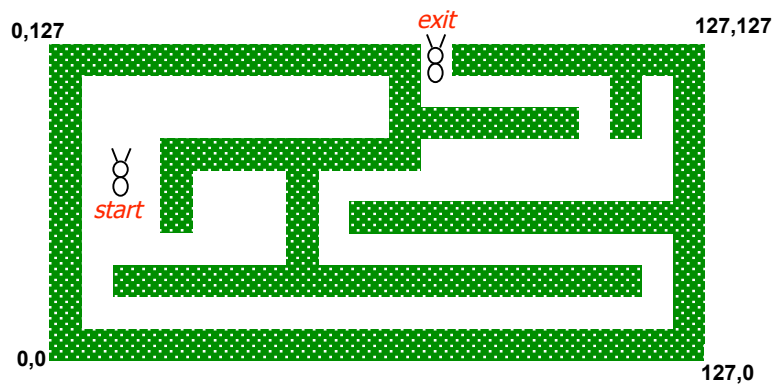


Overview

- ◆ Last lecture
 - Introduction to finite-state machines
 - ⇒ Example: A sequence detector FSM
 - ⇒ Example: A vending machine FSM
- ◆ Today
 - A bigger example
 - ⇒ Ant-brain FSM

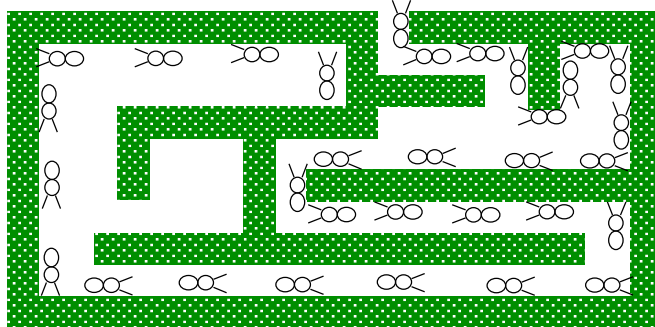
Ant in a maze

- ◆ 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

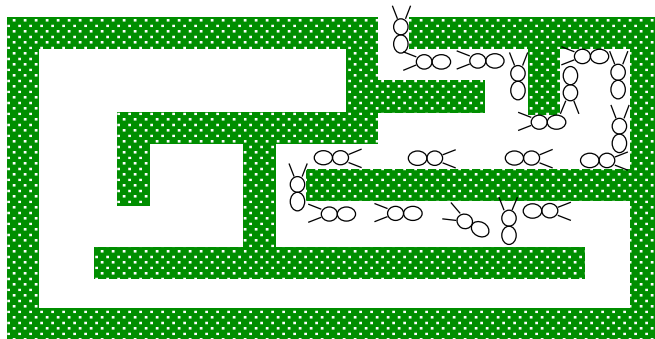


CSE370, Lecture 22

3

Example: ant brain (special case 1)

- ◆ Left (L) Antenna touching the wall

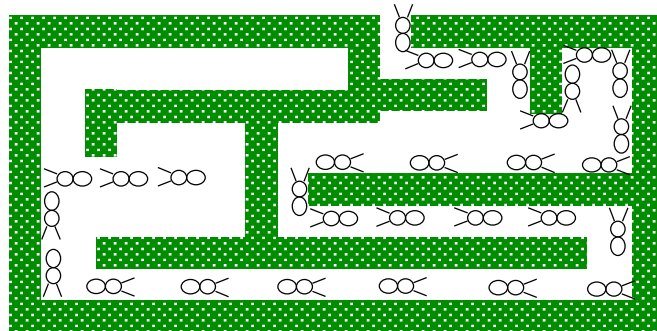


CSE370, Lecture 22

4

Example: ant brain (special case 2)

- ◆ Ant Lost

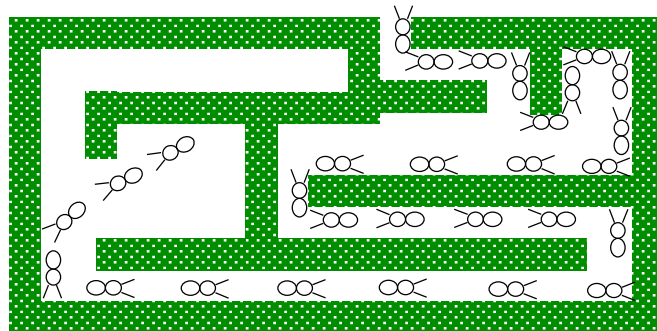


CSE370, Lecture 22

5

Example: ant brain (special case 2)

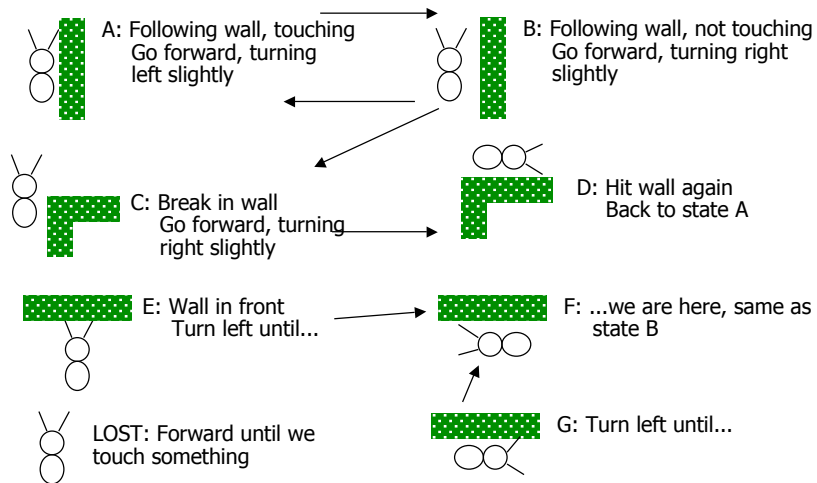
- ◆ Ant Lost (another example)



CSE370, Lecture 22

6

Ant behavior



CSE370, Lecture 22

7

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

CSE370, Lecture 22

8

Notes & strategy

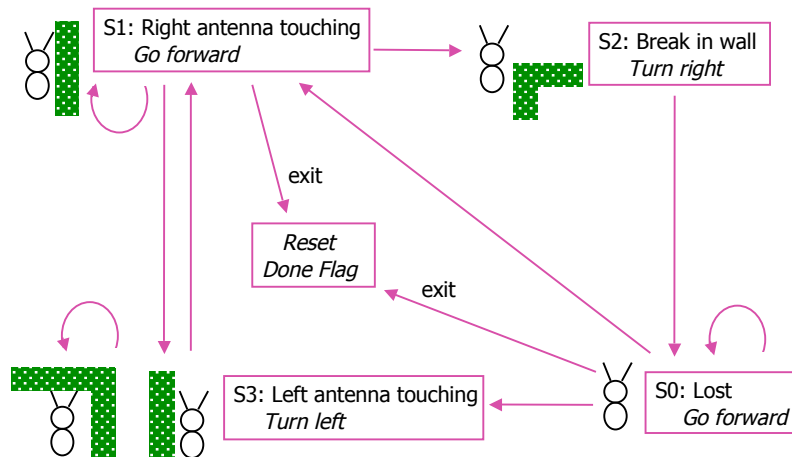
◆ Notes

- 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



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

Can have multiple walls
Example: 00001100
⇒ Walls on South and East

Where do you start???

Don't look ahead

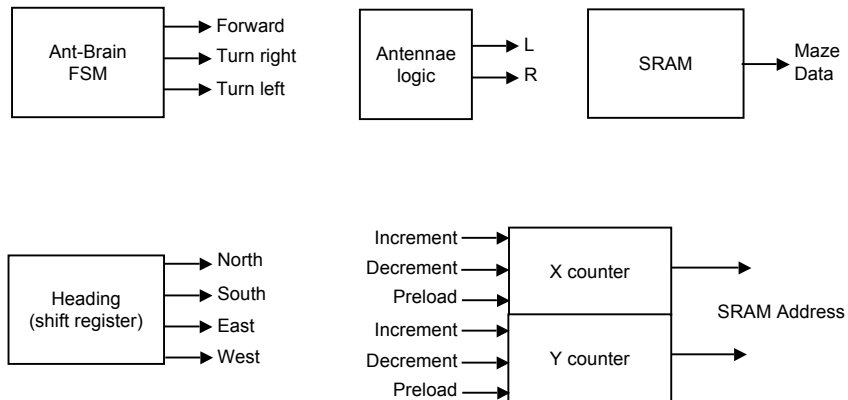
What you need

- ◆ 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

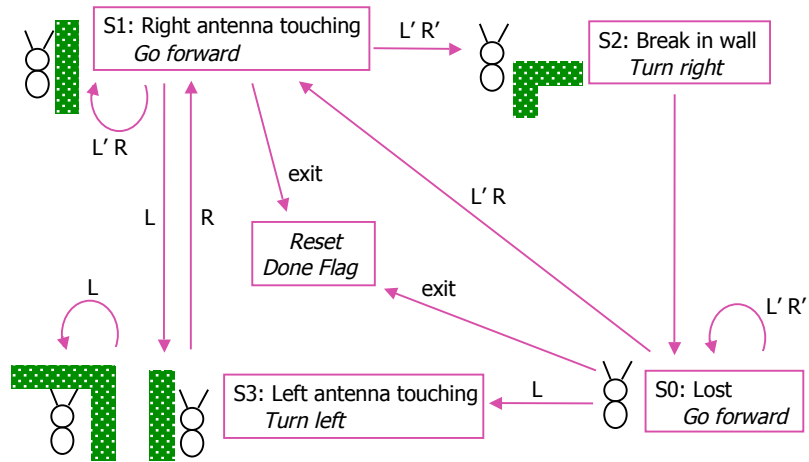
Partition the design



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



CSE370, Lecture 22

17

Step 1b: State-transition table

Exit	State	L	R	Next State	Output
1	Reset				
0	S0	0	0	S0	F
		0	1	S1	F
		1	0	S3	F
		1	1	S3	F
0	S1	0	0	S2	F
		0	1	S1	F
		1	0	S3	F
		1	1	S3	F
0	S2	0	0	S0	TR
		0	1	S0	TR
		1	0	S0	TR
		1	1	S0	TR
0	S3	0	0	S1	TL
		0	1	S1	TL
		1	0	S3	TL
		1	1	S3	TL

CSE370, Lecture 22

18

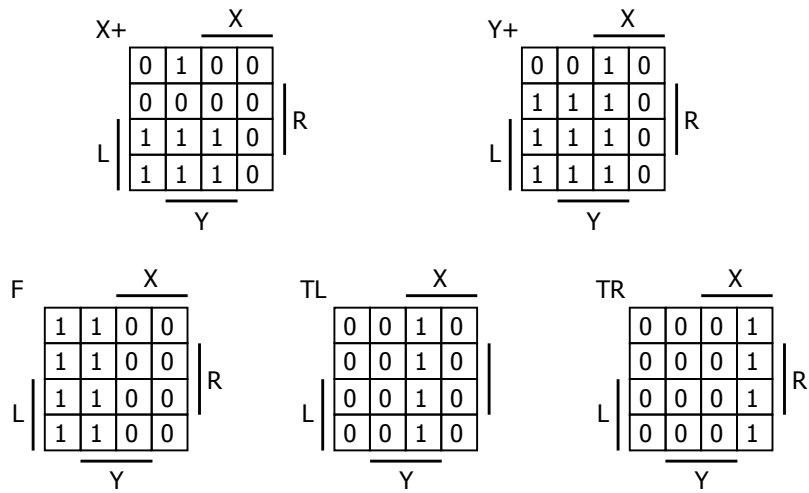
Step 2: State minimization

- ◆ 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

Exit	X Y	L R	X ⁺ Y ⁺	F TL TR	
1	Reset				
0	0 0	0 0	0 0	1 0 0	S0 → 00
	0 0	0 1	0 1	1 0 0	S1 → 01
	0 0	1 0	1 1	1 0 0	S2 → 10
	0 0	1 1	1 1	1 0 0	S3 → 11
0	0 1	0 0	1 0	1 0 0	
	0 1	0 1	0 1	1 0 0	
	0 1	1 0	1 1	1 0 0	
	0 1	1 1	1 1	1 0 0	
0	1 0	0 0	0 0	0 0 1	
	1 0	0 1	0 0	0 0 1	
	1 0	1 0	0 0	0 0 1	
	1 0	1 1	0 0	0 0 1	
0	1 1	0 0	0 1	0 1 0	
	1 1	0 1	0 1	0 1 0	
	1 1	1 0	1 1	0 1 0	
	1 1	1 1	1 1	0 1 0	

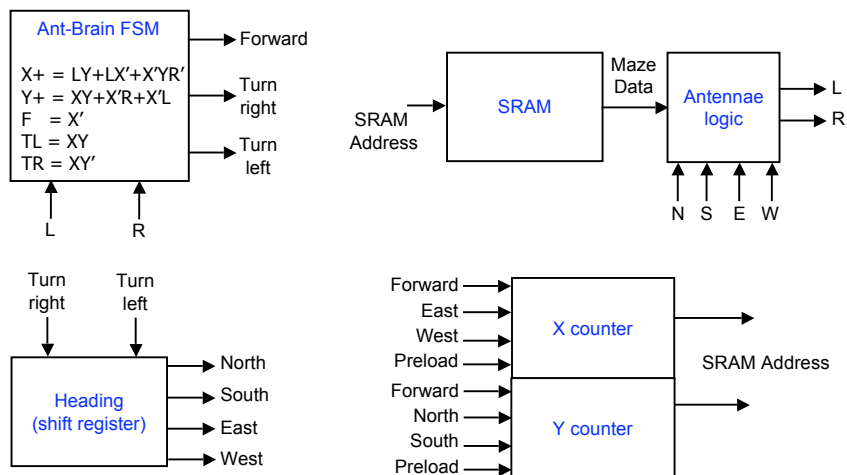
Step 4: Minimize the logic



CSE370, Lecture 22

21

Step 5: Implement the design



CSE370, Lecture 22

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

Gate count:
4 2-input ORs
8 2-input ANDs
2 4-input ORs

Logic for right antennae

$$R = NW(N + W) + WW(W + S) + SW(S + E) + EW(E + N)$$

Logic for left antennae

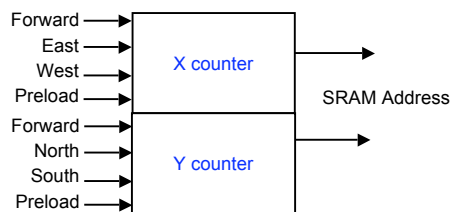
$$L = NW(N + E) + WW(W + N) + SW(S + W) + EW(E + S)$$

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:



- ◆ Due last day in class, Friday, March 9; printouts only
- ◆ Value: up to 1 homework
- ◆ Graded on clarity and completeness of explanation
- ◆ No questions will be answered