Parity Checker Example

A string of bits has "even parity" if the number of 1's in the string is even.

• Design a circuit that accepts a bit-serial stream of bits and outputs a 0 if the parity thus far is even and outputs a 1 if odd:

- Can you guess a circuit that performs this function?

Formal Design Process

• State Transition Table:

<table>
<thead>
<tr>
<th>present state</th>
<th>OUT</th>
<th>IN</th>
<th>next state</th>
</tr>
</thead>
<tbody>
<tr>
<td>EVEN</td>
<td>0</td>
<td>0</td>
<td>EVEN</td>
</tr>
<tr>
<td>EVEN</td>
<td>0</td>
<td>1</td>
<td>ODD</td>
</tr>
<tr>
<td>ODD</td>
<td>1</td>
<td>0</td>
<td>ODD</td>
</tr>
<tr>
<td>ODD</td>
<td>1</td>
<td>1</td>
<td>EVEN</td>
</tr>
</tbody>
</table>

• Invent a code to represent states:
  Let 0 = EVEN state, 1 = ODD state

<table>
<thead>
<tr>
<th>present state (ps)</th>
<th>OUT</th>
<th>IN</th>
<th>next state (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</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>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Derive logic equations from table (how?):

\[ OUT = PS \]
\[ NS = PS \oplus IN \]
Formal Design Process

Logic equations from table:
\[
\text{OUT} = \text{PS} \\
\text{NS} = \text{PS} \times \text{IN}
\]

- Circuit Diagram:
  - XOR gate for ns calculation
  - DFF to hold present state
  - no logic needed for output

Finite State Machines (FSMs)

- FSM circuits are a type of sequential circuit:
  - output depends on present and past inputs
    - effect of past inputs is represented by the current state

- Behavior is represented by State Transition Diagram:
  - traverse one edge per clock cycle.

FSM Implementation

Review of Design Steps:

1. Specify circuit function (English)
2. Draw state transition diagram
3. Write down symbolic state transition table
4. Write down encoded state transition table
5. Derive logic equations
6. Derive circuit diagram
   - FFs for state
   - CL for NS and OUT
Combination Lock Example

• Used to allow entry to a locked room:
  2-bit serial combination. Example 01,11:
  1. Set switches to 01, press ENTER
  2. Set switches to 11, press ENTER
  3. OPEN is asserted (OPEN=1).
     If wrong code, ERROR is asserted (after second combo word entry).
     Press Reset at anytime to try again.

Combinational Lock STD

Symbolic State Transition Table

<table>
<thead>
<tr>
<th>RESET</th>
<th>ENTER</th>
<th>COM1</th>
<th>COM2</th>
<th>Preset State</th>
<th>Next State</th>
<th>OPEN</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>*</td>
<td>*</td>
<td>START</td>
<td>START</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>*</td>
<td>START</td>
<td>BAD1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>*</td>
<td>START</td>
<td>OK1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>*</td>
<td>*</td>
<td>OK1</td>
<td>OK1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>*</td>
<td>0</td>
<td>OK1</td>
<td>BAD2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>*</td>
<td>1</td>
<td>OK1</td>
<td>OK2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>OK2</td>
<td>OK2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>*</td>
<td>*</td>
<td>BAD1</td>
<td>BAD1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>*</td>
<td>*</td>
<td>BAD1</td>
<td>BAD2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>BAD2</td>
<td>BAD2</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>*</td>
<td>START</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Decoder logic for checking combination (01,11):

Announcements

• Exam next Tuesday (2/22) in class:
  – Special review session Monday (2/21) 5-7pm, 125 Cory
  – Check webpage for information on exam room assignment.
• Exam coverage
• HW / Quiz
**Encoded ST Table**

- Assign states:
  - START=000, OK1=001, OK2=011
  - BAD1=100, BAD2=101
- Omit reset. Assume that primitive flip-flops has reset input.
- Rows not shown have don’t cares in output. Correspond to invalid PS values.

**State Encoding**

- In general:
  - \# of possible FSM state = 2\# of FFs
- Example:
  - state1 = 01, state2 = 11, state3 = 10, state4 = 00
- However, often more than \log_2(\# of states) FFs are used, to simplify logic at the cost of more FFs.
- Extreme example is one-hot state encoding.

**FSM Implementation Notes**

- General FSM form:

```
+-------------------+          +---------+
|   inputs          |  CL      |   outputs|
+-------------------+          +---------+
    |       |          |   |       |
    |          | present state |   |          |
    |       |          |   |          |
    +-------------------+          +---------+
    |       |          |   |          |
    |          | next state  |   |          |
    |       |          |   |          |
    +-------------------+          +---------+
```

- All examples so far generate output based only on the present state:
- Commonly name Moore Machine
  (If output functions include both present state and input then called a Mealy Machine)

**State Encoding**

- One-hot encoding of states.
- One FF per state.
- Example:
  - STATE1: 001
  - STATE2: 010
  - STATE3: 100
- Why one-hot encoding?
  - Simple design procedure.
  - Often can lead to simpler and faster “next state” and output logic.
- Why not do this?
  - Can be costly in terms of FFs for FSMs with large number of states.
  - FPGAs are “FF rich”, therefore one-hot state machine encoding is often a good approach.
One-hot encoded FSM

- Even Parity Checker Circuit:

Circuit generated through direct inspection of the STD.

- In General:
  - FFs must be initialized for correct operation (only one 1)

General FSM Design Process with Verilog Implementation

Design Steps:
1. Specify circuit function (English)
2. Draw state transition diagram
3. Write down symbolic state transition table
4. Assign encodings (bit patterns) to symbolic states
5. Code as Verilog behavioral description
   - Use parameters to represent encoded states.
   - Use separate always blocks for register assignment and CL logic block.
   - Use case for CL block. Within each case section assign all outputs and next state value based on inputs. Note: For Moore style machine make outputs dependent only on state not dependent on inputs.

Moore FSMs in Verilog

```verilog
module FSM(clk, rst, in1, in2, ... out1, out2, ...);
input clk, rst, in1, in2, ...;
output out1, out2, ...;
parameter ... = ...;
reg out1, out2, ...;
reg [1:0] ps, ns;
always @(posedge clk)
    if (rst) ps <= s0;
    else ps <= ns;
always @(ps in1 in2 ...)
    case (ps)
        s1: begin
            out1 = 1'b0;
            out2 = ...;
            if (in1 && in2 ...)
                ns = s2;
            else ns = s2;
            end
        s2: begin ... end
        default: begin out1 = 1'b1; ... ns = default; end
    endcase
endmodule
```

- If you deviate from this template, do so only with care.
- For example, for some FSMs you might simplify the code by factoring out common output assignments from the cases.

  ```verilog
  always @(ps in1 in2 ...)
  begin
    out1 = 1'b0;
    out2 = ...;
    if (in1 && in2 ...)
        ns = s2;
    else ns = s2;
    end
  case (ps)
    s1: begin
        out1 = 1'b0;
        out2 = ...;
        if (in1 && in2 ...)
            ns = s2;
        else ns = s2;
    end
  default: begin ... end
  endcase
  ```

- Similar example is unlock and error outputs in combo lock.