Finite State Machines & Hardware Control Language

Avin R. Lebeck

Some slides based on those developed by Gershon Kedem, and by Randy Bryant and Dave O'Hallaron

Administrivia

- Homework #4 is up, due Mar 14
- Read Sections 4.1-4.3 of text
- Today
  - Finite State Machines
  - Hardware Control Language (if time)
Review: Abstraction: The ALU

- General structure
- Two operand inputs
- Control inputs

- We can build circuits for
  * Multiplication
  * Division
  * They are more complex
Review: Set-Reset Latch (Continued)

![Set-Reset Latch Diagram](image)

Review: Address Decode Circuit

![Address Decode Circuit Diagram](image)
Review: Register File (Four 4-bit Registers)

Finite State Machine

- \( S = \{ s_0, s_1, \ldots s_{n-1} \} \) is a finite set of states.
- \( I = \{ i_0, i_1, \ldots i_{k-1} \} \) is a finite set of input values.
- \( O = \{ o_0, o_1, \ldots o_{m-1} \} \) is a finite set output values.

**Definition:** A finite state machine is a function \( F: (S \times I) \rightarrow (S \times O) \) that gets a sequence of input values \( i_k \in I, k = 0, 1, 2, \ldots \) and it produces a sequence of output values \( O_k \in O, k = 1, 2, \ldots \) such that:

\[
F(s_k, i_k) = (s_{k+1}, o_{k+1}) \quad k = 0, 1, 2, \ldots
\]
Finite State Machine
(Translation to English)

- **Finite State Machine** is:
  - A machine with a finite number of possible states.
  - A machine with a finite number of possible inputs.
  - A machine with a finite number of possible different outputs.

- At each period (Clock cycle) the machine receives an input and it produces an output.
- The output is a function of the machine input and current state.
- After each period the machine changes state.
- The new state is a function of the input and current state.

**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;Y): North-South red light, East-West yellow light
  - (R;G): North-South red light, East-West green 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

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>

- means don’t care

Format
(North/South; East/West)
Finite State Machine (cont.)

- 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 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</th>
<th>Code</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</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>(R;G)</td>
<td>001100</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

One bit for each Input
Input is either true or false

One bit per color for each light
GYRGYR
(North; East)
Example: Traffic Light Controller

S = State, bits are S0 and S1
NS = Next State, bits are NS0 and NS1

\[
\begin{align*}
\text{IN} & \quad \text{S} & \quad \text{NS} & \quad \text{OUT} \\
0 & \quad 01 & \quad 01 & \quad 010001 \quad \text{0- 00 00 100001} \\
1 & \quad 00 & \quad 01 & \quad 100001 \quad \text{1- 00 01 100001} \\
-- & \quad 01 & \quad 10 & \quad 010001 \\
-0 & \quad 10 & \quad 10 & \quad 001100 \\
-1 & \quad 10 & \quad 11 & \quad 001100 \\
-- & \quad 11 & \quad 00 & \quad 001010 \\
\end{align*}
\]

\[
\begin{align*}
\text{NS1} &= S0' \cdot S1' \cdot I0 + S0 \cdot S1' \cdot I1 \\
&= S1' \cdot (S0' \cdot I0 + S0 \cdot I1) \\
\text{NS0} &= S0' \cdot S1 + S0 \cdot S1' \cdot I1' + S0 \cdot S1' \cdot I1 \\
&= S0' \cdot S1 + S0 \cdot S1' \\
\text{OUT0} &= S0' \cdot S1' \\
\text{OUT1} &= S0' \cdot S1 \\
\text{OUT2} &= S0' \cdot S1' + S0 \cdot S1 = S0 \\
\text{OUT3} &= S0 \cdot S1' \\
\text{OUT4} &= S0 \cdot S1 \\
\text{OUT5} &= S0' \cdot S1' + S0' \cdot S1 = S0'
\end{align*}
\]

Traffic Controller FSM implementation
General Method for FSM design

- Determine the problem:
  1. Draw the state diagram,
  2. Write the truth table,
  3. Write sum-of-products equations

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
Sequence Recognizer/Combination Lock

- A digital lock system must recognize three number input in the correct order. Must restart if incorrect number is entered.

- Design the finite state machine to recognize the combination: 8, 14, 5 (assume you have three separate inputs that indicate if the current input is equal to 8, 14, or 5, respectively)

Bit Equality

- Generate 1 if a and b are equal
- Hardware Control Language (HCL)
  - Very simple hardware description language
    - Boolean operations have syntax similar to C logical operations
  - We’ll use it to describe control logic for processors

HCL Expression

```c
bool eq = (a&&b)||(a&&!b)
```
**Word Equality**

- **Word-Level Representation**
  - bool Eq = (A == B)

- **HCL Representation**
  - `bool Eq = (A == B)`

- **32-bit word size**
- **HCL representation**
  - Equality operation
  - Generates Boolean value

**Bit-Level Multiplexor**

- **HCL Expression**
  - `bool out = (s&&a)||(!s&&b)`

- **Control signal s**
- **Data signals a and b**
- **Output a when s=1, b when s=0**
**Word Multiplexor**

Word-Level Representation

![Diagram of Word Multiplexor](image)

HCL Representation

```c
int Out = [
  s : A;
  1 : B;
];
```

- Select input word A or B depending on control signal s
- HCL representation
  - Case expression
  - Series of test : value pairs
  - Output value for first successful test

**HCL Word-Level Examples**

**Minimum of 3 Words**

```c
int Min3 = [
  A < B && A < C : A;
  B < A && B < C : B;
  1     : C;
];
```

- Find minimum of three input words
- HCL case expression
- Final case guarantees match

**4-Way Multiplexor**

```c
int Out4 = [
  #s1&&!s0: D0;
  !s1    : D1;
  !s0    : D2;
  1      : D3;
];
```

- Select one of 4 inputs based on two control bits
- HCL case expression
- Simplify tests by assuming sequential matching
Random-Access Memory

- Stores multiple words of memory
  - Address input specifies which word to read or write
- Register file
  - Holds values of program registers
  - %eax, %esp, etc.
  - Register identifier serves as address
    - ID 8 implies no read or write performed
- Multiple Ports
  - Can read and/or write multiple words in one cycle
    - Each has separate address and data input/output

Register File Timing

- Reading
  - Like combinational logic
  - Output data generated based on input address
    - After some delay
- Writing
  - Like register
  - Update only as clock rises
Summary

Finite State Machines
- Inputs, Current State
- Compute Outputs and Next State

HCL
- Language to express boolean logic
- Also, word level functions

- Homework #4