Overview of Today's Lecture:
- Control for single cycle datapath
- Introduction to Finite State Machines (FSMs)
- Definition
- Example
- State transition diagram
- State encoding
- FSM realization
- PLAs and ROM implementation of FSMs.

Reading today Appendix B.10
Next time Chapter 5 multi cycle & Appendix C multicycle control
Review: Implementation of the Main Control

Putting it All Together: A Single Cycle Processor

Worst Case Timing: lw $1, $2 (offset)

Drawback of this Single Cycle Processor

Finite State Machine

Finite State Machine (Translation to English)
Example: Traffic Light Controller

Traffic light controller at an intersection.

Finite State Machine (cont.)

- Example: Traffic lights controller:
  - There are four states:
    - NG: Green light in the north-south direction.
    - NY: Yellow light in the north-south direction.
    - EG: Green light at the East-West direction.
    - EY: Yellow light at the East-West direction.
  - There are four outputs:
    - (G;R): North-South green light, East-West red light
    - (Y;R): North-South yellow light, East-West red light
    - (R;G): North-South red light, East-West green light
    - (R;Y): North-South red light, East-West yellow light
  - There are four inputs:
    - (c, c): Car at the North-South, Car at East-West
    - (c, nc) Car at North-South, No-car at East-West
    - (nc, c): No-car at North-South, Car at East-West
    - (nc, nc): No-car at North-South, No-car at East-West

Finite State Machines can be represented by a graph.
- The graph is called a State Diagram.
- The states are the nodes in the graph.
- The arcs in the graph represent state transitions.
- Each arc is labeled with the inputs that cause the transition.
- Nodes are labeled with the outputs.

FSM Example: Traffic Light

- State Transitions:

<table>
<thead>
<tr>
<th>State</th>
<th>Input</th>
<th>Next-State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>NG</td>
<td>(-;NC)</td>
<td>NG</td>
<td>(G;R)</td>
</tr>
<tr>
<td>NG</td>
<td>(-;C)</td>
<td>NY</td>
<td>(G;R)</td>
</tr>
<tr>
<td>NY</td>
<td>-</td>
<td>EG</td>
<td>(Y;R)</td>
</tr>
<tr>
<td>EG</td>
<td>(NC;-)</td>
<td>EG</td>
<td>(R;G)</td>
</tr>
<tr>
<td>EG</td>
<td>(C;-)</td>
<td>EY</td>
<td>(R;G)</td>
</tr>
<tr>
<td>EY</td>
<td>-</td>
<td>NG</td>
<td>(R;Y)</td>
</tr>
</tbody>
</table>

Format (North/South; East/West)

- means don’t care

Finite State Machine (cont.)

FSM State Diagram

Example: Traffic light Controller

State Coding

<table>
<thead>
<tr>
<th>State</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>NG</td>
<td>00</td>
</tr>
<tr>
<td>NY</td>
<td>01</td>
</tr>
<tr>
<td>EG</td>
<td>10</td>
</tr>
<tr>
<td>EY</td>
<td>11</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input Code</th>
<th>One bit for each input, input is either true or false</th>
</tr>
</thead>
<tbody>
<tr>
<td>(C;C)</td>
<td>11</td>
</tr>
<tr>
<td>(C;NC)</td>
<td>10</td>
</tr>
<tr>
<td>(NC;C)</td>
<td>01</td>
</tr>
<tr>
<td>(NC;NC)</td>
<td>00</td>
</tr>
</tbody>
</table>

Enumerate States

<table>
<thead>
<tr>
<th>Output Code</th>
<th>One bit per color for each light</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R;G)</td>
<td>000100</td>
</tr>
<tr>
<td>(G;R)</td>
<td>100001</td>
</tr>
<tr>
<td>(Y;R)</td>
<td>010001</td>
</tr>
<tr>
<td>(R;Y)</td>
<td>001010</td>
</tr>
</tbody>
</table>
### Coded State Diagram

- **NG**: Next Green
- **NY**: Next Yellow
- **EG**: End Green
- **EY**: End Yellow
- **O = (R;G)**
- **I = (-- ; NC)**

### Finite State Machine Realization

- Finite state machines can be implemented in digital hardware by selecting binary coding for the FSM.
- Use registers to hold the state.
- Use combinational logic or PLAs, or read-only Memory (ROM) to implement the transition function.

### Example: Traffic Light Controller

<table>
<thead>
<tr>
<th>Input</th>
<th>State</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>01</td>
<td>01</td>
<td>012345</td>
</tr>
<tr>
<td>0-</td>
<td>00</td>
<td>00</td>
<td>100001</td>
</tr>
<tr>
<td>1-</td>
<td>00</td>
<td>01</td>
<td>100001</td>
</tr>
<tr>
<td>--</td>
<td>01</td>
<td>10</td>
<td>010001</td>
</tr>
<tr>
<td>-0</td>
<td>10</td>
<td>10</td>
<td>001100</td>
</tr>
<tr>
<td>-1</td>
<td>11</td>
<td>00</td>
<td>001010</td>
</tr>
</tbody>
</table>

### Programmable Logic Array (PLA)

- The PLA has N inputs, K outputs and M product terms.
- Each input or its complement may be used in any product term.
- Any product term can be used in the sum.
- The PLA is "programmed" once by making connections (or putting a transistor) at the wires intersections.

### Read Only Memory (ROM) Implementation

- Read Only Memory (ROM) is programmed at manufacturing time.
- Programmable ROM can be electrically programed (EPROM).
- To implement FSM with k-inputs, N-bits of state, M-Outputs:
  - Connect the Inputs and State bits to the ROM address lines.
  - Connect Register to the ROM output.
  - Feed back the Next-State bits of the register into the State inputs.
  - Needs: $2^{(N+M)}$ words ROM. Each word at least (N+M) bits wide.
A Simple Arrow FSM

- Consider those flashing arrow signs
- No light, one light, two lights, three lights
- Let's design the FSM to control this sign