Lecture 19

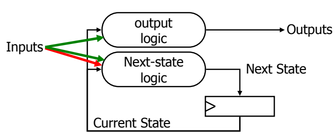
- Logistics
  - Lab 8 this week to be done in pairs
    - Find a partner before your lab period
    - Otherwise you will have to wait for a pairing which will slow you down
  - HW5 and HW6 solutions out today
  - HW7 out today due Wednesday March 4
  - Midterm 2 Wednesday
    - Covers material up to simple FSM
    - Review session tomorrow, 4:30 here, EEB 037
- Last lecture
  - Counter FSM design
  - General Finite State Machine Design
    - Vending machine example
- Today
  - Moore/Mealy machines
  - Midterm 2 topics and logistics

The "WHY" slide

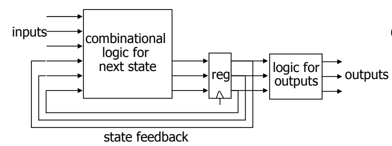
- Moore/Mealy machines
  - There are two different ways to express the FSMs with respect to the output. Both have different advantages so it is good to know them.

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)

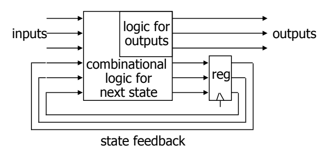


Moore versus Mealy machines



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**)

Impacts start of the FSM design procedure

- 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

State Diagrams

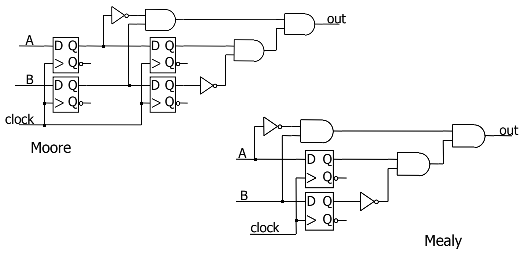
- Moore machine
  - Each state is labeled by a pair:

    state-name/output or state-name [output]
- Mealy machine
  - Each transition arc is labeled by a pair:

    input-condition/output

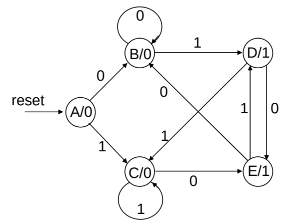
Example 10 → 01: Moore or Mealy?

- Circuits recognize AB=10 followed by AB=01
  - What kinds of machines are they?



Example "01 or 10" detector: a Moore machine

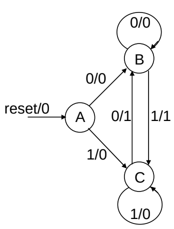
- Output is a function of state only
  - Specify output in the state bubble



reset	input	current state	next state	current output
1	–	–	A	0
0	0	A	B	0
0	1	A	C	0
0	0	B	B	0
0	1	B	D	0
0	0	C	E	0
0	1	C	C	0
0	0	D	E	1
0	1	D	C	1
0	0	E	B	1
0	1	E	D	1

Example "01 or 10" detector: a Mealy machine

- Output is a function of state and inputs
  - Specify outputs on transition arcs



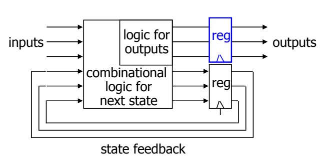
reset	input	current state	next state	current output
1	–	–	A	0
0	0	A	B	0
0	1	A	C	0
0	0	B	B	0
0	1	B	C	1
0	0	C	B	1
0	1	C	C	0

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

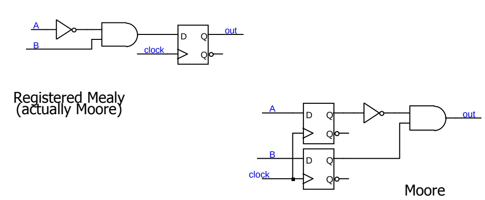
Synchronous (registered) Mealy machine

- Registered state and registered outputs
  - No glitches on outputs
  - No race conditions between communicating machines



Example "=01": Moore or Mealy?

- Recognize AB = 01
  - Mealy or Moore?



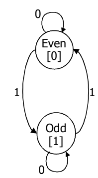
Example: A parity checker

- 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

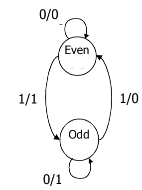
Example: A parity checker

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

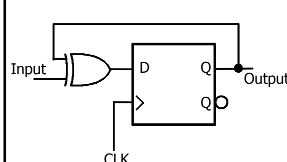
Mealy

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

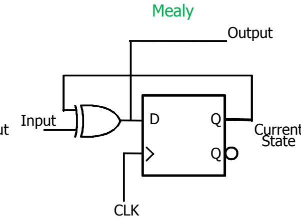
Example: A parity checker

5. Next-state logic minimization
   - Assume D flip-flops
   - Next state = (present state) XOR (present input)
6. Implement the design

Moore



Mealy



What was covered after midterm 1

- Combinational logic applications
  - PLAs/PALs
  - ROMs
  - Adders
  - Multi-level logic
  - Timing diagrams
  - Hazards

1010
+ 0110

????

What was covered after midterm 1

- Sequential logic building blocks
  - Latches (R-S and D)
  - Flip-flops (D and T)
  - Latch and flip-flop timing (setup/hold time, prop delay)
  - Timing diagrams
  - Asynchronous inputs and metastability
  - Registers

Remember that the last number was 1

What was covered after midterm 1

- Counters
  - Timing diagrams
  - Shift registers
  - Ring counters
  - State diagrams and state-transition tables
  - Counter design procedure
    1. Draw a state diagram
    2. Draw a state-transition table
    3. Encode the next-state functions
    4. Implement the design
  - Self-starting counters

1, 2, 3, 4, …

What was covered after midterm 1

- Finite state machines
  - 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
  - No Mealy machines

The last coin was 25cents and already had 50cents deposited so let's pop out a soda

Don' t expect to know a ton of FSM.

Just understand what was presented in the lectures.

Midterm 2 logistics

- 45 minutes long (starts 10:35)
- Materials covered from
  - Lectures 9 to 18 (but not Sequential Verilog or Moore/Mealy)
  - HW 4, 5, and 6
- Closed book/notes, no calculator
- Scratch papers provided
- Just have your pencil/pen and eraser
- Raise hand for questions (don't walk to get help)