• State Transition Table:
  Let 0 = EVEN state, 1 = ODD state
  present                   next
  ODD         1     0     ODD
  EVEN       0     0    EVEN

• 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
  • 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
  • Circuit Diagram:
    – xor gate for ns calculation
    – DFF to hold present state
    – no logic needed for output

• Finite State Machines (FSMs)
  • 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 cycle.
### FSM Implementation

- FFs form state register
- number of FFs at most log2(number of states)
- CL implements calculates next state and output

### 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

- Exam Friday
  - 5-8pm, lectures 1-8, closed book/notes
- Review Session Tonight
  - 8pm, Wiz Lounge, Soda Hall
- Interesting Seminar Tomorrow
  - "The Sight & Sound of Information - Defining the Future Beyond the PC" Brian Halla, CEO, National Semiconductor Corporation

### Combinational Lock STD

### Symbolic State Transition Table

<table>
<thead>
<tr>
<th>RESET</th>
<th>ENTER</th>
<th>COM1</th>
<th>CM2</th>
<th>Next State</th>
<th>Next State</th>
<th>OPEN</th>
<th>ERROR</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>START1</td>
<td>BAD1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>START1</td>
<td>BAD1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>START1</td>
<td>BAD1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</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>1</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>1</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>0</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>1</td>
<td>OK1</td>
<td>OK1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Decoder logic for checking combination [01,11]:

### 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 care in output. Correspond to invalid PS values.
Common name Moore Machine

All examples so far generate output based only on the present state:

General FSM form:

One-hot encoded combination lock

In general:

- FFs must be initialized for correct operation (only one 1)
- FFs must be initialized for correct operation (only one 0)

Even Parity Checker Circuit:

- FFs must be initialized for correct operation (only one 1)
- FFs must be initialized for correct operation (only one 0)