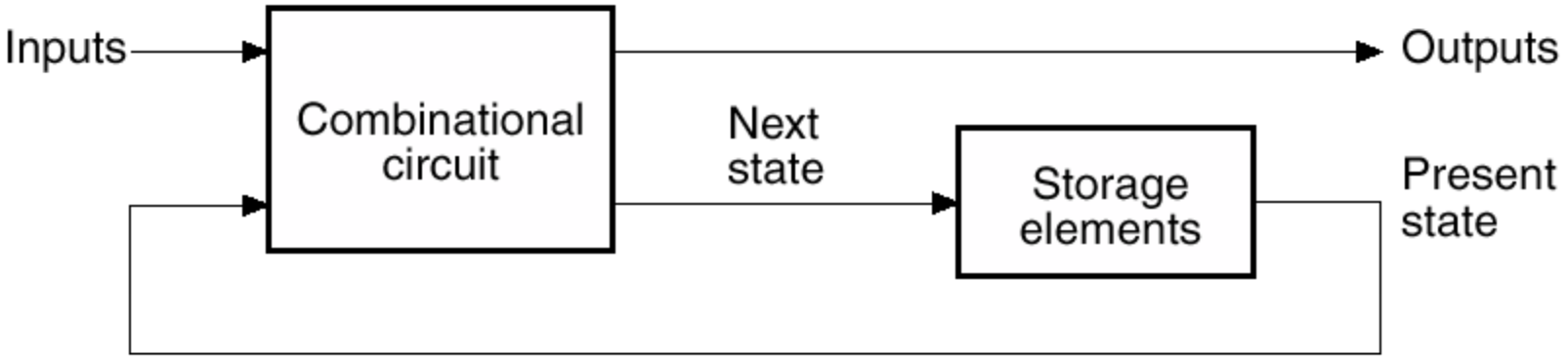
SEE 3243/4243
Digital System

Week 7: Sequential Circuits & Flip-Flops —

- ***Basic Latch***
- ***Gated SR & D Latches***
- ***D, T & JK Flip-Flops***
- ***Metastability***

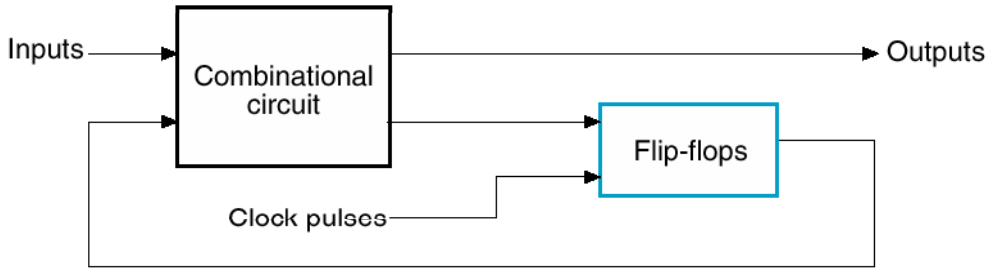
Sequential Circuits

- Combinational – output depends only on the input.
  - Do not have memory
  - Cannot store state
- Sequential – output depends on input and past behavior.
  - Require use of storage elements.
  - Contents of storage elements is called state.
  - Circuit goes through sequence of states as a result of changes in inputs.

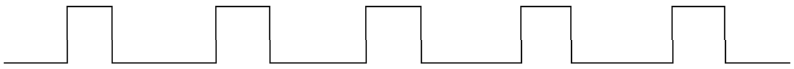


Sequential Circuits Types

- Synchronous
  - State changes synchronized by one or more clocks
  - Easier to analyze because can factor out gate delays
  - Set clock so changes allowed to occur before next clock pulse
- Asynchronous
  - Changes occur independently
  - Potentially faster
  - Harder to analyze

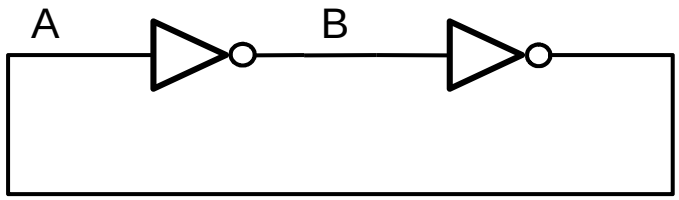


(a) Block diagram

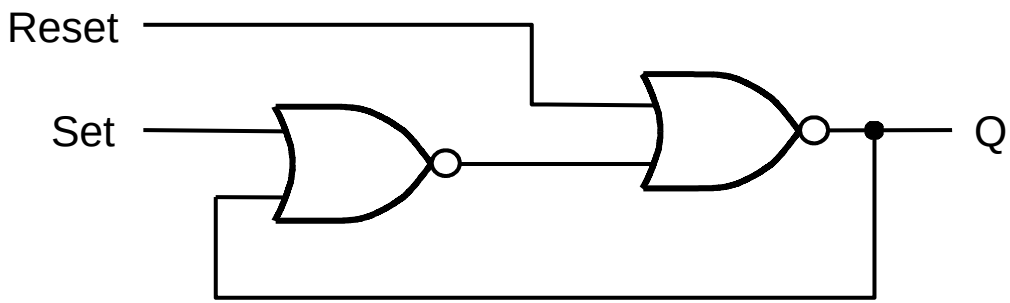


(b) Timing diagram of clock pulses

Simple Memory Elements



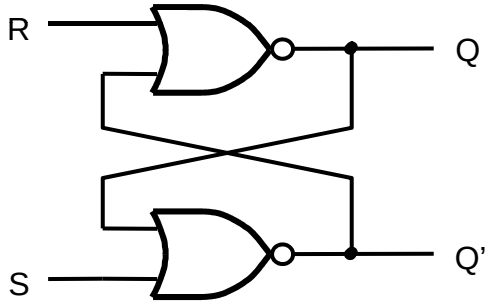
A simple memory element:
feedback will hold value



A memory element with NOR gates:
Use Set/Reset to change stored value

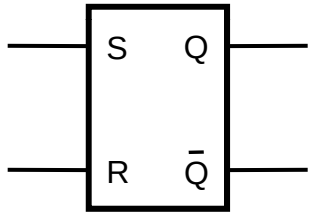
SR Latch

- Basic storage made from gates
- Rearrangement of memory element from previous slide!



Function Table

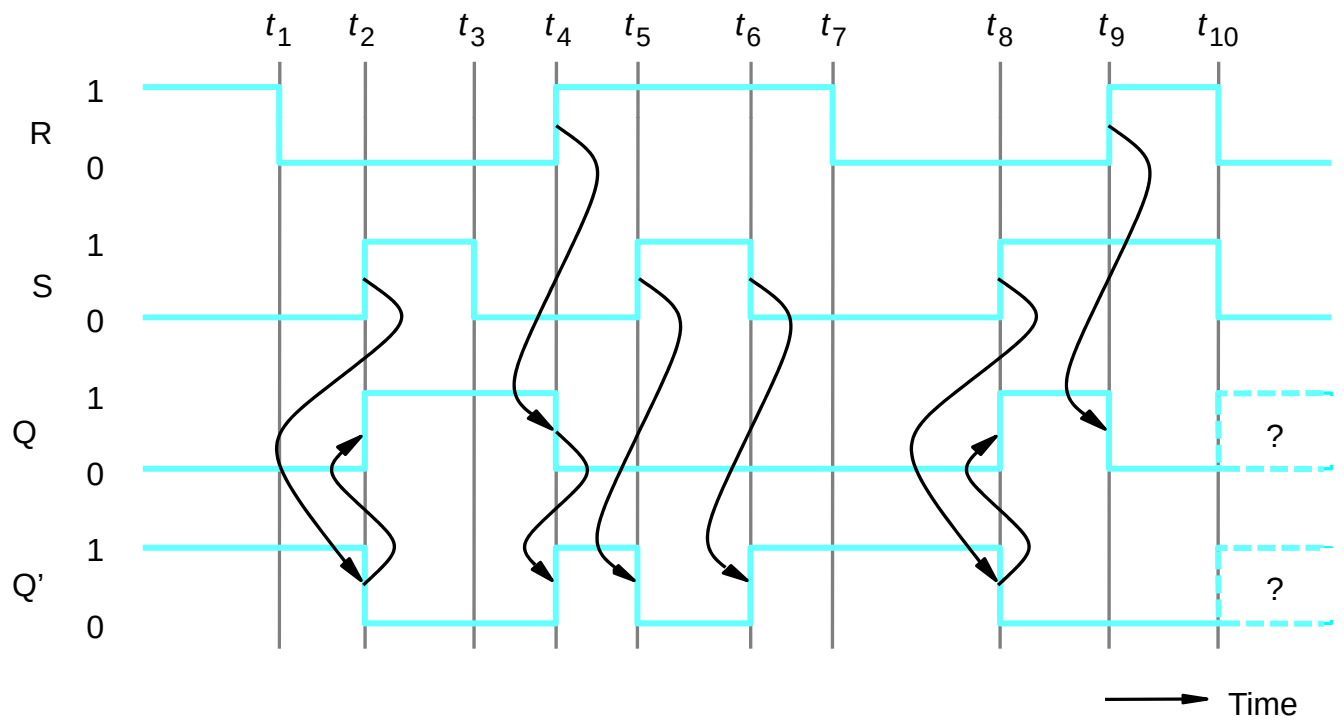
S	R	Q(t+1)	Function
0	0	Q(t)	Hold
0	1	0	Reset
1	0	1	Set
1	1	?	Not allowed



Graphical symbol

- If S & R both 1 at same time, Q = Q' = 1

SR Latch



SR Latch

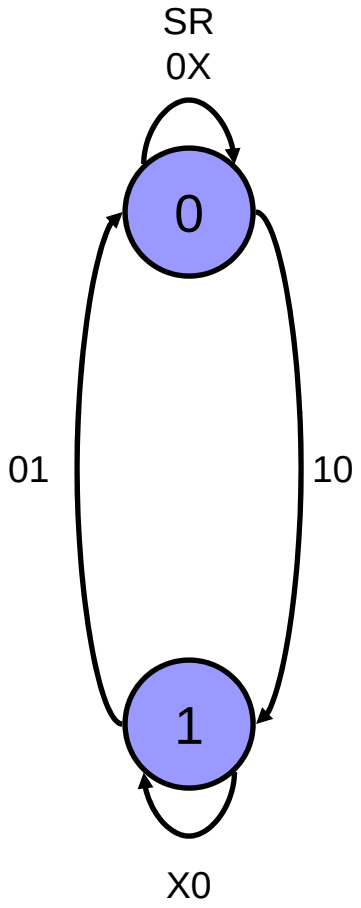
Detailed Function Table			
S	R	Q	Q+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	X
1	1	1	X

SR / Q	00	01	11	10
0	0	0	X	1
1	1	0	X	1

Characteristic Equation:

$Q+ = S + R'Q$

Excitation Table			
Q	Q+	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

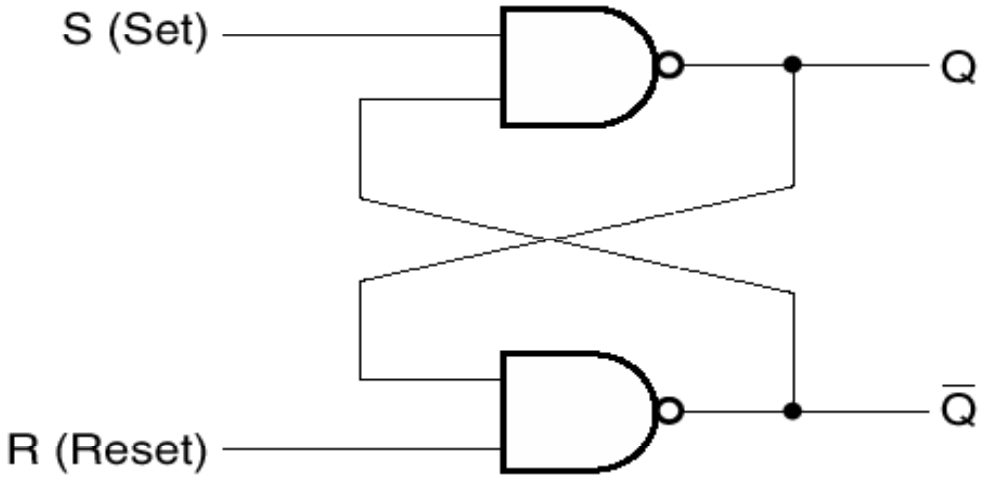


State Transition Diagram: The excitation table in graphical form

Excitation Table: What are the necessary inputs to cause a particular kind of change in state?

$\overline{S}\,\overline{R}$ Latch

- Similar – made from NANDs



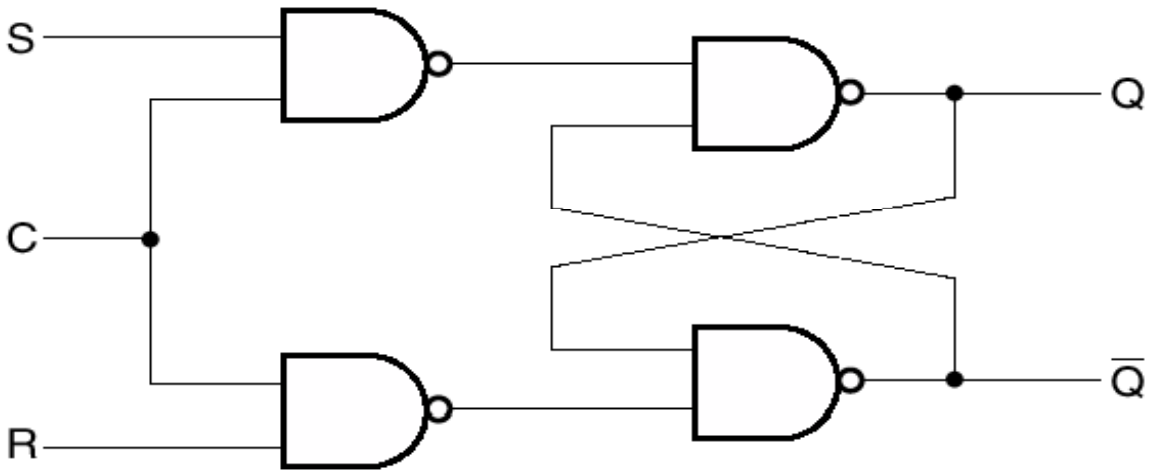
(a) Logic diagram

S	R	Q	$\overline{Q}$	
0	1	1	0	Set state
1	1	1	0	
1	0	0	1	Reset state
1	1	0	1	
0	0	1	1	Undefined

(b) Function table

Gated SR Latch

- Add Control Input
  - Typically, control signal is referred to as a clock
- Clock controls when state can change

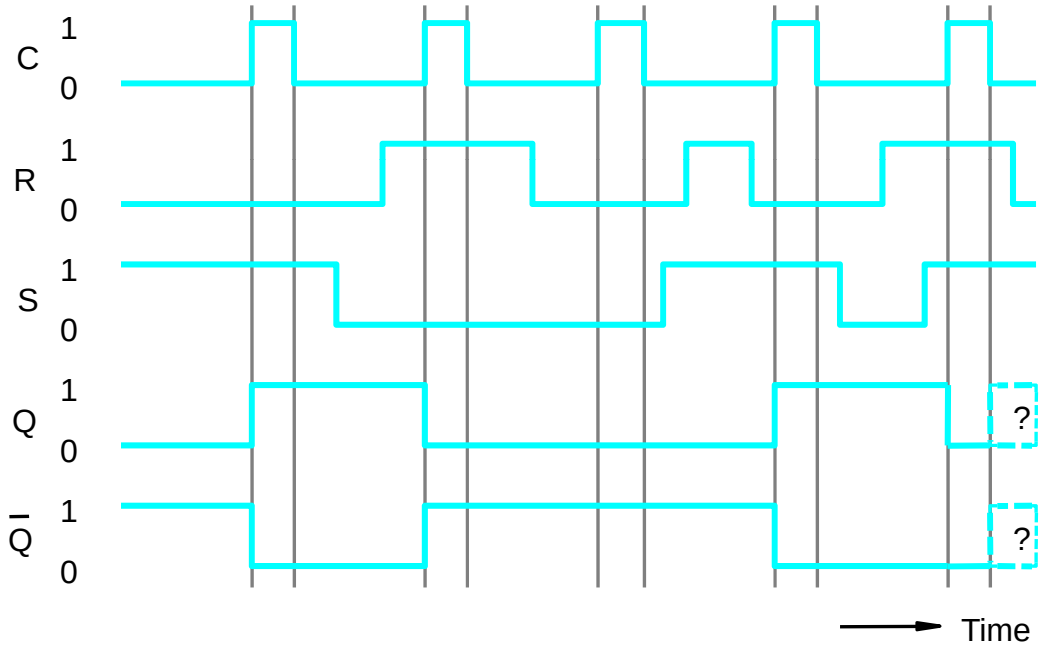


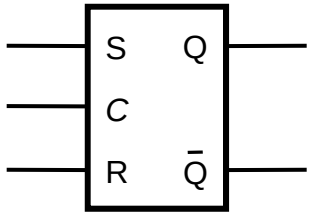
(a) Logic diagram

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	Undefined

(b) Function table

Gated SR Latch

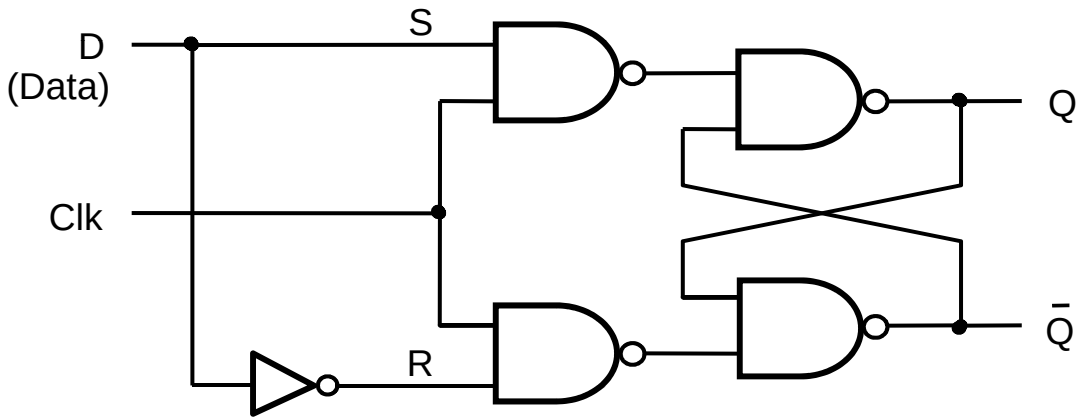




Graphical symbol

Gated D Latch

- No illegal state

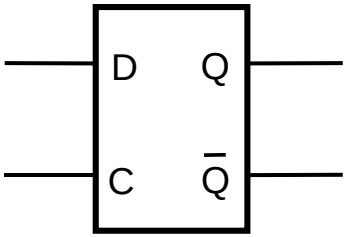


(a) Circuit

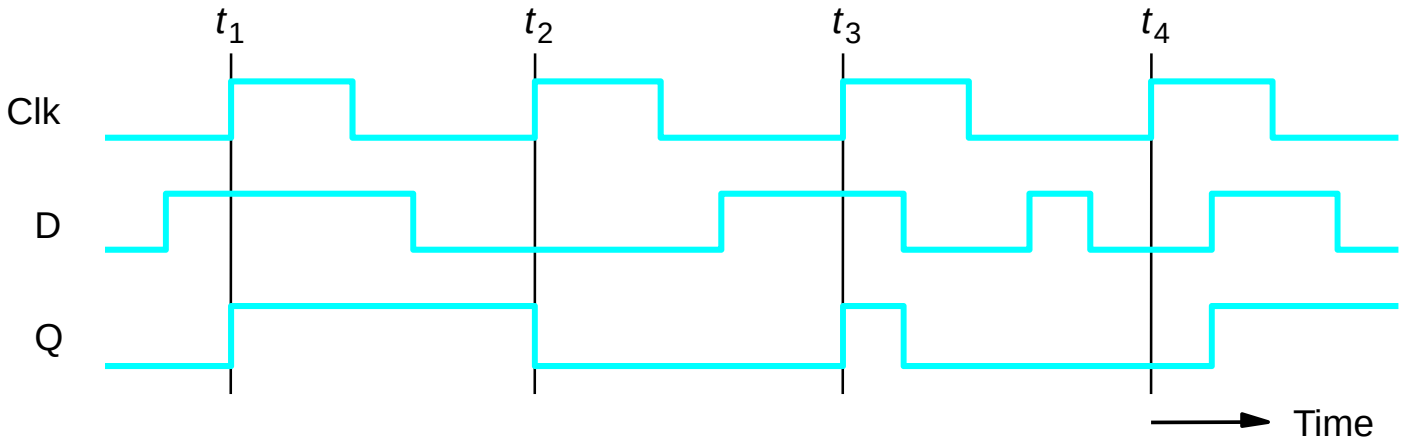
C	D	Next state of Q
0	X	No change
1	0	Q = 0; Reset state
1	1	Q = 1; Set state

(b) Function table

Gated D Latch



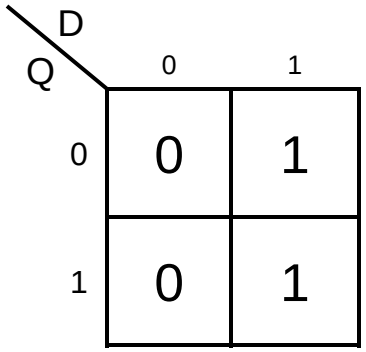
(c) Graphical symbol



(d) Timing diagram

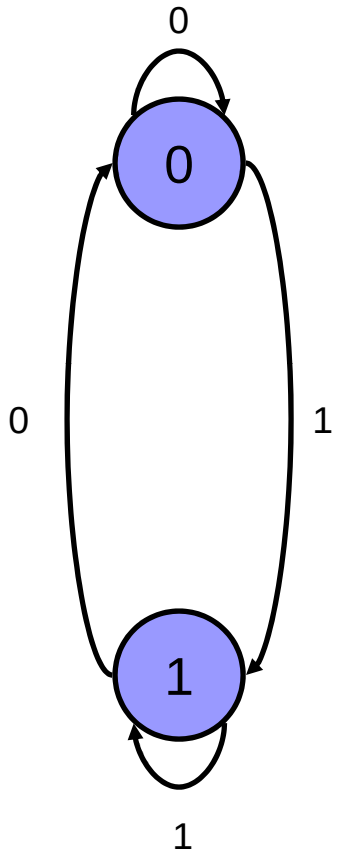
D Latch

Detailed Function Table		
D	Q	Q+
0	0	0
0	1	0
1	0	1
1	1	1



Characteristic Equation
Q+ = D

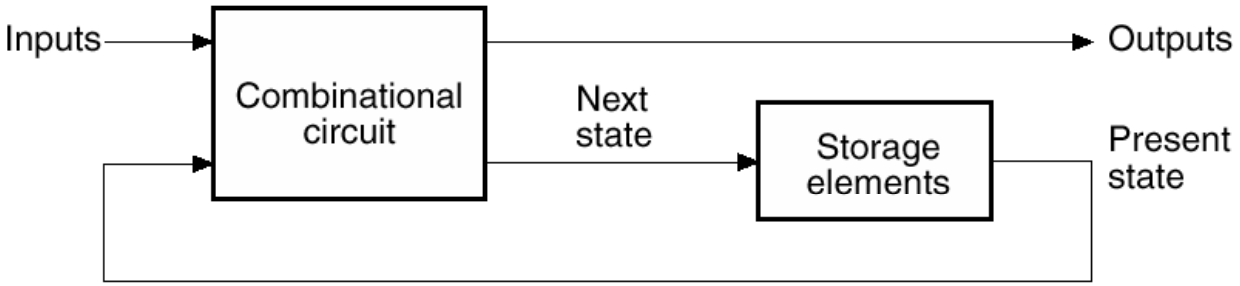
Excitation Table		
Q	Q+	D
0	0	0
0	1	1
1	0	0
1	1	1



State Transition Diagram

Transparency

- As long as C (the *trigger*) is high, state can change
- This is called *transparency*
- What's problem with that?
- Output of one latch may feedback
  - So more state changes may happen
  - Depends on gate delays



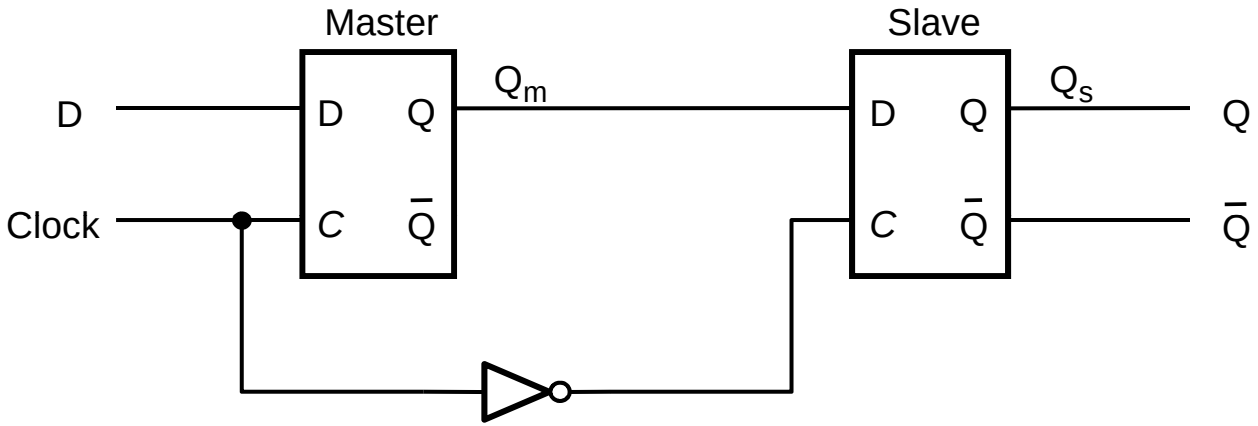
- Want to change latch state ***once***
  - Depending on inputs at time of clock

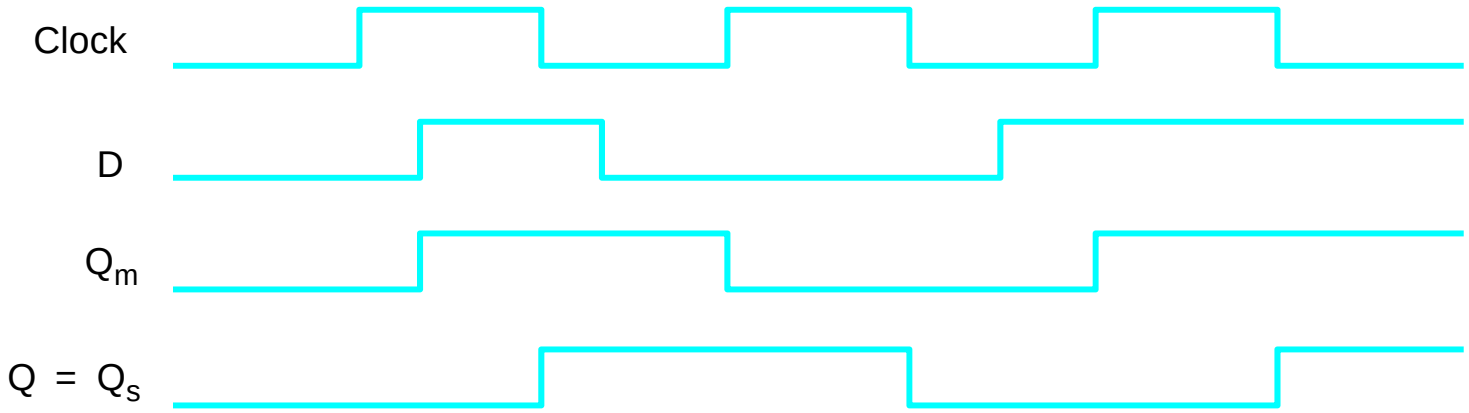
Flip-Flops

- Ensure only one transition
- Two major types
- Master-Slave
  - Two stage
  - Output not changed until clock disabled
- Edge triggered
  - Change happens when clock level changes

Master-Slave D Flip-Flop

- Either Master or Slave is enabled, not both



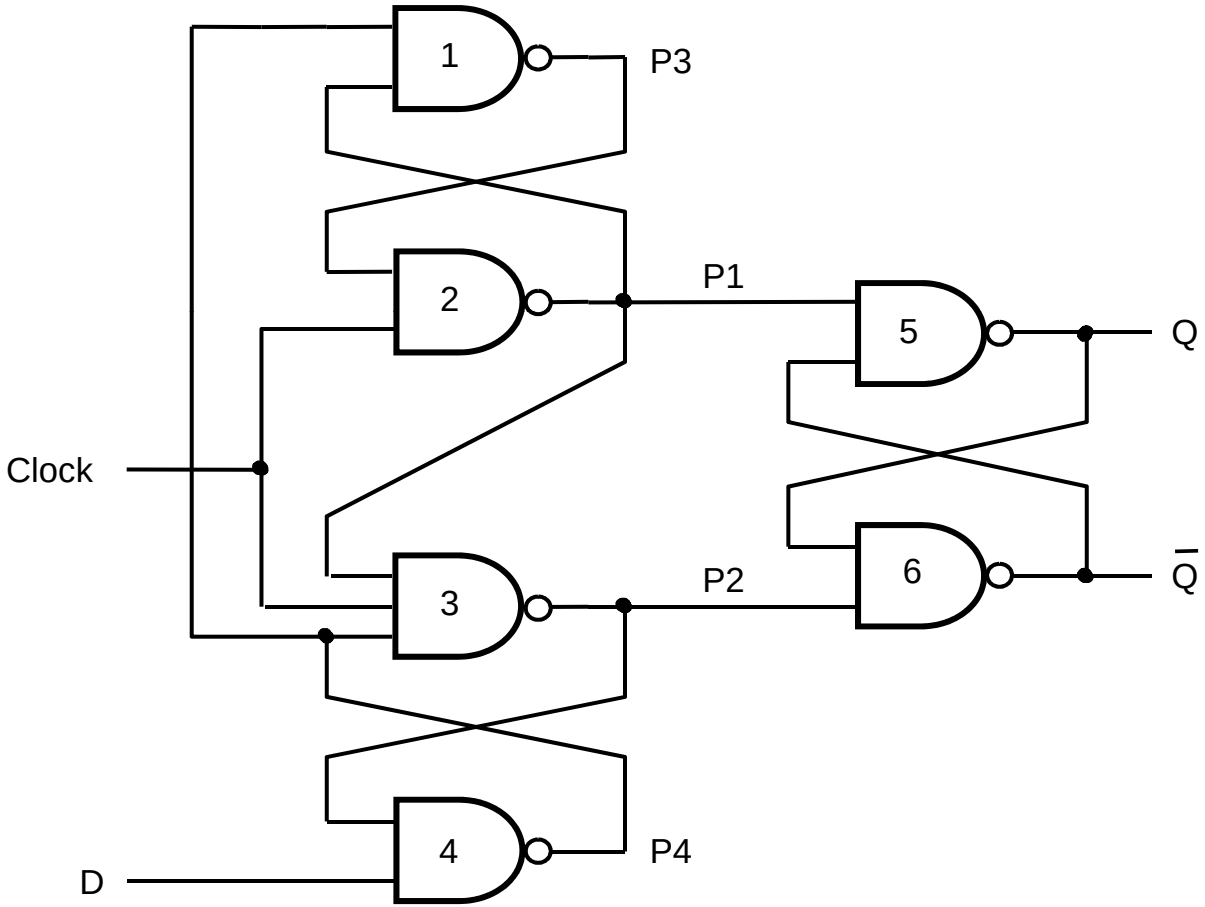


Timing diagram

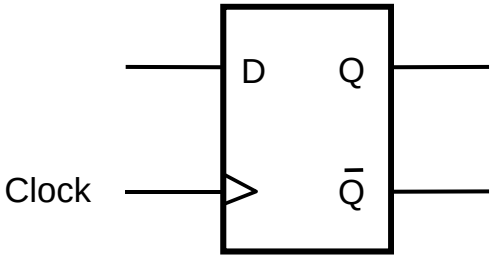
Have We Fixed the Transparency Problem?

- Output no longer transparent
  - Combinational circuit can use last values
  - New inputs appear at latches
  - Not sent to output until clock low
- But changes at input of FF when clock high can affect output
- Solution: ***edge-triggered flip-flops***
- New state latched on ***clock transition***
  - Low-to-high or high-to-low
- Changes when clock high are ignored
- Note: Master-Slave also called ***pulse triggered***

D Flip-Flop



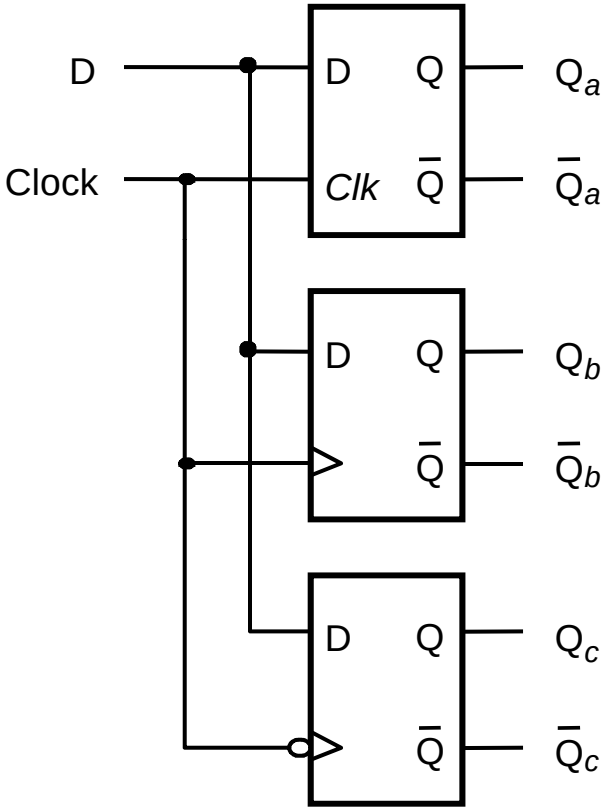
(a) Circuit



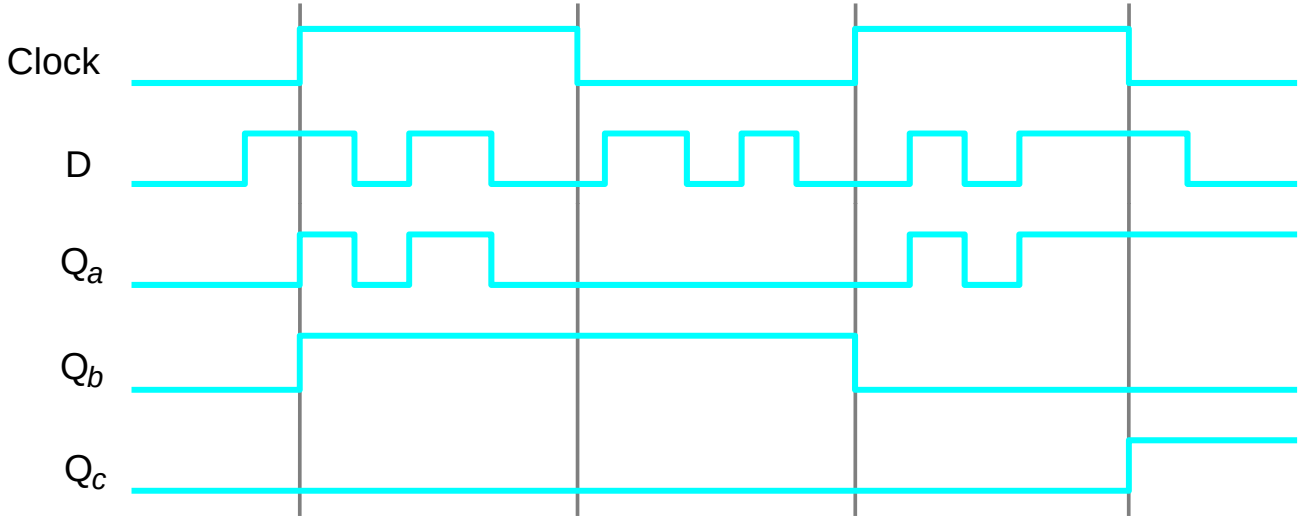
(b) Graphical symbol

A positive-edge-triggered D flip-flop

D Latch versus D Flip-Flop



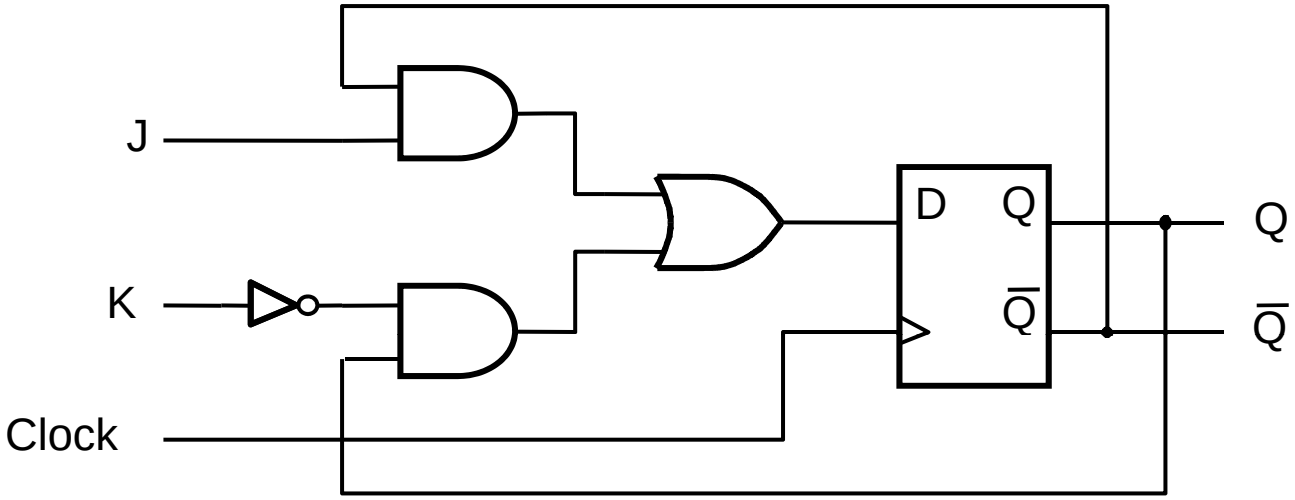
(a) Circuit



(b) Timing diagram

Comparison of level-sensitive and edge-triggered devices

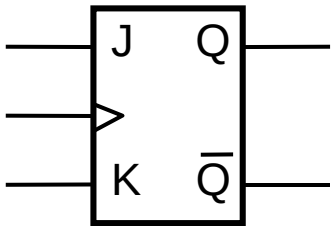
JK Flip-Flop



(a) Circuit

J	K	Q (t+1)
0	0	Q (t)
0	1	0
1	0	1
1	1	$\overline{Q}$ (t)

(b) Truth table



(c) Graphical symbol

- Not used much anymore in VLSI
- Advantageous only if using FF chips

JK Flip-Flop

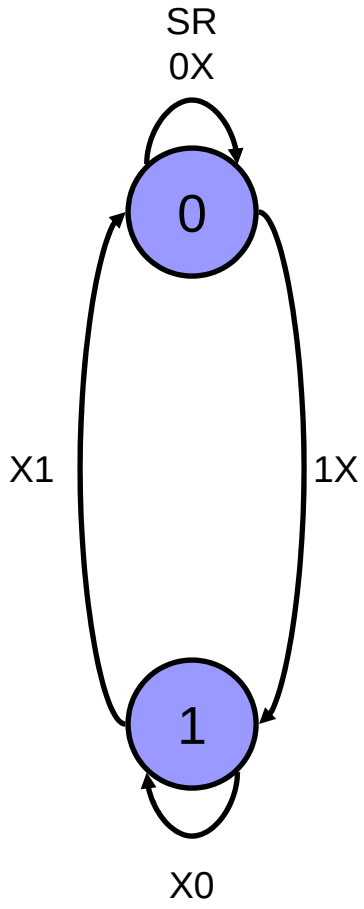
Detailed Function Table			
J	K	Q	Q+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

JK / Q	00	01	11	10
0	0	0	1	1
1	1	0	0	1

Characteristic Equation

$Q+ = JQ' + K'Q$

Excitation Table			
Q	Q+	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0



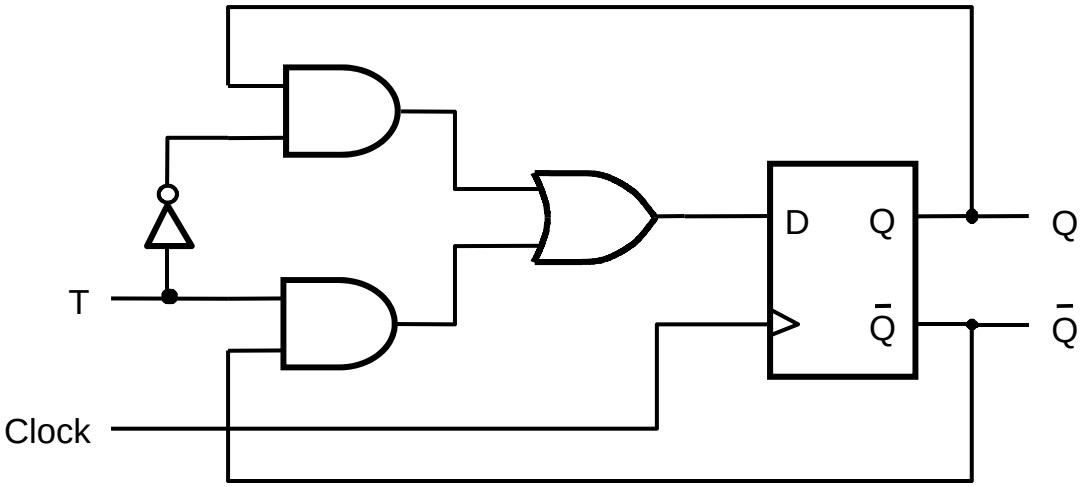
State Transition Diagram

T Flip-Flop

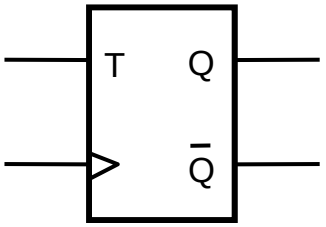
- Useful in counters
- Not available in IC form
- T Latches do not exist

T	Q(*t* + 1)
0	Q(*t*)
1	$\overline{Q}(t)$

(b) Truth table

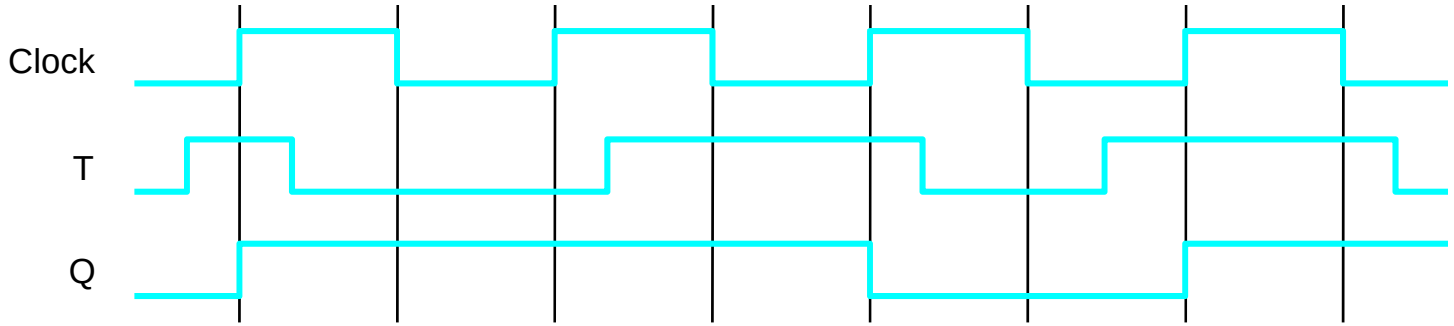


(a) Circuit



(c) Graphical symbol

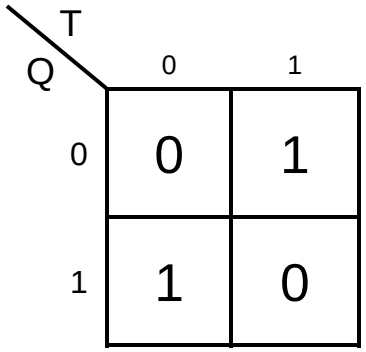
T Flip-Flop



(d) Timing diagram

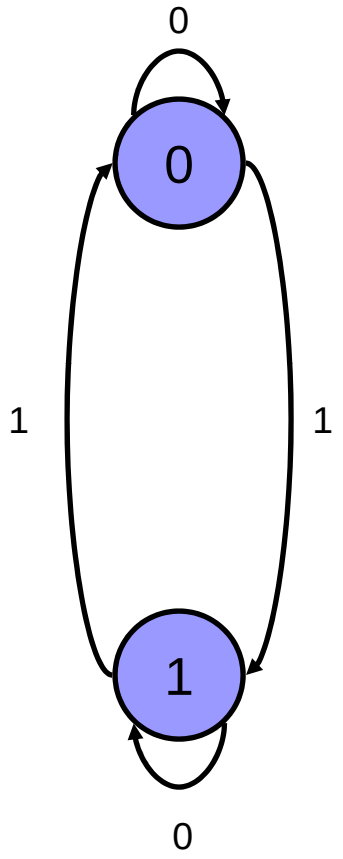
T Flip-Flop

Detailed Function Table		
T	Q	Q+
0	0	0
0	1	1
1	0	1
1	1	0



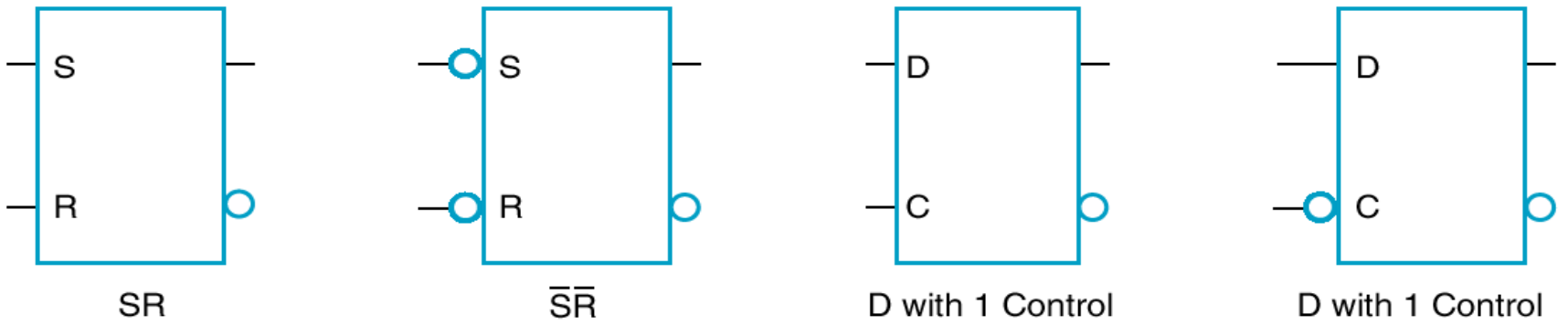
Characteristic Equation
$Q+ = T'Q + TQ' = T \oplus Q$

Excitation Table		
Q	Q+	D
0	0	0
0	1	1
1	0	1
1	1	0



State Transition Diagram

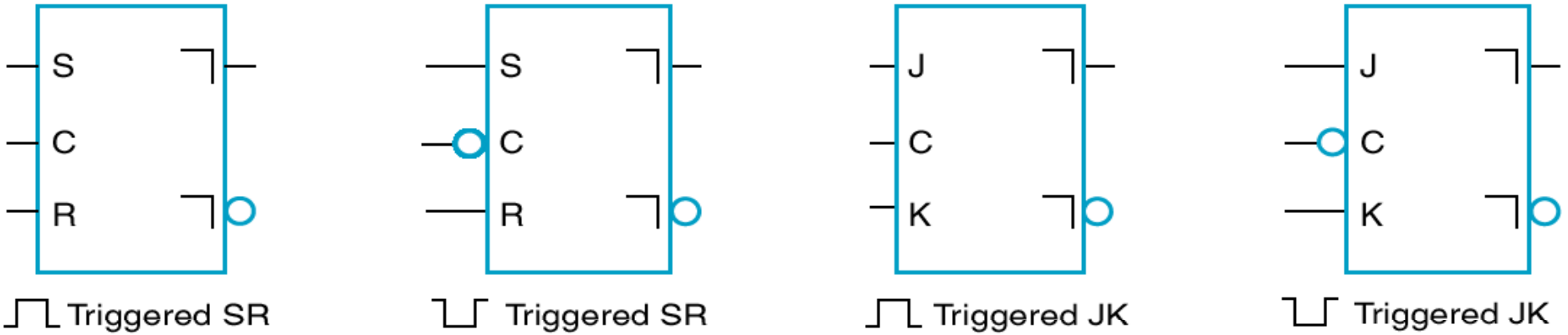
Standard Symbols – Latches



(a) Latches

Circle at input indicates negation

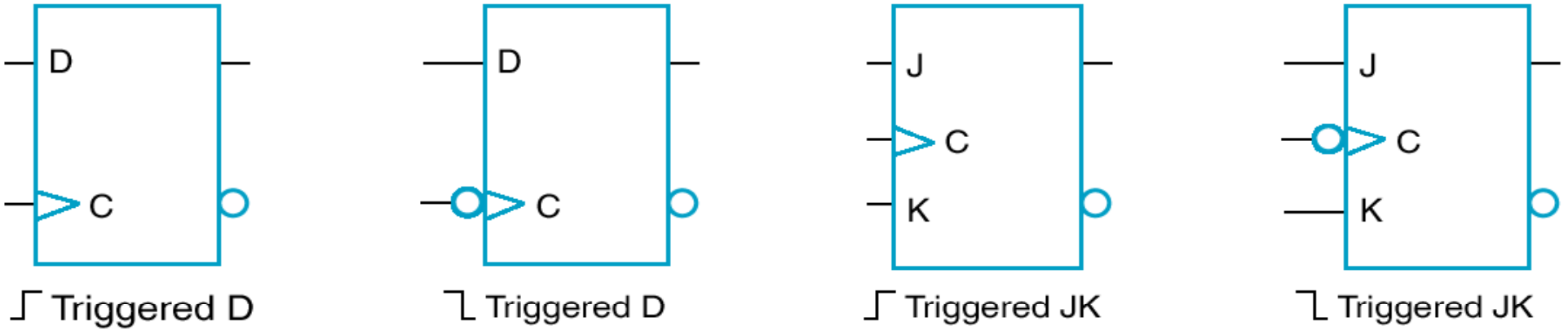
Symbols – Master-Slave



(b) Master-Slave Flip-Flops

Inverted L indicates postponed output
Circle indicates whether enable is positive or negative

Symbols – Edge-Triggered



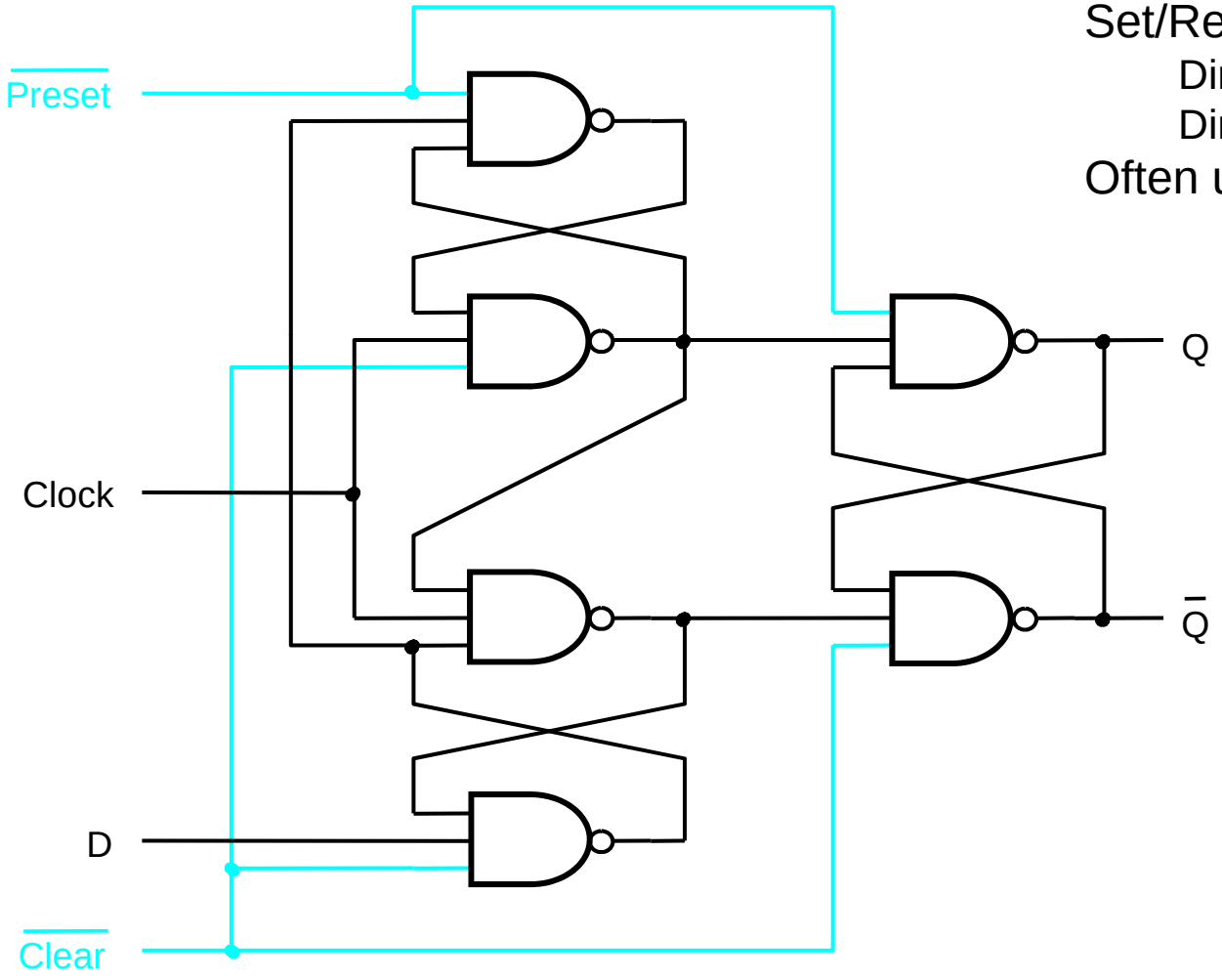
(c) Edge-Triggered Flip-Flops

Arrow indicates edge trigger

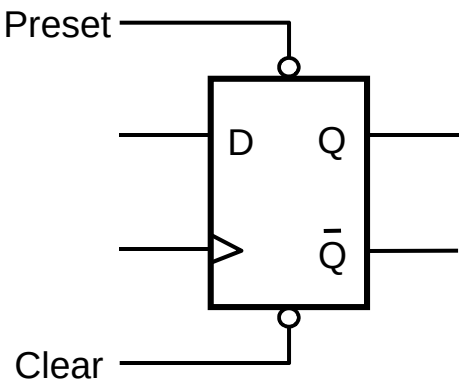
Clear and Preset Inputs

Set/Reset independent of clock
- Direct set or *preset*
- Direct reset or *clear*

Often used for power-up reset



(a) Circuit



(b) Graphical symbol

Choosing a Flip-Flop

- RS Clocked Latch:
  - used as storage element in narrow width clocked systems
  - its use is not recommended!
  - however, fundamental building block of other flipflop types
- D Flipflop:
  - minimizes wires, much preferred in VLSI technologies
  - simplest design technique
  - best choice for storage registers
- JK Flipflop:
  - versatile building block: can be used to implement D and T FFs
  - usually requires least amount of logic to control Q+
  - however, has two inputs with increased wiring complexity
- T Flipflop:
  - doesn't really exist, constructed from J-K FFs
  - usually best choice for implementing counters
- Preset and Clear inputs highly desirable!!

Characteristic Equation & Excitation Table Summary

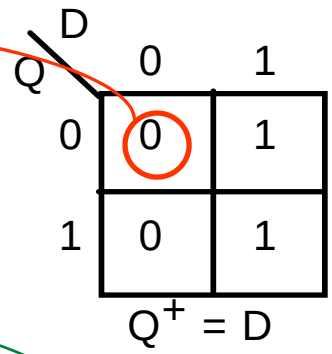
Device Type	*Characteristic Equation*
SR latch	Q+ = S + R'Q
D latch	Q+ = D
JK flip-flop	Q+ = JQ' + K'Q
T flip-flop	Q+ = TQ' + T'Q

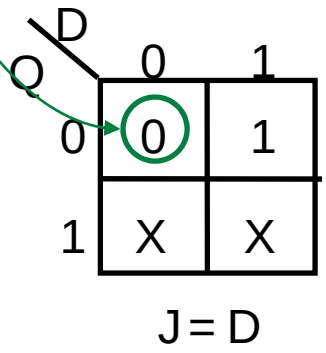
Q	Q+	S	R	D	J	K	T
0	0	0	X	0	0	X	0
0	1	1	0	1	1	X	1
1	1	0	1	1	X	1	1
1	1	X	0	0	X	0	0

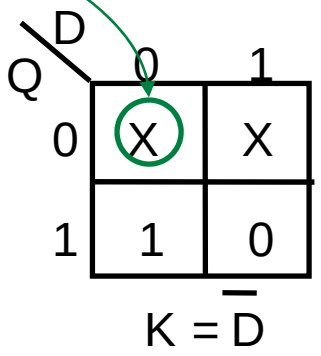
Implementing One FF in Terms of Another

- Design Procedure: Implementing D FF with a J-K FF:
  1. Start with K-map of Q+ = *f* (D, Q)
  2. Create K-maps for J and K with same inputs (D, Q)
  3. Referring to excitation table, fill in K-maps with appropriate values for J and K to cause the same state changes as in the original K-map

Q	Q+	S	R	D	J	K	T
0	0	0	X	0	0	X	0
0	1	1	0	1	1	X	1
1	1	0	1	1	X	1	1
1	1	X	0	0	X	0	0



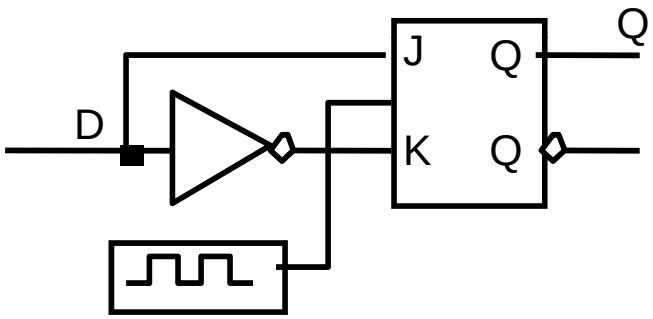




e.g., D = Q = 0, Q+ = 0
1. From table: J = 0, K = X
2. Fill 0 to QD = 00 for J map
3. Fill X to QD = 00 for K map

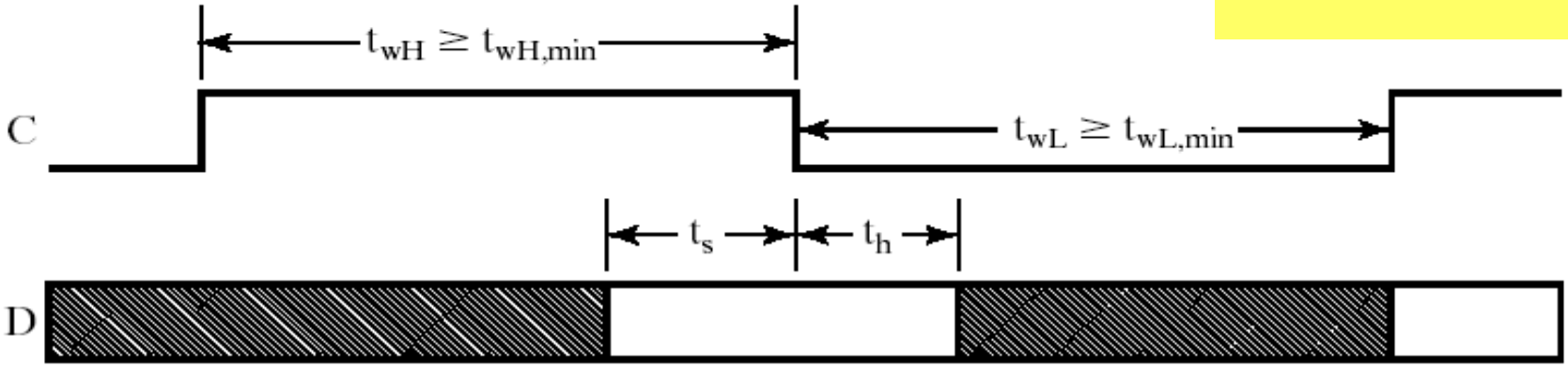
Implementing One FF in Terms of Another

- The circuit



Flip-Flop Timing

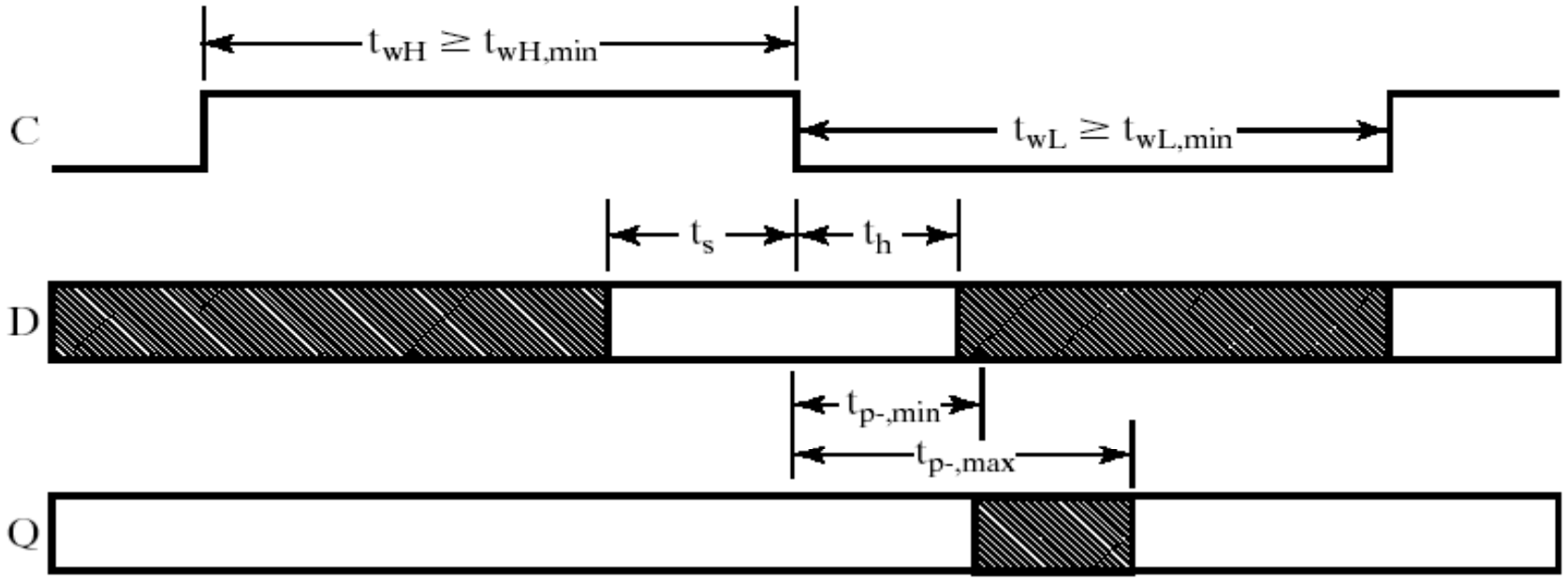
There is a timing "window" around the clocking event during which the input must remain stable and unchanged in order to be recognized



- Clock - Periodic event, causes state of memory element to change
- Setup time – time that D must be available before clock edge
- Hold time – time that D must be stable after clock edge

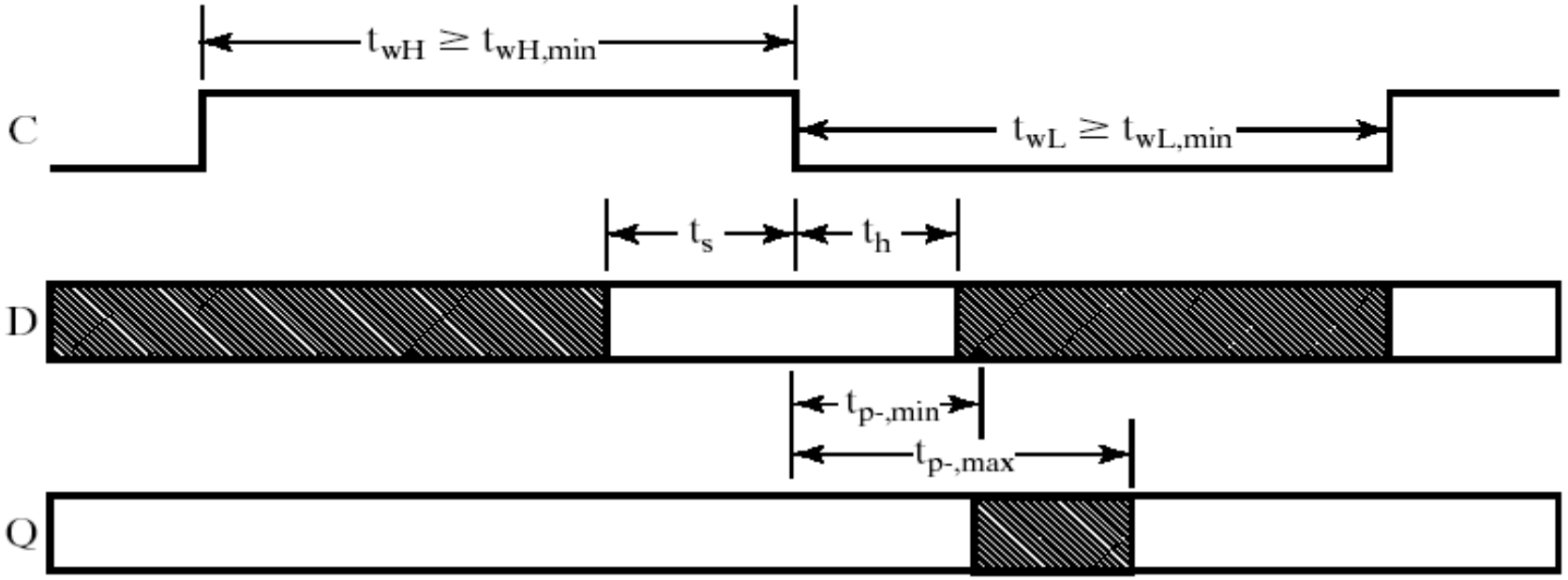
Propagation Delay

- Propagation delay – time after edge when output is available



Clock Pulse Requirements

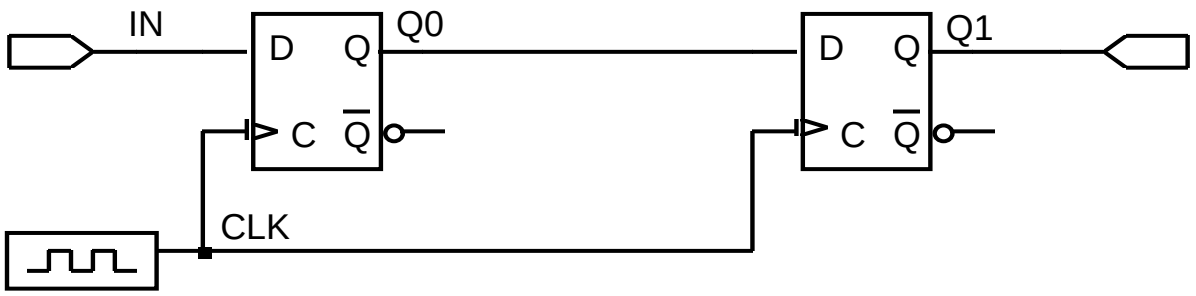
- Basically a max clock frequency
- Pulse cannot be too narrow



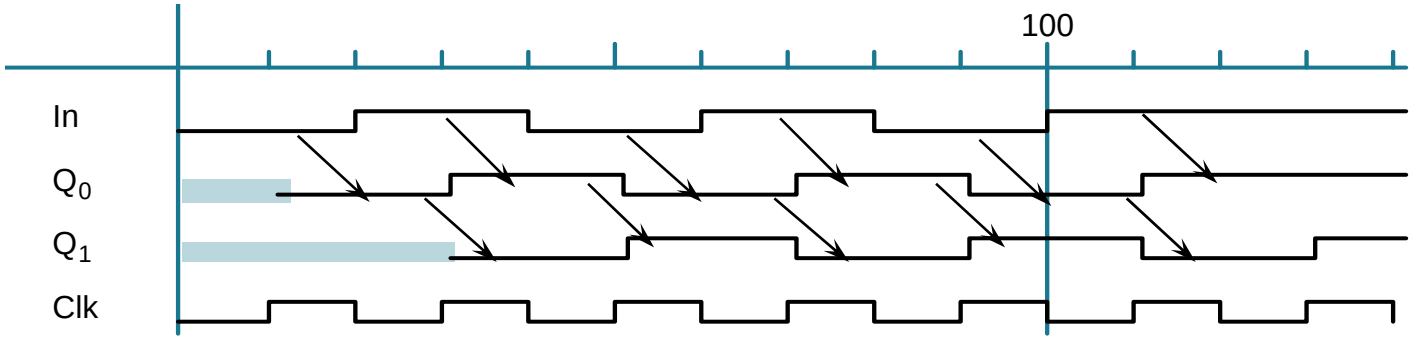
Cascaded Flip-Flops

Shift Register
Have S, R (preset, preclear) inputs

New value to first stage while second stage obtains current value of first stage

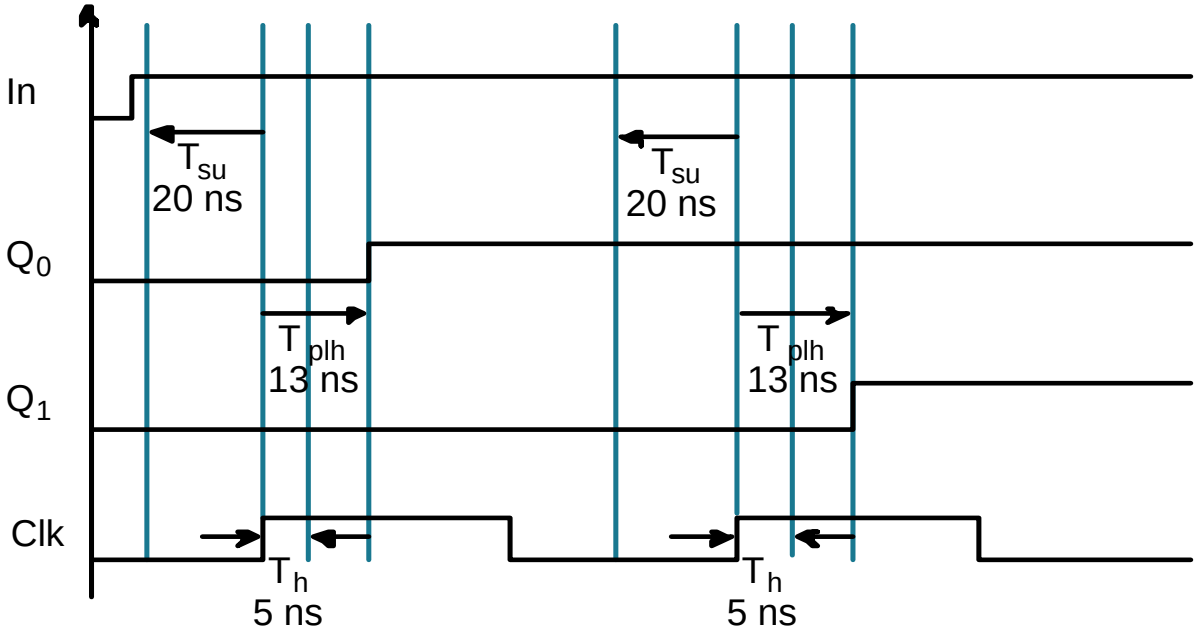


Correct Operation, assuming positive edge triggered FF



Cascaded Flipflops

- Why this works:
  - Propagation delays far exceed hold times;
  - Clock width constraint exceeds setup time - **incorrect operation if clock period too short!**
  - This guarantees following stage will latch current value before it is replaced by new value
  - Assumes infinitely fast distribution of the clock

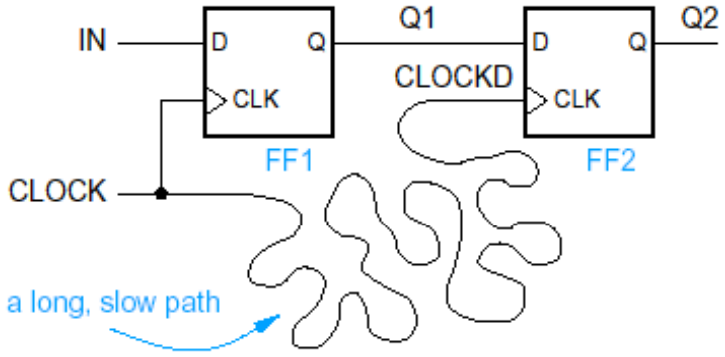


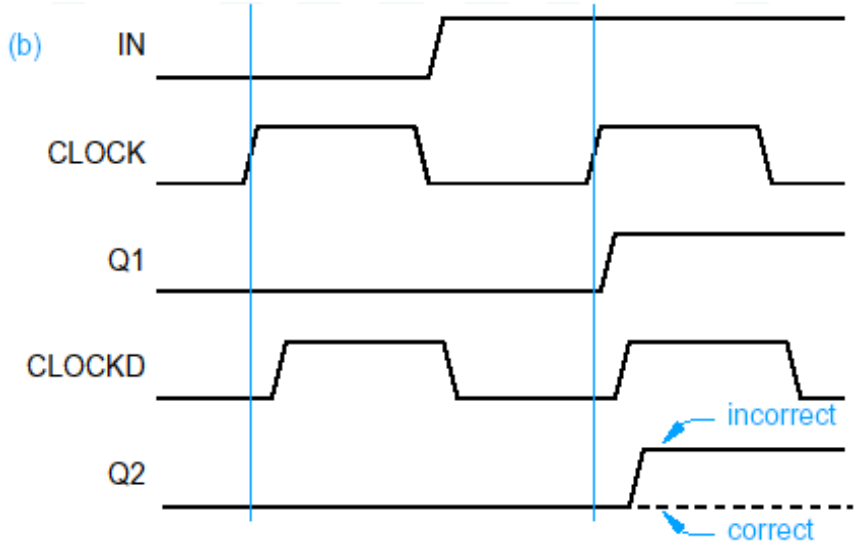
Timing constraints guarantee proper operation of cascaded components

* T_{plh} - L to H propagation delay

Clock Skew

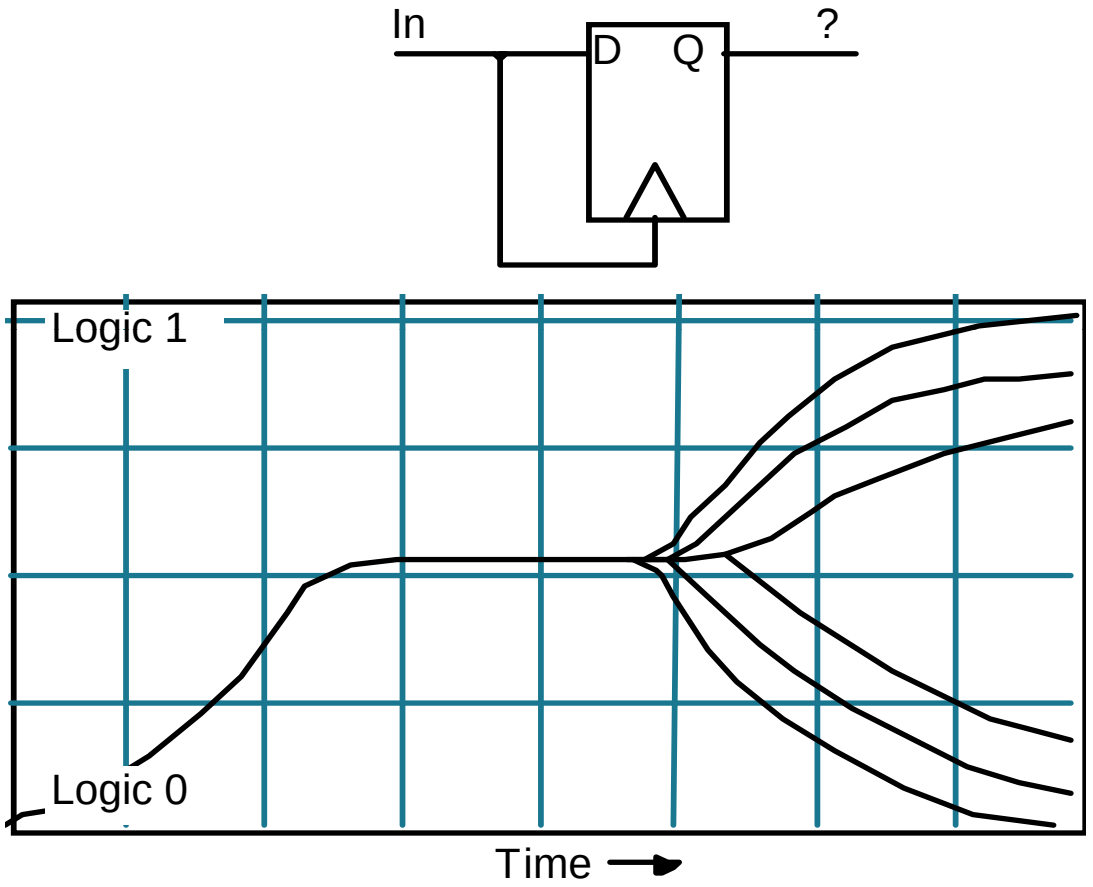
- ***Clock skew***
  - Definition: difference between arrival times of the clock at different FFs
- Clock skew also caused by difference setup and hold times of different devices
- Correct behavior assumes next state of all storage elements determined by all storage elements at the same time
- Not possible in real systems!





Asynchronous Inputs and Metastability

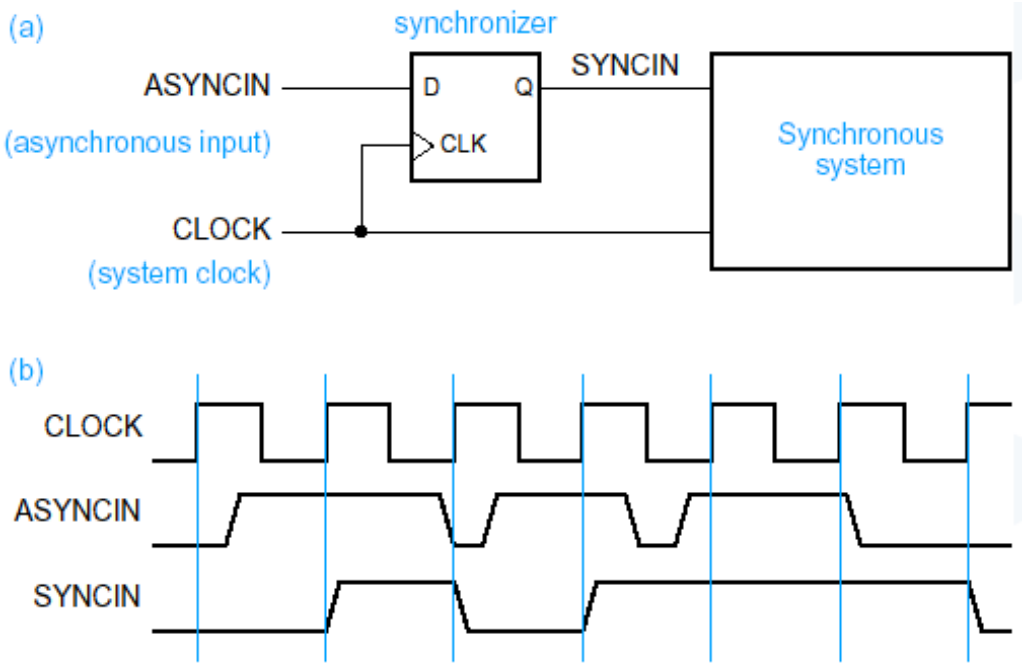
- Asynchronous Inputs Are Dangerous!
  - Since they take effect immediately, glitches can be disastrous
- Synchronous inputs are greatly preferred!
- What happens when an asynchronous input is not synchronized?
- When FF input changes close to clock edge, the FF may enter the metastable state: neither a logic 0 nor a logic 1
- It may stay in this state an indefinite amount of time, although this is not likely in real circuits



Oscilloscope Traces Demonstrating Synchronizer Failure and Eventual Decay to Steady State

Simple Synchronizer

- But asynchronous inputs cannot be avoided
  - e.g., reset signal, memory wait signal
- Initial versions of commercial IC's that suffered metastability
  - Zilog Z-80 Serial I/O Interface
  - Intel 8048 microcontroller
  - AMD 29000 RISC microprocessor
- *A synchronizer* samples an asynchronous input and produces an output that meets the setup and hold times required in a synchronous system



A Better Synchronizer

- The probability of failure can never be reduced to 0, but it can be reduced
  - Synchronizer failure becomes a big problem for very high speed systems
- How to get a flip-flop out of the metastable state:
  - Force the flip-flop into a valid logic state using input signals that meet the published specifications for minimum pulse width, setup time, and so on.
  - Wait “long enough,” so the flip-flop comes out of metastability on its own
    - slow down the system clock
    - this gives the synchronizer more time to decay into a steady state
  - Use fastest possible logic in the synchronizer
    - S or AS TTL D-FFs are recommended (applicable only to TTL circuits)
  - Cascade two synchronizers