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:

<table>
<thead>
<tr>
<th>bit stream</th>
<th>IN</th>
<th>OUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 even</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0 even</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1 odd</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1 odd</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

CLK

- Can you guess a circuit that performs this function?

Formal Design Process

- "State Transition Diagram"
  - circuit is in one of two states.
  - transition on each cycle with each new input, over exactly one arc (edge).
  - Output depends on which state the circuit is in.

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

Derive logic equations from table (how?):

- OUT = PS
- NS = PS xor IN
Formal Design Process

Logic equations from table:
OUT = PS
NS = PS xor IN

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

Review of Design Steps:
1. Circuit functional specification
2. State Transition Diagram
3. Symbolic State Transition Table
4. Encoded State Transition Table
5. Derive Logic Equations
6. Circuit Diagram

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

- FFs form state register
- number of states ≤ 2^n
- CL (combinational logic) calculates next state and output
- Remember: The FSM follows exactly one edge per cycle.

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.
**Announcements**

**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>OK1</td>
<td>OK1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>*</td>
<td>OK1</td>
<td>OK1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>*</td>
<td>OK1</td>
<td>BAD2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>OK1</td>
<td>OK2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>*</td>
<td>OK2</td>
<td>OK2</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</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):
### FSM Implementation Notes

- **General FSM form:**
  - 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).

### One-hot encoded FSM

- **Even Parity Checker Circuit:**
  - FFs must be initialized for correct operation (only one 1).
  - In General:
    - FFs must be initialized for correct operation (only one 1).
- **Ex: 3 States**
  - Simple design procedure.
  - Circuit matches state transition diagram.
  - Can be costly for FSMs with large number of states.

### State Encoding

- **One-hot encoding of states.**
  - One FF per state.
- **In general:**
  - # of possible state = $2^n$ of FFs
- **However, often more than log2(# of states) FFs are used, to simplify logic at the cost of more FFs.**
- **Extreme example is one-hot state encoding.**

### Encoded ST Table

- **Assign states:**
  - START=000, OK1=001, OK2=011
  - BAD=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.

- **What are the output functions for OPEN and ERROR?**

---

*Spring 2003 EECS150 - Lec07-FSM1 Page 13*

*Spring 2003 EECS150 - Lec07-FSM1 Page 14*

*Spring 2003 EECS150 - Lec07-FSM1 Page 15*

*Spring 2003 EECS150 - Lec07-FSM1 Page 16*