EECS 443 Digital Systems Design
Lecture 7: More and Mealy FSM
Instructor: Yang (Cindy) Yi
Outline

• Hardware Design with RTL VHDL
• Steps of the Design Process
• Generalized FSM model
  ▫ Moore
  ▫ Mealy
• State Diagram
• Algorithmic State Machine (ASM) Charts
• VHDL Code
• Required reading
  • P. Chu,
    *FPGA Prototyping by VHDL Examples*
    *Chapter 5, FSM*

• Recommended reading
  • S. Brown and Z. Vranesic,
    *Fundamentals of Digital Logic with VHDL Design*
    *Chapter 8, Synchronous Sequential Circuits*
    *Sections 8.1-8.5*
    *Section 8.10, Algorithmic State Machine (ASM) Charts*
Review of Previous Classes - 1

Datapath (Execution Unit)

Data Inputs

Control & Status Inputs

Control Signals

Status Signals

Controller (Control Unit)

FSM generating sequences of control signals instructs datapath what to do next

Data Outputs

Control & Status Outputs

Registers

Combinational Functional Units

Busses
Review of Previous Classes - 2

- Finite State Machines can be represented using
  - **State Diagrams and State Tables** - suitable for simple controllers with a relatively few inputs and outputs
  - **Algorithmic State Machine (ASM) Charts** - suitable for complex controllers with a large number of inputs and outputs

- Example: Odd Parity Checker
- Example: Vending Machine FSM

- All of these descriptions can be easily translated to the corresponding synthesizable VHDL code
Assert output whenever input bit stream has odd # of 1's

**State Diagram**

- **Even [0]**
  - Input 0: Next State Even, Output 0
  - Input 1: Next State Odd, Output 1

- **Odd [1]**
  - Input 0: Next State Odd, Output 1
  - Input 1: Next State Even, Output 1

**Present State | Input | Next State | Output**
--- | --- | --- | ---
Even | 0 | Even | 0
Even | 1 | Odd | 0
Odd | 0 | Odd | 1
Odd | 1 | Even | 1

**Symbolic State Transition Table**

- **Present State | Input | Next State | Output**
  - 0 | 0 | 0 | 0
  - 0 | 1 | 1 | 0
  - 1 | 0 | 1 | 1
  - 1 | 1 | 0 | 1

**Encoded State Transition Table**
Review of Previous Classes - 4

- FSM is Used to model many kinds of machines
  - Vending machines
  - Delay machines
  - Binary adders
  - Language recognizer
  - Spell checking
  - Recognizing speech, etc.
Questions

• What are those two key components in the Finite State Machines?

• Please list two methods for representing the Finite State Machines.
Hardware Design with RTL VHDL

- Interface
- Pseudocode

Datapath
- Block diagram

Controller
- Block diagram
- State diagram or ASM chart

VHDL code
Steps of the Design Process

1. Text description
2. Pseudocode
3. Block diagram of the Datapath
4. Interface divided into Datapath and Controller
6. State diagram or ASM chart of the Controller
7. RTL VHDL code of the Datapath, Controller, and Top-Level Unit
8. Testbench for the Datapath, Controller, and Top-Level Unit
9. Functional simulation and debugging
10. Synthesis and post-synthesis simulation
11. Implementation and timing simulation
12. Experimental testing using FPGA board
Finite State Machines (FSMs)

- Design of FSMs involves
Generalized FSM model: Moore and Mealy

- There are two different ways to express the FSMs with respect to the output. Both have different advantages so it is good to know them.

- Combinational logic computes next state and outputs
  - Next state is a function of current state and inputs
  - Outputs are functions of
    - Current state (Moore machine)
    - Current state and inputs (Mealy machine)
Moore FSM

• Output is a Function of the Present State Only
Mealy FSM

- Output is a Function of the Present State and the Inputs
Example 10 → 01: Moore or Mealy?

- Circuits recognize $AB=10$ followed by $AB=01$
  - What kinds of machines are they?

Diagram:

- Two circuits are shown, each with inputs $A$, $B$, and a clock input.
- The output of each circuit is labeled as $out$.
- The circuits are configured to recognize the specified input sequences.
State Diagrams

- Moore Machine
- Mealy Machine
Moore FSM - Example
Mealy FSM - Example
Example: A parity checker

- Serial input string
  - OUT=1 if odd # of 1s in input
  - OUT=0 if even # of 1s in input

- Let’s do the State Diagram for Moore and Mealy

<table>
<thead>
<tr>
<th>Present State</th>
<th>Input</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Even</td>
<td>0</td>
<td>Even</td>
<td>0</td>
</tr>
<tr>
<td>Even</td>
<td>1</td>
<td>Odd</td>
<td>0</td>
</tr>
<tr>
<td>Odd</td>
<td>0</td>
<td>Odd</td>
<td>1</td>
</tr>
<tr>
<td>Odd</td>
<td>1</td>
<td>Even</td>
<td>1</td>
</tr>
</tbody>
</table>

Symbolic State Transition Table

<table>
<thead>
<tr>
<th>Present State</th>
<th>Input</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Encoded State Transition Table
Algorithmic State Machine

Algorithmic State Machine – representation of a Finite State Machine suitable for FSMs with a larger number of inputs and outputs compared to FSMs expressed using state diagrams and state tables.
Elements used in ASM charts
ASM Chart - Example 1

Moore FSM that Recognizes Sequence “10”

Mealy FSM that Recognizes Sequence “10”
ASM Chart - Example 1

Moore Machine

Mealy Machine

Recognizes Sequence “10”
ASM Charts - Example 2

Asserts the single output whenever its input string has at least two 1’s in sequence.

Mealy Machine typically has fewer states than Moore Machine for same output sequence.
Moore & Mealy FSMs without delays

Recognizes Sequence “10”
Moore & Mealy FSMs with delays

Recognizes Sequence “10”
Comparing Moore and Mealy machines
Which Way to Go?

Mealy FSM

Moore FSM
Next Class

- ASM Chart of Moore Machine
- VHDL Code of Moore Machine
- ASM Chart of Mealy Machine
- VHDL Code of Mealy Machine
- Examples

Required reading
P. Chu,
*FPGA Prototyping by VHDL Examples*
*Chapter 5, FSM*