Topic 6

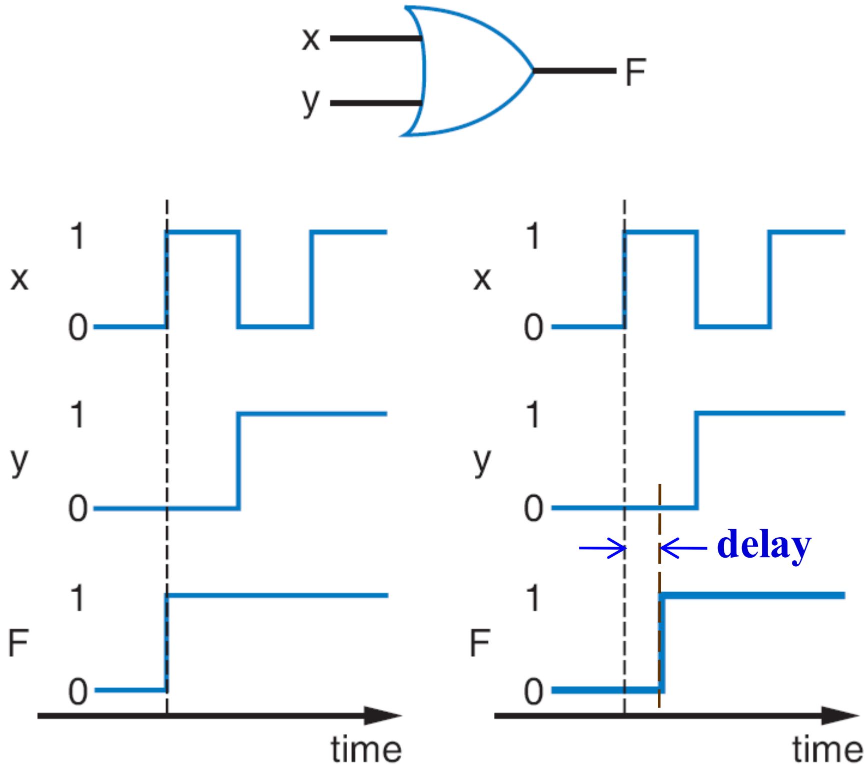
Latches and Flip Flops

Introduction

- Beginning from this lecture, we will:
  - Design a new type of building blocks, **latch** & **flip-flop**, that store value of a bit
  - Combine the blocks to build multi-bit storage – a **register**
  - Describe the sequential behavior of a digital circuit using a **finite state machine (FSM)**
  - Convert a finite state machine to a **controller** – a sequential circuit having a register and combinational logic

Reality of Combinational Circuit

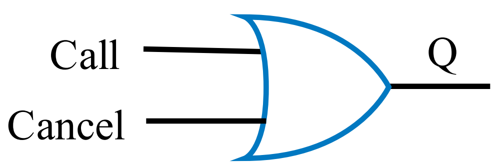
Non-Ideal Gate Behavior -- Delay



- Real gates have some delay
  - Outputs don't change immediately after inputs change

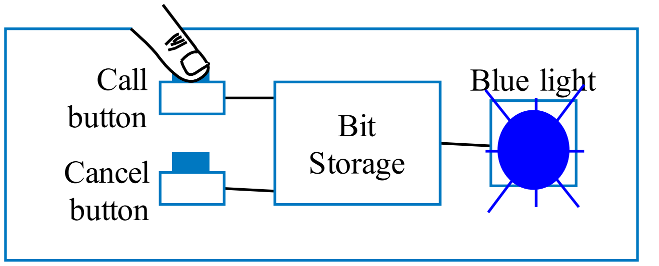
Example of Needing Bit Storage

- Flight attendant call button
  - Button pressed: provides a "1"
  - Press call: light turns on
    - ***Stays on*** after button released
  - Press cancel: light turns off
  - Logic gate circuit to implement this?

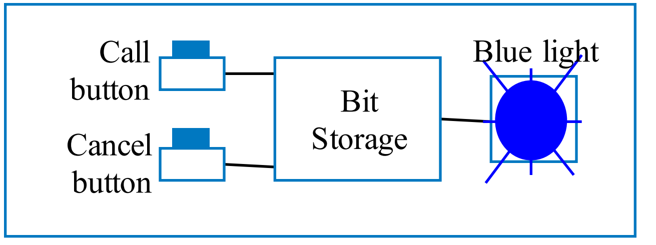


Doesn't work. Q=1 when Call=1, but doesn't stay 1 when Call returns to 0

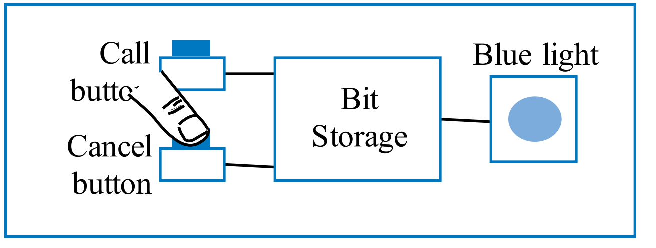
Need some form of "storage" in the circuit



1. Call button pressed – light turns on



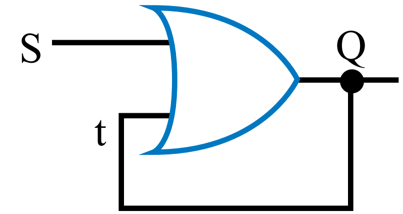
2. Call button released – light ***stays on***

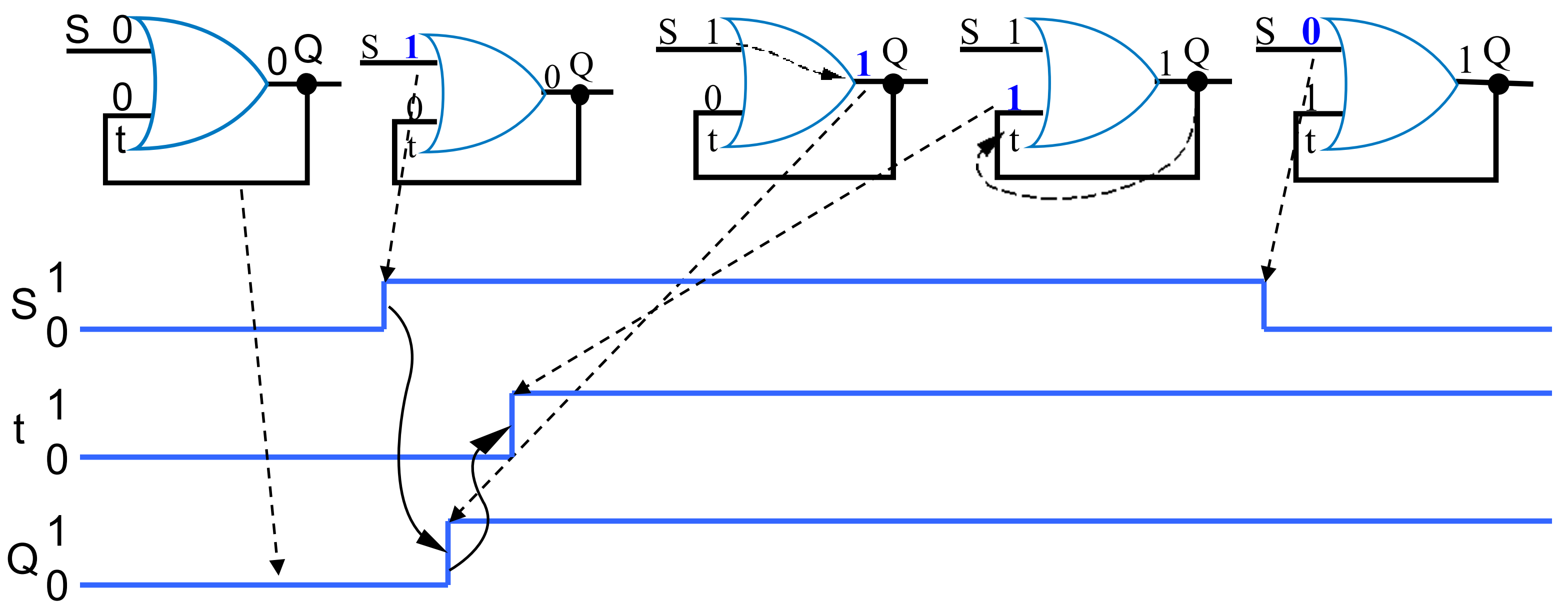


3. Cancel button pressed – light turns off

First Attempt at Implementation of Bit Storage

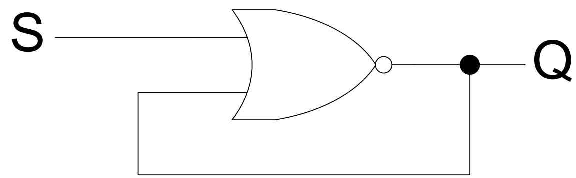
- We need some sort of feedback
  - Does circuit on the right do what we want?
    - Partially: Once Q becomes 1 (when S=1), Q stays 1 forever – no value of S can bring Q back to 0





Concepts of Sequential Circuit

- Sequential circuit
  - Combinational circuit with feedbacks

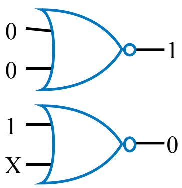


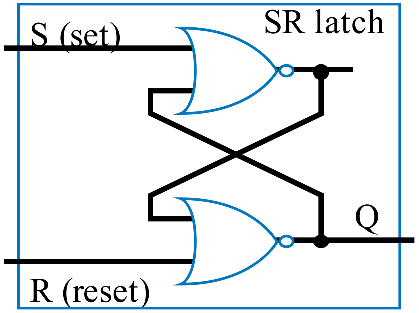
  - Due to the feedback, output of a sequential circuit is decided by
    - Present inputs, and
    - Past input sequence and
    - Past outputs sequence
- Timing concepts
  - input-output propagation delay
  - clock
  - Other timing issues

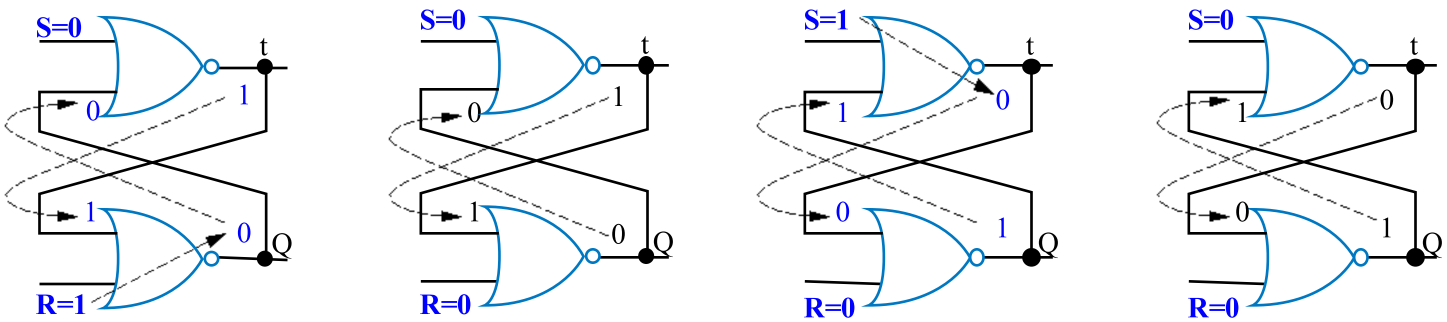
Second Attempt at Bit Storage – SR Latch

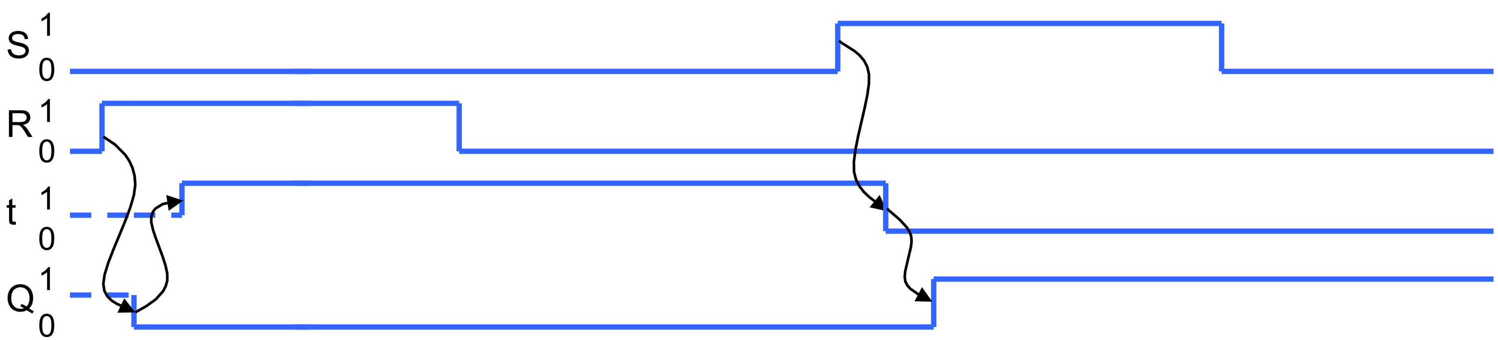
- Cross-coupled NOR gates
  - S: set (or preset) to 1
  - R: reset (or clear) to 0

Recall…





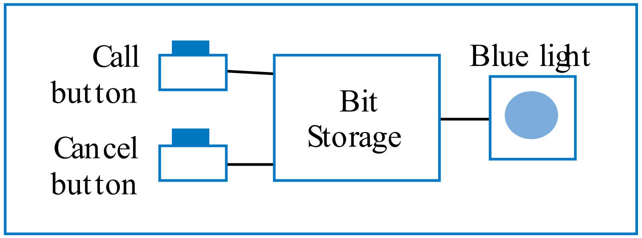


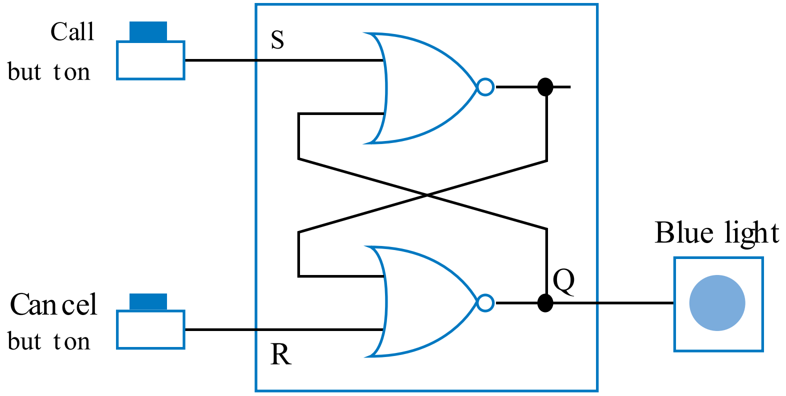


Example Using SR Latch for Bit Storage

- SR latch can serve as a bit storage, for example:
  - Call=1 : sets Q to 1
    - Q stays 1 even after Call=0
  - Cancel=1 : resets Q to 0

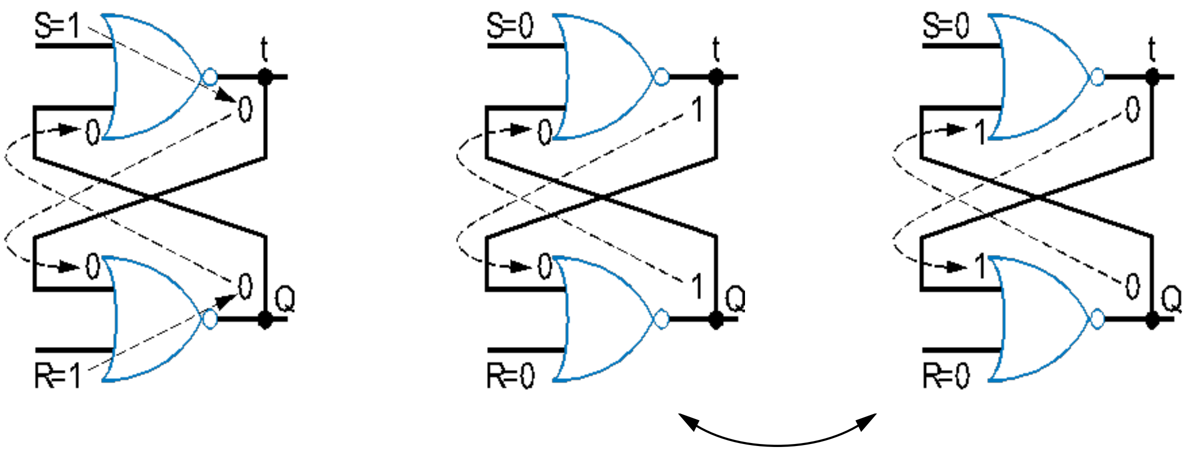
- But, there's a problem...

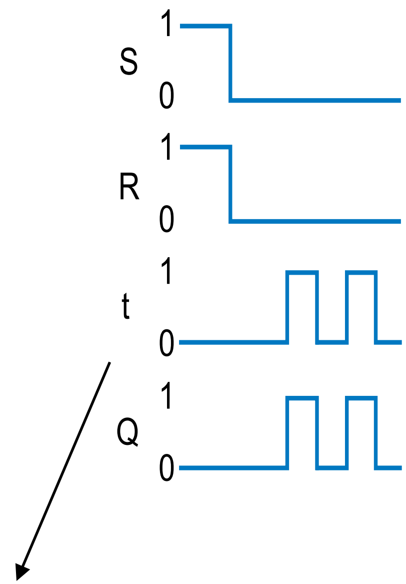




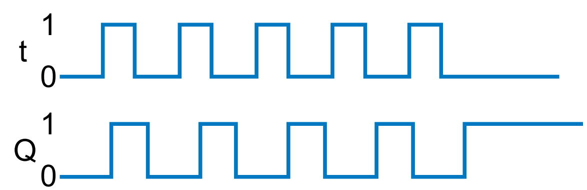
Problem with SR Latch

- Problem
  - If S=1 and R=1, we don't know what value Q will take when they both return to 0 ***simultaneously***





Q may oscillate. Then, because one path will be slightly longer than the other, Q will eventually settle to 1 or 0 – but we don't know which.



Representation of SR Latch

- When discussing latches and flip-flops, we use
  - ***present state*** to represent current value of the Q output
  - ***next state*** to represent the new value of Q output responding to the current inputs and feedback of current output
- Characteristic table

S(t)	R(t)	Q(t)	$Q(t+\Delta) \longrightarrow Q^+$	
0	0	0	0	hold
0	0	1	1	
0	1	0	0	reset
0	1	1	0	
1	0	0	1	set
1	0	1	1	
1	1	0	X	not allowed
1	1	1	X	

Q^+

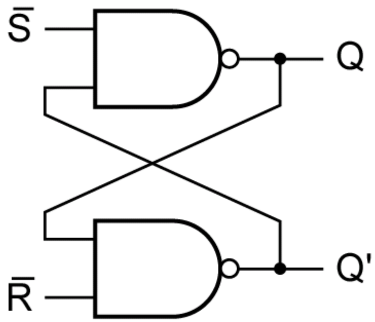
R(t) Q(t) \ S(t)	0	1
00	0	1
01	1	1
11	0	X
10	0	X

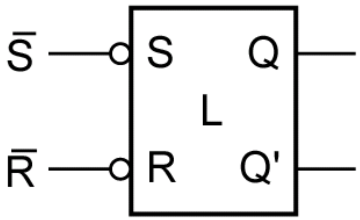
- Characteristic equation

  $Q^+ = S + R'Q$

Alternative Implementation of SR Latch

- The cross-coupled RS latch can be implemented using NAND gates



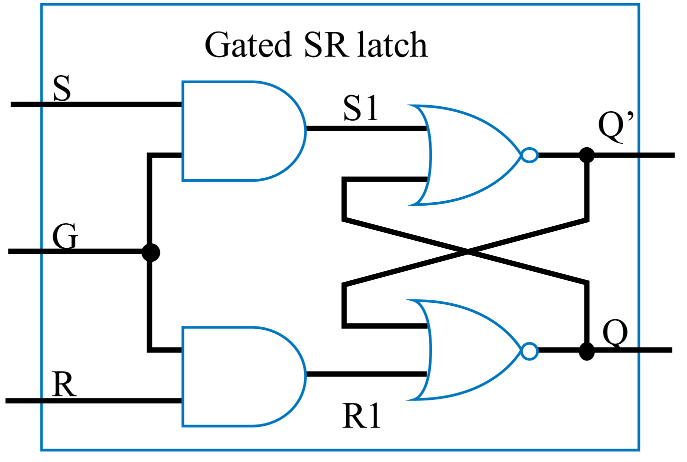


- Characteristic table

S	R	Q	Q^+	
0	0	0	X	not allowed
0	0	1	X	
0	1	0	1	set
0	1	1	1	
1	0	0	0	reset
1	0	1	0	
1	1	0	0	hold
1	1	1	1	

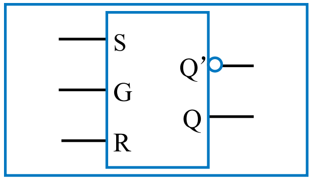
Gated SR Latch

- SR latch is enabled by a gate control signal G



Characteristic Table

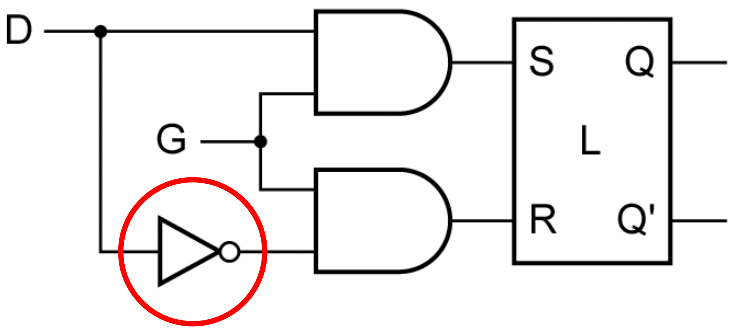
G S R	Q^+
0 x x	Q; Latch locked
1 0 0	Q; Hold state
1 0 1	0; Reset state
1 1 0	1; Set state
1 1 1	not allowed



Gated SR latch symbol

Solution to SR Latch Restriction – Gated D Latch

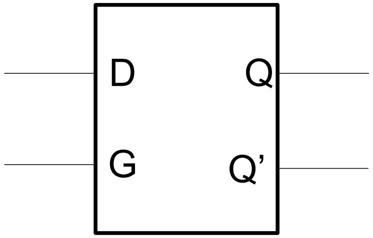
- Solution to the unstable state problem caused by S = R = 1 in SR latch



Characteristic Table

G	D	Q^+
1	0	0
1	1	1
0	X	Q

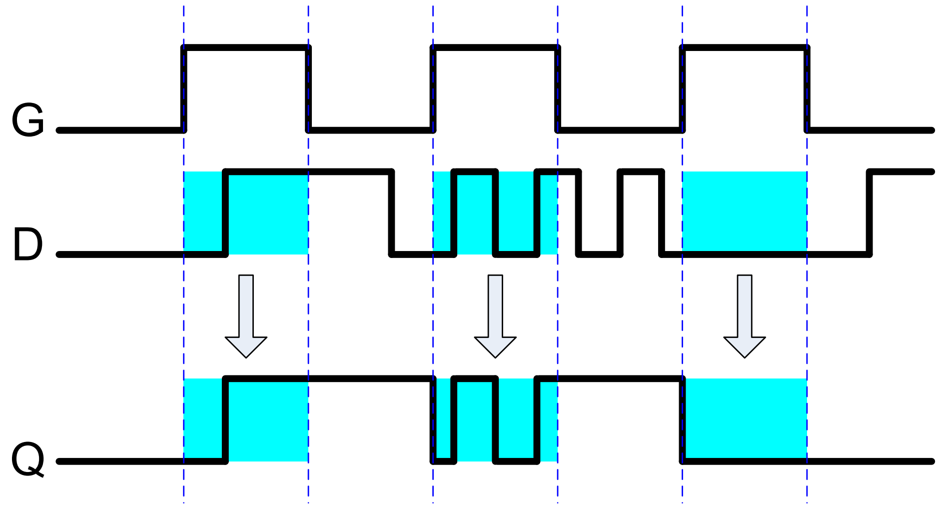
- The input value is stored into the latch only when gate control G has high level – **Level Sensitive**



D latch symbol

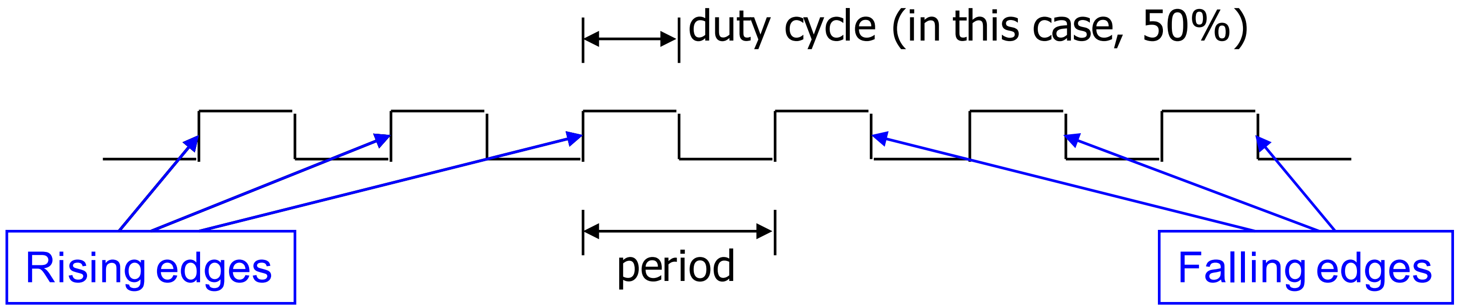
Gated D Latch – Transparent Latch

- Properties of the D latch
  - D latch is used as a **temporary storage** for a bit
  - The binary information at the data input of the D latch is **copied** to the Q output when the control input G is high (or enabled)
  - The output Q follows changes on the data input D as long as the control input G is enabled, so called a **transparent** latch



A Typical Control Input – Clock Signal

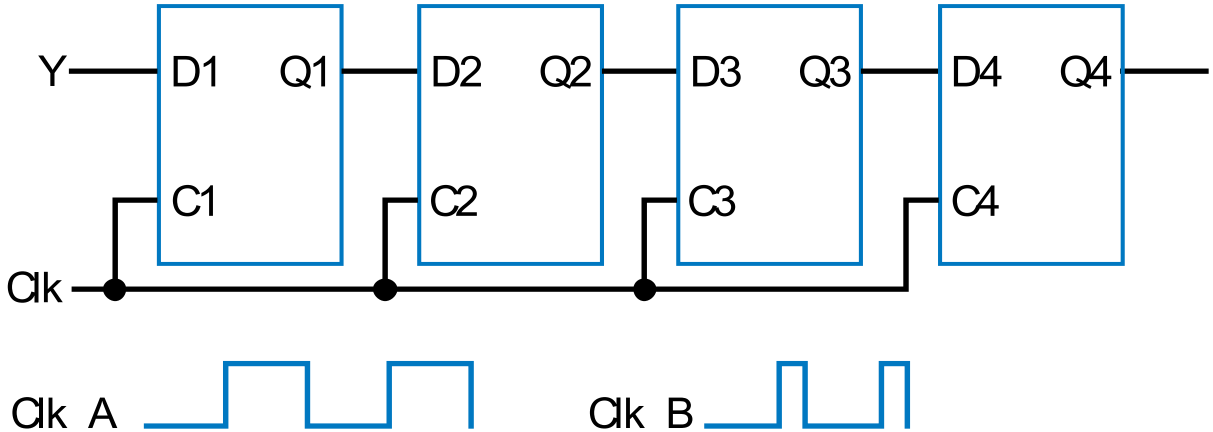
- Periodic pulse train used in sequential circuit to synchronize circuit behaviors



  - ***Clock period***: time interval between pulses
  - ***Clock cycle***: one such time interval
  - ***Clock frequency***: 1/period

Freq	Period
100 GHz	0.01 ns
10 GHz	0.1 ns
1 GHz	1 ns
100 MHz	10 ns
10 MHz	100 ns

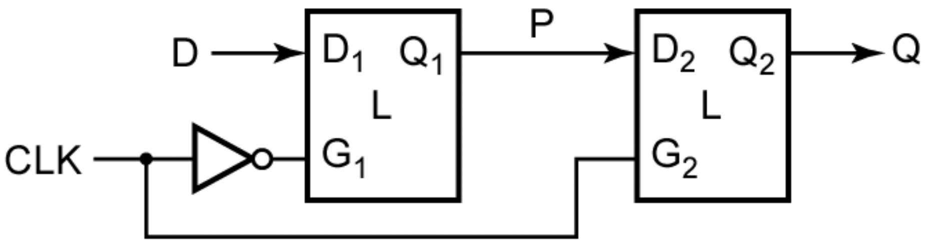
Problem with Level-Sensitive D Latch

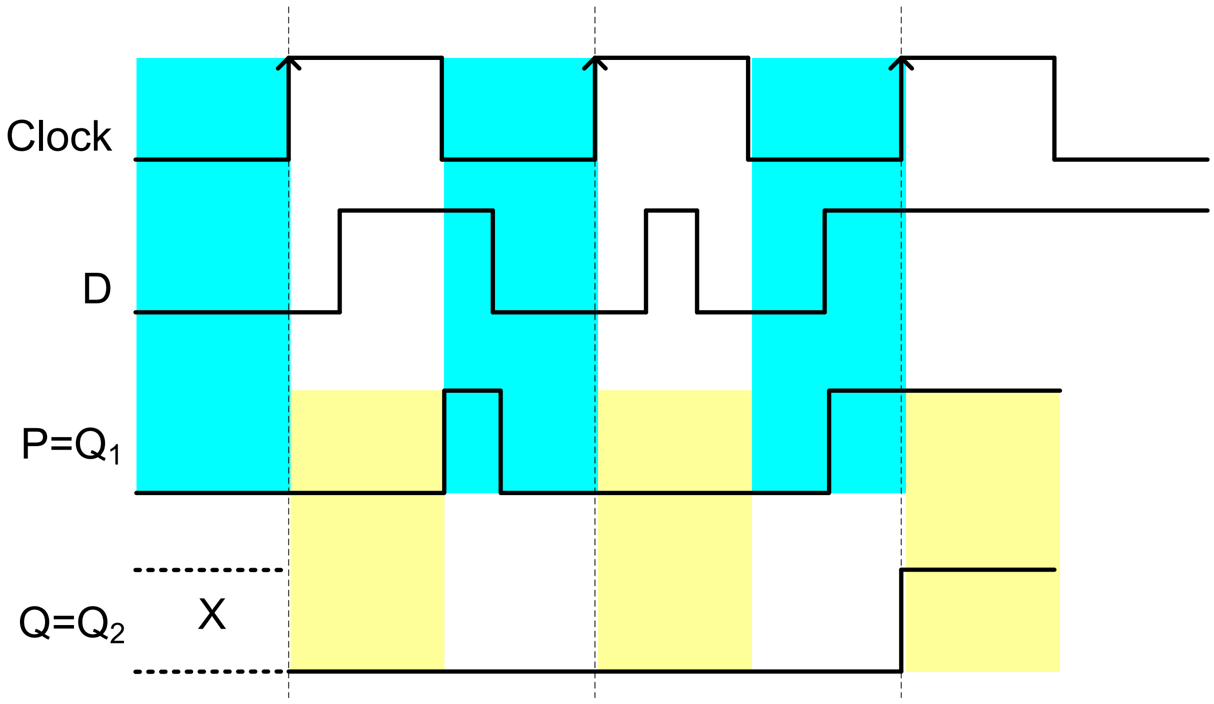


- D latch still has problem
  - When Clk=1, through how many latches will a signal travel?
  - Depends on for how long Clk=1
    - Clk_A -- signal may travel through multiple latches
    - Clk_B -- signal may travel through fewer latches
  - Hard to control the storage elements

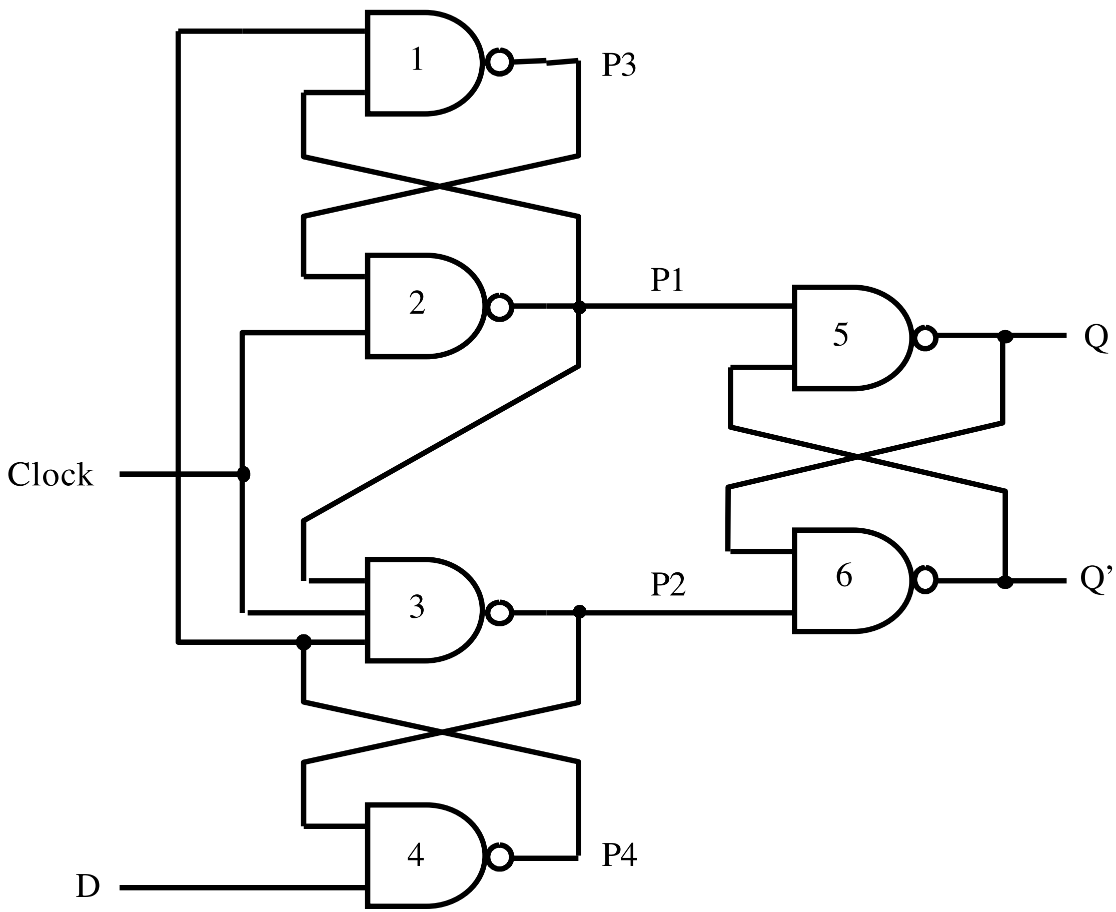
Rising-Edge Triggered D Flip Flop

- Rising-edge triggered Master-Slave D flip flop

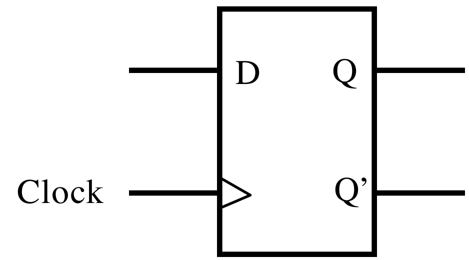




Alternative Implementation of Rising-Edge Triggered D Flip Flop



Alternative Implementation

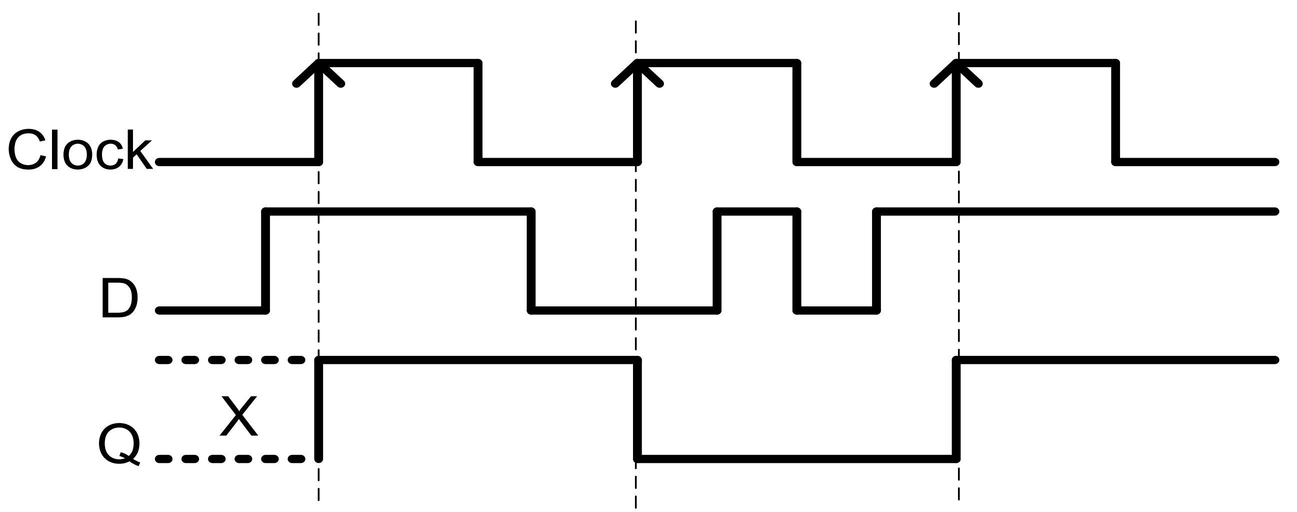


clock	D	Q^+
↑	0	0
↑	1	1
0	X	Q
1	X	Q

Characteristic equation:
$Q^+ = D$ (at active clock edges)

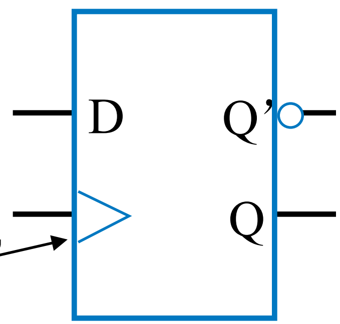
Rising-Edge Triggered D Flip Flop

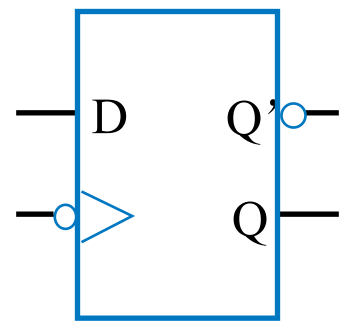
- Properties of the rising edge triggered D Flip Flop
  - The output changes only at the rising edges of the clock signal – **Edge Sensitive**
  - The output Q gets the value of input D at the time point of rising edge of clock



Symbols for D Flip-Flop

The triangle means clock input, edge triggered



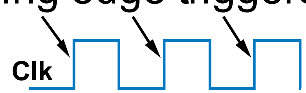


Internal design: Just invert servant clock rather than master

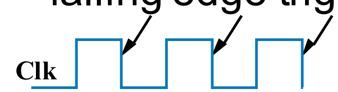
Symbol for rising-edge triggered D flip-flop

Symbol for falling-edge triggered D flip-flop

rising edge triggered

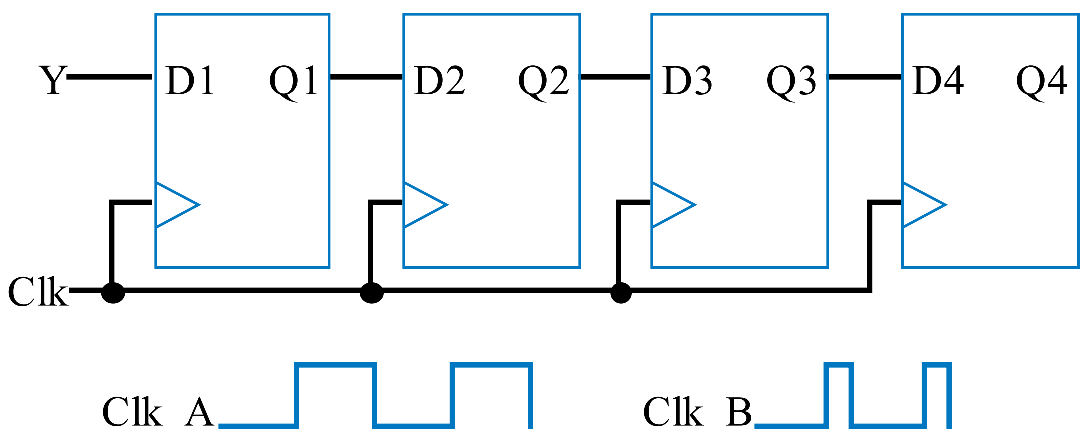


falling edge triggered



Application of D Flip-Flop

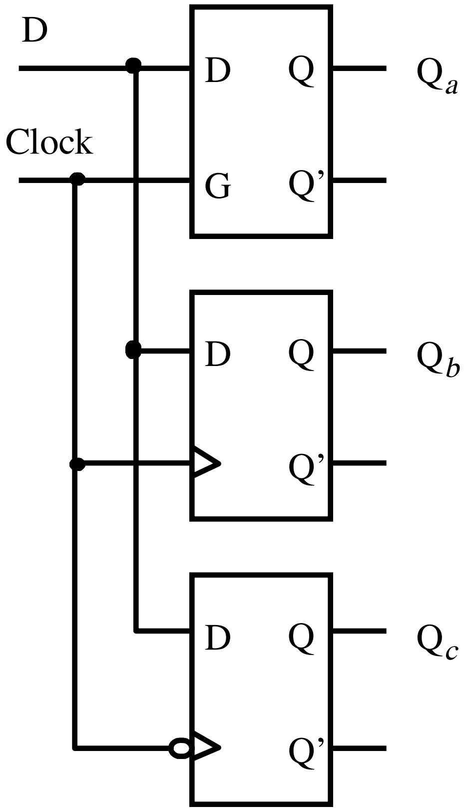
- Solves problem of concatenated D latches when G=1
  - In figure below, signal travels through exactly one flip-flop, for either Clk_A or Clk_B
  - On each rising edge of Clk, all four flip-flops are loaded simultaneously, doesn't matter how long Clk is 1.

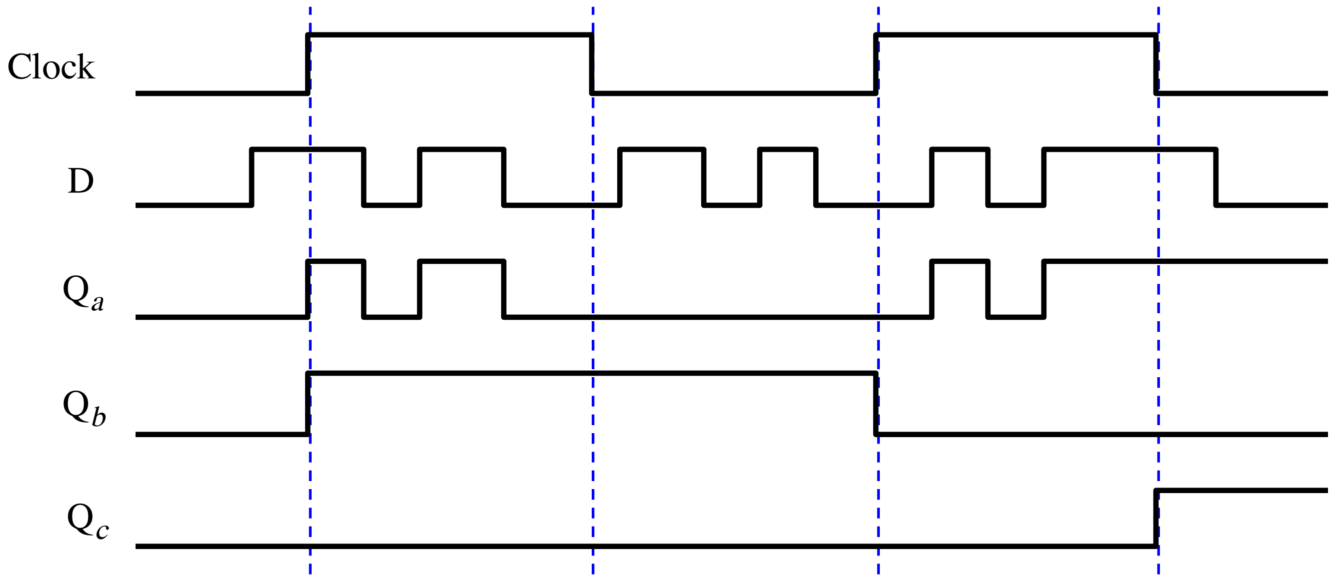


Flip Flop vs. Latch

- **Both are** storage elements **in sequential circuits**
- **Flip flop**
  - **edge-sensitive**, the input matters only at active edges (rising or falling)
  - behaviors are **synchronous** to the clock signal
- **Latch**
  - **level-sensitive**, the input matters whenever control has active level (high or low)
  - behaviors are **asynchronous** to the clock signal

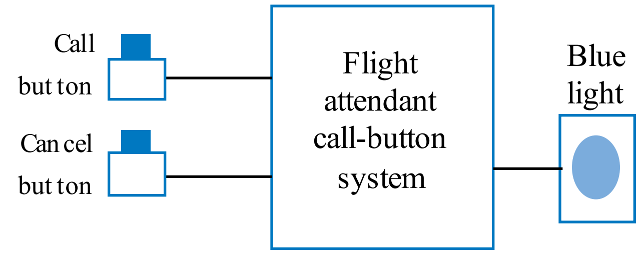
Flip Flop vs. Latch





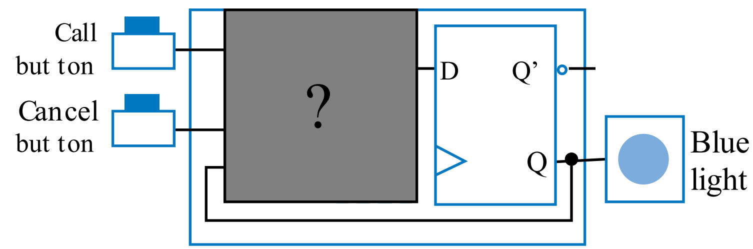
Flight-Attendant Call Button Using D Flip-Flop

- D flip-flop will store bit
- Inputs are Call, Cancel, and present output Q of D flip-flop
- Truth table

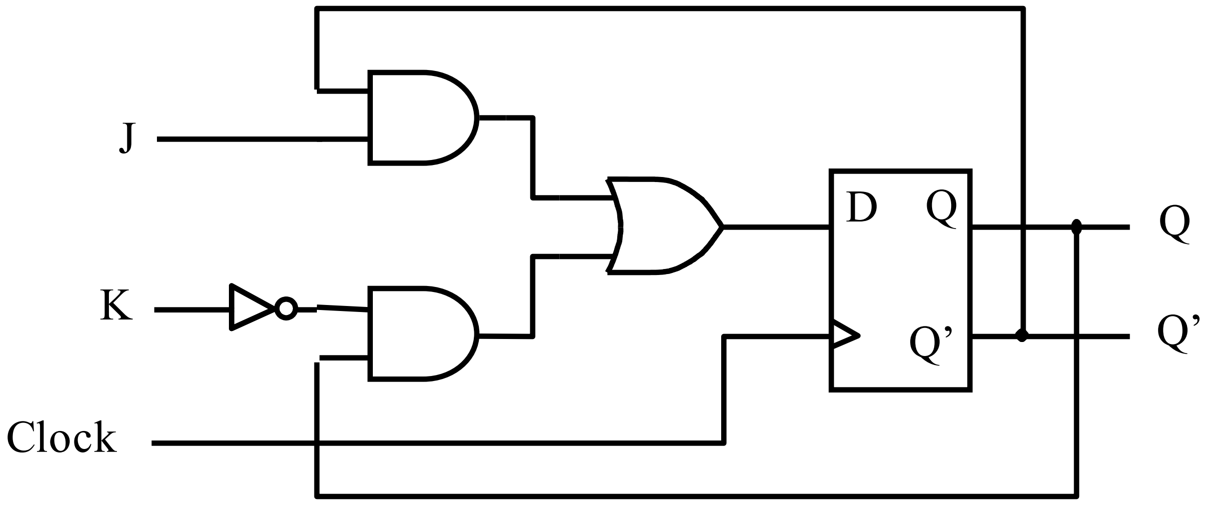


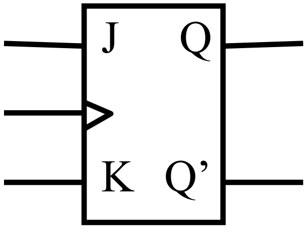
Call	Cancel	Q	$Q^+ = D$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Circuit derived from truth table, using Chapter 2 combinational logic design process



Rising Edge-triggered J-K Flip Flop

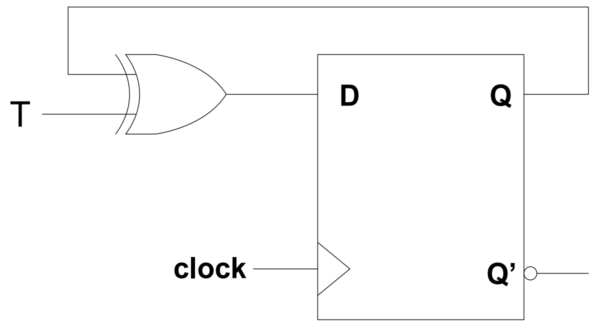




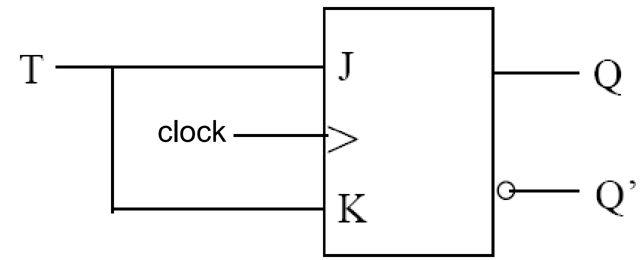
J	K	Q^+
0	0	Q
0	1	0
1	0	1
1	1	Q'

Characteristic equation:
$Q^+ = JQ' + K'Q$

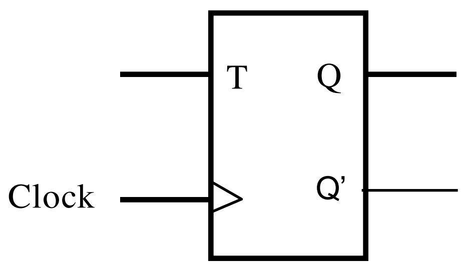
Rising Edge-triggered T Flip Flop



Implemented with D ff

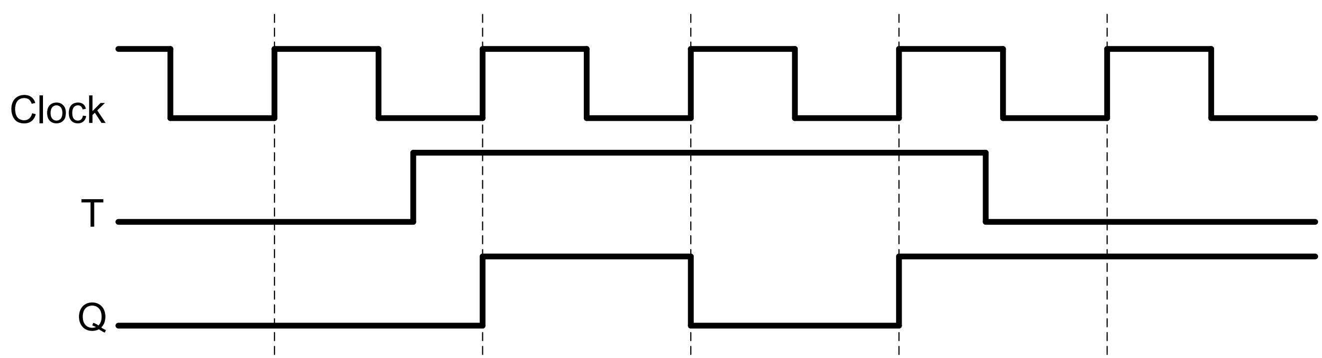


Implemented with JK ff



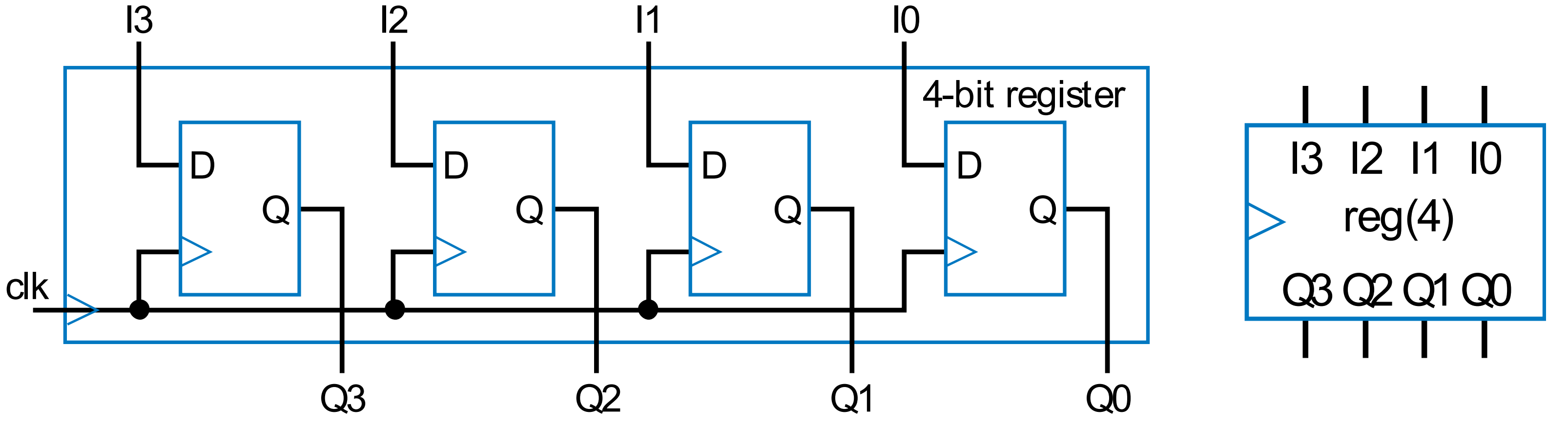
clock	T	Q^+
↑	0	Q
↑	1	Q'

Characteristic equation:
$Q^+ = T'Q + TQ' = T \oplus Q$



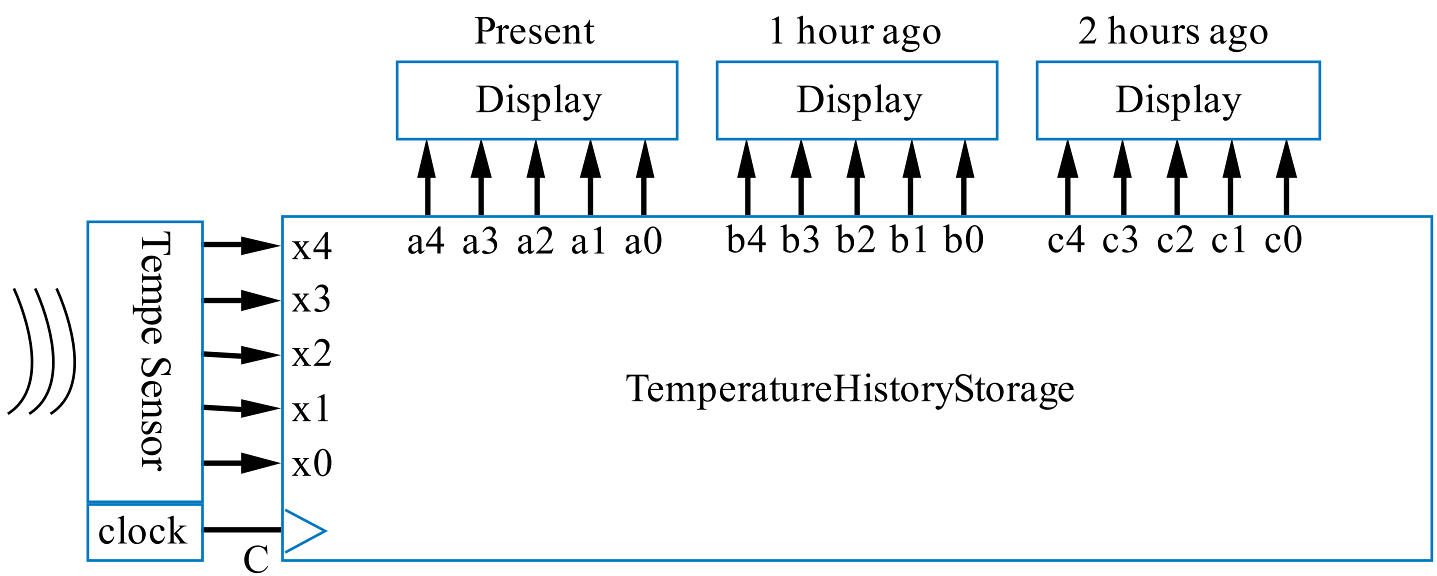
Basic Register

- Typically, we store multi-bit items
  - e.g., storing a 4-bit binary number
- ***Register***: multiple flip-flops sharing clock signal



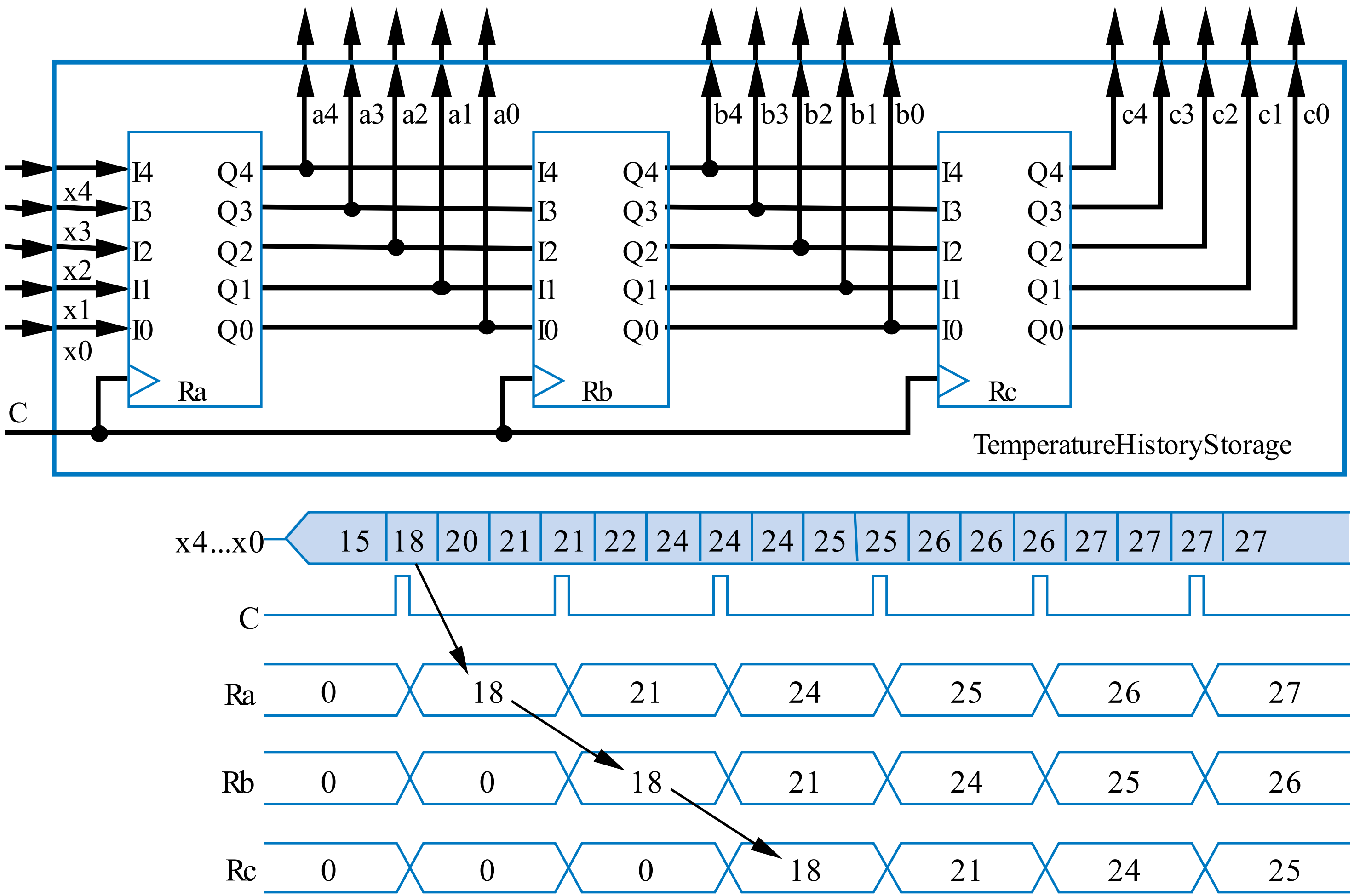
Example Using Registers: Temperature Display

- Temperature history display
  - Sensor outputs temperature as 5-bit binary number
  - Timer pulses C every hour
  - Record temperature on each pulse, display last three recorded values

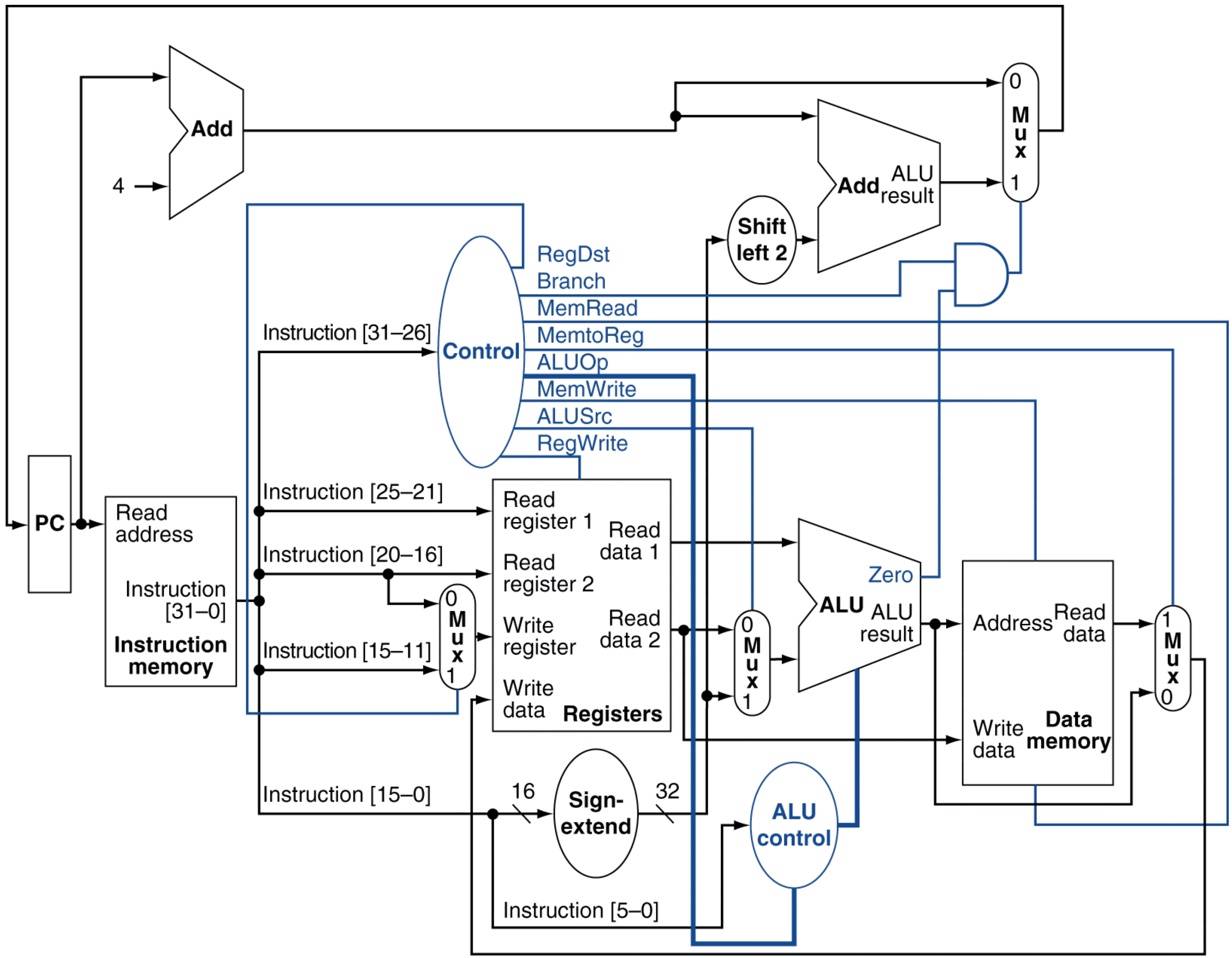


Example Using Registers: Temperature Display

- Use three 5-bit registers

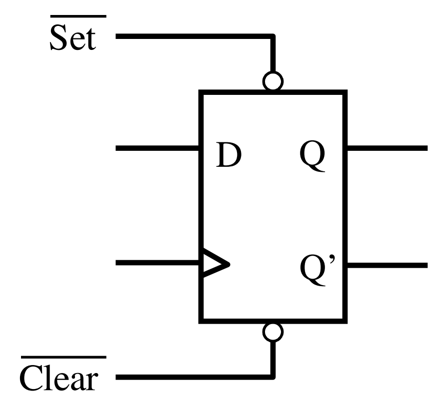


Big Picture – Simplified CPU



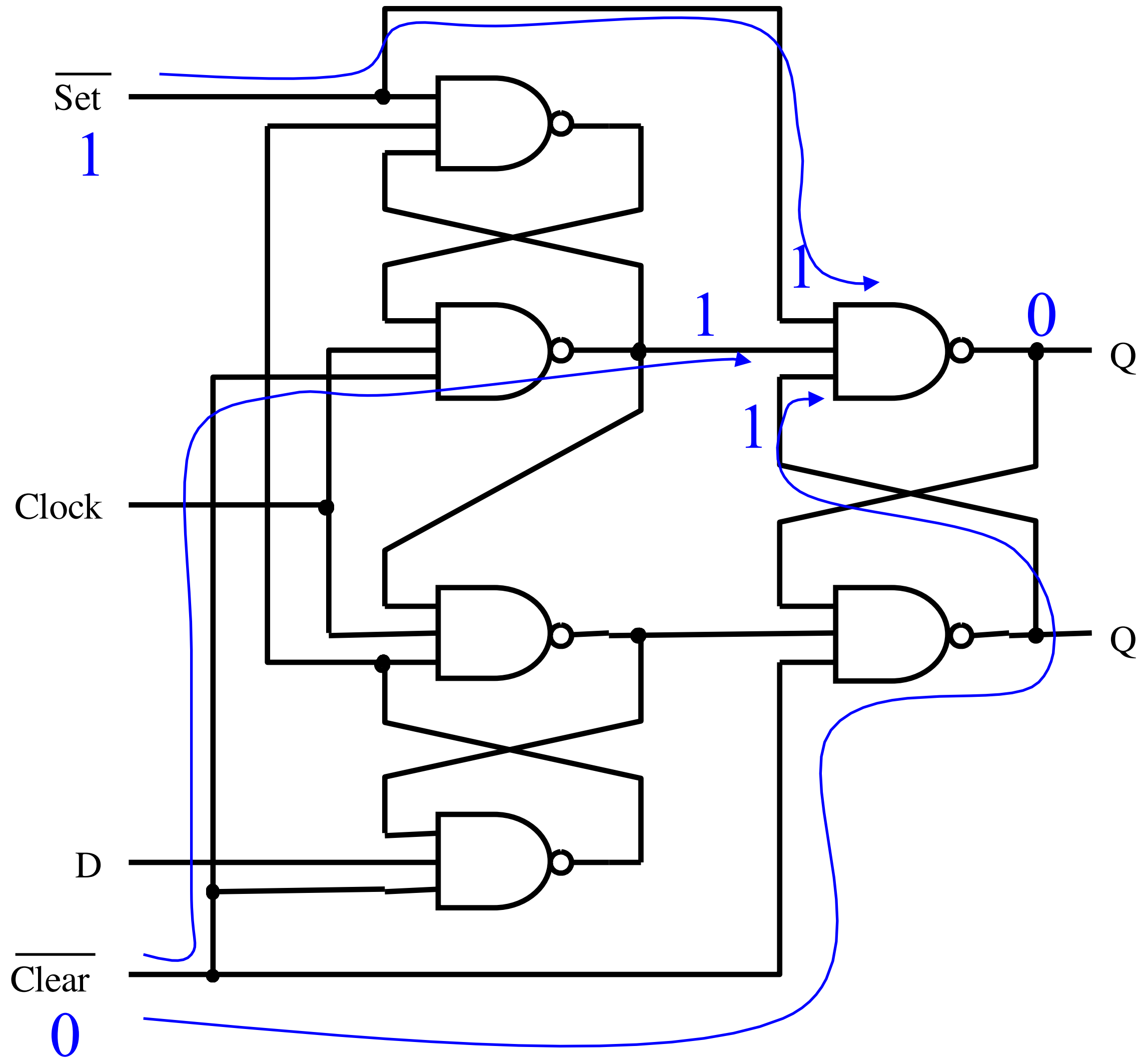
Control Inputs for Flip Flops

- Asynchronous:
  – control signals do not depend on the clock signal
- Synchronous:
  – control signals depend on the clock signal
- Active low:
  – It controls when it's low
- Active high:
  – It controls when it's high



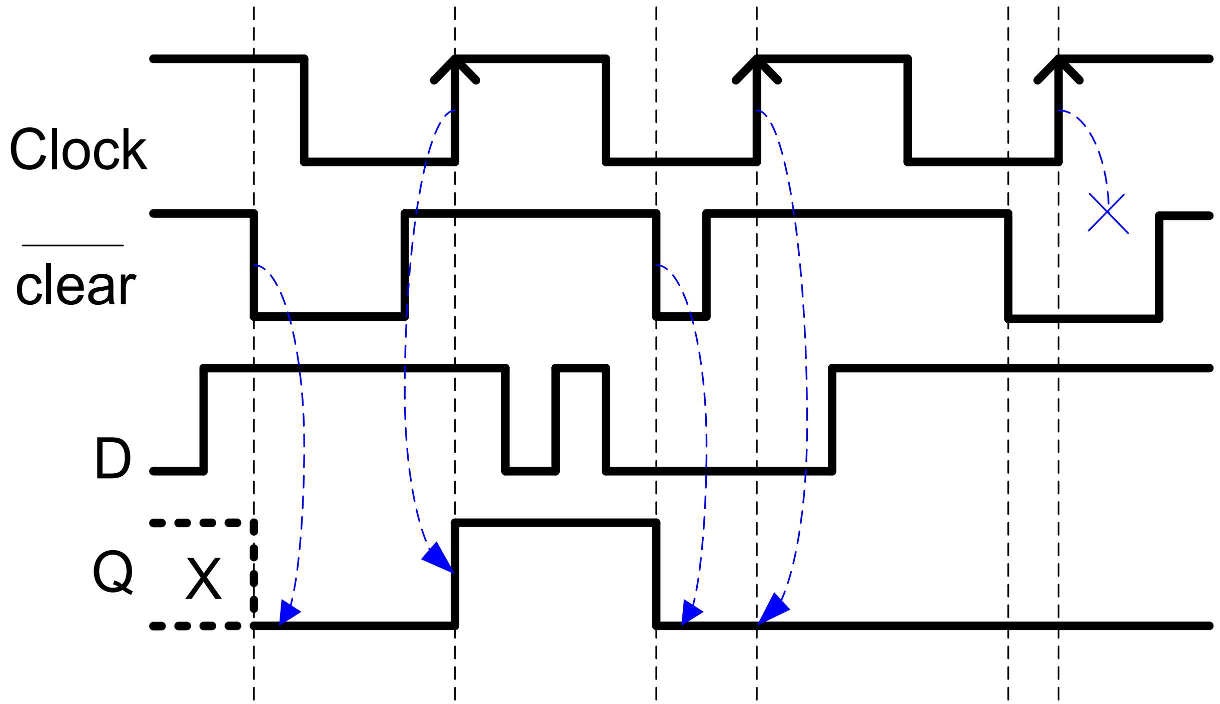
Implementation of Asynchronous Control Input

Control signals decides the output value directly independent of the clock signal



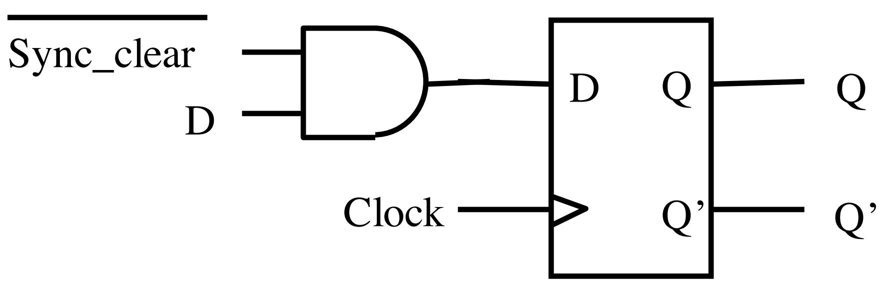
Asynchronous Control Input

- D flip flop with active low asynchronous Clear



Implementation of Synchronous Control Input

- Synchronous Clear
  - control signal depends on the active edge (either rising or falling) of the clock signal



Flip-Flops with Control Inputs

- D flip flop with active low synchronous Clear