

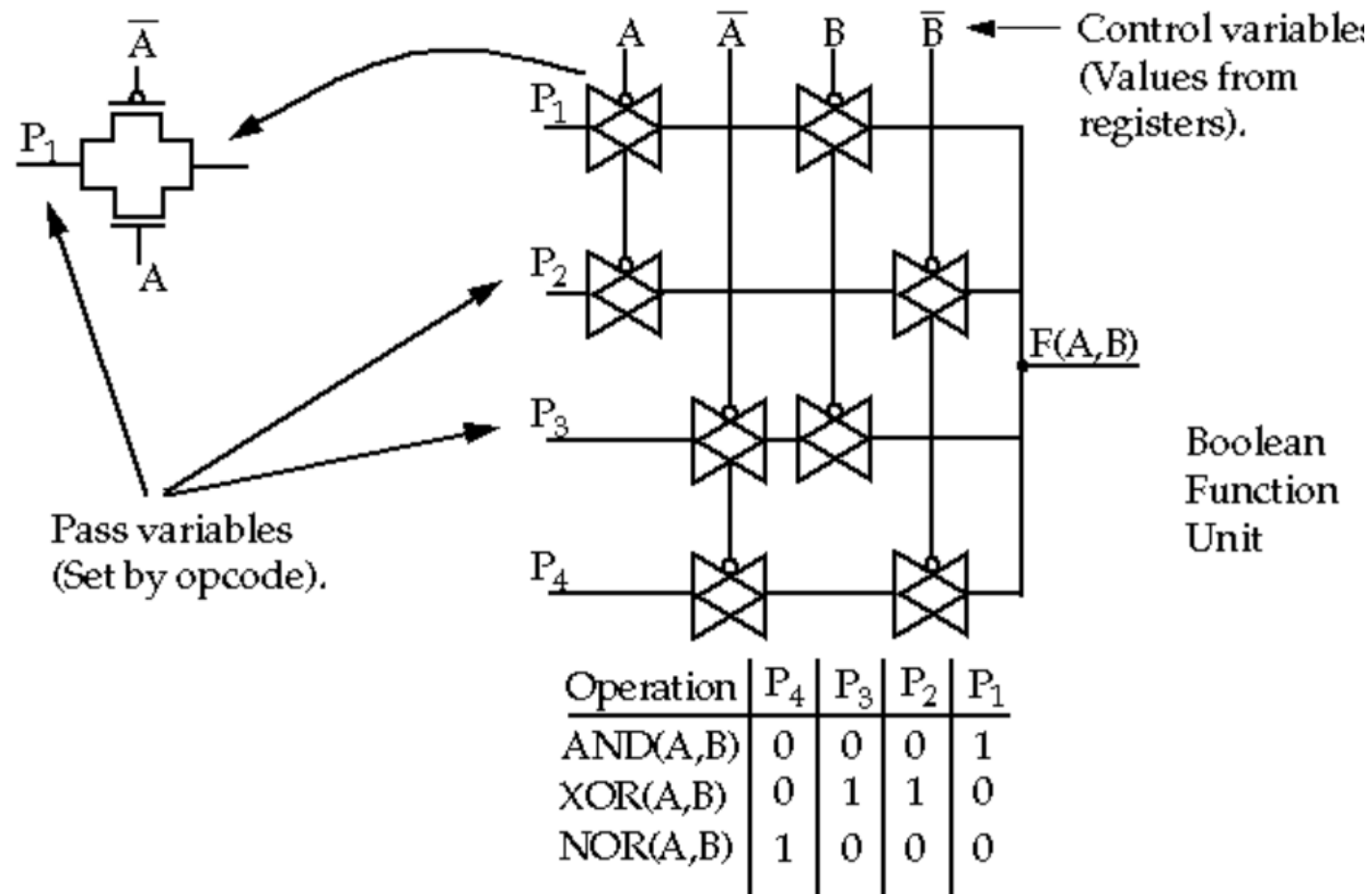
Applied Digital Logic Design

COMBINATIONAL LOGIC : No Clock

- ADDER
- SUBTRACTOR
- MULTIPLIER
- ALU
- etc

COMBINATIONAL LOGIC : No Clock (Transmission gate/ Mux-Demux/Coder/Dcoder etc)

• *Pass-Transistor Logic:*



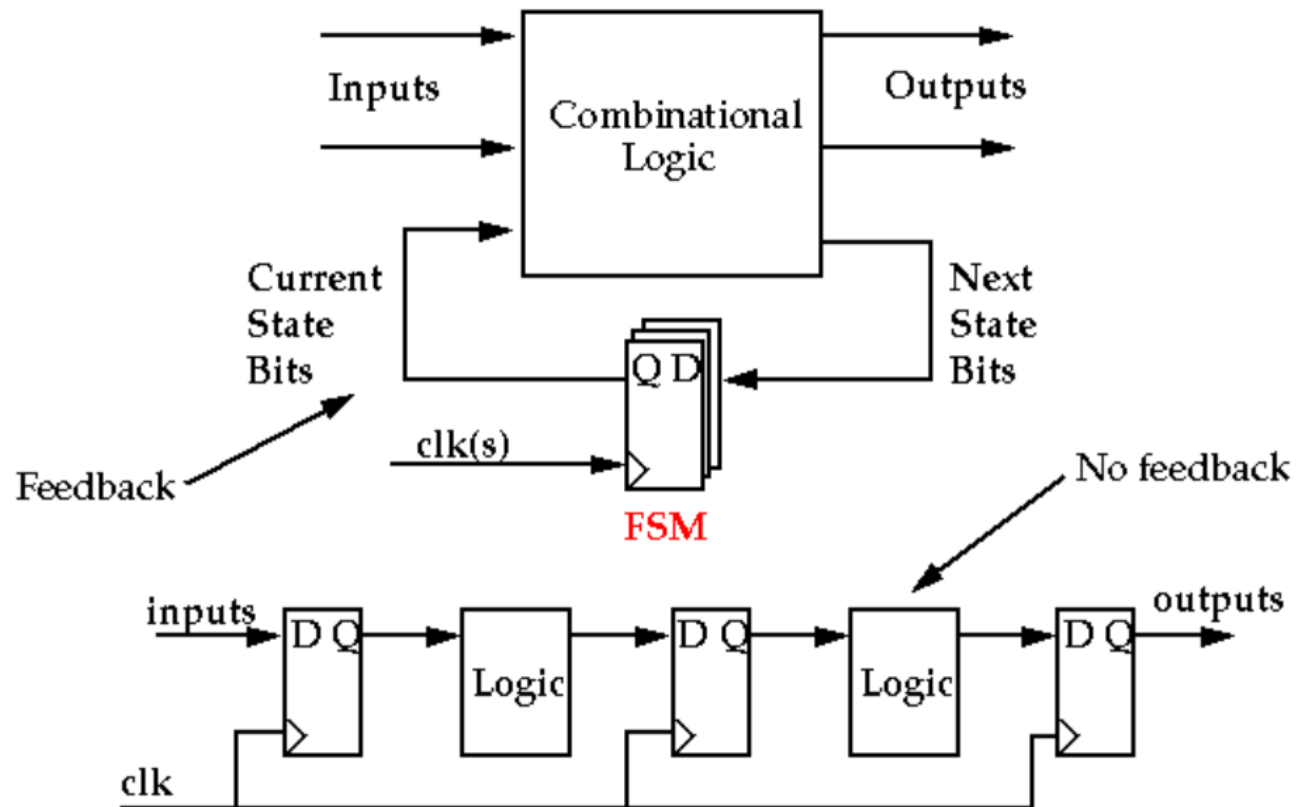
SEQUENTIAL LOGIC : Clock

- FLIP-FLOP (COVER IN LECTURE)
- COUNTER & SHIFT REGISTER (HANDS OUT will given in LAB EXPERIMENT TIME)
- DYNAMIC LOGIC
- MEMORY (IN BRIEF)
- DAC/ADC
- etc

SEQUENTIAL LOGIC

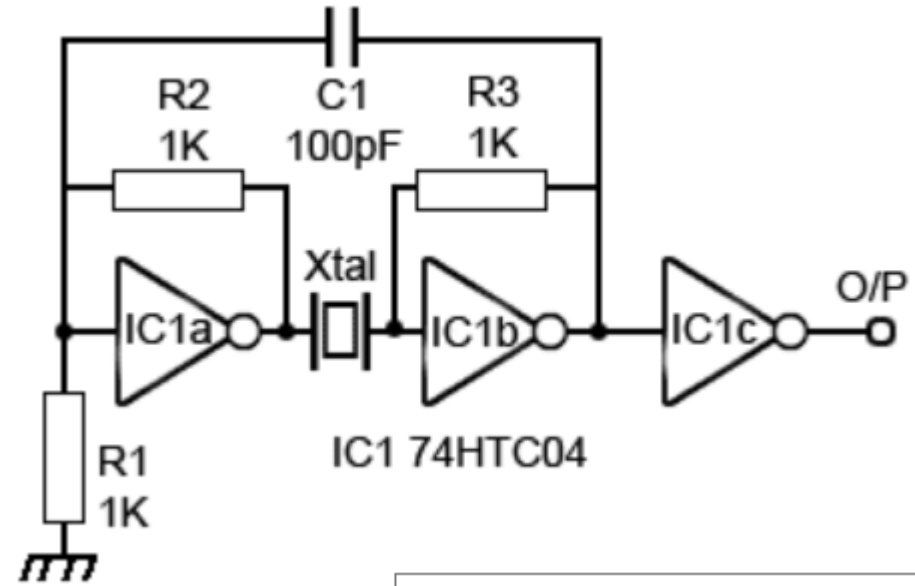
Clocked Systems

- *Majority of VLSI systems are Finite State machines and Pipelined machines:*

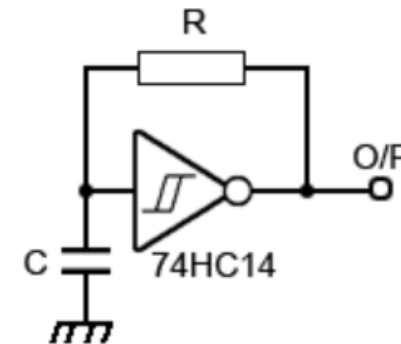


Clock

- DIGITAL CLOCK :
 - ✓ Inverter based
 - ✓ Low speed
- ANALOG CLOCK
 - ✓ LC based
 - ✓ High speed
- REFERENCE CLOCK
 - ✓ x-tal based
 - ✓ Low speed with strong purity

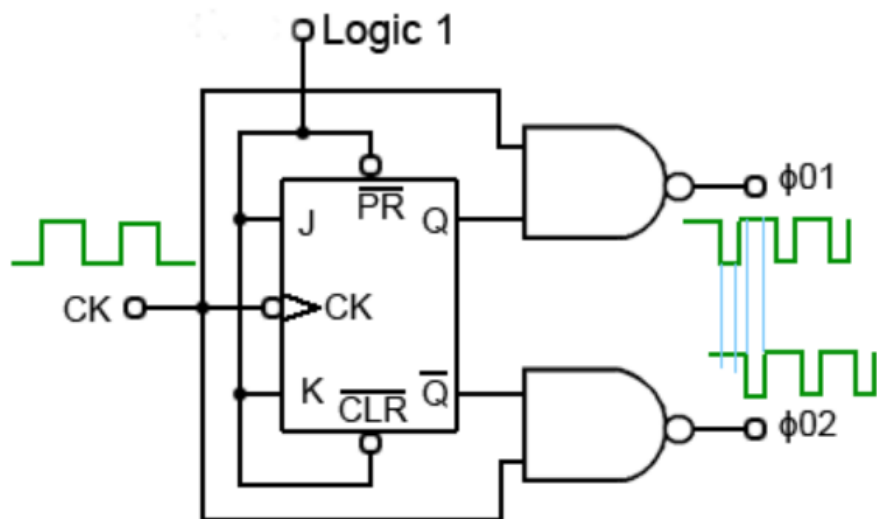
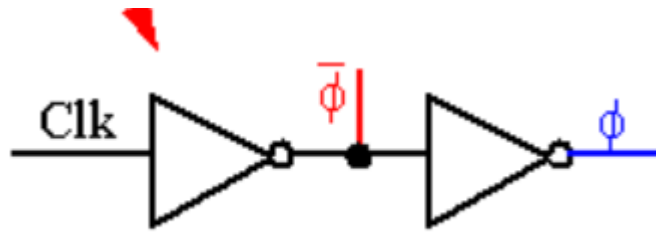


Crystal Controlled Clock Oscillator

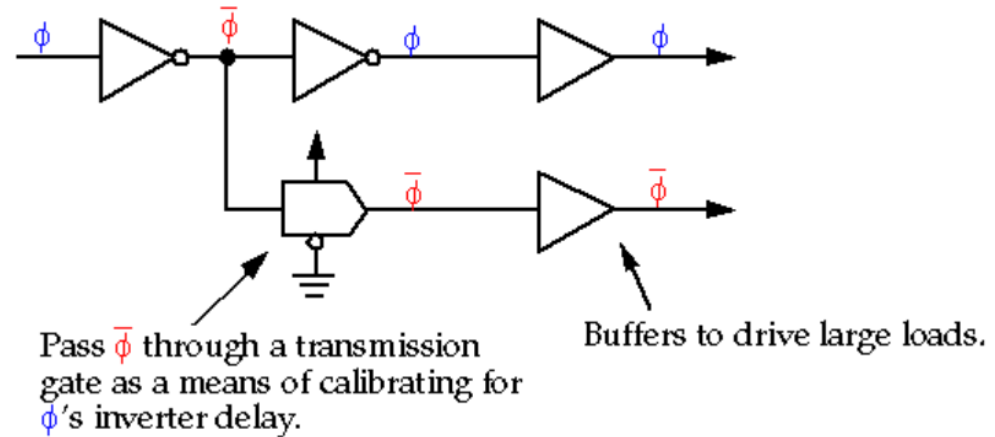


Simple Schmitt
Inverter Clock Oscillator

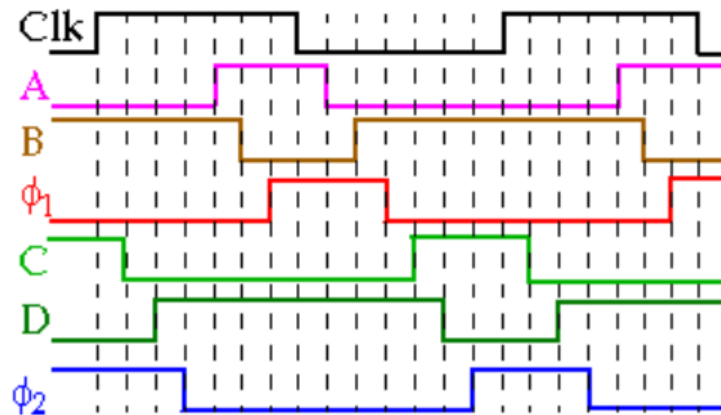
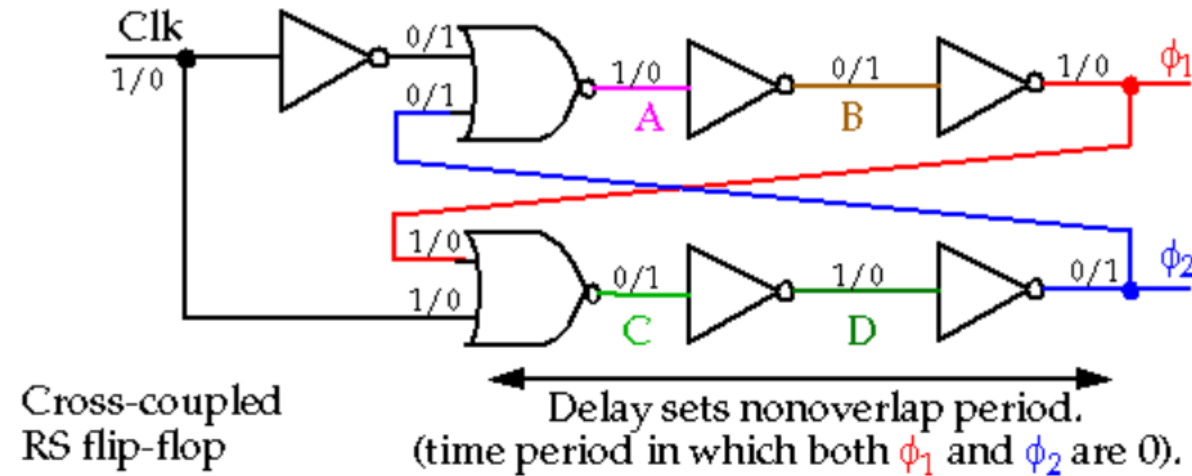
Clock: frame (CLK -0 and CLKBar-180)



Single Phase Global Clock Generation



Clock: frame (Non Overlapping)

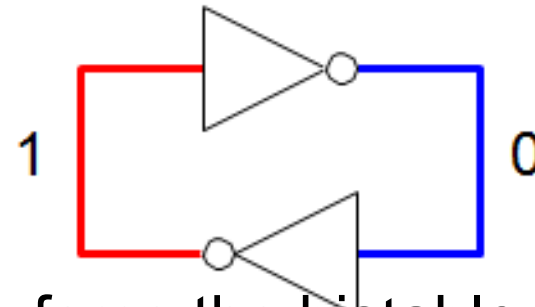
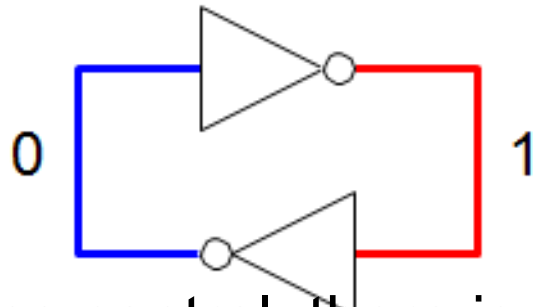


The “WHY” slide

- Memory storage elements
 - In order to do fun problems like the door combination lock, we must know the building blocks (like how you had to learn AND and OR before you could do functional things). Be patient --- once you know these elements, you can build a lot of meaningful functions
- State diagrams
 - For combinational logic, truth table was an invaluable visualization tool for a function. For sequential logic, state diagram serves as a way to visualize a function.

Bistable Circuit

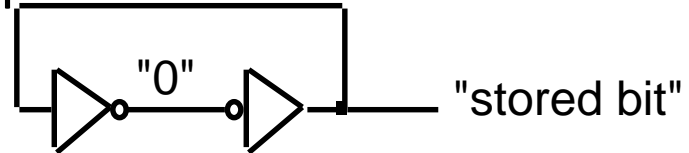
- At the heart of a bistable circuit is a pair of inverters connected in a loop — with **feedback**, in other words. It has two stable states.



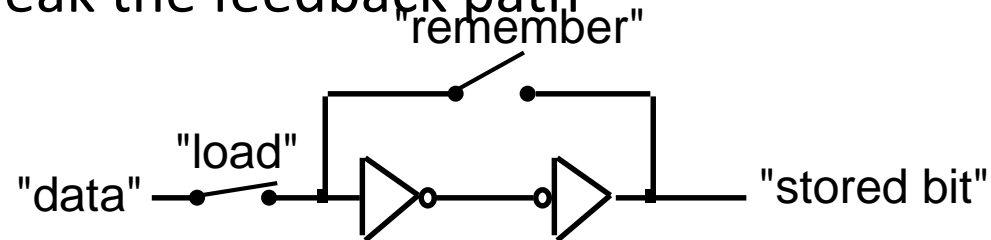
- Without some control, there isn't a way to force the bistable circuit into one or the other state.

How do we store info? Feedback

- Two inverters can hold a bit
 - As long as power is applied "1"

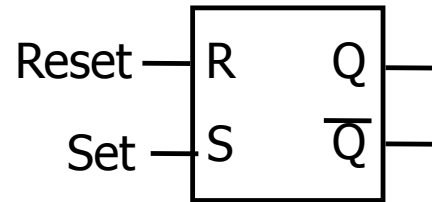


- Storing a new memory
 - Temporarily break the feedback path



The SR latch

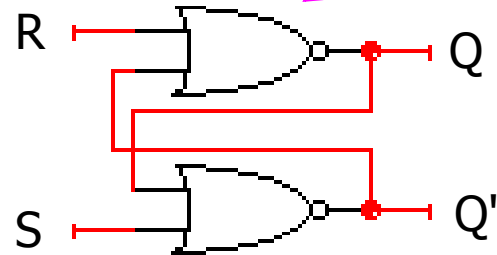
- Cross-coupled NOR gates
 - Can set ($S=1, R=0$) or reset ($R=1, S=0$) the output



S	R	Q
0	0	hold
0	1	0
1	0	1
1	1	disallow

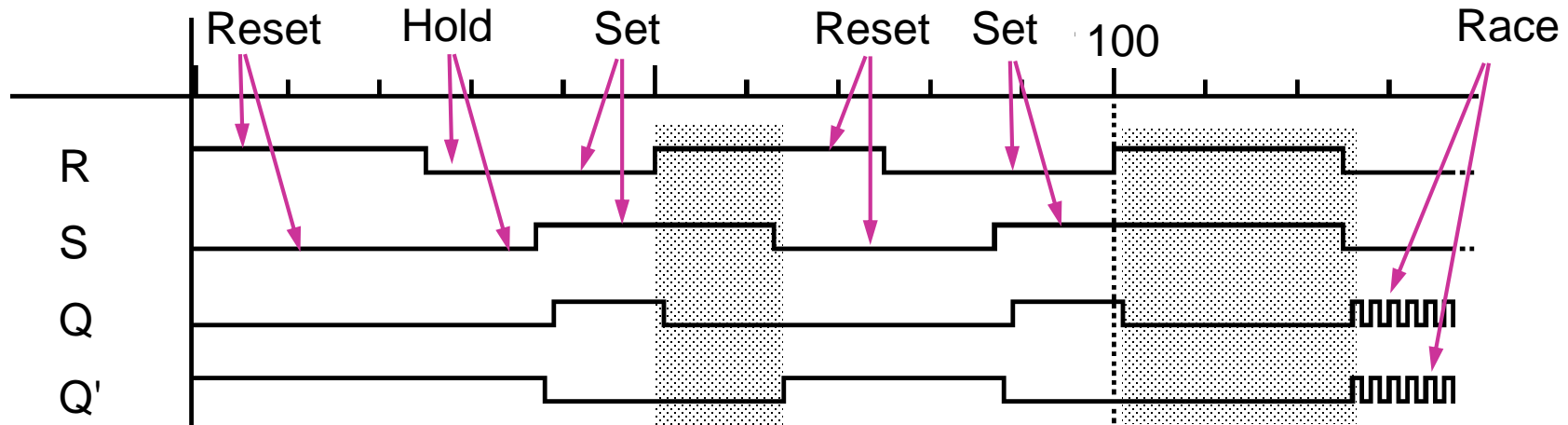
SR latch behavior

- Truth table and timing



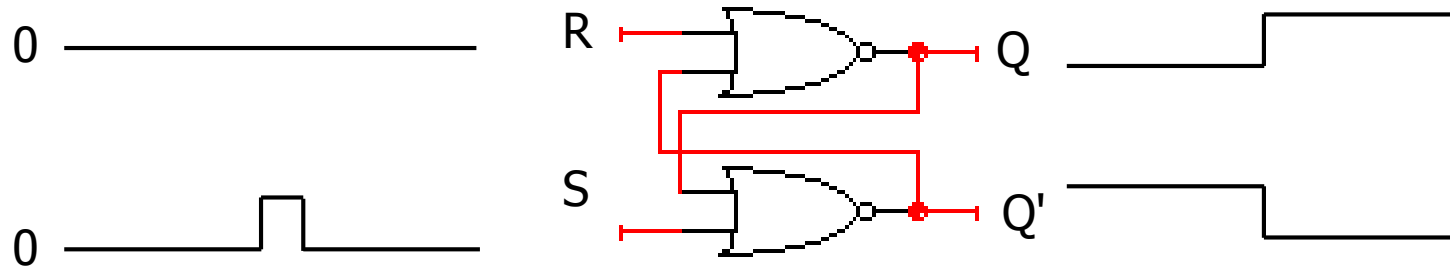
NOR output is 1
Only when both inputs are 0

S	R	Q
0	0	hold
0	1	0
1	0	1
1	1	disallow



SR latch is glitch sensitive

- Static 0 hazards can set/reset latch
 - Glitch on S input sets latch
 - Glitch on R input resets latch



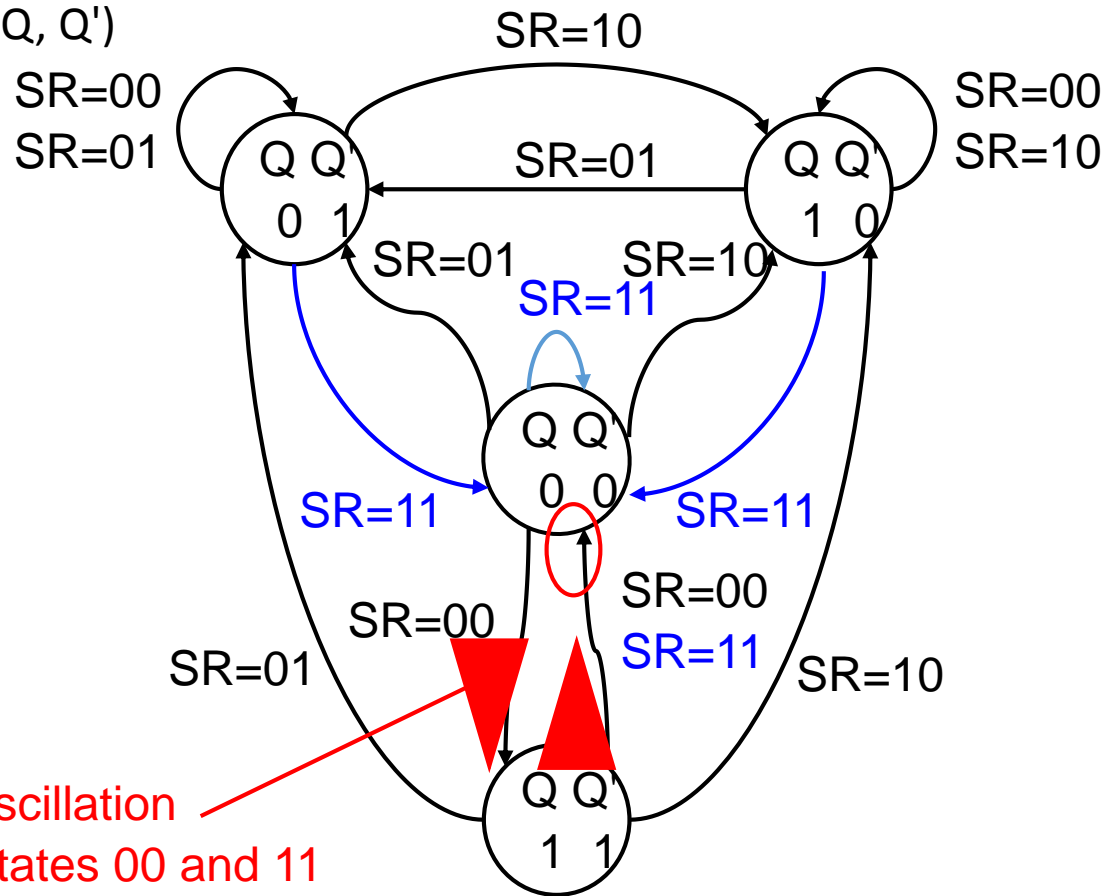
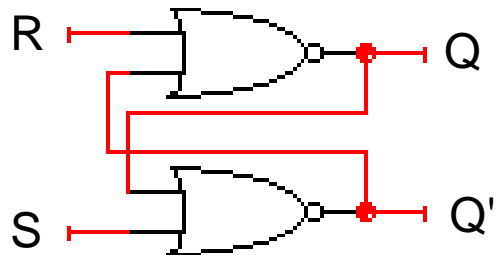
State diagrams

- How do we characterize logic circuits?
 - Combinational circuits: Truth tables
 - Sequential circuits: State diagrams
- First draw the states
 - States \equiv Unique circuit configurations
- Second draw the transitions between states
 - Transitions \equiv Changes in state caused by inputs

Example: SR latch

- Begin by drawing the states
 - States** \equiv Unique circuit configurations
 - Possible values for feedback (Q, Q')

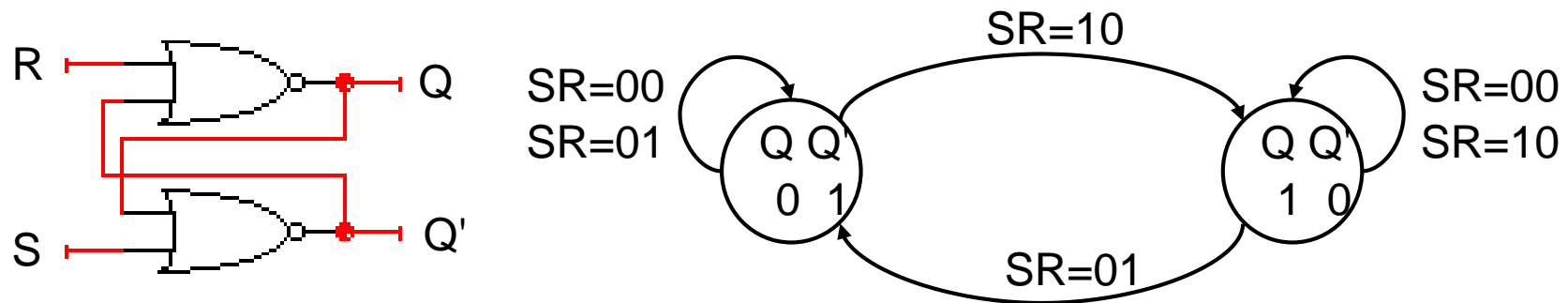
S	R	Q
0	0	hold
0	1	0
1	0	1
1	1	disallow



possible oscillation
between states 00 and 11
(when SR=00)

Observed SR latch behavior

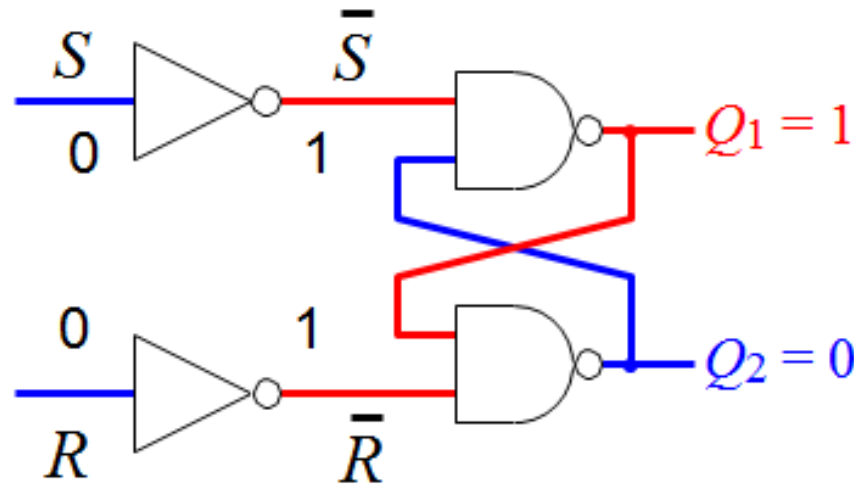
- The 1–1 state is transitory
 - Either R or S “gets ahead”
 - Latch settles to 0–1 or 1–0 state ambiguously
 - Race condition → non-deterministic transition
 - Disallow $(R,S) = (1,1)$



Core of a Flip-Flop:

The set–reset or *SR* Latch

- Acts as a simple memory with **two stable states at the two output** when $S = R = 0$

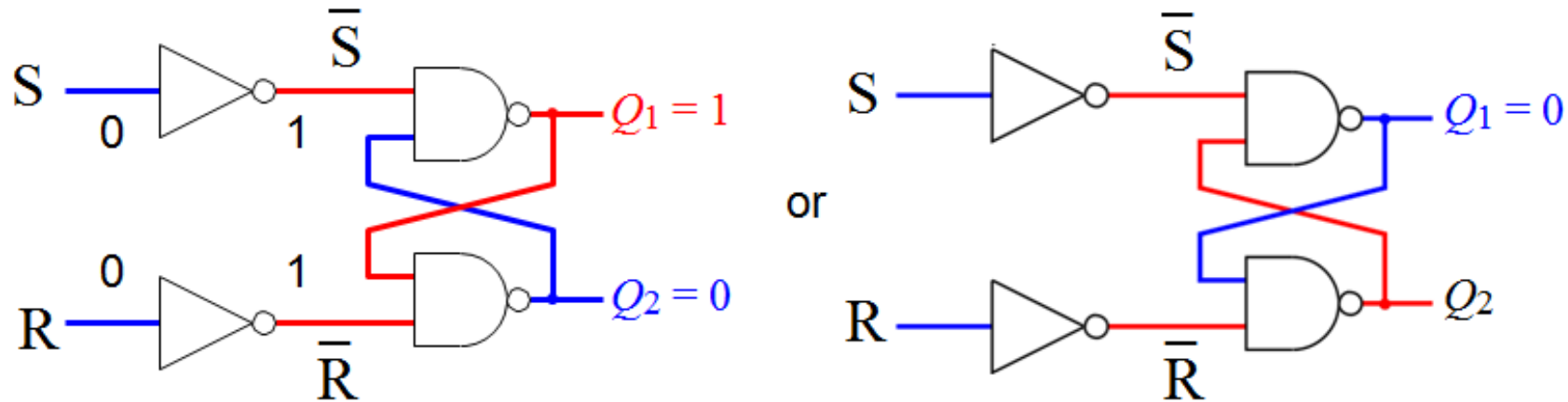


\bar{Q}

- Q_1 and Q_2 are the outputs of the S-R latch.
- When Q_1 is known as Q and Q_2 is also called Q' or \bar{Q} (spoken as Q bar), meaning that its value is not Q or the opposite of Q .

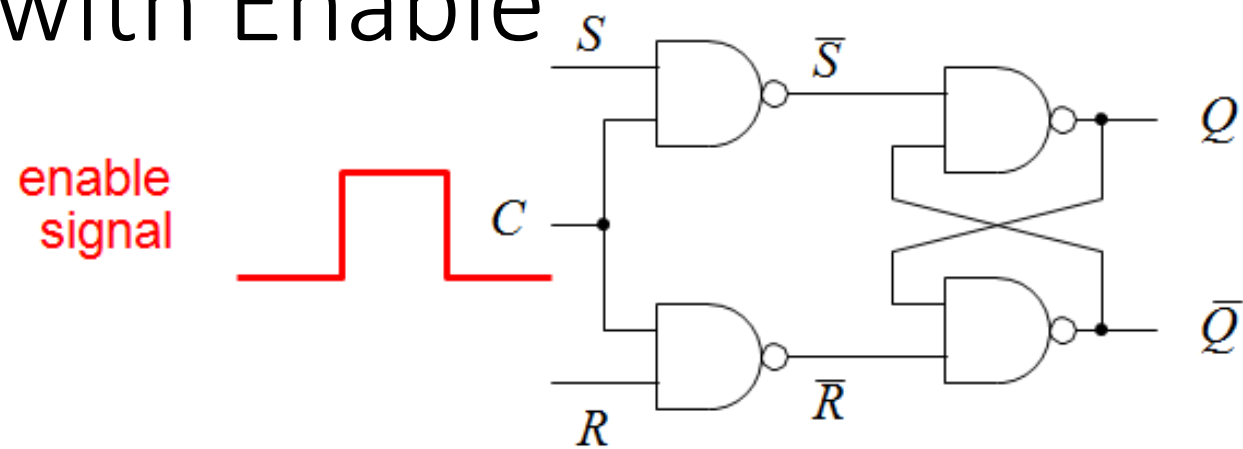
S-R Latch

Acts as a simple memory with **two stable states** when $S = R = 0$:



- The latch
 - **holds** (stores) when $S = R = 0$
 - is **set (to 1)** by bringing $S = 1$ with $R = 0$
 - is **reset (to 0)** or **cleared** by bringing $R = 1$ with $S = 0$
- The condition $S = R = 1$ must be avoided because it leads to an **indeterminate** condition, where the output can not be predicted at any one point in time. This can cause a **race** condition to occur when the inputs change to $S = R = 0$.

SR Latch with Enable



- The S and R inputs only effect the output states when the **enable** input C is high.
 - This controls when the latch responds to its inputs.
- The latch holds (stores) its value while the enable input is low — latches it!
 - Any changes in the inputs **during** the time when enable is high will affect the output immediately: the circuit is said to be **transparent**.
 - This circuit still has a major problem: the stored value is indeterminate if $S = R = 1$ when the clock goes low

Logic Table

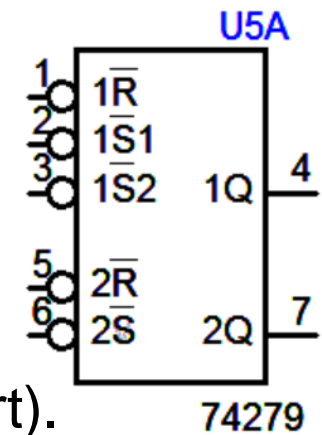
SR Latch

S	R	Q
0	0	Last Q
0	1	0
1	0	1
1	1	

SR Latch with Enable

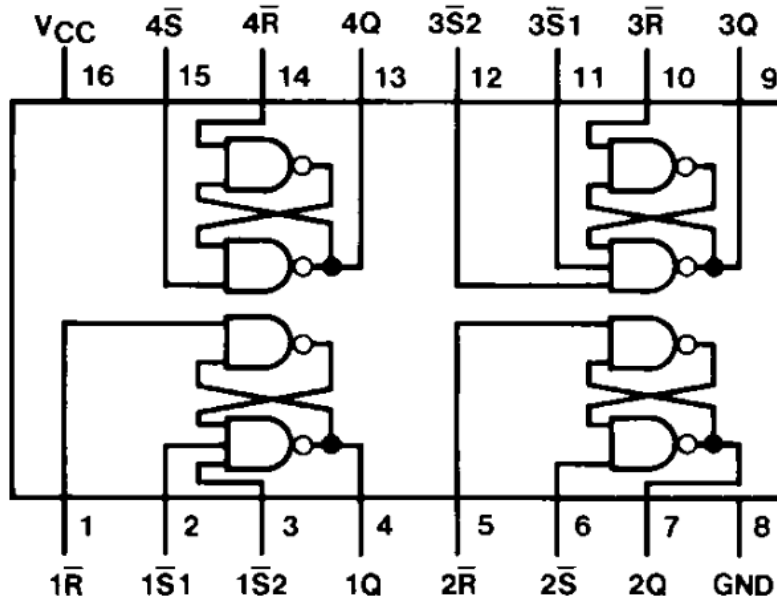
S	R	E	Q
0	0	1	Last Q
0	1	1	0
1	0	1	1
1	1	1	
X	X	0	Last Q

74279



- Note that there is dual SR bar latch in PSpice (2 in 1 part).
 - It may appear that the undefined operation has been designed out of its operation when you use this part in a simulation. However, the datasheet indicates that the race condition may show up.

Connection Diagram



Function Table

Inputs		Output
\bar{S} (Note 1)	\bar{R}	Q
L	L	H (Note 2)
L	H	H
H	L	L
H	H	Q_0

H = HIGH Level

L = LOW Level

Q_0 = The Level of Q before the indicated input conditions were established.

Note 1: For latches with double \bar{S} inputs:

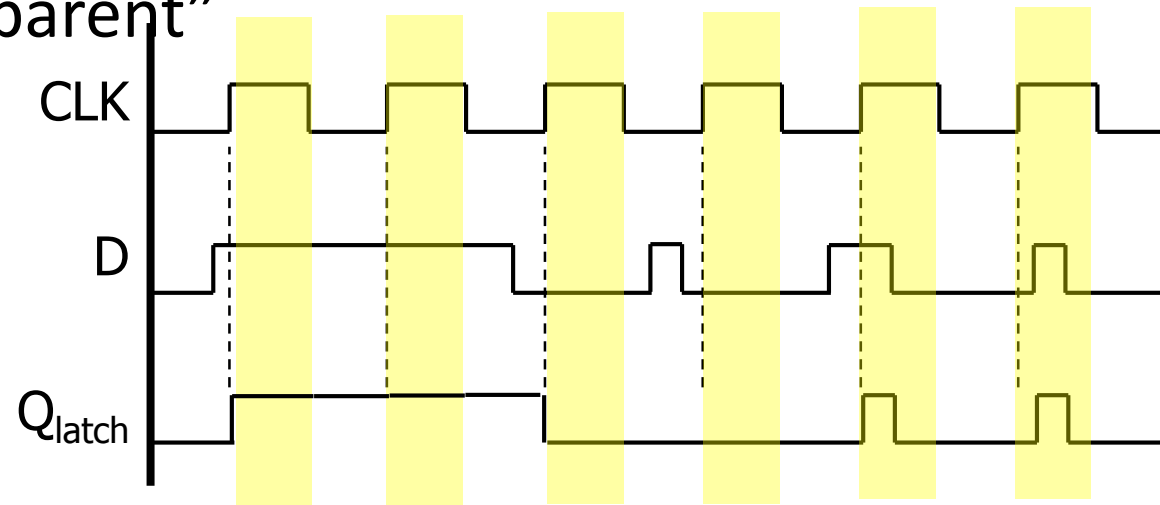
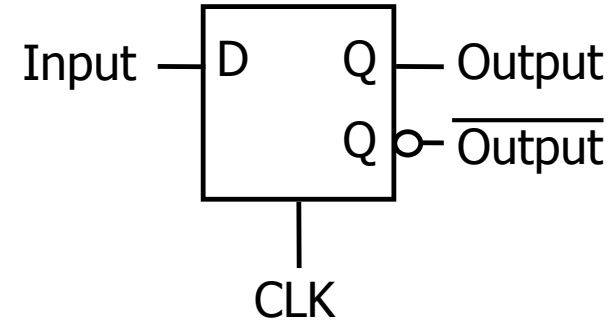
H = both \bar{S} inputs HIGH

L = one or both \bar{S} inputs LOW

Note 2: This output level is pseudo stable; that is, it may not persist when the \bar{S} and \bar{R} inputs return to their inactive (HIGH) level.

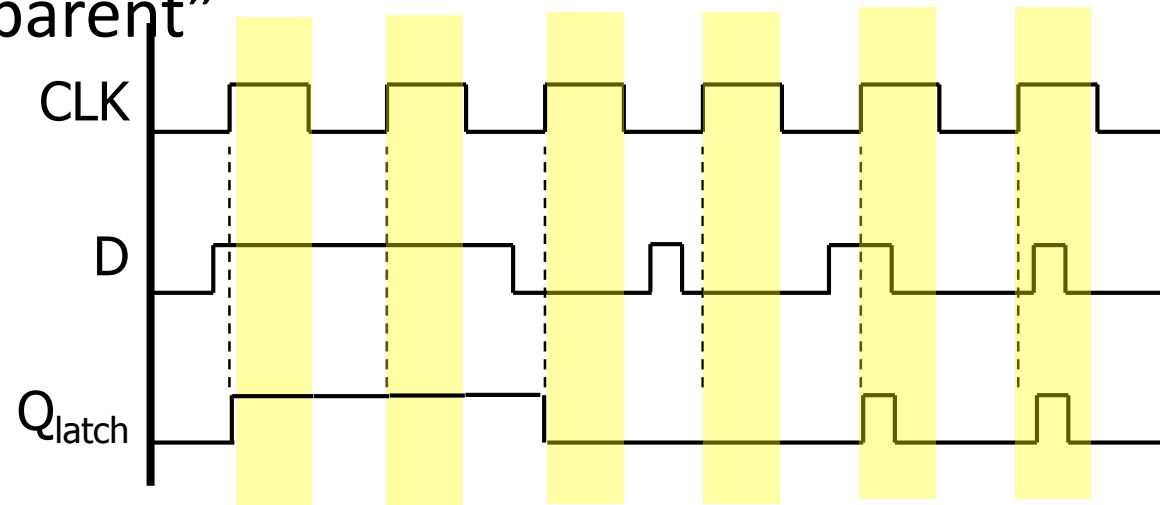
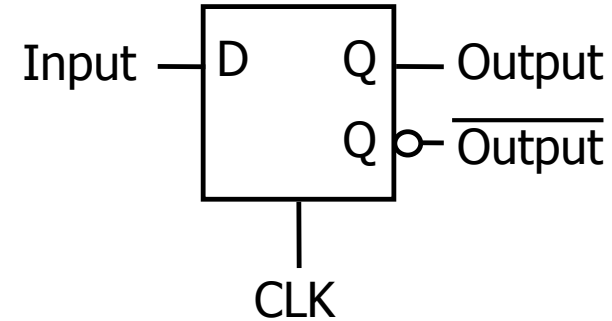
The D latch: store it and look it up

- Output depends on clock
 - Clock high: Input passes to output
 - Clock low: Latch holds its output
- Latches are level sensitive and “transparent”



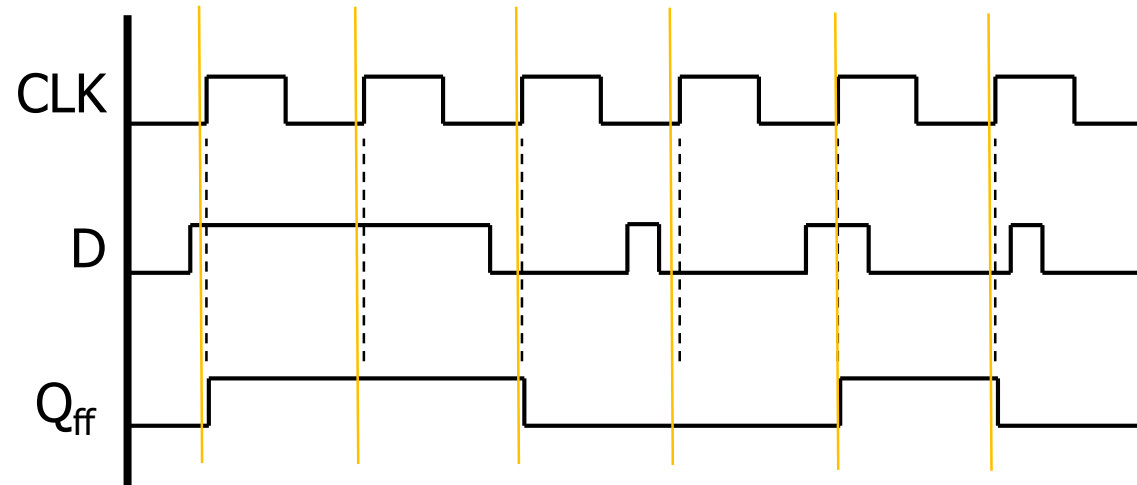
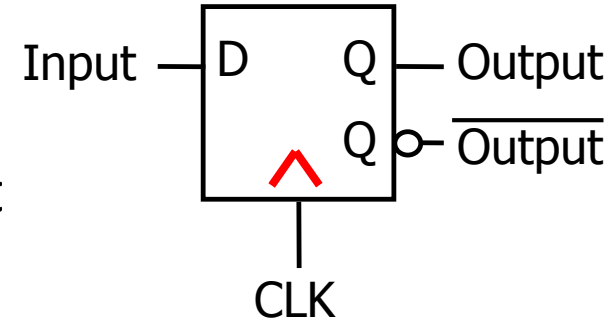
The D latch: store it and look it up

- Output depends on clock
 - Clock high: Input passes to output
 - Clock low: Latch holds its output
- Latches are level sensitive and “transparent”

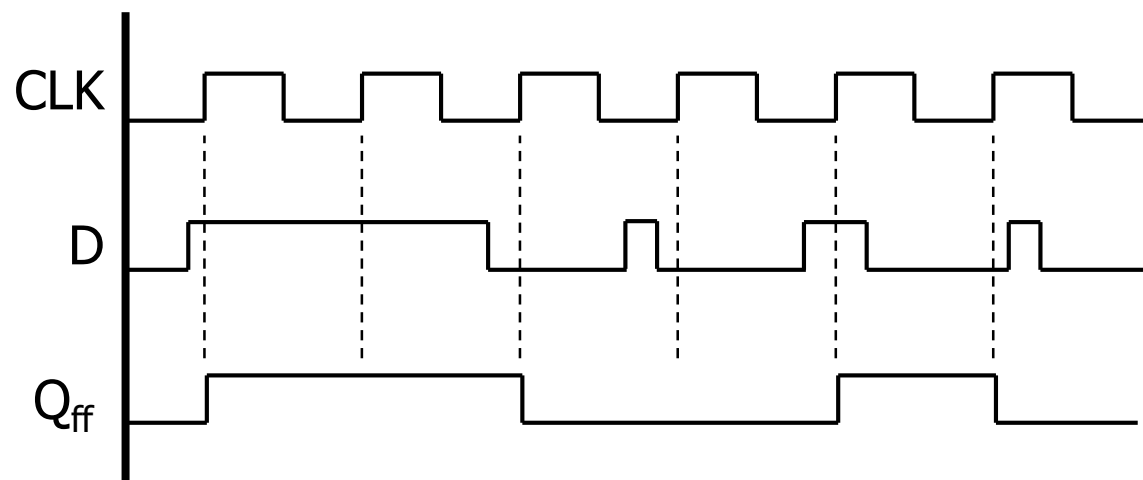
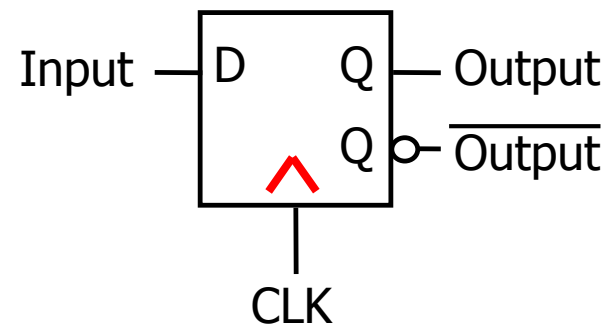


The D flip-flop

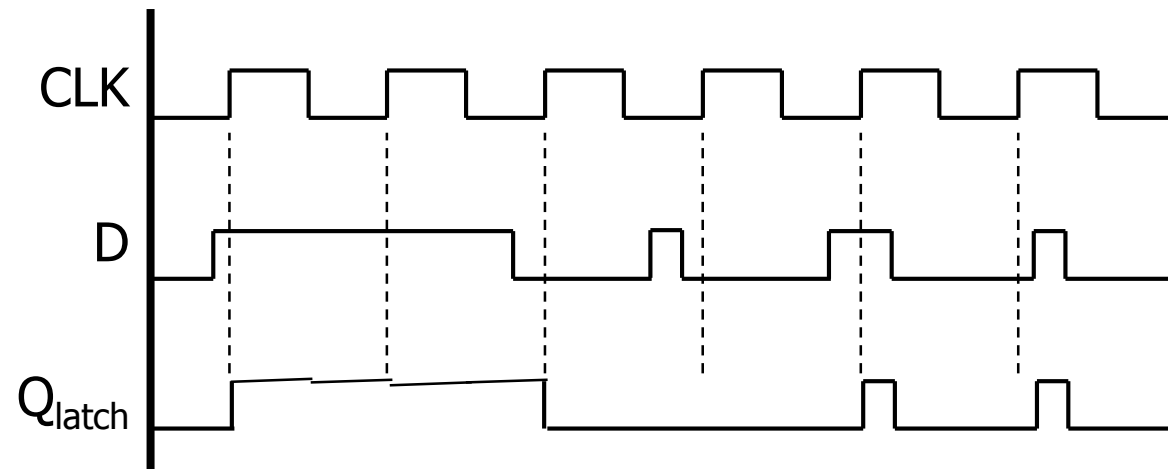
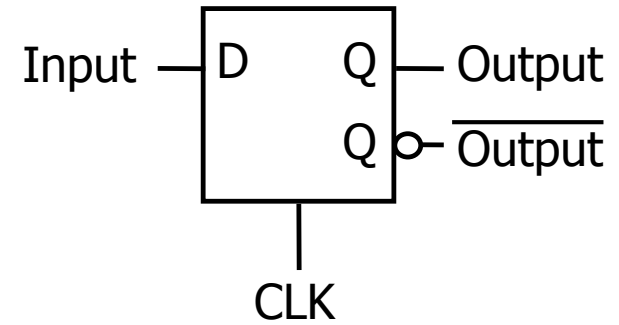
- Input sampled at clock edge
 - Rising edge: Input passes to output
 - Otherwise: Flip-flop holds its output
- Flip-flops can be rising-edge triggered or falling-edge triggered



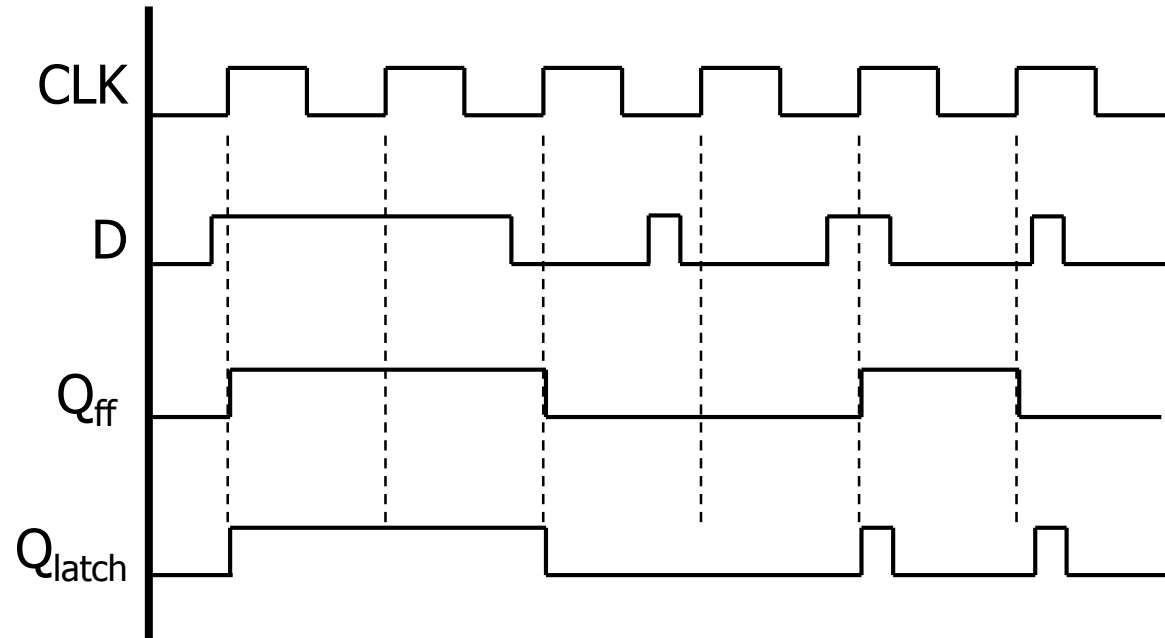
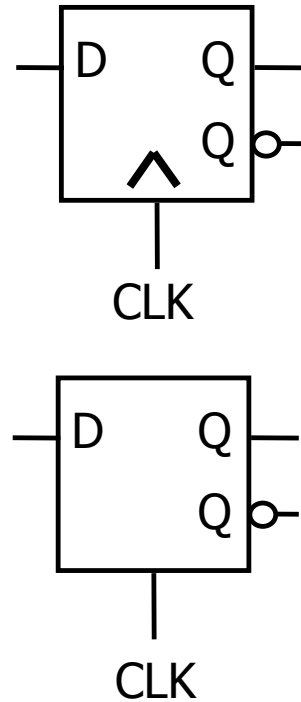
The D flip-flop



The D latch

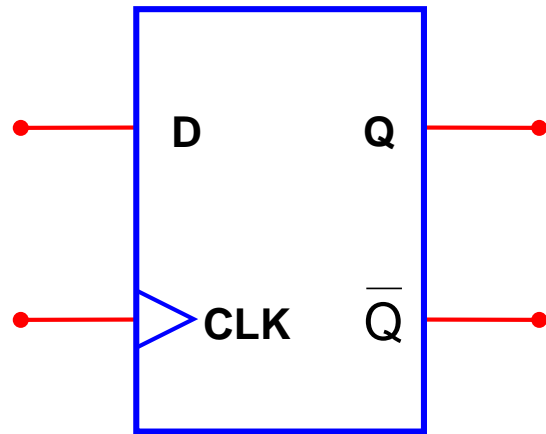


Latches versus flip-flops



behavior is the same **unless** input changes while the clock is high

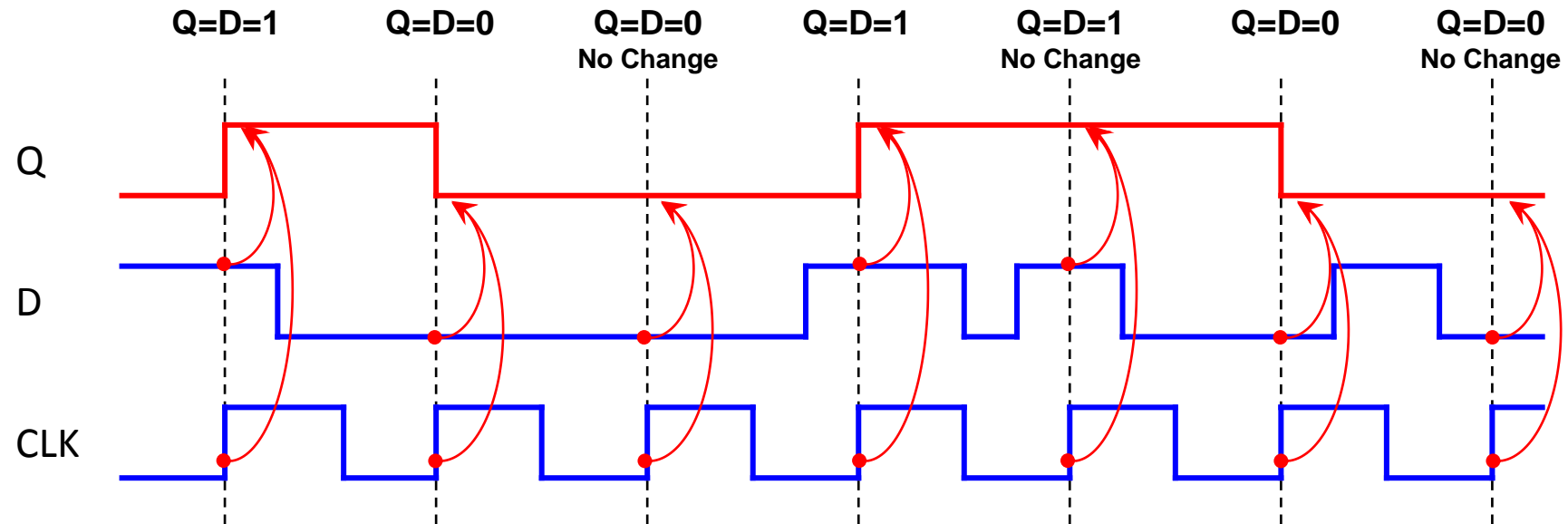
D Flip-Flop: Excitation Table



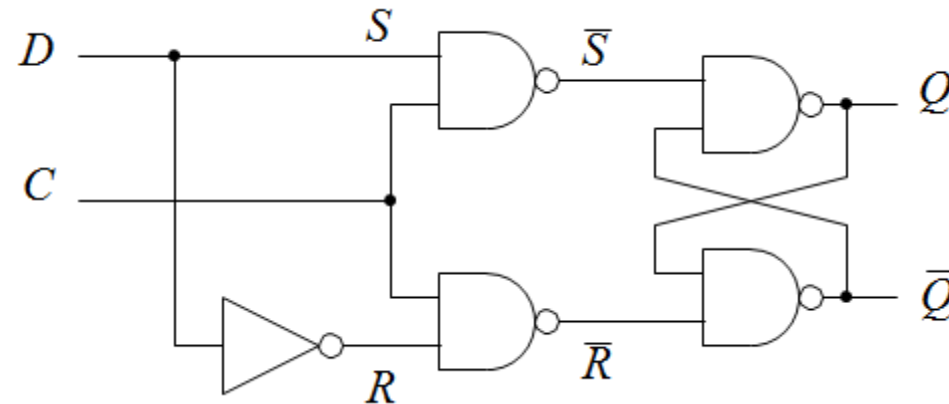
D	CLK	Q	\overline{Q}
0	↑	0	1
1	↑	1	0

↑ : Rising Edge of Clock

D Flip-Flop: Example Timing

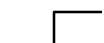
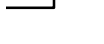




D Flip-Flop



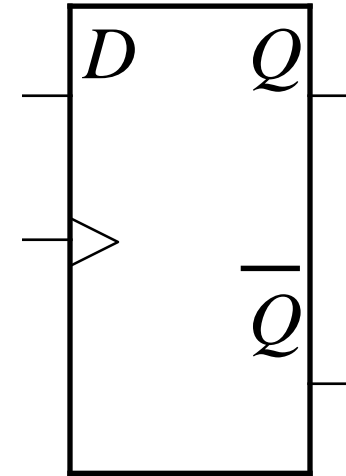
- The problem with $S = R = 1$ can be avoided using a common input D as shown above so that $S = \bar{R}$.
- The output of the latch now:
 - **follows the D input** while $C = 1$ (transparent)
 - **holds its value** while $C = 0$ (Q = last Q when C went low) no matter what happens at the input
- This circuit is often called a **transparent latch**. It can be bought as an integrated circuit, usually with several latches in a package.
 - The input C may be called **control**, **clock**, **gate**, or **enable**.

D Flip-Flop

D	C	Q_n	Q_{n+1}	description
0		0	0	Clear (reset)
0		1	0	
1		0	1	Set
1		1	1	

input output output
at before after
clock clock clock

$$Q_{n+1} = D$$



A D flip-flop simply stores the value on its D input at the clock transition.

The previous value stored, Q_n , has no effect, unlike other flip-flops.

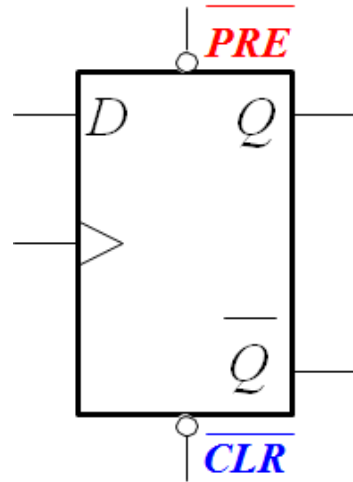
It therefore acts as a simple memory or 'latch'.

The most widely used flip-flops: simple to build and design with.

A **register** comprises several D flip-flops, one for each bit to be stored.

Control Pins

Flip-flops and more complicated circuits often have inputs, such as **clear**, **preset**, **enable**, and **load**. The state of the control pins has priority of the D and Clock inputs when determining the state of the output.

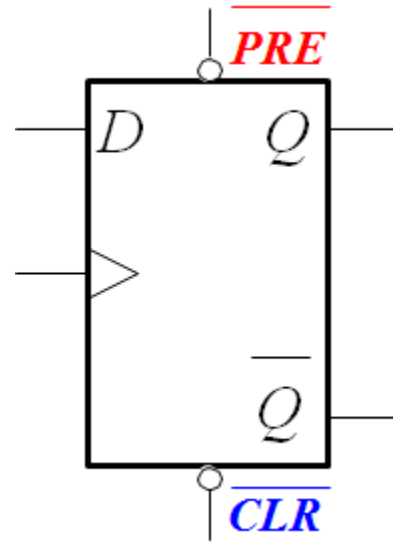


- **Clear** (CLR) resets the flip-flop output to 0 — the most common control input
- **Preset** (PRE) sets the flip-flop output to 1

More complicated circuits such as counters may have additional control inputs for up/down, count/hold, load, *etc.* Microprocessors usually have a reset pin.

Active Low Controls

Control inputs are often **active low**, shown by a bar over the label or a circle for negation (or both as in the component below).



Active low inputs should be:

- **kept high for normal operation**
- changed to low to **preset** or **clear** the device.

The reason for making these active low is historical.

Check the data sheet to be sure!

Asynchronous Control

Another feature of control inputs is that they are often **asynchronous**. This means that they take effect **immediately** and do not wait for a clock transition.

Compare

- **D** takes effect **only at a clock transition** (positive edge)
- **clear** and **preset** act **immediately** to 'overrule' **D**

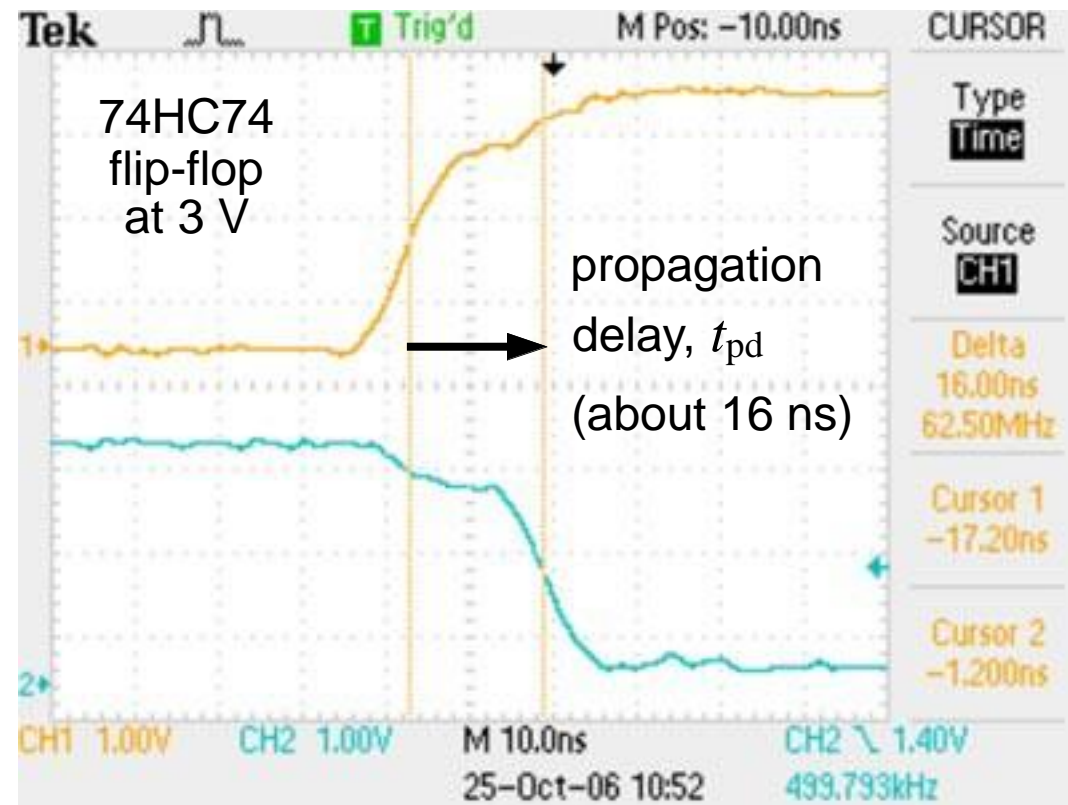
Asynchronous inputs are sometimes called **direct** or **jam** inputs.

Always check the data sheet because some control inputs are synchronous!

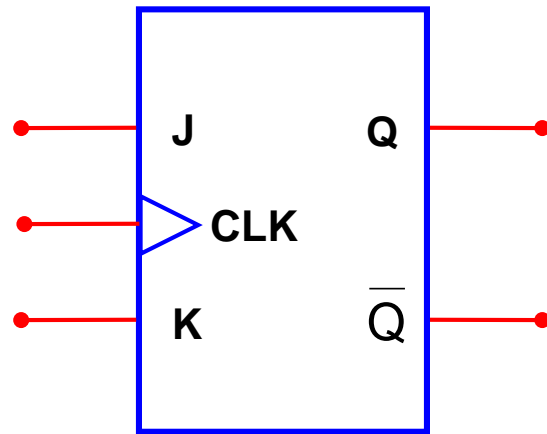
Propagation Delay

We have seen that most modern logic devices are triggered on either the rising or falling edge of the clock. The output does not respond instantly, but only after a time called the **propagation delay**, t_{pd} .

Here is an example for a *D* flip-flop that was measured in a UoG lab class.



J/K Flip-Flop: Excitation Table



J	K	CLK	Q
0	0	↑	Q_0
0	1	↑	0
1	0	↑	1
1	1	↑	$\overline{Q_0}$

No Change

Clear

Set

Toggle

↑ : Rising Edge of Clock

\overline{Q} : Complement of Q

K-MAPS-J-K Latch

- Characteristic table and equation
 - Karnaugh map of characteristic table
 - Characteristic equation
 - $Q^+ = JQ' + K'Q$

Q	J	K	$Q(t+1)$
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

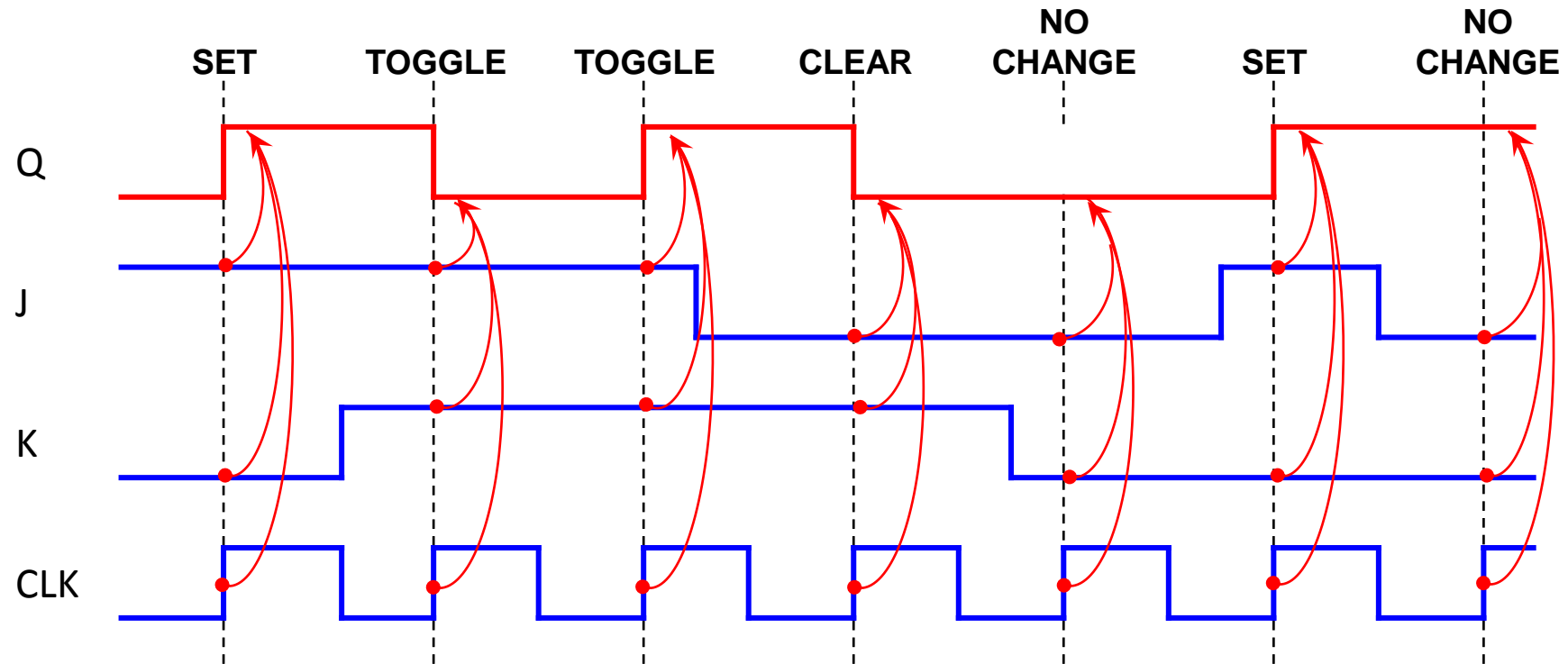
(b) Characteristic table

	JK		J
	00	01	11 10
Q			
0			1 1
1	1		1
		K	

$$Q(t+1) = JQ' + K'Q$$

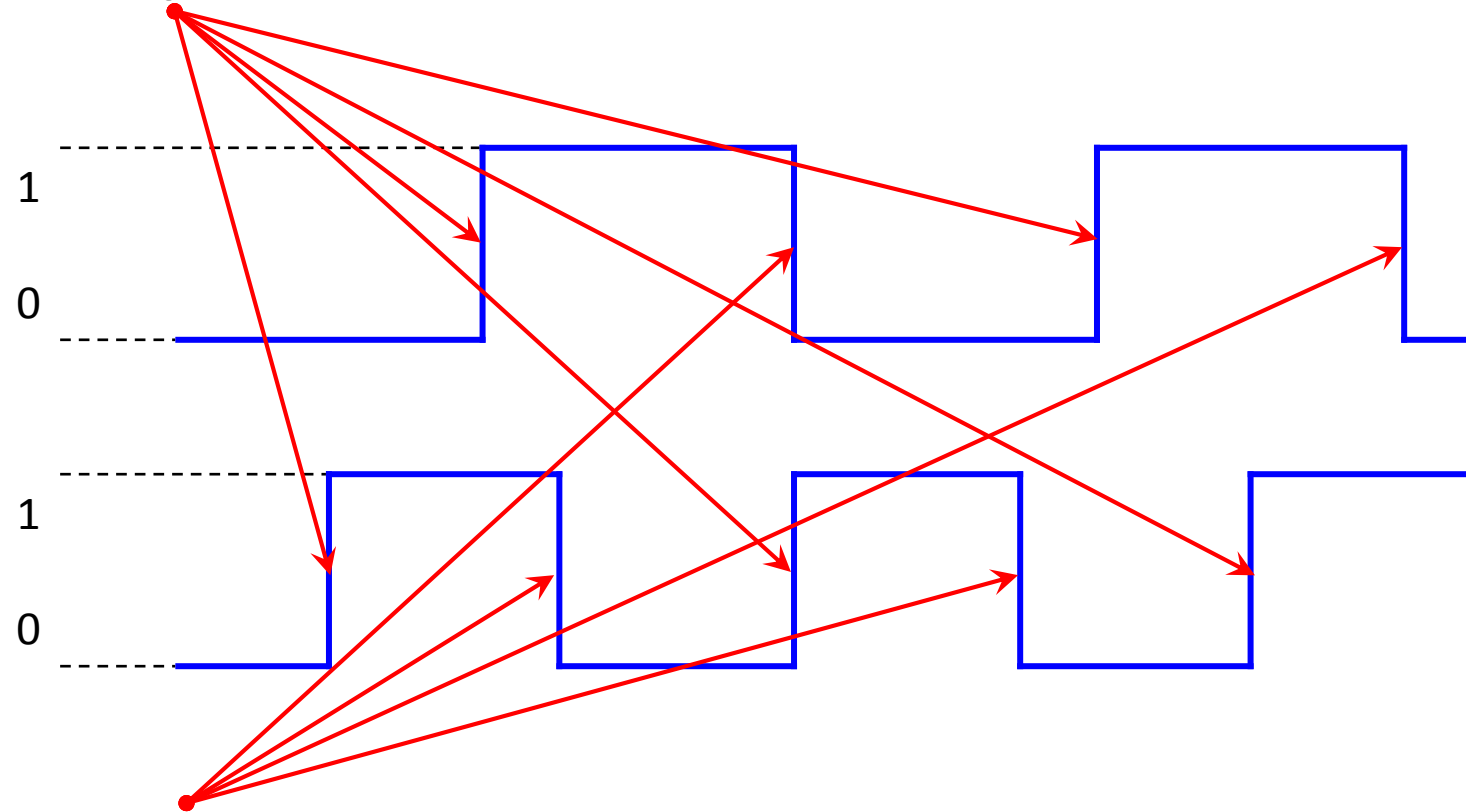
(c) Characteristic equation

J/K Flip-Flop: Example Timing



Clock Edges

Positive Edge Transition



Negative Edge Transition

Edge-Triggered Flip-Flops

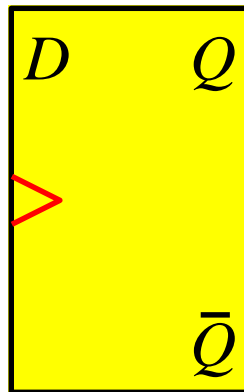
These circuits respond to their inputs on either the **rising** or **falling** edge of the clock — a precise point in time rather than an interval.

Positive edge triggered



rising edge of clock

wedge shows **positive** edge triggering

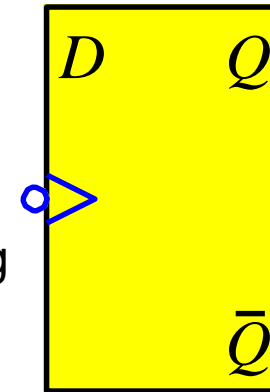


Negative edge triggered



falling edge of clock

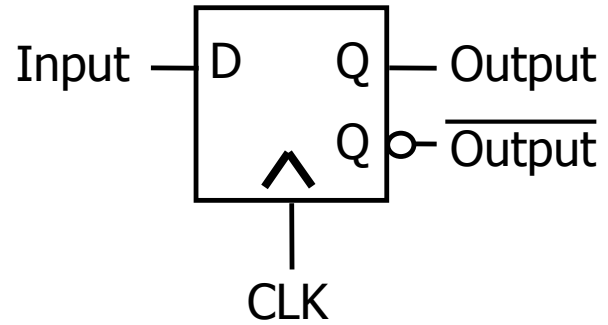
additional of a circle means that there is **negative** edge triggering



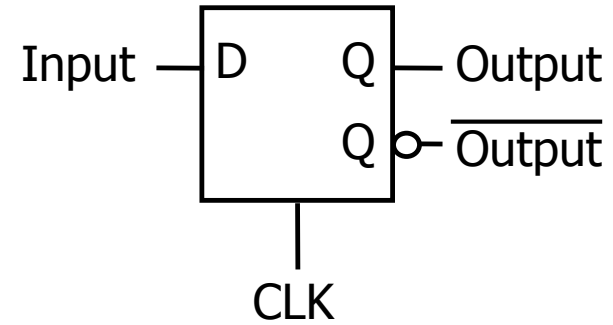
Older flip-flops may be 'pulse-triggered', which require a clock pulse that goes from $0 \rightarrow 1 \rightarrow 0$ or a 'master-slave' types but these are now obsolete.

Terminology & notation

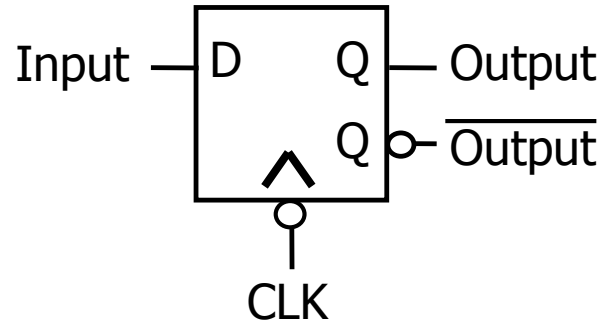
Rising-edge triggered
D flip-flop



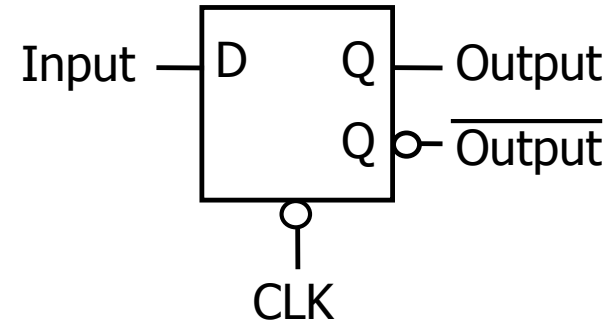
Positive D latch



Falling-edge triggered
D flip-flop

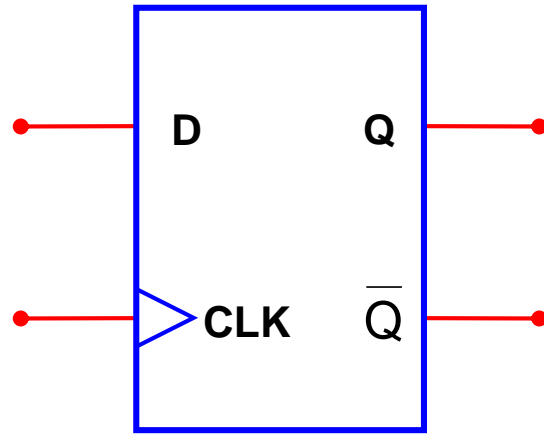


Negative D latch



POS & NEG Edge Triggered D

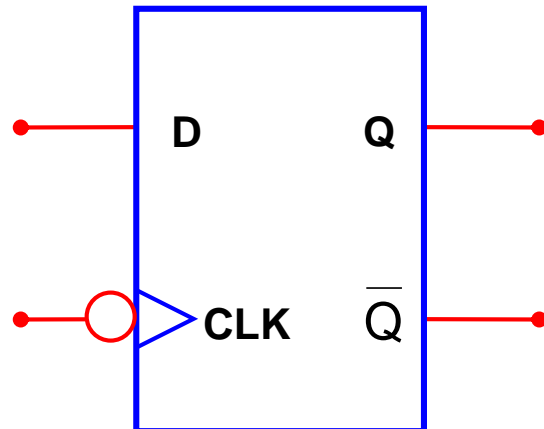
Positive Edge Trigger



D	CLK	Q	\overline{Q}
0	↑	0	1
1	↑	1	0

↑ : Rising Edge of Clock

Negative Edge Trigger

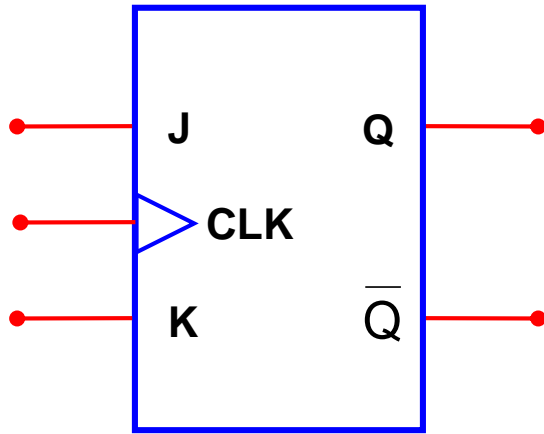


D	CLK	Q	\overline{Q}
0	↓	0	1
1	↓	1	0

↓ : Falling Edge of Clock

POS & NEG Edge Triggered J/K

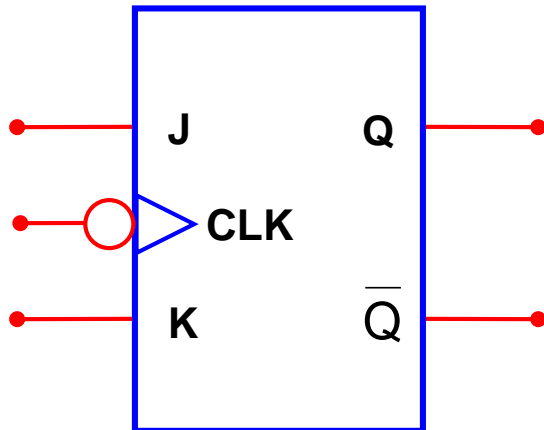
Positive Edge Trigger



J	K	CLK	Q
0	0	↑	Q_0
0	1	↑	0
1	0	↑	1
1	1	↑	$\overline{Q_0}$

↑ : Rising Edge of Clock

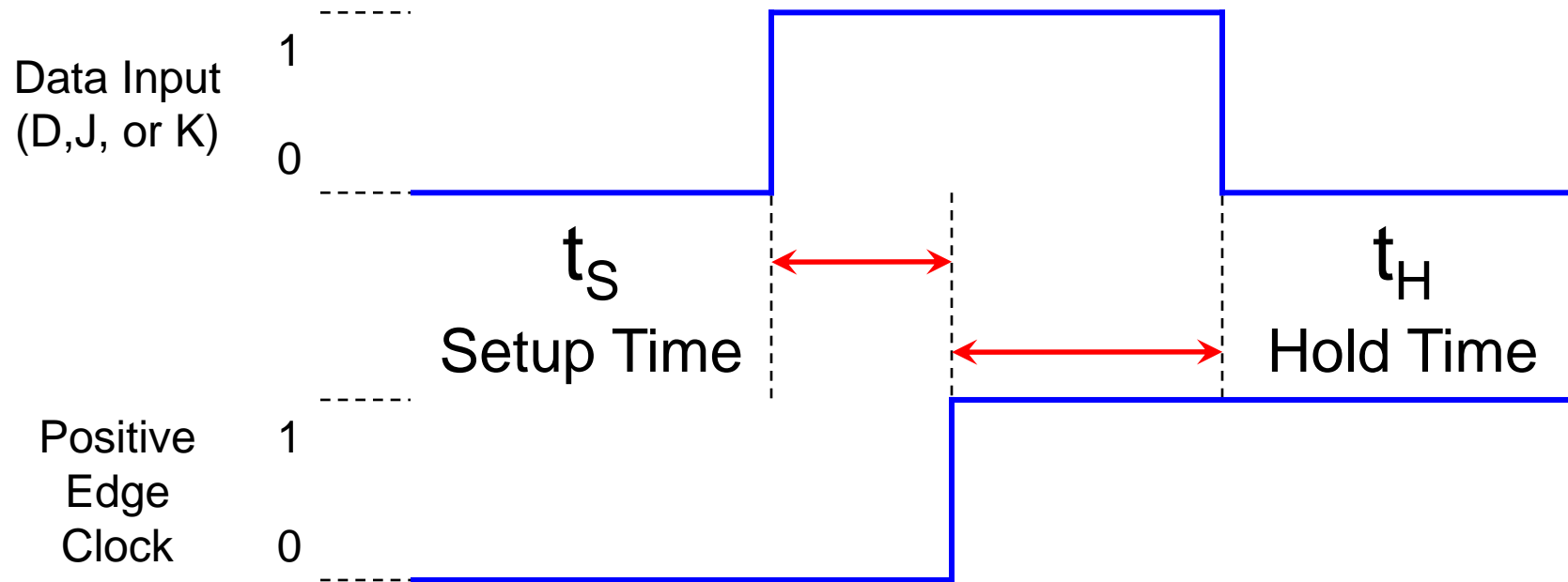
Negative Edge Trigger



J	K	CLK	Q
0	0	↓	Q_0
0	1	↓	0
1	0	↓	1
1	1	↓	$\overline{Q_0}$

↓ : Rising Edge of Clock

Flip-Flop Timing



Setup Time (t_s): The time interval before the active transition of the clock signal during which the data input (D, J, or K) must be maintained.

Hold Time (t_H): The time interval after the active transition of the clock signal during which the data input (D, J, or K) must be maintained.

Excitation Tables

Logic tables show the state of the output(s) of a logic circuit as a function of its inputs **at the same time**.

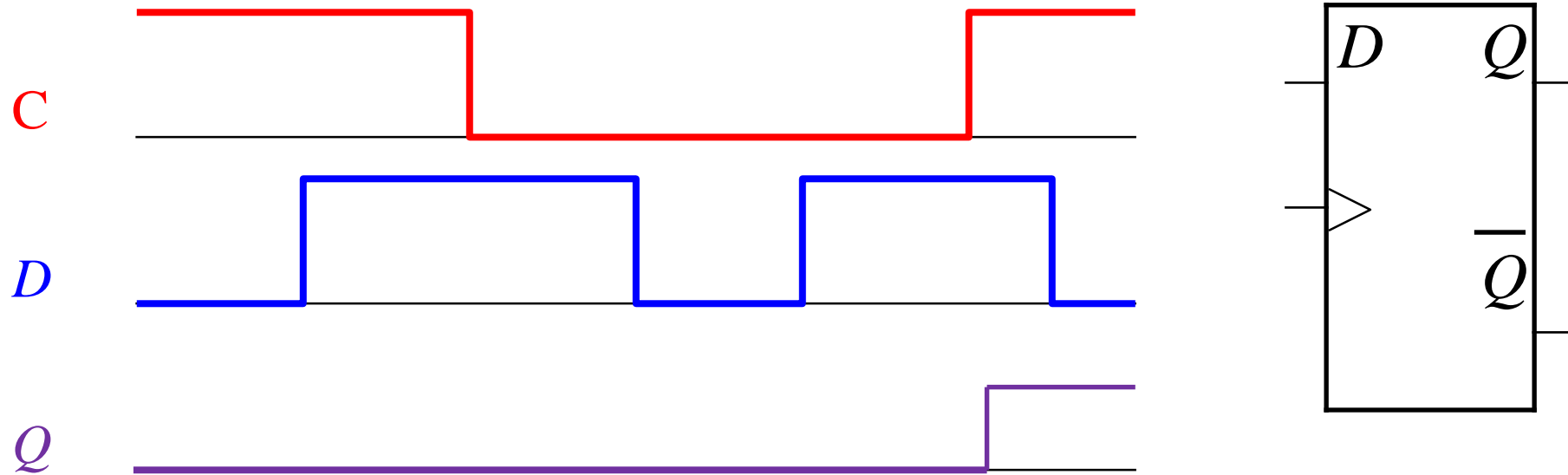
Since, clocked digital systems have **memory**, their behaviour depends on inputs **in the past** as well as the present values of the inputs.

Thus, flip-flops cannot be described by simple truth tables. Instead, we use **excitation** or **transition tables**. These show:

- **output before** the clock transition — often labelled Q_n
- **inputs at** the clock transition — such as S and R
- occasionally the type of clock transition – positive/negative edge-triggered
- the resulting **output after** the clock transition — often labelled Q_{n+1}

It is important to remember that Q_n and Q_{n+1} describe the **same** signal but at **different times**. The notation can vary, e.g. Q_0 and Q instead.

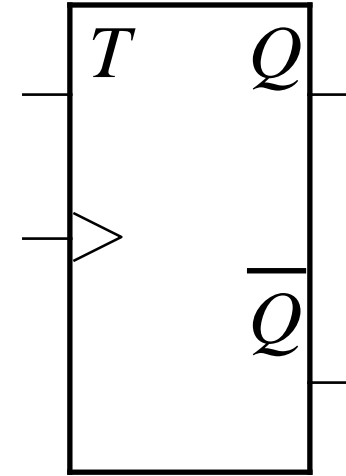
Timing Diagram: Edge-Triggered FF



The input D can change at any time because it comes from other parts of the system — it is not necessarily synchronized to the clock (it may be from a switch on the front panel, for instance). However, the flip-flop only changes its output when the clock pulse rises.

Toggle (T) Flip-Flop

T	Q_n	Q_{n+1}	description
0	0	0	hold
0	1	1	
1	0	1	toggle
1	1	0	



$$Q_{n+1} = T \oplus Q_n = T \oplus \overline{Q_n} + \overline{T} \oplus Q_n$$

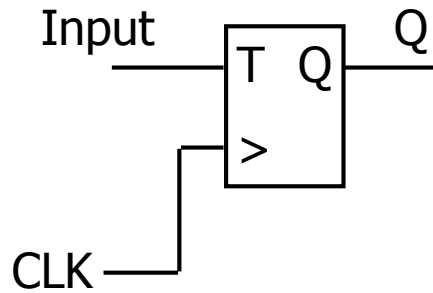
Note that the output depends on the previous value stored, Q_n .

This type of flip-flop is rarely bought as a ‘dedicated’ device — you can easily make a T from a D flip-flop.

Aside: it is not possible to put a specific value into a T flip-flop – it can only toggle or hold.

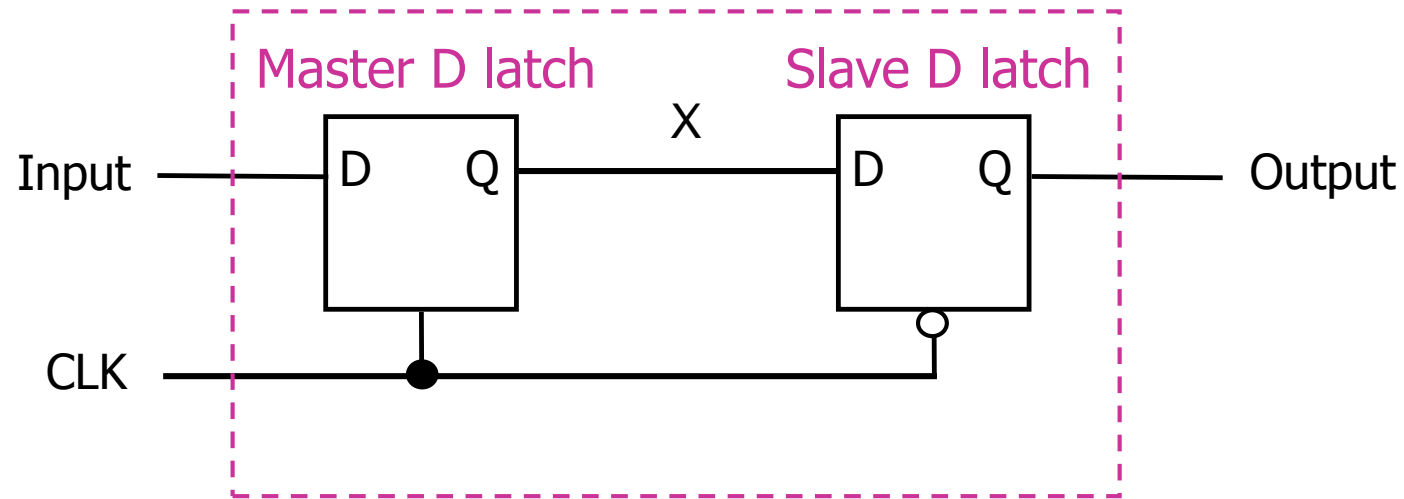
T flip-flop

- Full name: Toggle flip-flop
- Output toggles when input is asserted
 - If $T=1$, then $Q \rightarrow Q'$ when $\text{CLK} \uparrow$
 - If $T=0$, then $Q \rightarrow Q$ when $\text{CLK} \uparrow$

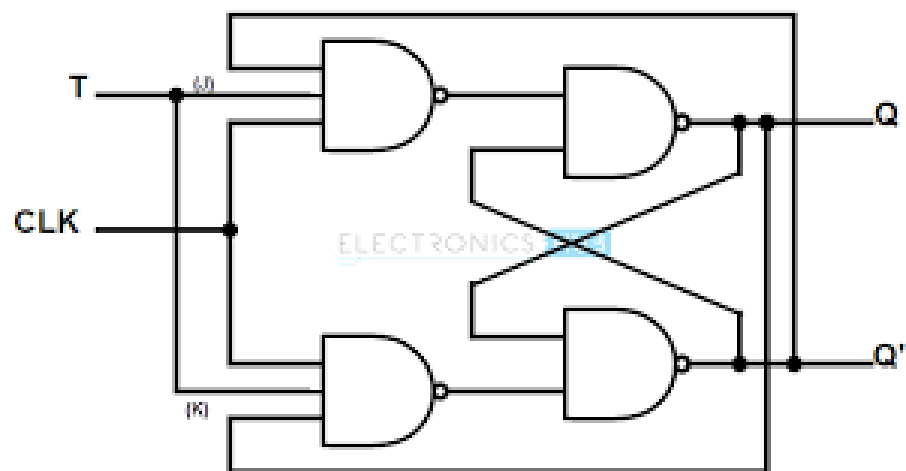
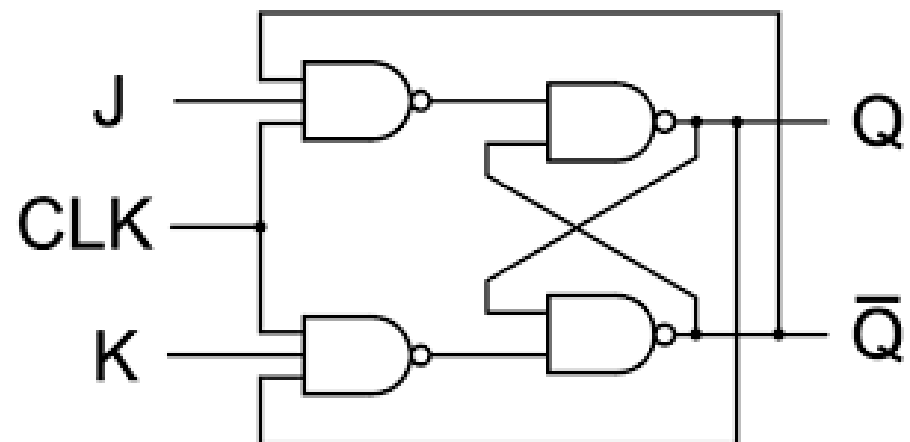


Input(t)	Q(t)	Q($t + \Delta t$)
0	0	0
0	1	1
1	0	1
1	1	0

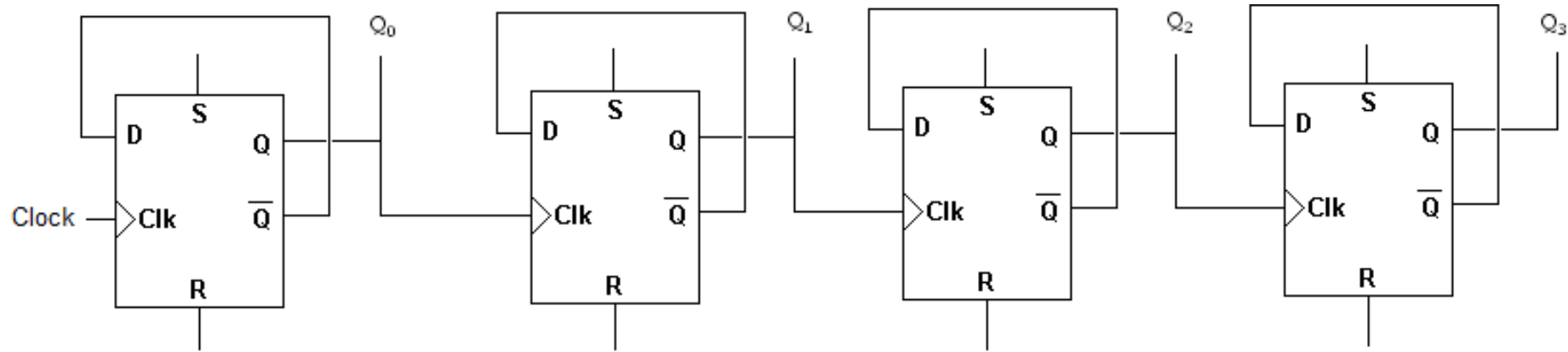
The master-slave D



- Full name: Master Slave flip-flop
- Master Output changes at positive EDGE of Clock
- Slave Output changes at Negative EDGE of Clock

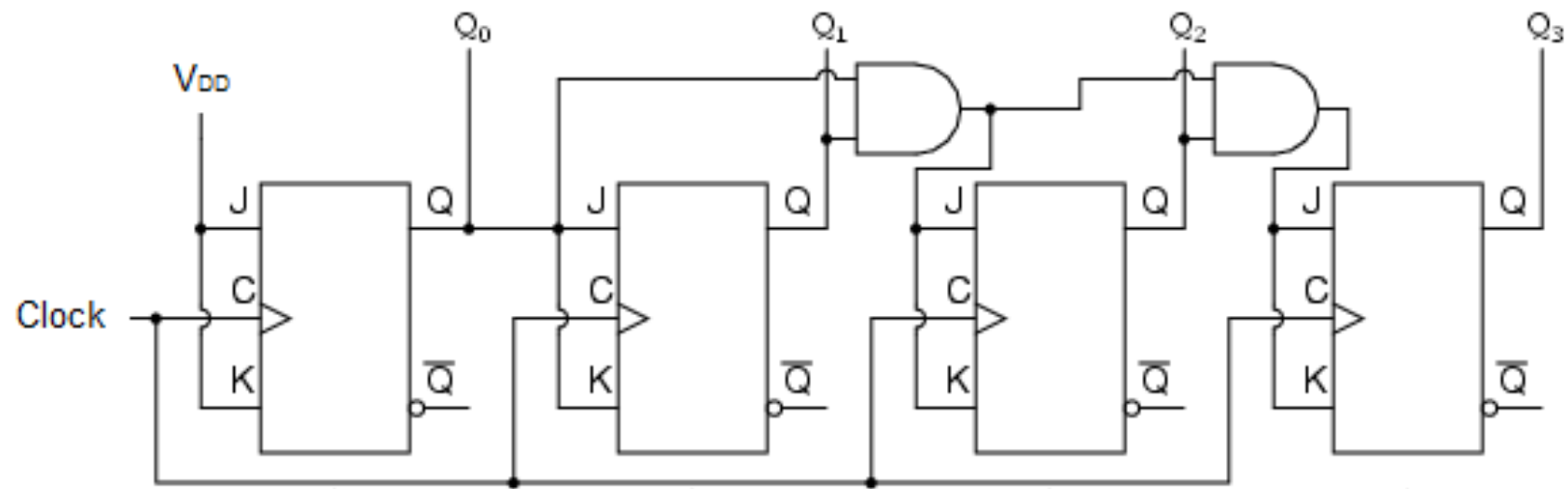


What does this circuit do?



Assume that the initial state of Q_0 , Q_1 , Q_2 , and Q_3 are all logical zeros.

1. Determine how Q_0 changes as the clock pulse goes from $0 \rightarrow 1 \rightarrow 0 \rightarrow 1$.
2. After how many clock pulses will Q_1 change state from '0' to '1'?
3. After how many clock pulses will Q_2 change state from '0' to '1'?
4. After how many clock pulses will Q_3 change state from '0' to '1'?



Task: Prove this to yourself.

Limitations on Circuit: t_{pd}

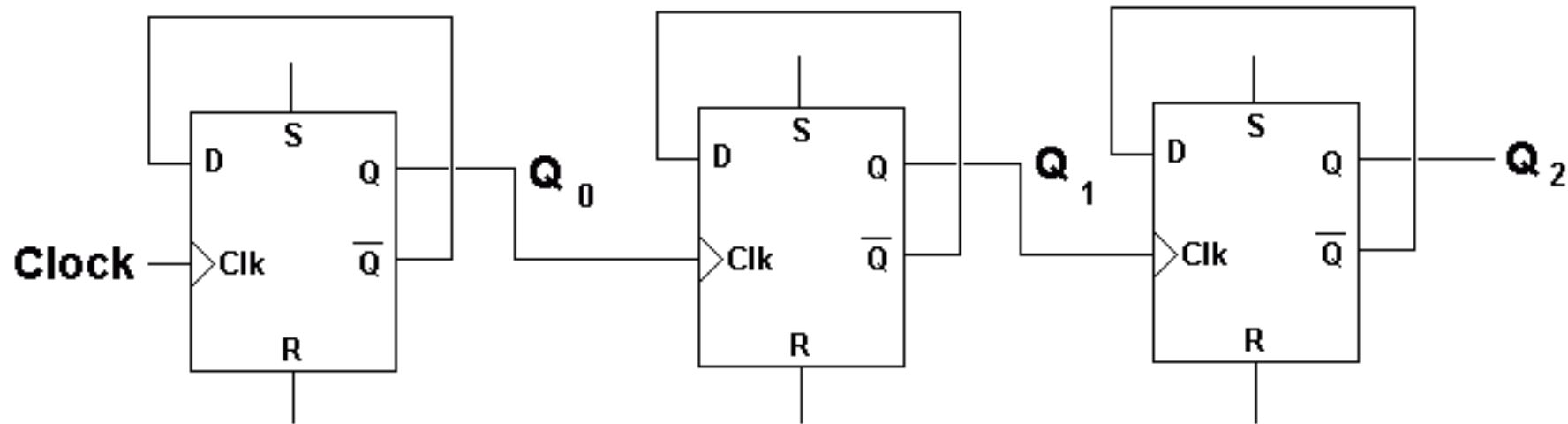
The propagation delay is important for several reasons:

- It limits the speed at which circuits can be clocked (20 or 30 MHz for the 'HC' family of components, used in the laboratory)
- Signals that pass through different numbers of components receive different delays, as in a ripple counter. Time must be allowed for all outputs to settle down before the system attempts to change state again.
- The delay helps to keep digital circuits with feedback stable (this applies to virtually all practical circuits).
 - Each logic gate responds to its at the clock transition.
 - Because of t_{pd} , the outputs change after the propagation delay.
 - This affects the inputs that are connected to outputs.
 - However, these gates are no longer acting on their inputs until the next clock transition arrives.

Would this circuit work?

What would happen if t_{pd} is longer than the period of the clock?

What would happen if t_{pd} is about equal to the clock's period?



Clear and preset in flip-flops

- **Clear** and **Preset** set flip-flop to a known state
 - Used at startup, reset
- **Clear** or **Reset** to a logic 0
 - Synchronous: $Q=0$ when next clock edge arrives
 - Asynchronous: $Q=0$ when reset is asserted
 - Doesn't wait for clock
 - Quick but dangerous
- **Preset** or **Set** the state to logic 1
 - Synchronous: $Q=1$ when next clock edge arrives
 - Asynchronous: $Q=1$ when reset is asserted
 - Doesn't wait for clock
 - Quick but dangerous

BASIS FOR COMPARISON	REGISTER	MEMORY
Basic	Registers hold the operands or instruction that CPU is currently processing.	Memory holds the instructions and the data that the currently executing program in CPU requires.
Capacity	Register holds the small amount of data around 32-bits to 64-bits.	Memory of the computer can range from some GB to TB.
Access	CPU can operate on register contents at the rate of more than one operation in one clock cycle.	CPU accesses memory at the slower rate than register.
Type	Accumulator register, Program counter, Instruction register, Address register, etc.	RAM.