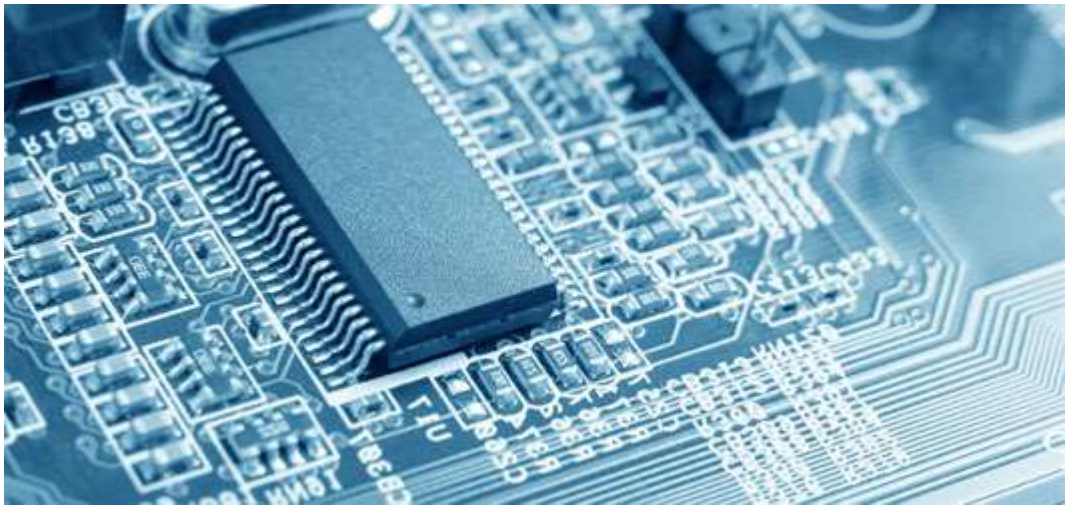
NAZARBAYEV UNIVERSITY
SCHOOL OF SCIENCE AND TECHNOLOGY



ROBT206 – Microcontrollers with Lab

Lecture 13 – Sequential Logic

1 March, 2018

Topics

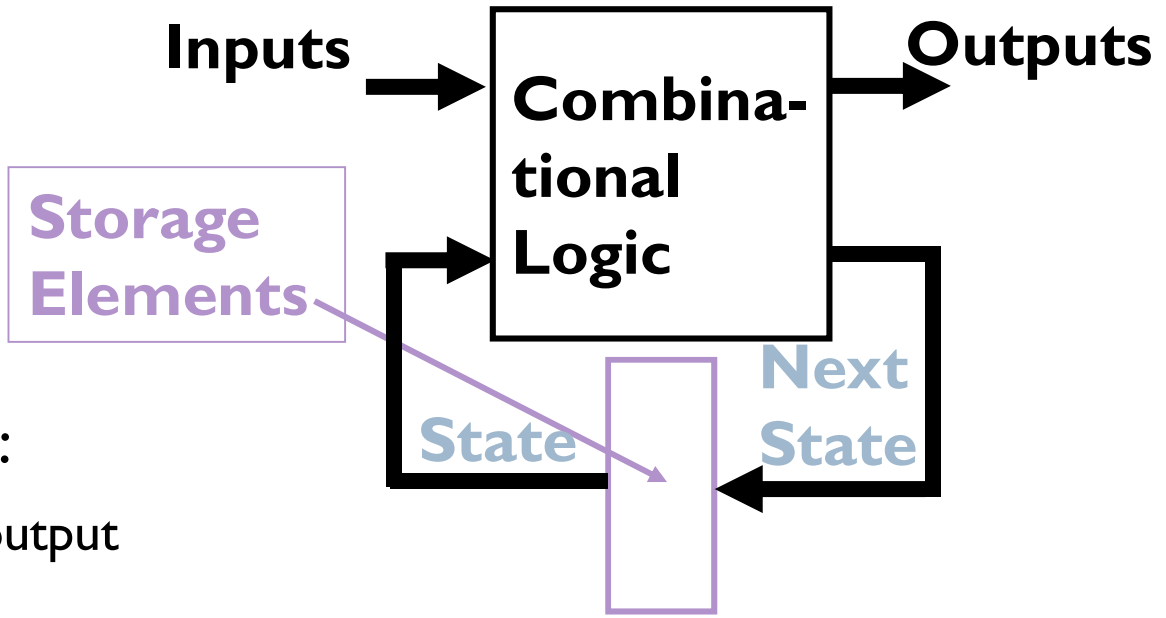
Today's Topics

Storage Elements and Analysis

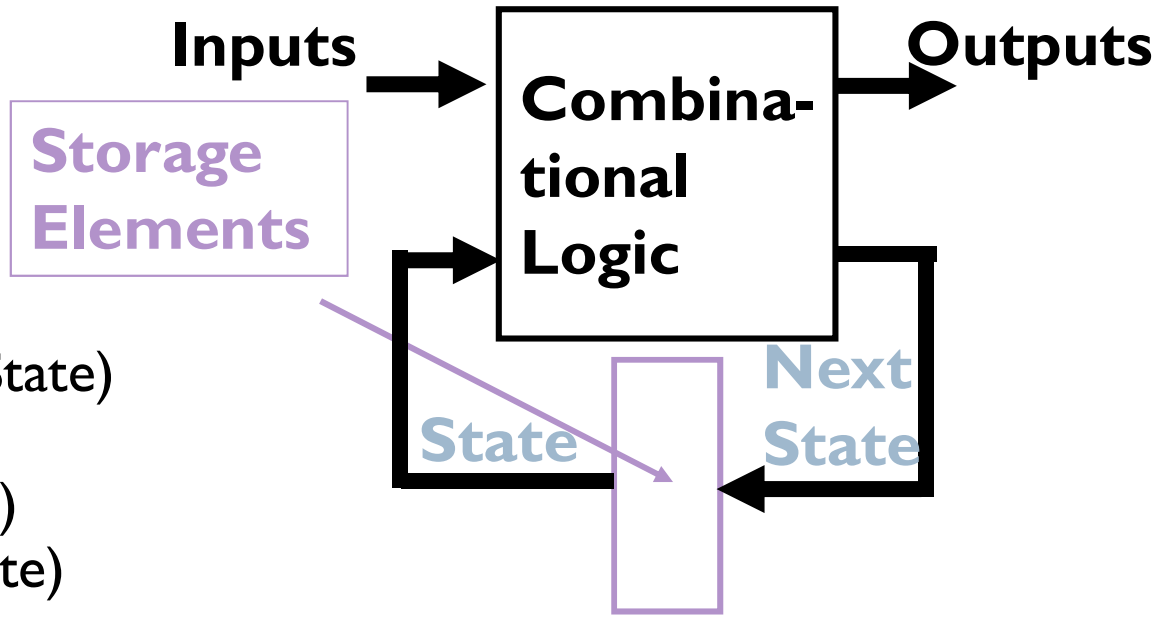
- Introduction to sequential circuits
- Types of sequential circuits
- Storage elements
  - Latches
  - Flip-flops

Introduction to Sequential Circuits

- A Sequential circuit contains:
  - Storage elements: Latches or Flip-Flops
  - Combinational Logic:
    - Implements a multiple-output switching function
    - Inputs are signals from the outside.
    - Outputs are signals to the outside.
    - Other inputs, State or Present State, are signals from storage elements.
    - The remaining outputs, Next State are inputs to storage elements.



Introduction to Sequential Circuits



- Combinational Logic
  - *Next state function*
    Next State = f(Inputs, State)
  - *Output function* (Mealy)
    Outputs = g(Inputs, State)
  - *Output function* (Moore)
    Outputs = h(State)
- Output function type depends on specification and affects the design significantly

Types of Sequential Circuits

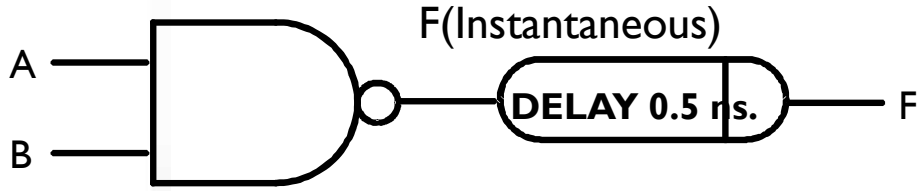
- Depends on the times at which:
  - storage elements observe their inputs, and
  - storage elements change their state
- **Synchronous**
  - Behavior defined from knowledge of its signals at discrete instances of time
  - Storage elements observe inputs and can change state only in relation to a timing signal (clock pulses from a clock)
- **Asynchronous**
  - Behavior defined from knowledge of inputs an any instant of time and the order in continuous time in which inputs change
  - If clock just regarded as another input, all circuits are asynchronous!
  - Nevertheless, the synchronous abstraction makes complex designs tractable!

Discrete Event Simulation

- In order to understand the time behavior of a sequential circuit we use **discrete event simulation**.
- Rules:
  - Gates modeled by an ideal (instantaneous) function and a fixed gate delay
  - Any change in input values is evaluated to see if it causes a change in output value
  - Changes in output values are scheduled for the fixed gate delay after the input change
  - At the time for a scheduled output change, the output value is changed along with any inputs it drives

Simulated NAND Gate

- Example: A 2-Input NAND gate with a 0.5 ns. delay:

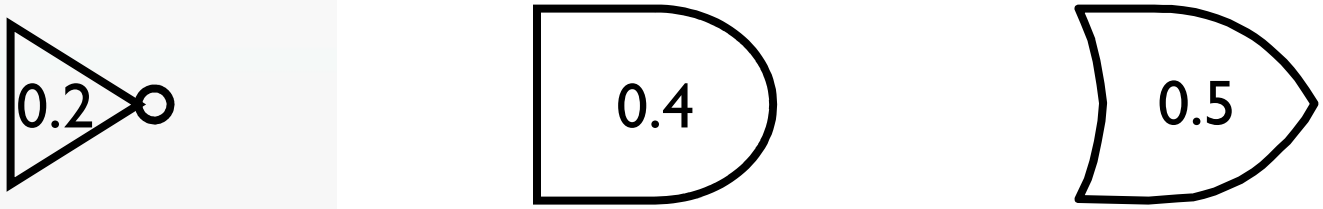


- Assume A and B have been 1 for a long time
- At time t=0, A changes to a 0 at t= 0.8 ns, back to 1.

t (ns)	A	B	F(I)	F	Comment
$-\infty$	1	1	0	0	A=B=1 for a long time
0	$1 \Rightarrow 0$	1	$1 \Leftarrow 0$	0	F(I) changes to 1
0.5	0	1	1	$1 \Leftarrow 0$	F changes to 1 after a 0.5 ns delay
0.8	$1 \Leftarrow 0$	1	$1 \Rightarrow 0$	1	F(Instantaneous) changes to 0
0.13	1	1	0	$1 \Rightarrow 0$	F changes to 0 after a 0.5 ns delay

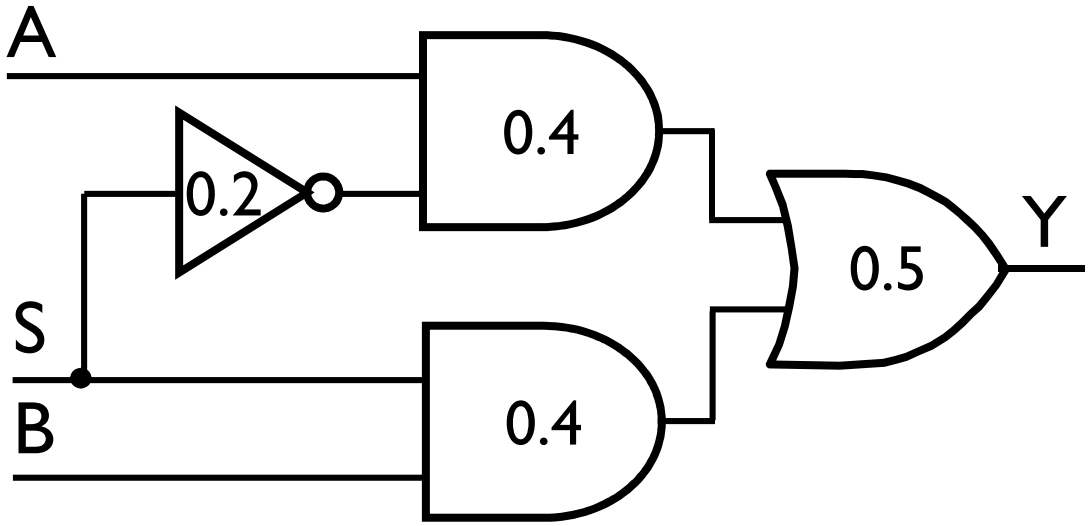
Gate Delay Models

- Suppose gates with delay *n* ns are represented for *n* = 0.2 ns,
  n = 0.4 ns,
  n = 0.5 ns, respectively:



Circuit Delay Model

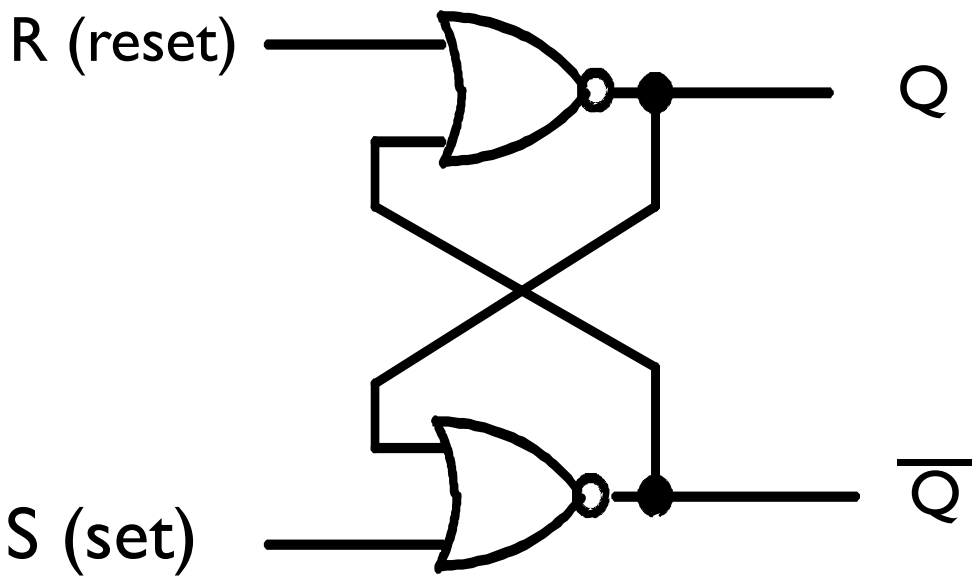
- Consider a simple 2-input multiplexer:
- With function:
  - Y = A for S = 0
  - Y = B for S = 1



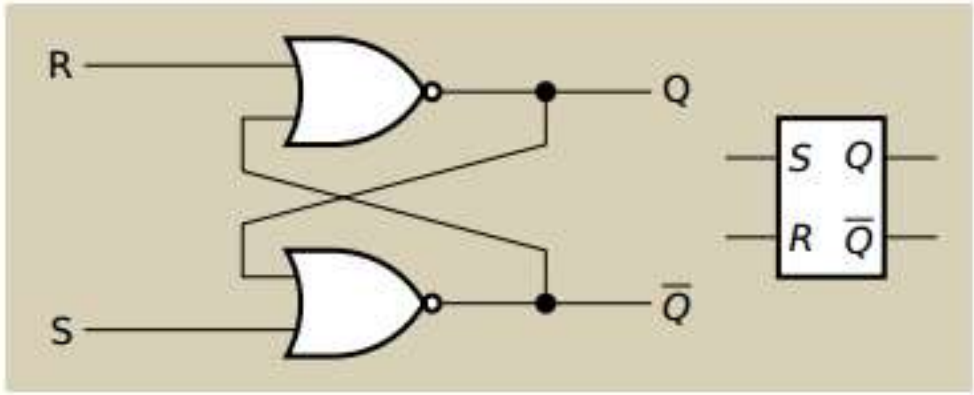
- "Glitch" is due to delay of inverter

Basic (NOR) S – R Latch

- Cross-coupling two NOR gates gives the S – R Latch:
- Which has the time sequence behavior:

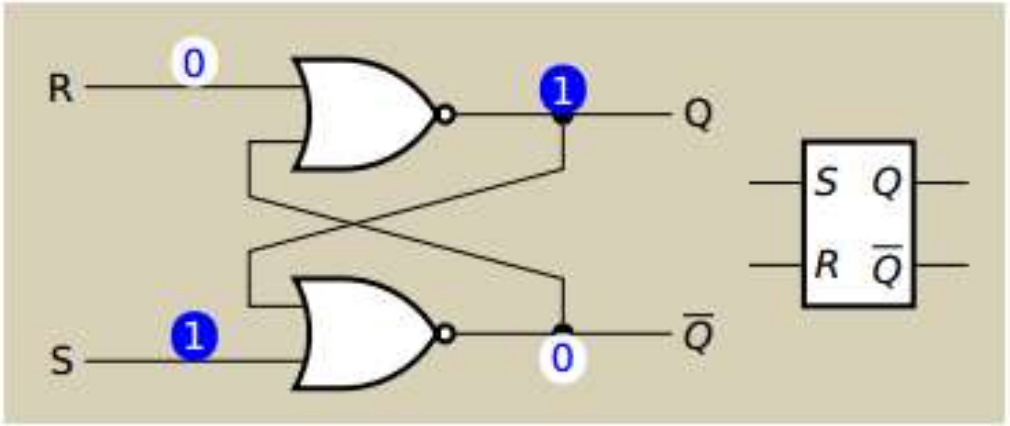


Basic (NOR) S – R Latch



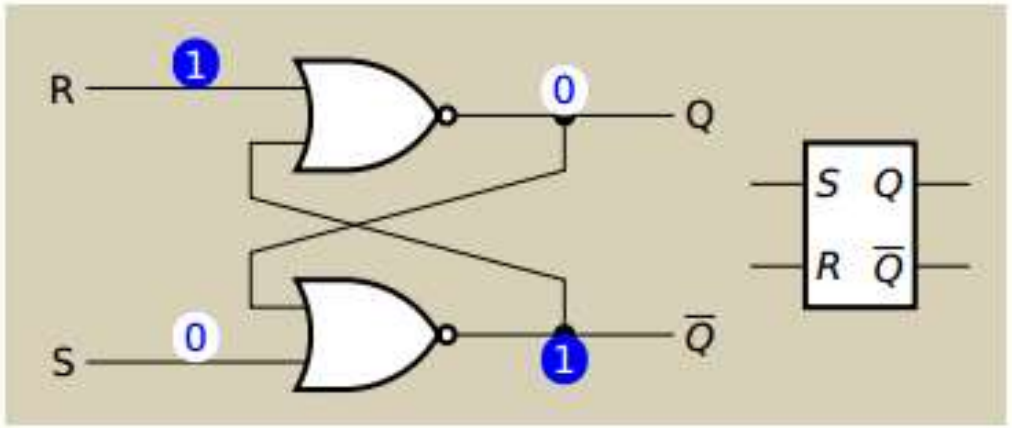
R	S	Q	$\overline{Q}$
0	0		
0	1		
1	0		
1	1		

Basic (NOR) S – R Latch



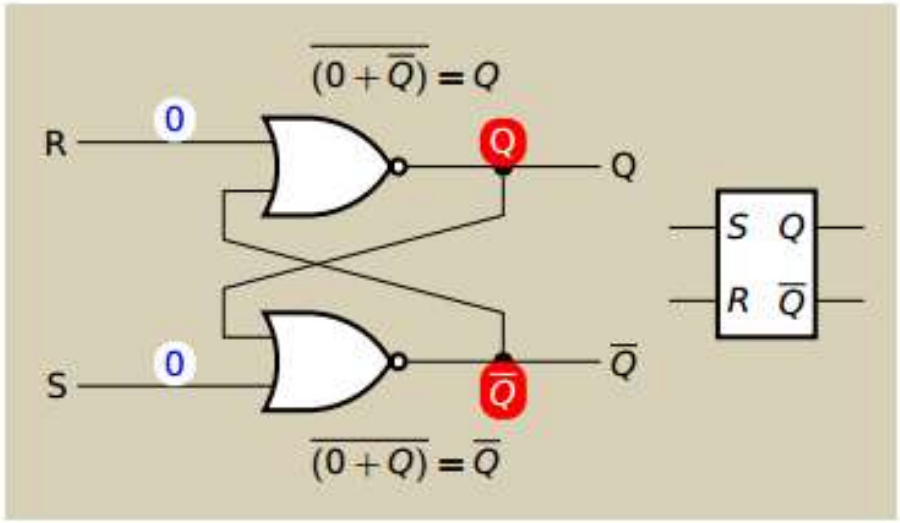
R	S	Q	$\overline{Q}$	
0	0			
0	1	1	0	Set ($Q = 1$)
1	0			
1	1			

Basic (NOR) S – R Latch



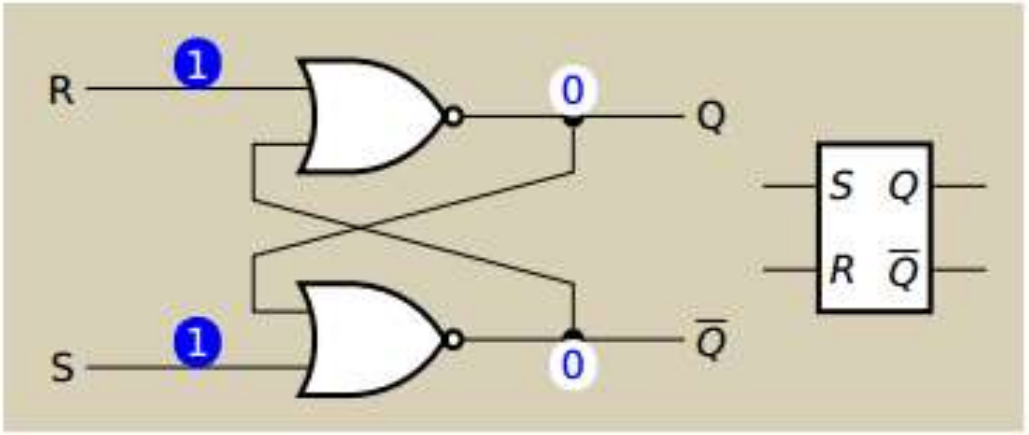
R	S	Q	$\overline{Q}$	
0	0			
0	1	1	0	Set ($Q = 1$)
1	0	0	1	Reset ($Q = 0$)
1	1			

Basic (NOR) S – R Latch



R	S	Q	$\overline{Q}$	
0	0	Q	$\overline{Q}$	Hold previous value
0	1	1	0	Set ($Q = 1$)
1	0	0	1	Reset ($Q = 0$)
1	1			

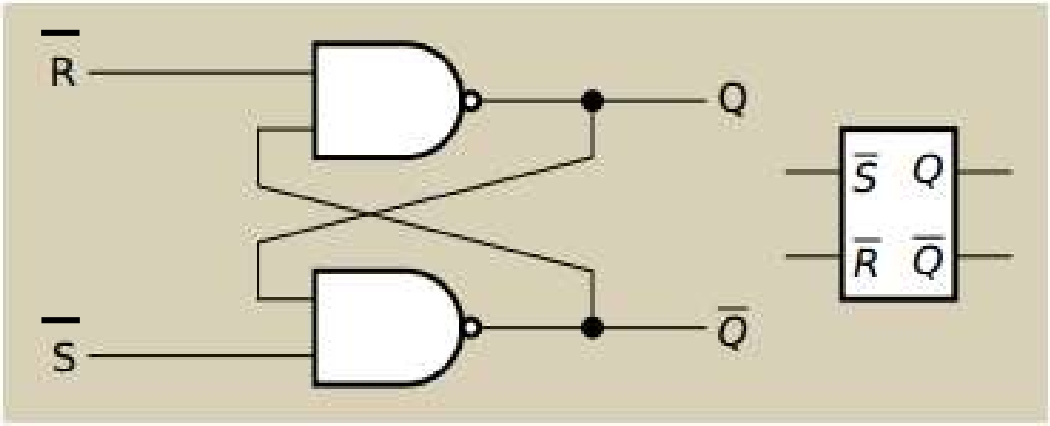
Basic (NOR) S – R Latch



R	S	Q	$\overline{Q}$	
0	0	Q	$\overline{Q}$	Hold previous value
0	1	1	0	Set ($Q = 1$)
1	0	0	1	Reset ($Q = 0$)
1	1	0	0	Bad. Do not use.

Basic (NAND) $\bar{S}$ – $\bar{R}$ Latch

- "Cross-Coupling" two NAND gates gives $\bar{S}$ - $\bar{R}$ latch:
- Which has the time sequence behavior:

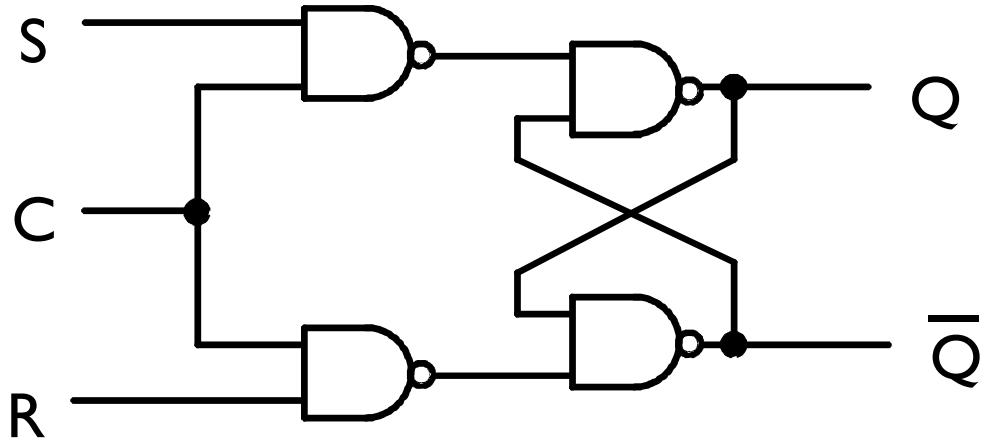


- S = 0, R = 0 is forbidden as input pattern

$\bar{R}$	$\bar{S}$	Q	$\bar{Q}$	
0	0	1	1	Bad. Do not use.
0	1	0	1	Reset ($Q = 0$)
1	0	1	0	Set ($Q = 1$)
1	1	Q	$\bar{Q}$	Hold previous value

Clocked S - R Latch

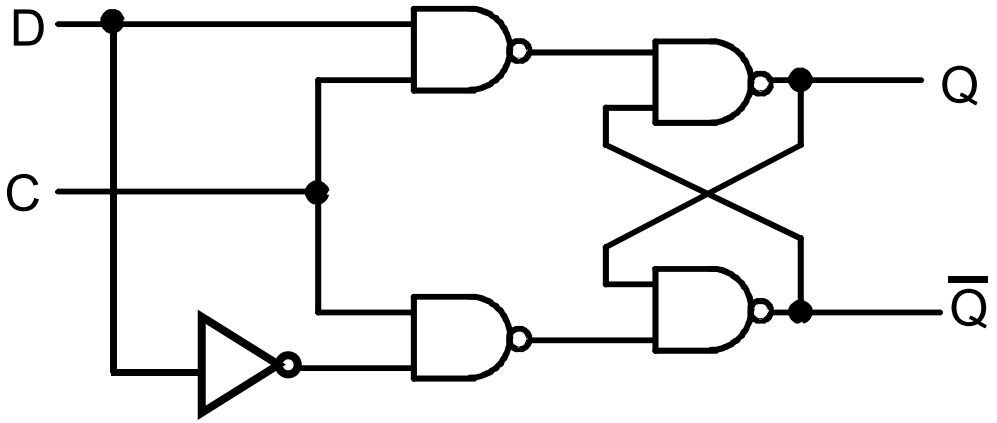
- Adding two NAND gates to the basic $\overline{S}$ - $\overline{R}$ NAND latch gives the clocked S – R latch:



- Has a time sequence behavior similar to the basic S-R latch except that the S and R inputs are only observed when the line C is high.
- C means "control" or "clock".

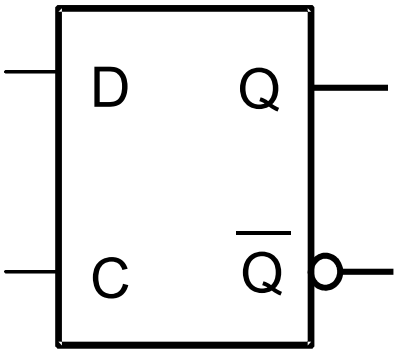
D Latch

- Adding an inverter to the S-R Latch, gives the D Latch:
- Note that there are no "indeterminate" states!



C	D	Q	$\overline{Q}$	
0	X	Q	$\overline{Q}$	No change
1	0	0	1	Reset state
1	1	1	0	Set state

The graphic symbol for a D Latch is:



Flip-Flops

- The latch timing problem
- Edge-triggered flip-flop
- Standard symbols for storage elements
- Direct inputs to flip-flops

The Latch Timing Problem

Latches are circuits that can store "state"
- set the latch to a value (0 or 1)
- put the latch in a "same value" mode to hold the value

To do complicated computations
- intermediate "state" must be maintained
- various steps of the computation must be coordinated

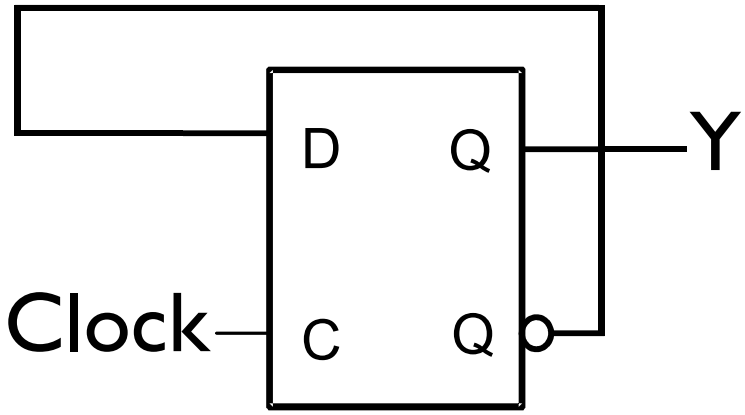
Q: How to coordinate computations and the changing of state values across lots of different parts of a circuit

A: Introduce a clocking mechanism
- each clock pulse, combinational computations can be performed, results stored (in latches)

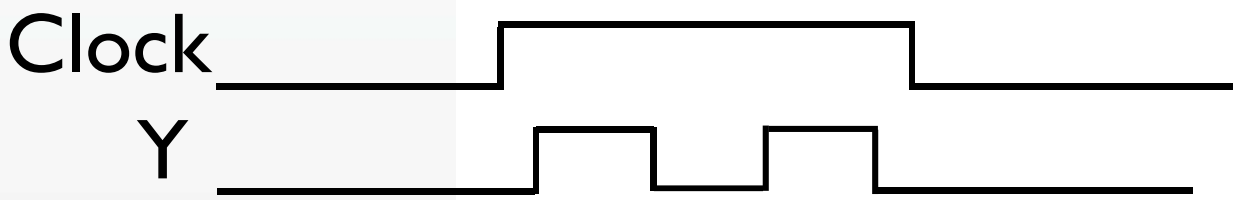
Q: How to introduce clocks into latches?

The Latch Timing Problem (continued)

- Consider the following circuit:



- Suppose that initially Y = 0.

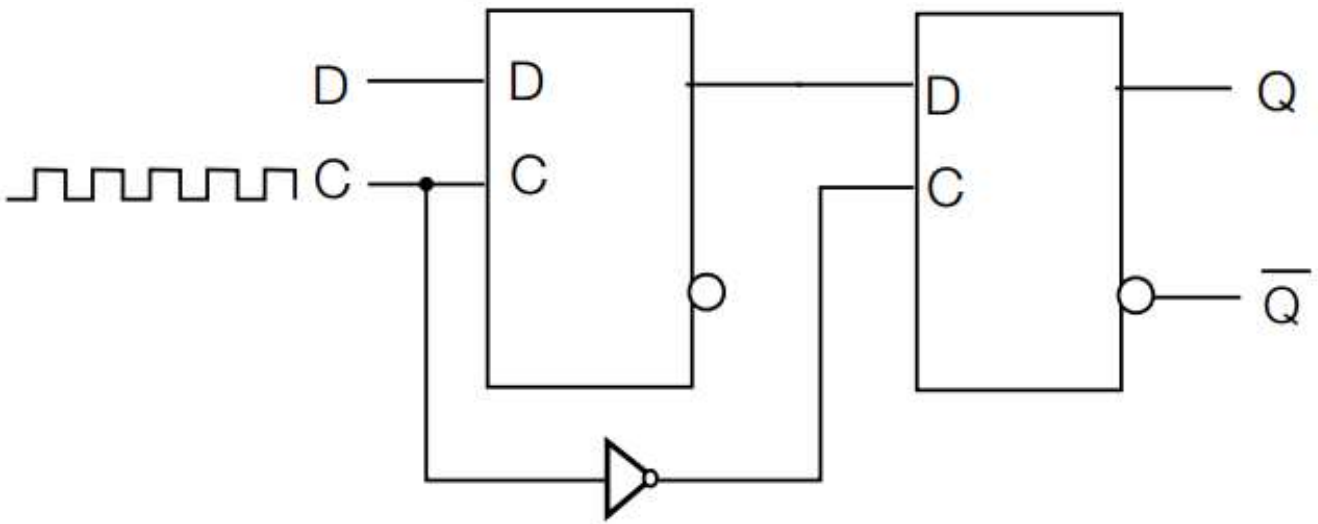


- As long as C = 1, the value of Y continues to change!
- The changes are based on the delay present on the loop through the connection from Y back to Y.
- This behavior is clearly unacceptable.
- Desired behavior: Y changes only once per clock pulse
- **Flip flops are designed so that outputs will not change within a single clock pulse**

The Latch Timing Problem (continued)

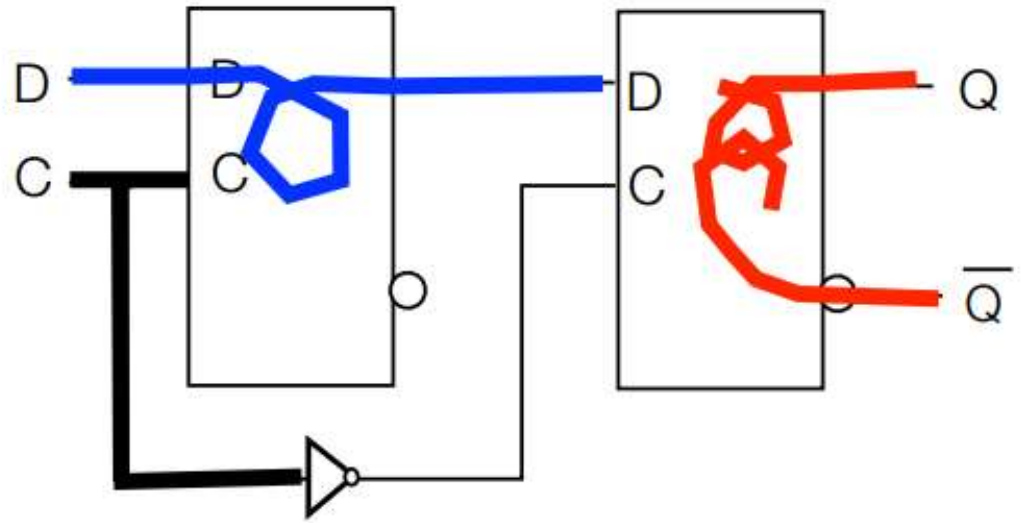
- A solution to the latch timing problem is to break the closed path from Y to Y within the storage element
- The commonly-used, path-breaking solutions replace the clocked D-latch with an edge-triggered flip-flop
- An edge-triggered flip-flop ignores the pulse while it is at a constant level and triggers only during a **transition** of the clock signal
- Edge-triggered flip-flops can be built directly at the electronic circuit level

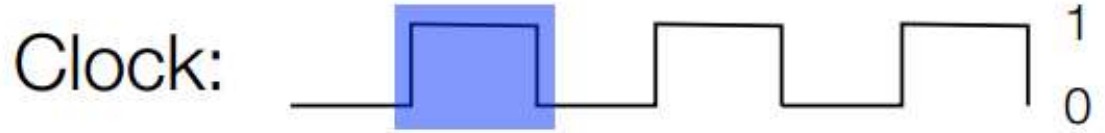
Edge-Triggered D Flip-Flop



- C (Control) is fed a clock pulse (alternates between 0 and 1 with fixed period)
  - C=1: Master latch "on", Slave latch "off"
    - New D input read into master
    - Previous Q values still emitted (not affected by new D inputs)
  - C=0: Master latch "off", Slave latch "on"
    - Changing D inputs has no effect on Master (or Slave) latch
    - D inputs from last time C=1 stored safely in Master and transferred into Slave and reflected on output Q

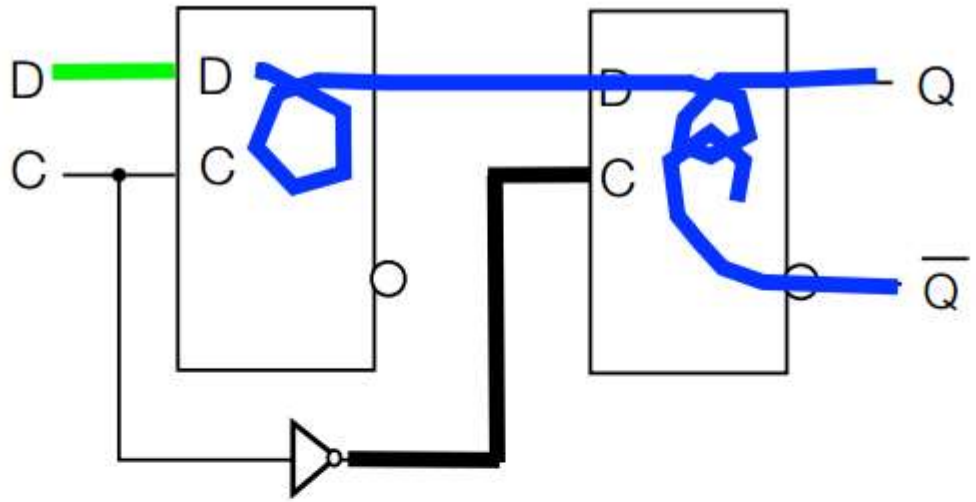
Negative-Edge Triggered D Flip-Flop

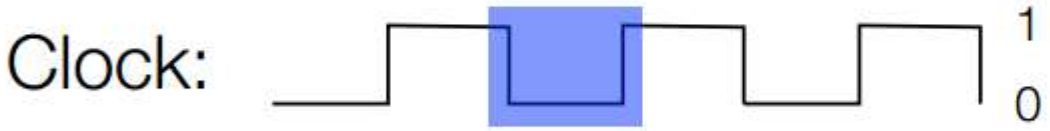




- C=1: Master latch "on", Slave latch "off"
  - New S & R inputs read into master
  - Previous Q values still emitted (not affected by new S&R inputs)

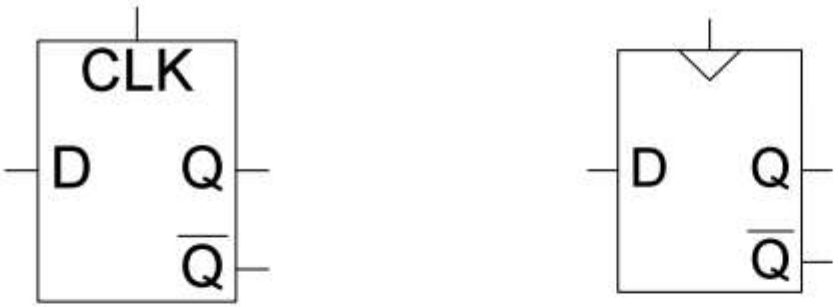
Negative-Edge Triggered D Flip-Flop

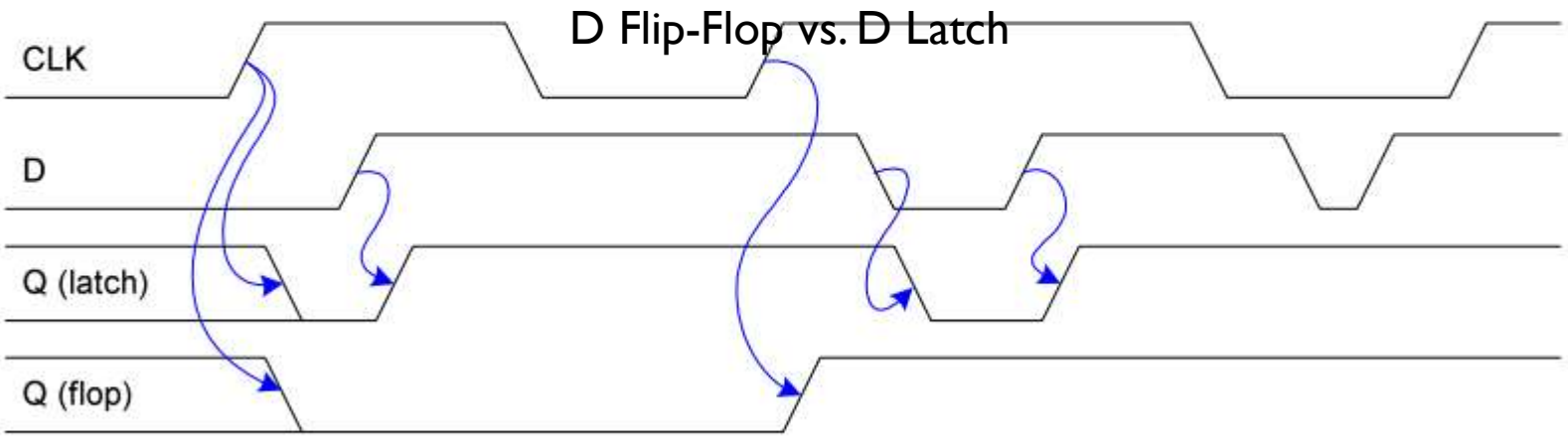




- C=0: Master latch "off", Slave latch "on"
  - Changing D inputs has no effect on Master (or Slave) latch
  - D inputs from last time C=1 stored safely in Master and transferred into Slave and reflected on output Q

D Flip-Flop vs. D Latch

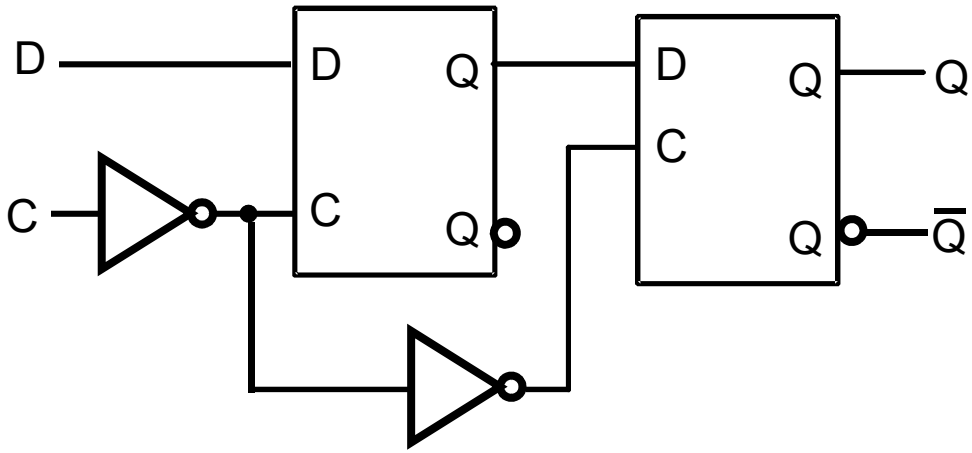




Latch outputs change at any time, flip-flops only during clock transitions

Positive-Edge Triggered D Flip-Flop

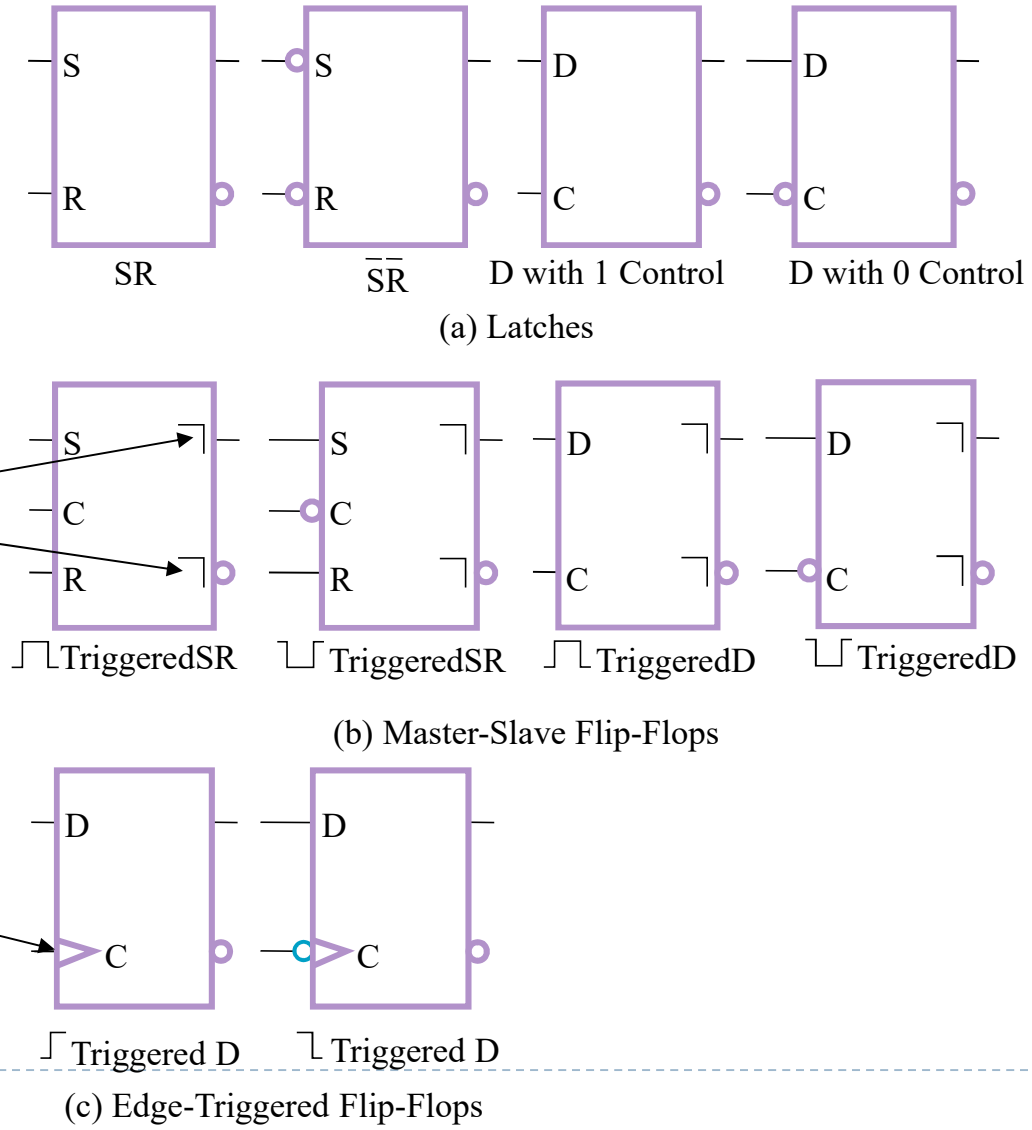
- Formed by adding inverter to clock input



- Q changes to the value on D applied at the positive clock edge within timing constraints to be specified
- Our choice as the **standard flip-flop** for most sequential circuits

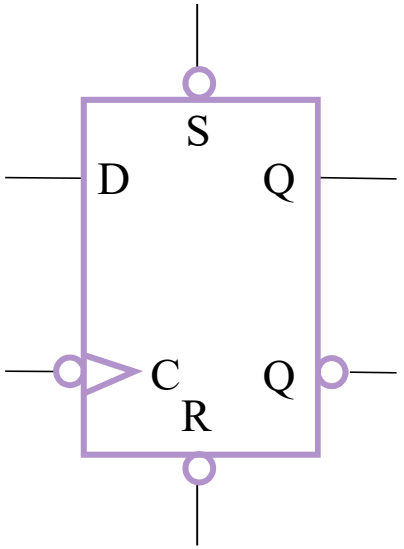
Standard Symbols for Storage Elements

- Master-Slave: Postponed output indicators
- Edge-Triggered: Dynamic indicator



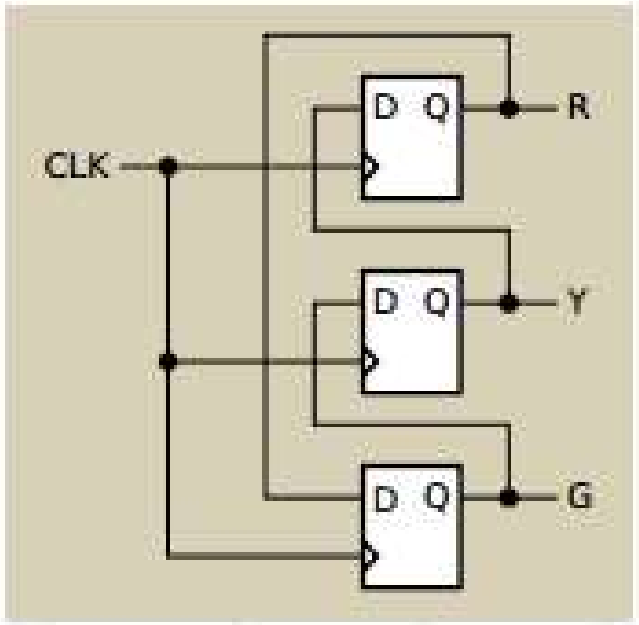
Direct Inputs

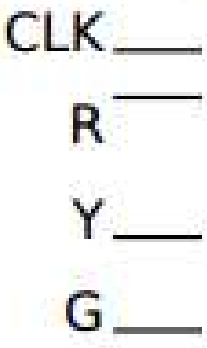
- At power up or at reset, all or part of a sequential circuit usually is initialized to a known state before it begins operation
- This initialization is often done outside of the clocked behavior of the circuit, i.e., asynchronously.



- Direct R and/or S inputs that control the state of the latches within the flip-flops are used for this initialization.
- For the example flip-flop shown
  - 0 applied to $\overline{R}$ resets the flip-flop to the 0 state
  - 0 applied to $\overline{S}$ sets the flip-flop to the 1 state

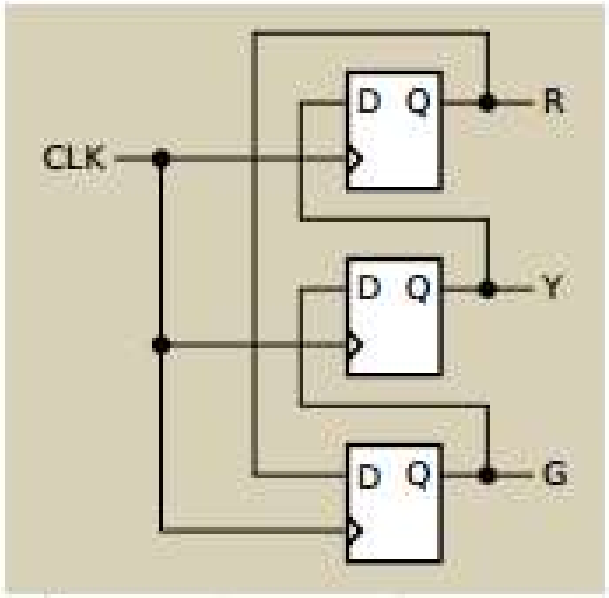
Positive-Edge Triggered D Flip-Flop: Traffic Light Controller

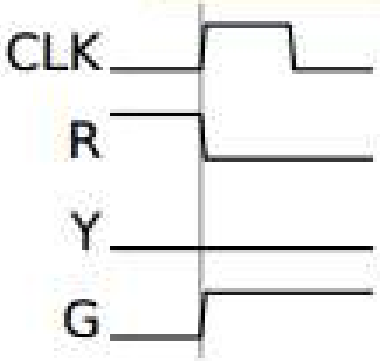




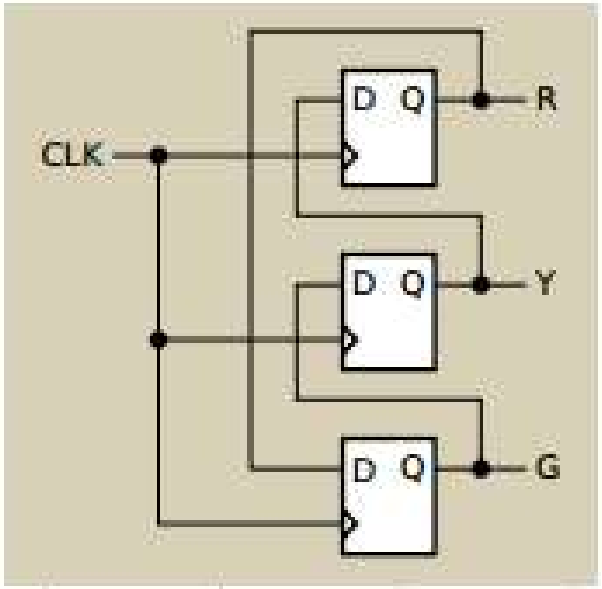
Initial state: Red light

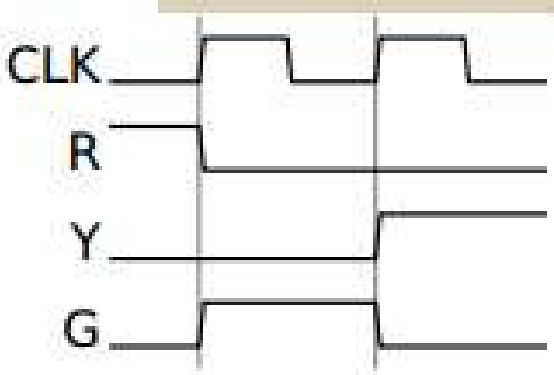
Positive-Edge Triggered D Flip-Flop: Traffic Light Controller



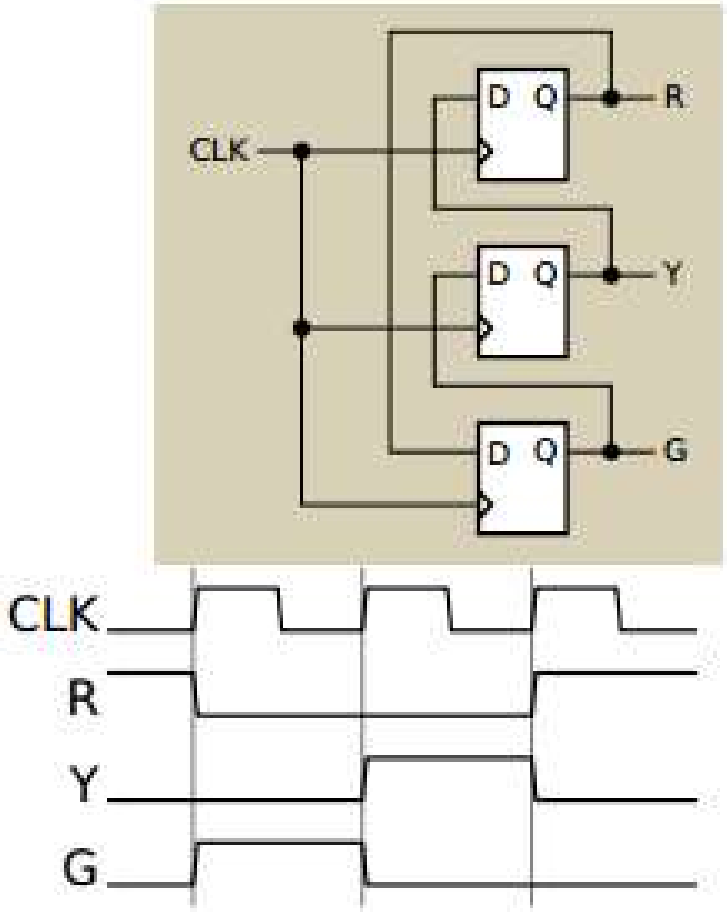


Positive-Edge Triggered D Flip-Flop: Traffic Light Controller

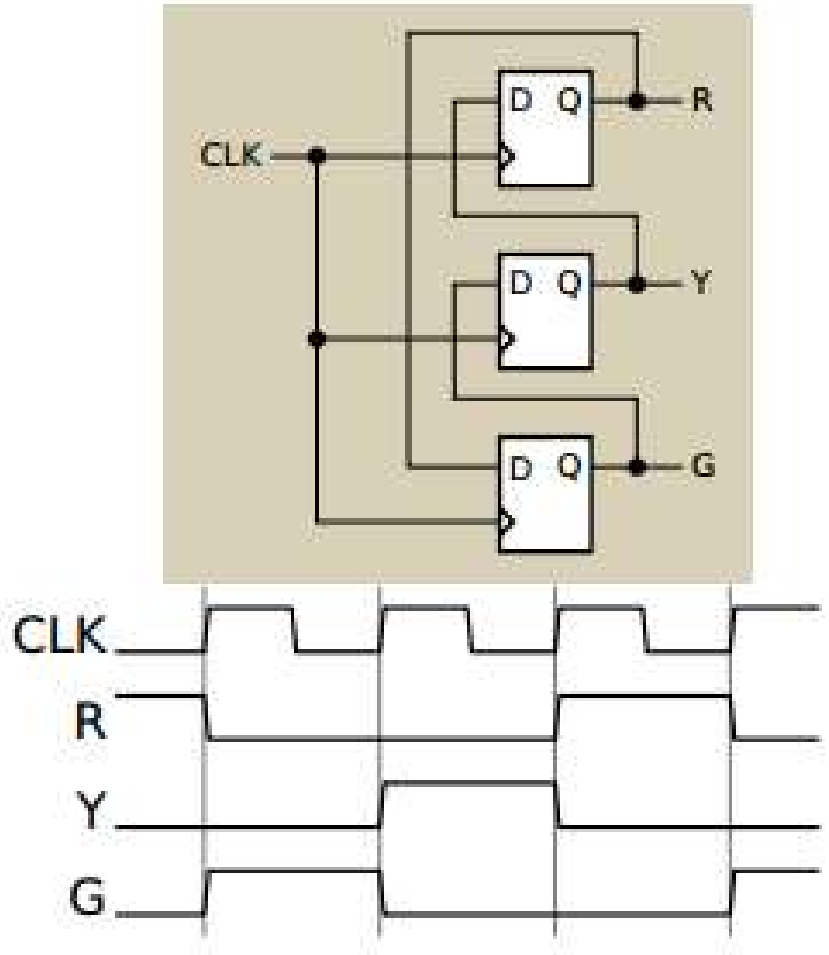




Positive-Edge Triggered D Flip-Flop: Traffic Light Controller



Positive-Edge Triggered D Flip-Flop: Traffic Light Controller



Any Questions?