FPGA Workshop

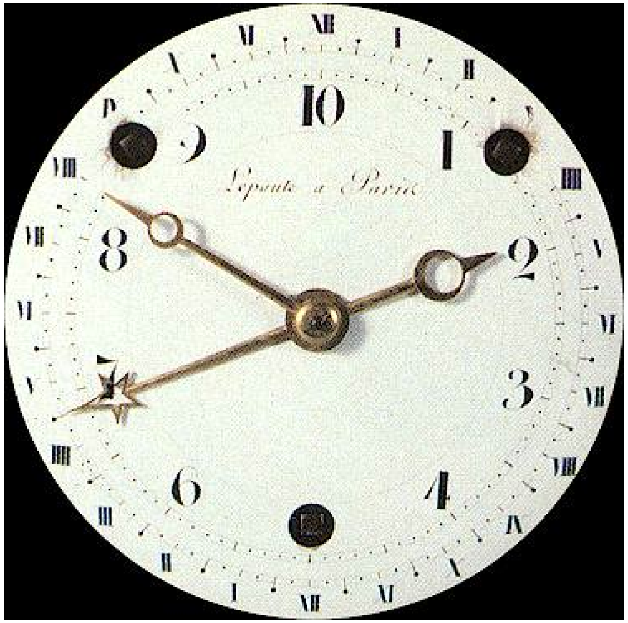
Introduction to Sequential Circuits

HacDC
Nov 2009.

Why Sequential?

- It helps when the regulator is in town
- Time can be used to regulate operations
  - I will be ready by tomorrow/one hour/(1/1Mhz)
- So far, we have implemented circuits that are asynchronous (meaning do not use a clock to regulate when the output becomes ready)
- This may be useful when you just have one function to implement, but is problematic if you need to implement functions in series.

Introducing the Clock



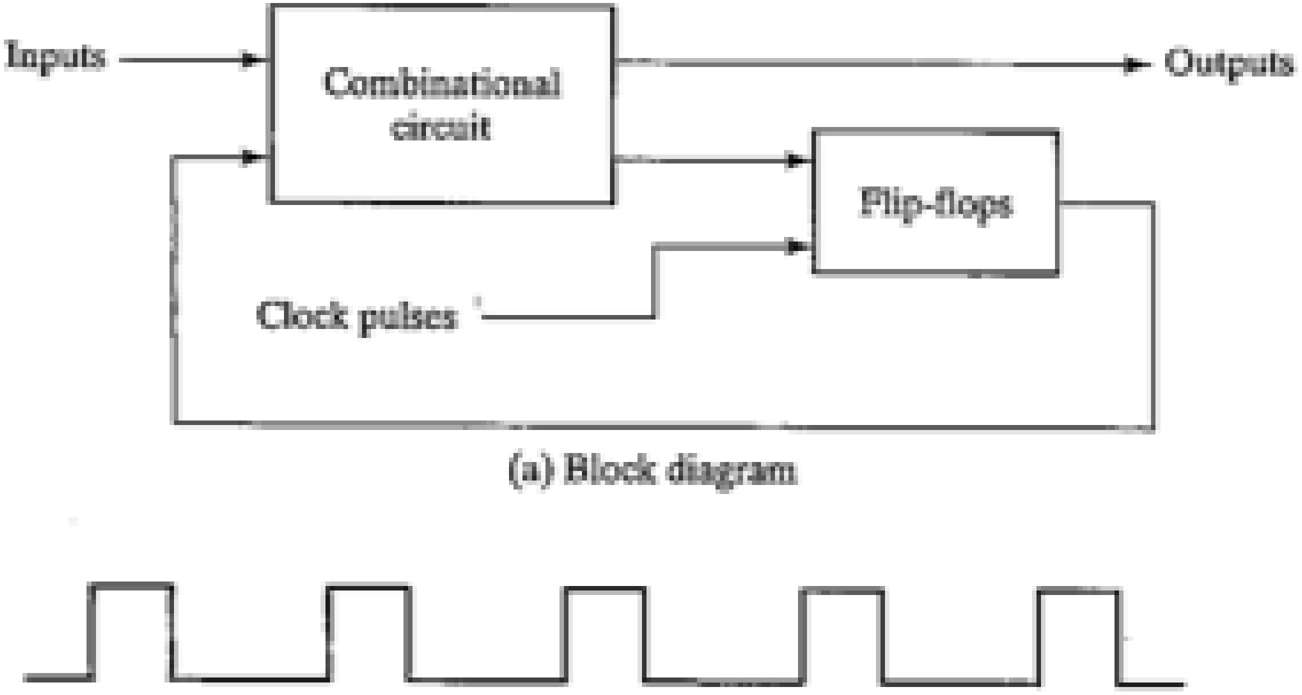
- **Combinatorial** vs **Sequential logic**
- A defining feature of Sequential logic is the presence of a clock and a **memory** element
- Not only does the output depend on where we are in (clock) time, but also what were the previous outputs (memory)
- Will be used to create Finite State Machines
  - "Where am I? What has changed? Where should I go?"

Latch vs Flip Flop

- Often (historically) has been used interchangeably
  - In this class, we will define a distinction between the two, for logical reasons
  - All Latches/Flip Flops are defined by a function table

- **Latch** is a non-clocked memory

- **Flip Flop** is a Latch with a clock input

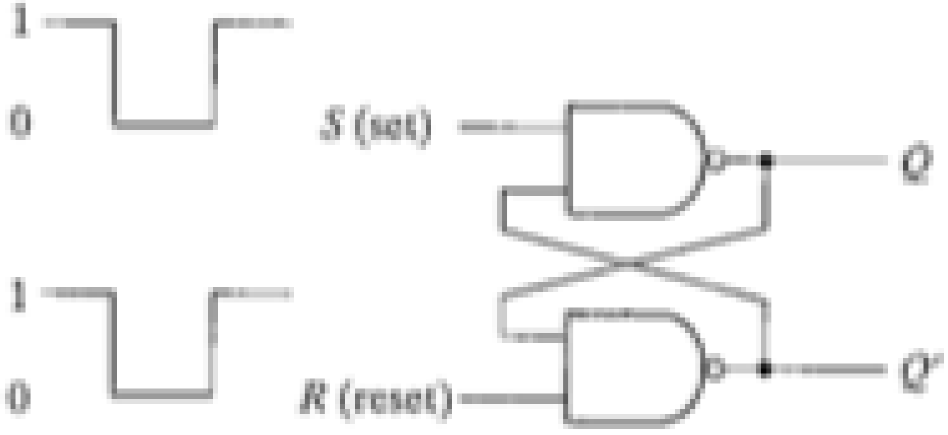


(a) Block diagram

(b) Timing diagram of clock pulses

Latches – SR Latch

- Set or Reset?
- Bistate vibrator!!



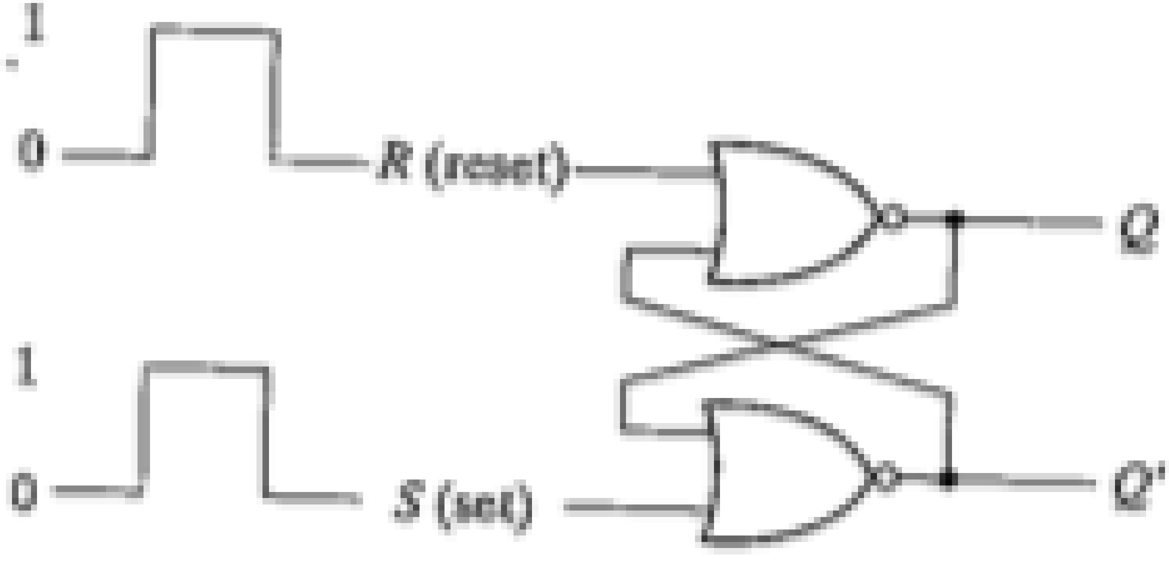
(a) Logic diagram

S	R	Q	Q'	
1	0	0	1	
1	1	0	1	(after $S = 1, R = 0$)
0	1	1	0	
1	1	1	0	(after $S = 0, R = 1$)
0	0	1	1	

(b) Function table

Latches – SR Latch

- Implemented with NOR Gates



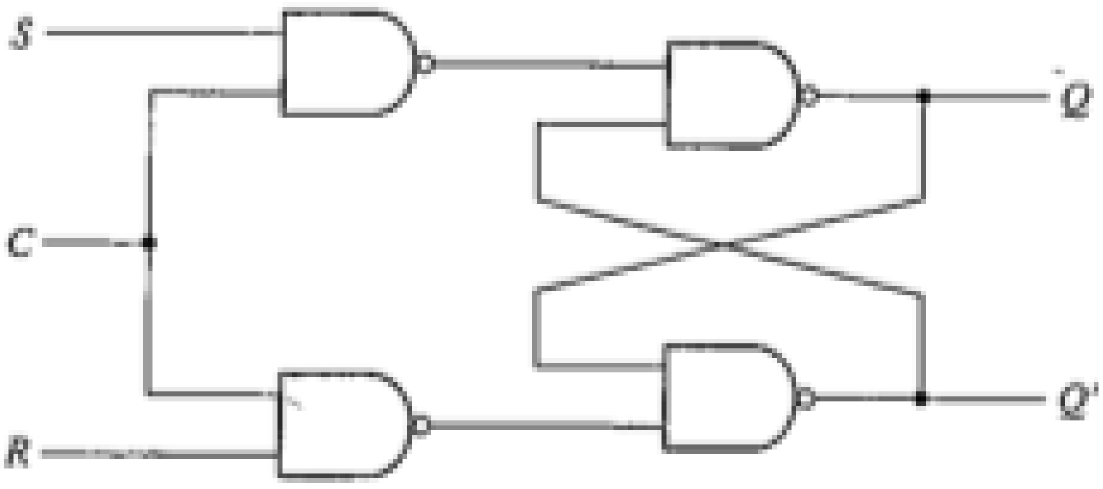
(a) Logic diagram

S	R	Q	Q'
1	0	1	0
0	0	1	0
0	1	0	1
0	0	0	1
1	1	0	0

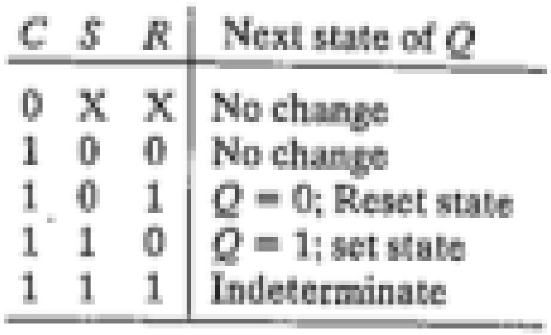
(b) Function tabl

Latches - SR Latch

- Implemented with an additional **control input**



(a) Logic diagram

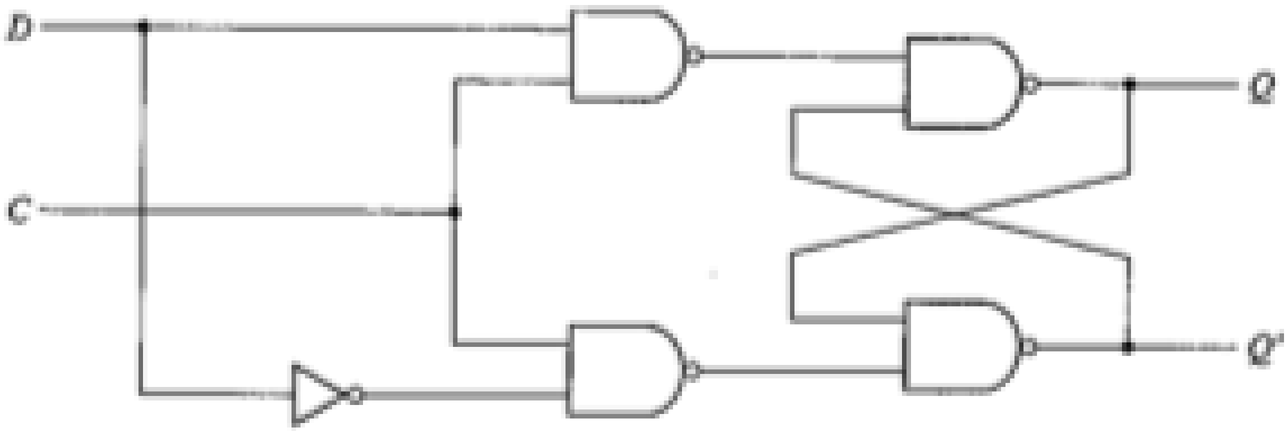


C	S	R	Next state of Q
0	X	X	No change
1	0	0	No change
1	0	1	Q = 0; Reset state
1	1	0	Q = 1; set state
1	1	1	Indeterminate

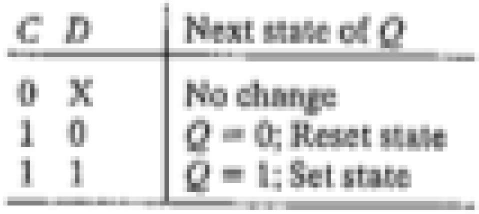
(b) Function table

Latches – D Latch

- Simply follows the input
- An SR Latch with no ambiguous states



(a) Logic diagram

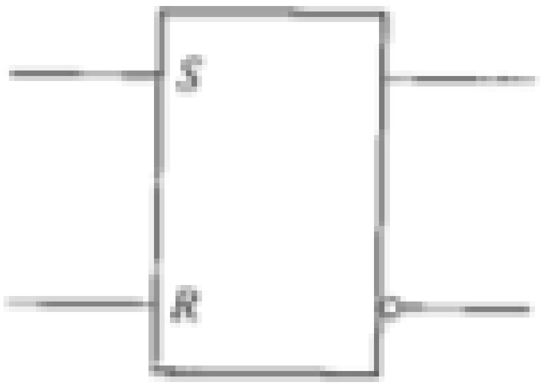


C	D	Next state of Q
0	X	No change
1	0	$Q = 0$; Reset state
1	1	$Q = 1$; Set state

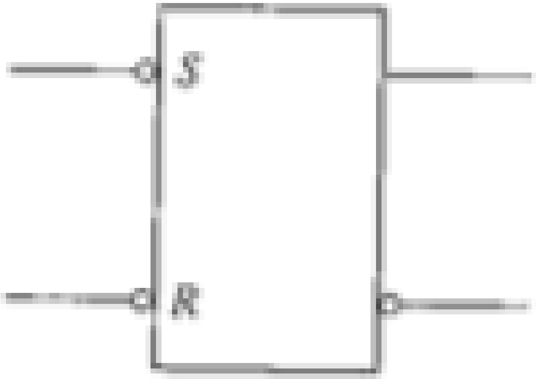
(b) Function table

Latches - Symbols

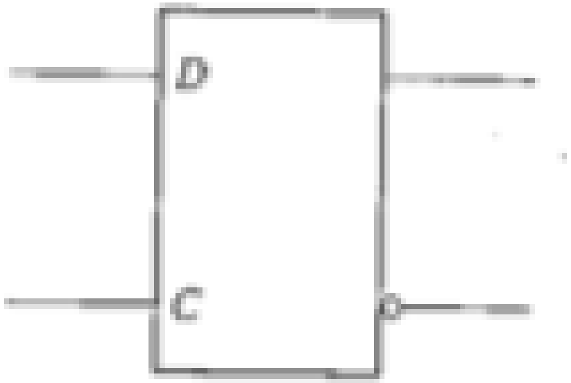
- Circle means invert!



SR



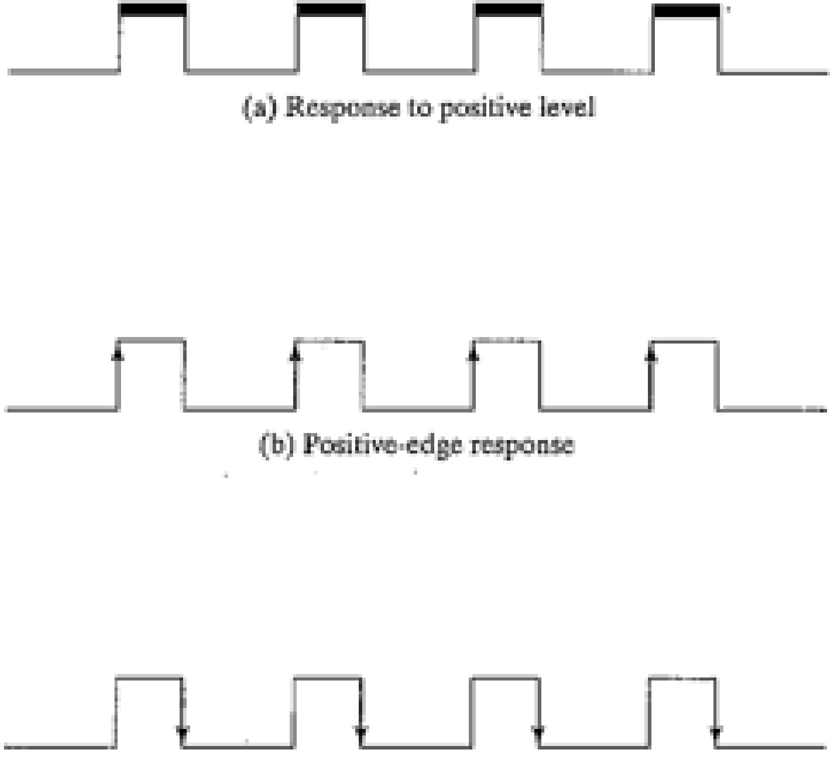
$\overline{SR}$



D

What to do with the Clock?

- Its useful to define a time period in which the output will be ready (e.g. high time, or low time)
- Even better, get locked to edge of the clock!



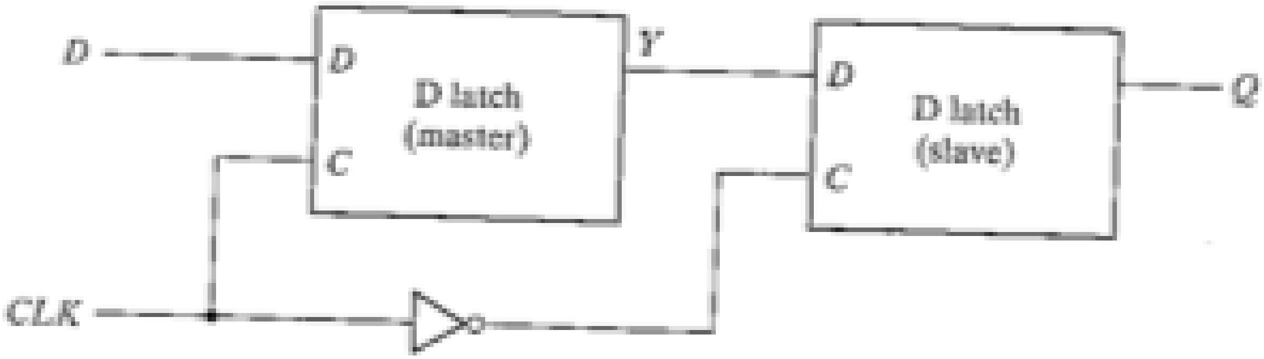
(a) Response to positive level

(b) Positive-edge response

(c) Negative-edge response

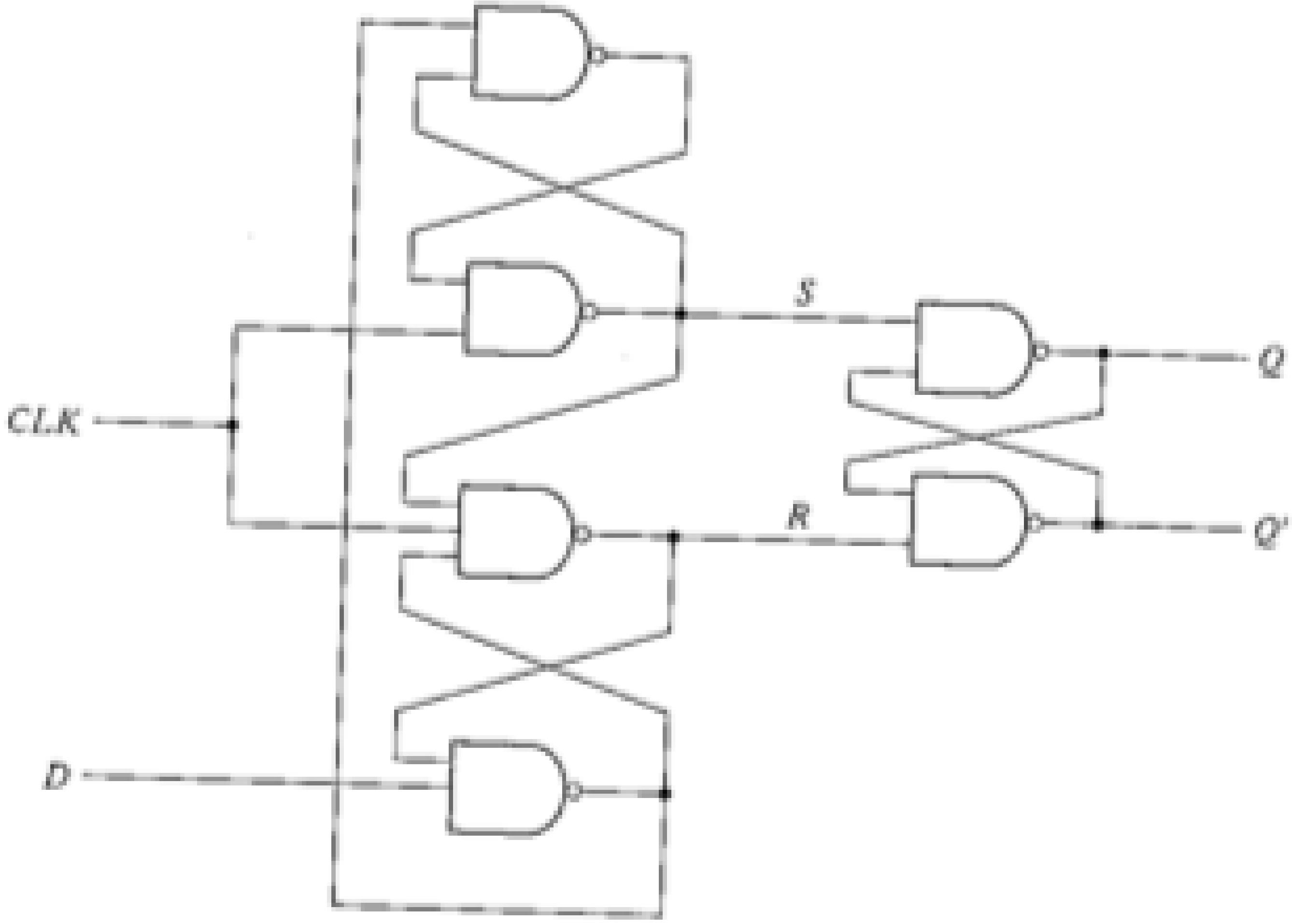
Introducing the Flip Flop!

- Add a clock, and you have a Flip-Flop
- Output change in response to inputs **and** the clock!
- One way to implement an edge triggered Flip Flop is using a Master/Slave Implementation

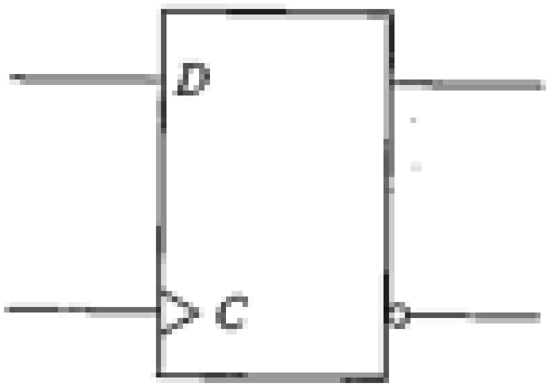


Flip Flops – D

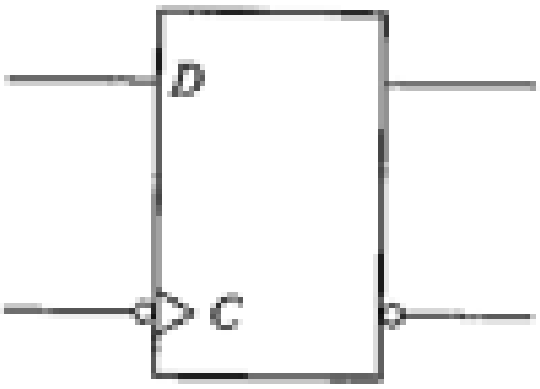
- Positive Edge Triggered Flip Flop using three SR Latches



Where to trigger?



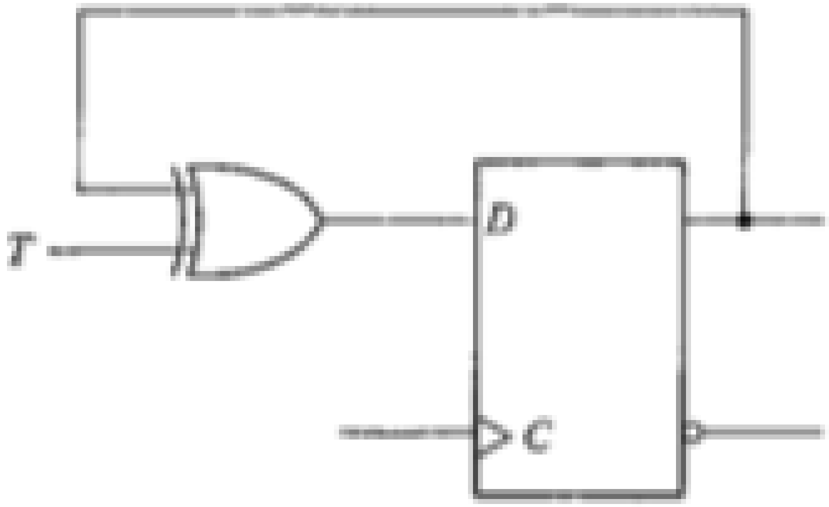
(a) Positive-edge



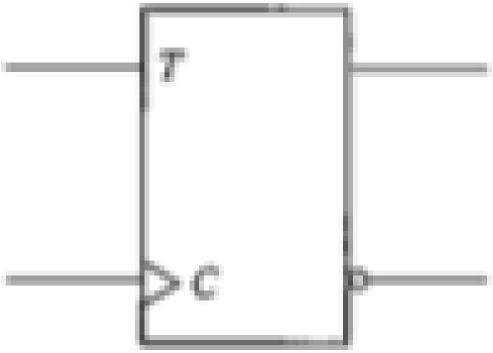
(a) Negative-edge

Flip Flops - T

- Boring! but sometimes useful



(b) From *D* flip-flop



(c) Graphic symbol

Flip Flops - JK

- Badass flip-flop
- Allows you to set/reset or complement the previous input

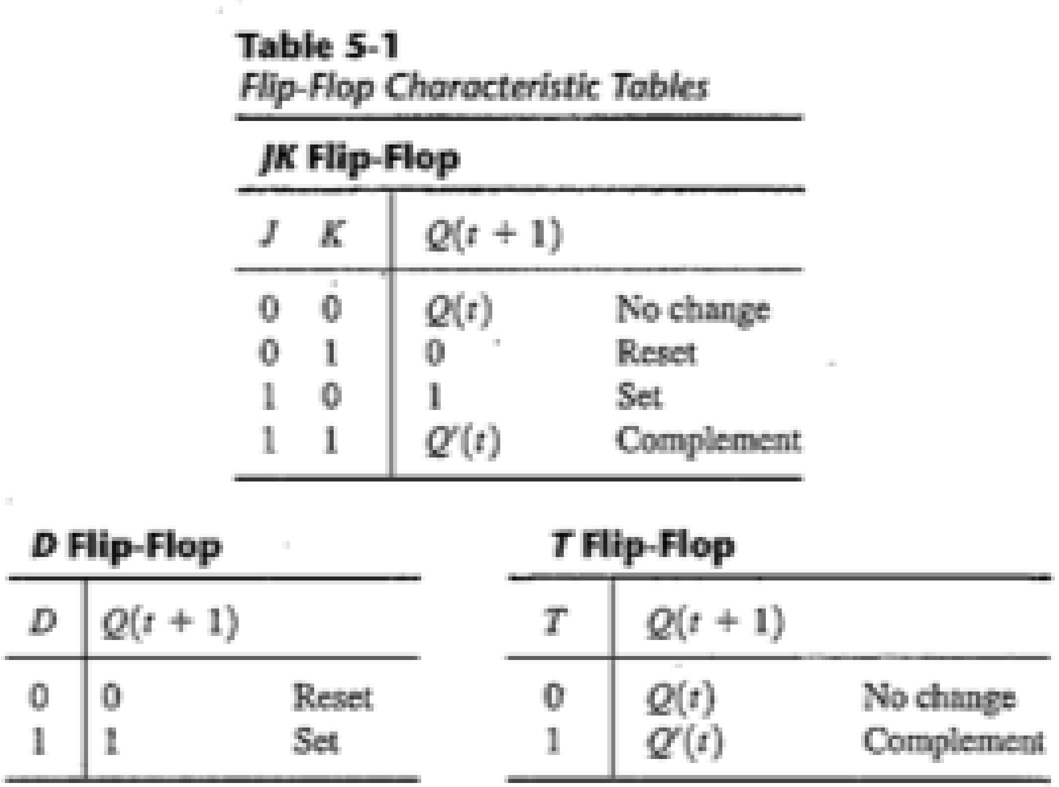


Table 5-1
Flip-Flop Characteristic Tables

***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

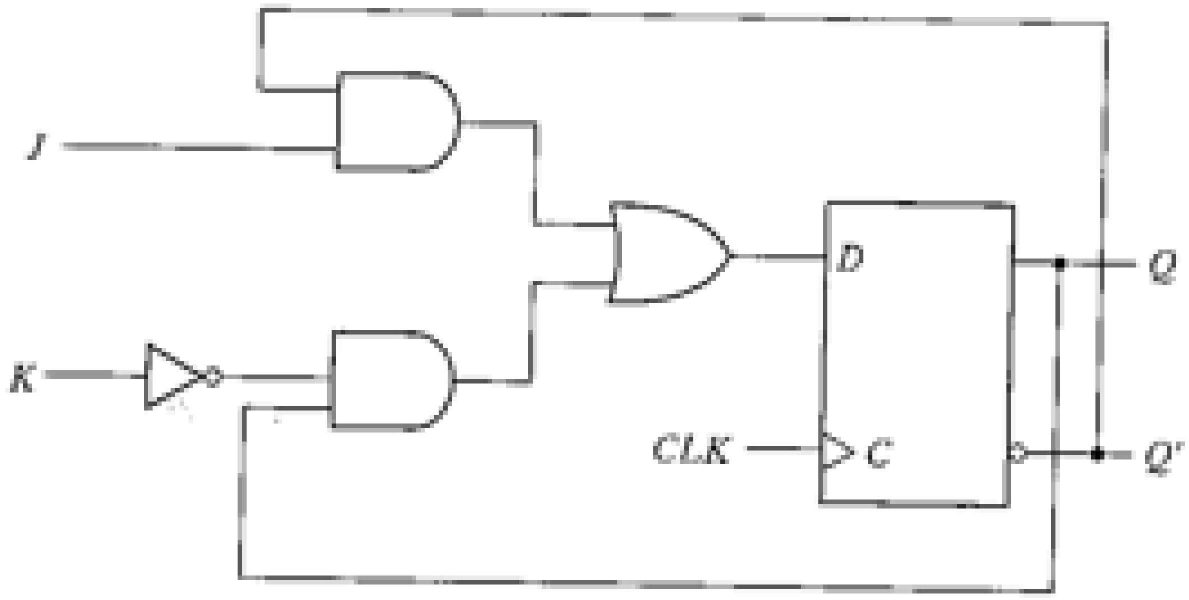
***D* Flip-Flop**

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

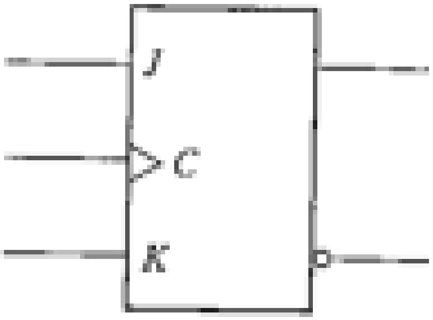
***T* Flip-Flop**

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

Flip Flops - JK



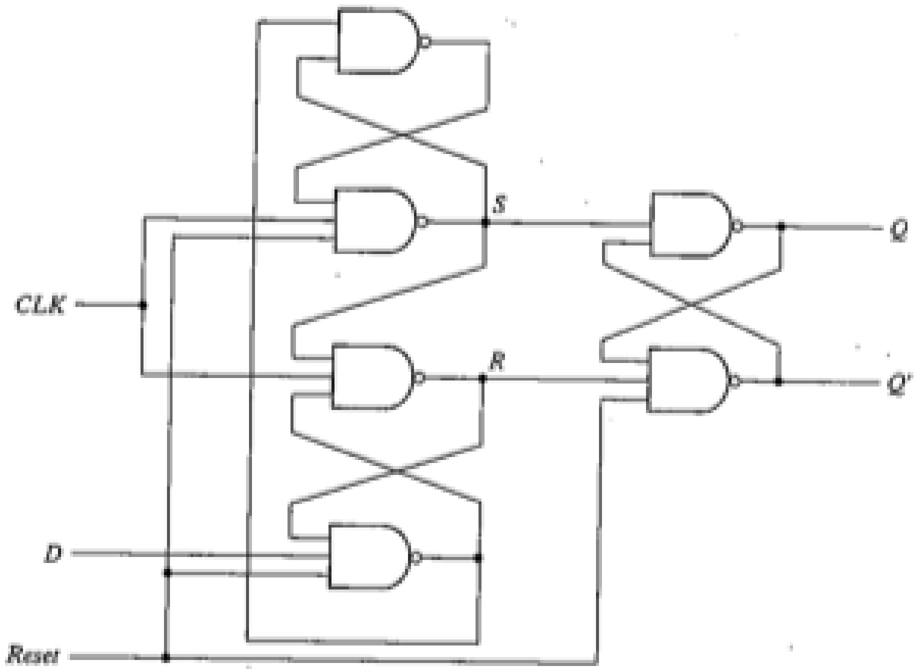
(a) Circuit diagram



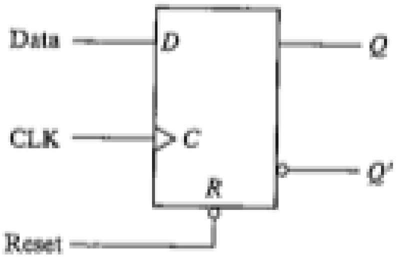
(b) Graphic symbol

What if I want a flip flop with a hammer?

Asynchronous Reset



(a) Circuit diagram



(b) Graphic symbol

R	C	D	Q	Q'
0	X	X	0	1
1	↑	0	0	1
1	↑	1	1	0

(b) Function table

Fine.

Problems

-
-
-
-
  - None.
-
-
-