Finite State Machines

Dr. Paul D. Franzon
Outline

- Types of Finite State Machines
- Coding Template
- Controllers
- Reset

References
1. Smith & Franzon, 8.1, 8.2 (I suggest avoiding the implicit style)
2. Ciletti, Chapters 7.1
Finite State Machine Types

- Finite State Machines can be classified by the following attributes:
  - **Moore or Mealy type outputs**

  **Moore Outputs**
  Outputs depend solely on state vector (generally, a Moore FSM is the simplest to design)

  **Mealy Outputs**
  Outputs depend on inputs and state vector (only use if it is significantly smaller or faster)
Finite State Diagrams

Moore:
- Outputs associated with state

Mealy:
- Outputs associated with input and state
... FSM Types

- **State Vector Encoding**
  - Minimal encoding
    - Minimum number of bits
  - Minimum, sequential encoding
    - Minimum number of bits and states in sequence
  - Gray encoding
    - State bit changes by only one bit between sequential states
  - One-hot encoding
    - One bit per state
      - Usually gives fastest ‘next state’ logic
## FSM Types

- What is the encoding for the first 3 states in a 7 state FSM for each style?

<table>
<thead>
<tr>
<th>State</th>
<th>Min.</th>
<th>Gray</th>
<th>1-hot</th>
</tr>
</thead>
<tbody>
<tr>
<td>S0</td>
<td>000</td>
<td>000</td>
<td>0000001</td>
</tr>
<tr>
<td>S1</td>
<td>001</td>
<td>001</td>
<td>0000010</td>
</tr>
<tr>
<td>S2</td>
<td>010</td>
<td>011</td>
<td>0000100</td>
</tr>
</tbody>
</table>
… FSM Types

- **Resets:**
  - Reset usually occurs only on power-up and when someone hits the ‘reset’ button
  - **Asynchronous** reset:
    - FSM goes to reset state whenever reset occurs
  - **Synchronous** reset:
    - FSM goes to reset state on the next clock edge after reset occurs
  - Asynchronous reset leads to smaller flip-flops while synchronous reset is ‘safer’ (noise on the reset line is less likely to accidently cause a reset).
… FSM Types

- **Fail-Safe Behavior:**
  - In case the FSM enters an ‘illegal’ state due to noise it is best to have transitions from illegal states to legal states.

- **Registered Outputs**
  - Sometimes useful to register FSM outputs – adds a stage of pipelining to the “control”
  - Or to interface with an asynchronous sub-system
Summary

- In a Moore machine, the output depends solely on the state vector
- In a Mealy machine, the output depends both on the state vector and the inputs
- State encoding affects speed and state transitions
- Reset should be applied to all FSMs

→ Quiz and then next sub-module
Outline

- Types of Finite State Machines
- Coding Template
- Controllers
- Reset
At the start of a new race (‘car’), go through the Red-Yellow-Green sequence:

**Moore Machine:**
- Nomenclature: inputs
- Inputs: car?
- On states: red yellow green

**Mealy Machine:**
- Nomenclature: inputs / outputs
- Inputs: car? / red yellow green

![Digital ASIC Design Diagram]
module traffic_light_controller (clock, reset, car, red, yellow, green);

input clock;
input reset;
input car;
output red, yellow, green;

parameter [1:0] // synopsys enum states
S0 = 2'b00,
S1 = 2'b01,
S2 = 2'b10,
S3 = 2'b11;

reg [1:0] /* synopsys enum states */ current_state, next_state;// synopsys state_vector current_state
reg red, yellow, green;

always@ (posedge clock or negedge reset)
if (!reset) current_state <= S0;
else current_state <= next_state;

/* next state logic and output logic – combined so as to share state decode logic */
always@ (current_state or car)
begin
    red = 0; yellow = 0; green = 0; /* defaults to prevent latches */
    case (current_state) // synopsys full_case parallel_case
        S0: begin
            red = 1;
            if (car) next_state = S1
            else next_state = S0;
        end
        S1: begin
            yellow = 1;
            next_state = S2;
        end
        S2: begin
            green = 1;
            next_state = S0;
        end
        default: next_state = S0;
    endcase
end
endmodule

Use this as a template for all you FSMs
**FSM Verilog Notes**

1. Code each FSM by itself in one module.
   - Exception: Might combine Mealy machine with other logic to prevent unconstrained paths

2. Separate Sequential and Combinational Logic

3. Is this reset Synchronous or Asynchronous?
   
   *Asynchronous*

4. Note use of Synthesis directives:

   ```verilog
   //synopsys enum states and //synopsys state_vector
   current_state tell Synopsys what the state vector is.
   
   You can optionally use Synopsys FSM optimization procedures
   
   // full_case ➔ Should be no latches
   // parallel_case ➔ NOT priority logic
   ```
Mealy Alternative

Exercise: Code Mealy

Mealy Machine:

Nomenclature: inputs / outputs

- car? / red yellow green

```
always@(*)
begin
  red = 0; yellow = 0; green = 0;
  case (current_state) // synopsys f
    S0:
      if (car) begin yellow=1;
        next_state=S1; end;
    else begin red=1; next_state=S0; end;
    S1:
      begin green=1; next_state=S2; end;
      S2 :
      begin red=1; next_state=S0; end;
    default : begin
      next_state = S0; red=1; end;
  endcase
end
```
One-hot alternative

Original Coding:

```vhdl
parameter [1:0] // synopsys enum states
  S0 = 2'b00,
  S1 = 2'b01,
  S2 = 2'b10,
  S3 = 2'b11;

reg [1:0] /* synopsys enum states */ current_state, next_state;
// synopsys state_vector current_state
```

Modified Coding:

```vhdl
parameter [2:0] // synopsys enum states
  S0 = 2'b001,
  S1 = 2'b010,
  S2 = 2'b100;

reg [2:0] /* synopsys enum states */ current_state, next_state;
// synopsys state_vector current_state
```

Nothing else changes!
Outline

- Types of Finite State Machines
- Coding Template
- Controllers
- Reset
Controllers

- A useful design approach is to clearly separate the datapath from the controller

- Controller styles
  - FSMs
  - Hierarchy of FSMs
  - Pipelined FSMs
  - Counters
Hierarchical FSMs

- Most chips are too large to be controlled via one FSM
- Large FSMs (>50-100 states +/-) tend to be too slow
- Might need a hierarchical controller:
Reset

- Reset is a global signal that **the designer can not modify**
- It is generally asserted on power up or a “hard” reset
- It is used to get the machine into a “known” state
- Thus it must be distributed to
  - All FSMs
  - Selected counters
  - Selected status registers
Summary

- Moore and Mealy Verilog differ in the output logic
  - Moore is generally preferred as it guarantees a flip-flop in any paths from input to output
    - Needed to help Synopsys manage timing
- Reset is a “golden” external signal used to get the FSM into a known state

→ Sub-module quiz and final quiz