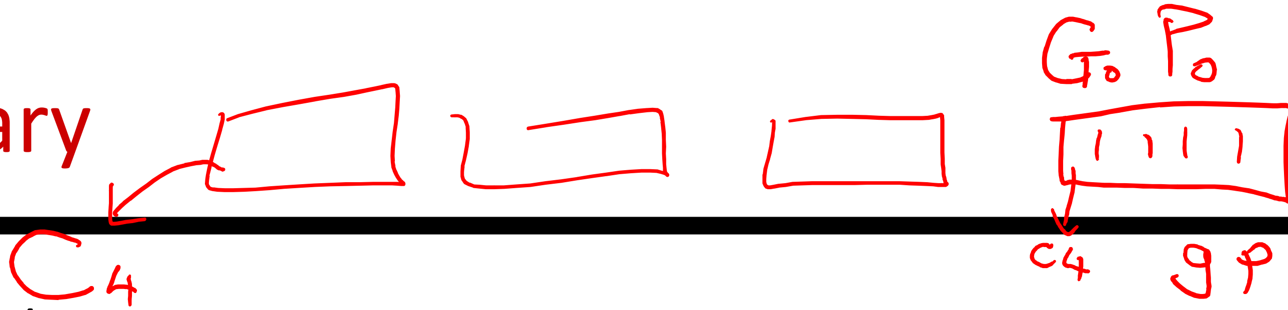
Lecture 14: Sequential Circuits, FSM

- Today's topics:
  - Adder wrap-up
  - Sequential circuits
  - Finite state machines

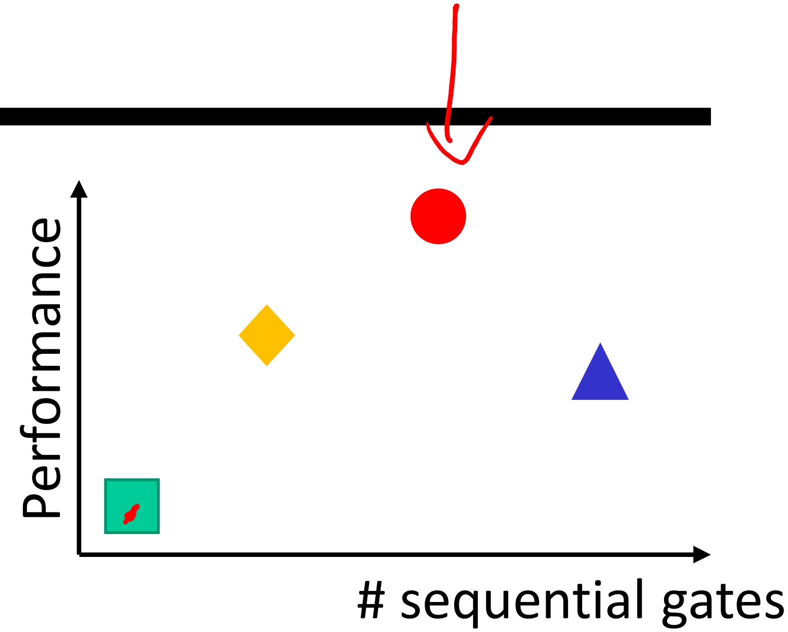
Adder Summary

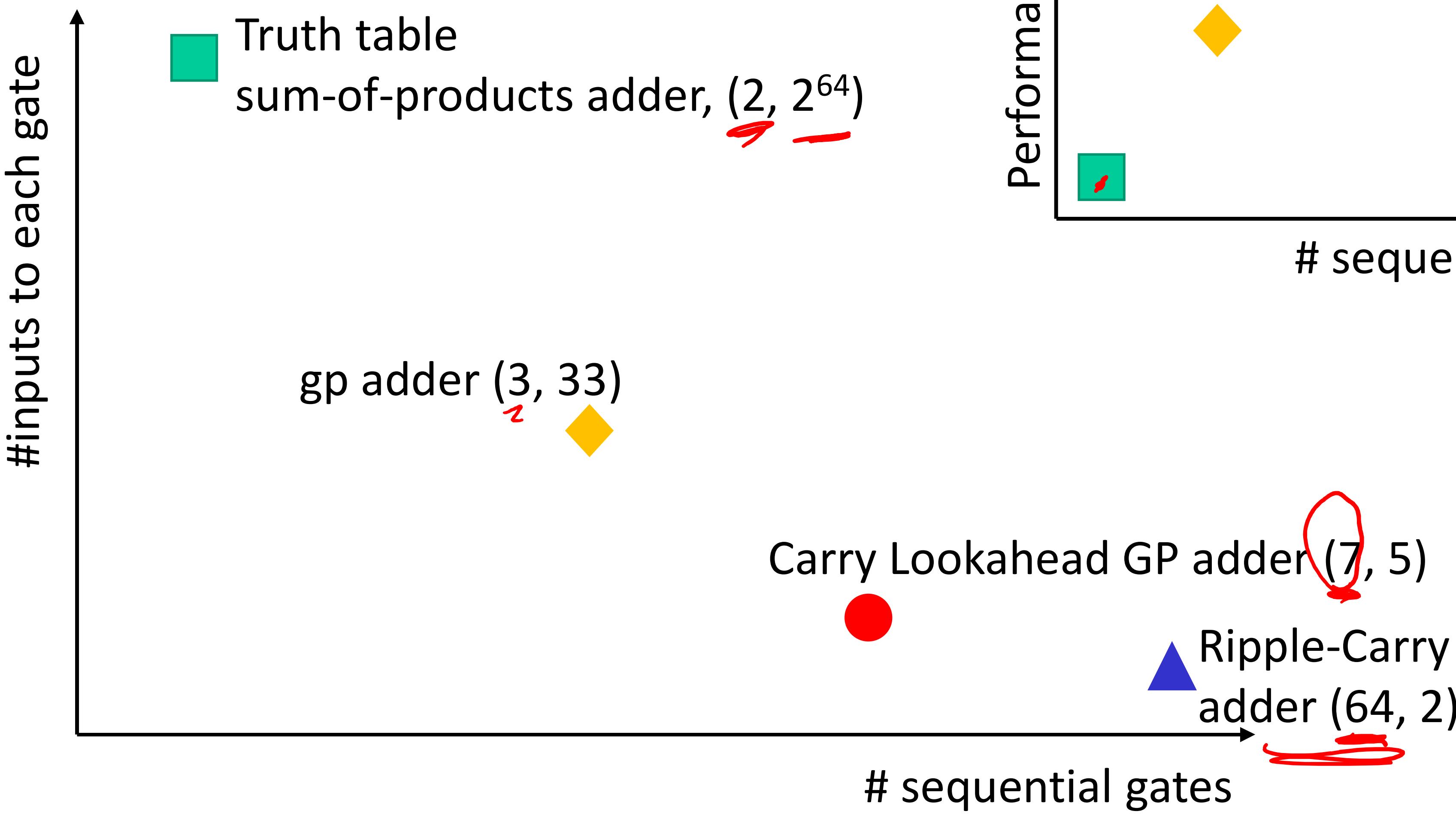


- Using the generate/propagate abstraction to add layers of ccts
- Key: all g/p/G/P signals can be calculated based on a/b inputs
  (they don't need carry-in as inputs, so they can all be done rightaway in parallel)
- First calculate g/p with 1 gate delay: gi = ai.bi ; pi = ai + bi
- Then calculate G/P with up to 2 gate delays (for a block of 4 bits):
  Gi = g3 + g2.p3 + g1.p2.p3 + g0.p1.p2.p3
  Pi = p0.p1.p2.p3
- Then calculate all the carries, including for the 16th bit, with 2 more gate delays:
  C4 = G3 + (P3.G2) + (P3.P2.G1) + (P3.P2.P1.G0) + (P3.P2.P1.P0.c0)
- Thus, this abstraction enables a design with a modest number of total gates, a modest number of delays, and a modest number of inputs per gate.

Trade-Off Curve

32b addition



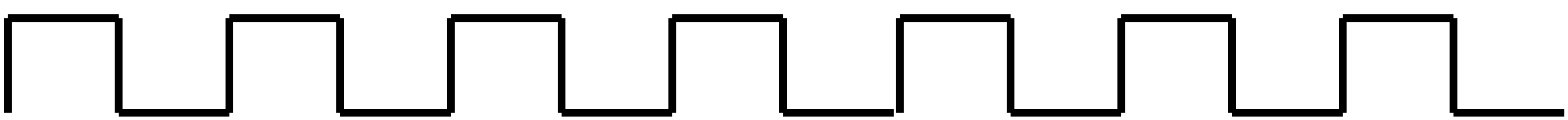


Clocks

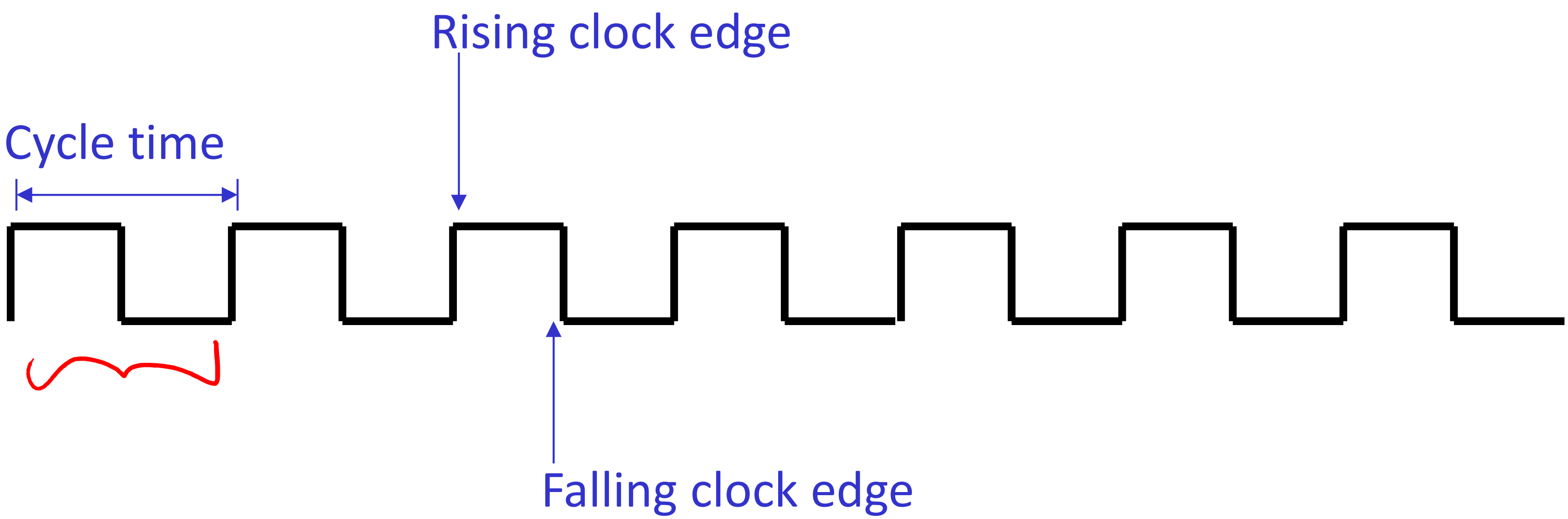
FSM (HW 6 – posted later today
Due next week)

- A microprocessor is composed of many different circuits that are operating simultaneously – if each circuit X takes in inputs at time TI_X, takes time TE_X to execute the logic, and produces outputs at time TO_X, imagine the complications in co-ordinating the tasks of every circuit

- A major school of thought (used in most processors built today): all circuits on the chip share a clock signal (a square wave) that tells every circuit when to accept inputs, how much time they have to execute the logic, and when they must produce outputs



Clock Terminology

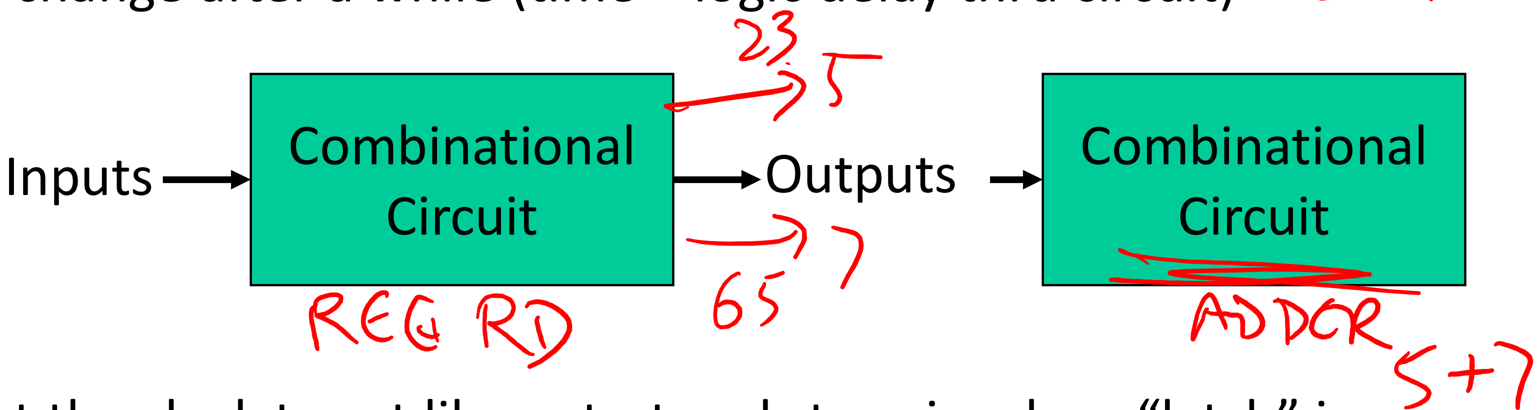


$$4 \text{ GHz} = \text{clock speed} = \frac{1}{\text{cycle time}} = \frac{1}{250 \text{ ps}}.$$

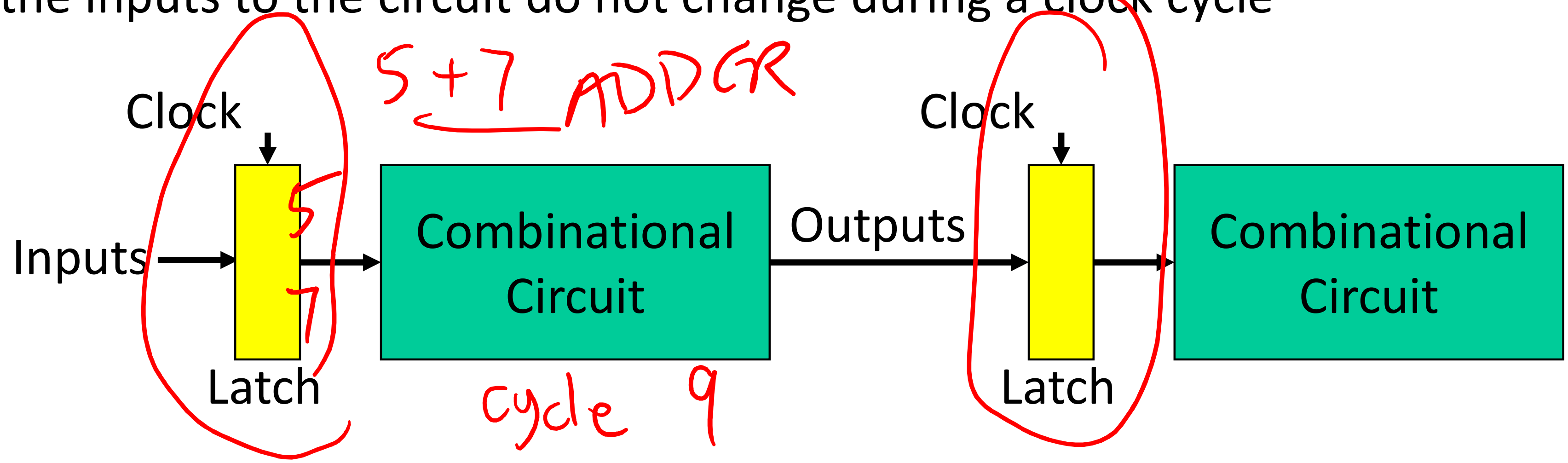
Sequential Circuits

ADDER 800ps
CCT X 300ps
CCT Y 500ps

- Until now, circuits were combinational – when inputs change, the outputs change after a while (time = logic delay thru circuit)



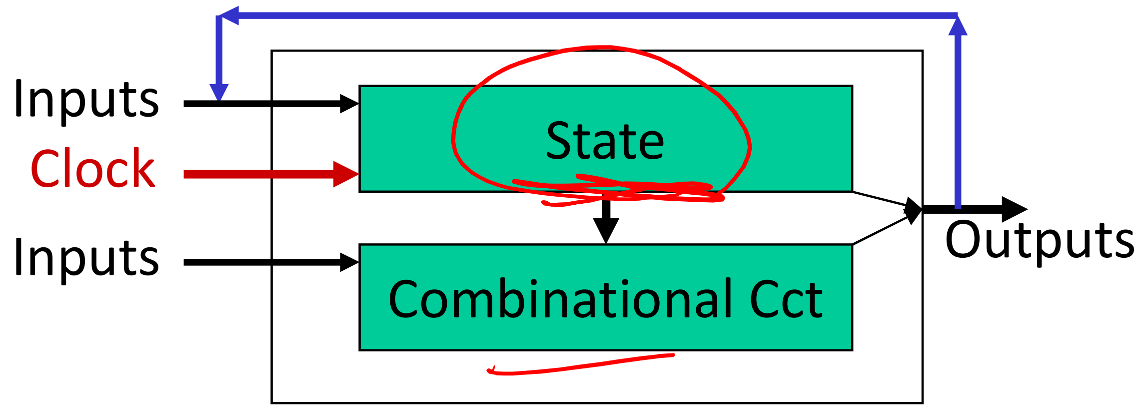
- We want the clock to act like a start and stop signal – a "latch" is a storage device that separates these circuits – it ensures that the inputs to the circuit do not change during a clock cycle



Sequential Circuits

- Sequential circuit: consists of combinational circuit and a storage element

- At the start of the clock cycle, the rising edge causes the "state" storage to store some input values



- This state will not change for an entire cycle (until next rising edge)

- The combinational circuit has some time to accept the value of "state" and "inputs" and produce "outputs"

- Some of the outputs (for example, the value of next "state") may feed back (but through the latch so they're only seen in the next cycle)

Designing a Latch

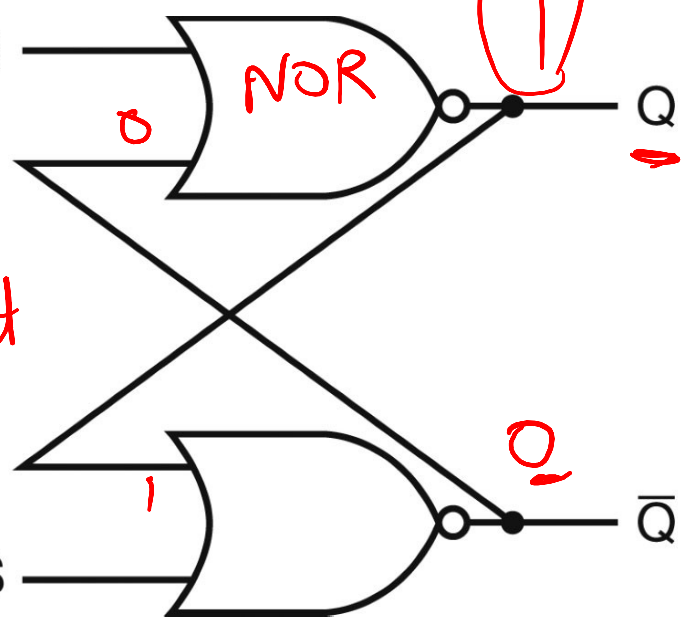
⇒ SET = 1 ⇒ then Q = 1
RESET = ~~0~~ 1 ⇒ then Q = 0

- An S-R latch: set-reset latch
  - When Set is high, a 1 is stored
  - When Reset is high, a 0 is stored
  - When both are low, the previous state is preserved (hence, known as a storage or memory element)
  - Both are high – this set of inputs is not allowed

Verify the above behavior!

If R = 0, S = 0
outputs are unchanged
it remembers the last output

0 = R



output

0 ⇐ 1 = S

Source: H&P textbook

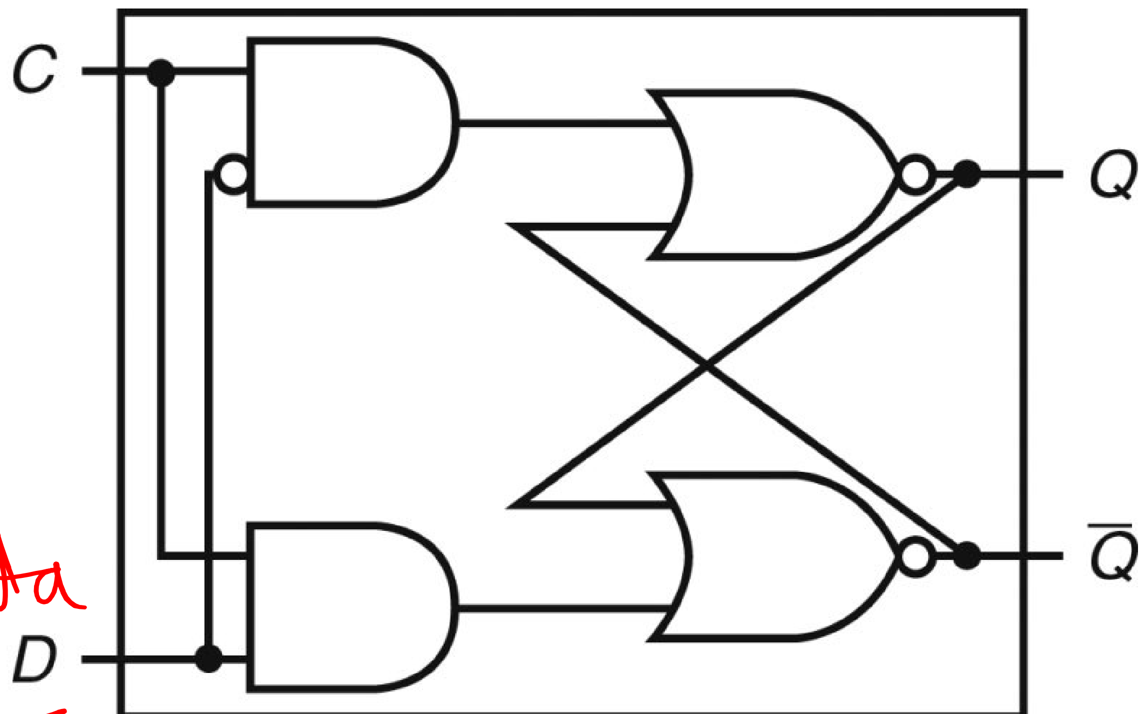
D Latch

when clk=1
input D & $\bar{D}$
being sent to
SR latch

when clk=0
inputs to
SR-latch
are both 0

- Incorporates a clock
- The value of the input D signal (data) is stored only when the clock is high – the previous state is preserved when the clock is low

remember
state



when clk=1,
Q=D

when clk=0,
Q=Remembe

Source: H&P textbook

D Flip Flop

→Q' captures the last D seen

L2 accepting inputs

Q = Q' = D seen at this pt

Q' = D

L2 not accepting inputs

L1 not accepting inputs

Remembers last D seen

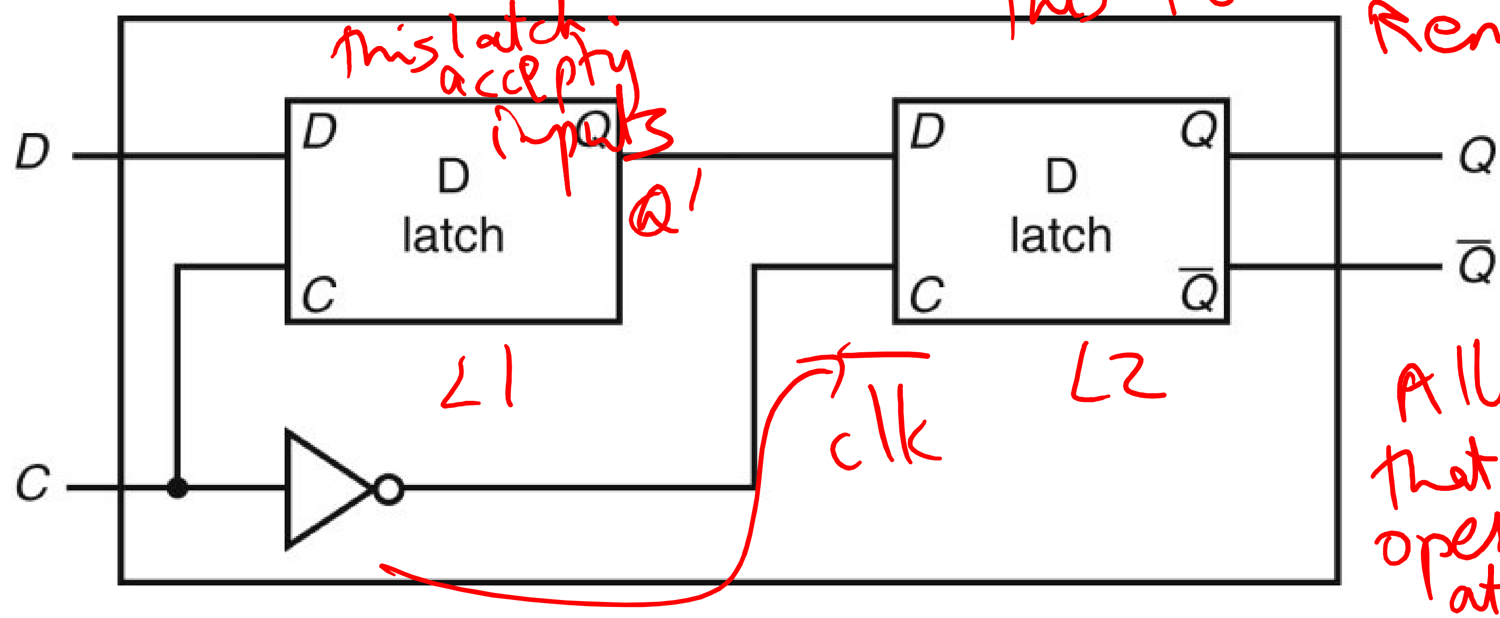
- Terminology:
  Latch: outputs can change any time the clock is high (asserted)
  Flip flop: outputs can change only on a clock edge

- Two D latches in series – ensures that a value is stored only on the falling edge of the clock

this latch has clk=0

Remember state

this latch accepting inputs



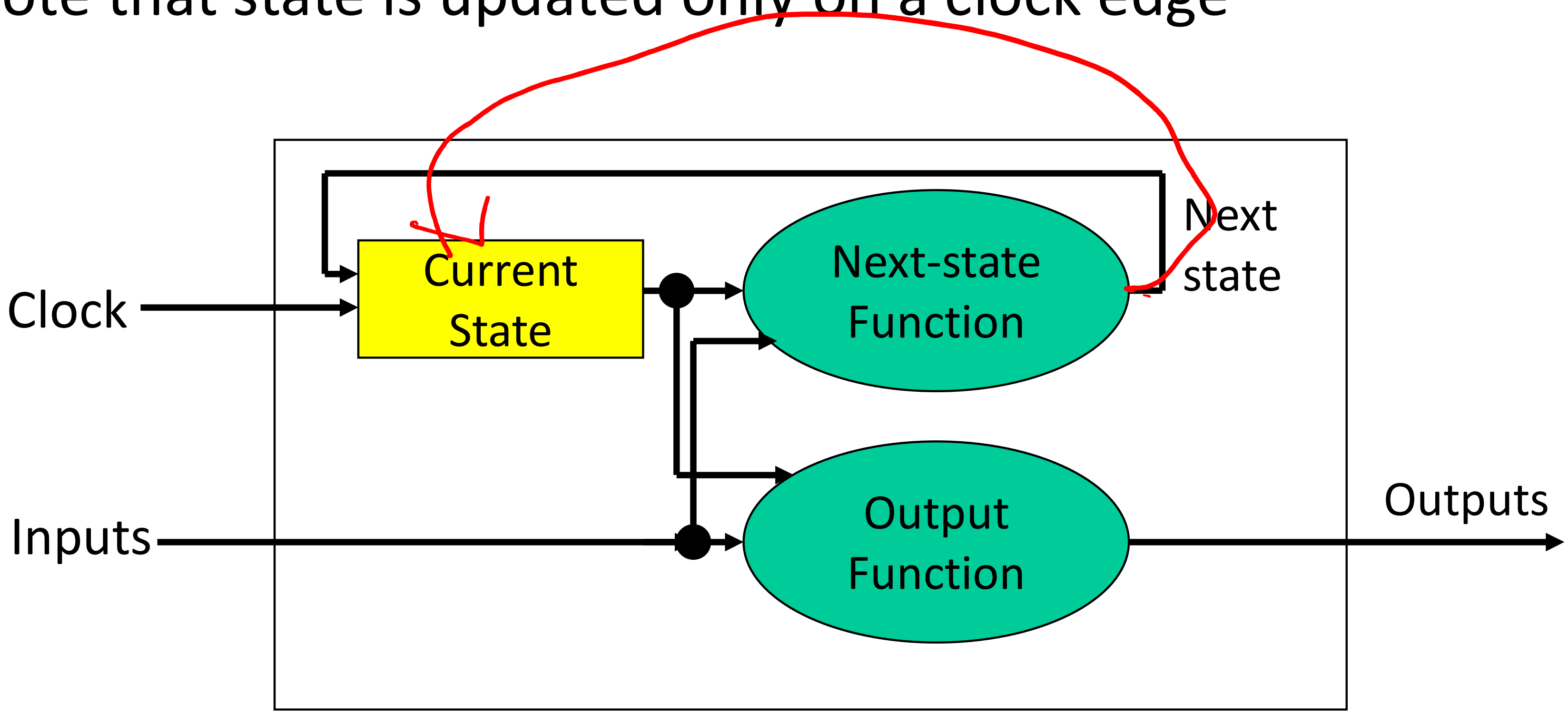
Alligator that snaps open & shut at every falling edge

Source: H&P textbook

Finite State Machine

- A sequential circuit is described by a variation of a truth table – a finite state diagram (hence, the circuit is also called a finite state machine)

- Note that state is updated only on a clock edge

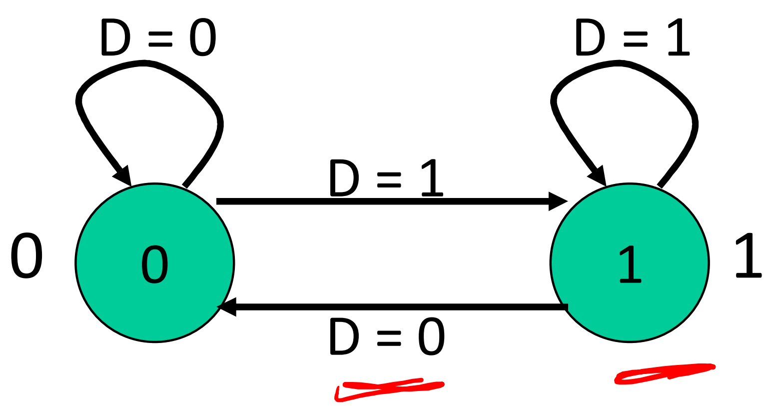


State Diagrams

Multiple States

- Each state is shown with a circle, labeled with the state value – the contents of the circle are the outputs
- An arc represents a transition to a different state, with the inputs indicated on the label

1 input D has 2 values 0, 1



This is a state diagram for ___?

Latch

3-Bit Counter

- Consider a circuit that stores a number and increments the value on every clock edge – on reaching the largest value, it starts again from 0

  Draw the state diagram:
  - How many states?
  - How many inputs?

8

0, 1, 2 7

values of the single output

input ?

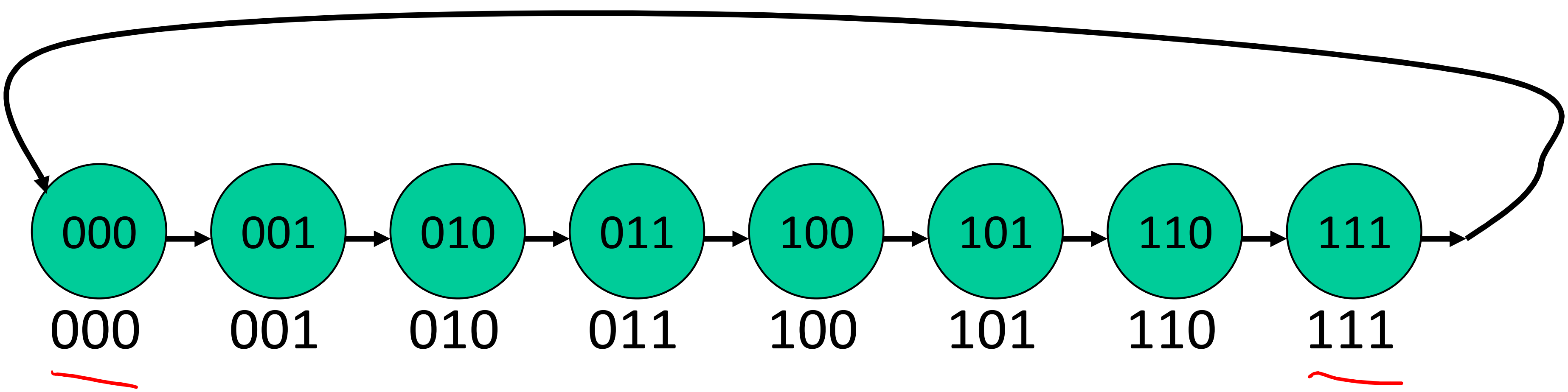
no input

(clk is kinda input)

3-Bit Counter

- Consider a circuit that stores a number and increments the value on every clock edge – on reaching the largest value, it starts again from 0

  Draw the state diagram:
  - How many states?
  - How many inputs?

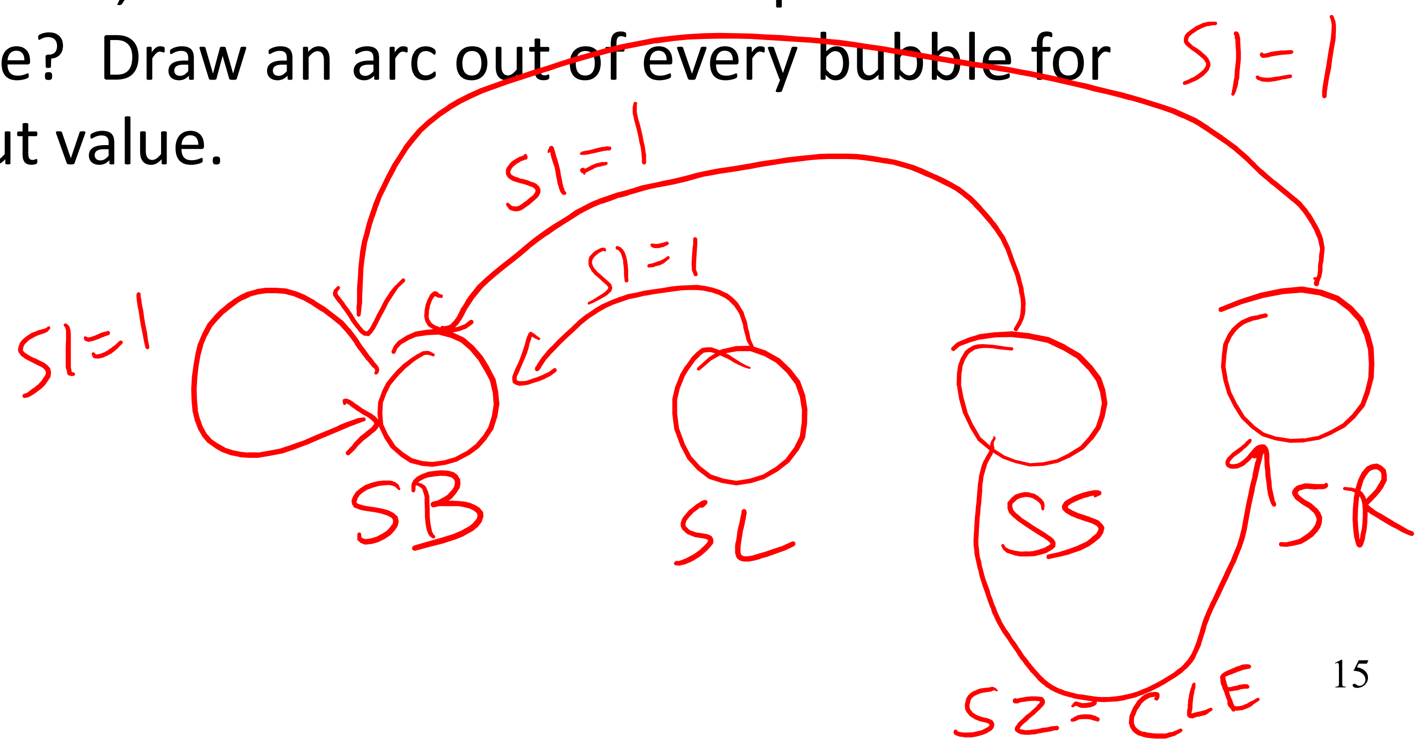


Tackling FSM Problems

Ped sensor S1 1 or 0

Rd edge sensor S2 CRE CLE MID

- Three questions worth asking:
  - What are the possible output states? Draw a bubble for each.
  - What are inputs? What values can those inputs take?
  - For each state, what do I do for each possible input value? Draw an arc out of every bubble for every input value.



Traffic Light Controller

clock with a cycle time of 30 sec

- Problem description: A traffic light with only green and red; either the North-South road has green or the East-West road has green (both can't be red); there are detectors on the roads to indicate if a car is on the road; the lights are updated every 30 seconds; a light need change only if a car is waiting on the other road

State Transition Table:

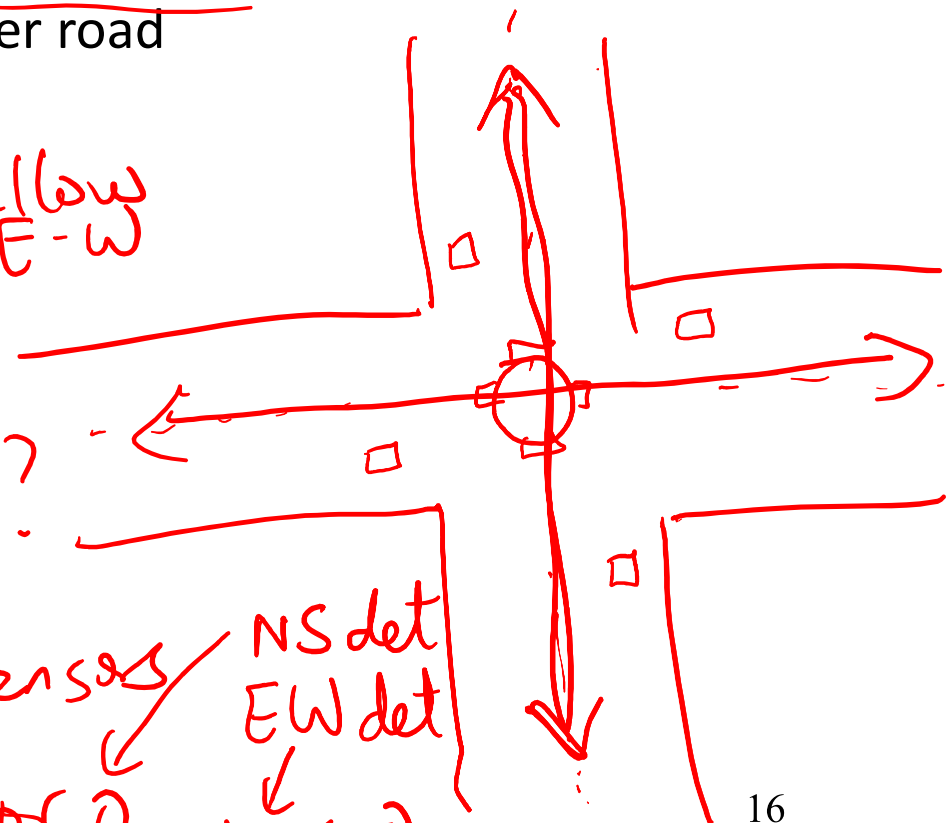
How many states? 2 allow N-S allow E-W

How many inputs?

How many outputs? 2

How many sensor inputs?

4, but logically can be viewed as 2 sensors



NS det

EW det

1 or 0

1 or 0

State Transition Table

state – 2 val, NS or EW
sensor EWdet – 2 val
NSdet – 2 val

- Problem description: A traffic light with only green and red; either the North-South road has green or the East-West road has green (both can't be red); there are detectors on the roads to indicate if a car is on the road; the lights are updated every 30 seconds; a light must change only if a car is waiting on the other road

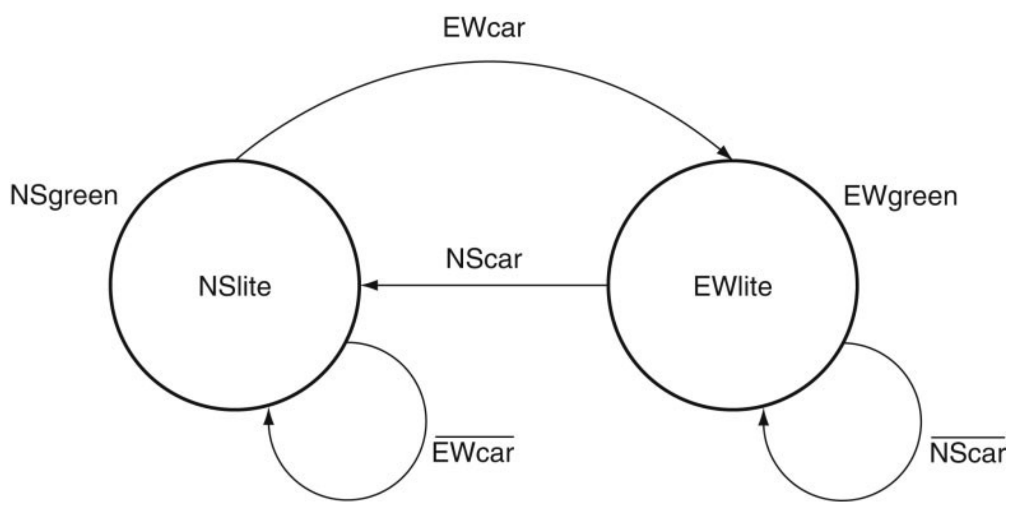
State Transition Table:

CurrState	InputEW	InputNS	NextState=Output
N	0	0	N
N	0	1	N
N	1	0	E
N	1	1	E
E	0	0	E
E	0	1	N
E	1	0	E
E	1	1	N

State Diagram

State Transition Table:

CurrState	InputEW	InputNS	NextState=Output
N	0	0	N
N	0	1	N
N	1	0	E
N	1	1	E
E	0	0	E
E	0	1	N
E	1	0	E
E	1	1	N



Source: H&P textbook

Tackling FSM Problems

- Three questions worth asking:
  - What are the possible output states? Draw a bubble for each.
  - What are inputs? What values can those inputs take?
  - For each state, what do I do for each possible input value? Draw an arc out of every bubble for every input value.

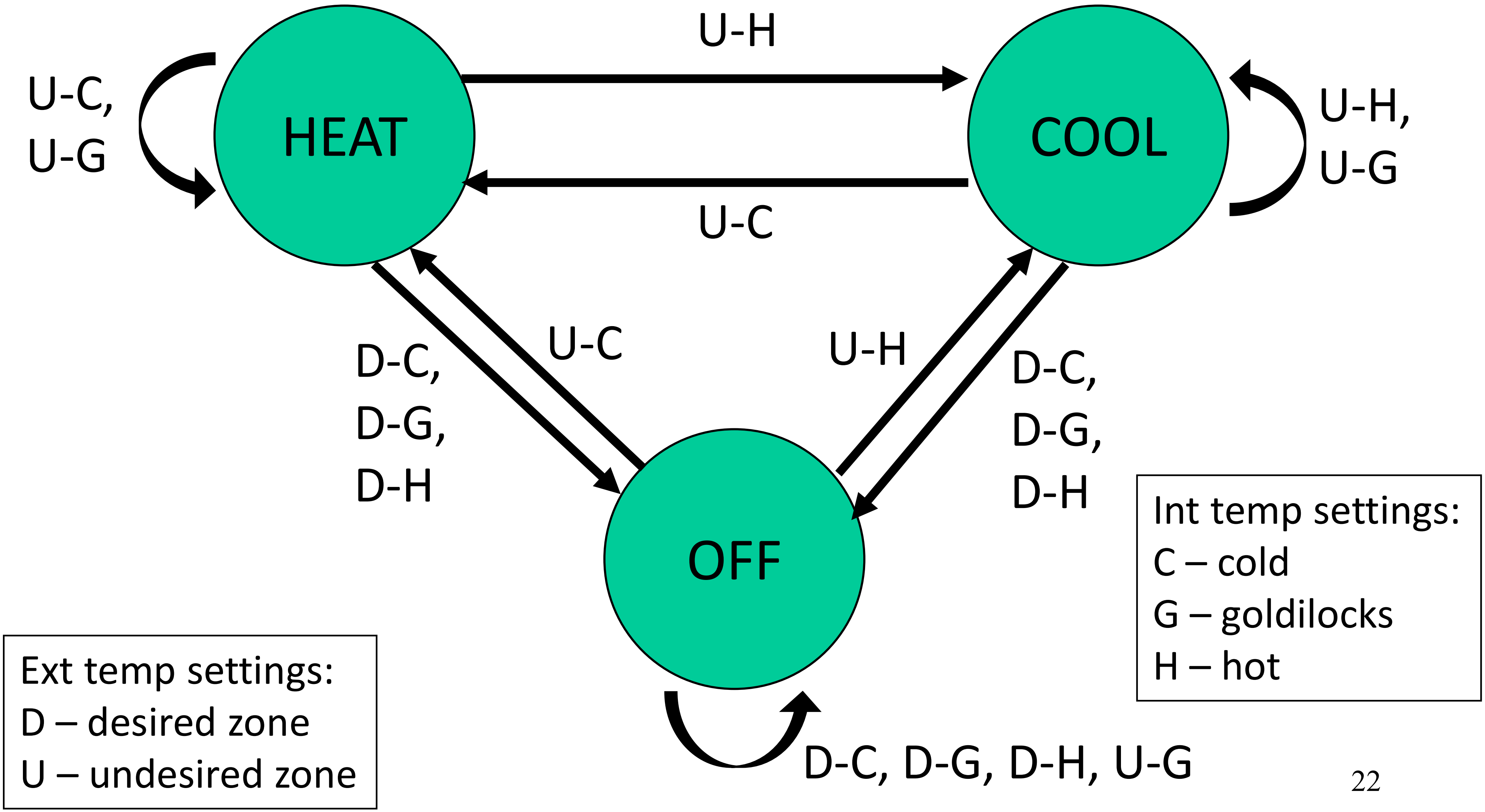
Example – Residential Thermostat

- Two temp sensors: internal and external
- If internal temp is within 1 degree of desired, don’t change setting
- If internal temp is > 1 degree higher than desired, turn AC on; if internal temp is < 1 degree lower than desired, turn heater on
- If external temp and desired temp are within 5 degrees, disregard the internal temp, and turn both AC and heater off

Finite State Machine Table

Current State	Input E	Input I	Output State
HEAT	D	C	OFF
HEAT	D	G	OFF
HEAT	D	H	OFF
HEAT	U	C	HEAT
HEAT	U	G	HEAT
HEAT	U	H	COOL
COOL	D	C	OFF
COOL	D	G	OFF
COOL	D	H	OFF
COOL	U	C	HEAT
COOL	U	G	COOL
COOL	U	H	COOL
OFF	D	C	OFF
OFF	D	G	OFF
OFF	D	H	OFF
OFF	U	C	HEAT
OFF	U	G	OFF
OFF	U	H	COOL

Finite State Diagram



Latch vs. Flip-Flop

- Recall that we want a circuit to have stable inputs for an entire cycle – so I want my new inputs to arrive at the start of a cycle and be fixed for an entire cycle

- A flip-flop provides the above semantics (a door that swings open and shut at the start of a cycle)

- But a flip-flop needs two back-to-back D-latches, i.e., more transistors, delay, power

- You can reduce these overheads with just a single D-latch (a door that is open for half a cycle) as long as you can tolerate stable inputs for just half a cycle