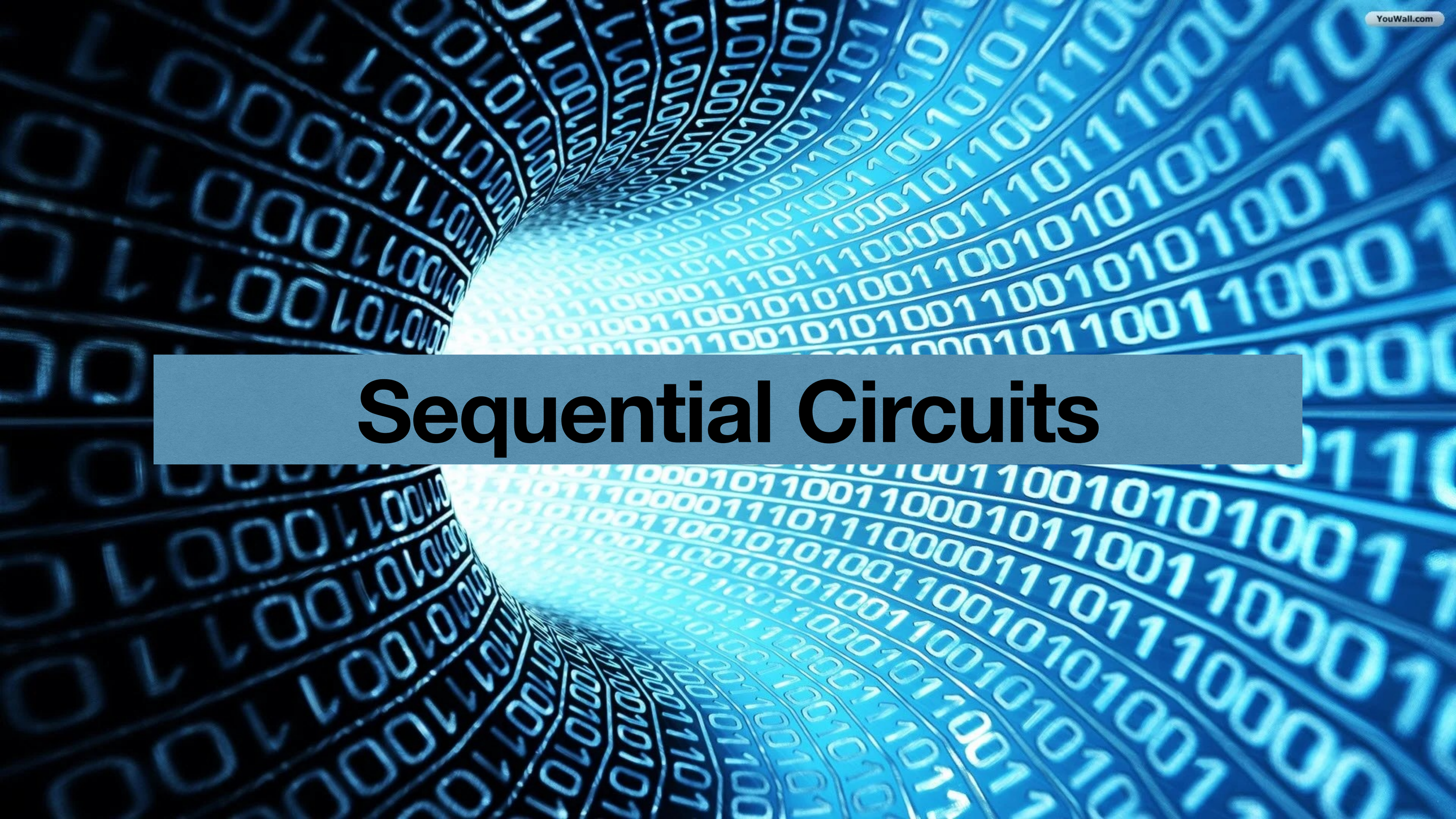
Digital Logic Design + Computer Architecture

Sayandeep Saha

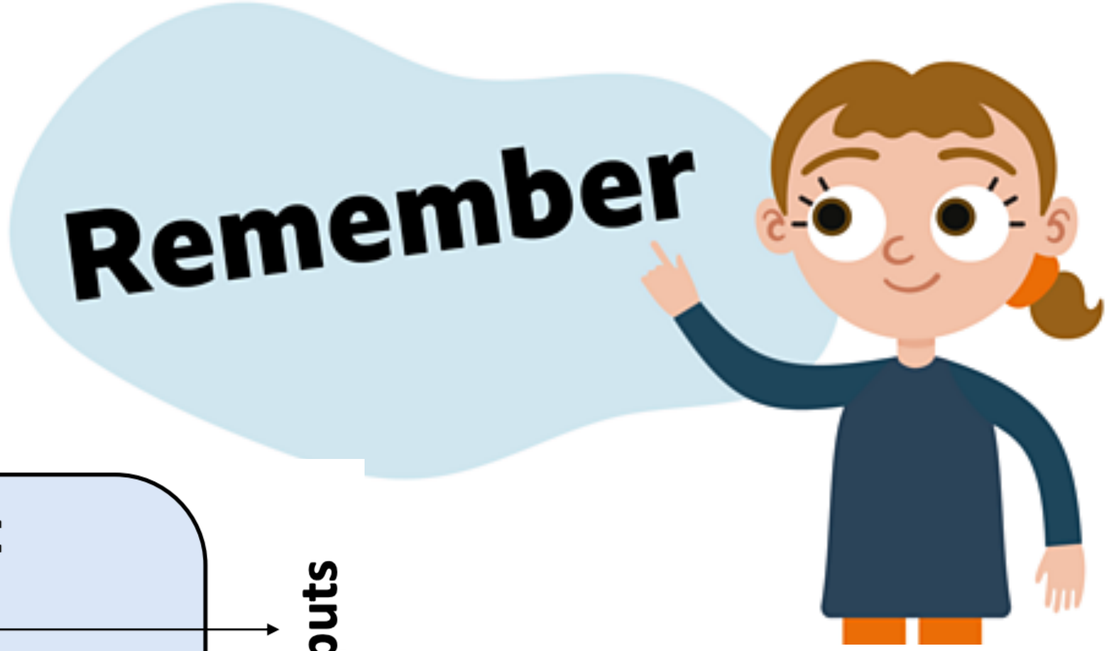
Assistant Professor
Department of Computer Science and Engineering
Indian Institute of Technology Bombay

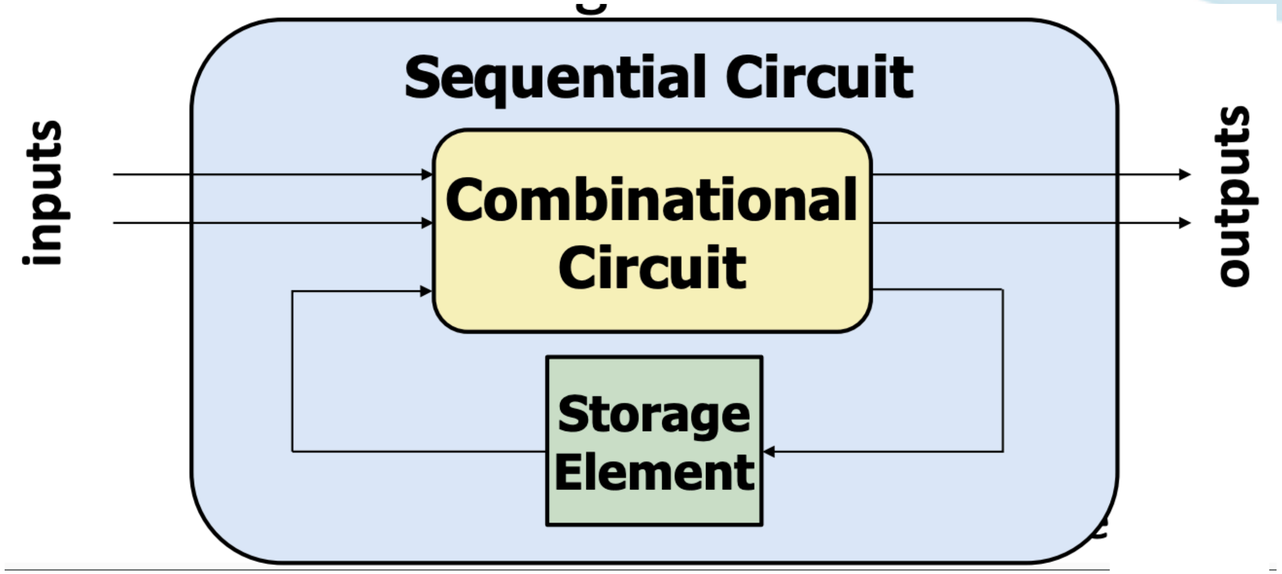


Sequential Circuits

A Circuit that Remembers

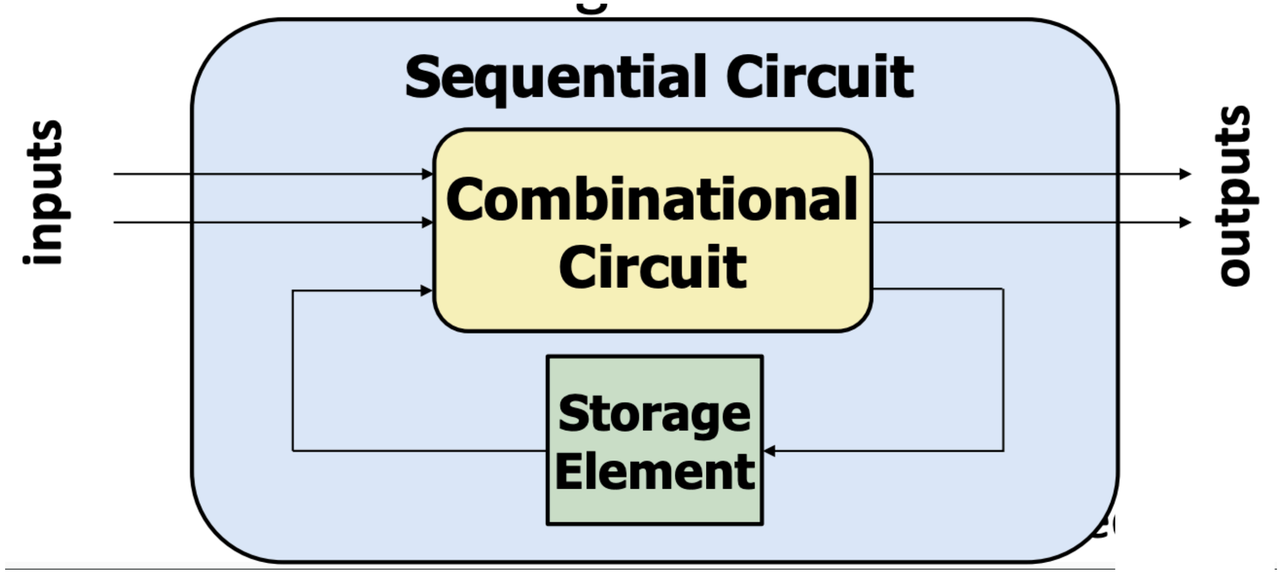
- **How do you remember things?**
  - Memory
- **Can we design a circuit which remembers?**
  - A formal way to model this capability is called a **state**
  - **So we will be modelling circuits to create a state.**



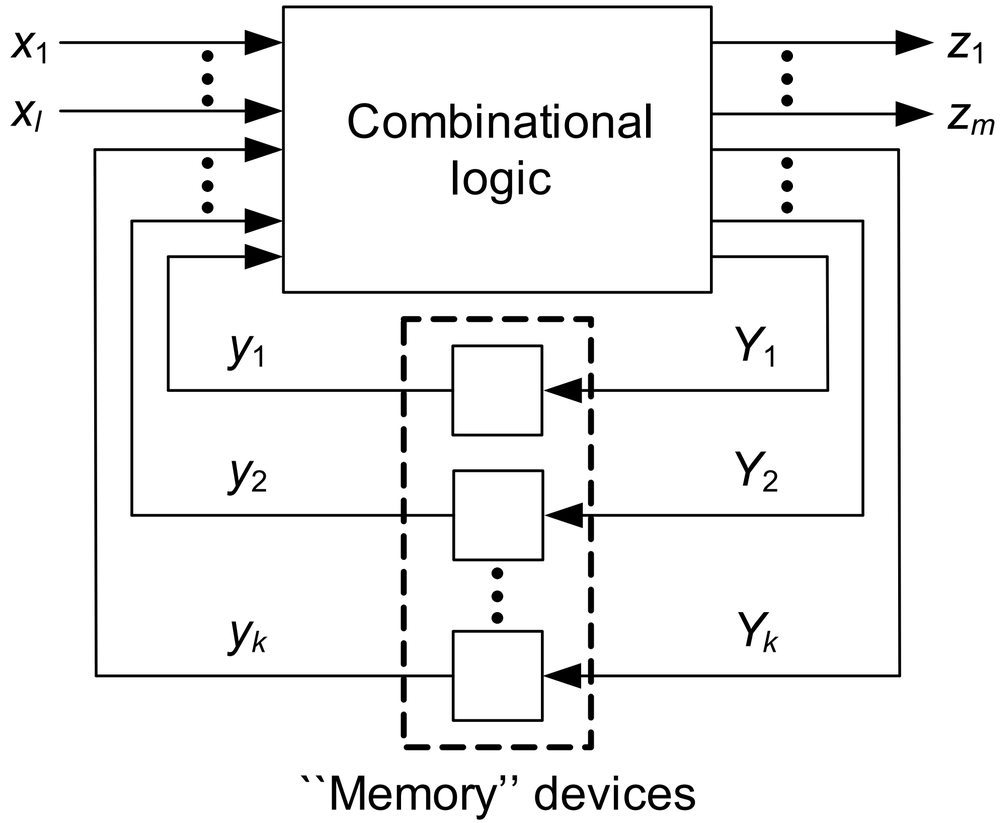


A Circuit that Remembers

- **Every digital logic you see in real life is sequential**
  - Your processors — that you going to see in the rest of the course
  - Your washing machine — it remembers your setting and washes accordingly
  - Your elevator — it remembers which floors to stop
  - Your ATM machine — it remembers your choice and updates your account after despatching money



Sequential Circuits



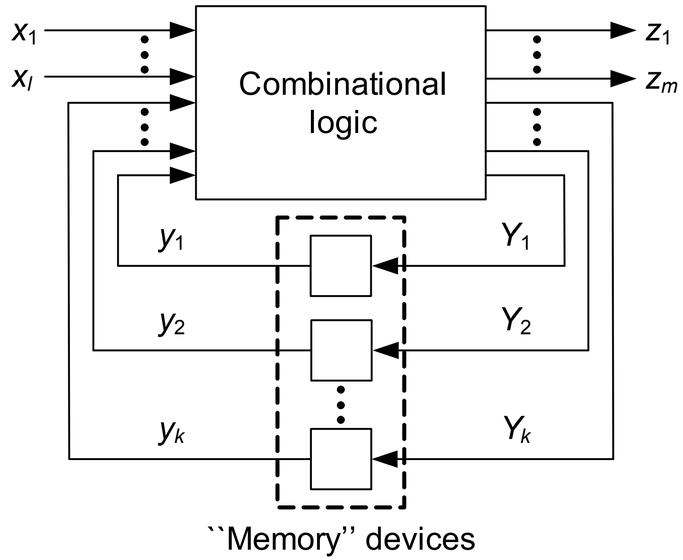
Sequential Circuits

To generate the Y's: memory devices must be supplied with appropriate input values

- **Characteristic table/functions**: switching functions that describe the impact of x_i's and y_j's on the memory-element input
- **Excitation table**: its entries are the values of the memory-element inputs

Most widely used memory elements: **flip-flops**, which are made of **latches**

- **Latch**: remains in one state indefinitely until an input signals directs it to do otherwise

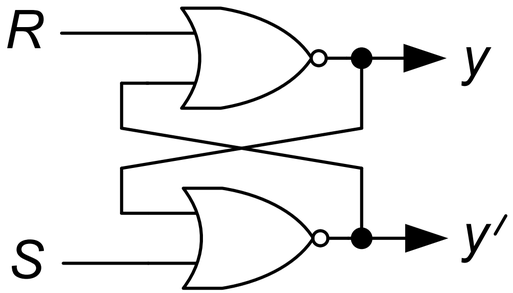


Memory Element: Latches

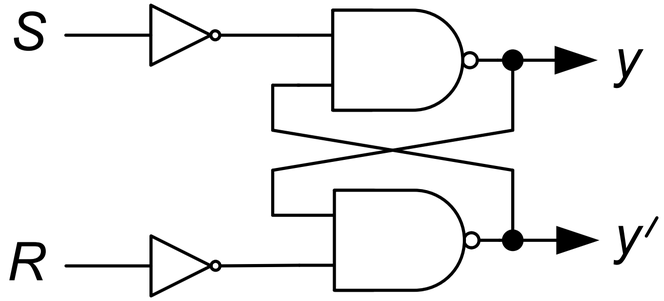
Latch: remains in one state indefinitely until an input signals directs it to do otherwise

Set-reset of *SR* latch:

(*a*) Block diagram.



(*b*) NOR latch.



(*c*) NAND latch.

Memory Element: Latches

Characteristic table and excitation requirements:

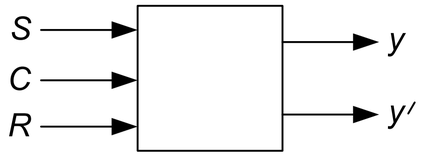
$y(t)$	$S(t)$	$R(t)$	$y(t+1)$
0	0	0	0
0	0	1	0
0	1	1	?
0	1	0	1
1	1	0	1
1	1	1	?
1	0	1	0
1	0	0	1

Circuit change		*Required value*	
From:	*To:*	S	R
0	0	0	–
0	1	1	0
1	0	0	1
1	1	–	0

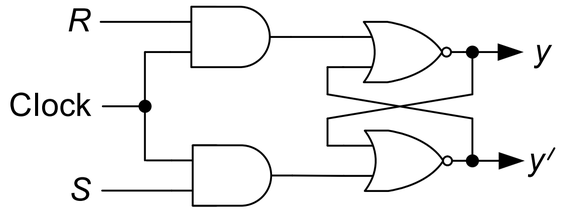
$$RS = 0$$
$$y(t+1) = R'y(t) + S$$

Clocked *SR* latch: all state changes synchronized to clock pulses

- Restrictions placed on the length and frequency of clock pulses: so that the circuit changes state no more than once for each clock pulse



(*a*) Block diagram.



(*b*) Logic diagram.

Memory Element: Latches

Why is the (1,1) input forbidden?

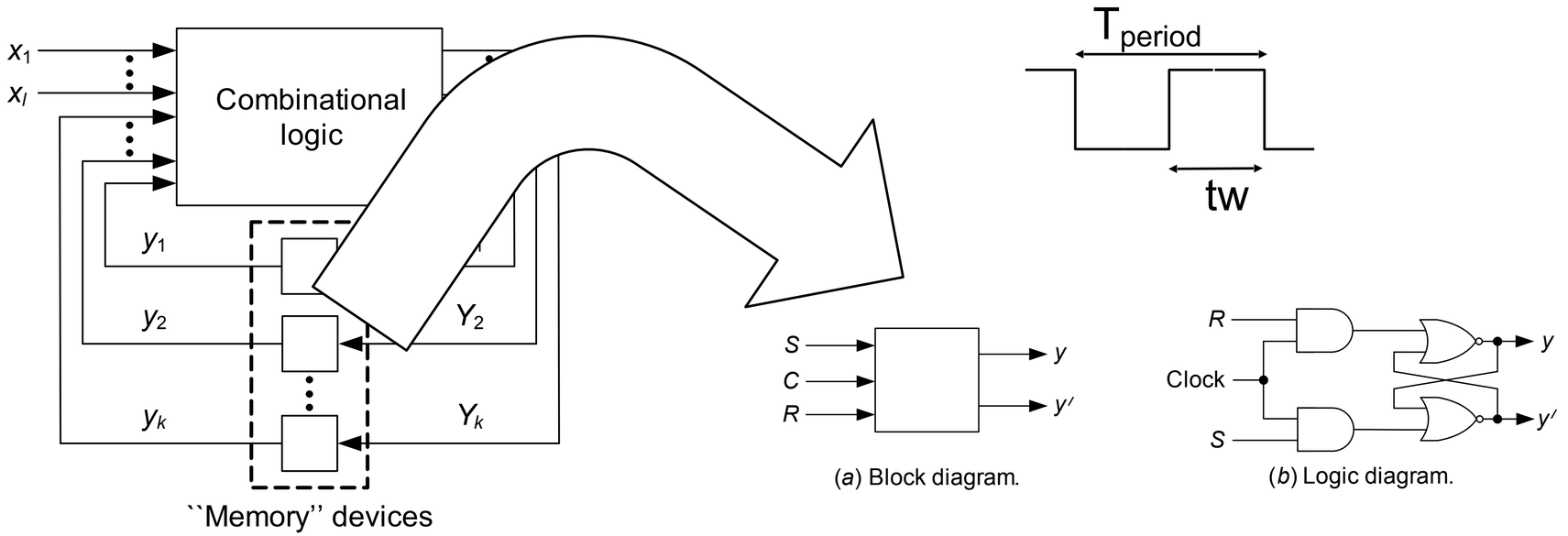
$y(t)$	$S(t)$	$R(t)$	$y(t+1)$
0	0	0	0
0	0	1	0
0	1	1	?
0	1	0	1
1	1	0	1
1	1	1	?
1	0	1	0
1	0	0	1

$$RS = 0$$

$$y(t+1) = R'y(t) + S$$

1. If **R=S=1, Q** and **Q'** will both settle to 1, which **breaks** our invariant that **Q** = !**Q'**
2. If **S** and **R** transition back to 0 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**

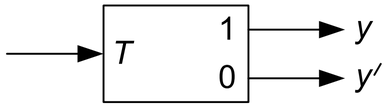
Memory Element: Latches



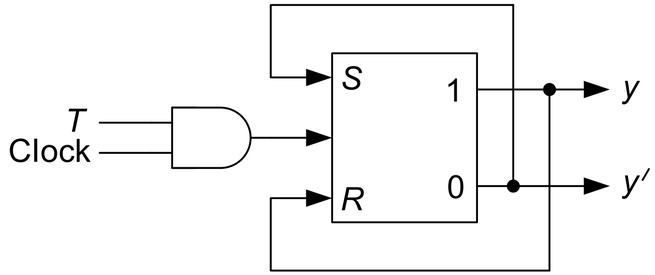
- A **clock** is a periodic signal that is used to keep time in sequential circuits.
- **Duty Cycle** is the ration of t_w/T_{period}
- We want to keep t_w small so that in the same clock pulse only a single computation is performed.
- We want to keep T_{period} sufficient so that there is enough time for the next input to be computed.

Memory Element: T Latch

Value 1 applied to its input triggers the latch to change state



(*a*) Block diagram.



(*b*) Deriving the T latch from the clocked SR latch.

Excitations requirements:

Circuit change		*Required*
From:	*To:*	*value T*
0	0	0
0	1	1
1	0	1
1	1	0

"Q" is basically "y"

Characteristic Table

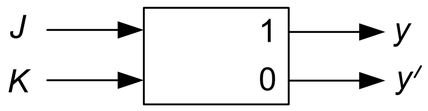
T Flip-Flop

T	**Q(t + 1)**	
0	$Q(t)$	No change
1	$Q'(t)$	Complement

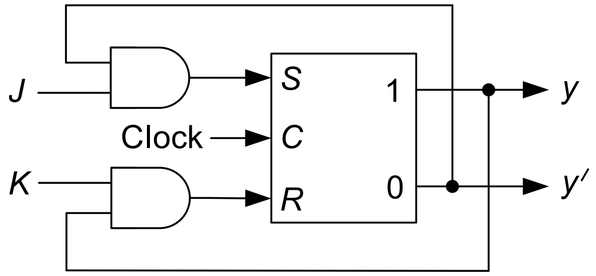
$$y(t+1) = Ty'(t) + T'y(t)$$
$$= T \oplus y(t)$$

Memory Element: JK Latch

Unlike the *SR* latch, $J = K = 1$ is permitted: when it occurs, the latch acts like a trigger and switches to the complement state



(*a*) Block diagram.



(*b*) Constructing the JK latch from the clocked SR latch.

Excitation requirements:

Circuit change		*Required value*	
From:	*To:*	*J*	*K*
0	0	0	–
0	1	1	–
1	0	–	1
1	1	–	0

"Q" is basically "y"

Can you write the characteristic equation?

Characteristic Table

***JK* Flip-Flop**			
J	***K***	**$Q(t+1)$**	
0	0	$Q(t)$	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q'(t)$	Complement

$$y(t+1) = Jy(t)' + K'y(t)$$

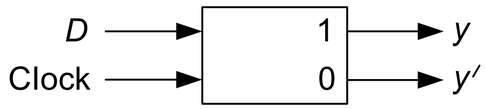
D Latch — The Latch of Your Life

The next state of the D latch is equal to its present excitation:

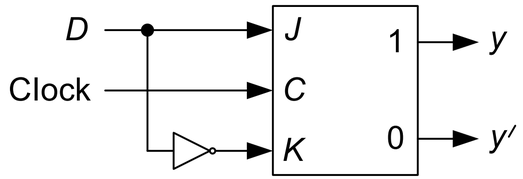
$$y(t+1) = D(t)$$

D Flip-Flop

D	Q(t + 1)	
0	0	Reset
1	1	Set



(*a*) Block diagram.



(*b*) Transforming the JK latch to the D latch.

Excitation Table

Q(t)	Q(t+1)	D
0	0	0
0	1	1
1	0	0
1	1	1

How is Your Clock?

Clocked latch: changes state only in synchronization with the clock pulse and no more than once during each occurrence of the clock pulse

Duration of clock pulse: determined by circuit delays and signal propagation time through the latches

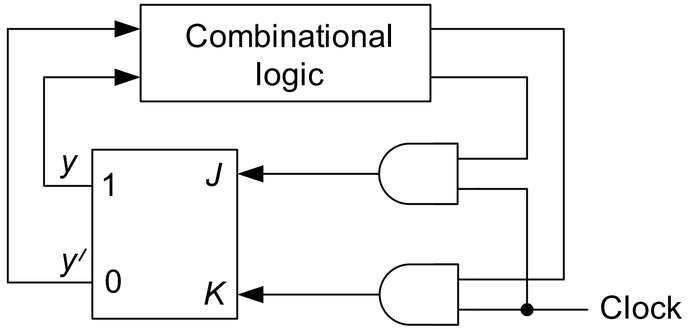
- Must be long enough to allow latch to change state, and
- Short enough so that the latch will not change state twice due to the same excitation

Excitation of a *JK* latch within a sequential circuit:

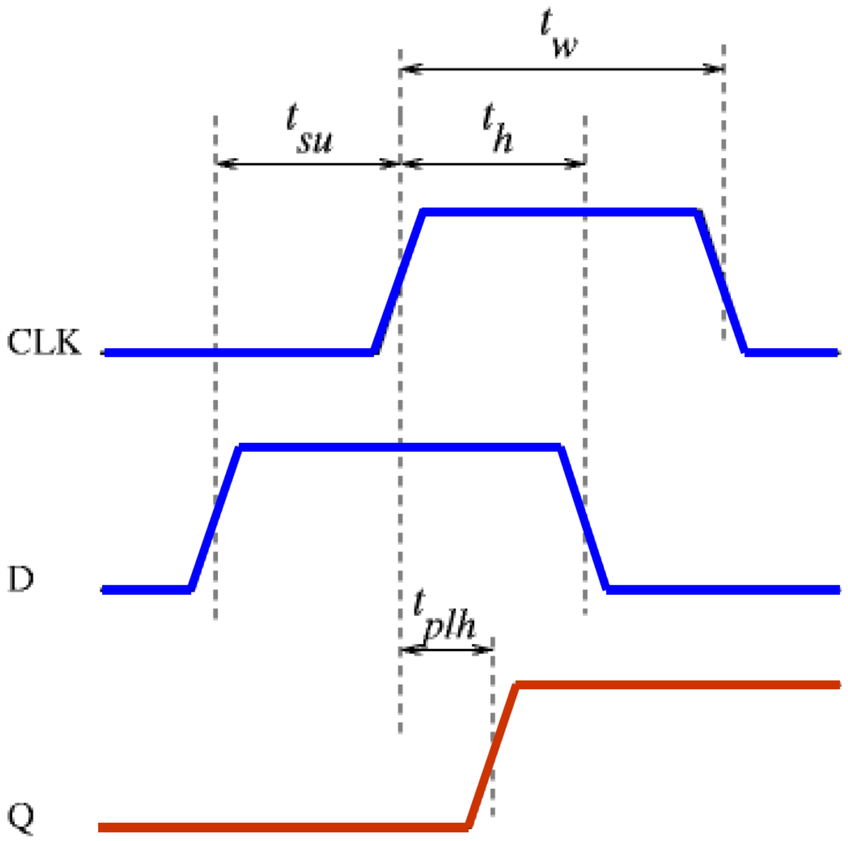
- Length of the clock pulse must allow the latch to generate the *y*'s
- But should not be present when the values of the *y*'s have propagated through the combinational circuit

How fast/slow should be the clock really?

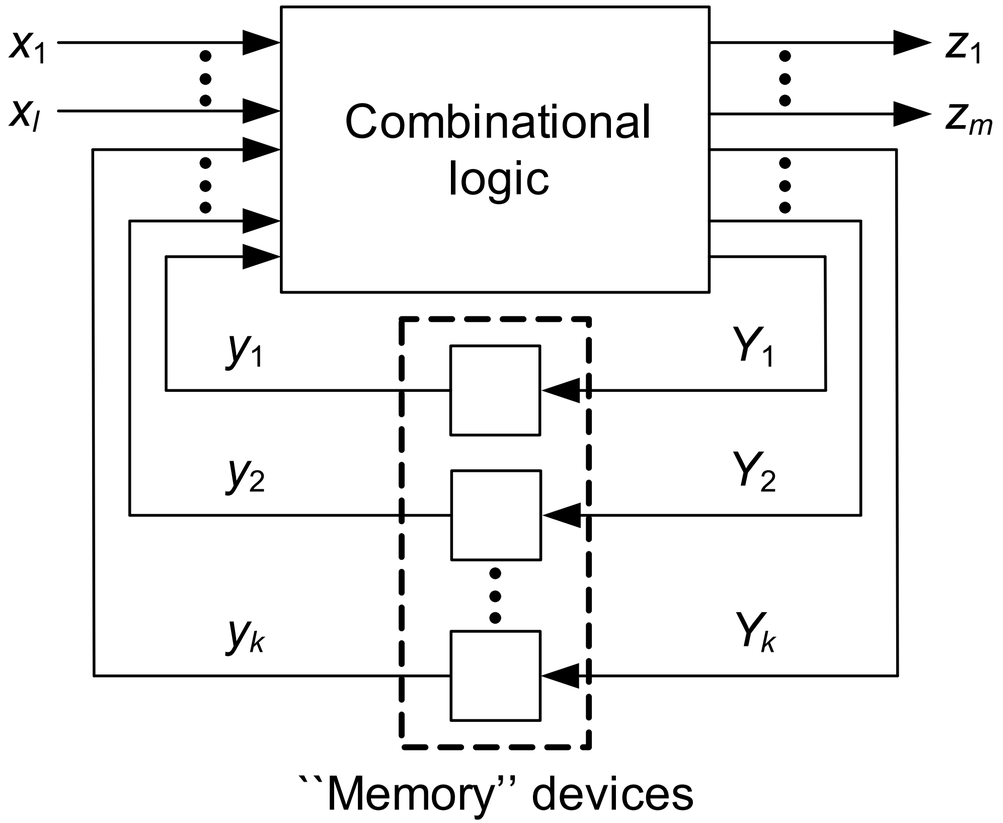
But when does the flip-flop changes its state???



All in One



D Flip–flop (edge–triggered)
(positive edge triggering)



``Memory'' devices

Delay to make sure all is well

- **Setup time**, t_{su}, is the time period prior to the clock becoming active (edge or level) during which the flip-flop inputs must remain stable.

- **Hold time**, t_h, is the time after the clock becomes inactive during which the flip-flop inputs must remain stable.

- Setup time and hold time define a *window of time during which the flip-flop inputs cannot change* – quiescent interval.

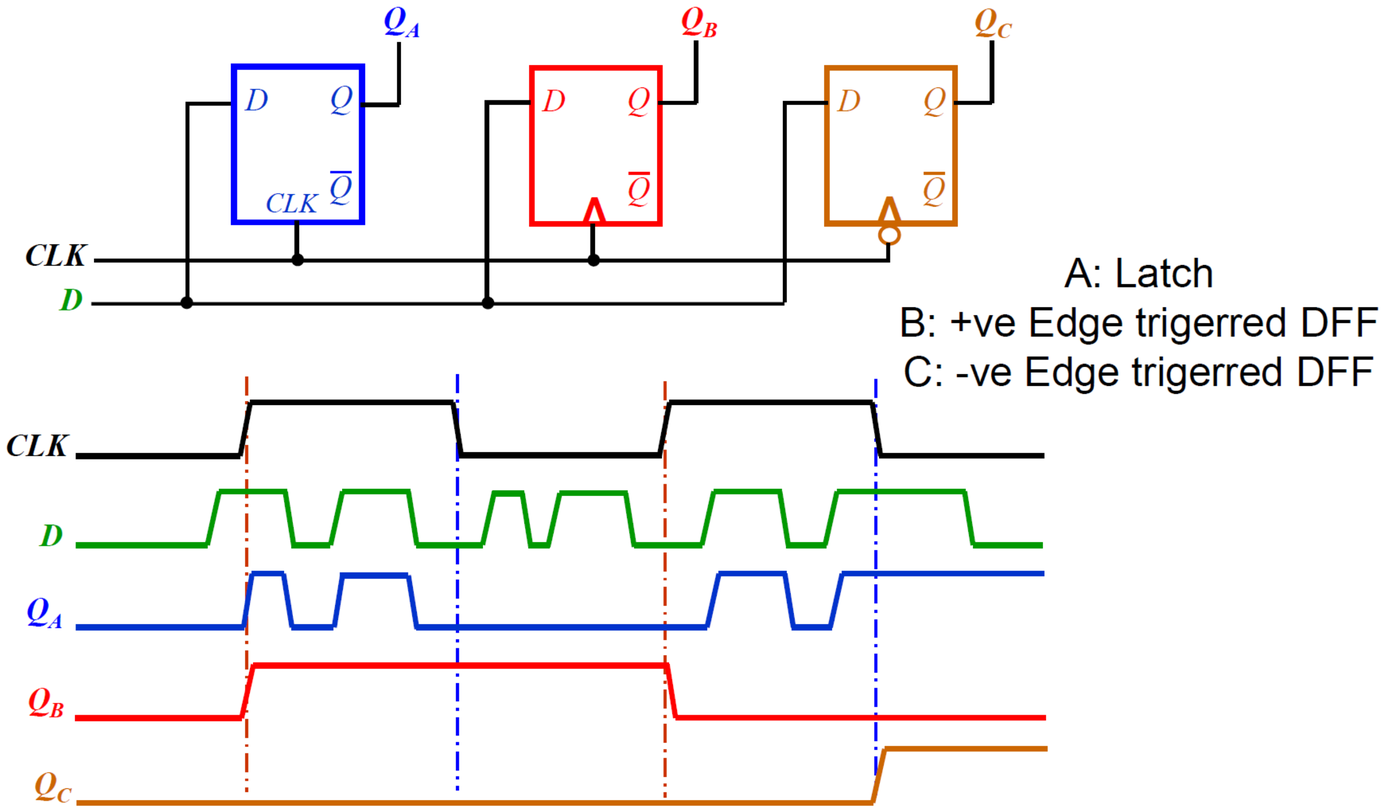
More Delay

- **Propagation delay,** t_{pHL} and t_{pLH} , has the same meaning as in combinational circuit – beware propagation delays usually will not be equal for all input to output pairs. There can be two propagation delays: $t_{C\text{-}Q}$ (*clock*→*Q* delay) and $t_{D\text{-}Q}$ (*data*→*Q* delay).

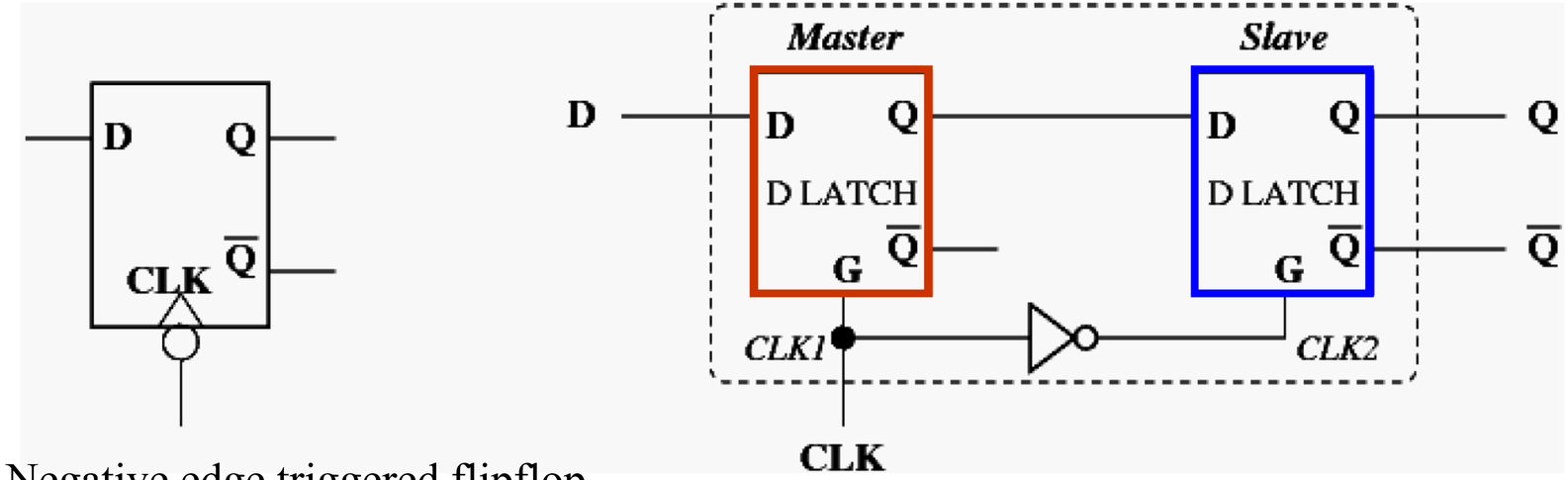
- For a level or pulse triggered latch:
  - Data input should remain stable till the clock becomes inactive.
  - Clock should remain active till the input change is propagated to Q output. That is, active period of the clock,

$$t_w > \max \{t_{pLH}, t_{pHL}\}$$

The Triggering Dilemma



Master Slave Flip-Flop



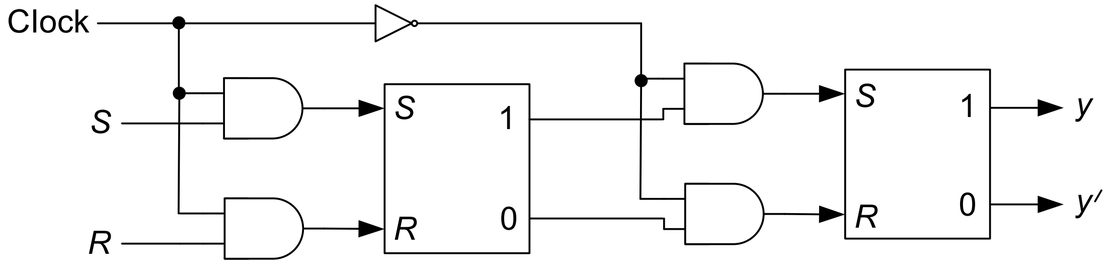
Negative edge triggered flipflop

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

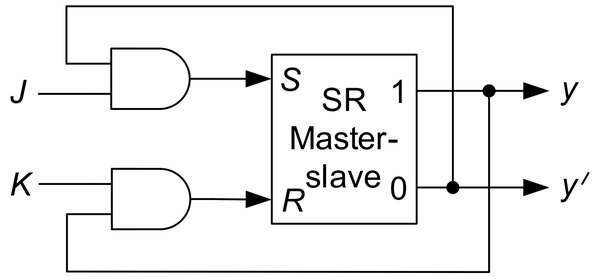
Master Slave Flip-Flop

Master-slave flip-flop: a type of synchronous memory element that eliminates the timing problems by isolating its inputs from its outputs

Master-slave *SR* flip-flop:



Master-slave *JK* flip-flop: since master-slave *SR* flip-flop suffers from the problem that both its inputs cannot be 1, it can be converted to a *JK* flip-flip



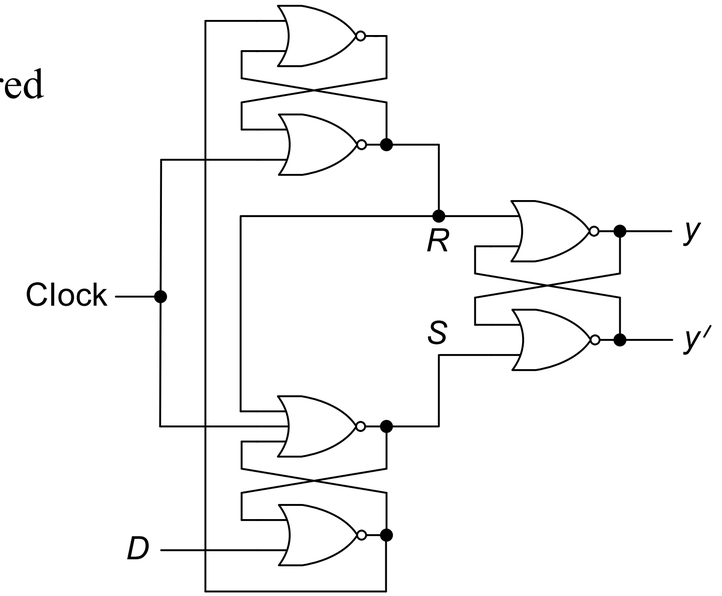
Edge Triggered Flip-Flop

Positive (negative) edge-triggered *D* flip-flip: stores the value at the *D* input when the clock makes a 0 -> 1 (1 -> 0) transition

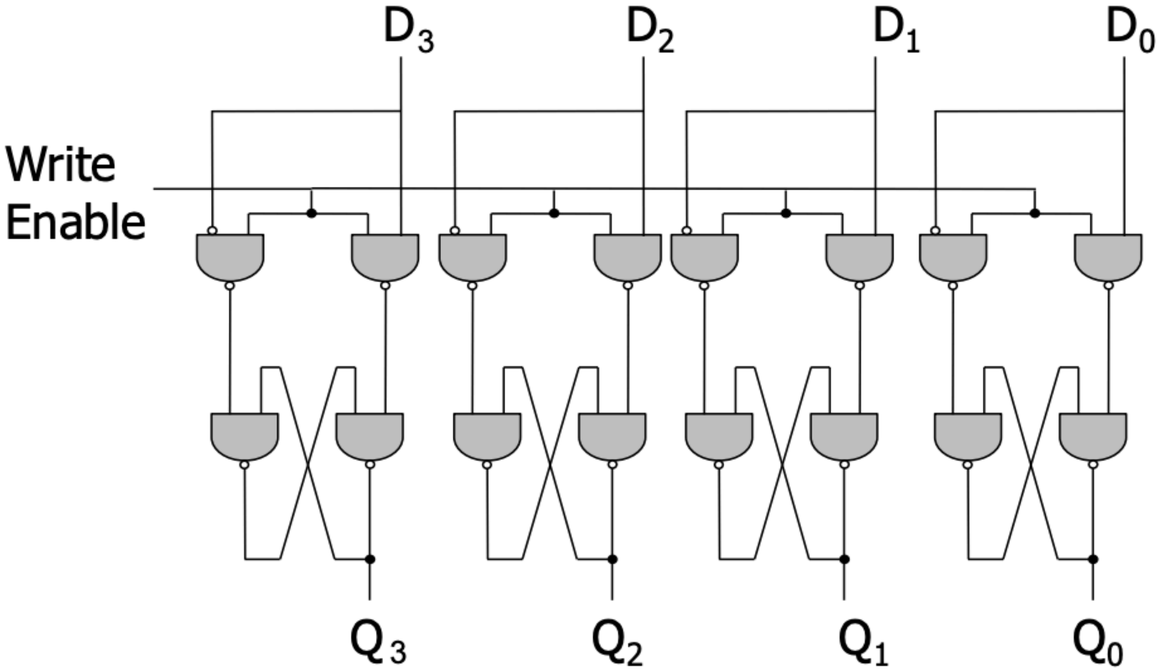
- Any change at the *D* input after the clock has made a transition does not have any effect on the value stored in the flip-flop

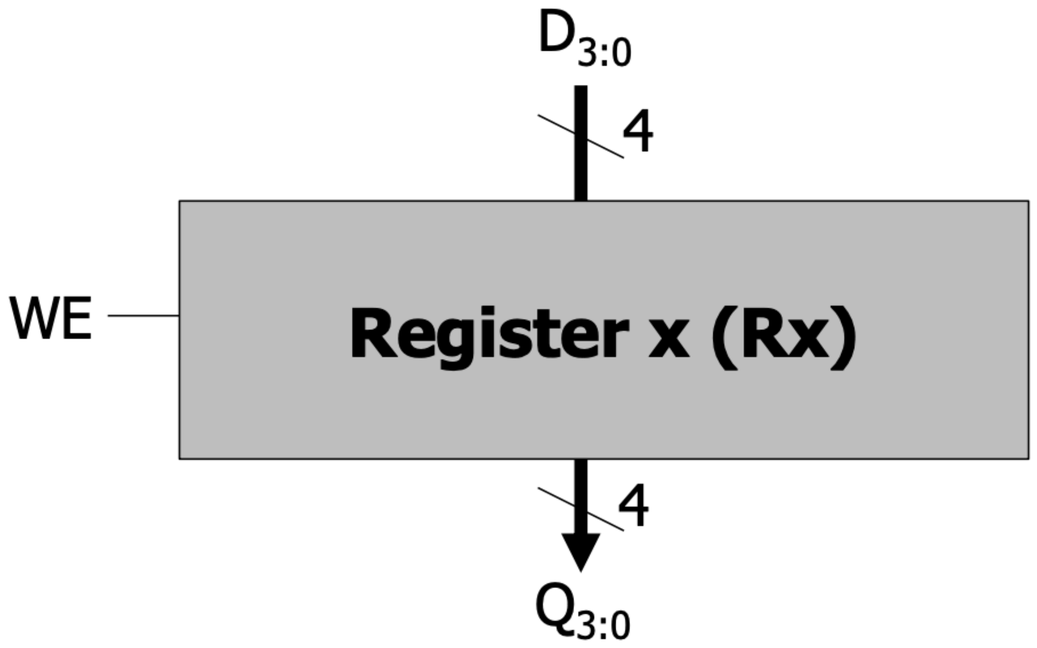
A negative edge-triggered *D* flip-flop:

- When the clock is high, the output of the bottommost (topmost) NOR gate is at *D'* (*D*), whereas the *S*-*R* inputs of the output latch are at 0, causing it to hold previous value
- When the clock goes low, the value from the bottommost (topmost) NOR gate gets transferred as *D* (*D'*) to the *S* (*R*) input of the output latch
  - Thus, output latch stores the value of *D*
- If there is a change in the value of the *D* input after the clock has made its transition, the bottommost NOR gate attains value 0
  - However, this cannot change the *SR* inputs of the output latch



Registers: Your Main Sequential Element





- Used to store data
- Basically an **array of D-flip-flops**
- You can **load data**, **reset it to zero**, and **shift it to left and right**