Overview

◆ Last lecture
  ■ Introduction to finite-state machines
    🔄 Example: A sequence detector FSM
    🔄 Example: A vending machine FSM

◆ Today
  ■ A bigger example
    🔄 Ant-brain FSM
Ant in a maze

- Electronic ant, electronic maze
  - Design the ant
Example: ant brain (Ward, MIT)

- **Sensors:** L and R antennae, 1 if in touching wall
- **Actuators:** F - forward step, TL/TR - turn left/right slightly
- **Goal:** find way out of maze
- **Strategy:** keep the wall on the right
Example: ant brain (special case 1)

- Left (L) Antenna touching the wall
Example: ant brain (special case 2)

- Ant Lost
Example: ant brain (special case 2)

- Ant Lost (another example)
Ant behavior

A: Following wall, touching
   Go forward, turning
   left slightly

B: Following wall, not touching
   Go forward, turning right
   slightly

C: Break in wall
   Go forward, turning
   right slightly

D: Hit wall again
   Back to state A

E: Wall in front
   Turn left until...

F: ...we are here, same as
   state B

G: Turn left until...

LOST: Forward until we
      touch something
Goal: Find a way out of maze

- **Sensors on L and R antennae**
  - Sensor = “1” if touching wall; “0” if not touching wall
    - L'R' ≡ no wall
    - L'R ≡ wall on right
    - LR' ≡ wall on left
    - LR ≡ wall in front
    - *** ≡ exit

- **Movement:**
  - F ≡ forward one step
  - TL ≡ turn left 90 degrees
  - TR ≡ turn right 90 degrees
Notes & strategy

- **Notes**
  - Maze has no islands
  - Corridors are wider than ant
  - Don’t worry about startup
  - Assume a Moore machine
  - Assume D flip-flops

- **Strategy**
  - Partition your design into datapath and control
  - Keep the wall on the right
The ant’s behavior

S0: Lost
   Go forward

S1: Right antenna touching
    Go forward

S2: Break in wall
    Turn right

S3: Left antenna touching
    Turn left

Reset
Done Flag

exit

CSE370, Lecture 22
Can have multiple walls

Example: 00001100
⇒ Walls on South and East

The maze

● Virtual maze
  ■ 128 × 128 grid
    ➔ Stored in memory
    ➔ 16384 8-bit words
  ■ YX is maze addresses
    ➔ X is the ant’s horizontal position (7 bits)
    ➔ Y is the ant’s vertical position (7 bits)
  ■ Each memory location says
    ➔ 00000001 ≡ No wall
    ➔ 00000010 ≡ North wall
    ➔ 00000100 ≡ West wall
    ➔ 00001000 ≡ South wall
    ➔ 00010000 ≡ East wall
    ➔ 00100000 ≡ Exit
Where do you start???

Don’t look ahead
What you need

- An FSM for the ant
  - 3 outputs
    - Go forward
    - Turn left
    - Turn right

- Two 7-bit registers for $X$ and $Y$
  - With preload, increment, decrement

- A register to hold the ant’s heading

- Logic to convert memory data to antennae info
Recommendations

- 7-bit counters for $X$, $Y$
  - Move horizontally: Increment or decrement $X$
  - Move vertically: Increment or decrement $Y$

- Shift register for heading
  - N: 0001
  - W: 0010
  - S: 0100
  - E: 1000
  - Rotate right when ant turns right
  - Rotate left when ant turns left

- Combinational logic for antennae decoder
Partition the design

Ant-Brain FSM
- Forward
- Turn right
- Turn left

Antennae logic
- L
- R

SRAM
- Maze
- Data

Heading (shift register)
- North
- South
- East
- West

X counter
- Increment
- Decrement
- Preload

Y counter
- Increment
- Decrement
- Preload

SRAM Address
Design the ant-brain FSM

1. State diagram and state-transition table
2. State minimization
3. State assignment (or state encoding)
4. Minimize next-state logic
5. Implement the design
Step 1a: State diagram

S1: Right antenna touching
   *Go forward*

S2: Break in wall
   *Turn right*

S3: Left antenna touching
   *Turn left*

S0: Lost
   *Go forward*

Reset
   *Done Flag*

L’ R’

L’ R

L

R

exit

exit

L’ R

L’ R’

CSE370, Lecture 22
## Step 1b: State-transition table

<table>
<thead>
<tr>
<th>Exit</th>
<th>State</th>
<th>L</th>
<th>R</th>
<th>Next State</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Reset</td>
<td>0</td>
<td>0</td>
<td>S0</td>
<td>F</td>
</tr>
<tr>
<td>0</td>
<td>S0</td>
<td>0</td>
<td>1</td>
<td>S1</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>S3</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>S3</td>
<td>F</td>
</tr>
<tr>
<td>0</td>
<td>S1</td>
<td>0</td>
<td>0</td>
<td>S2</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>S1</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>S3</td>
<td>F</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>S3</td>
<td>F</td>
</tr>
<tr>
<td>0</td>
<td>S2</td>
<td>0</td>
<td>0</td>
<td>S0</td>
<td>TR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>S0</td>
<td>TR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>S0</td>
<td>TR</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>S0</td>
<td>TR</td>
</tr>
<tr>
<td>0</td>
<td>S3</td>
<td>0</td>
<td>0</td>
<td>S1</td>
<td>TL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
<td>1</td>
<td>S1</td>
<td>TL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>0</td>
<td>S3</td>
<td>TL</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>1</td>
<td>S3</td>
<td>TL</td>
</tr>
</tbody>
</table>
Step 2: State minimization

- Two states are equivalent if they cannot be distinguished at the outputs of the FSM
  - The outputs are the same for any input sequence

- Two conditions for two states to be equivalent
  1) Outputs must be the same in both states
  2) Machine must transition to equivalent states for all inputs

- Any equivalent states in our state diagram?
Step 3: State encoding

<table>
<thead>
<tr>
<th>Exit</th>
<th>X Y L R</th>
<th>X+ Y+</th>
<th>F TL TR</th>
<th>S0 ➔ 00</th>
<th>S1 ➔ 01</th>
<th>S2 ➔ 10</th>
<th>S3 ➔ 11</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Reset</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0 0 0 0</td>
<td>0 0</td>
<td>1 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 0 0 1</td>
<td>0 1</td>
<td>1 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 0 1 0</td>
<td>1 1</td>
<td>1 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 0 1 1</td>
<td>1 1</td>
<td>1 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0 1 0 0</td>
<td>1 0</td>
<td>1 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 1 0 1</td>
<td>0 1</td>
<td>1 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 1 1 0</td>
<td>1 1</td>
<td>1 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>0 1 1 1</td>
<td>1 1</td>
<td>1 0 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1 0 0 0</td>
<td>0 0</td>
<td>0 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 0 0 1</td>
<td>0 0</td>
<td>0 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 0 1 0</td>
<td>0 0</td>
<td>0 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 0 1 1</td>
<td>0 0</td>
<td>0 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1 1 0 0</td>
<td>0 1</td>
<td>0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 1 0 1</td>
<td>0 1</td>
<td>0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 1 1 0</td>
<td>1 1</td>
<td>0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1 1 1 1</td>
<td>1 1</td>
<td>0 1 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Step 4: Minimize the logic
Step 5: Implement the design

Ant-Brain FSM

\[ X^+ = LY + L'X + X'Y'R' \]
\[ Y^+ = XY + X'R + X'L \]
\[ F = X' \]
\[ TL = XY \]
\[ TR = XY' \]

Forward

Turn right

Turn left

L
R

Heading (shift register)

North
South
East
West

Forward
 East
 West
 Preload

North
South
Preload

X counter

Y counter

SRAM

Maze Data

Antennae logic

SRAM Address

N
S
E
W

L
R

CSE370, Lecture 22
Antennae logic

Each memory location says
- 00000001 ≡ No wall
- 00000010 ≡ North wall (NW)
- 00000100 ≡ West wall (WW)
- 00001000 ≡ South wall (SW)
- 00010000 ≡ East wall (EW)
- 00100000 ≡ Exit

The ant can be heading
- N: 0001
- W: 0010
- S: 0100
- E: 1000

Logic for right antennae
\[ R = NW(N + W) + WW(W + S) + SW(S + E) + EW(E + N) \]

Logic for left antennae
\[ L = NW(N + E) + WW(W + N) + SW(S + W) + EW(E + S) \]

Gate count:
- 4 2-input ORs
- 8 2-input ANDs
- 2 4-input ORs
What we left out...

- **Crumbs in cell**
  - Ant eats crumbs in every cell it visits
  - Writes crumb file back to SRAM
  - Read crumb file, for future display on monitor

- **Need a memory controller**
  - A state machine to talk to the SRAM

- **Need to deal with startup, exit states!**
Extra Credit:

- Design the memory controller:

  ![Diagram](image)

- Due last day in class, Friday, March 14; printouts only
- Value: up to 1 quiz
- Graded on clarity and completeness of explanation
- No questions will be answered