CS230: Digital Logic Design and Computer Architecture

Lecture 4: Sequential Circuits

https://www.cse.iitb.ac.in/~biswa/courses/CS230/main.html

https://www.cse.iitb.ac.in/~biswa/

Phones (smart/non-smart) on silence plz, Thanks

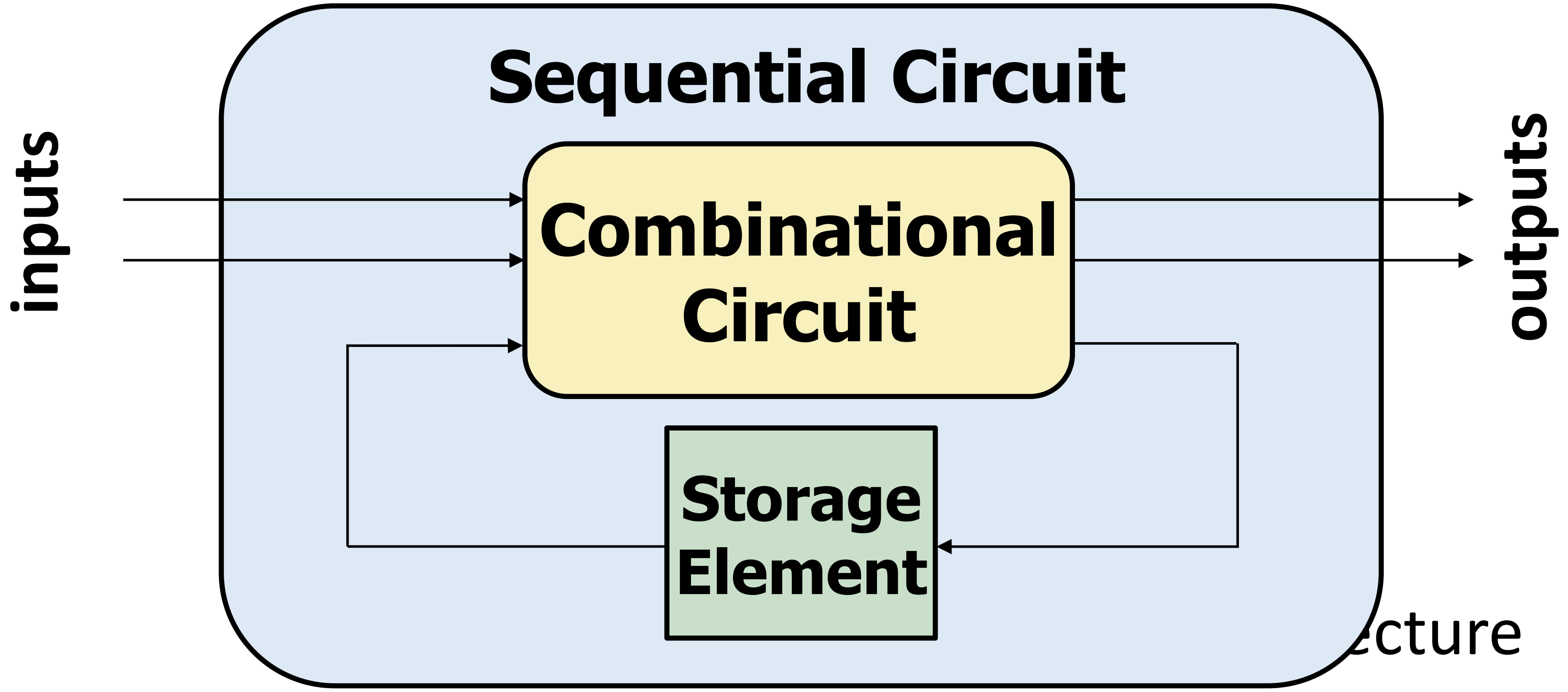
Logistics

- Join Piazza now, plz
- Lab-1 will be up after today's lecture
- Do come with your queries for Monday's lab.
- Tutorial for selected students on Monday 4:20 PM, Try the problem set-I before coming to tutorial

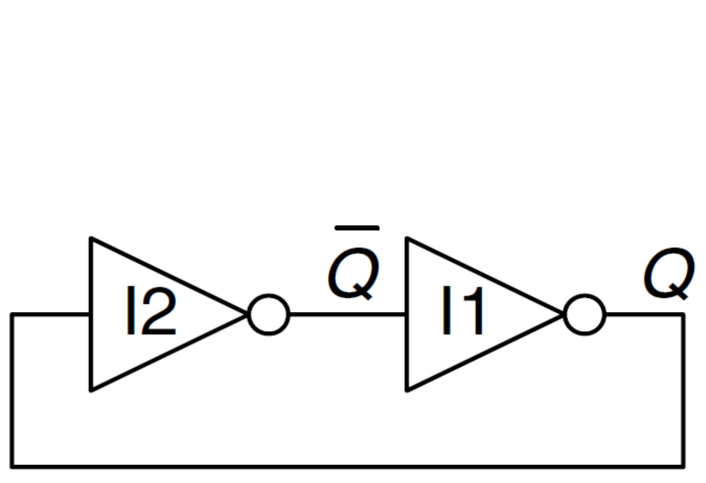
Quiz-1 on January 27

Sequential Circuit

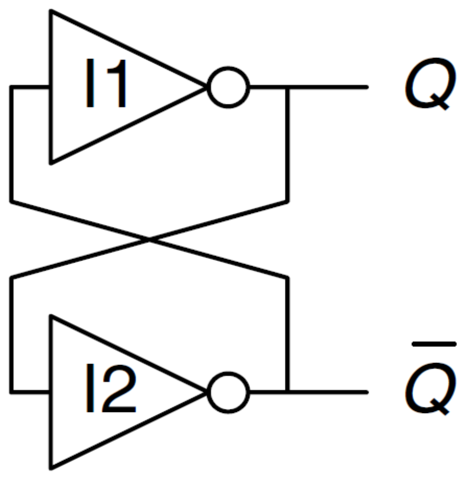
- Combinational circuit output depends **only** on current input
- We want circuits that produce output depending on **current** and **past** input values – circuits with **memory**
- How can we design a circuit that **stores information**?



The base for any storage



(a)

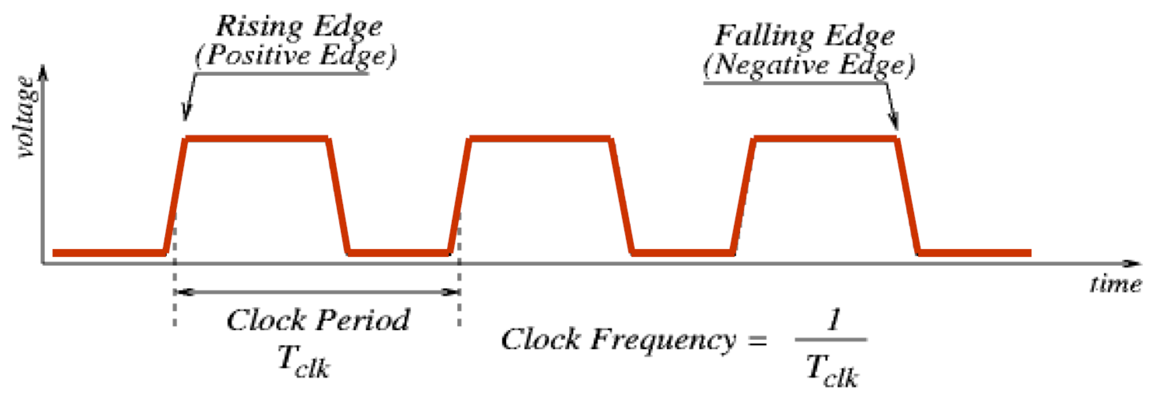


(b)

If Q is zero then Q complement is 1. Note that the circuit has no inputs ☺

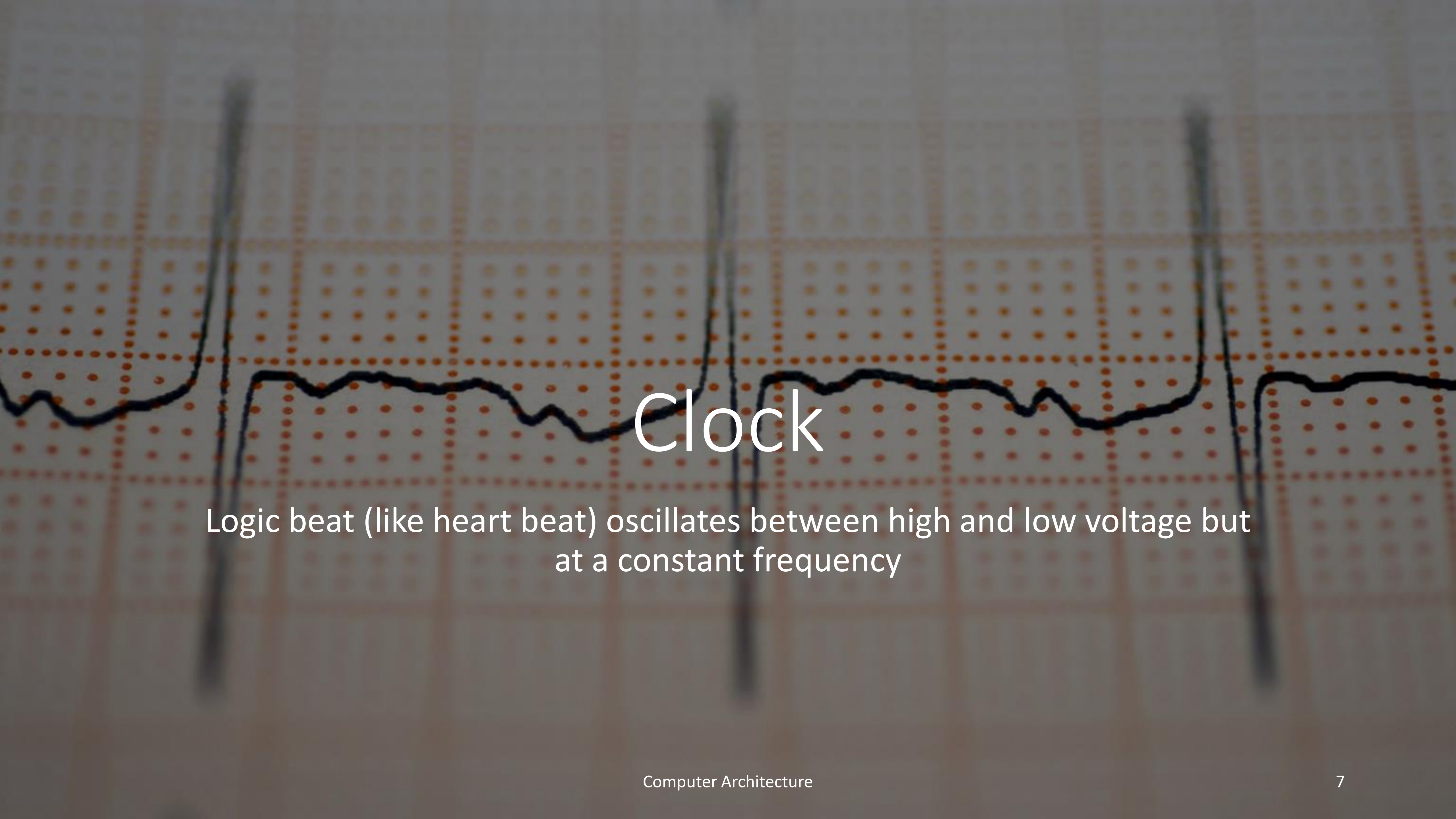
The Clock as some need it

- Clock signals are usually periodic.



- Duty cycle = ON Time / Clock Period
- Frequency = 1/Time Period
  - Units are in Hz

Clock is driven by the slowest combinational circuit/path.
Clock is responsible for triggering a state change



Clock

Logic beat (like heart beat) oscillates between high and low voltage but at a constant frequency

Clock and Storage Elements

- Storage elements are affected only at the arrival of a clock pulse.
- Storage elements are usually called as a latch/flip-flop.
- They maintain a binary state until directed by a clock pulse.

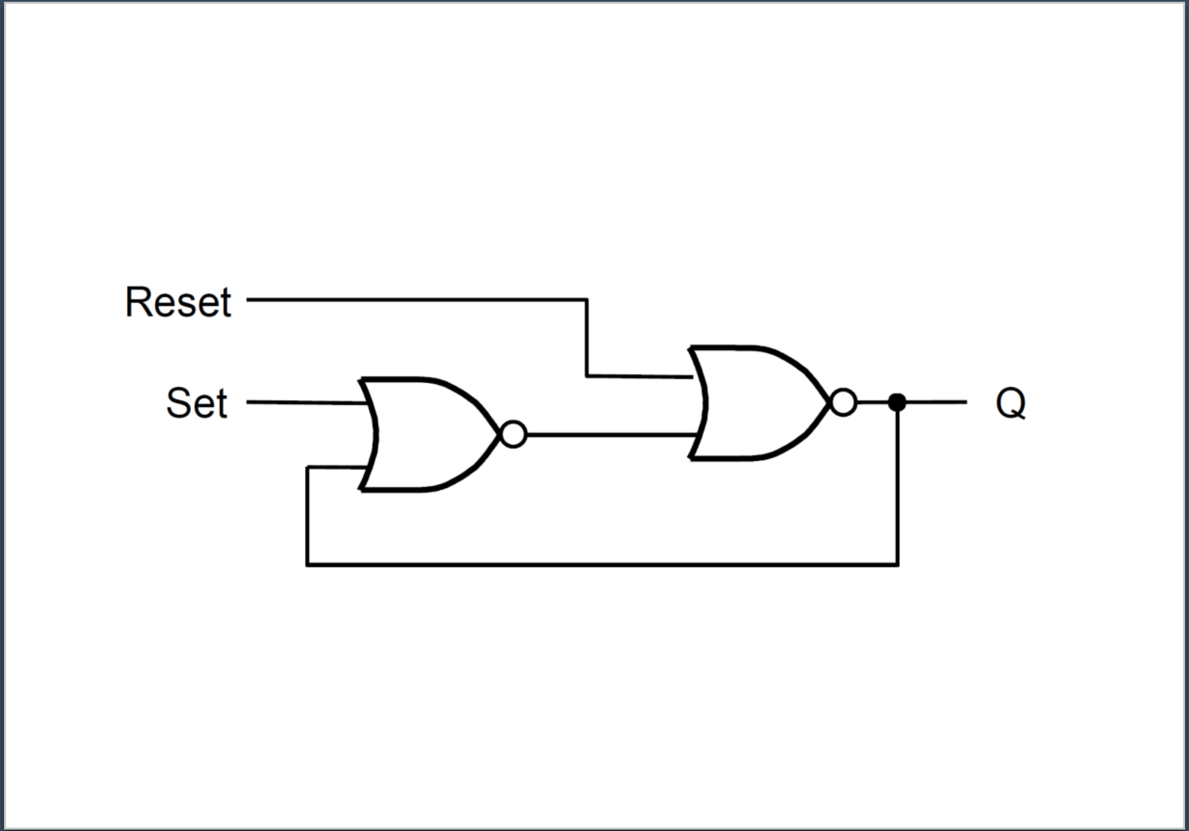
S-R Latch

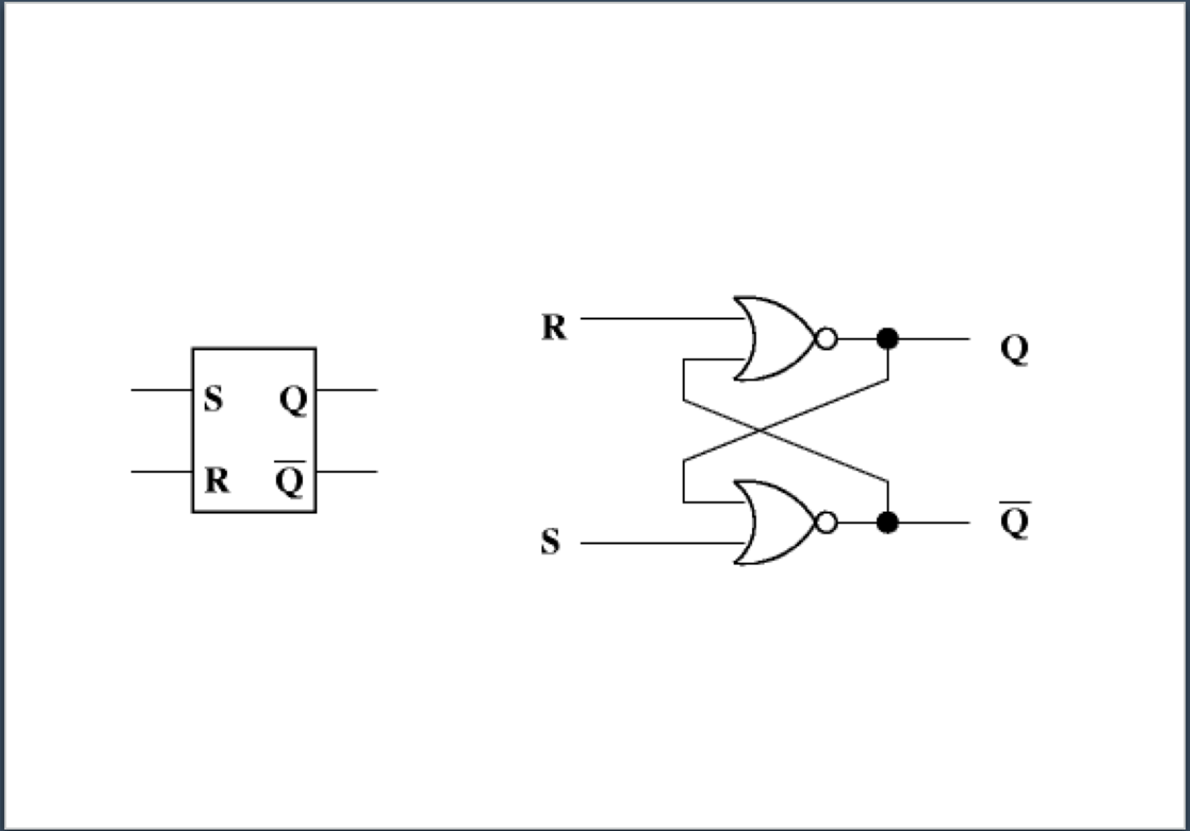
- Cross-coupled NOR/NAND **gates**
  - Data is stored at **Q** (inverse at **Q'**)
  - **S** and **R** are control inputs
  - S = Set, Q=1 -> S=1, Q=0; S=0
  - R = Reset

S-R Latch

Three inputs: Set, Reset, and a proxy clock

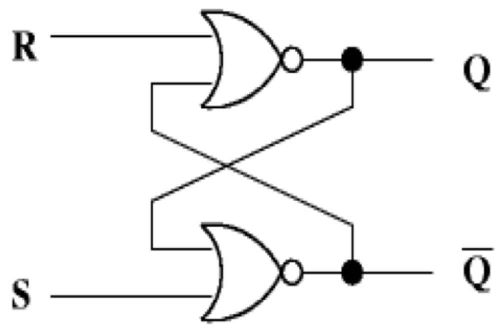
Circuit works only when the proxy clock is ON　　*S=Set*　*R=Reset*





Contd.

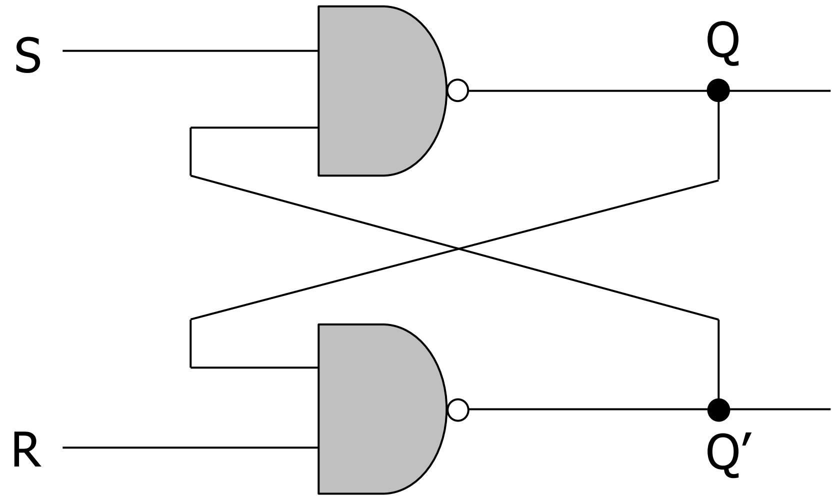
Given the current state and inputs to a latch, what is the next state. Typically, symbols Q, Q_{n-1}, Q^t, etc. are used to denote the current state, and correspondingly , Q^*, Q_n, Q^{t+1}, etc. denote the next state.



S	R	Q*
0	0	Q
0	1	0
1	0	1
1	1	*Undefined* (0 0)

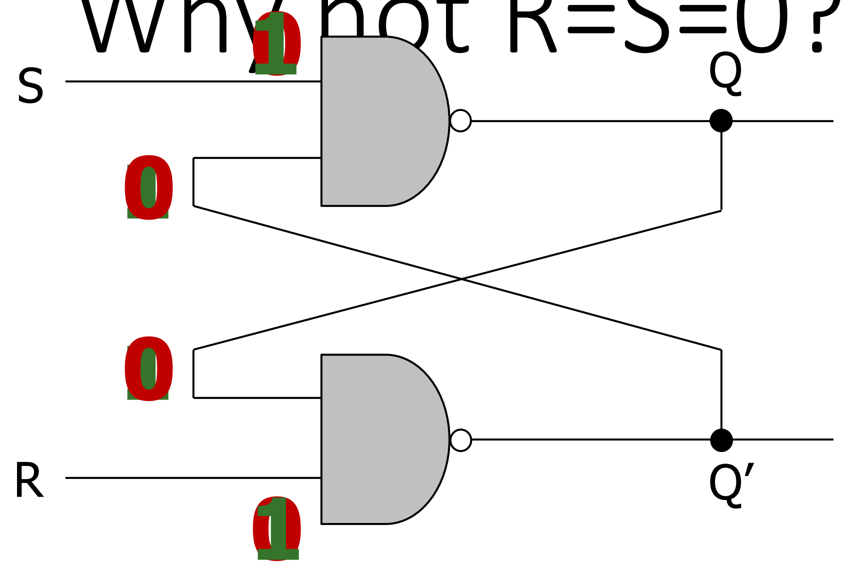
NAND gates

- Cross-coupled **NAND gates**
  - Data is stored at **Q** (inverse at **Q’**)
  - **S** and **R** are control inputs
    - In *quiescent* (*idle*) *state*, **both S and R are held at 1**
    - **S (set):** drive **S** to 0 (keeping **R** at 1) to change **Q** to 1
    - **R (reset):** drive **R** to 0 (keeping **S** at 1) to change **Q** to 0
- **S** and **R** should never **both** be 0 at the same time



Input		Output
R	S	Q
1	1	Q_{prev}
1	0	1
0	1	0
0	0	Forbidden

Why not R=S=0?

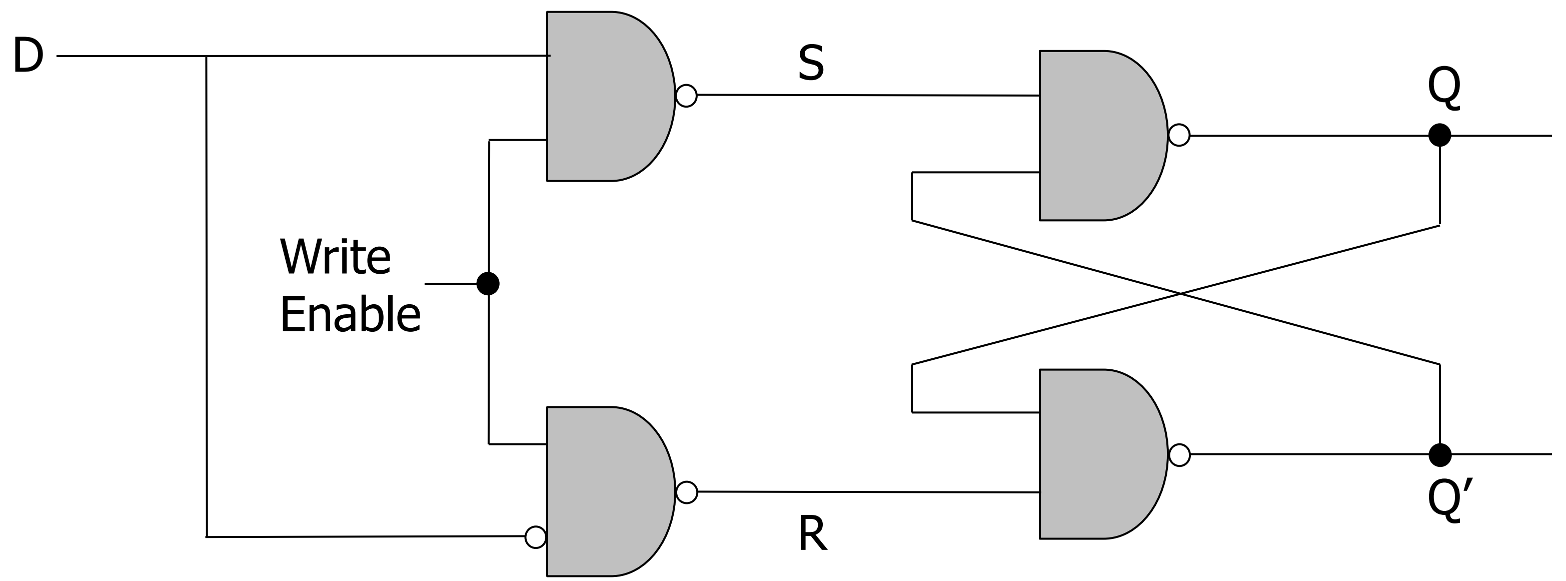


Input		Output
R	S	Q
1	1	Q_{prev}
1	0	1
0	1	0
0	0	Forbidden

1. If **R=S=0, Q** and **Q’** will both settle to 1, which **breaks** our invariant that **Q** = !**Q’**
2. If **S** and **R** transition back to 1 at the same time, **Q** and **Q’** begin to oscillate between 1 and 0 because their final values depend on each other (**metastability**)
   - This eventually settles depending on **variation in the circuits**

Gated D-latch

- How do we **guarantee** correct operation of an S-R Latch?
  - Add two more NAND gates!



Input		Output
WE	D	Q
0	0	Q_{prev}
0	1	Q_{prev}
1	0	0
1	1	1

- **Q** takes the value of **D**, when **write enable (WE)** is set to 1
- **S** and **R** can never be 0 at the same time!

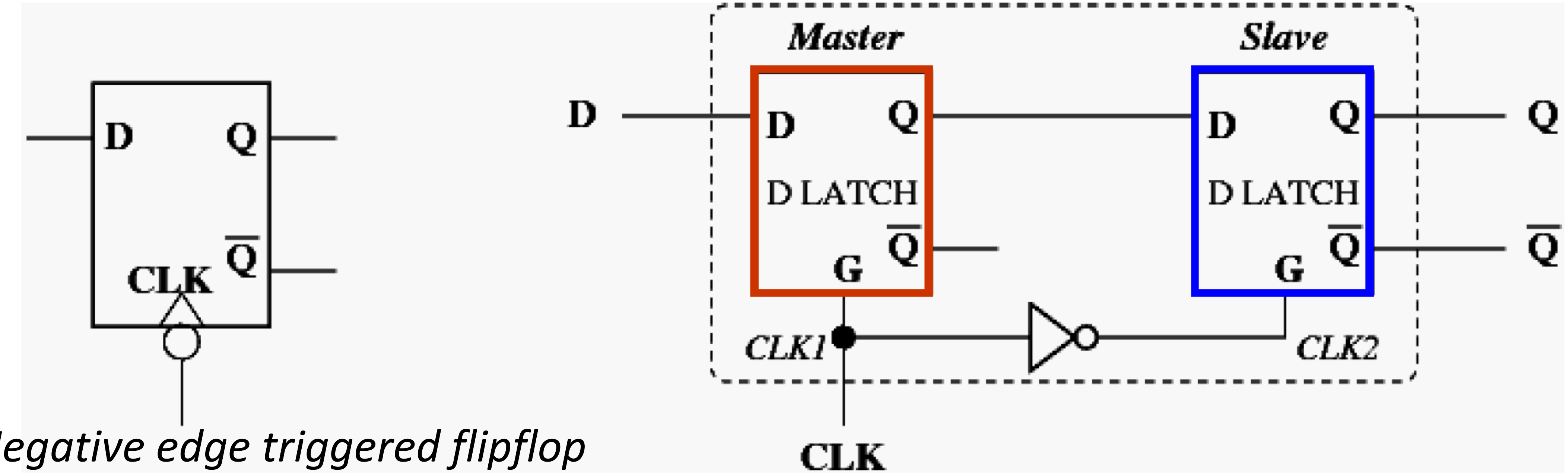
Why are latches not preferred? Coffee points++

The inputs should not change while the gate signal is asserted (otherwise there are multiple state changes which can lead to problems in a circuit).

One Solution

- What if we change our states only on clock edge and call me edge-triggered

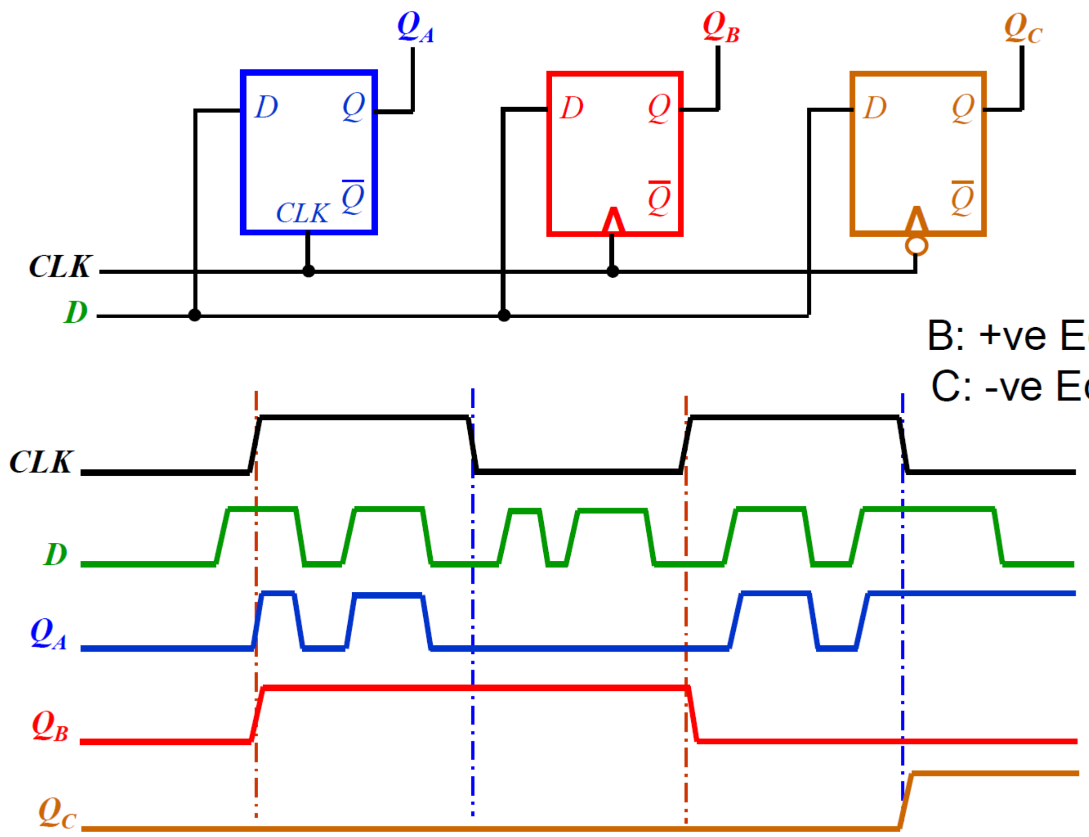
Edge Triggering Master-slave D Flip-flop with two latches



Negative edge triggered flipflop

At a given time, only one latch is alive (either master or slave)

Level/Edge Triggered



A: Latch
B: +ve Edge trigerred DFF
C: -ve Edge trigerred DFF

Summary

Gates are building blocks of combinational circuits

Latches are

……………………………

sequential circuits

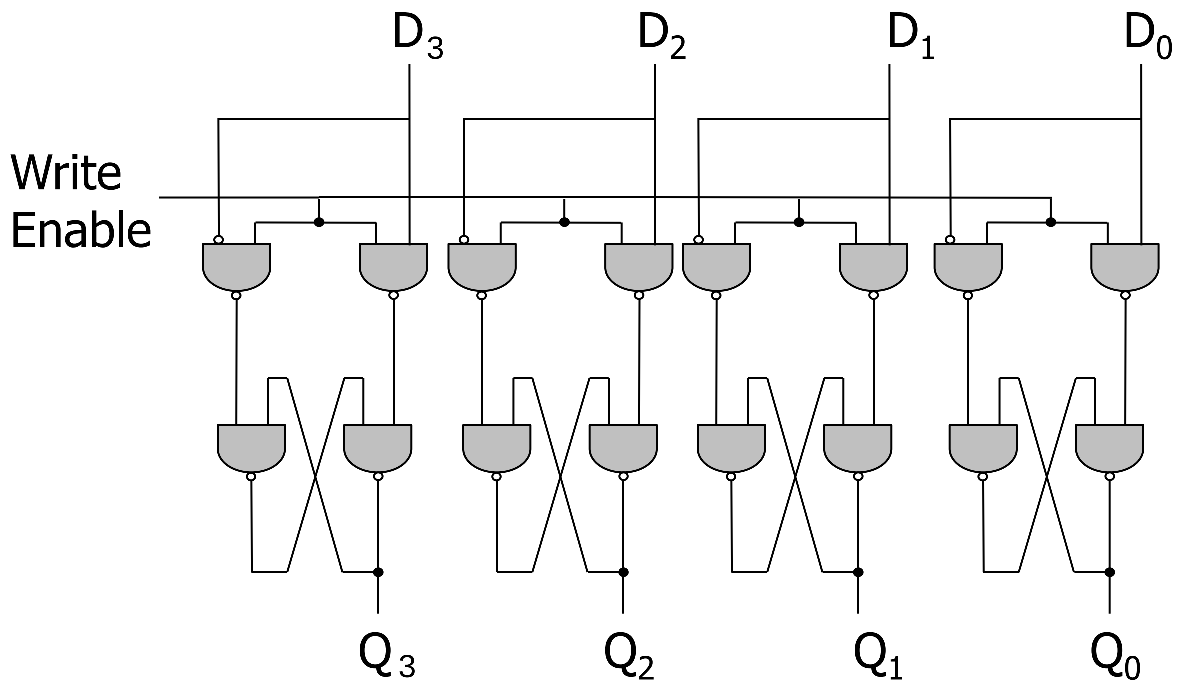
Latches are built from gates

Flip-flops are built from latches

Register

How can we use D latches to store **more** data?

- Use **more** D latches!
- A single WE signal for all latches for simultaneous writes



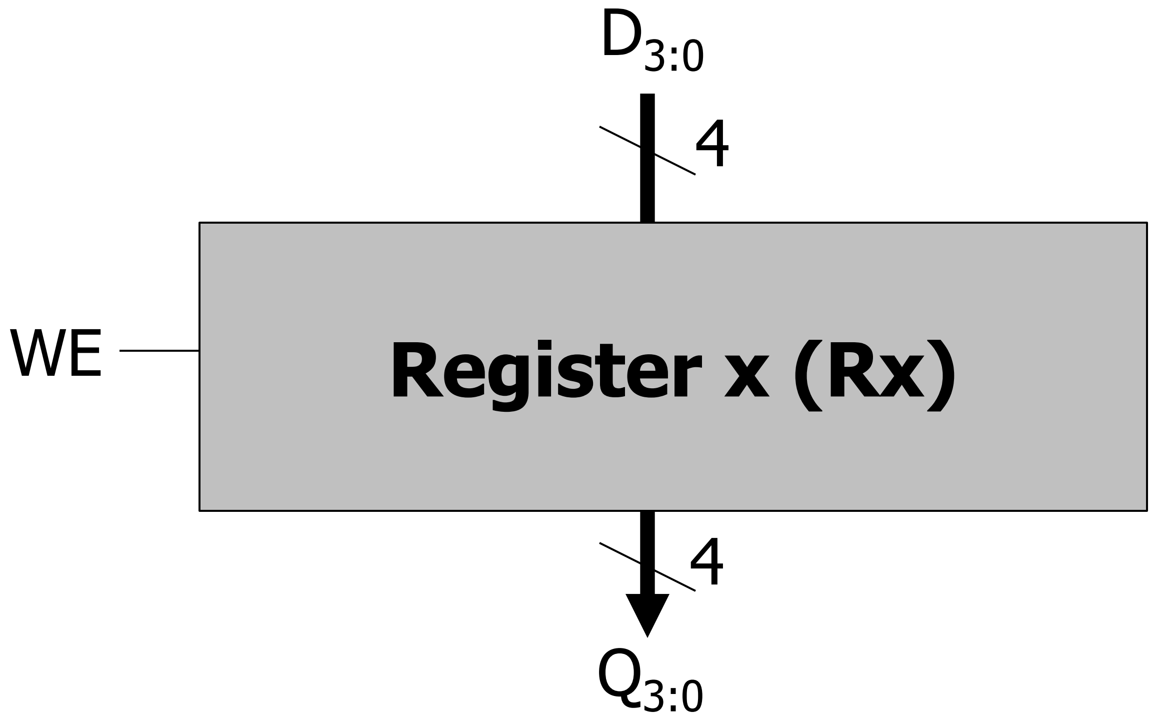
Here we have a **register,** or a structure that stores more than one bit and can be read from and written to

This **register** holds 4 bits, and its data is referenced as Q[3:0]

Register

How can we use D latches to store **more** data?

- Use **more** D latches!
- A single WE signal for all latches for simultaneous writes



Here we have a **register,** or a structure that stores more than one bit and can be read from and written to

This **register** holds 4 bits, and its data is referenced as Q[3:0]

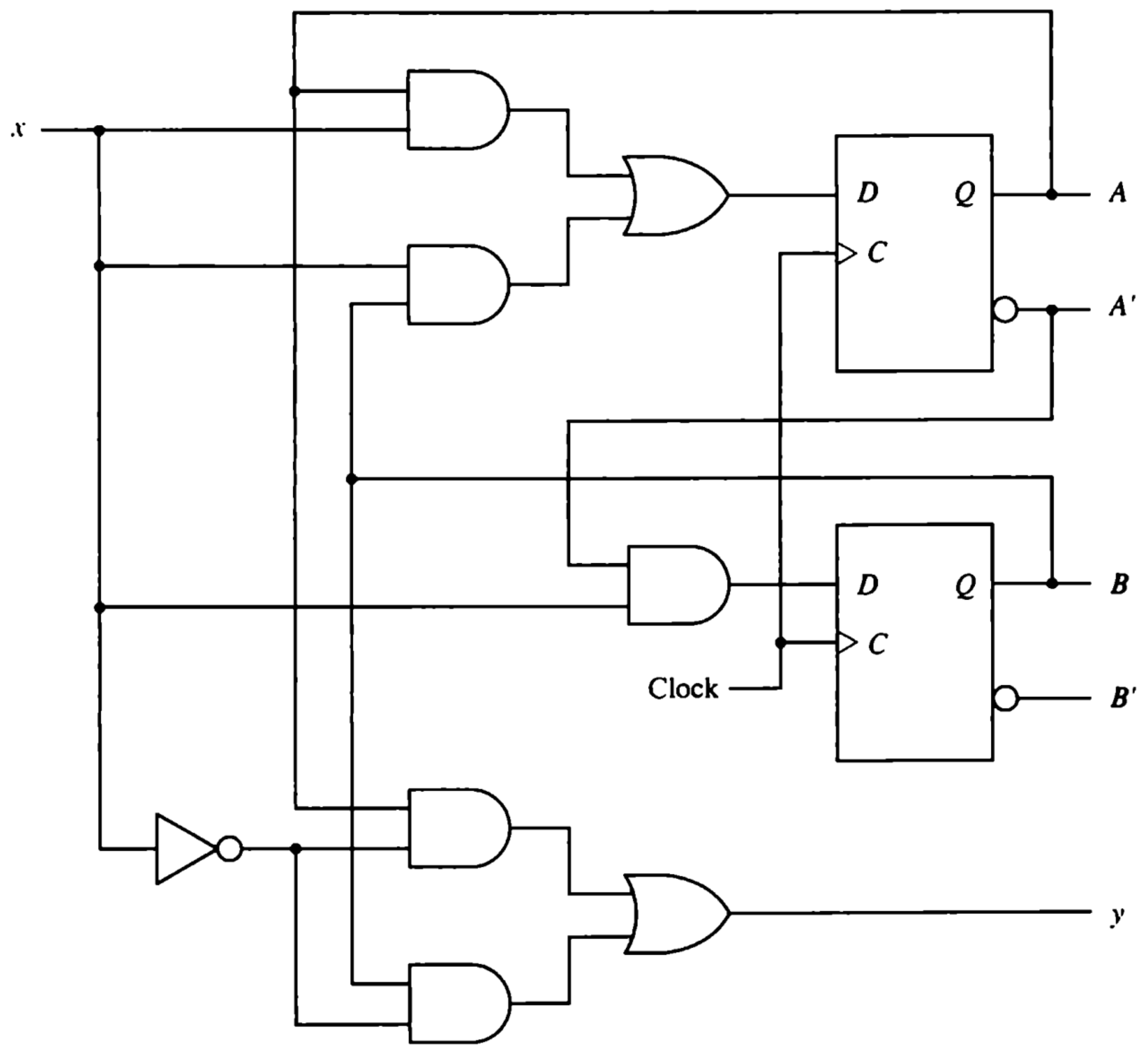
Sequential circuit 101 again

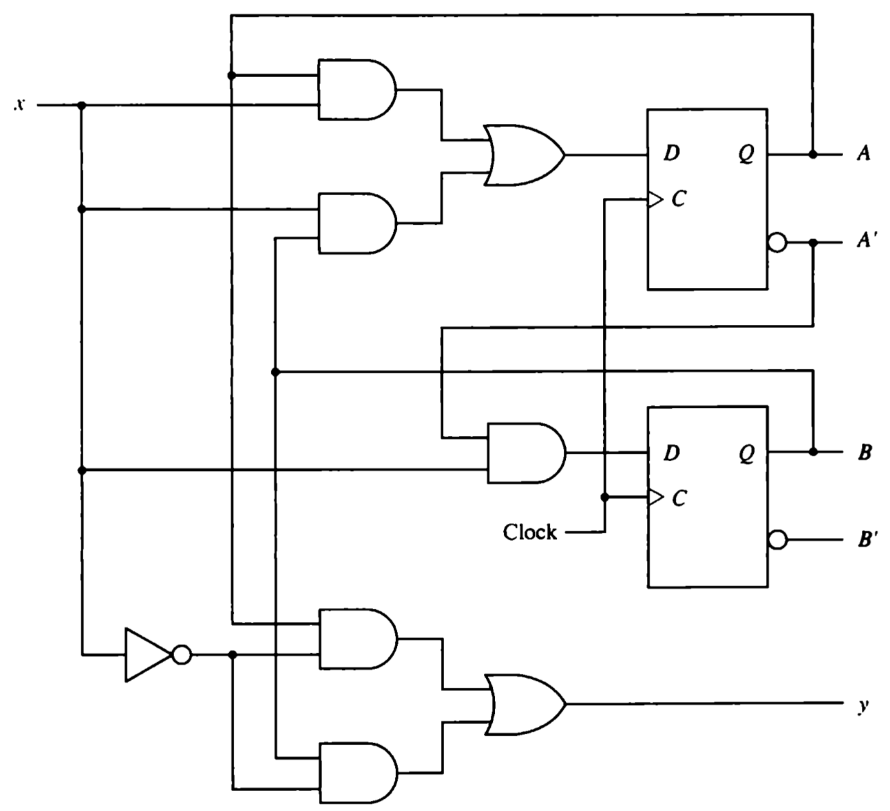
Combination of latches/flip-flops and combinational elements (gates).

Flip-flops need a clock

An Example

(Let’s try it)





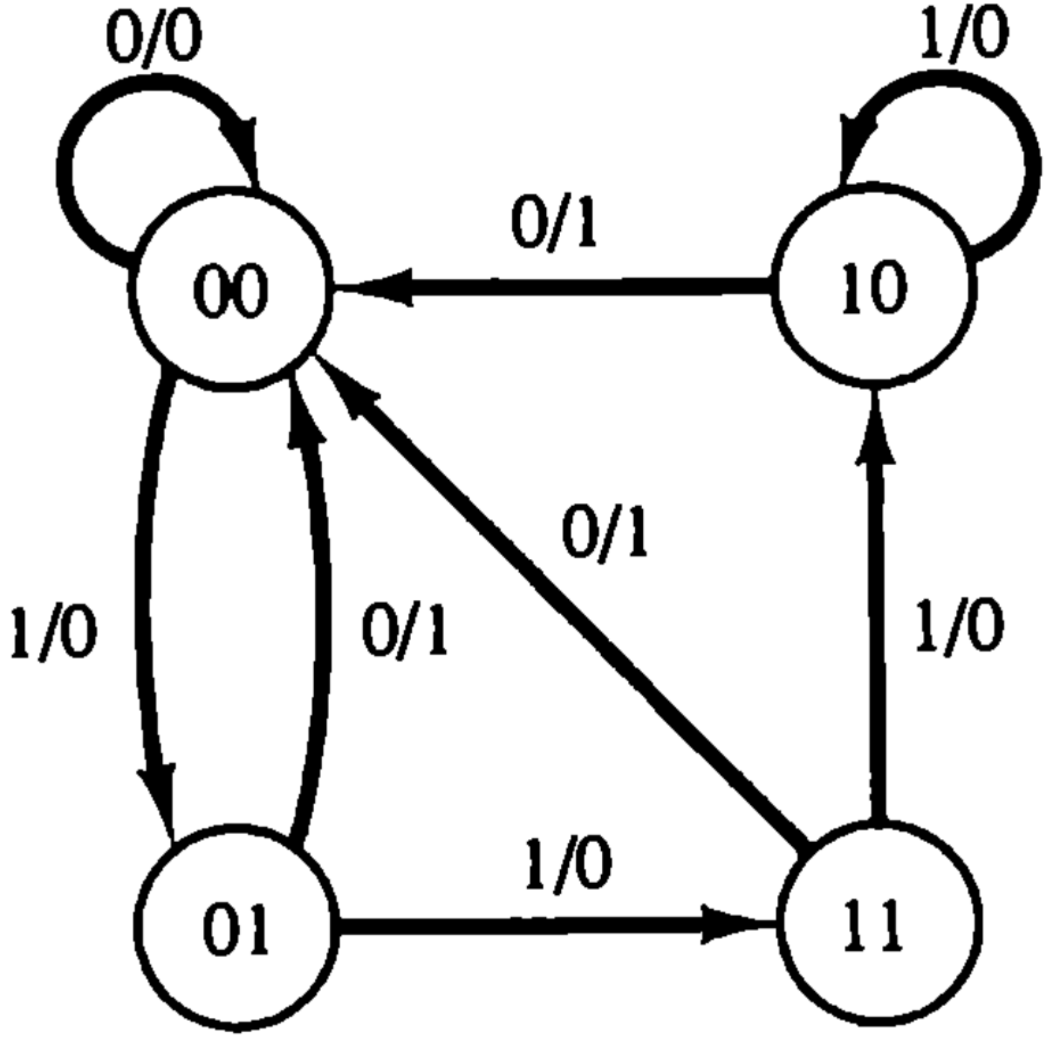
Present state		Input	Next state		Output
A	*B*	*x*	*A*	*B*	*y*
0	0	0	0	0	0
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	1	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

A (t+1) = Ax + Bx, B (t+1) = A\`x ,  y = Ax\` + Bx\`

State Table

State Diagram

x/y where x is input and y is ouput after the transition



Try it for a binary counter ☺

Lab-1 coming up

World of State machines (FSMs) Moore and Mealy Machines

Computer Architecture

Moore vs Mealy

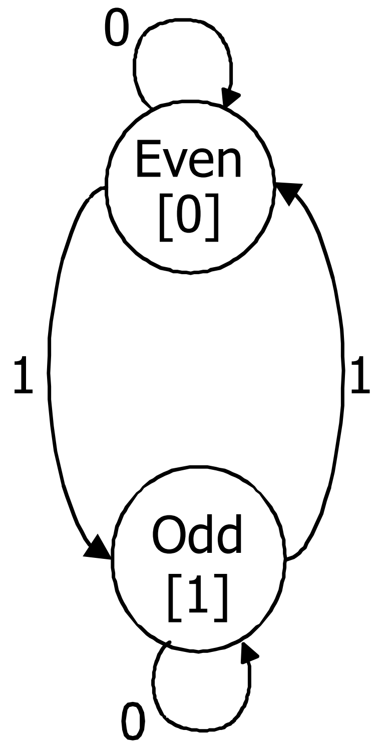
Moore machine: Output depends on the current state

Mealy machine: Output depends on the current state and inputs

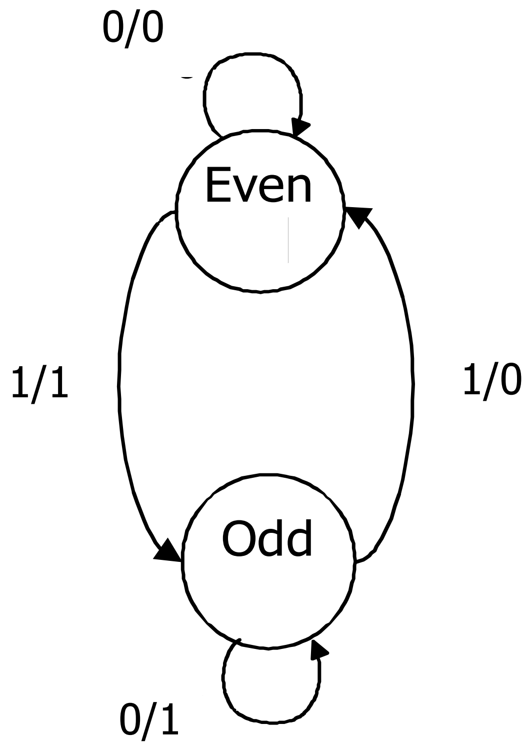
Odd Parity Checker

- Serial input string
  - OUT=1 if odd # of 1s in input
  - OUT=0 if even # of 1s in input
- Let's do this for Moore and Mealy

Moore



Mealy



State Transitions

Output changes only when the state changes

Appears after the state transition takes place

outputs change at clock edge

Even = 0

Odd = 1

Moore

Present State	Input	Next State	Present Output
Even	0	Even	0
Even	1	Odd	0
Odd	0	Odd	1
Odd	1	Even	1

Output changes when the state and input changes

Appears before the state transition is completed

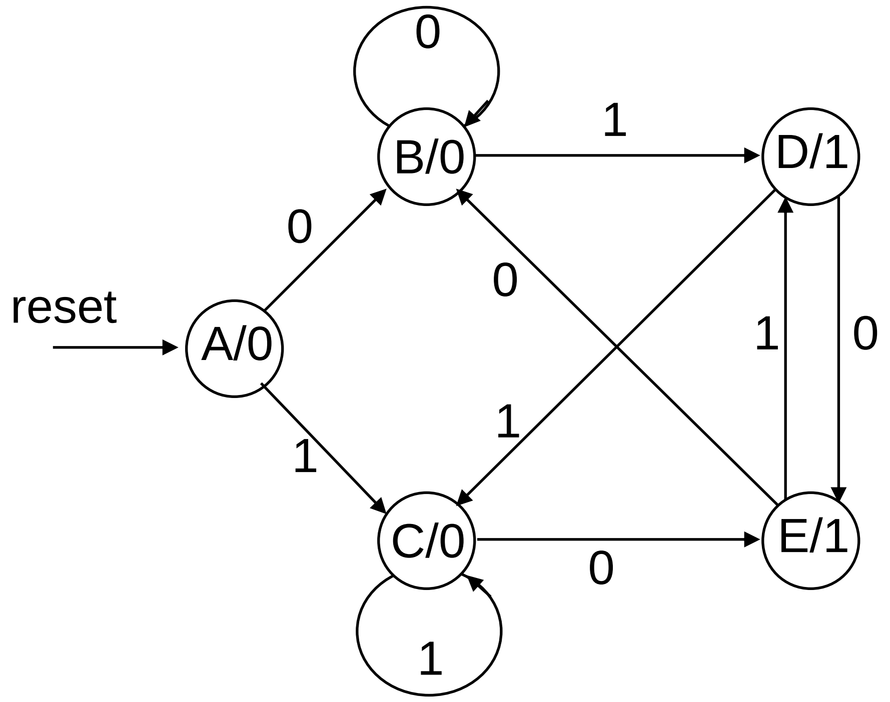
React faster to inputs — don't wait for clock

Mealy

Present State	Input	Next State	Present Output
Even	0	Even	0
Even	1	Odd	1
Odd	0	Odd	1
Odd	1	Even	0

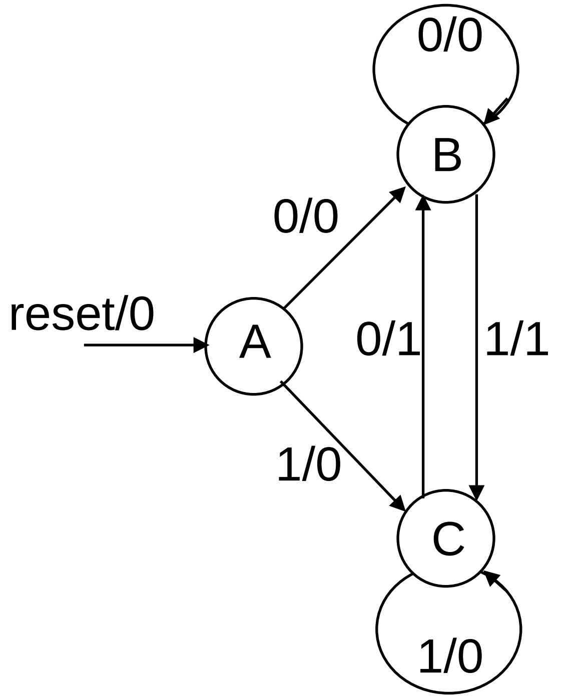
Try on your own

01/10 detector: Moore Machine



reset	input	current state	next state	current output
1	–	–	A	0
0	0	A	B	0
0	1	A	C	0
0	0	B	B	0
0	1	B	D	0
0	0	C	E	0
0	1	C	C	0
0	0	D	E	1
0	1	D	C	1
0	0	E	B	1
0	1	E	D	1

01/10 detector: Mealy Machine



reset	input	current state	next state	current output
1	–	–	A	0
0	0	A	B	0
0	1	A	C	0
0	0	B	B	0
0	1	B	C	1
0	0	C	B	1
0	1	C	C	0

Textbook Reading

H&H, 3.2 and 3.4

Coffee Credits

Anshika 210050014

இந்த நாள் இனிய நாளாகட்டும்