University of Massachusetts Amherst

Engin112 – Lectures 25-26

Sequential Circuits
Latches and Flip-flops

Maciej Ciesielski
Department of Electrical and Computer Engineering
11/02-04/2011

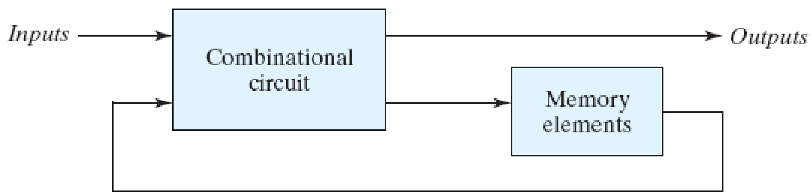
Sequential vs Combinational Ckts

- Recall definition of combinational logic
  - Outputs are a function of inputs
  - Combinational delay (each gate has some propagation delay)
- Sequential circuits
  - Outputs depend on inputs <u>and</u> previous values of outputs
    - » Outputs depend on previous state of the circuit
    - » State is stored in memory elements (registers, latches, fliplops)
  - Need binary storage elements
    - » Latches
    - » Flip-flops

Sequential Circuits

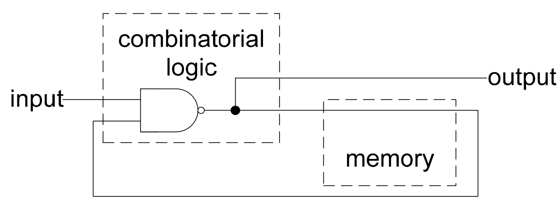
- So far: combinational circuits
  - Output of circuit depends only on values of inputs
- Most real-world circuits also contain memory
  - Pocket calculators
  - Counters
  - Computer systems, etc.

- Sequential circuit:
  - Outputs depend on inputs and previous output values



Flavors of Sequential Circuits

- What is the output of this sequential circuit?



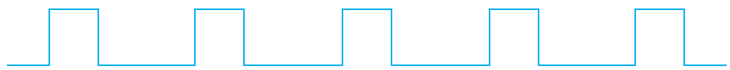
  - Sequence of alternating 1's and 0's
  - Exhibits "asynchronous" behavior

- Circuit can change at any moment in time
  - Asynchronous sequential circuit
  - Behavior depends on physical characteristics of circuit
    - » Speed of logic gate, signal propagation

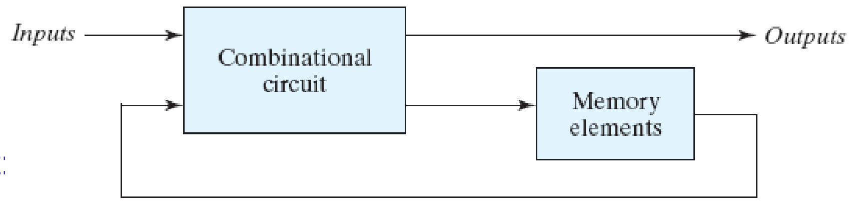
- Very difficult to design and analyze

Synchronous Sequential Circuits

- Synchronous sequential circuits
  - Storage elements change only at discrete instances of time
- Timing controlled by “clock”, a periodic enable signal
  - Clock generator provides train of clock pulses:



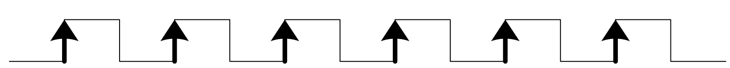
- Clocked (synchronous) sequential circuit:



Storage Elements

- Binary storage device capable of storing one bit
- Latch = *level-sensitive* device

  - Control signal: Enable
  - State changes with input when enabled (e.g., when Enable = 1)
  - Holds last input value when disabled (when Enable = 0)
- Flip-flop = *edge-triggered* device
  - Control signal: periodic Clock
  - State of flip-flop can only change during clock *transition*
  - Example: Flip-flops change on rising/falling edge of clock



- Why change on an edge?
  - Couldn’t we change state while clock is 1?
  - That would be a latch!
- Edge is moment in time, state is duration
  - Feedback would continue during clock being 1, causing possible race conditions

Level-sensitive vs Edge-triggered

- Latches are <u>level-sensitive</u>

(a) Response to positive level

- Flip-flops are <u>edge-sensitive</u>

(b) Positive-edge response

(c) Negative-edge response

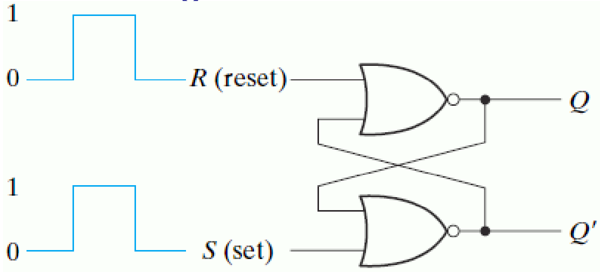
Latches

- Characteristics
  - Can store one bit of binary information
  - <u>Level-sensitive</u> devices, asynchronous

- SR Latch
  - Named after functionality: *S* = set, *R* = reset
  - Specification:
    - » Inputs: *S* and *R*
    - » Outputs: *Q* and *Q'*
  - Operation:
    - » *Q=1* and *Q'=0* when in set state
    - » *Q=0* and *Q'=1* when in reset state
    - » Inputs should be 0 unless pulse on *S* or *R* sets or resets latch

SR Latch

- Circuit diagram:

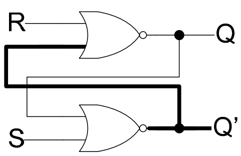


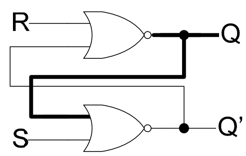
(a) Logic diagram

S	R	Q	Q′	
1	0	1	0	
0	0	1	0	(after S = 1, R = 0)
0	1	0	1	
0	0	0	1	(after S = 0, R = 1)
1	1	0	0	

(b) Function table

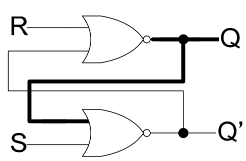
- *Set* and *Reset* are stable states
  - If *S=0* and *R=0*, then state will not change by itself



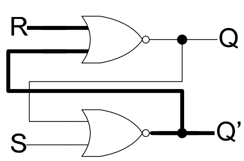


SR Latch - Operation

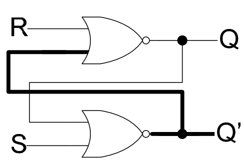
- Operation:



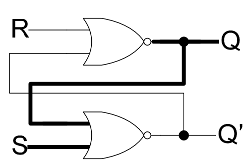
set state
R,S=0
Q=1
Q'=0



reset active
R=1
Q=0
Q'=1



reset state
R,S=0
Q=0
Q'=1

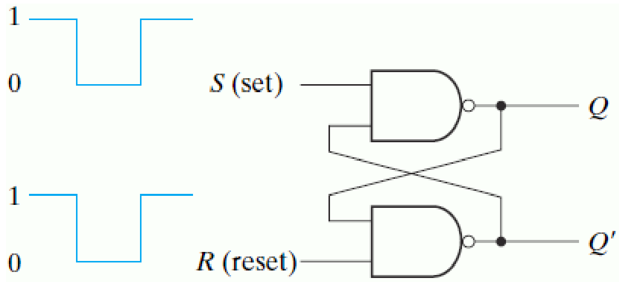


set active
S=1
Q=1
Q'=0

- What happens if both *S,R* = 1?
  - Both NOR outputs become 0
  - Unstable state after releasing *S* and *R*

SR Latch with NAND Gates

- Latch can also be implemented with NAND gates



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	

  - Input of 0 indicates set or reset

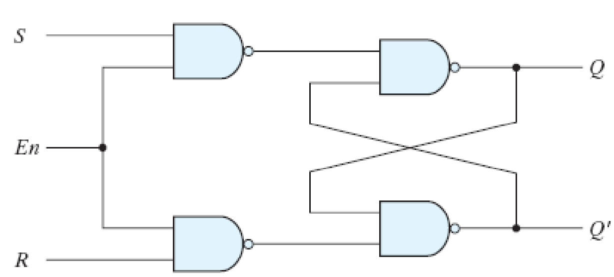
- Referred to as *S'R'* latch or $\overline{SR}$ latch

SR Latch with Control Input

- We want to control when latch can change
  - Extra control input *En* (enable)

- SR latch with control input
  - Problem: indeterminate state for *S=R=1*, when *En=1*

En	*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



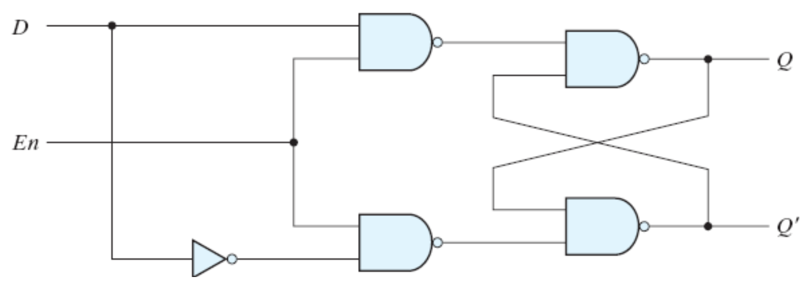
- Circuit:
  - NAND gates and *S'R'* latch

D Latch

- How to remove state for *S=1*, *R=1* ?
- Solution
  - Just use one input pin *D* to indicate set or reset
  - Enable bit (*En*) ensures that latch is only set when intended

En	*D*	Next state of *Q*
0	X	No change
1	0	*Q* = 0; reset state
1	1	*Q* = 1; set state

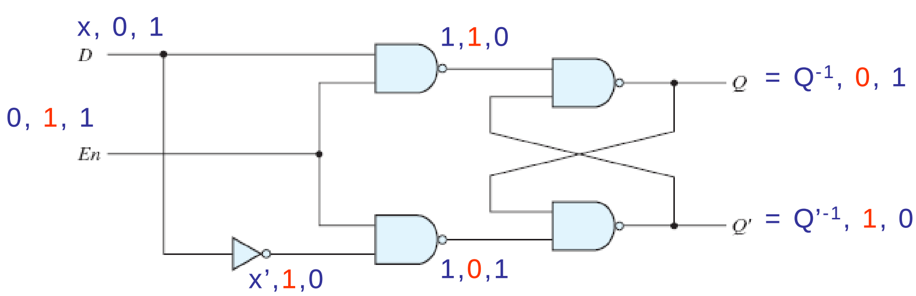
- D latch
  - Inputs:
    - *D* (data)
    - *En* (enable)
  - Circuit:



D Latch in Operation

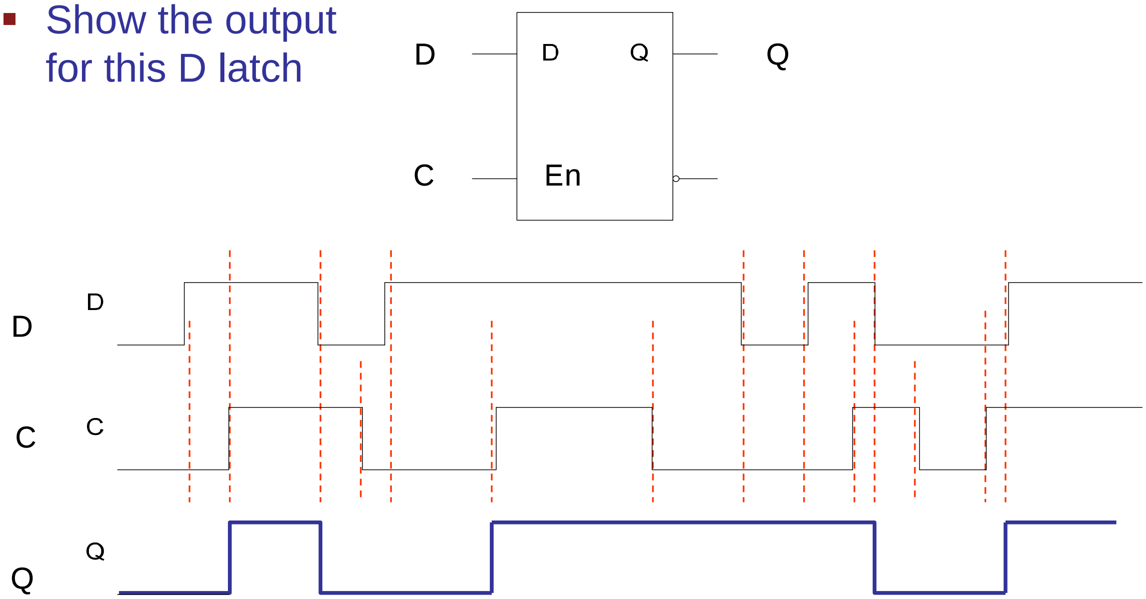
- Removes disallowed *R,S* combination
  - D latch forwards data while *En=1*
  - D latch holds data when *En=0*

En	*D*	Next state of *Q*
0	X	No change
1	0	*Q* = 0; reset state
1	1	*Q* = 1; set state



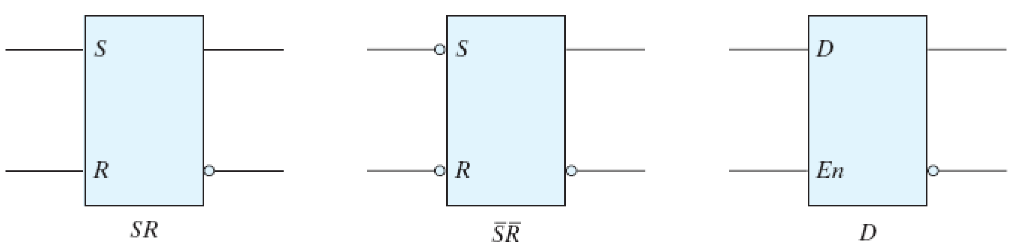
D Latch Operation

- Show the output for this D latch



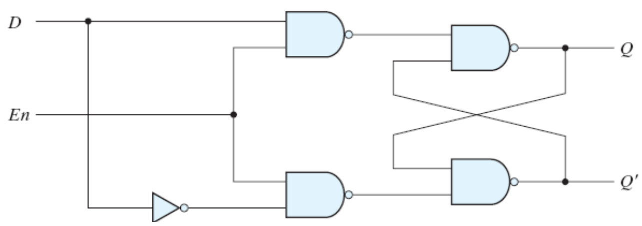
Latch Symbols

- Graphic symbols for latches:



D Latch - Summary

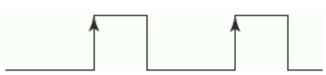
- D latch circuit



En	*D*	Next state of *Q*
0	X	No change
1	0	*Q* = 0; reset state
1	1	*Q* = 1; set state

- Data is stored while clock is high

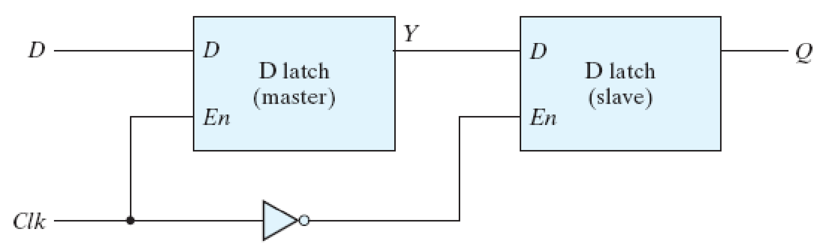
- How can we build a flip-flop that stores on edge transition?



or

Edge-triggered D Flip-Flop

- Construct D flip-flop from two latches:

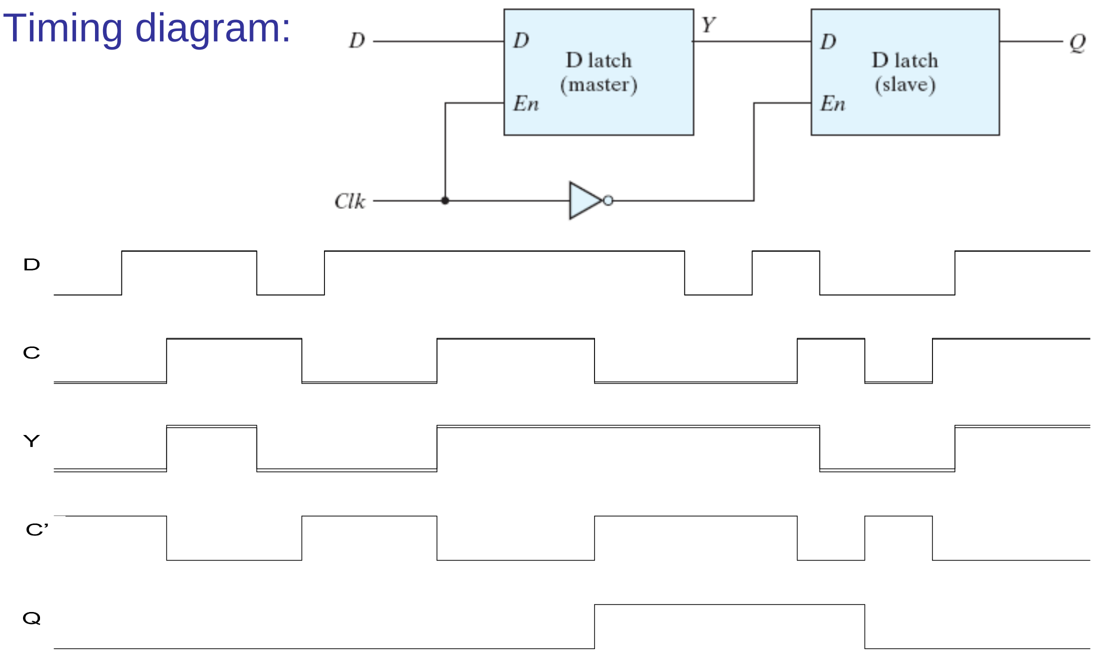


- Primary latch:
  - Reads value of *D* while *CLK* is high
  - Is disabled when clock is low

- Secondary latch:
  - Is disabled when *CLK* is high (i.e., holds previous value)
  - Takes value from master on negative edge of clock

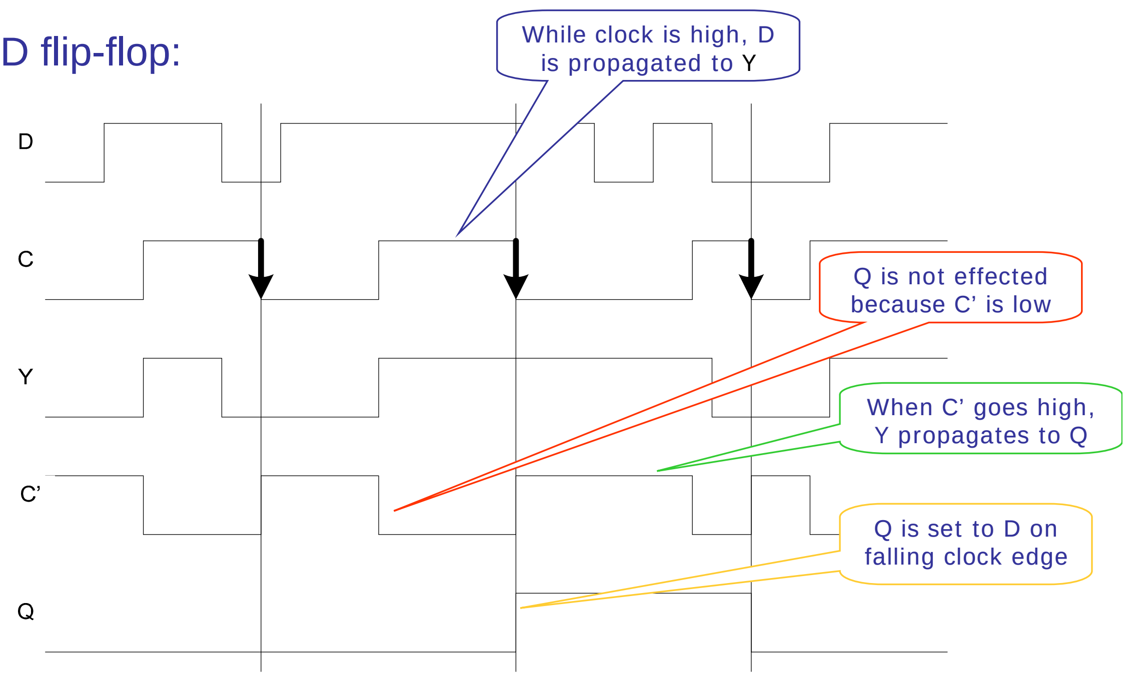
Operation of D Flip-Flop

- Timing diagram:



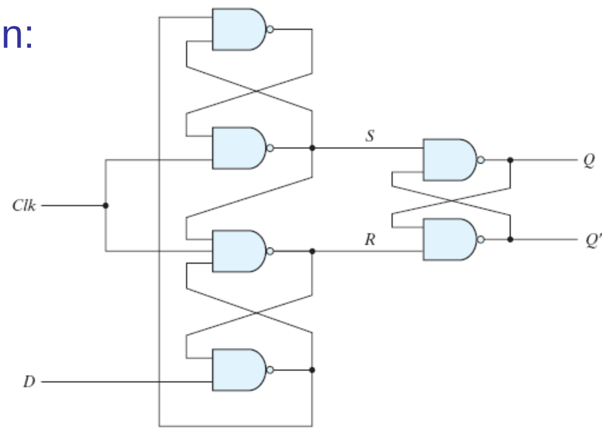
Edge Triggering

- D flip-flop:



D Flip-Flop

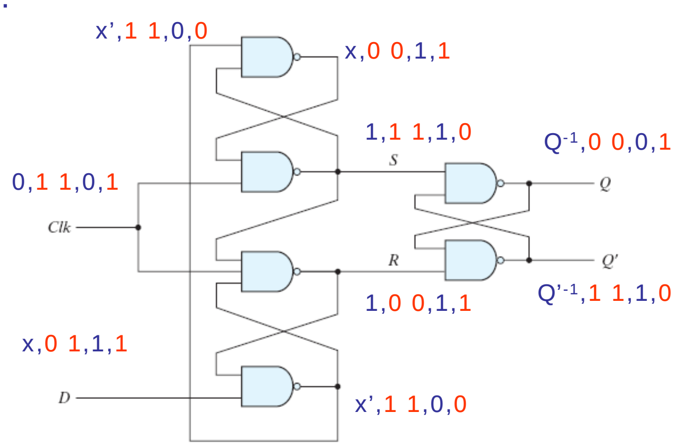
- Requires 2 D latches and one inverter
- More efficient implementation:



- Operation:
  - If *CLK=0*, then *S=R=1*
    - » *Q* remains stable
  - If *CLK=1* and *D=0*, then *R=0*
    - » *Q* is reset to 0
    - » *Q* remains 0 independent of D
  - If *CLK=0*, then *S=R=1*
  - If *CLK=1* and *D=1*, then *S=0*
    - » *Q* is set to 1
    - » *Q* remains 1 independent of *D*
- Is this flip-flop positive or negative edge triggered?

D Flip-Flop - Operation

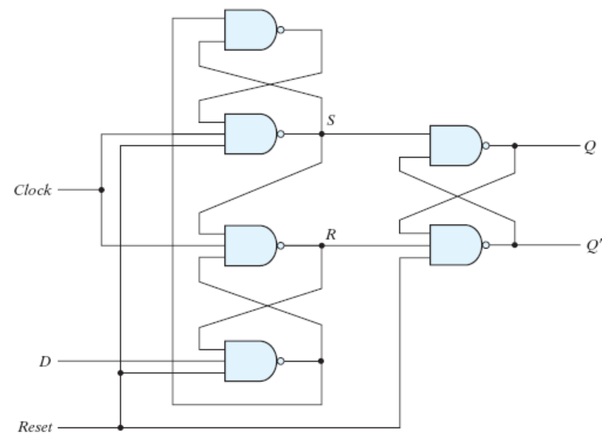
- More efficient implementation:
  - If *CLK=0*, then *S=R=1*
    - » *Q* remains stable = Q^{-1} regardless of *D* (= *x*)
  - If *CLK=1* and *D=0*, then *R=0*
    - » *Q* is reset to 0
  - If *CLK=1* and *D=1*, *R=0*
    - » *Q* = 0
  - If *CLK=0*, then *S=R=1*
    - » *Q* remains 0 independent of *D*
  - If *CLK=1* and *D=1*, then *S=0*
    - » *Q* is set to 1
    - » remains 1 independent of D



- Is this flip-flop positive or negative edge triggered?

Direct (Asynchronous) Inputs

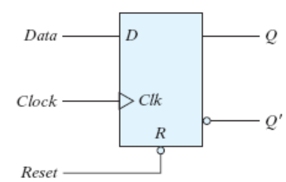
- State of output of latches and flip-flops are unknown on power-up
- Direct inputs can reset flip-flop
  - Forces latches to reset
  - These are *asynchronous* inputs
    - » Can be applied at any time unrelated to the clock



- Function table:

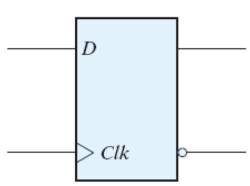
R	Clk	D	Q	Q'
0	X	X	0	1
0	↑	0	0	1
0	↑	1	1	0

Graphic symbol:

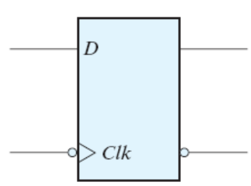


Dynamic Indicator

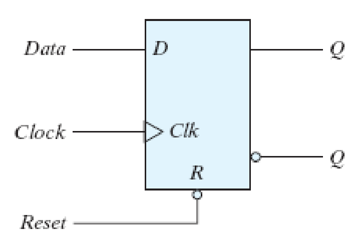
- Flip-flop symbols:



(a) Positive-edge



(a) Negative-edge



  - Triangle indicates clock (periodic signal)
  - Set and Reset are asynchronous
- Edge trigger:
  - No bubble at clock: positive edge triggered
  - Bubble at clock: negative edge triggered

Other Flip-Flops

D flip-flop summary

- D flip-flop are small and are the most popular in industry
- D flip-flop can only set or reset the flip-flop
  - No option for holding *previous state*
  - State changes in synchrony with clock depending on *D*
- D FF characteristic table
  - Shows output as function of data input D
  - Clock is assumed to change periodically

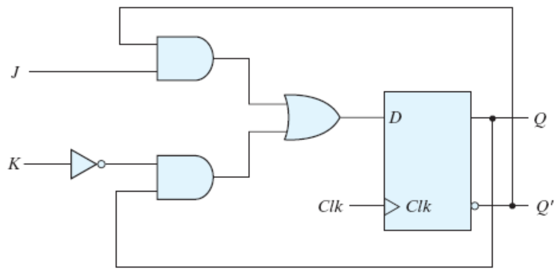
D	*Q(t+1)*
0	0
1	1

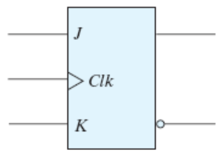
- Asynchronous set/reset is possible (FF with "direct inputs")
- We may need other FFs to *synchronously* set or reset the FF state and/or complement the previous state
  - JK flip-flops
  - T flip-flops (*toggle* FF)

JK Flip-Flop

- JK flip-flop:
  - *J=1* sets *Q* to 1
  - *K=1* sets *Q* to 0
  - *J=1* and *K=1* complement *Q*
- JK flip-flop can be built from D flip-flop
  - *D = JQ' + K'Q*

J	*K*	*D*
0	0	*Q*
0	1	0
1	0	1
1	1	*Q'*

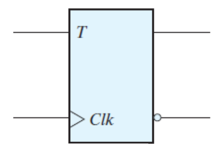




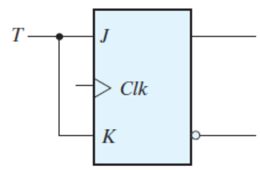
Graphic symbol

T Flip-Flop

- Single input *T*
  - "Toggles" (complements) *Q*

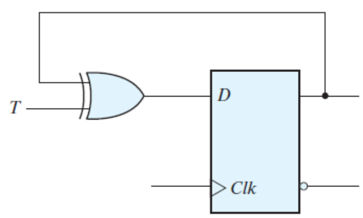


- Can be built from JK flip-flop:
  - $J = K = T$



(a) From *JK* flip-flop

- Can be built from D flip-flop:
  - $D = T \oplus Q = TQ' + T'Q$



(b) From *D* flip-flop

Characteristic Tables and Equations

- Functionality of flip-flops can be specified in two ways
- Characteristic table
  - *Q(t)* is current state/output at time *t*
  - *Q(t+1)* is state/output at next clock period

J	*K*	*Q(t+1)*
0	0	*Q(t)*
0	1	0
1	0	1
1	1	*Q'(t)*

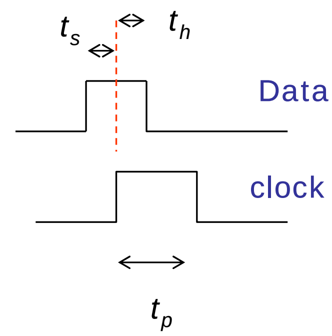
- Characteristic equation:
  equation describing FF functionality
  - JK flip-flop
    - » $Q(t+1) = J\,Q(t)' + K'\,Q(t)$
  - D flip-flop
    - » $Q(t+1) = D$
  - T flip-flop
    - » $Q(t+1)=Q'(t)$

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

T	*Q(t+1)*
0	*Q(t)*
1	*Q'(t)*

Flip-flop Timing

- Clock and data needs to propagate through circuit
- *D* must remain constant before clock transition
  - "Setup time" t_s
- *D* must remain constant after clock transition
  - "Hold time" t_h
- *CLK* needs to be high/low for a minimum time
  - "Pulse time" t_p
- Total time from clock edge to stabilized output
  - "Propagation time"



Summary

- Sequential circuits
  - Outputs depend on inputs and previous output values ("history")
- Binary storage elements (state registers)
  - Latches
    - » Level sensitive (Enable signal)
  - Flip-flops
    - » Edge sensitive (periodic Clock signal)