

Finite State Recognizers and Sequence Detectors

ECE 152A – Summer 2009

Reading Assignment

- **Brown and Vranesic**
 - 8 Synchronous Sequential Circuits
 - 8.4 Design of Finite State Machines Using CAD Tools
 - 8.4.1 Verilog Code for Moore-Type FSMs
 - 8.4.2 Synthesis of Verilog Code
 - 8.4.3 Simulating and Testing the Circuit
 - 8.4.4 Alternative Styles of Verilog Code
 - 8.4.5 Summary of Design Steps When Using CAD Tools
 - 8.4.6 Specifying the State Assignment in Verilog Code
 - 8.4.7 Specification of Mealy FSMs Using Verilog

Reading Assignment

- Roth

- 14 Derivation of State Graphs and Tables
 - 14.1 Design of a Sequence Detector
 - 14.2 More Complex Design Problems
 - 14.2 Guidelines for Construction of State Graphs

Mealy and Moore Machines

- Mealy Machine

- Output is a function of present state and present input
 - Outputs valid on clock edge (transition)
- Simpler (possibly)
- Faster (possibly)
- Outputs “glitch”
- Used for synchronous (clocked) designs

Mealy and Moore Machines

- **Moore Machine**

- Output is a function of present state only
 - Outputs valid after state transition
- More “stable” than Mealy machine
 - Outputs do not glitch
- Asynchronous (no clock) or synchronous designs

Deterministic Recognizers

- **State Diagram**

- Also referred to as Deterministic Transition Graph
- Next state transition is determined uniquely by present state and present input

- **Deterministic Recognizer**

- Classifies input strings into two classes:
 - Those it accepts
 - Those it rejects

Deterministic Recognizers

- **Sequential Lock Analogy**
 - Accepted string corresponds to of the combination of the lock
 - Accepted string opens the lock
 - Rejected string leaves the lock closed
- **Provides a basis for general purpose, finite state machine (FSM) design**
 - Controllers, peripheral interfaces, etc.

Deterministic Recognizers

- **Definition of states**
 - Starting (or initial) state must be defined
 - The states whose assigned output is 1 are referred to as *accepting* (or *terminal*) states
 - The states whose assigned output is 0 are called *rejecting* (or *nonterminal*) states
- **Above definition of states and control implies a Moore finite-state machine**
 - With the requirement of a defined initial state

Deterministic Recognizers

- Definition of acceptance and recognition
 - A string is accepted by a machine if and only if the state that the machine enters after having read the rightmost symbol is an accepting state
 - Otherwise, the string is rejected
 - The set of strings recognized by a machine thus consists of all the input strings that take the machine from its starting state to an accepting state

Regular Expressions

- Concerned here with the characterization of sets of strings recognized by finite automata
- A compact language for describing such sets of strings is known as the language of *regular expressions*
 - Example $01(01)^*$ describes the set consisting of those strings that can be formed by concatenating one or more 01 strings
 - $01 + 0101 + 010101 + 01010101 + \dots$

Design Example

- Design a Moore machine that recognizes the input string ending with 101
 - Any string ending in 101 will be accepted
 - Regular expression is $(1+0)^*(101)$
 - 111101 recognizes (accepts) string on sixth input
 - The machine's output goes to one each time the sequence 101 is detected
 - 10101 recognizes (accepts) string on the fifth input
 - Circuit's output goes high on third input and fifth input

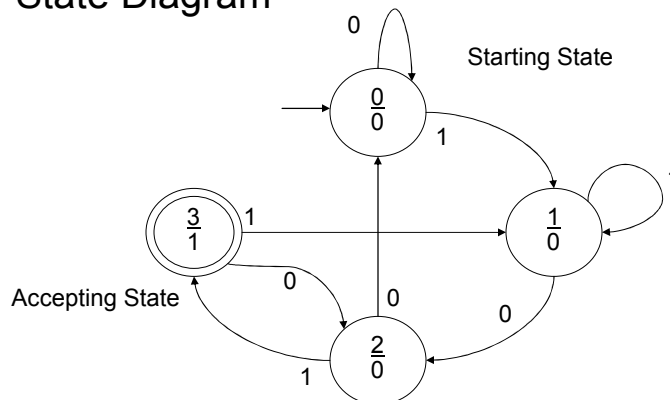
August 10, 2009

ECE 152A - Digital Design Principles

11

Design Example

- State Diagram



August 10, 2009

ECE 152A - Digital Design Principles

12

Design Example

- State table with secondary state assignment

PS	AB	NS		Z
		x=0	x=1	
0	00	00	01	0
1	01	10	01	0
2	10	00	11	0
3	11	10	01	1

Design Example

- Next State Maps

x \ AB	00	01	11	10
0		1	1	
1				1

$$A^+ = x'B + xAB'$$

x \ AB	00	01	11	10
0				
1	1	1	1	1

$$B^+ = x$$

$$z = AB \text{ (from state table)}$$

Design Example

- Design can now be implemented
 - In discrete hardware, directly from next state maps with D flip-flops or using excitation tables for T or JK flip-flops
 - In Verilog directly from state table
 - Verilog implementation follows

Moore Machine – Verilog Implementation

- Verilog Code
 - state[1] = state variable A
 - state[0] = state variable B
 - Symbolic states
 - zero, one, two, three

```
module moore (clk, x, z);
input clk, x;
output z;
reg [1:0] state;

parameter [1:0] zero = 2'b00, one = 2'b01, two = 2'b10, three = 2'b11;
assign z = state[1] & state[0];

always @ (posedge clk)
case (state)
zero :   if (x == 0)
          state <= zero;
        else
          state <= one;
one  :   if (x == 0)
          state <= two;
        else
          state <= one;
two  :   if (x == 0)
          state <= zero;
        else
          state <= three;
three:  if (x == 0)
          state <= two;
        else
          state <= one;
default: state <= zero;
endcase
endmodule
```


Conversion to Mealy Machine

- Assign Moore output (state) to Mealy transition

PS	AB	NS		Z		PS	AB	NS	
		x=0 A*B*	x=1 A*B*					x=0 A*B*, Z	x=1 A*B*, Z
0	00	00	01	0		0	00	00,0	01,0
1	01	10	01	0		1	01	10,0	01,0
2	10	00	11	0		2	10	00,0	11,1
3	11	10	01	1		3	11	10,0	01,0

August 10, 2009

ECE 152A - Digital Design Principles

19

Conversion to Mealy Machine

- Note that rows 1 and 3 of the state table are identical
 - Identical rows can be combined into a single state

PS	AB	NS		Z		PS	AB	NS	
		x=0 A*B*, Z	x=1 A*B*, Z					x=0 A*B*	x=1 A*B*
0	00	00,0	01,0	0		0	00	00,0	01,0
1	01	10,0	01,0	0		1	01	10,0	01,0
2	10	00,0	11,1	0		2	10	00,0	01,1
3	11	10,0	01,0	1					

August 10, 2009

ECE 152A - Digital Design Principles

20

Conversion to Mealy Machine

- Because outputs in a Mealy machine are associated with the transition and not the next state, states 1 and 3 can be combined
 - Call combined state “state 1” and eliminate state 3
 - New state 1 entered with output of 0 from old state 1
 - New state 1 entered with output of 1 from unchanged state 2
 - Technically, no longer a finite state recognizer because of Mealy implementation
 - No longer an acceptance “state”

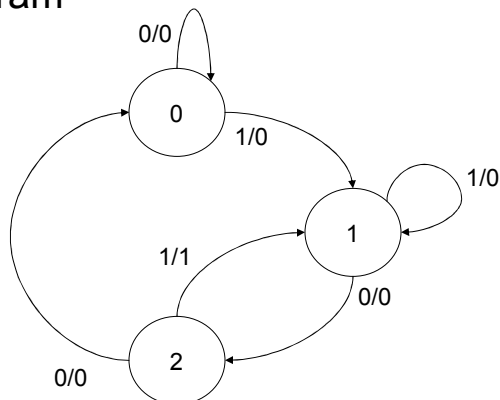
August 10, 2009

ECE 152A - Digital Design Principles

21

Conversion to Mealy Machine

- State diagram



August 10, 2009

ECE 152A - Digital Design Principles

22

Conversion to Mealy Machine

- Next state and output maps

	AB			
	00	01	11	10
x				
0		1	X	
1			X	

$$A^+ = x'B$$

$$B^+ = x$$

$$z = xA$$

	AB			
	00	01	11	10
x				
0			X	
1	1	1	X	1

	AB			
	00	01	11	10
x				
0			X	
1			X	1

August 10, 2009

ECE 152A - Digital Design Principles

23

Mealy Machine – Verilog Implementation

- Verilog Code

- Output assigned to declaratively (wire)
- Implementation with case statement

```

module mealy (clk, x, z);
    input clk, x;
    output z;
    reg [1:0] state;

    parameter [1:0] zero = 2'b00, one = 2'b01, two = 2'b10;

    assign z = x & state[1];

    always @ (posedge clk)
        case (state)
            zero: if (x == 0)
                    state <= zero;
                else
                    state <= one;
            one: if (x == 0)
                    state <= two;
                else
                    state <= one;
            two: if (x == 0)
                    state <= zero;
                else
                    state <= one;
            default:
                state <= zero;
        endcase
endmodule
    
```

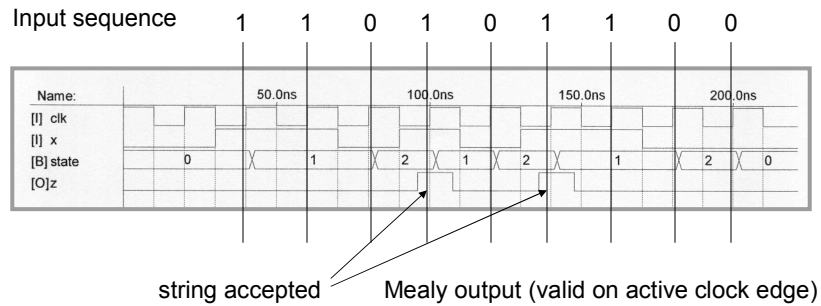
August 10, 2009

ECE 152A - Digital Design Principles

24

Mealy Machine – Verilog Implementation

■ Timing Simulation



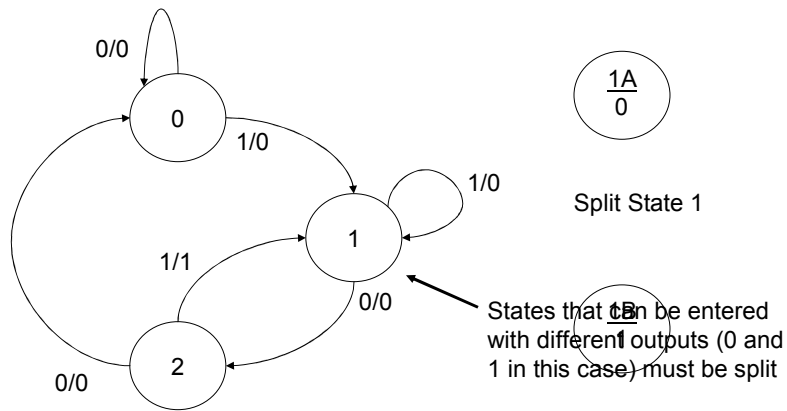
Mealy Machine – Verilog Implementation

■ Alternative Verilog Code

- Implemented directly from next state equations for state variables A and B

```
module mealy2 (clk, x, z);  
  input clk, x;  
  output z;  
  reg A,B;  
  
  assign z = x & A;  
  
  always @ (posedge clk)  
  begin  
    A <= ~x & B;  
    B <= x;  
  end  
  
endmodule
```


Conversion from Mealy to Moore

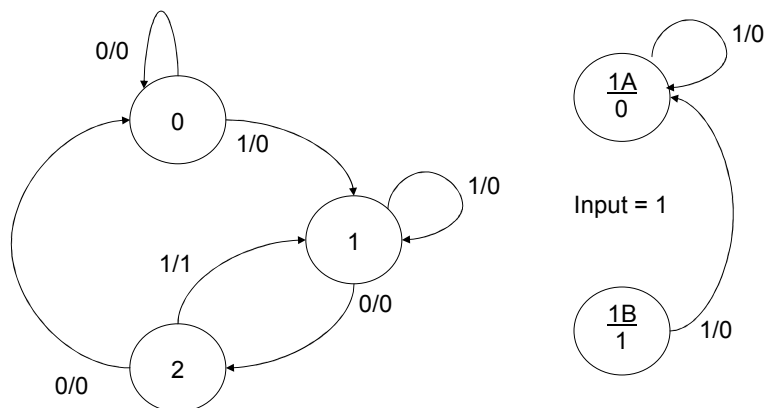


August 10, 2009

ECE 152A - Digital Design Principles

29

Conversion from Mealy to Moore

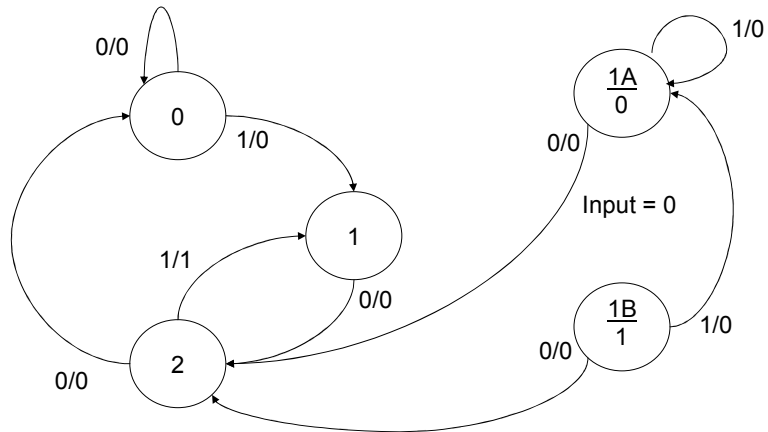


August 10, 2009

ECE 152A - Digital Design Principles

30

Conversion from Mealy to Moore

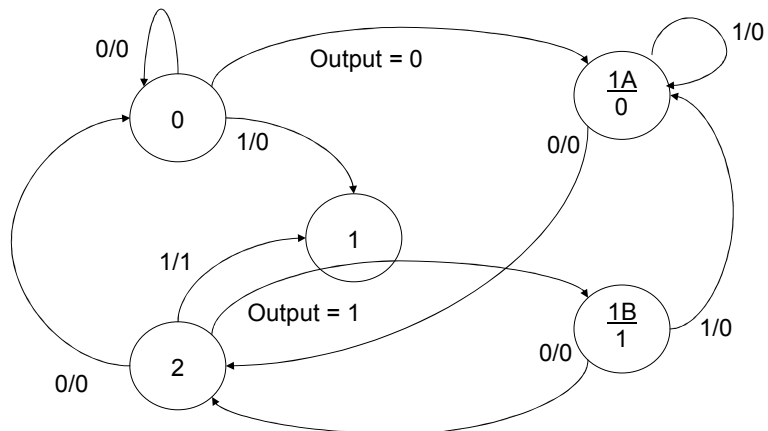


August 10, 2009

ECE 152A - Digital Design Principles

31

Conversion from Mealy to Moore



August 10, 2009

ECE 152A - Digital Design Principles

32

Conversion from Mealy to Moore

