Lecture 22

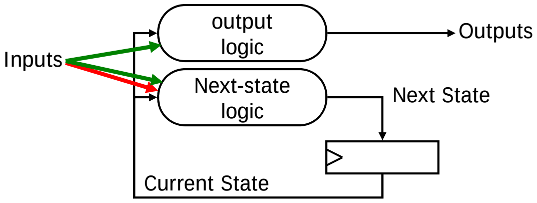
- Logistics
  - HW7 is due on Friday
  - Lab 8 this week
- Last lecture
  - FSMs
  - Intro to Moore and Mealy machines
- Today
  - More Moore and Mealy machines

The "WHY" slide

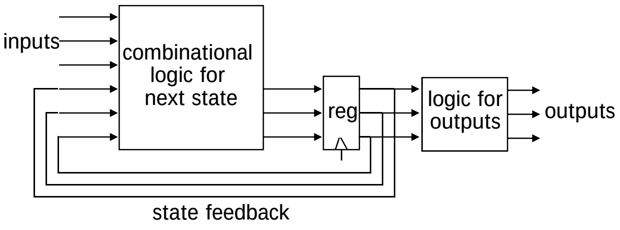
- Moore/Mealy machines
  - There are two different ways to express the FSMs with respect to the output. Both have different advantages so it is good to know them.

Generalized FSM model: Moore and Mealy

- Combinational logic computes next state and outputs
  - Next state is a function of current state and inputs
  - Outputs are functions of
    - Current state (**Moore** machine)
    - Current state and inputs (**Mealy** machine)

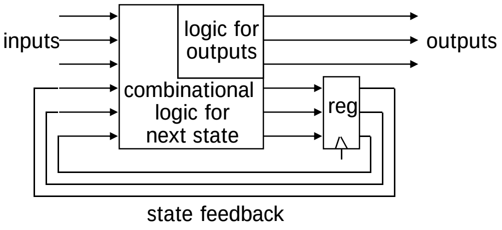


Moore versus Mealy machines



Moore machine
Outputs are a function of current state

Outputs change synchronously with state changes



Mealy machine
Outputs depend on state and on inputs

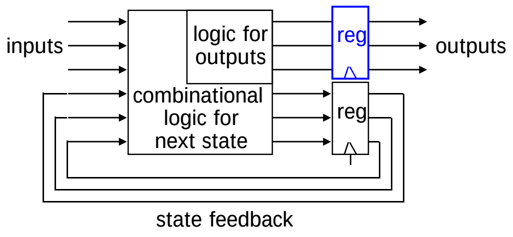
Input changes can cause immediate output changes (**asynchronous**)

Comparing Moore and Mealy machines

- Moore machines
  - + Safer to use because outputs change at clock edge
  - – May take additional logic to decode state into outputs
- Mealy machines
  - + Typically have fewer states
  - + React faster to inputs — don't wait for clock
  - – Asynchronous outputs can be dangerous
- We often design synchronous Mealy machines
  - Design a Mealy machine
  - Then register the outputs

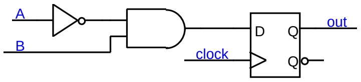
Synchronous (registered) Mealy machine

- Registered state and registered outputs
  - No glitches on outputs

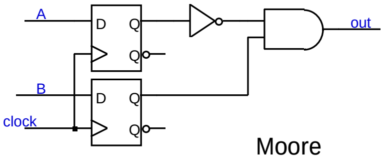


Example 0 -> 1: Moore or Mealy?

- Recognize A,B = 0,1
  - Mealy or Moore?



Registered Mealy
(actually Moore)



Moore

FSM design procedure reminder

- Counter-design procedure
  1. State diagram
  2. State-transition table
  3. Next-state logic minimization
  4. Implement the design

- FSM-design procedure
  1. State diagram
  2. state-transition table
  3. State minimization
  4. State encoding
  5. Next-state logic minimization
  6. Implement the design

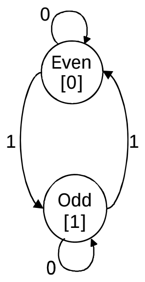
Example: A parity checker

- Serial input string
  - OUT=1 if odd # of 1s in input
  - OUT=0 if even # of 1s in input
- Let's do this for Moore and Mealy

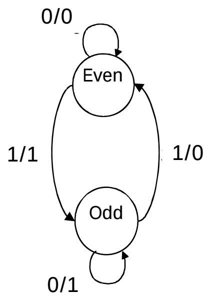
Example: A parity checker

1. State diagram

Moore



Mealy



Example: A parity checker

1. State-transition table

Moore

Present State	Input	Next State	Present Output
Even	0	Even	0
Even	1	Odd	0
Odd	0	Odd	1
Odd	1	Even	1

Mealy

Present State	Input	Next State	Present Output
Even	0	Even	0
Even	1	Odd	1
Odd	0	Odd	1
Odd	1	Even	0

Example: A parity checker

3. State minimization: Already minimized

- Need both states (even and odd)
- Use one flip-flop

Example: A parity checker

4. State encoding

Assignment
Even 0
Odd 1

Moore

Present State	Input	Next State	Present Output
0	0	0	0
0	1	1	0
1	0	1	1
1	1	0	1

Mealy

Present State	Input	Next State	Present Output
0	0	0	0
0	1	1	1
1	0	1	1
1	1	0	0

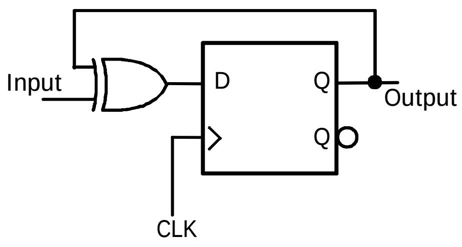
Example: A parity checker

5. Next-state logic minimization

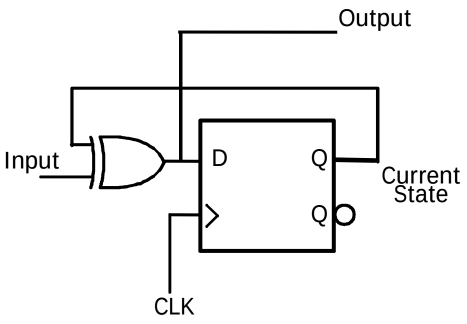
- Assume D flip-flops
- Next state = (present state) XOR (present input)

6. Implement the design

Moore

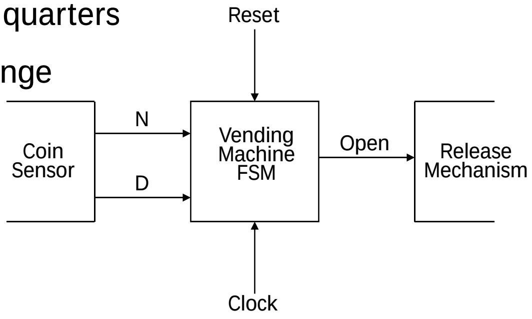


Mealy



Example: A vending machine

- 15 cents for a cup of coffee
- Doesn’t take pennies or quarters
- Doesn’t provide any change

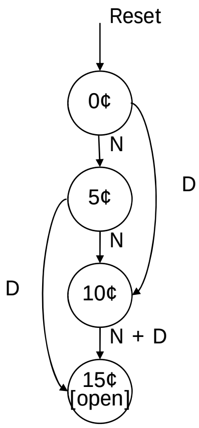


Last lecture

We had mix of

Moore and Mealy

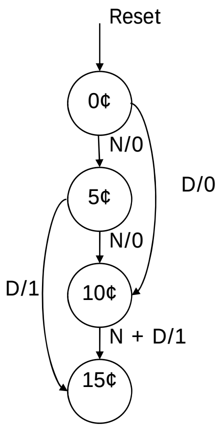
A vending machine: Moore machine



present state	inputs D	N	next state	present output
0¢	0	0	0¢	0
	0	1	5¢	0
	1	0	10¢	0
	1	1	–	–
5¢	0	0	5¢	0
	0	1	10¢	0
	1	0	15¢	0
	1	1	–	–
10¢	0	0	10¢	0
	0	1	15¢	0
	1	0	15¢	0
	1	1	–	–
15¢	–	–	15¢	1

symbolic state table

A vending machine: Mealy machine



present state	inputs D	inputs N	next state	present output
0¢	0	0	0¢	0
	0	1	5¢	0
	1	0	10¢	0
	1	1	–	–
5¢	0	0	5¢	0
	0	1	10¢	0
	1	0	15¢	1
	1	1	–	–
10¢	0	0	10¢	0
	0	1	15¢	1
	1	0	15¢	1
	1	1	–	–
15¢	–	–	15¢	1

symbolic state table

A vending machine: State encoding

Moore

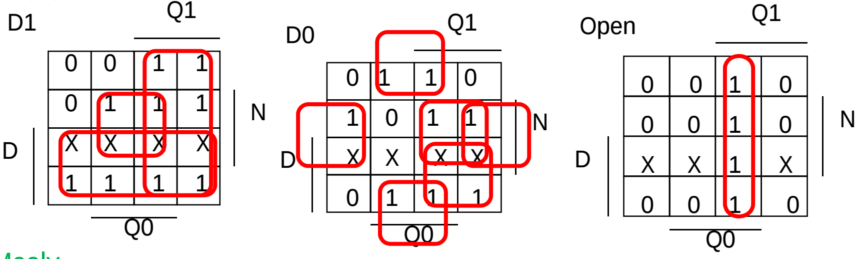
present state Q1	present state Q0	inputs D	inputs N	next state D1	next state D0	present output
0	0	0	0	0	0	0
		0	1	0	1	0
		1	0	1	0	0
		1	1	–	–	–
0	1	0	0	0	1	0
		0	1	1	0	0
		1	0	1	1	0
		1	1	–	–	–
1	0	0	0	1	0	0
		0	1	1	1	0
		1	0	1	1	0
		1	1	–	–	–
1	1	–	–	1	1	1

Mealy

present state Q1	present state Q0	inputs D	inputs N	next state D1	next state D0	present output
0	0	0	0	0	0	0
		0	1	0	1	0
		1	0	1	0	0
		1	1	–	–	–
0	1	0	0	0	1	0
		0	1	1	0	0
		1	0	1	1	1
		1	1	–	–	–
1	0	0	0	1	0	0
		0	1	1	1	1
		1	0	1	1	1
		1	1	–	–	–
1	1	–	–	1	1	1

A vending machine: Logic minimization

Moore



D1

0	0	1	1
0	1	1	1
X	X	X	X
1	1	1	1

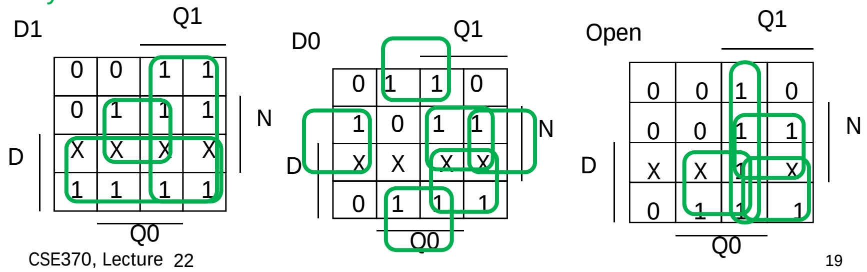
D0

0	1	1	0
1	0	1	1
X	X	X	X
0	1	1	1

Open

0	0	1	0
0	0	1	0
X	X	1	X
0	0	1	0

Mealy



D1

0	0	1	1
0	1	1	1
X	X	X	X
1	1	1	1

D0

0	1	1	0
1	0	1	1
X	X	X	X
0	1	1	1

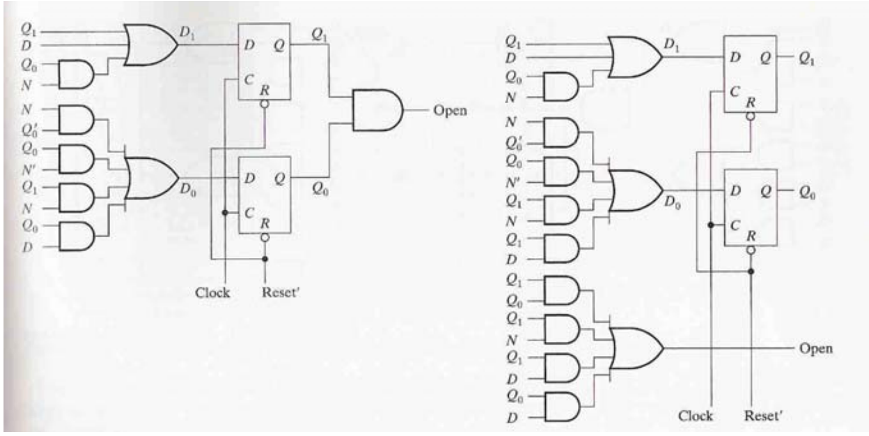
Open

0	0	1	0
0	0	1	1
X	X	1	X
0	1	1	1

A vending machine: Implementation

Moore

Mealy


Q1
D
Q0
N
N
Q0'
Q0
N'
Q1
N
Q0
D
D1
D0
D Q
C R
Q1
Q0
Open
Clock
Reset'
Q1
D
Q0
N
N
Q0'
Q0
N'
Q1
N
Q1
D
Q1
Q0
Q1
N
Q1
D
Q0
D
D1
D0
Open
Clock
Reset'