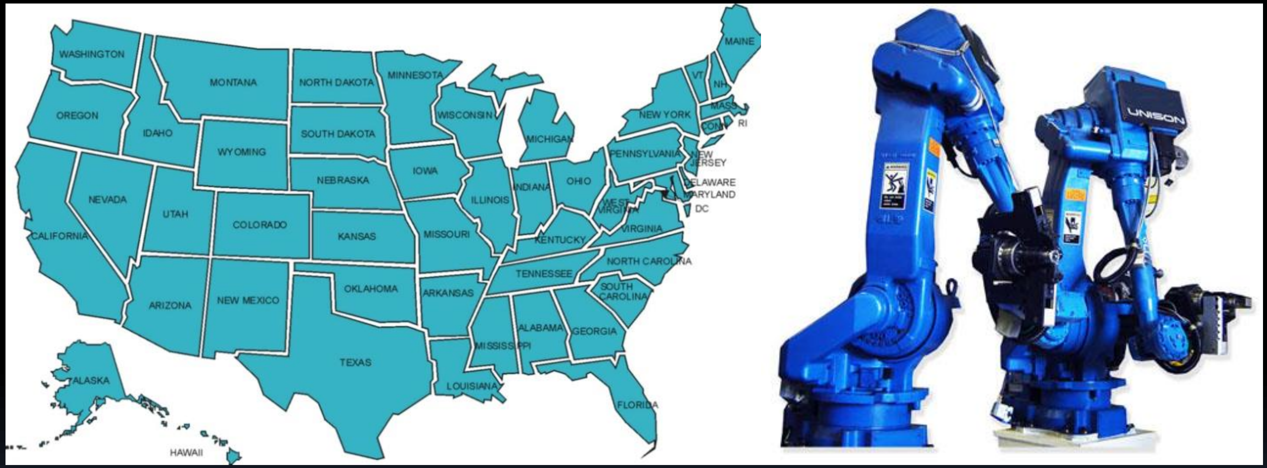
Week 5, part C:
State Machines

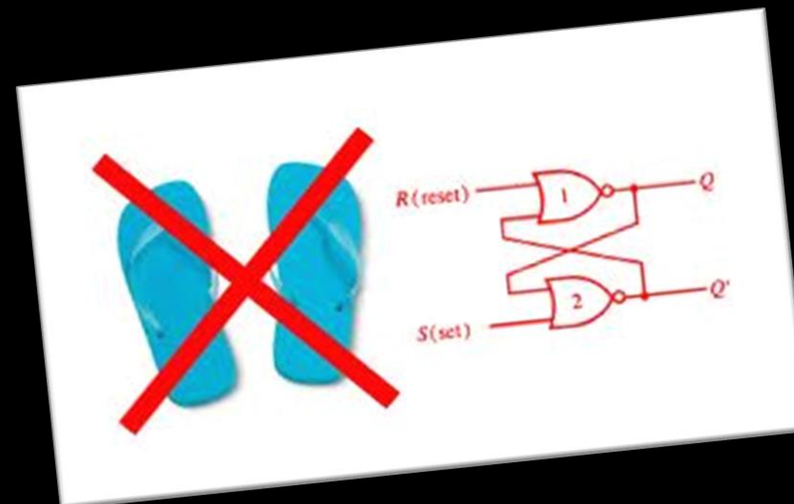


Reminder

- Sequential circuits are the basis for memory, instruction processing, and any other operation that requires the circuit to remember past data values.
- Our memory of the past is called the state of the circuit.

Designing with flip-flops

- We can use flip-flops to store bits of state for sequential circuits.

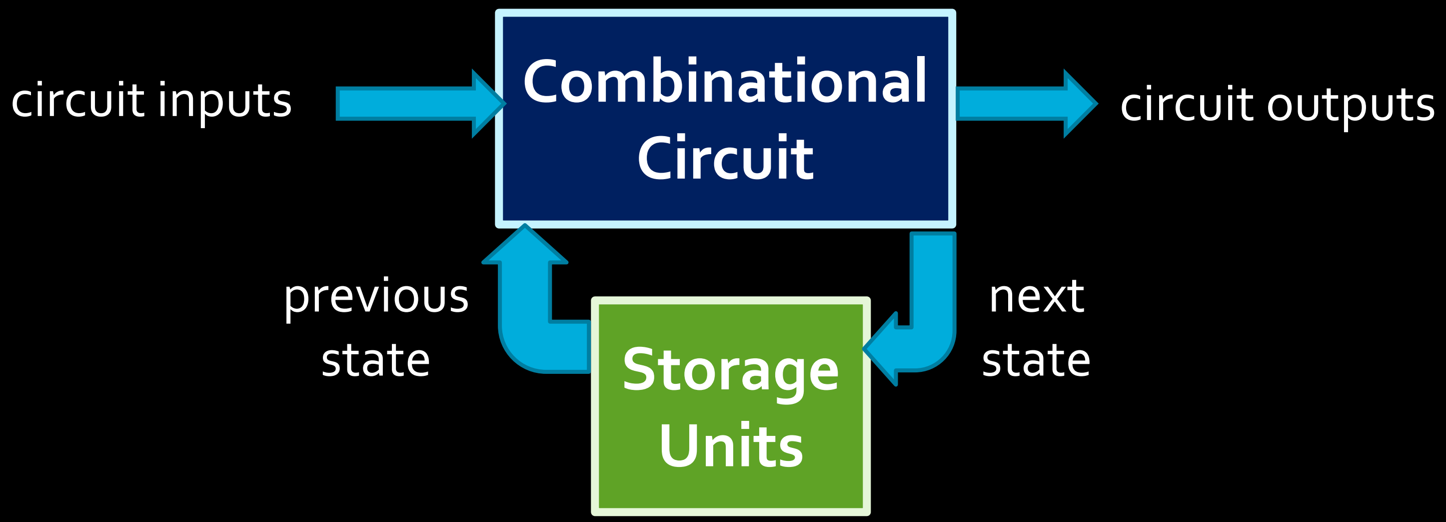


- But how do you design these circuits?

Designing with flip-flops

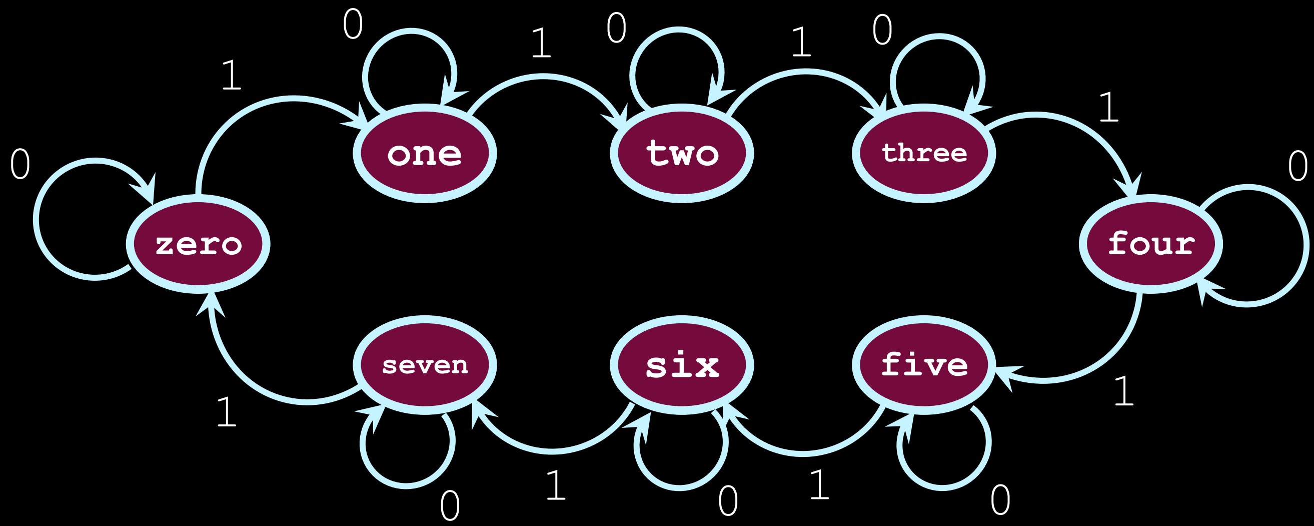
- Sequential circuits use combinational logic to determine what the next state of the system should be, based on the past state and the current input values:

input + previous 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)

prev

next

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.
  - <u>Note:</u> Flip-flop values are both inputs and outputs of the circuit here.

F_2	F_1	F_0	Write	F_2	F_1	F_0
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

Finite State Machines (FSMs)

- From theory courses…
  - A Finite State Machine is an abstract model that captures behaviour (e.g., 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.

As seen in other courses…

- You will see (or have seen) finite state machines in other context:
  - Grammars of a language
  - Modeling sequence data.
  - Modeling behaviour.
- 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.

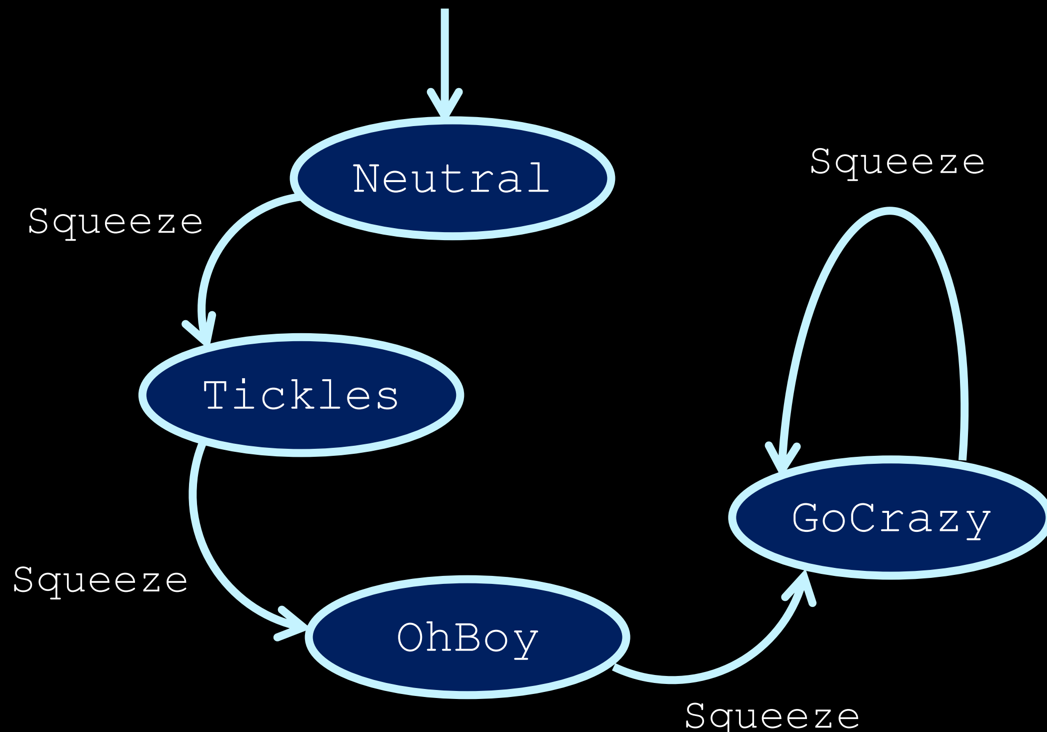
Example #1: Tickle Me Elmo

- 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



More elaborate FSMs

- Usually our FSM has more than one input, and will trigger a transition based on some input signals 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).
  - This is sometimes called self transition.

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.

Let’s Draw the State Machine

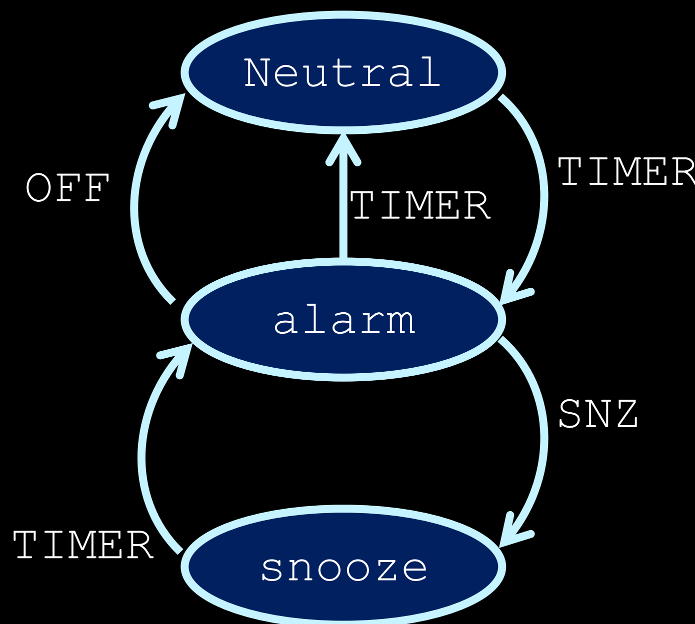
- Starts in neutral state, until timer signal goes off.
  - Clock moves to alarm state.
- Alarm state continues until one of:
  - snooze button is pushed (move to snooze state)
  - alarm is turned off (move to neutral state)
  - timer goes off again (move to neutral state)
- In snooze state, clock returns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

Let’s Draw the State Machine

- Starts in neutral state, until timer signal goes off.
  - Clock moves to alarm state.
- Alarm state continues until one of:
  - 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.

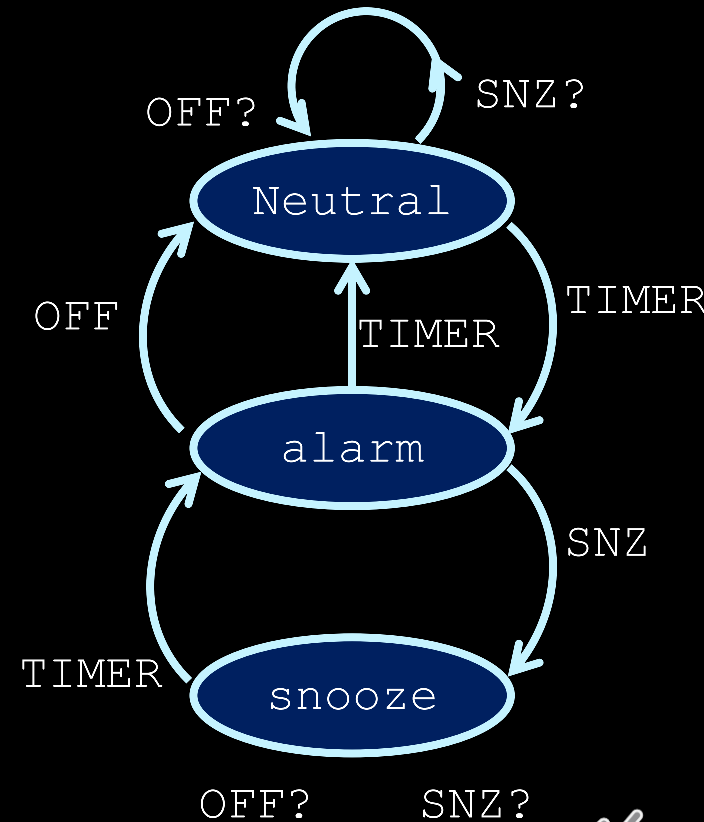
Let's Draw the State Machine

- Starts in neutral state, until timer signal goes off.
  - Clock moves to alarm state.
- Alarm state continues until one of:
  - 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.

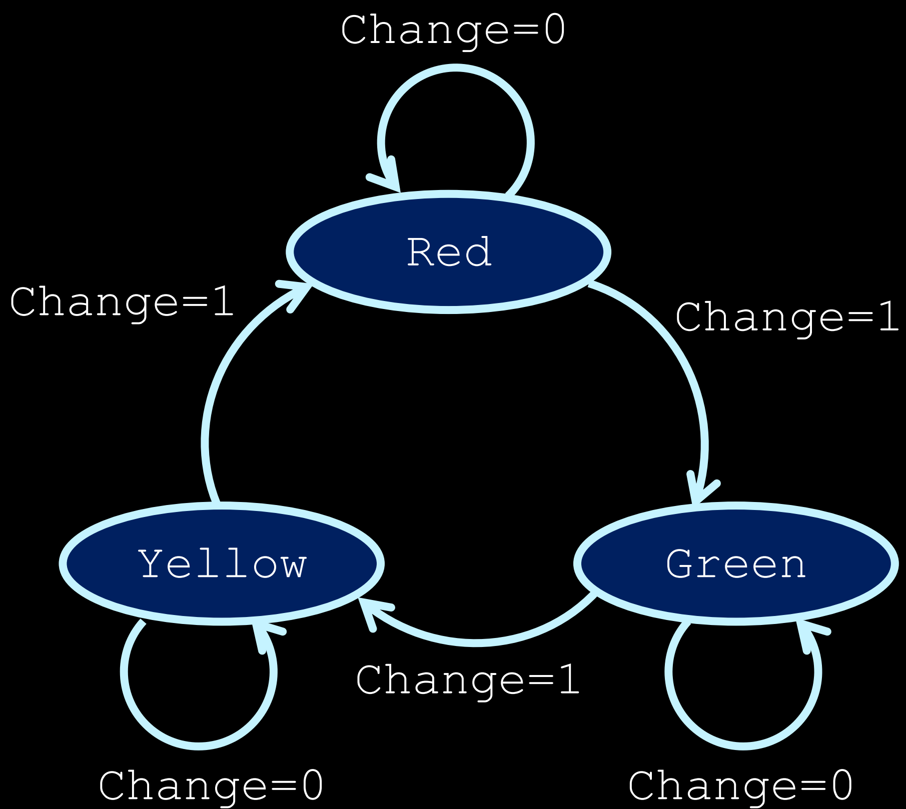


Let's Draw the State Machine

- Starts in neutral state, until timer signal goes off.
  - Clock moves to alarm state.
- Alarm state continues until one of:
  - 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



Finite State Machine

- A state machine should be complete including all potential inputs and self transitions.
- OK, so given a story, we can convert it to a state machine (diagram)
- But then what?
  - Move to next part!