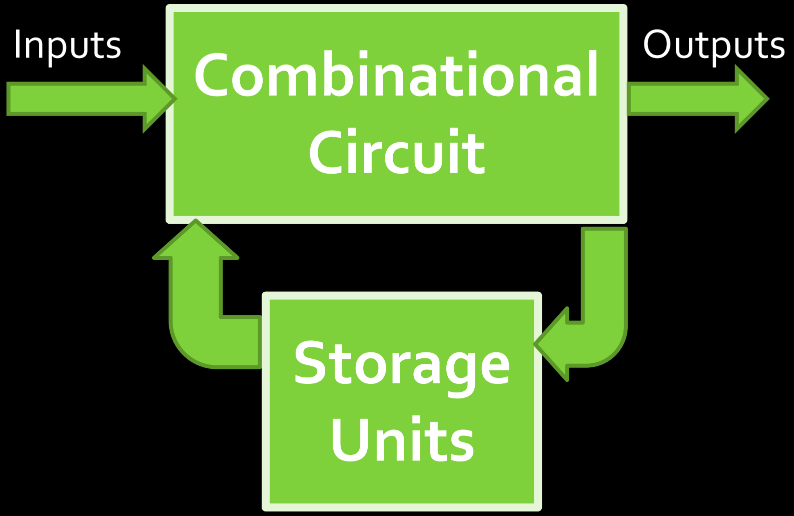
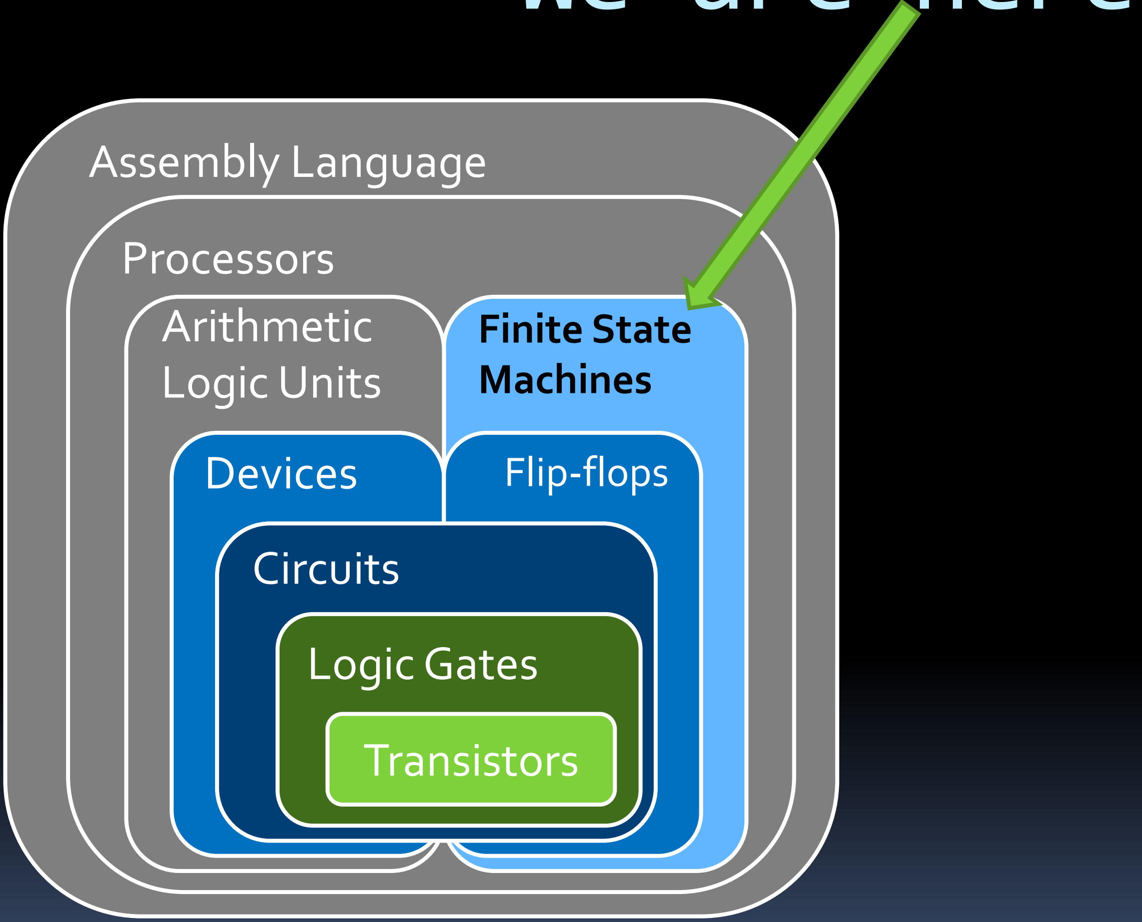
Lecture 5: Sequential Circuit Design

Circuits using flip-flops

- Now that we know about flip-flops and what they do, how do we use them in circuit design?
- What's the benefit in using flip-flops in a circuit at all?



We are here



Assembly Language

Processors

Arithmetic Logic Units

Finite State Machines

Devices

Flip-flops

Circuits

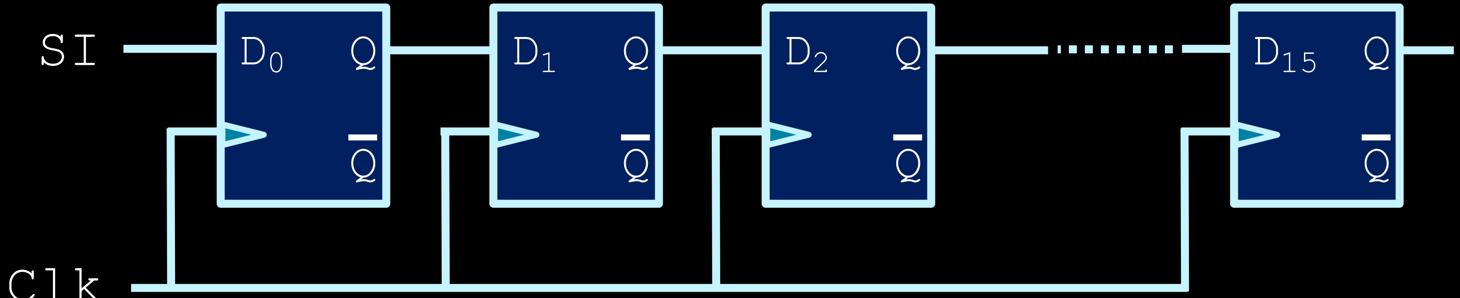
Logic Gates

Transistors

Example #1: Registers

Shift registers

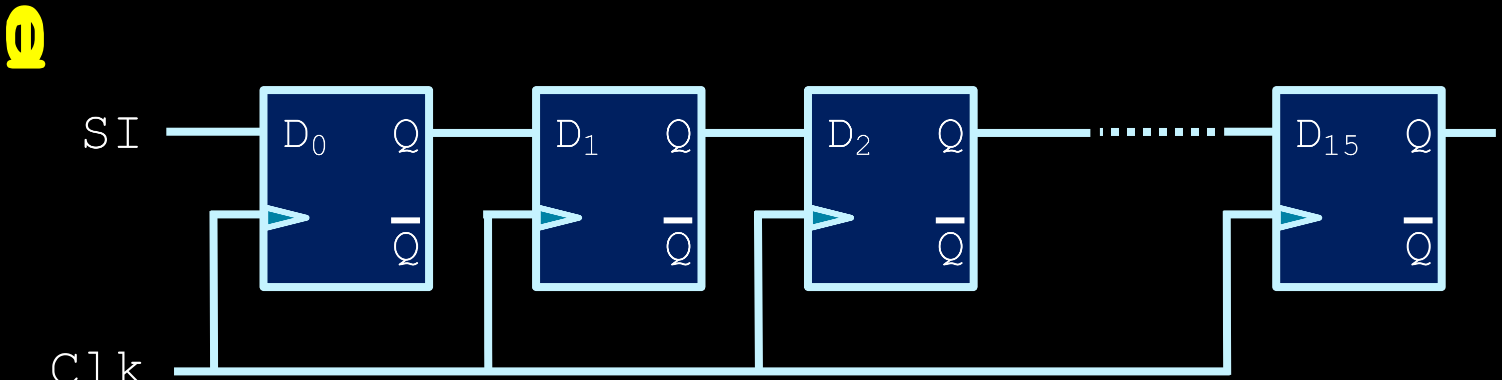
- A series of D flip-flops can store a multi-bit value (such as a 16-bit integer, for example).



- Data can be shifted into this register one bit at a time, over 16 clock cycles.
  - Known as a shift register.

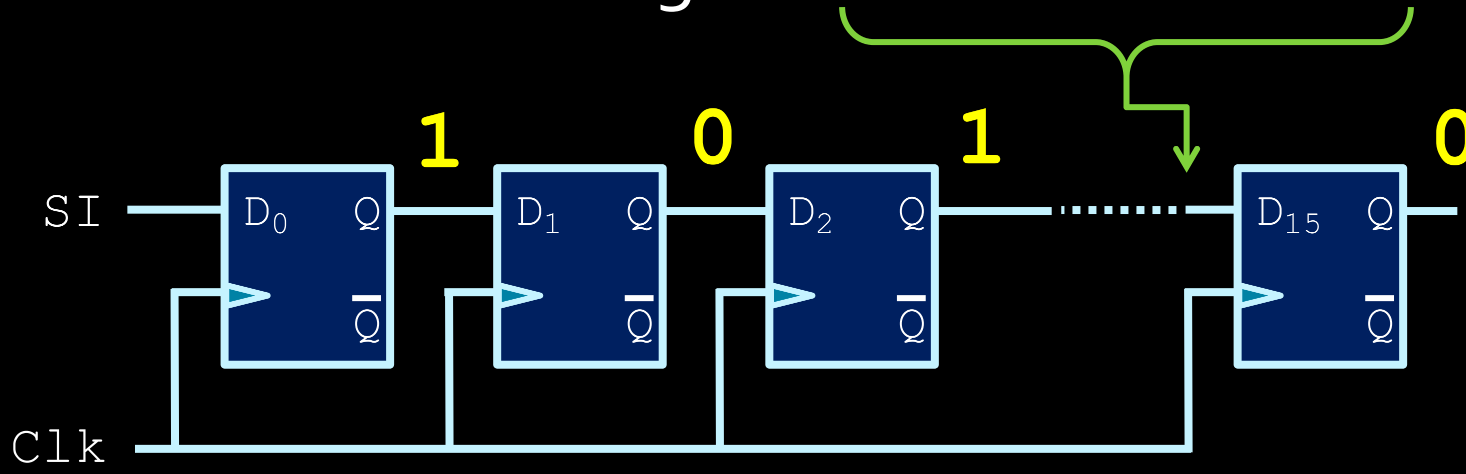
Shift registers

- Illustration: shifting in `0101010101010101`



Shift registers

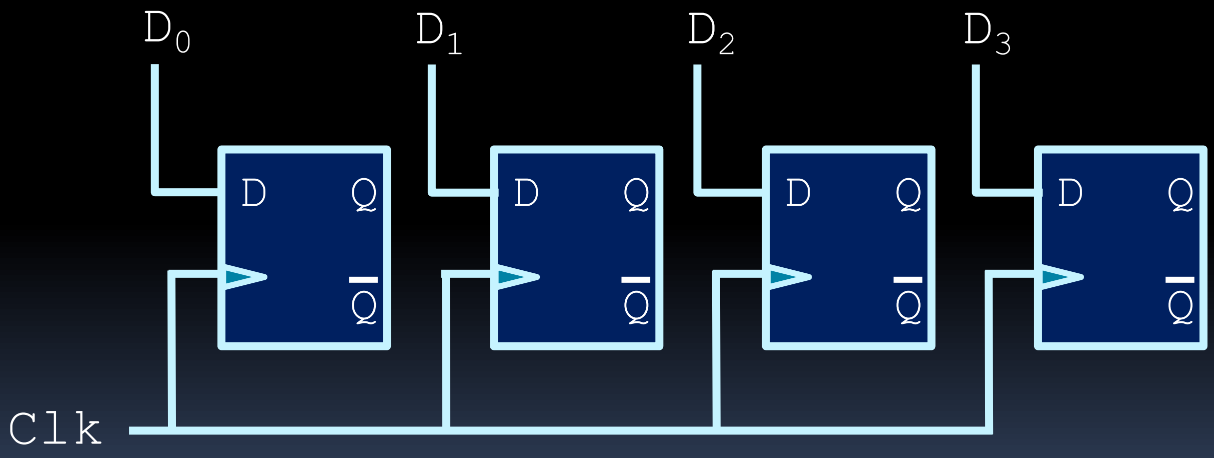
- Illustration: shifting in 0101010101010101



- After 16 clock cycles….

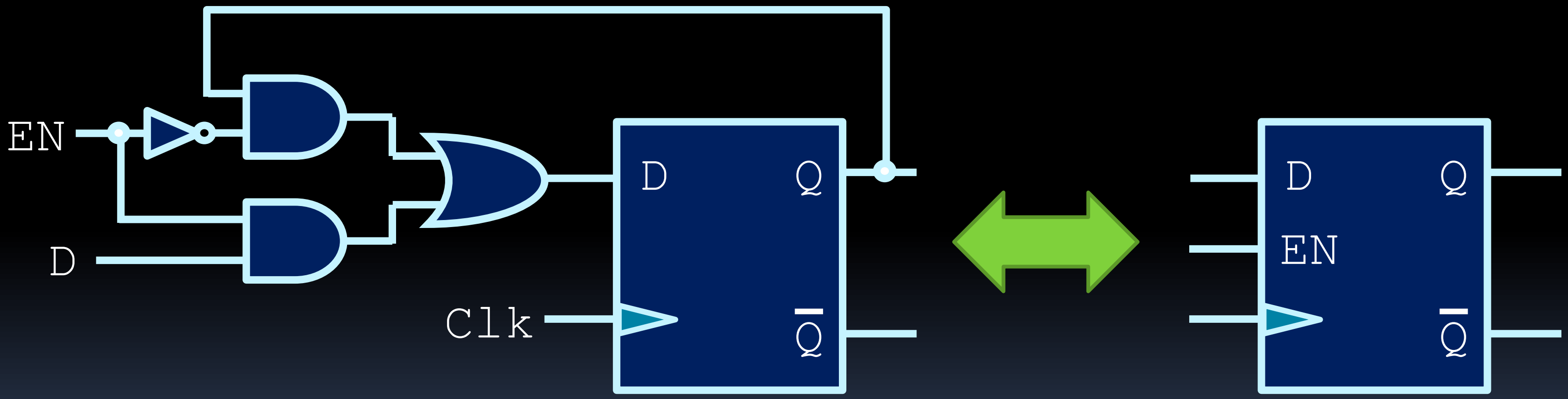
Load registers

- One can also load a register's values all at once, by feeding signals into each flip-flop:
  - In this example: a 4-bit load register.



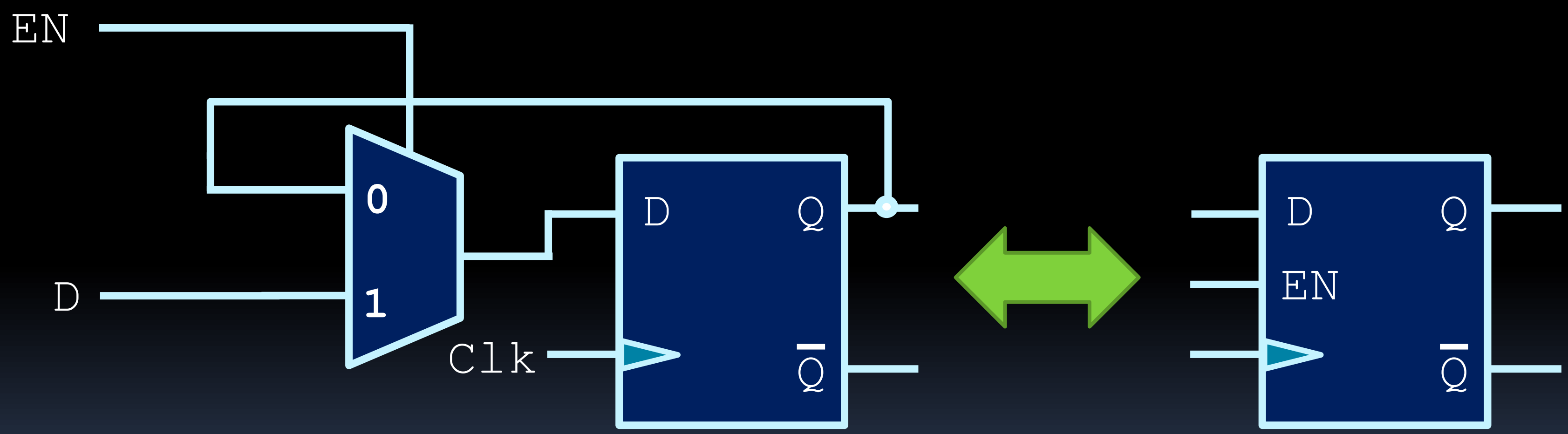
Load registers

- To control when this register is allowed to load its values, we introduce the D flip-flop with enable:



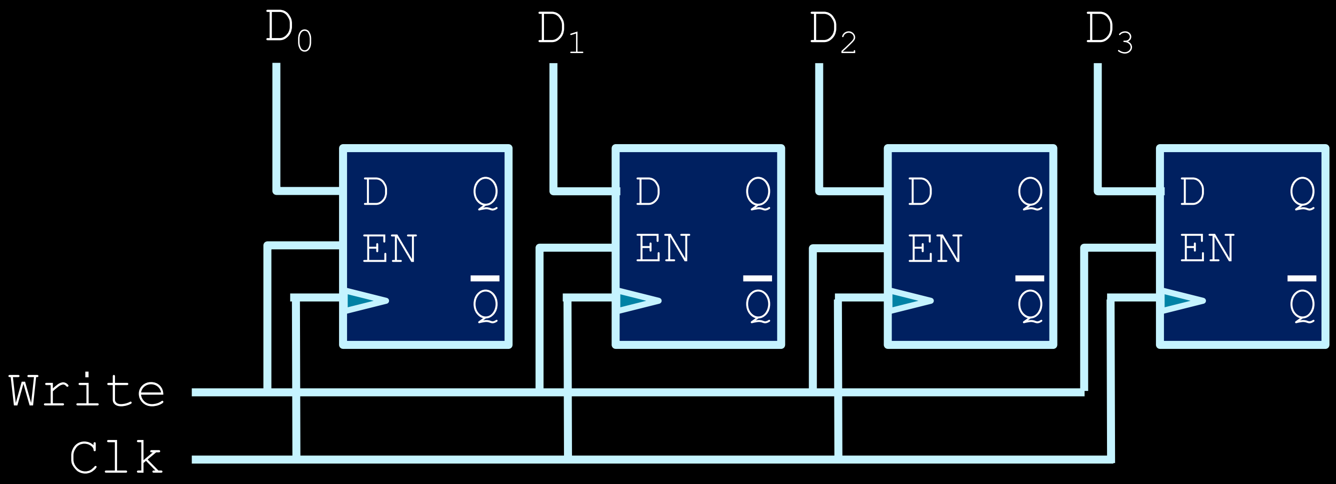
Load registers

- To control when this register is allowed to load its values, we introduce the D flip-flop with enable:



- It’s a MUX!

Load registers

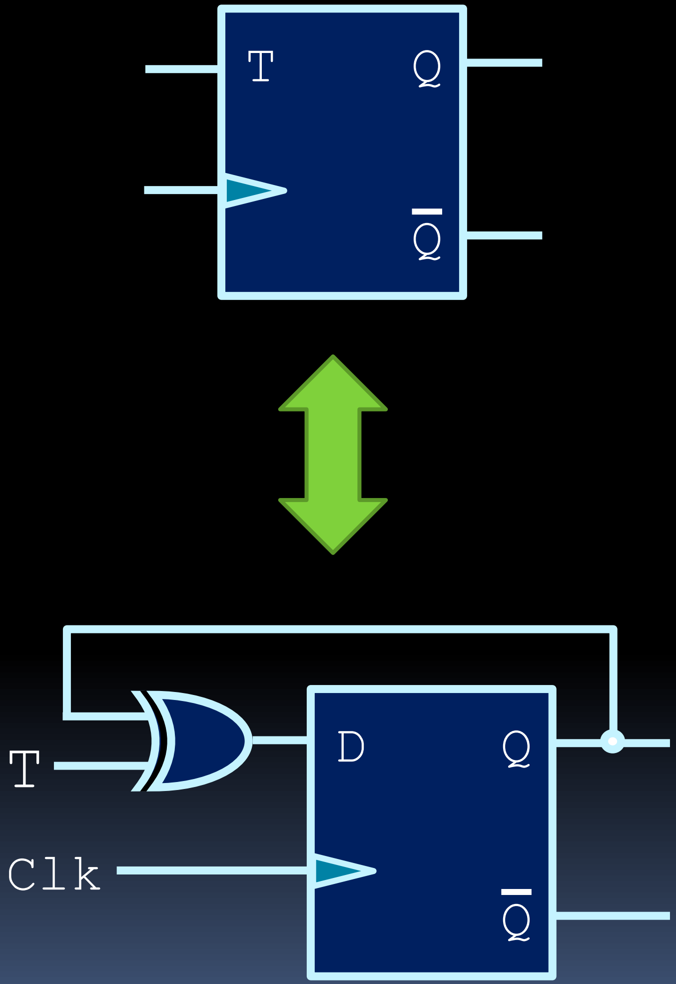


- Implementing the register with these special D flip-flops will now maintain values in the register until overwritten by setting EN high.

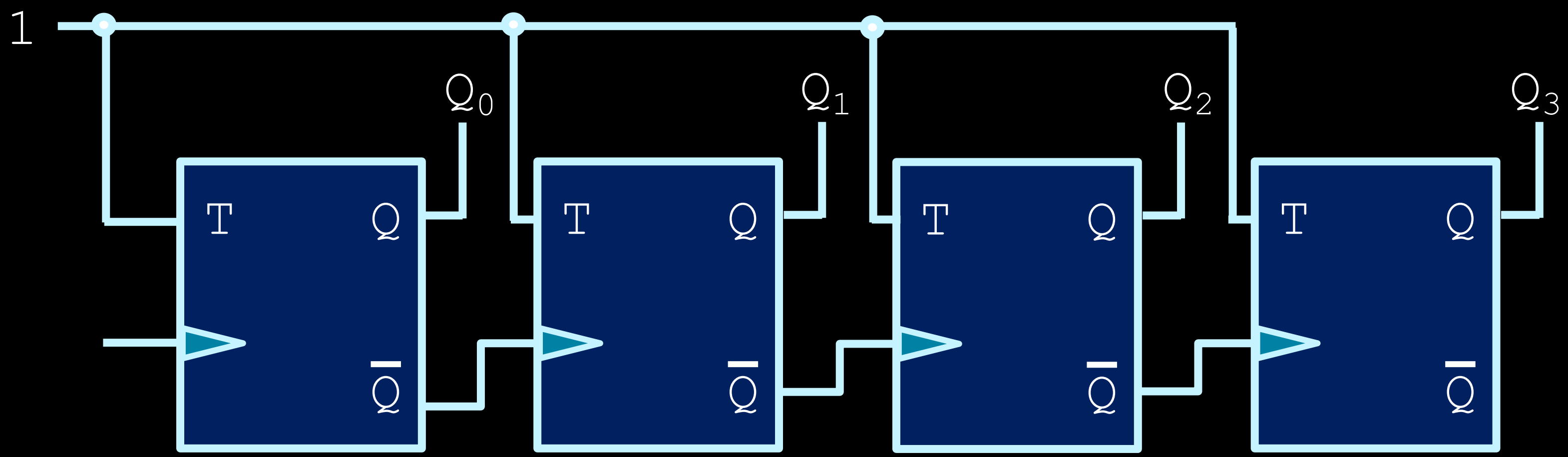
Example #2: Counters

Counters

- Consider the T flip-flop:
  - Output is inverted when input T is high.
- What happens when a series of T flip-flops are connected together in sequence?
- More interesting: Connect the *output* of one flip-flop to the clock input of the next!

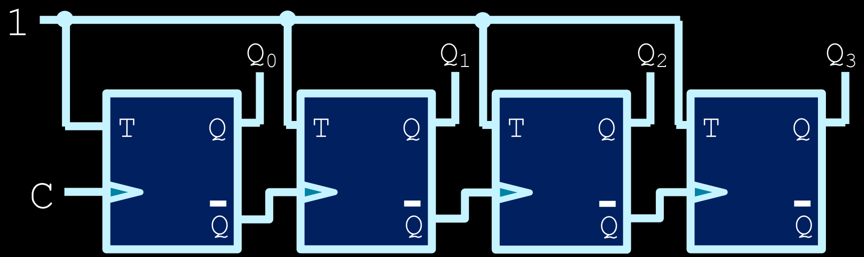


Counters

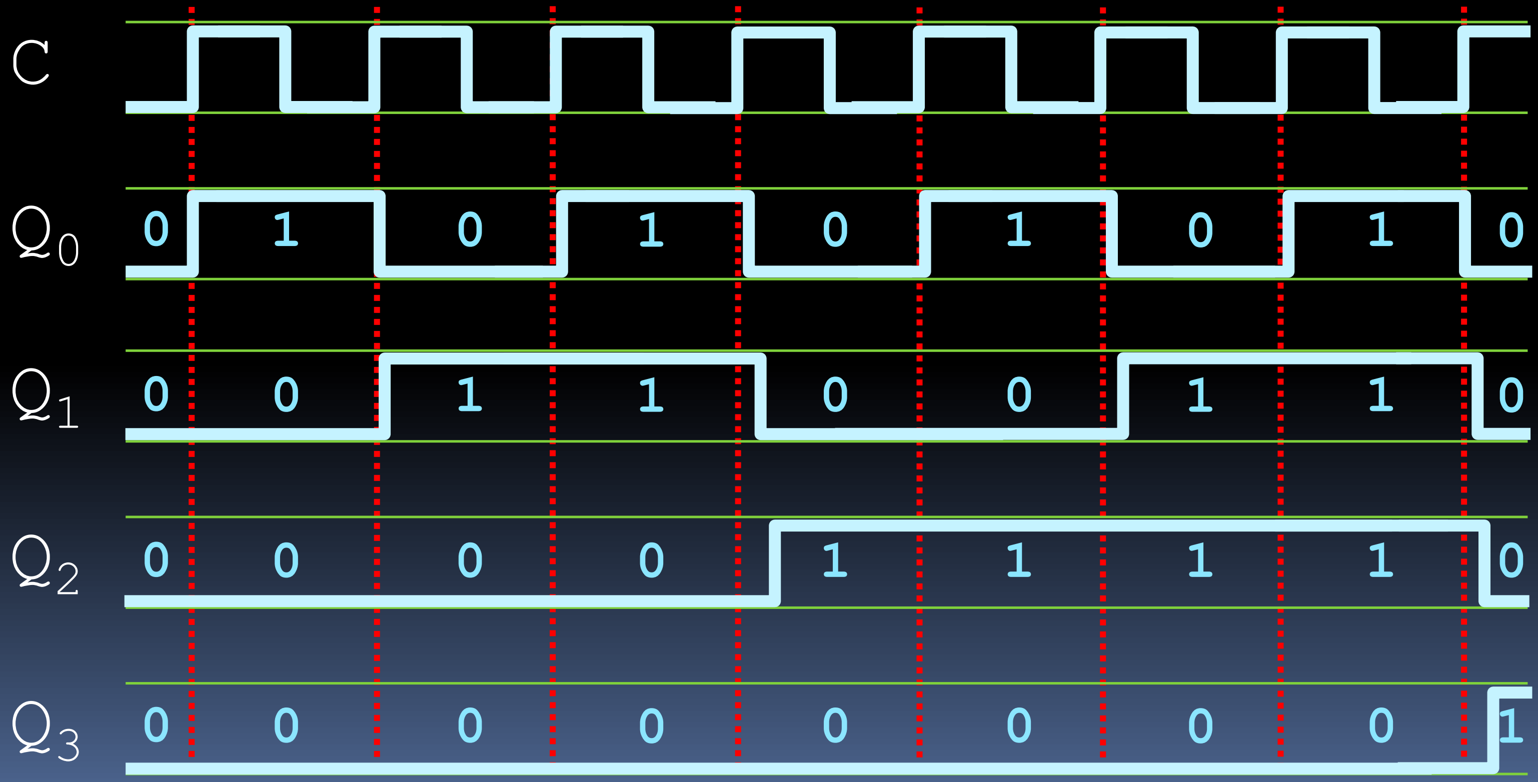


- This is a 4-bit ripple counter, which is an example of an asynchronous circuit.
  - Timing isn't quite synchronized with the rising clock pulse → hard to know when output is ready.
  - Cheap to implement, but unreliable for timing.

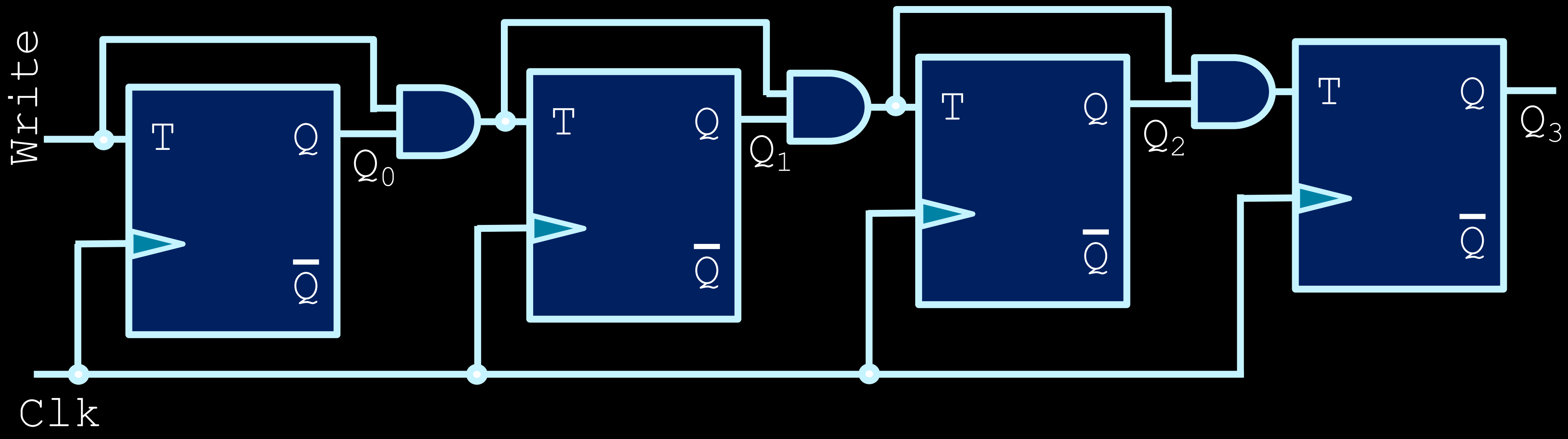
Counters



- Timing diagram
  - Note how propagation delay increases for later Qs



Counters

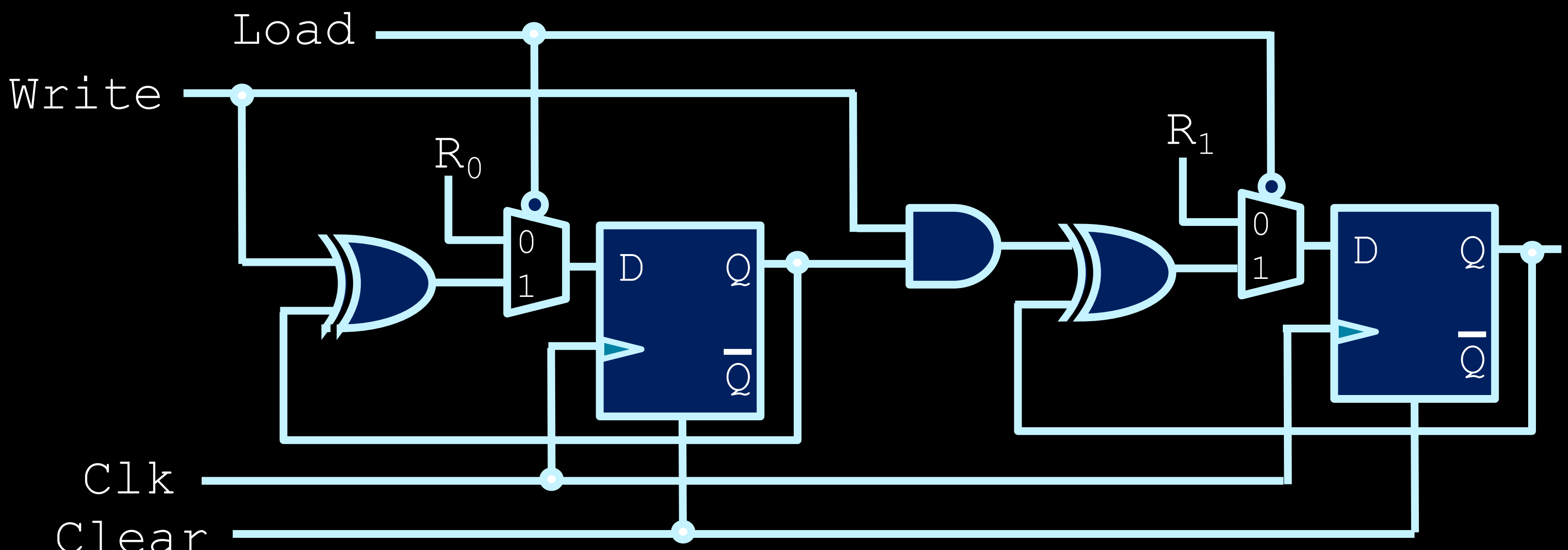


- This is a synchronous counter, with a slight delay.
- Each AND gate combine outputs of all previous flip-flops
- Each flip-flop only changes when all previous flip flops are set

It’s time for…

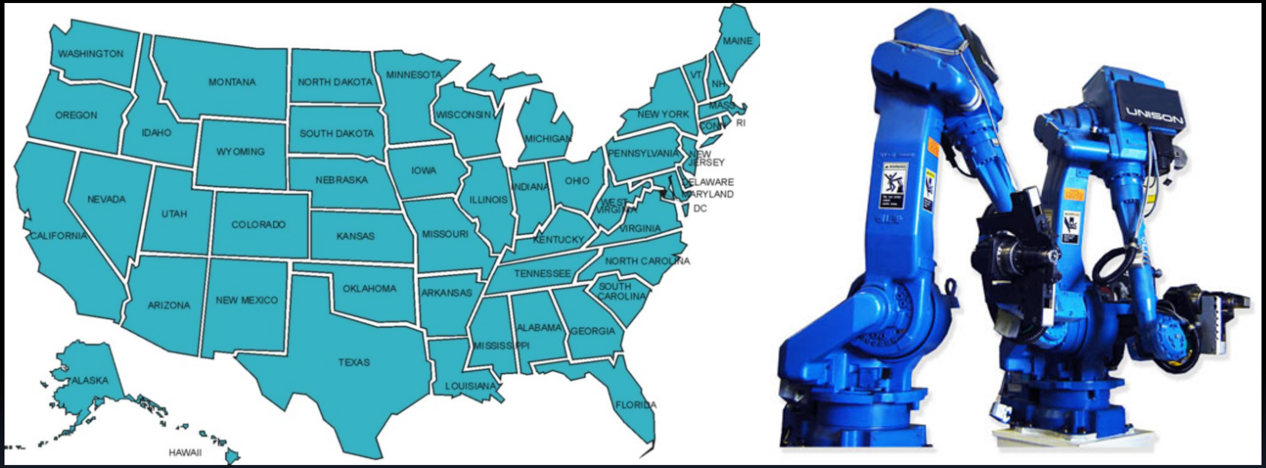
- **The Count!**

Example #3: Counters



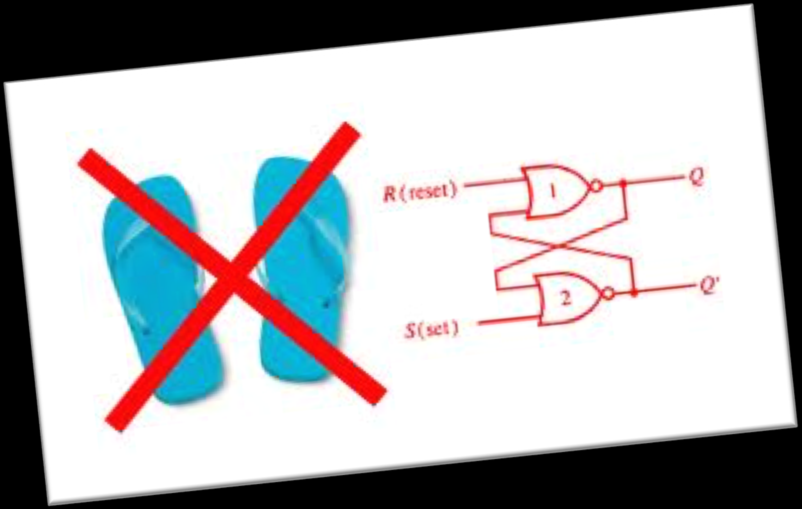
- Counters are often implemented with a parallel load and clear inputs.
  - Loading a counter value is used for countdowns.

State Machines



Designing with flip-flops

- Counters and registers are examples of how flip-flops can implement useful circuits that store values.



- How do you design these circuits?
- What would you design with these circuits?

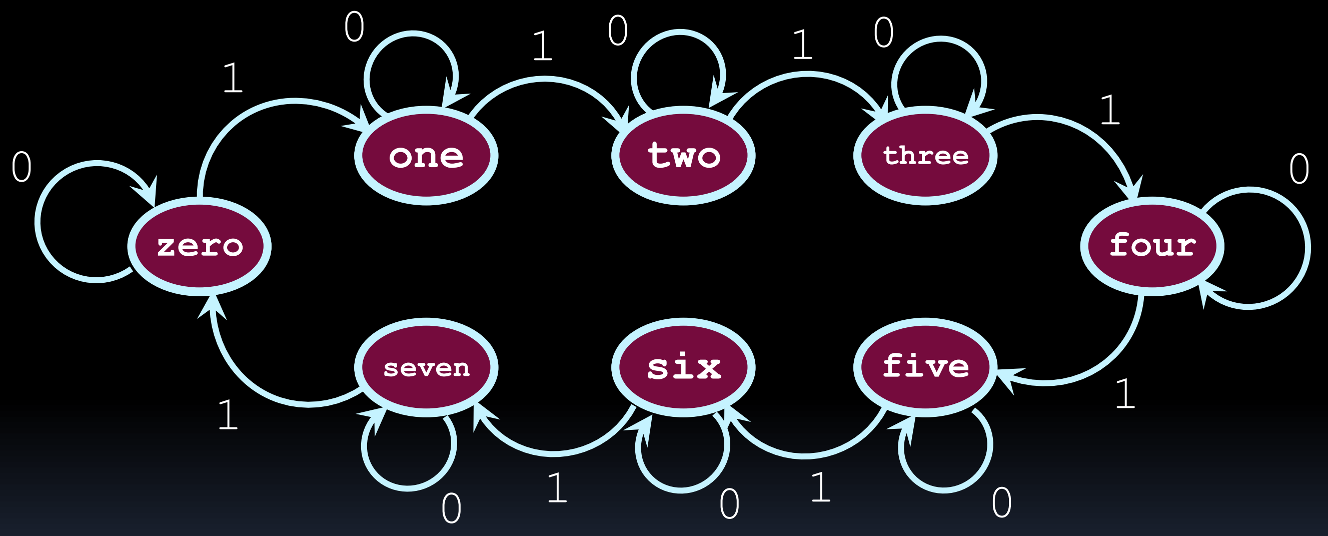
Designing with flip-flops

- Sequential circuits are the basis for memory, instruction processing, and any other operation that requires the circuit to remember past data values.
- These past data values are also called the states of the circuit.
- Sequential circuits use combinational logic to determine what the next state of the system should be, based on the past state and the current input values:

input + prev state → next state

State example: Counters

- With counters, each state is the current number that is stored in the counter.



- On each clock tick, the circuit transitions from one state to the next, based on the inputs.

State Tables

- State tables help to illustrate how the states of the circuit change with various input values.
  - Transitions are understood to take place on the clock ticks
  - (e.g., rising edge)

State	Write	State
zero	0	zero
zero	1	one
one	0	one
one	1	two
two	0	two
two	1	three
three	0	three
three	1	four
four	0	four
four	1	five
five	0	five
five	1	six
six	0	six
six	1	seven
seven	0	seven
seven	1	zero

State Tables

- Same table as on the previous slide, but with the actual flip-flop values instead of state labels.
    - Note: Flip-flop values are both inputs and outputs of the circuit here.

F_1	F_2	F_3	Write	F_1	F_2	F_3
0	0	0	0	0	0	0
0	0	0	1	0	0	1
0	0	1	0	0	0	1
0	0	1	1	0	1	0
0	1	0	0	0	1	0
0	1	0	1	0	1	1
0	1	1	0	0	1	1
0	1	1	1	1	0	0
1	0	0	0	1	0	0
1	0	0	1	1	0	1
1	0	1	0	1	0	1
1	0	1	1	1	1	0
1	1	0	0	1	1	0
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	0	0	0

and this brings us to…

Finite State Machines

As seen in other courses…

- You may have seen finite state machines before, but in a different context.
    - Used mainly to describe the grammars of a language, or to model sequence data.
- In CSCB58, finite state machines are models for an actual circuit design.
    - The states represent internal states of the circuit, which are stored in the flip-flop values.

Finite State Machines (FSMs)

- From theory courses…
  - A Finite State Machine is an abstract model that captures the operation of a sequential circuit.
- A FSM is defined (in general) as:
  - A finite set of states,
  - A finite set of transitions between states, triggered by inputs to the state machine,
  - Output values that are associated with each state or each transition (depending on the machine),
  - Start and end states for the state machine.

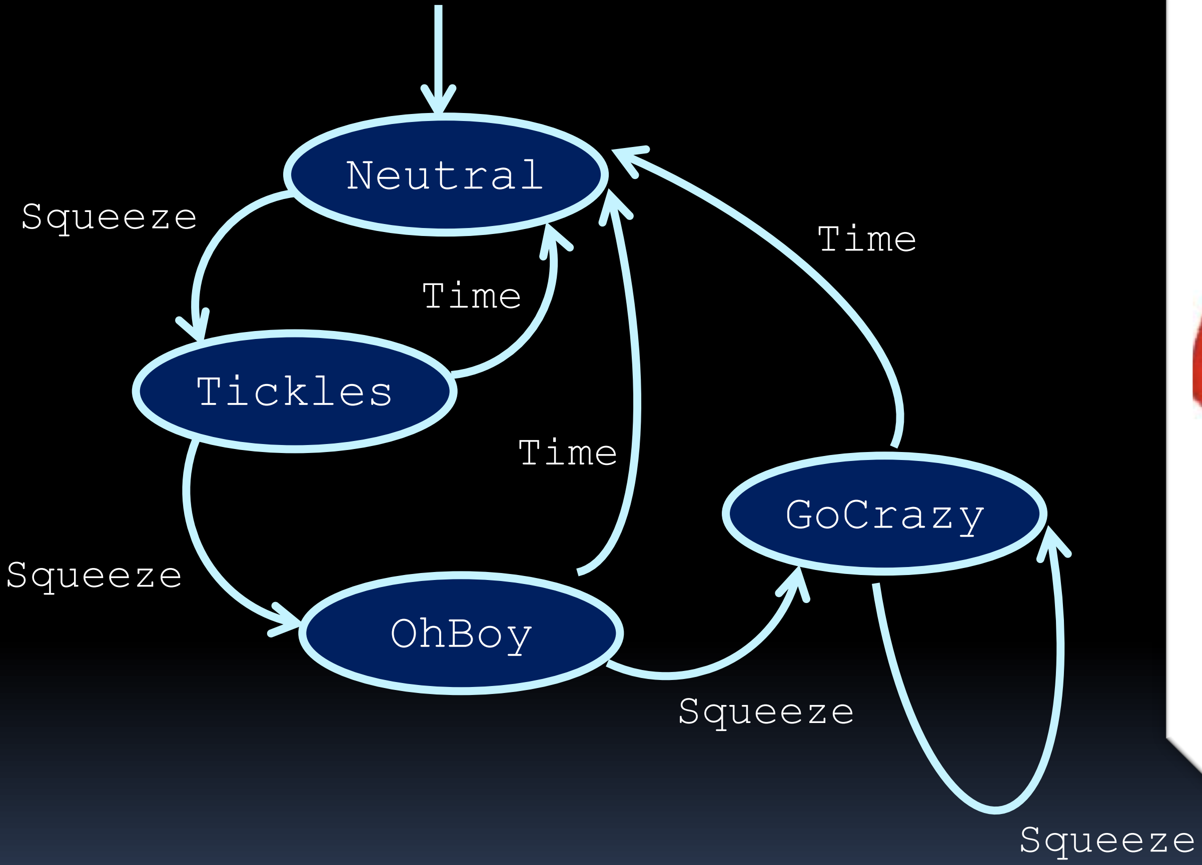
Example #1: Tickle Me Elmo

- Remember how the Tickle Me Elmo works!

Example #1: Tickle Me Elmo

- Toy reacts differently each time it is squeezed:
  - First squeeze → *"Ha ha ha…that tickles."*
  - Second squeeze → *"Ha ha ha…oh boy."*
  - Third squeeze → *"HA HA HA HA…HA HA HA HA…etc"*
- Questions to ask:
  - What are the inputs?
  - What are the states of this machine?
  - How do you change from one state to the next?
  - Who thought this is a good toy for children!?

Example #1: Tickle Me Elmo



Neutral

Squeeze

Time

Tickles

Time

Time

GoCrazy

Squeeze

OhBoy

Squeeze

Squeeze

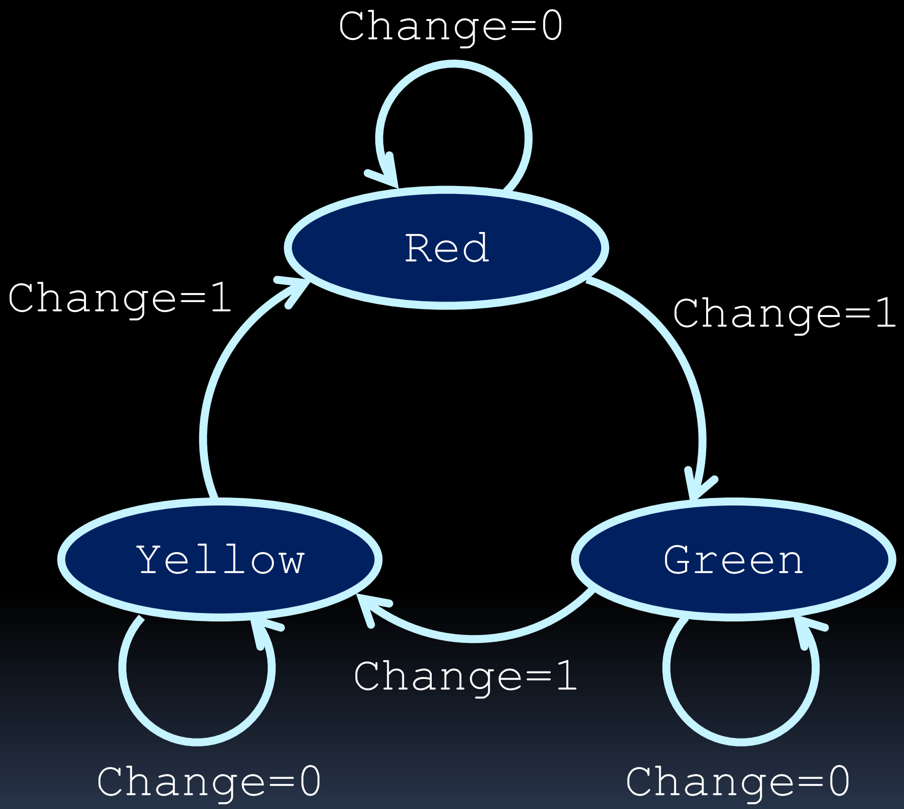
More elaborate FSMs

- Usually our FSM has more than one input, and will trigger a transition based on certain input values but not others.
- Also might have input values that don’t cause a transition, but keep the circuit in the same state (transitioning to itself).

Example #2: Alarm Clock

- Internal state description:
  - Starts in neutral state, until timer signal goes off.
    - Clock moves to alarm state.
  - Alarm state continues until:
    - snooze button is pushed (move to snooze state)
    - alarm is turned off (move to neutral state)
    - timer goes off again (move to neutral state)
  - In snooze state, clock returns to alarm state when the timer signal goes off again.

Example #3: Traffic Light



Change=0

Red

Change=1

Change=1

Yellow

Green

Change=1

Change=0

Change=0