Lecture 4

ECE2883 HP

T. Collins / K. Johnson

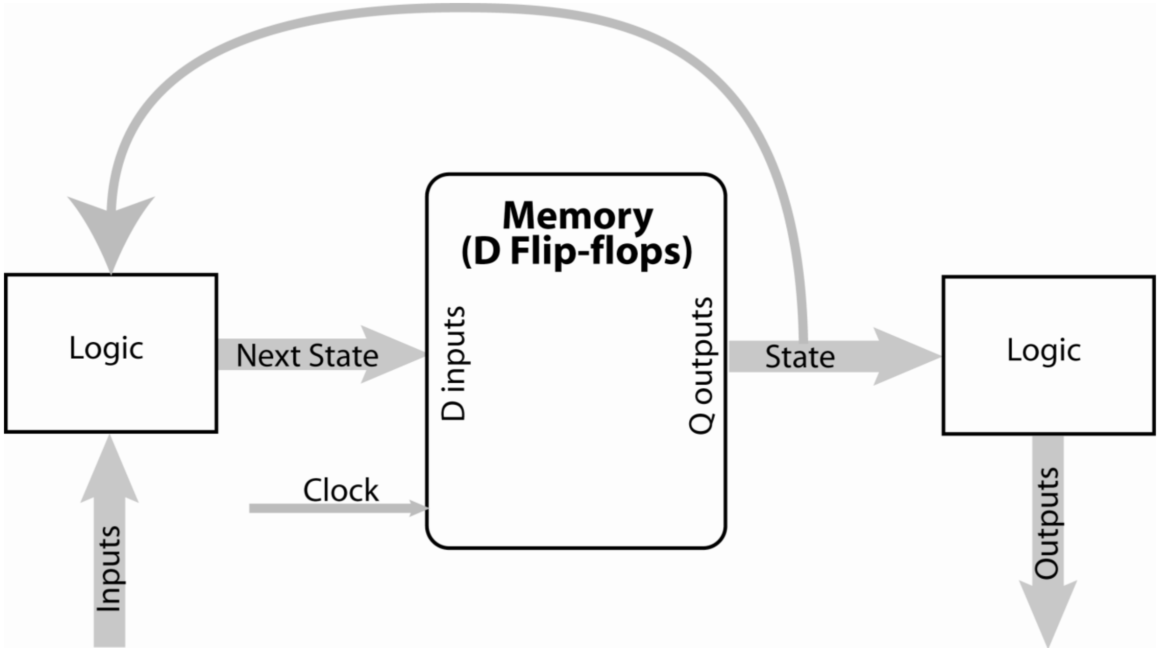
State machines

- State machines (specifically *finite state machines*) are *sequential logic*
- Like combinational logic, they have inputs and outputs
- Unlike combinational logic, they have multiple unique conditions (states) where the output may be different, even with the same input

Why consider state machines?

- Suppose you want an “intelligent” traffic light controller that responds to the presence of waiting cars in each direction
  - Inputs are two bits indicating the presence of cars
  - Outputs are bits to control each light color in each direction of traffic
- Would it be reasonable to build the controller with only combinational logic?
  - Do you instantly give a green light to traffic that shows up?
  - What do you do when traffic is waiting in both directions?
  - How to you present timed outputs like a yellow light?
- Such a controller needs to consider a HISTORY of inputs
  - Combinational circuits don’t “remember” anything

State machines: The big picture



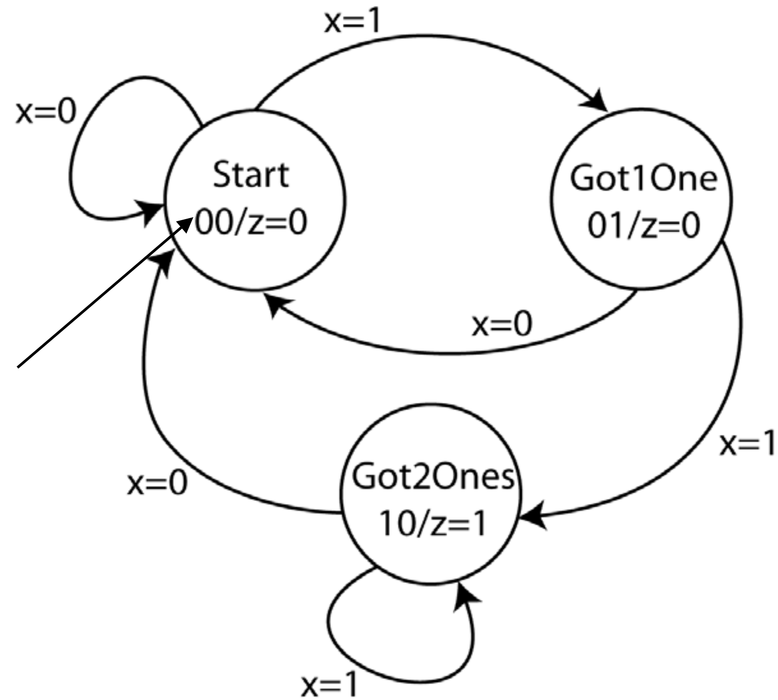
- Combinational logic without memory has no concept of *state*
- State machines combine memory elements with combinational logic
- The result is a device that reacts not only to the current input, but also to the history of how inputs have been applied
- Shown here is a general Moore state machine

Classic state diagram

- A *classic* diagram represents each state with a circle, and each state transition with an arc
- What does this state machine do?

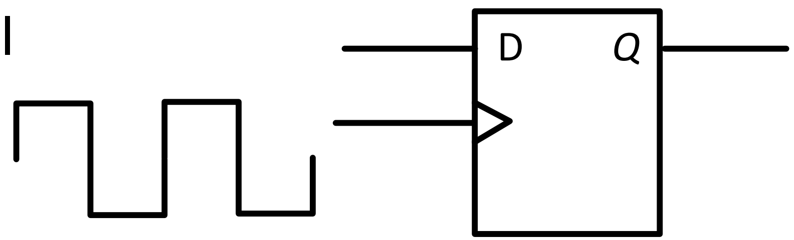
Binary input x,
Binary output z

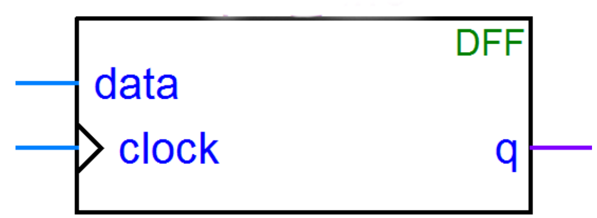
Arbitrary
State Number
(binary)



Memory and D Flip-flops

- Think of a bit of memory as being a place that “remembers” a 0 or 1
  - If you “set” it (write a “1” to it, it will stay that way
  - If you “reset” it (write a “0”, or “clear”) it, it will stay that way, too
- One common way to implement is a “D Flip-flop”
  - When a positive clock edge comes along, the value on the D input is remembered (and appears at Q output indefinitely)
  - After that, D can change

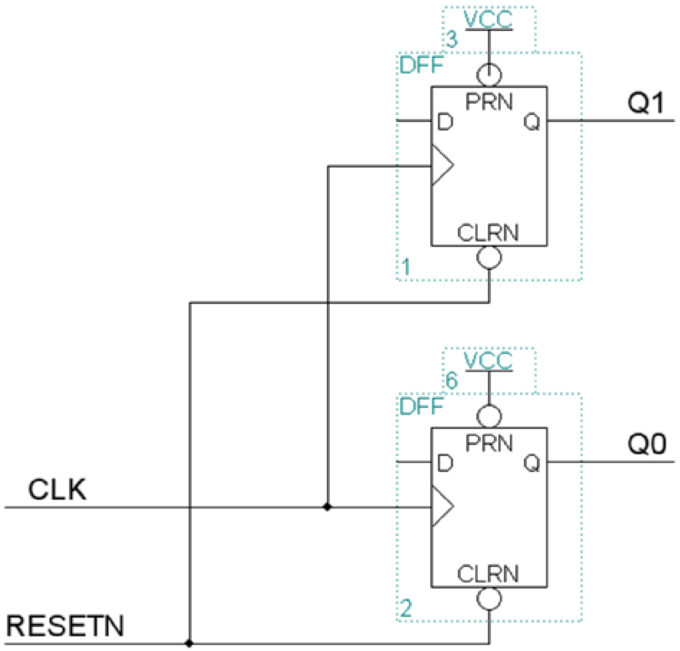




Symbol in our CAD

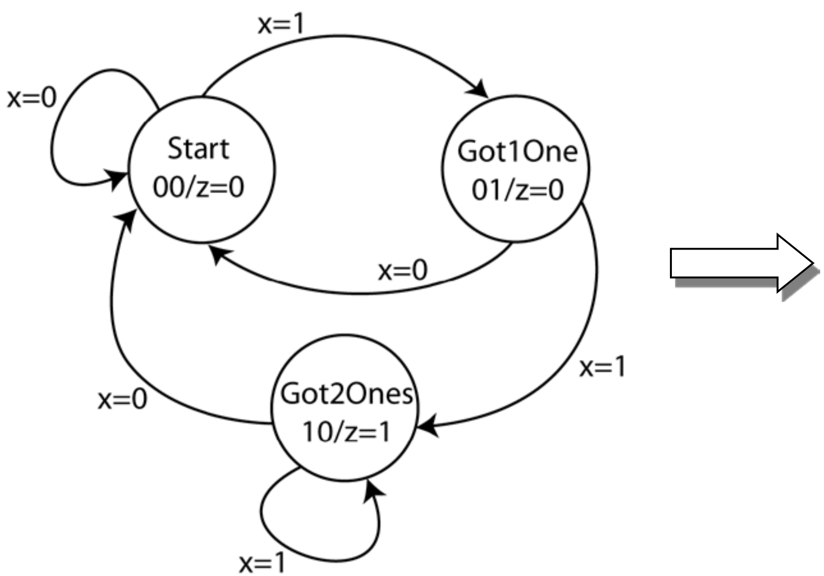
Implementing the example (LEs)

- Start by noting how many state variables (Qs) are needed
  - If you have *m* states, the number of bits you need will be the smallest integer that is greater than or equal to $\log_2(m)$
  - $m=3 \rightarrow 1<\log_2(3)<2$, so we know we need 2 Qs
- Include a flip-flop for each Q
- Connect a common clock
- Use asynchronous inputs to achieve the proper startup behavior
  - Our start state is 00, so an active-low reset signal works



Creating the *transition table*

- There are two places in the "Big Picture" where combinational logic is required
  - Generating "next states" (Q+ in this example)
  - Generating outputs (Z in this example)
- This step simply gathers the information needed to solve for that logic



Present State			Next State		
Q_1	Q_0	X	Q_1^+	Q_0^+	Z
0	**0**	**0**	**0**	**0**	**0**
0	**0**	**1**	**0**	**1**	**0**
0	**1**	**0**	**0**	**0**	**0**
0	**1**	**1**	**1**	**0**	**0**
1	**0**	**0**	**0**	**0**	**1**
1	**0**	**1**	**1**	**0**	**1**
1	**1**	**0**	**d**	**d**	**d**
1	**1**	**1**	**d**	**d**	**d**

Generating the equations

- LEs know this part. For information only
- Each next state or output column can be put on a K-map and solved
- Q_0^+ (Next Q_0) is shown here
- Solution of other two columns is similar

Present State			Next State		
Q_1	Q_0	X	Q_1^+	Q_0^+	Z
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	0
0	1	1	1	0	0
1	0	0	0	0	1
1	0	1	1	0	1
1	1	0	d	d	d
1	1	1	d	d	d

Q_1Q_0 \ X	0	1
00	**0**	**1**
01	**0**	**0**
11	**d**	**d**
10	**0**	**0**

Q_0^+

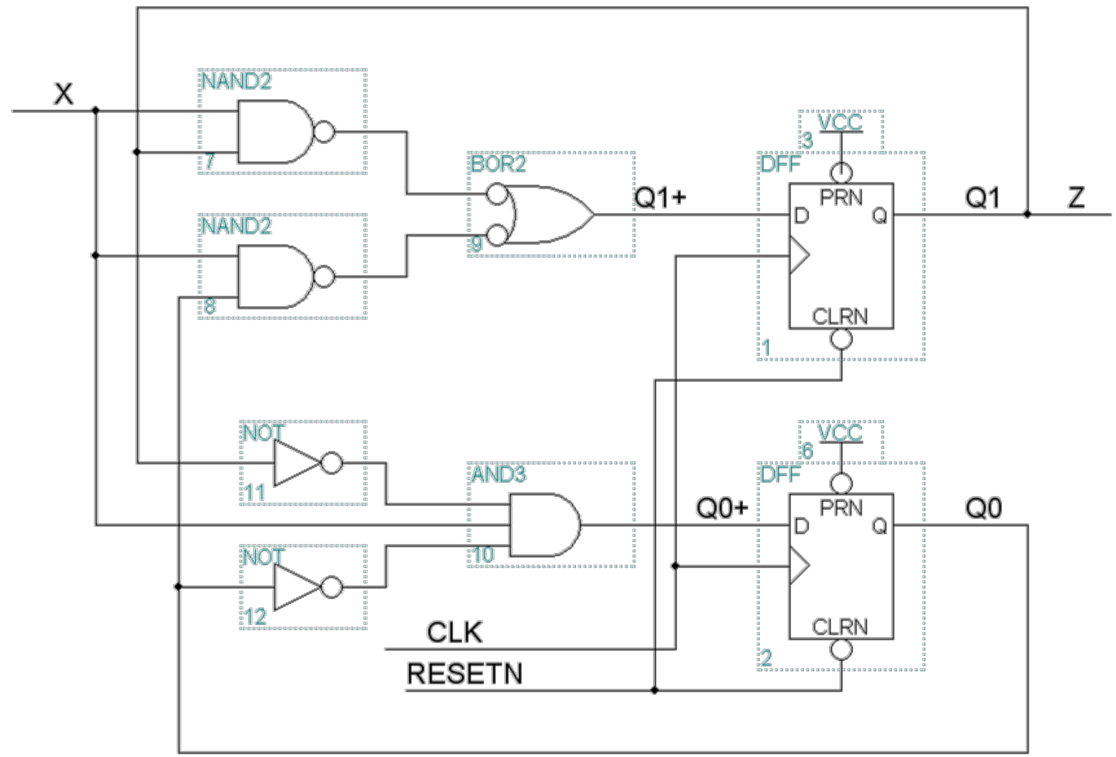
Designing the circuit

$$Q_1^{\,+} = XQ_0 + XQ_1$$

$$Q_0^{\,+} = X\,\overline{Q_1}\,\overline{Q_0}$$

$$Z = Q_1$$

- Start with the two flip-flops
- For state machines built with D flip-flops, D is Q+



State machines in ECE2883HP

- Will usually be embedded in other devices
- Everyone will be able to conceptually understand what they do
- LEs can do detailed design to make sure that they work as intended
- A simple example is foreshadowed in Lab 3 assignment
  - Related to Waterfall Swing

Project team formation