**Digital Logic Design EEE241**

*Lab Manual*

|  |  |
| --- | --- |
| Name | **M.Ruslan Babar** |
| Registration Number | **FA20-BSE-094** |
| Class | **BSE-2B** |
| Instructor’s Name | **SIR SAJID ALI GILAL** |

## Lab #11Mealy /Moore Machine implementation of Sequence Detector

### INTRODUCTION

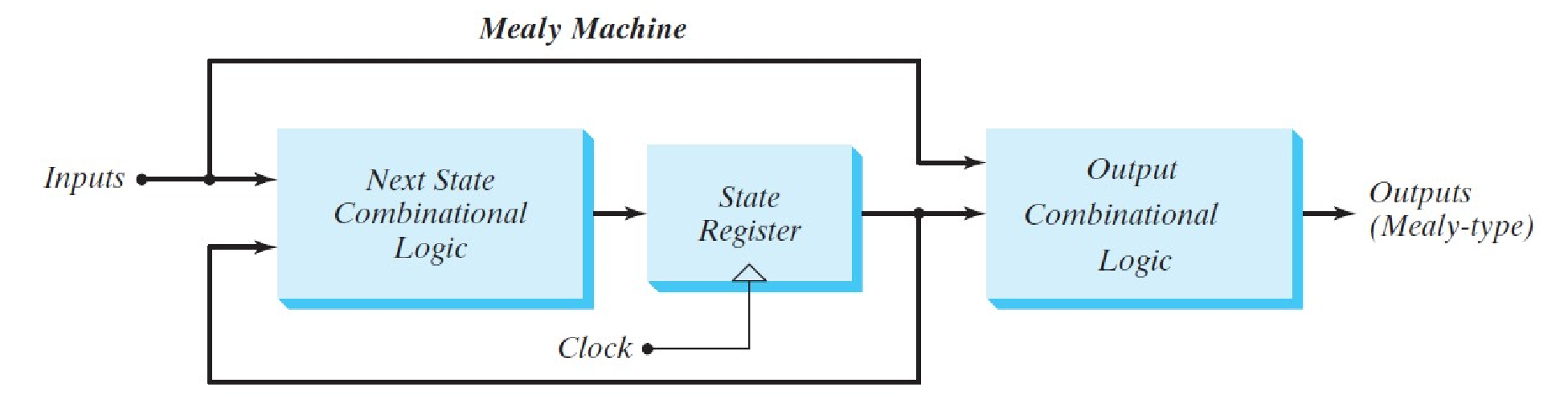
This experiment is to design behavioral description of Sequence detector using Mealy/Moore models of Finite State Machine (FSM) and implement designs into Xilinx

**Pre-Lab:**

**Background theory:**

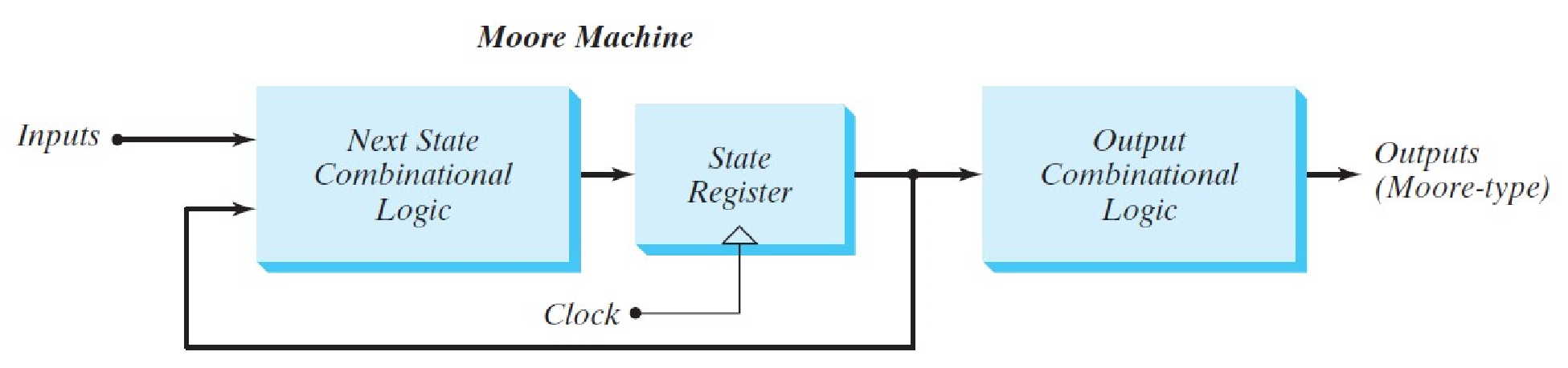
The most general model of a sequential circuit has inputs, outputs, and internal states. It is customary to distinguish between two models of sequential circuits: the Mealy model and the Moore model. They differ only in the way the output is generated. The two models of a sequential circuit are commonly referred to as a finite state machine, abbreviated FSM.

In the Mealy model, the output is a function of both the present state and the input as shown in Figure 11.1. the outputs may change if the inputs change during the clock cycle. the output of the Mealy machine is the value that is present immediately before the active edge of the clock.



#### Figure 11.2 Mealy machine

In the Moore model, the output is a function of only the present state. A circuit may have both types of outputs as shown in Figure 11. 2. The outputs of the sequential circuit are synchronized with the clock, because they depend only on flip-flop outputs that are synchronized with the clock.



**Figure 11.3 Moore machine**

#### Lab Task 2: Implementation of Moore machine

Write the HDL ( Verilog ) behavioural description for Moore based sequence detector for the given sequence. Simulate the design in Xilinx. Record the simulation output waveforms in observations.

**HDL (Verilog)**

`timescale 1ns / 1ps

module moore\_sequence\_detector(sequence\_in,clock,reset,detector\_out

);

input clock;

input reset;

input sequence\_in;

output reg detector\_out;

parameter Zero=3'b000,

One=3'b001,

OneZero=3'b011,

OneZeroOne=3'b010,

OneZeroOneOne=3'b110;

reg [2:0] current\_state, next\_state;

always @(posedge clock, posedge reset)

begin

if(reset==1)

current\_state <= Zero;

else

current\_state <= next\_state;

end

always @(current\_state,sequence\_in)

begin

case(current\_state)

Zero:begin

if(sequence\_in==1)

next\_state = One;

else

next\_state = Zero;

end

One:begin

if(sequence\_in==0)

next\_state = OneZero;

else

next\_state = One;

end

OneZero:begin

if(sequence\_in==0)

next\_state = Zero;

else

next\_state = OneZeroOne;

end

OneZeroOne:begin

if(sequence\_in==0)

next\_state = OneZero;

else

next\_state = OneZeroOneOne;

end

OneZeroOneOne:begin

if(sequence\_in==0)

next\_state = OneZero;

else

next\_state = One;

end

default:next\_state = Zero;

endcase

end

always @(current\_state)

begin

case(current\_state)

Zero: detector\_out = 0;

One: detector\_out = 0;

OneZero: detector\_out = 0;

OneZeroOne: detector\_out = 0;

OneZeroOneOne: detector\_out = 1;

default: detector\_out = 0;

endcase

end

endmodule

**TEST BENCH**

module MooreTestBench;

// Inputs

reg sequence\_in;

reg clock;

reg reset;

// Outputs

wire detector\_out;

// Instantiate the Unit Under Test (UUT)

moore\_sequence\_detector uut (

.sequence\_in(sequence\_in),

.clock(clock),

.reset(reset),

.detector\_out(detector\_out)

);

initial begin

clock = 0;

forever #5 clock = ~clock;

end

initial begin

// Initialize Inputs

sequence\_in = 0;

reset = 1;

// Wait 100 ns for global reset to finish

#30;

reset = 0;

#40;

sequence\_in = 1;

#10;

sequence\_in = 0;

#10;

sequence\_in = 1;

#20;

sequence\_in = 0;

#20;

sequence\_in = 1;

#20;

sequence\_in = 0;

end

endmodule

## Post Lab

Write a Verilog model of the Mealy FSM described by the state diagram in Figure11. 3.Develop a test bench and demonstrate that the machine state transitions and output correspond to its state diagram.

**a**

**b**

**c**

**d**

**/**

**1**

**0**

**0**

**/**

**0**

**0**

**0**

**/**

**0**

**/**

**1**

**1**

**/**

**0**

**1**

**/**

**1**

**1**

**/**

**0**

**/**

**1**

**1**

**Figure 11.4 State diagram**

**MEALY\_Verilog Module**

module mealyMachine(f, dinput, reset, clock);

input dinput, reset, clock;

output reg f;

reg [1:0] cst, nst;

parameter S0 = 2'b00,

S1 = 2'b01,

S2 = 2'b10,

S3 = 2'b11;

always @(cst or dinput)

begin

case (cst)

S0: if(dinput == 1'b1)

begin

nst = S1;

f = 1'b0;

end

else

begin

nst = cst;

f = 1'b0;

end

S1: if(dinput == 1'b0)

begin

nst = S2;

f = 1'b0;

end

else

begin

nst = cst;

f = 1'b0;

end

S2: if(dinput == 1'b1)

begin

nst = S3;

f = 1'b0;

end

else

begin

nst = S0;

f = 1'b0;

end

S3: if(dinput == 1'b0)

begin

nst = S0;

f = 1'b1;

end

else

begin

nst = S0;

f = 1'b0;

end

default: nst = S0;

endcase

end

always @(posedge clock)

begin

if (reset)

cst <= S0;

else

cst <= nst;

end

endmodule

**MEALY\_TEST BENCH**

module MealyTestBench;

// Inputs

reg dinput;

reg reset;

reg clock;

// Outputs

wire f;

// Instantiate the Unit Under Test (UUT)

mealyMachine uut (

.f(f),

.dinput(dinput),

.reset(reset),

.clock(clock)

);

initial begin

// Initialize Inputs

dinput = 0;

reset = 0;

clock = 0;

$monitor($time, , ,"c=%b",clock, , "f=%b",f, ,"r=%b",reset, ,"d=%b",dinput);

#100 dinput = 1;

#100 dinput = 1;

#100 dinput = 1;

#100 dinput = 0;

#100 dinput = 1;

#100 dinput = 0;

#100 dinput = 1;

#100 dinput = 0;

#100 dinput = 1;

//#50 reset = 1;

//#50 reset = 0;

end

always

#50 clock =~ clock;

initial

#100 $finish;

Endmodule

### Critical Analysis/Conclusion

In this lab we learn the basics of Moore/Mealy Machine. The pupose of both are given below:

* A **Moore** machine is a **finite-state machine** whose output values are determined only by its current state.
* This is in contrast to a Mealy machine, whose (Mealy) output values are determined both by its current state and by the values of its inputs.

Usage of Mealy and moore:

* Modern CPUs, computers, cell phones, digital clocks and basic electronic devices/**machines** have some kind of finite state **machine** to control it.
* They are used in processors whose values are determined by its current states

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Lab Assessment** | |  |
| **Pre Lab** | **/5** |  | **/25** |
| **Performance** | **/5** |
| **Results** | **/5** |
| **Viva** | **/5** |
| **Critical Analysis** | **/5** |
| **Instructor Signature and Comments** | | |  |