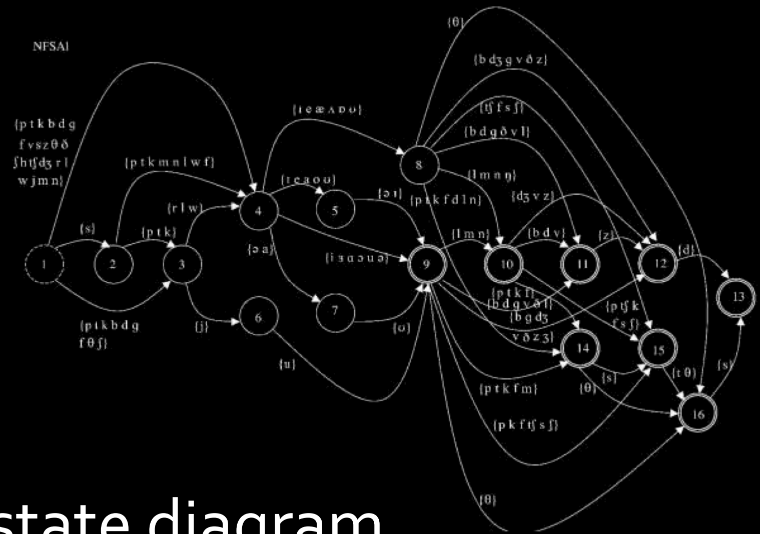
Lecture 5-2: Sequential Circuit Design continued

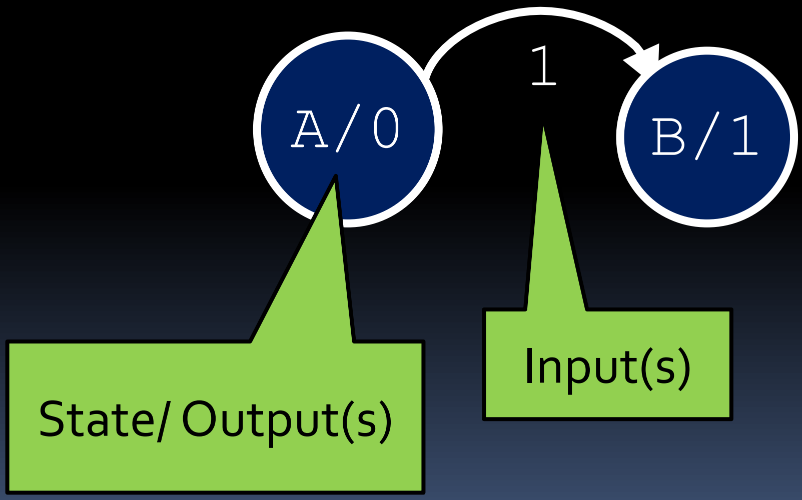
FSM design

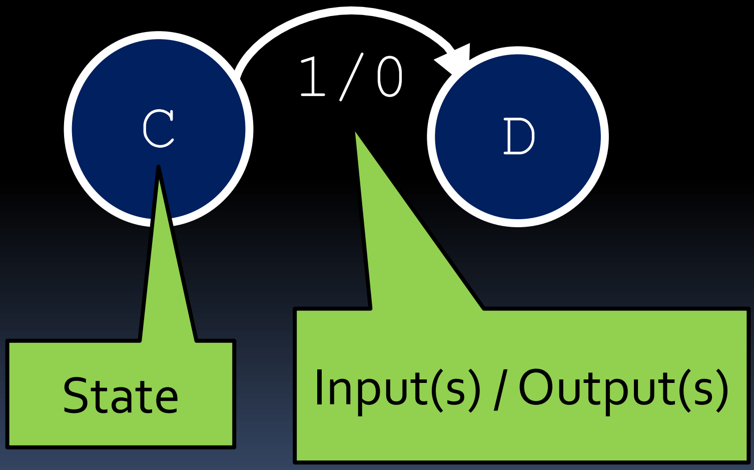


- Design steps for FSM:
  1. Draw state diagram
  2. Derive state table from state diagram
  3. Assign flip-flop configuration to each state
     - Number of flip-flops needed is: $\lceil \log_2(\# \text{ of states}) \rceil$
  4. Redraw state table with flip-flop values
  5. Derive combinational circuit for output and for each flip-flop input.

State diagrams with output

- Output values are incorporated into the state diagram, depending on the type of machine.
  - Moore Machine
  - Mealy Machine



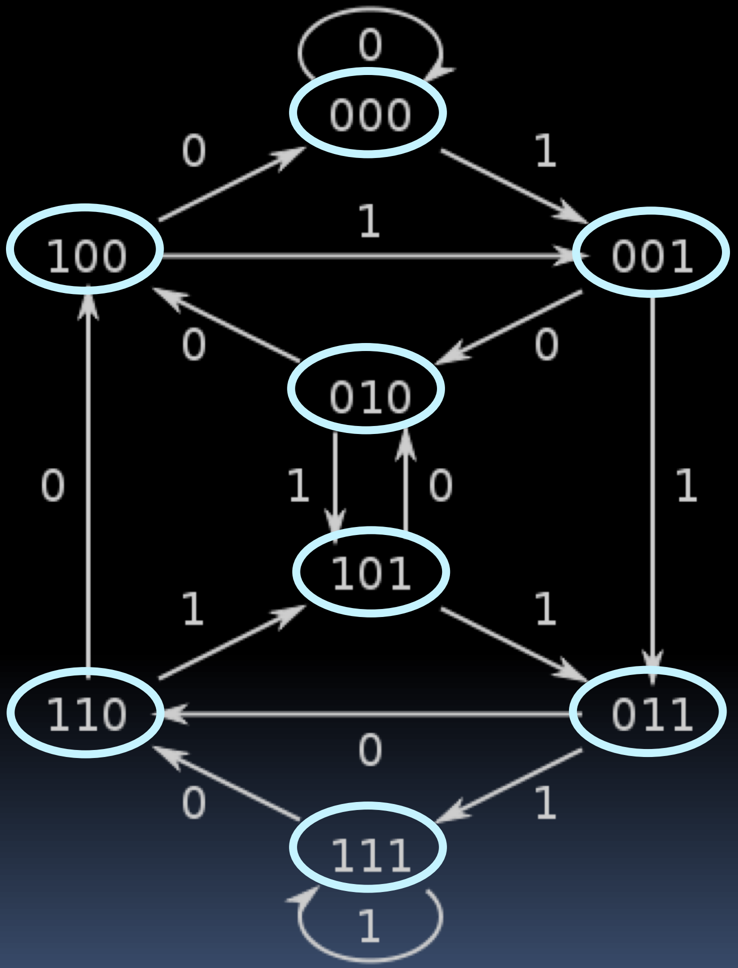


Example #4: Sequence Recognizer

- Recognize a sequence of input values, and raise a signal if that input has been seen.
- Example: Three high values in a row
  - Understood to mean that the input has been high for three rising clock edges.
  - Assumes a single input `IN` and a single output `Z`.

Step 1: State diagram

- In this case, the states are labeled with the input values that have been seen up to now.
- Transitions between states are indicated by the values on the transition arrows.



Step 2: State table

- Make sure that the state table lists all the states in the state diagram, and all the possible inputs that can occur at that state.

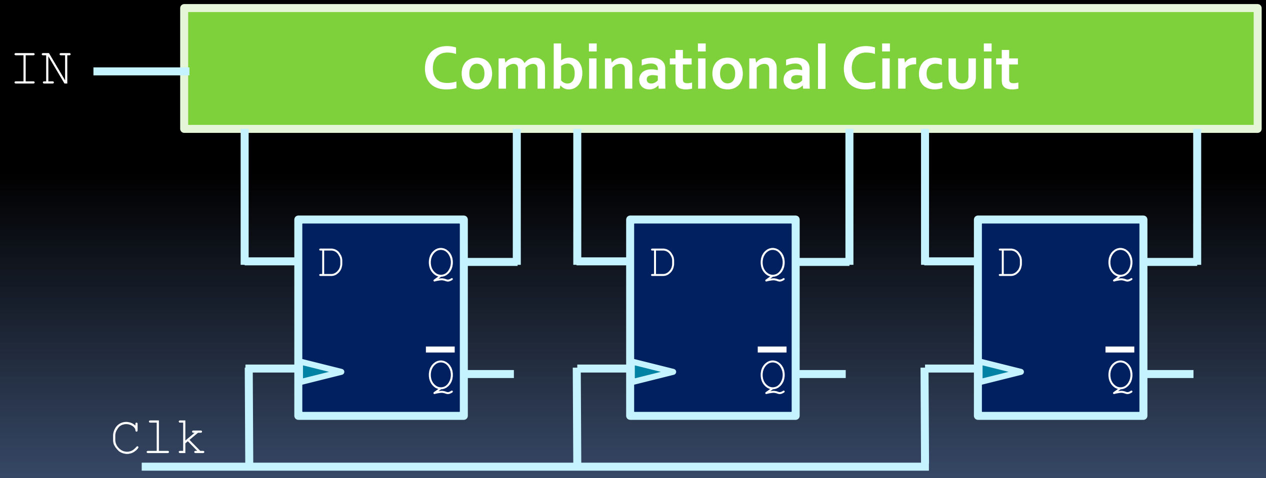
Previous State	Input	Next State
"000"	0	"000"
"000"	1	"001"
"001"	0	"010"
"001"	1	"011"
"010"	0	"100"
"010"	1	"101"
"011"	0	"110"
"011"	1	"111"
"100"	0	"000"
"100"	1	"001"
"101"	0	"010"
"101"	1	"011"
"110"	0	"100"
"110"	1	"101"
"111"	0	"110"
"111"	1	"111"

Step 3: Assign flip-flops

- The flip-flops are the circuit units that are responsible for actually storing states.
- When deciding how many states are needed, remember that a single flip-flop can store two values (`0` and `1`), and thus two states.
- How many states can be stored with each additional flip-flop?
  - One flip-flop → 2 states
  - Two flip-flops → 4 states
  - Three flip-flops → 8 states
  - …
  - Eight flip-flops? → $2^8 = 256$ states

Step 3: Assign flip-flops

- In this case, we need to store 8 states.
  - 8 states = 3 flip-flops ($3 = \log_2 8$)
- For now, assign a flip-flop to each digit of the state names in the FSM & state table.



Step 4: State table

- Usually, the states have names that don't map over to flip-flops so easily.
- It may be an easy mapping, but is it a good one?
  - Not really, but we'll get to why later.

Prev. State	IN	Next State
"000"	0	"000"
"000"	1	"001"
"001"	0	"010"
"001"	1	"011"
"010"	0	"100"
"010"	1	"101"
"011"	0	"110"
"011"	1	"111"
"100"	0	"000"
"100"	1	"001"
"101"	0	"010"
"101"	1	"011"
"110"	0	"100"
"110"	1	"101"
"111"	0	"110"
"111"	1	"111"

Step 5: Circuit design

- Karnaugh map for $F_{2(t+1)}$:

	$\overline{F_0} \cdot \overline{IN}$	$\overline{F_0} \cdot IN$	$F_0 \cdot IN$	$F_0 \cdot \overline{IN}$
$\overline{F_2} \cdot \overline{F_1}$	0	0	0	0
$\overline{F_2} \cdot F_1$	1	1	1	1
$F_2 \cdot F_1$	1	1	1	1
$F_2 \cdot \overline{F_1}$	0	0	0	0

$$F_{2(t+1)} = F_{1(t)}$$

Step 5: Circuit design

- Karnaugh map for $F_{1(t+1)}$:

	$\overline{F_0} \cdot \overline{IN}$	$\overline{F_0} \cdot IN$	$F_0 \cdot IN$	$F_0 \cdot \overline{IN}$
$\overline{F_2} \cdot \overline{F_1}$	0	0	1	1
$\overline{F_2} \cdot F_1$	0	0	1	1
$F_2 \cdot F_1$	0	0	1	1
$F_2 \cdot \overline{F_1}$	0	0	1	1

$$F_{1(t+1)} = F_{0(t)}$$

Step 5: Circuit design

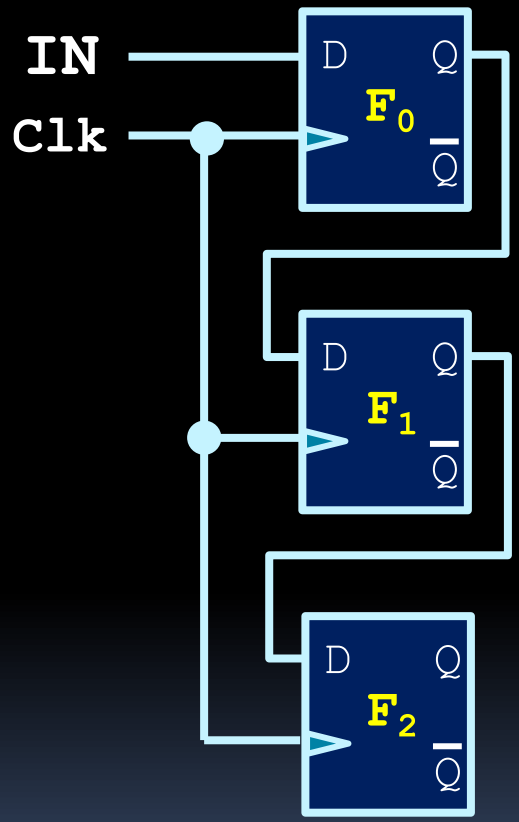
- Karnaugh map for $F_{0(t+1)}$:

	$\overline{F_0}\cdot\overline{IN}$	$\overline{F_0}\cdot IN$	$F_0\cdot IN$	$F_0\cdot\overline{IN}$
$\overline{F_2}\cdot\overline{F_1}$	0	1	1	0
$\overline{F_2}\cdot F_1$	0	1	1	0
$F_2\cdot F_1$	0	1	1	0
$F_2\cdot\overline{F_1}$	0	1	1	0

$$F_{0(t+1)} = IN_{(t)}$$

Step 5: Circuit design

- Resulting circuit looks like the diagram on the right.
- This will record the states and make the state transitions happen based on the input, but what about the output value Z?
  - Z should go high when EN has been high for three clock cycles in a row.



Step 5: Circuit design

- Boolean equation for Z:

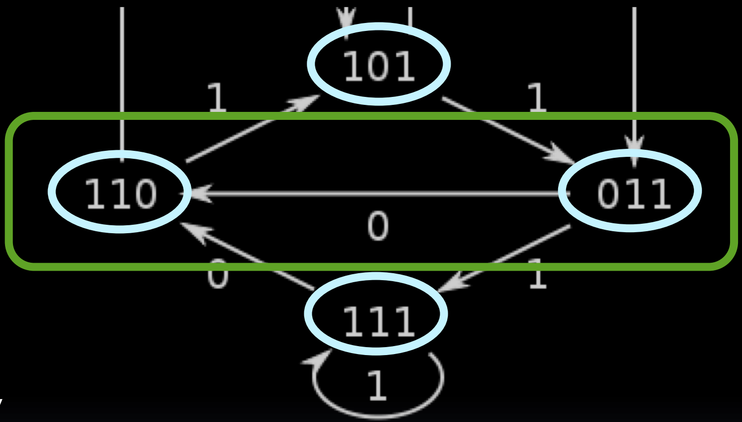
$$Z = F_0 \cdot F_1 \cdot F_2$$

FSM design

- Design steps for FSM:
  1. Draw state diagram
  2. Derive state table from state diagram
  3. Assign flip-flop configuration to each state
     - Number of flip-flops needed is: $\lceil \log_2(\# \text{ of states}) \rceil$
  4. Redraw state table with flip-flop values
  5. Derive combinational circuit for output and for each flip-flop input.

Timing and state assignments

- When assigning states, you need to consider the issue of timing with the states.
- <u>Example:</u> if recognizer circuit is in state 011 and gets a 0 as an input, it moves to state 110.
  - The first and last digits change "at the same time"
  - If the first flip-flop changes first, the output would go high for an instant (incorrectly!), which could cause unexpected behaviour.



Timing and state assignments

- So how do you solve this?
- Possible solutions:
  1. Whenever possible, make flip-flop assignments such that neighbouring states differ by at most one flip-flop value (state encoding differs by one bit).
  2. If "intermediate" state output is the same as starting or destination state → no problem
  3. Add intermediate transition states between start and end
     1. Use unused flip-flop states or may need to add more

Question #4

- How would we make the following Finite State Machine?

Example #5

- Exploding Pen
  - Starts disarmed
  - 3 clicks to arm
  - 3 clicks to disarm

- https://youtu.be/Vi4LmILZUog

  - Note: Please do not use the knowledge you've gained in this course to develop exploding pens.
  - Note 2: If you do, please don't use them for evil