Overview

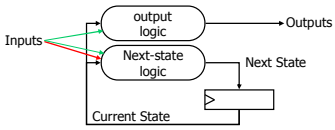
- Last lecture
  - Finished counter design
    - Design example
    - Self-starting counters
- Today
  - Introduction to finite-state machines
    - Moore versus Mealy machines
    - Synchronous Mealy machines
    - Example: A parity checker

Finite state machines

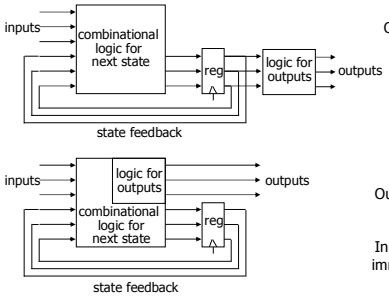
- FSM: A system that visits a finite number of logically distinct states
- Counters are simple FSMs
  - Outputs and states are identical
  - Visit states in a fixed sequence
- FSMs are more complex than counting
  - Outputs can depend on current state and on inputs
  - State sequencing depends on current state and on inputs

Generalized FSM model

- State variables (state vector) holds circuit state
  - Stored in registers
- 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

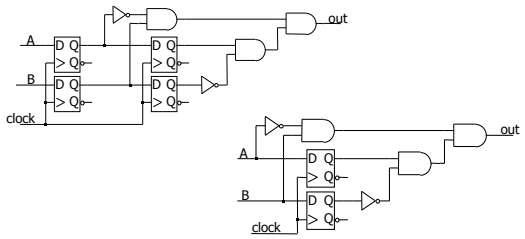
Outputs change synchronously with state changes

Mealy machine
Outputs depend on state and on inputs

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

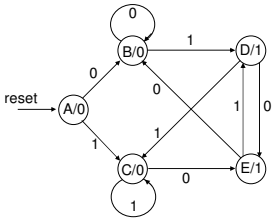
Example: Moore versus Mealy

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



Specifying outputs for a Moore machine

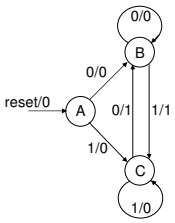
- Output is a function of state only
  - Specify output in the state bubble
  - Example: Detector for 01 or 10



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

Specifying outputs for a Mealy machine

- Output is a function of state and inputs
  - Specify outputs on transition arcs
  - Example: Detector for 01 or 10



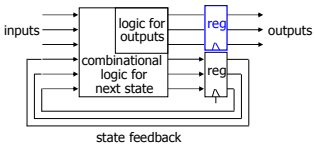
reset	input	current state	next state	current output
1	–	–	A	0
0	0	A	B	0
0	1	A	C	0
0	0	B	B	0
0	1	B	C	1
0	0	C	B	1
0	1	C	C	0

Comparing Moore and Mealy machines

- Moore machines
  - \+ Safer do 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 will 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



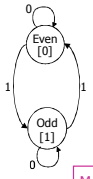
FSM design

- Generalized counter design
  - 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 and state-transition table
    2. State minimization
    3. State assignment (or state encoding)
    4. Next-state logic minimization
    5. 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

1. State diagram and state-transition table



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

Moore-machine state diagram

Parity checker (con't)

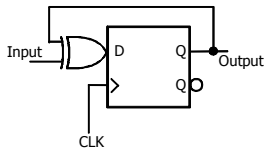
2. State minimization: Already minimized
   - Need both states (even and odd)
   - Use one flip-flop
3. State assignment (or state encoding)

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

Assignment
Even = 0
Odd = 1

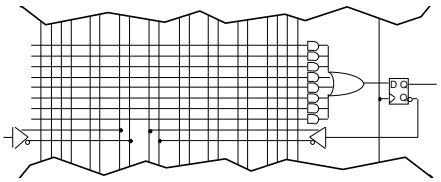
Parity checker (con't)

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



Implementation

- Can map FSMs to programmable logic devices
  - Macro-cell = DFF + two-level logic



- Other options: Gate arrays, semicustom ICs, etc.