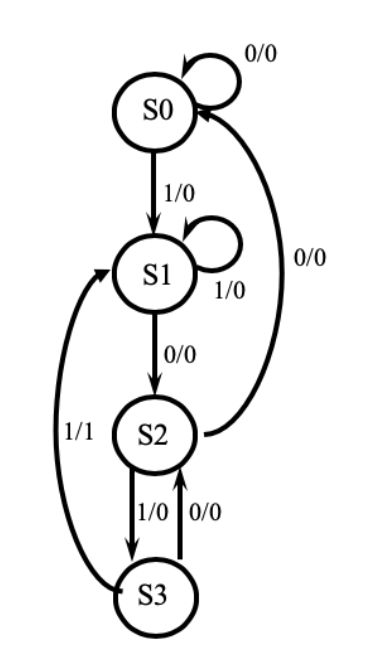
**KEY COMPONENTS:**

* States (Q): A finite set of states represents the different conditions or configurations that the machine can be in.
* Initial State (q0): The initial state from which the machine starts processing. It’s also known as the start state.
* Input Alphabet (Σ): A finite set of input symbols or characters that the machine can accept.
* Output Alphabet (O): A finite set of symbols representing the possible output values produced by the machine.
* Transition Function (δ): This function maps pairs of a state and an input symbol to the corresponding next state. In other words, it defines how the machine transitions from one state to another based on the input.
* Output Function (λ’): The output function maps pairs of a state and an input symbol to the output value produced by the machine. Unlike Moore machines, where the output depends only on the present state, Mealy machines consider both the present state and the current input symbol to determine the output.

6

**STATE DIAGRAM**

1011 state diagram by mealy overlapping.



7

**SOFTWARE USED FOR SIMULATION**

VIVADO

Vivado Design Suite is a [software suite](https://en.wikipedia.org/wiki/Software_suite) for synthesis and analysis of [hardware description language (HDL)](https://en.wikipedia.org/wiki/Hardware_description_language) designs, superseding [Xilinx ISE](https://en.wikipedia.org/wiki/Xilinx_ISE) with additional features for [system on a chip](https://en.wikipedia.org/wiki/System_on_a_chip) development and [high-level synthesis](https://en.wikipedia.org/wiki/High-level_synthesis).Vivado represents a ground-up rewrite and re-thinking of the entire design flow (compared to ISE).Like the later versions of [ISE](https://en.wikipedia.org/wiki/Xilinx_ISE), Vivado includes the in-built logic simulator.[[11]](https://en.wikipedia.org/wiki/Vivado#cite_note-11) Vivado also introduces high-level synthesis, with a toolchain that converts C code into programmable logic

Vivado was introduced in April 2012, and is an integrated design environment (IDE) with system-to-IC level tools built on a shared scalable data model and a common debug environment. Vivado includes electronic system level (ESL) design tools for synthesizing and verifying C-based algorithmic IP; standards based packaging of both algorithmic and RTL IP for reuse; standards based IP stitching and systems integration of all types of system building blocks; and the verification of blocks and systems. A free version WebPACK Edition of Vivado provides designers with a limited version of the design environment.

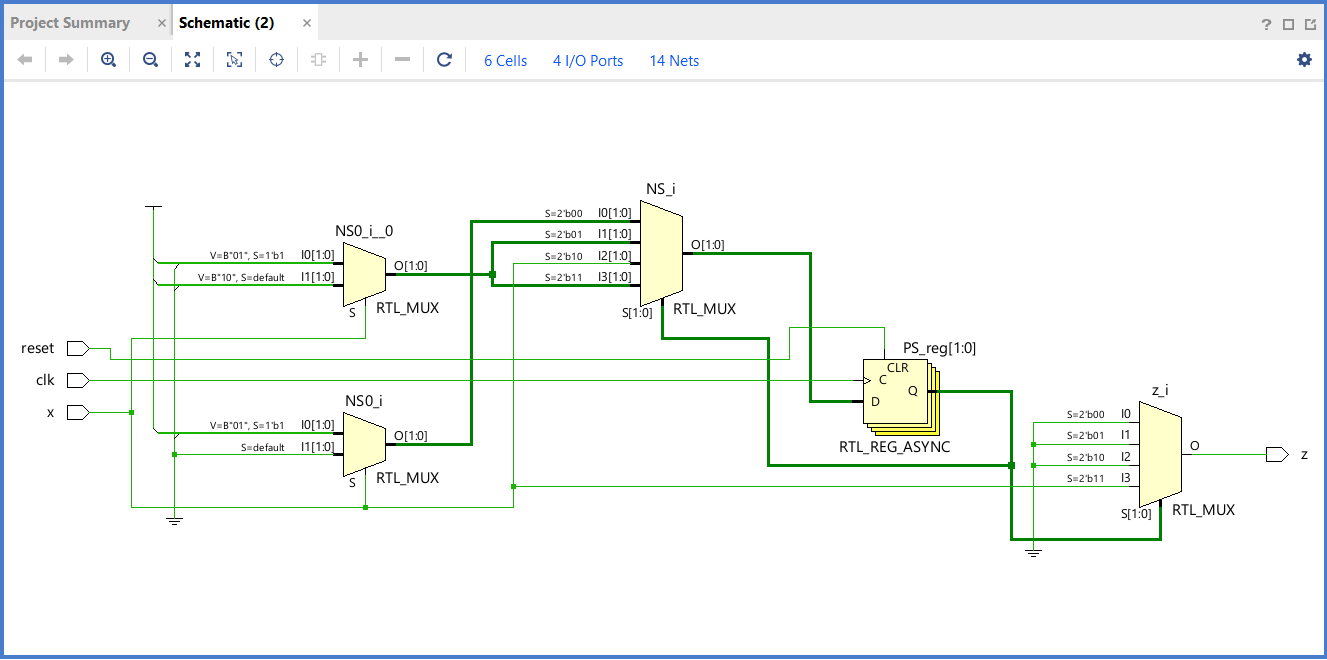
XILINX

Xilinx develops highly flexible and adaptive processing platforms that enable rapid innovation across a variety of technologies – from the cloud, to the edge, to intelligent end devices. Founded in 1984, Xilinx invented the field-programmable gate array (FPGA) and was the first fabless semiconductor company. In 2012, the company introduced the first product based on 3-D stacked silicon using silicon interposer technology.

More recently, in 2019, Xilinx introduced an adaptive computing technology that delivers breakthrough AI inference and signal processing performance. And its innovations in developer tools have fueled developers’ ambitions for more than two decades. For Xilinx, being a leader in adaptive computing is more than algorithms, artificial intelligence (AI), and machine learning. It means helping customers create scalable, differentiated and intelligent solutions that enable the adaptable, intelligent and connected world of the future.

8

**SCHEMATIC DIAGRAM**



9

**SIMULATION**

Code for design:

timescale 1ns / 1ps

module sequencedetector(

input x,clk,reset,

output reg z );

parameter S0 = 0 , S1 = 1 , S2 = 2 , S3 = 3 ;

reg [1:0] PS,NS ;

always@(posedge clk or posedge reset)

begin

if(reset)

PS <= S0;

else

PS <= NS ;

end

always@(PS or x)

begin

case(PS)

S0 : begin

z = 0 ;

NS = x ? S1 : S0 ;

$display(PS);

end

S1 : begin

z = 0 ;

NS = x ? S1 : S2 ;

$display(PS);

end

S2 : begin

z = 0 ;

NS = x ? S3 : S0 ;

$display(PS);

end

S3 : begin

z = x ? 1 : 0 ;

NS = x ? S1 : S2 ;

$display(PS);

10

end

endcase

end

endmodule

code for testbench:

timescale 1ns / 1ps

module mealyfsm;

// Inputs

reg x;

reg clk;

reg reset;

// Outputs

wire z;

// Instantiate the Unit Under Test (UUT)

sequencedetector uut (

.x(x),

.clk(clk),

.reset(reset),

.z(z) );

initial

begin

clk = 1'b0;

reset = 1'b1;

#15 reset = 1'b0;

end

always #5 clk = ~ clk;

initial begin

#12 x = 0;#10 x = 0 ; #10 x = 1 ; #10 x = 0 ;

#12 x = 1;#10 x = 1 ; #10 x = 0 ; #10 x = 1 ;

#12 x = 1;#10 x = 0 ; #10 x = 0 ; #10 x = 1 ;

#12 x = 0;#10 x = 1 ; #10 x = 1 ; #10 x = 0 ;

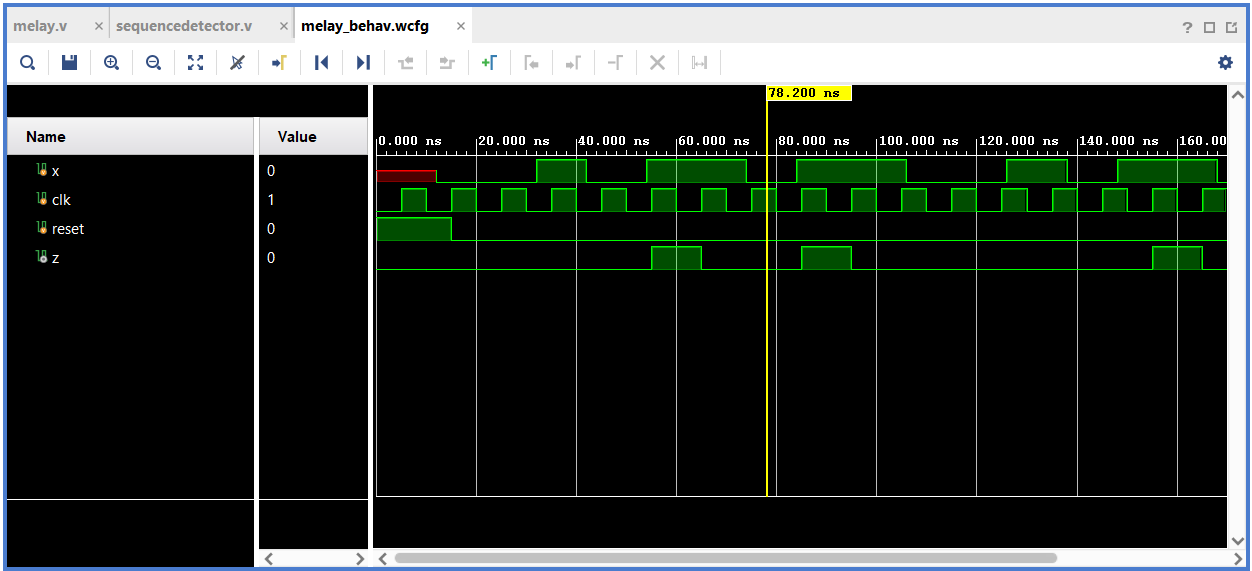
#10 $finish;

end

endmodule

11

OUTPUT :



## 12

## **CONCLUSION**

We successfully designed a Mealy sequence detector using Vivado, considering the input stream and the desired output behavior. The Mealy machine provides a flexible way to model sequential behavior and respond to specific input patterns. Mealy machines offer flexibility in terms of output behavior but require careful consideration when designing interconnected systems. We found implementation of the model sequence detector simple and easier. We understand the concept of mealy finite state machine. Their responsiveness to input changes can be both an advantage and a potential pitfall.

**Course outcome**

CO1: To understand the VLSI Design flow and design styles

Thus CO1 is justified.

CO2: To introduce the VHDL Hardware Description Langueage (HDL) for front end design implementation

Thus CO2 is justified.

13

**REFERENCE**

* <https://electronics.stackexchange.com>
* <https://www.mathworks.com/matlabcentral>
* <https://www.studytonight.com>
* https://www.electronicsforu.com/
* [www.geeksforgeeks.org](http://www.geeksforgeeks.org)
* http://www.edwardbosworth.com/

14