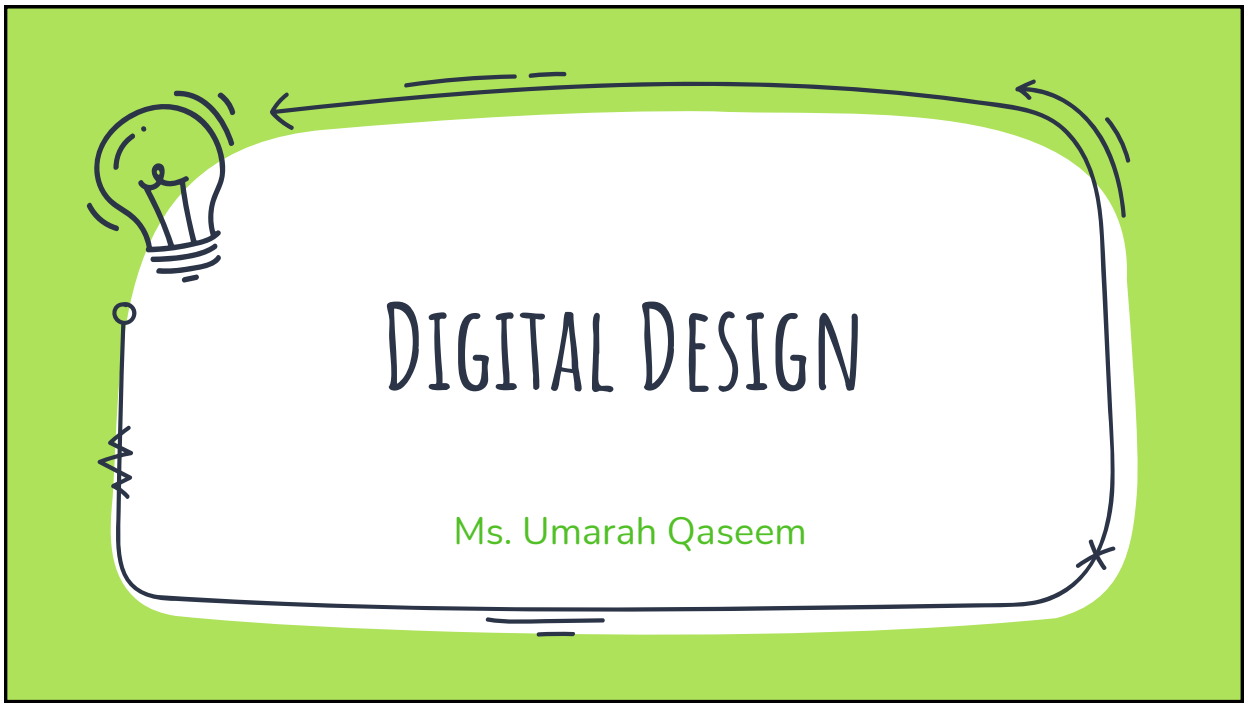
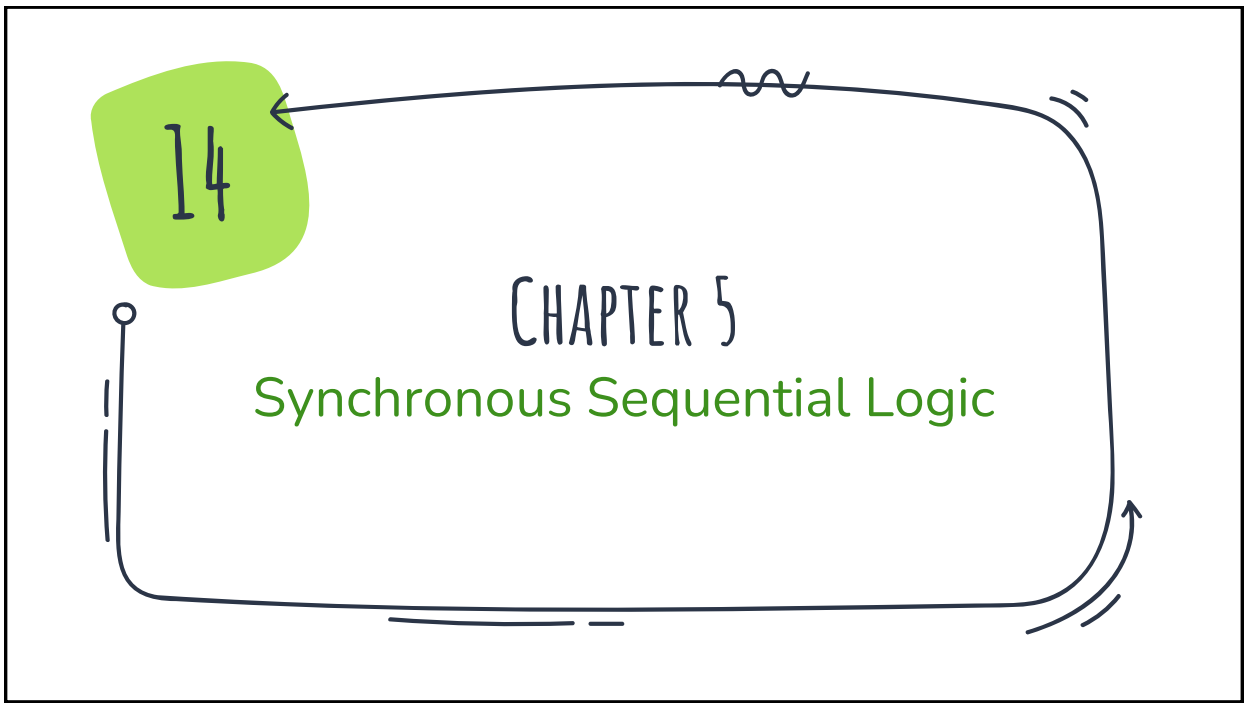
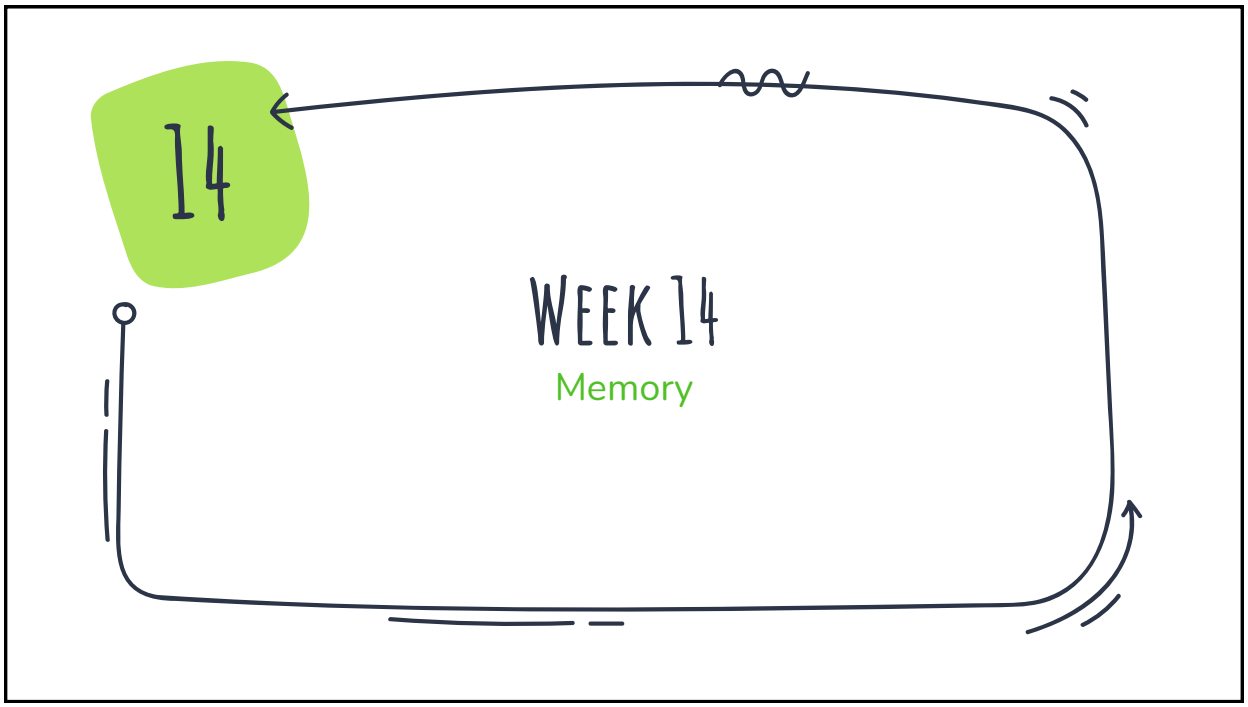

Digital Design
Ms. Umarah Qaseem

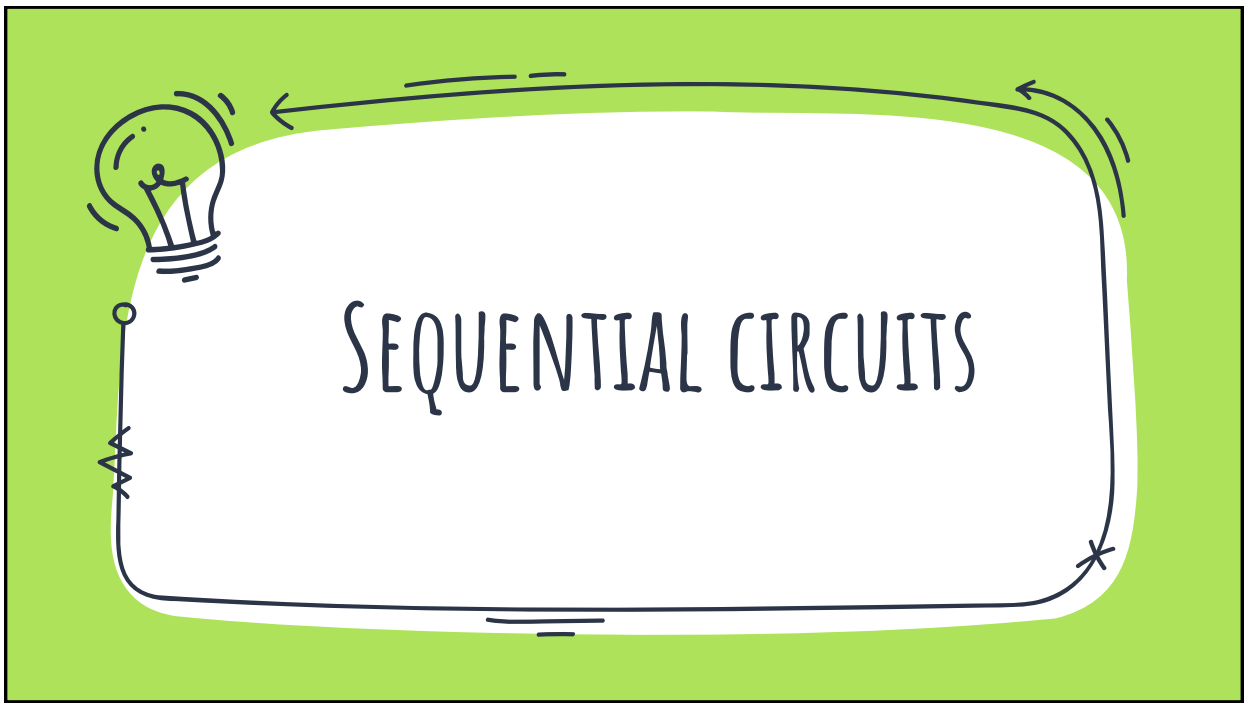
1


14
Chapter 5
Synchronous Sequential Logic

2



3



8

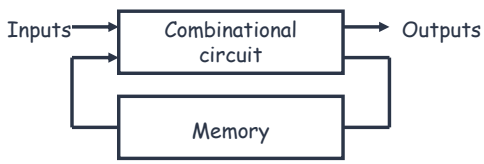
Combinational Circuit

- So far we've just worked with combinational circuits, where applying the same inputs always produces the same outputs.
- This corresponds to a mathematical function, where every input has a single, unique output.
- No way of remembering or storing information after inputs have been removed.

9

Sequential Circuit

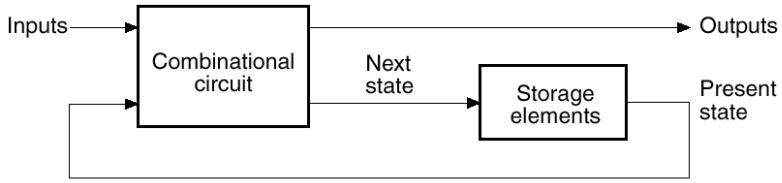
- The outputs of a sequential circuit depend on not only the inputs, but also the state, or the current contents of some memory.
- This makes things more difficult to understand, since the same inputs can yield *different* outputs, depending on what's stored in memory.
- The memory contents can also change as the circuit runs.



10

Sequential Circuit

- Many real-life devices are sequential in nature:
  - Combination locks open if you enter numbers in the right order.
  - Elevators move up or down and open or close depending on the buttons that are pressed on different floors and in the elevator itself.



11

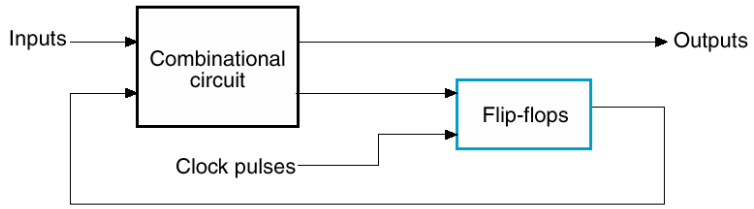
Types of Sequential Circuits

- Synchronous
  - State changes synchronized by one or more clocks
- Asynchronous
  - Timing of changes are independent of any clocks

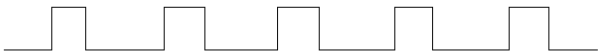
12

Synchronous Sequential Circuits

- Changes enabled by clock



(a) Block diagram



(b) Timing diagram of clock pulses

13

Storage Elements

- Latch
- Flip-Flop – a latch that transitions on a clock
- Registers

14

Memory

- A memory should have at least three properties.
  1. It should be able to hold a value.
  2. You should be able to *read* the value that was saved.
  3. You should be able to *change* the value that's saved.
- We'll start with the simplest case, a one-bit memory.
  1. It should be able to hold a single bit, 0 or 1.
  2. You should be able to read the bit that was saved.
  3. You should be able to change the value. Since there's only a single bit, there are only two choices:
     - Set the bit to 1
     - Reset, or clear, the bit to 0.

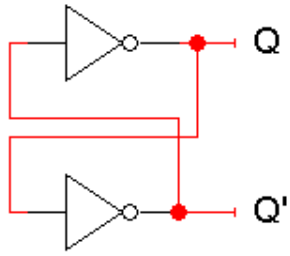
15

Basic Idea of Storage

- How can a circuit "remember" anything, when it's just a bunch of gates that produce outputs according to the inputs?
- The basic idea is to make a loop, so the circuit outputs are also inputs.
- First Attempt – Cross Coupled Inverter

16

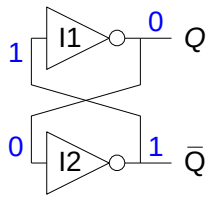
Cross Coupled Inverter



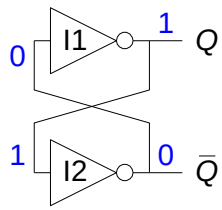
17

Cross Coupled Inverter

- Consider 2 possible cases:
  - Q = 0: then Q' = 1 and Q = 0 (consistent)



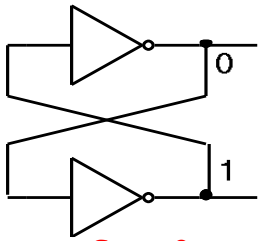
  - Q = 1: then Q' = 0 and Q = 1 (consistent)



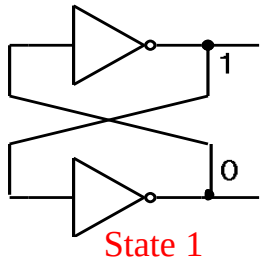
18

Cross Coupled Inverter

- Does this satisfy the properties of memory?
  - These circuits "**remember**" Q, because its value never changes. (Similarly, Q' never changes either.)
  - We can also "**read**" Q, by attaching a probe or another circuit.
  - But we can't ***change*** Q! There are no external inputs here, so we can't control whether Q=1 or Q=0.



State 0

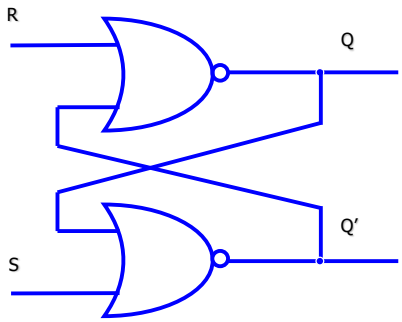


State 1

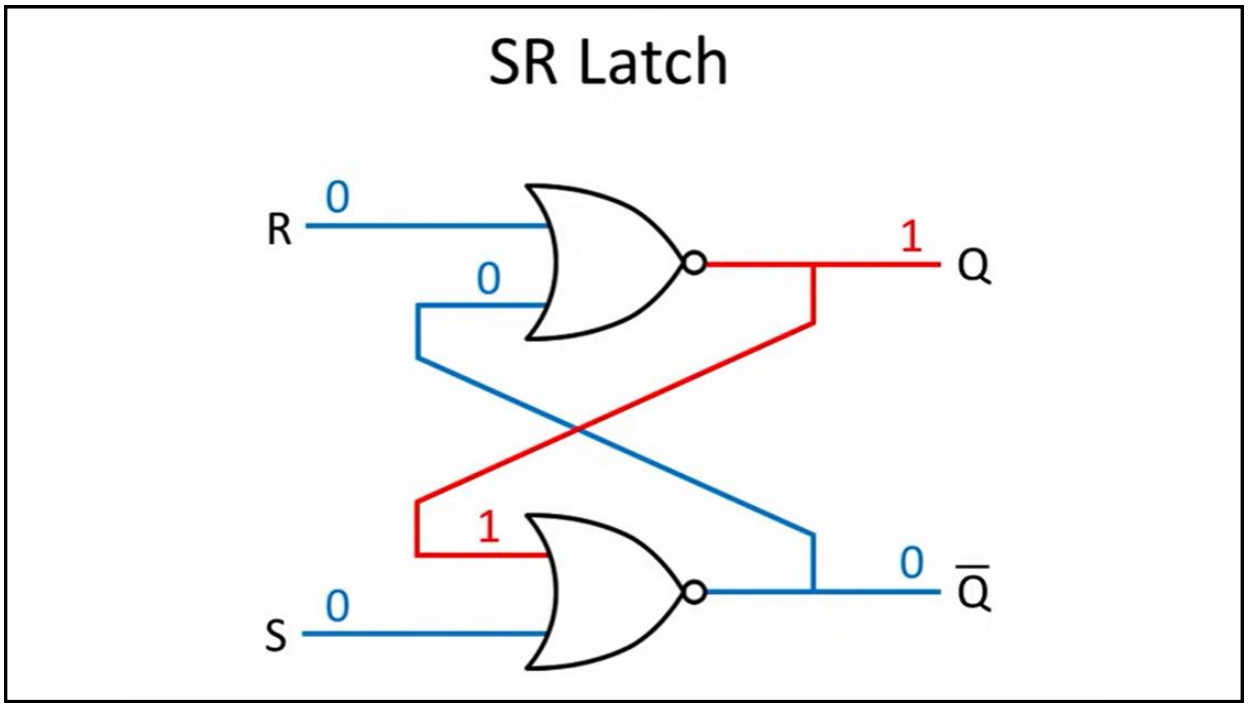
19

SR (set-reset) Latches

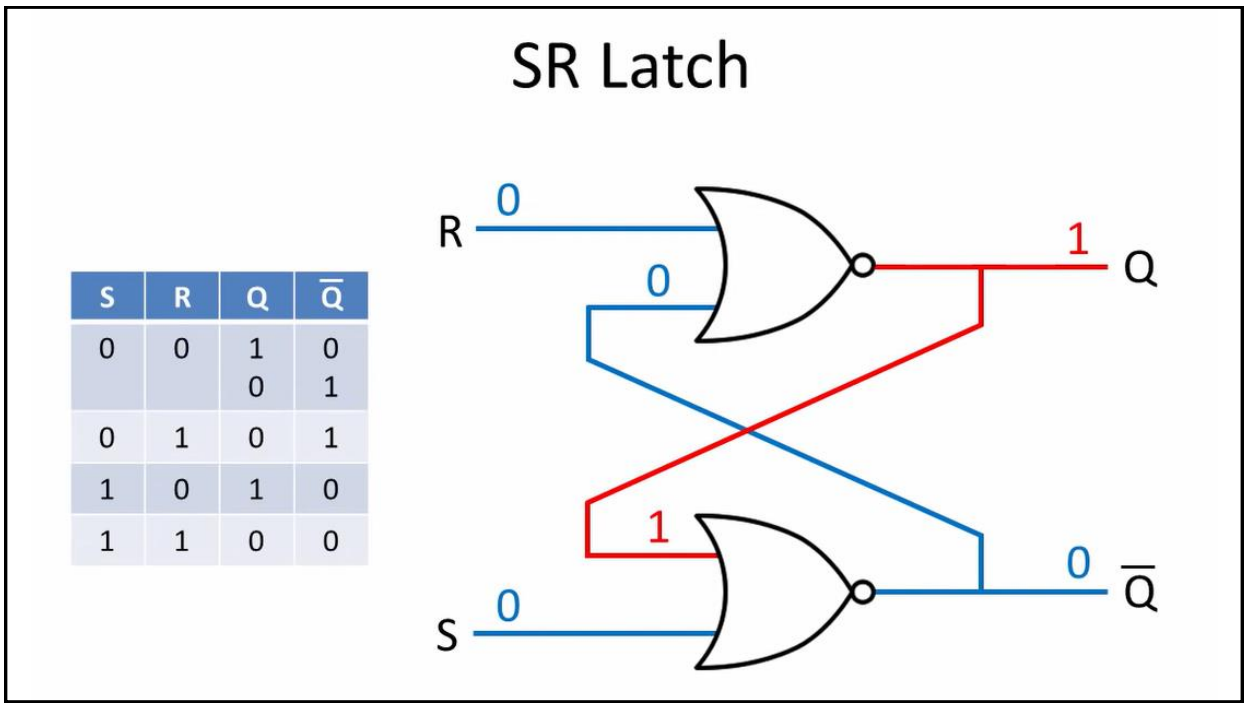
- Replace the Inverters with NOR gates
- Add two inputs – R & S



20


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

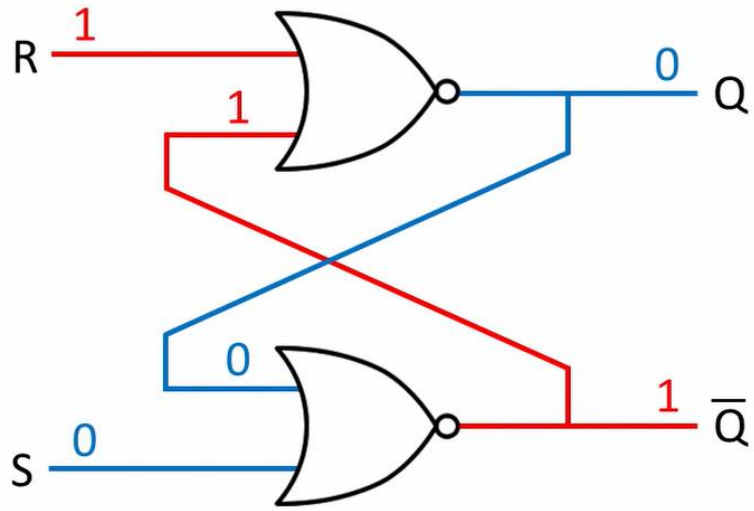
21


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

22

SR Latch

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

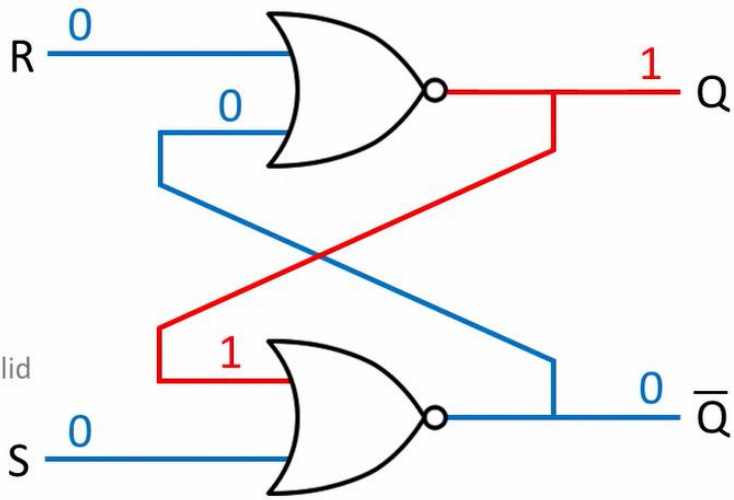


23

SR Latch

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

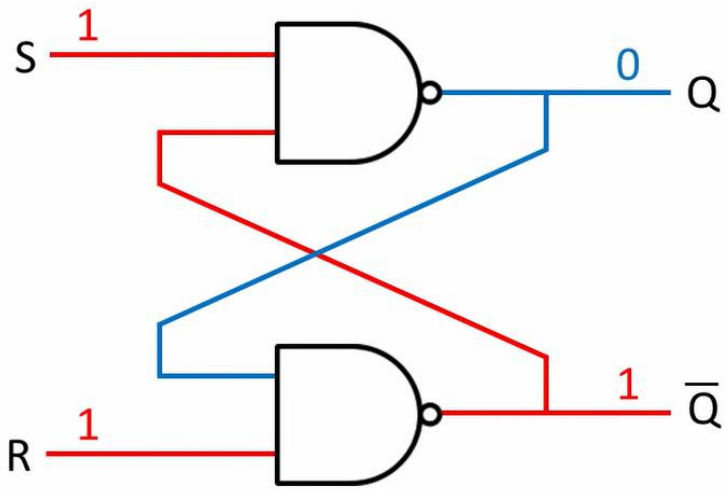
Invalid



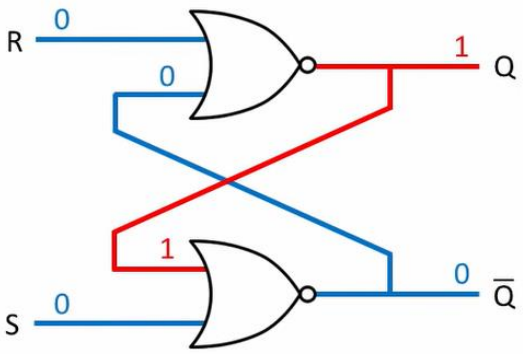
24

SR Latch

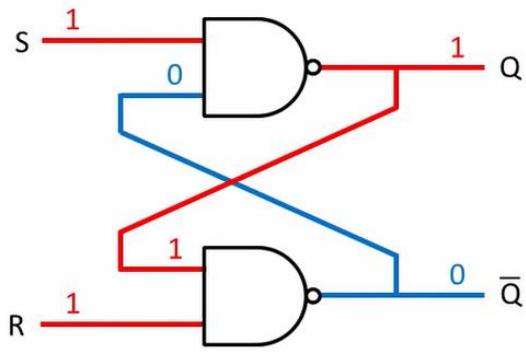
S	R	Q	$\overline{Q}$
0	0	1	1
0	1	1	0
1	0	0	1
1	1	0 1	1 0



25

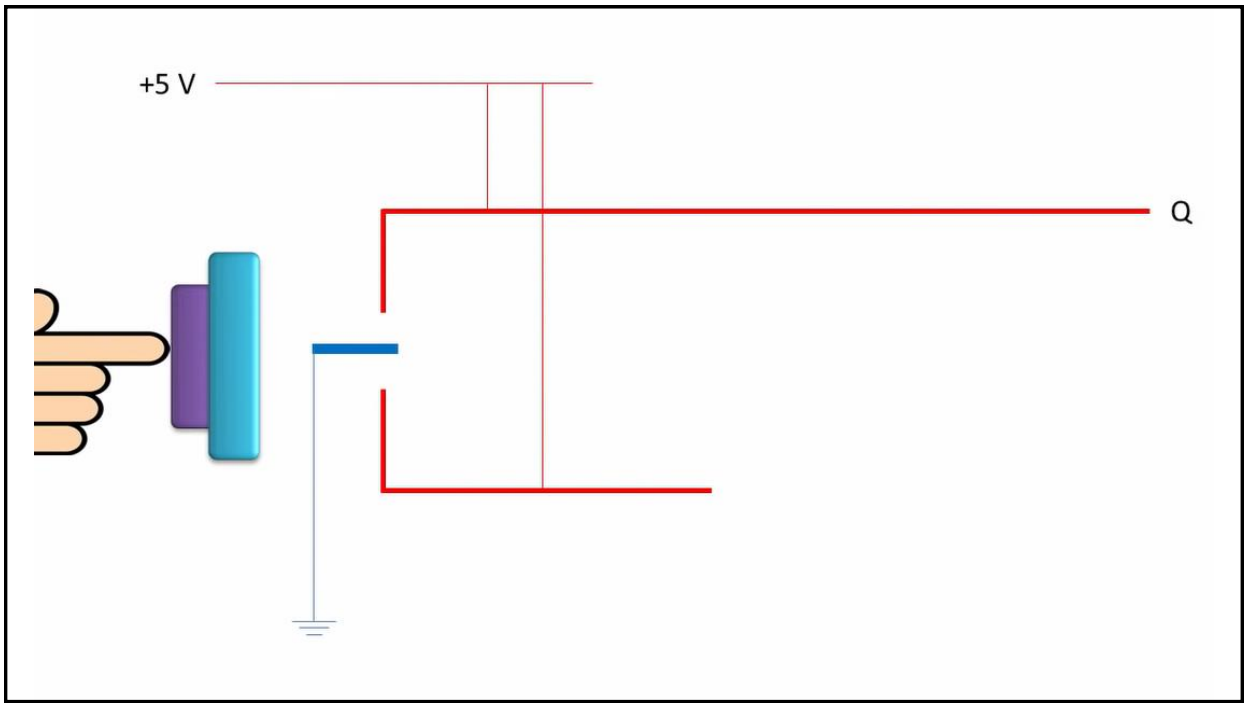


S	R	Q	$\overline{Q}$	
0	0	1 0	0 1	
0	1	0	1	
1	0	1	0	
1	1	0	0	Invalid

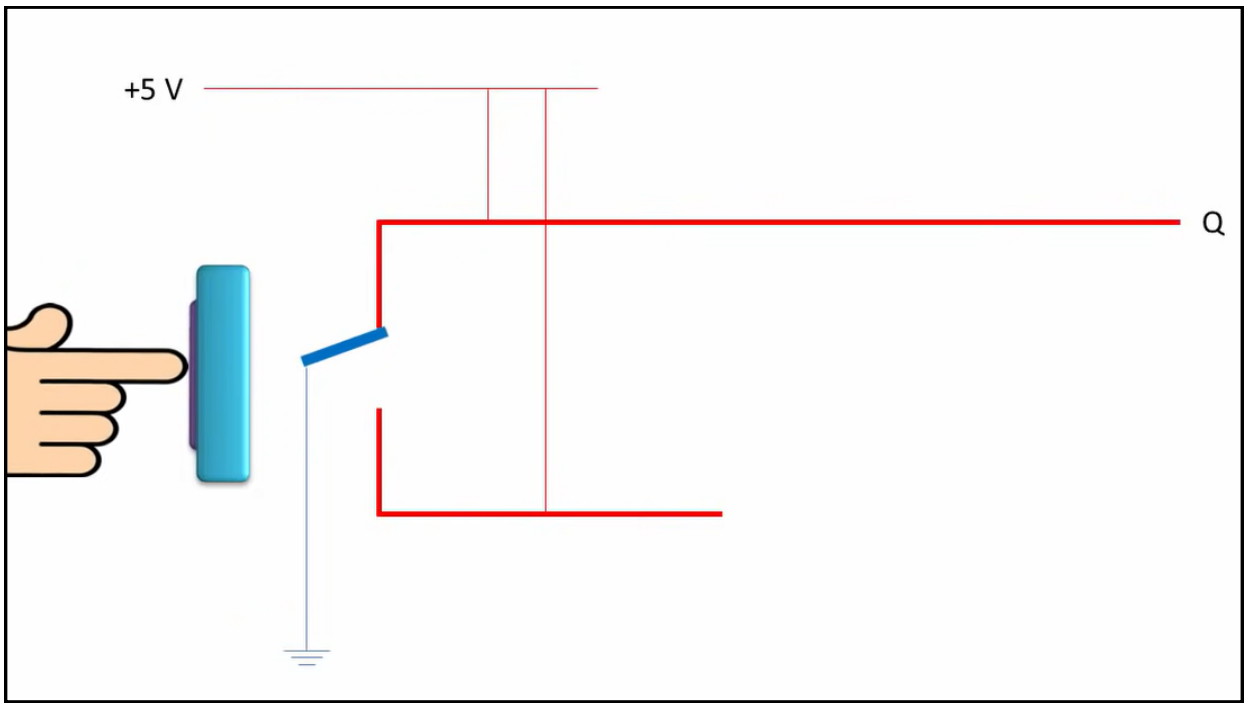


S	R	Q	$\overline{Q}$	
0	0	1	1	Invalid
0	1	1	0	
1	0	0	1	
1	1	0 1	1 0	

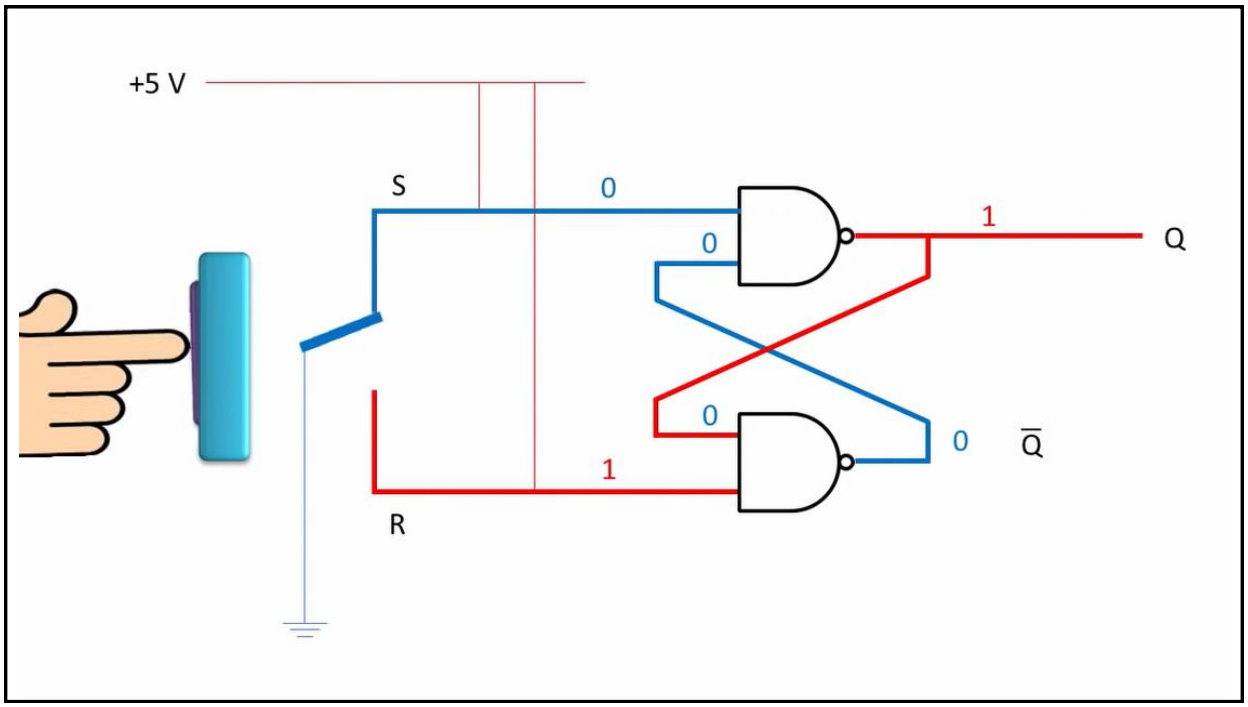
26



27



28

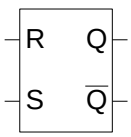


29

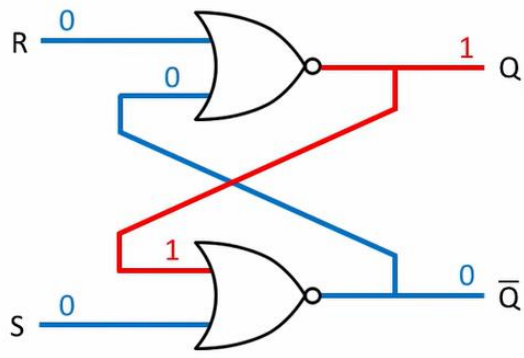
SR Latch Summary

- SR stands for Set/Reset Latch
  - Stores one bit of state (Q)
- Control what value is being stored with S, R inputs
  - Set: Make the output 1 (S = 1, R = 0, Q = 1)
  - Reset: Make the output 0 (S = 0, R = 1, Q = 0)
- Behavior undefined/invalid when:
  - S = R = 1

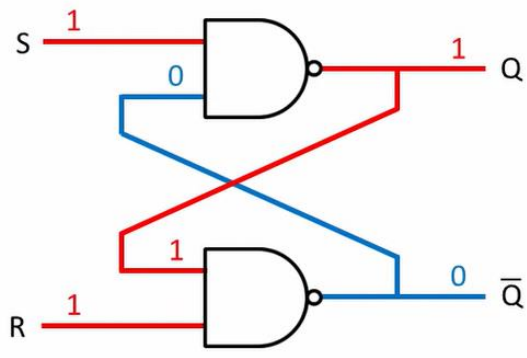
SR Latch Symbol



46

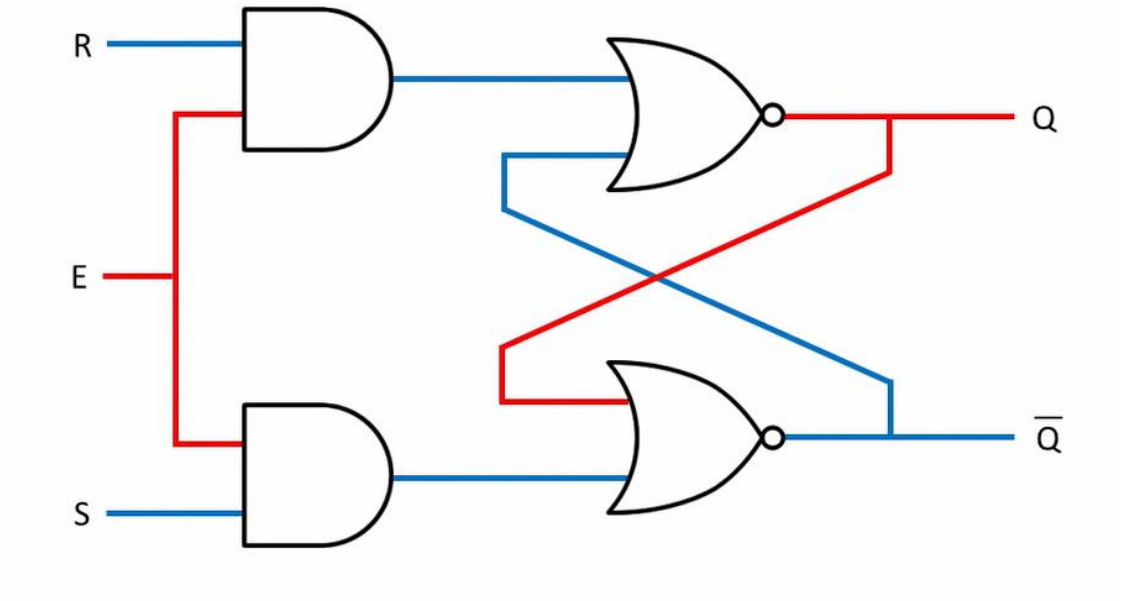


S	R	Q	$\overline{Q}$	
0	0	1 0	0 1	
0	1	0	1	
1	0	1	0	
1	1	0	0	Invalid



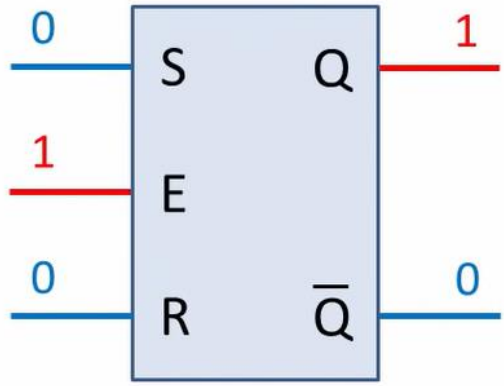
S	R	Q	$\overline{Q}$	
0	0	1	1	Invalid
0	1	1	0	
1	0	0	1	
1	1	0 1	1 0	

47

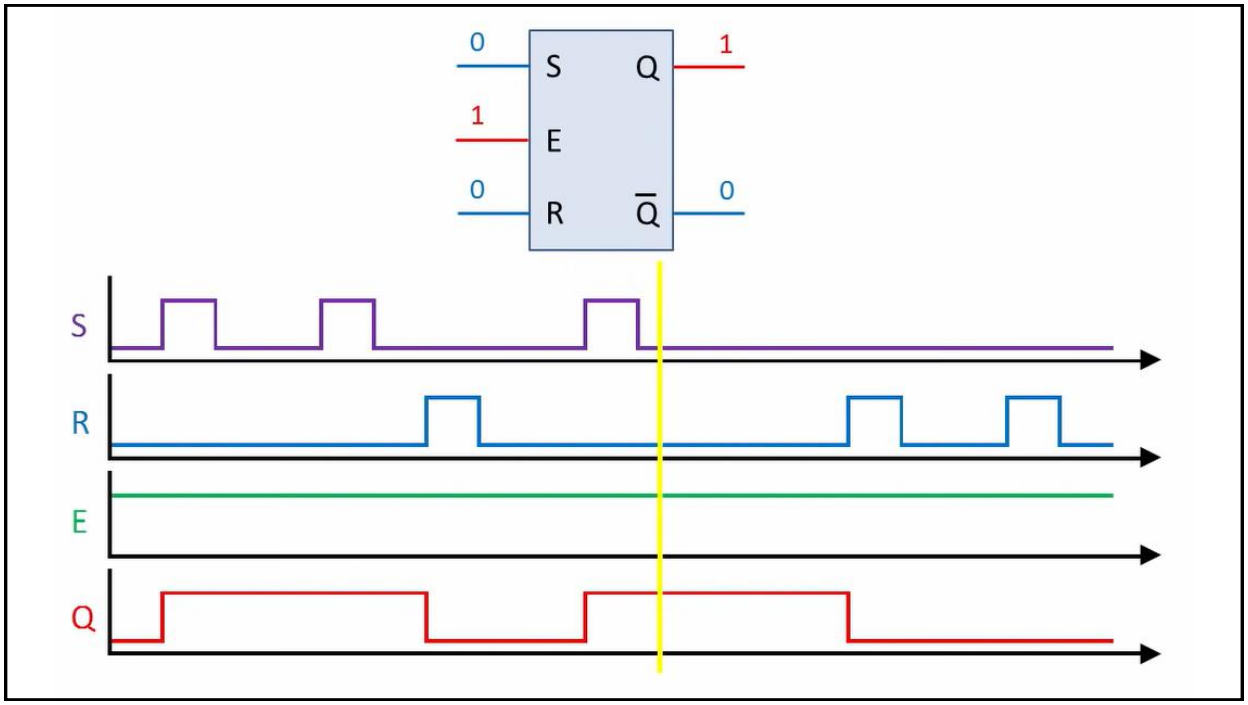


48

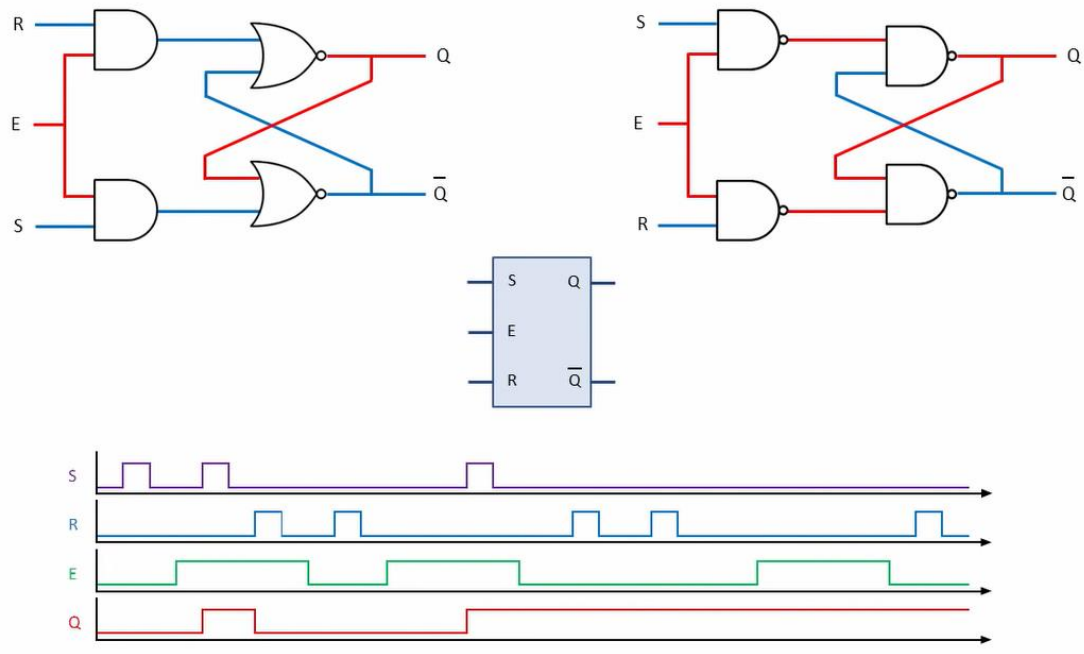
Gated SR Latch



49

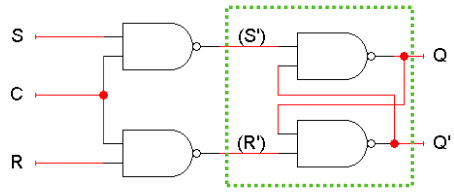


50



51

SR Latch with Control Input

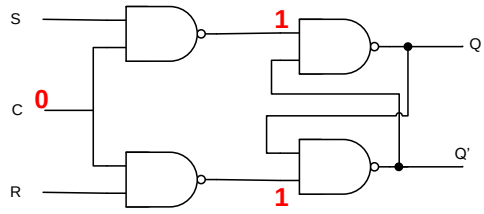


C	S	R	S'	R'	Q
0	x	x	1	1	No change
1	0	0	1	1	No change
1	0	1	1	0	0 (reset)
1	1	0	0	1	1 (set)
1	1	1	0	0	Avoid!

- Notice the hierarchical design!
  - The dotted blue box is the S'R' latch from the previous slide.
  - The additional NAND gates are simply used to generate the correct inputs for the S'R' latch.
- The control input acts just like an enable.

52

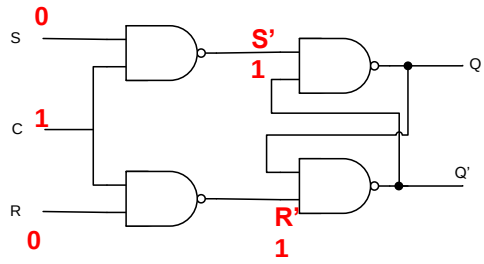
S'R' Latch with control input



c	S	R	Q
0	x	x	No change

53

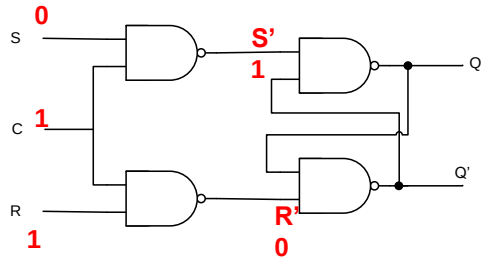
S'R' Latch with control input



c	S	R	Q
0	x	x	No change
1	0	0	No change

54

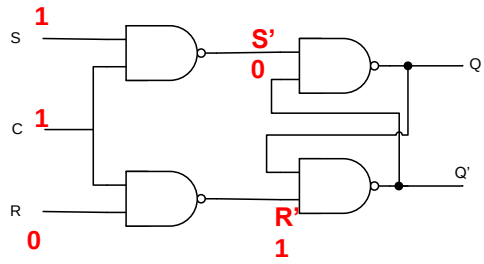
S'R' Latch with control input



c	S	R	Q
0	x	x	No change
1	0	0	No change
1	0	1	0

55

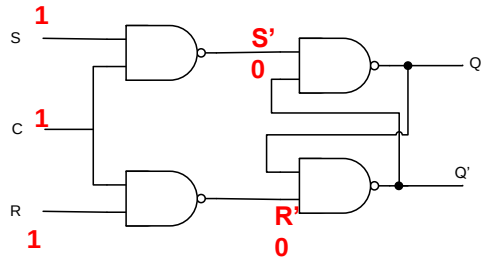
S'R' Latch with control input



c	S	R	Q
0	x	x	No change
1	0	0	No change
1	0	1	0
1	1	0	1

56

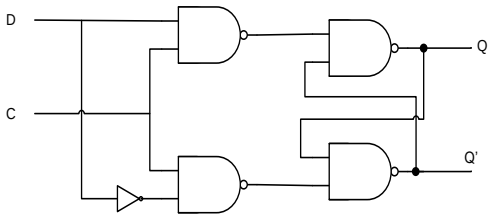
S'R' Latch with control input



c	S	R	Q
0	x	x	No change
1	0	0	No change
1	0	1	0
1	1	0	1
1	1	1	indeterminate

57

D-Latch



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

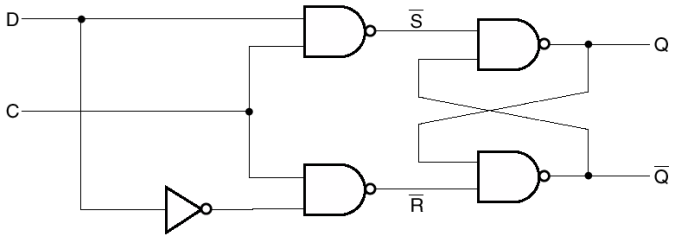
58

D Latch

- No illegal state

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

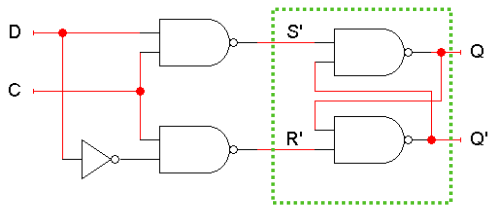


(a) Logic diagram

59

D Latch

- A D latch is based on an S'R' latch. The additional gates generate the S' and R' signals, based on inputs D ("data") and C ("control").
  - When C = 0, S' and R' are both 1, so the state Q does not change.
  - When C = 1, the latch output Q will equal the input D.
- No more messing with one input for set and another input for reset!

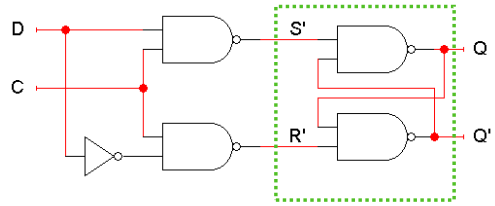


C	D	Q
0	x	No change
1	0	0
1	1	1

60

D Latch

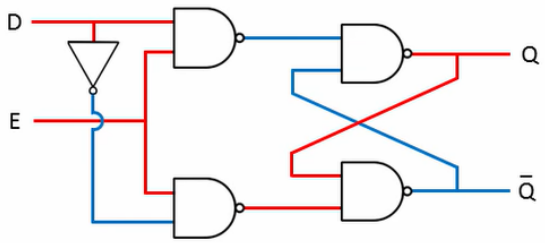
- D Latch is also called *transparent latch*
- Whatever is on D is passed onto Q

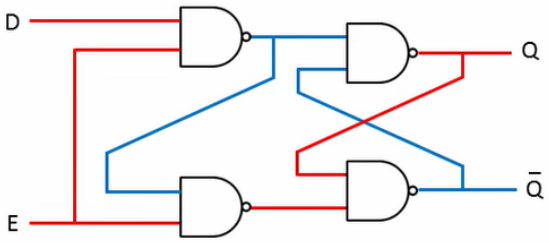


C	D	Q
0	x	No change
1	0	0
1	1	1

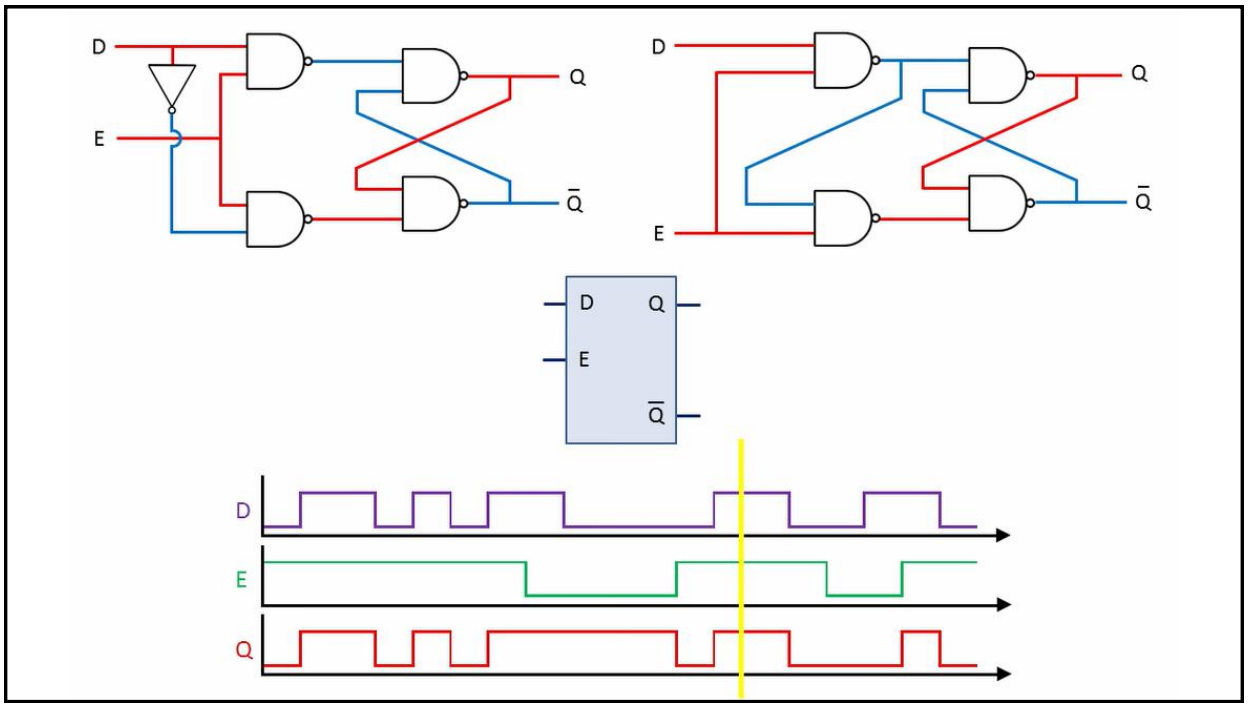
61

D Latch





63



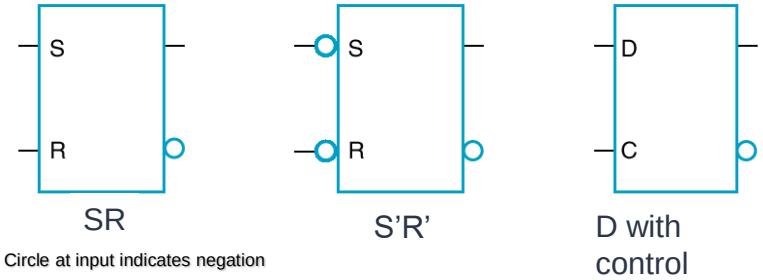
64

Summary

- A sequential circuit has memory. It may respond differently to the same inputs, depending on its current state.
- Memories can be created by making circuits with feedback.
  - Latches are the simplest memory units, storing individual bits.
  - It's difficult to control the timing of latches in a larger circuit.
- Next, we'll improve upon latches with flip-flops, which change state only at well-defined times. We will then use flip-flops to build all of our sequential circuits.

65

Standard Symbols - Latches



SR

Circle at input indicates negation

S'R'

D with control

66

Latches vs Flip Flops

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

67

Flip-Flops

- The state of the flip-flop or a latch is switched by the change in the control input – C.
- This momentary change is called trigger
- and the transition it causes is said to trigger the flip-flop.
- The D-Latch with pulses in its control input is essentially a flip-flop that is triggered every time the pulse goes to logic 1.

68

Flip-Flops

- Flip-flops are constructed in such a way to operate them properly when working in a proper clock.
- The key to the proper operation of flip-flop is to trigger it only during a *signal transition*.

69

Level Sensitive vs Edge Triggered

- Latches are level-sensitive

(a) Response to positive level

- Flip-flops are edge-sensitive

(b) Positive-edge response

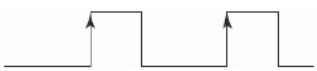
(c) Negative-edge response

70

Flip Flops

- D Latch – Data is stored when the clock is

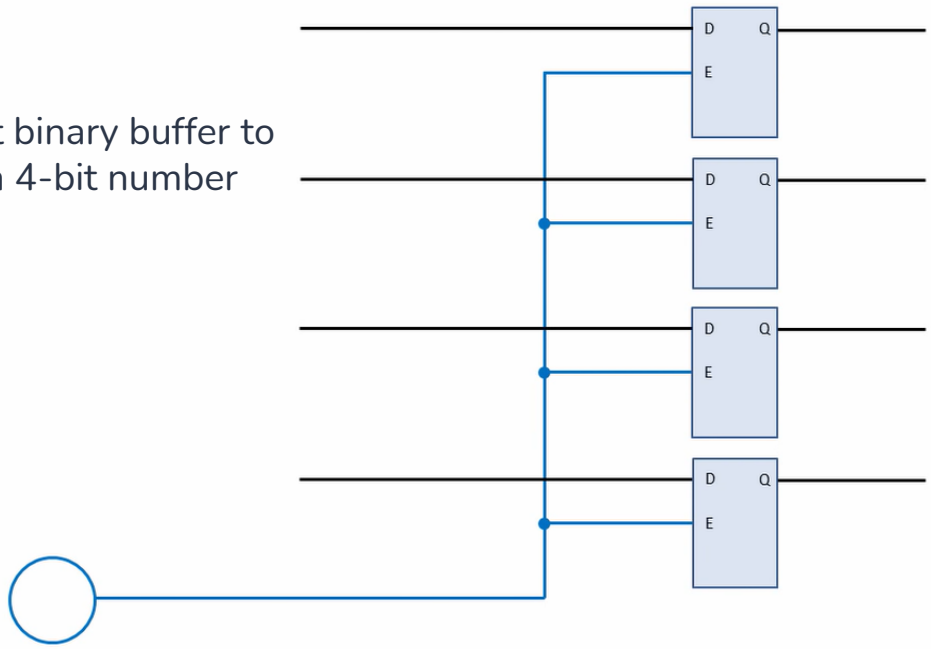
- How can we build a flip flop that stores on edge transitions?



or

71

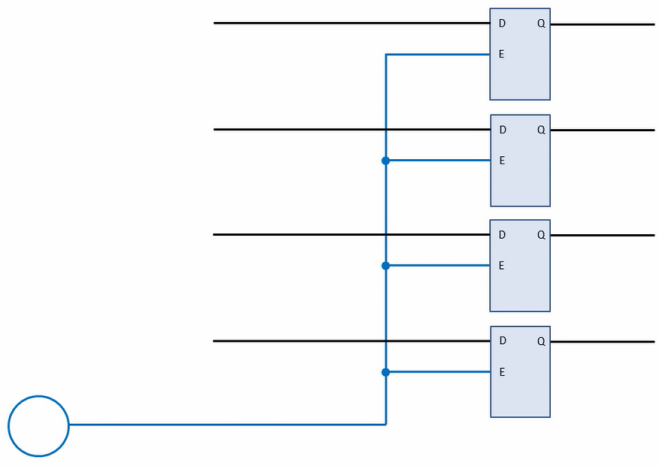
E.g:
A 4-bit binary buffer to
Store a 4-bit number



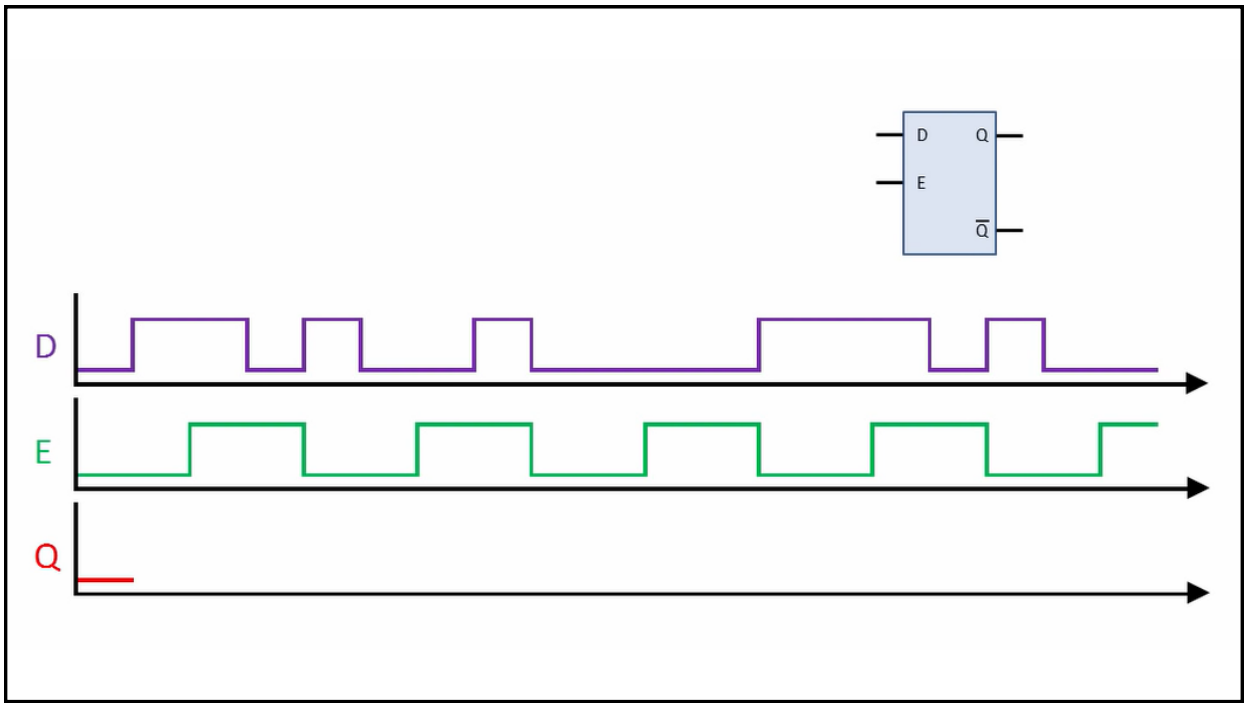
72

Clock is used to:
Synchronize the one-bit binary memories with each other and with other circuits in the computer.

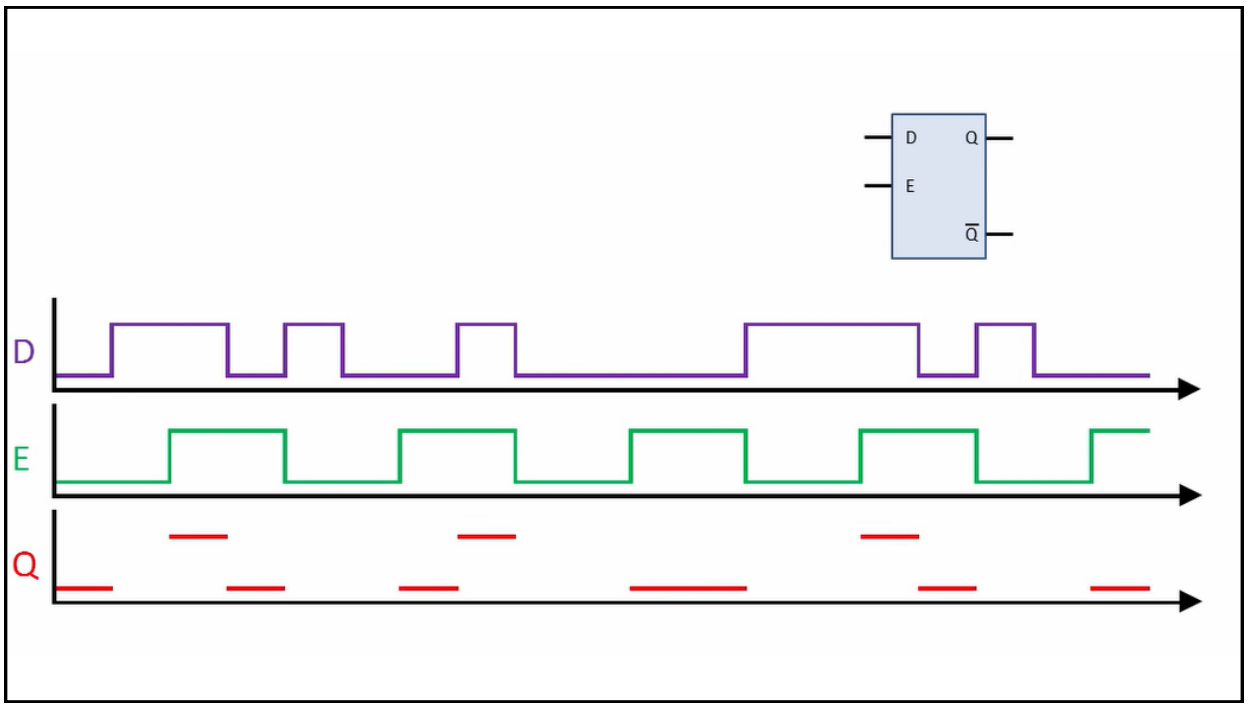
Clock regulates all kind of operations by generating electrical signal that alternates between high and low at regular intervals.



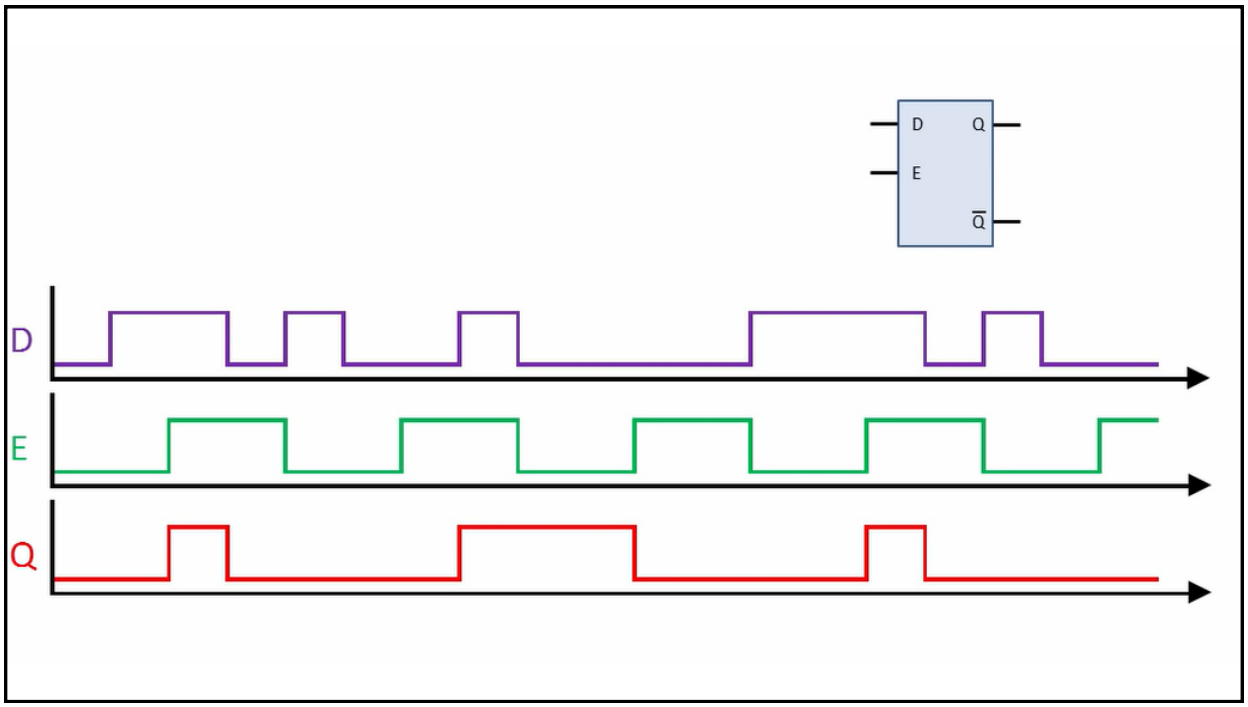
73



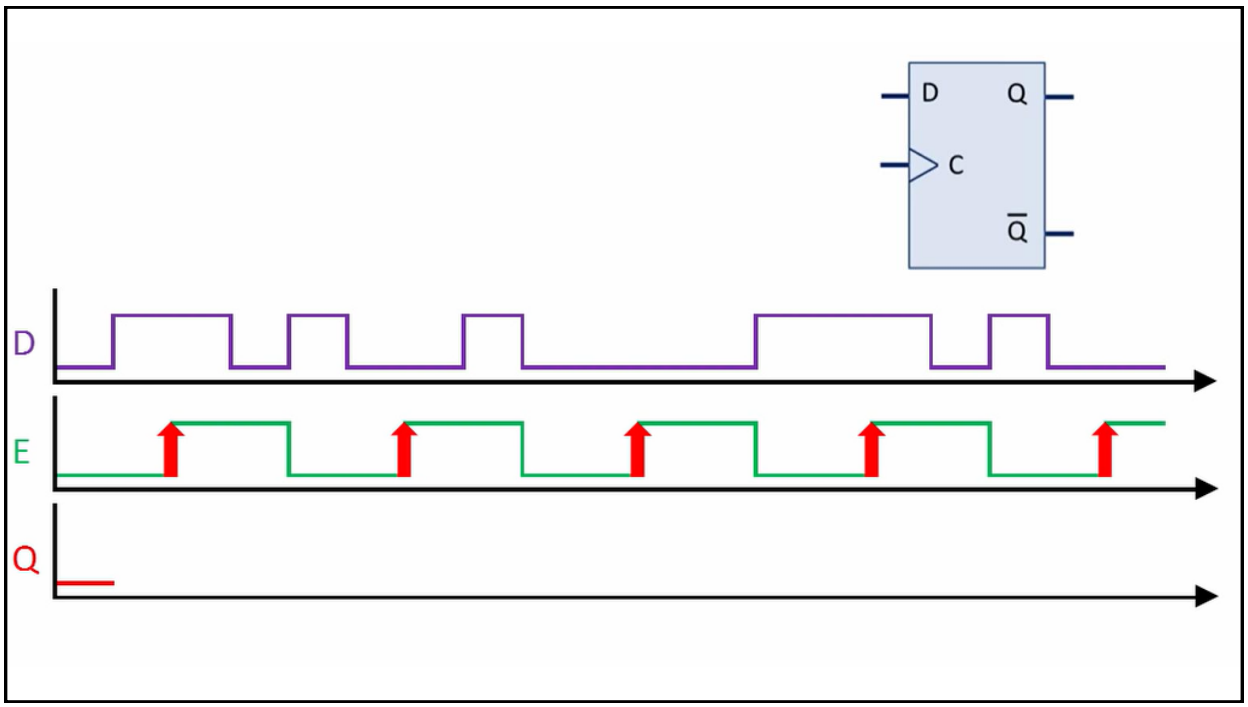
74



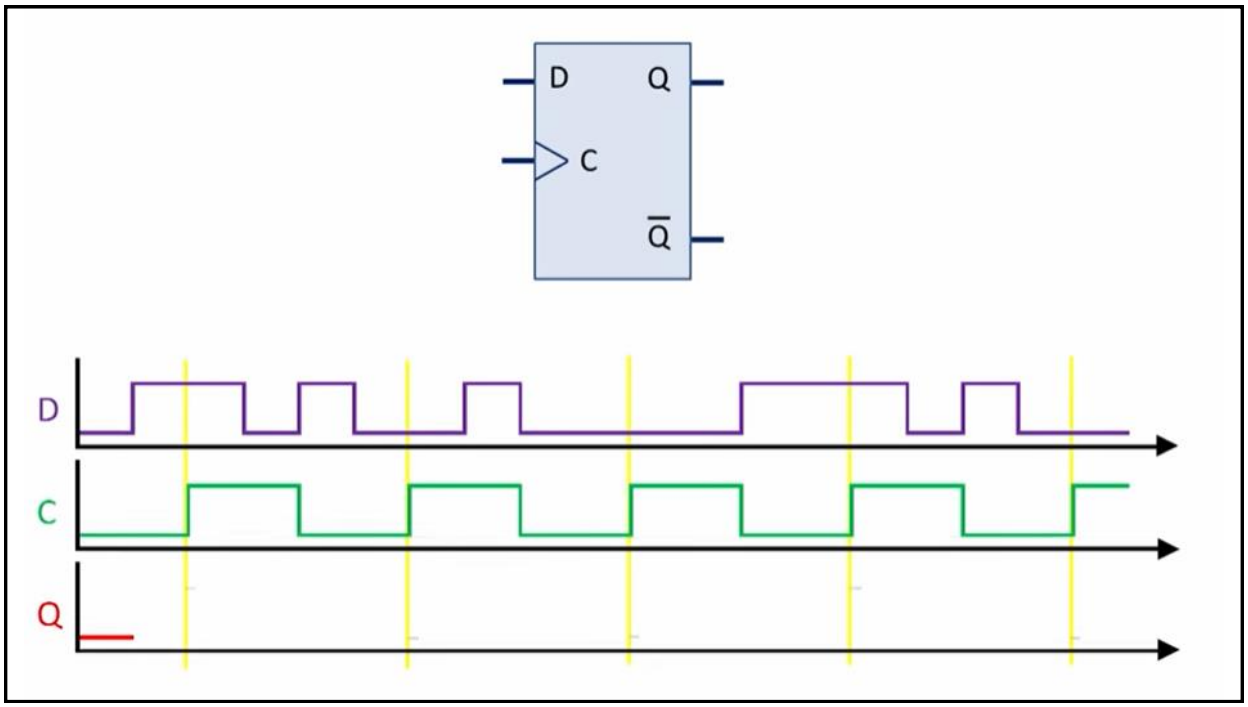
75



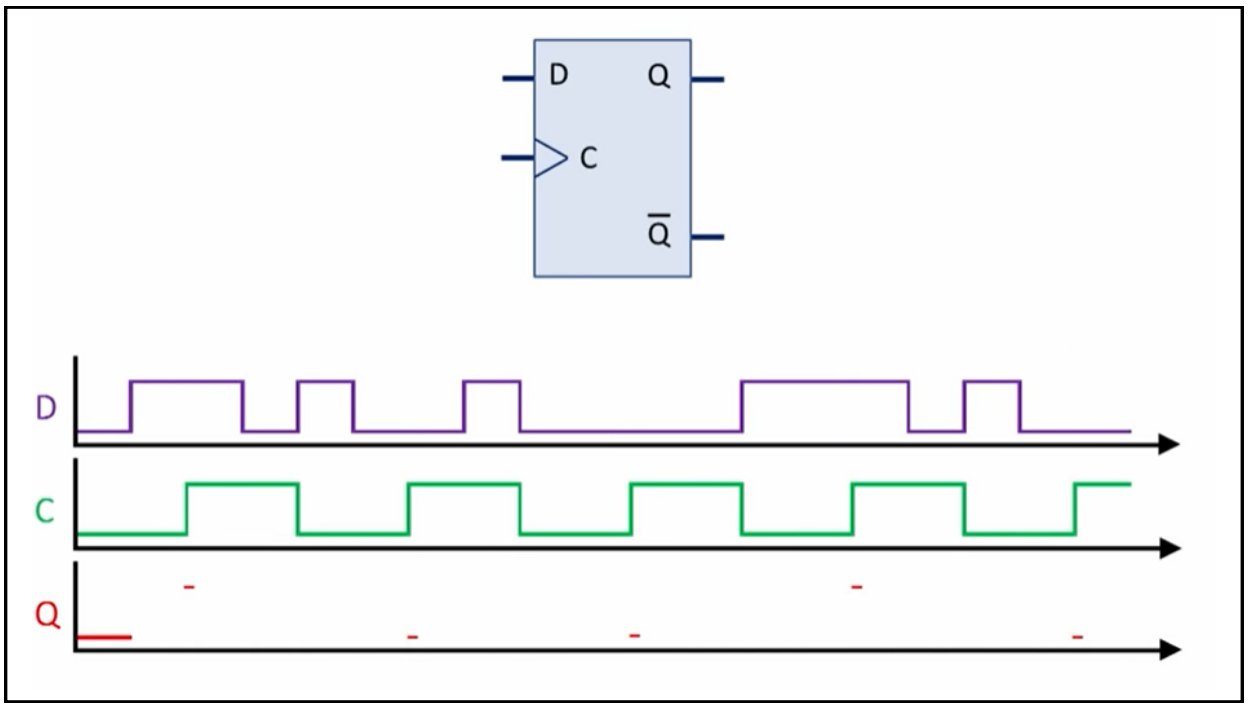
76



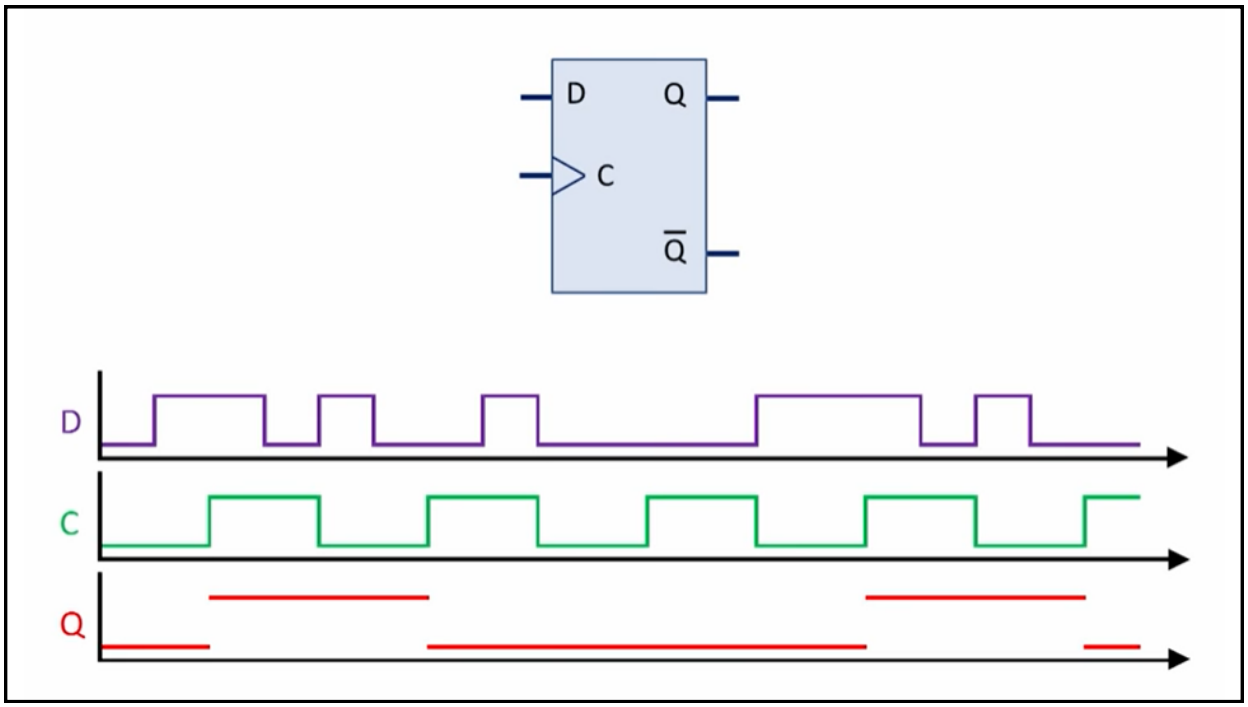
77



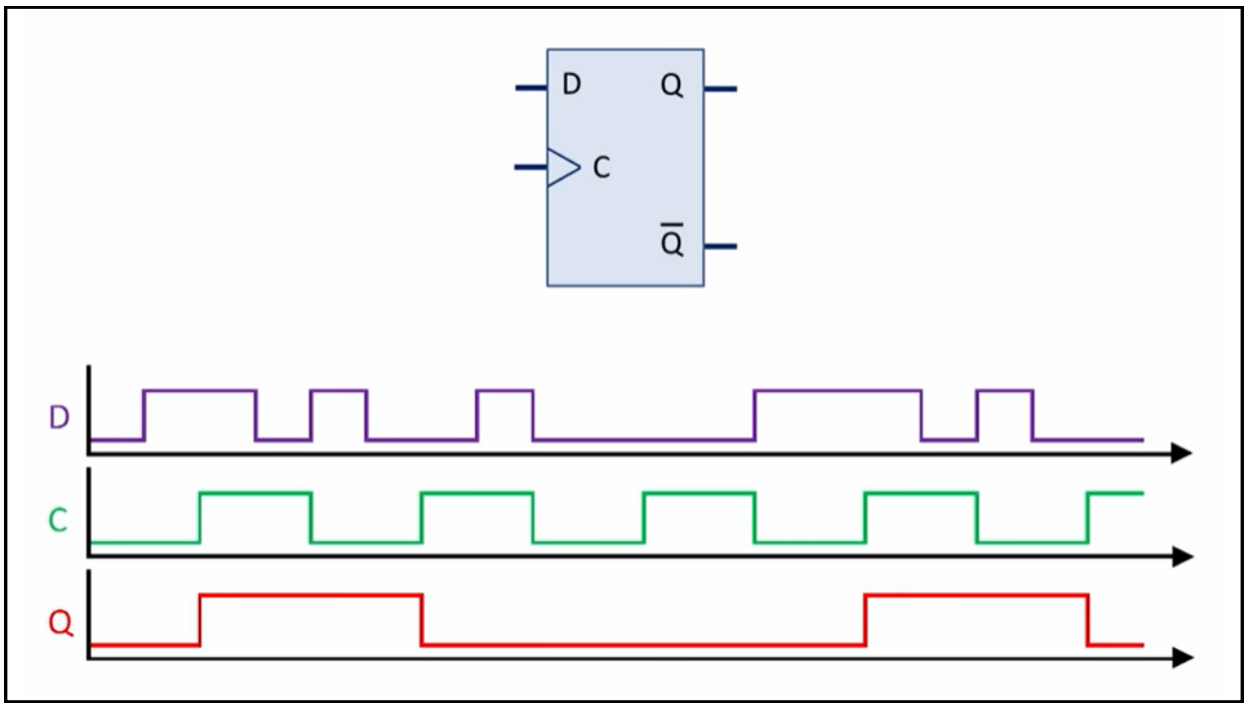
78



79

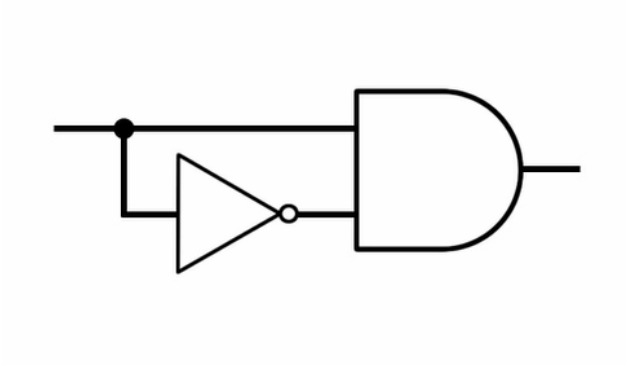


80



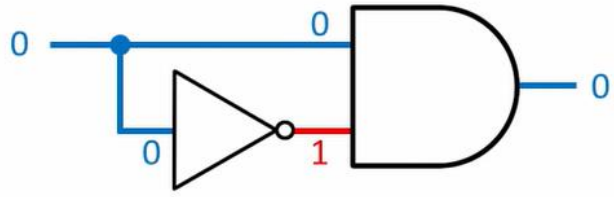
81

How to create a positive edge triggered device?



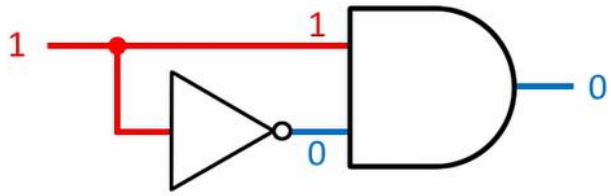
82

How to create a positive edge triggered device?



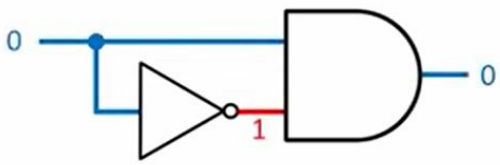
83

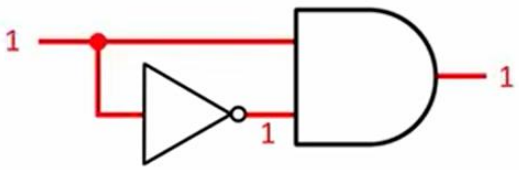
How to create a positive edge triggered device?



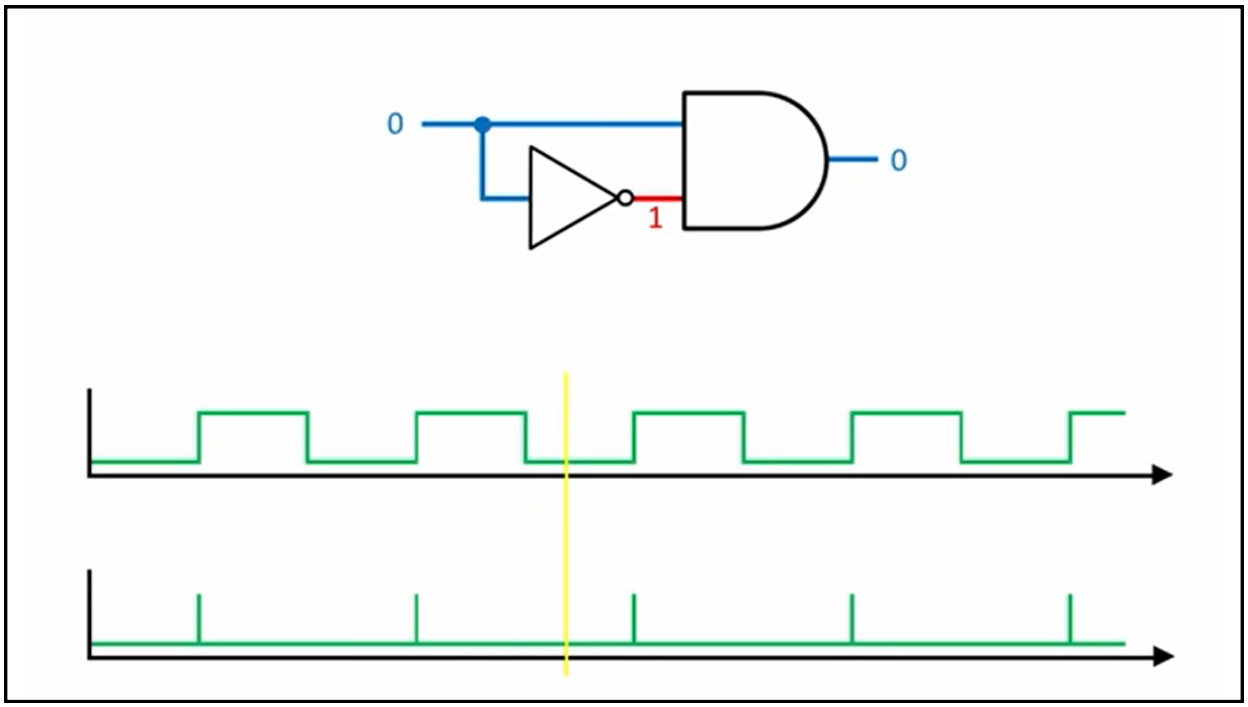
84

How to create a positive edge triggered device?



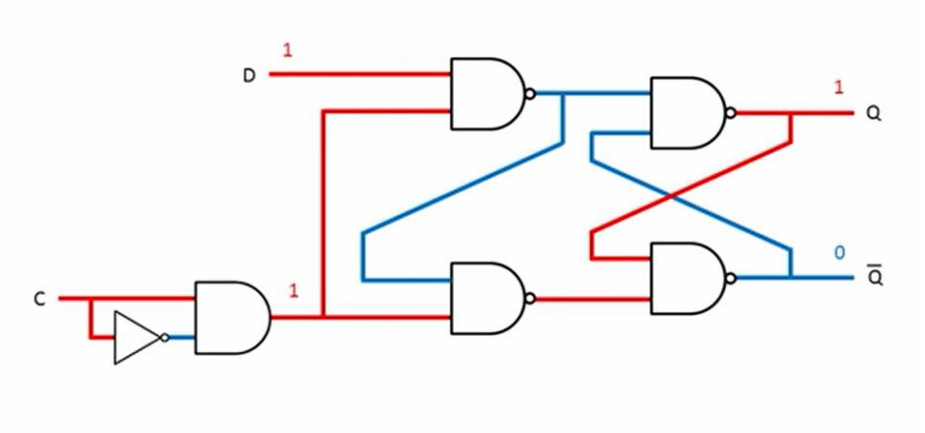


85

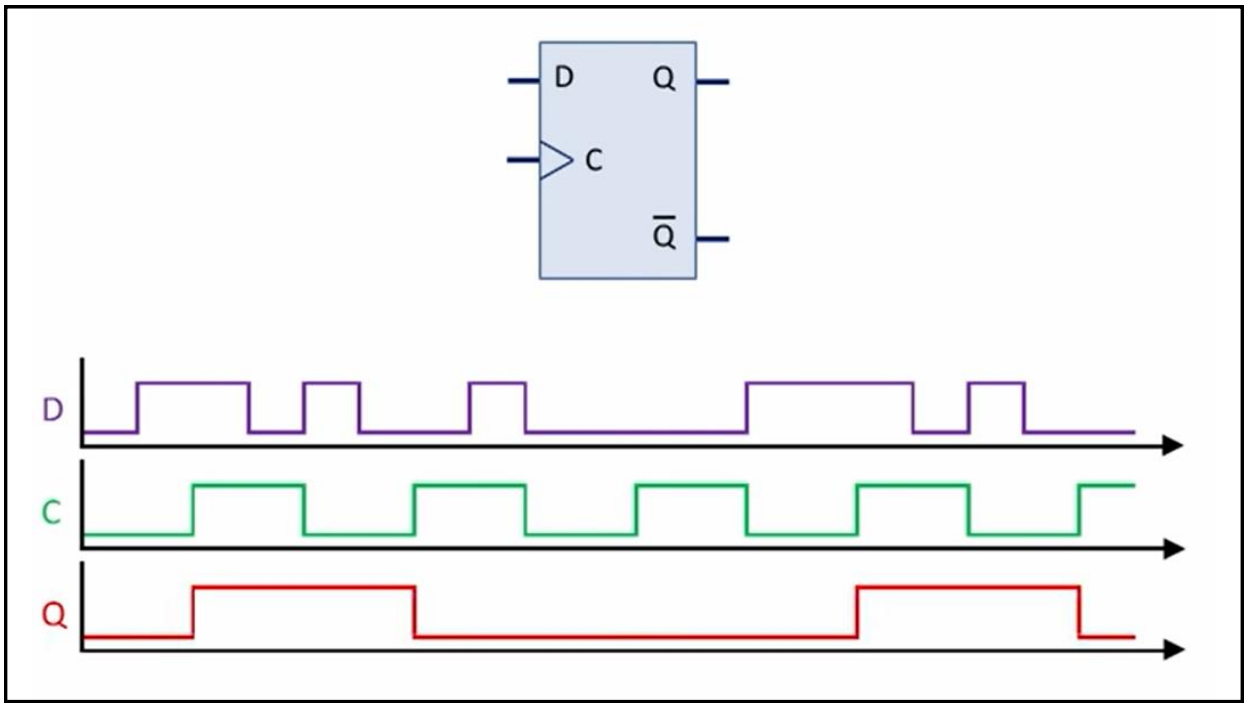


86

Now this is a positive edge triggered device

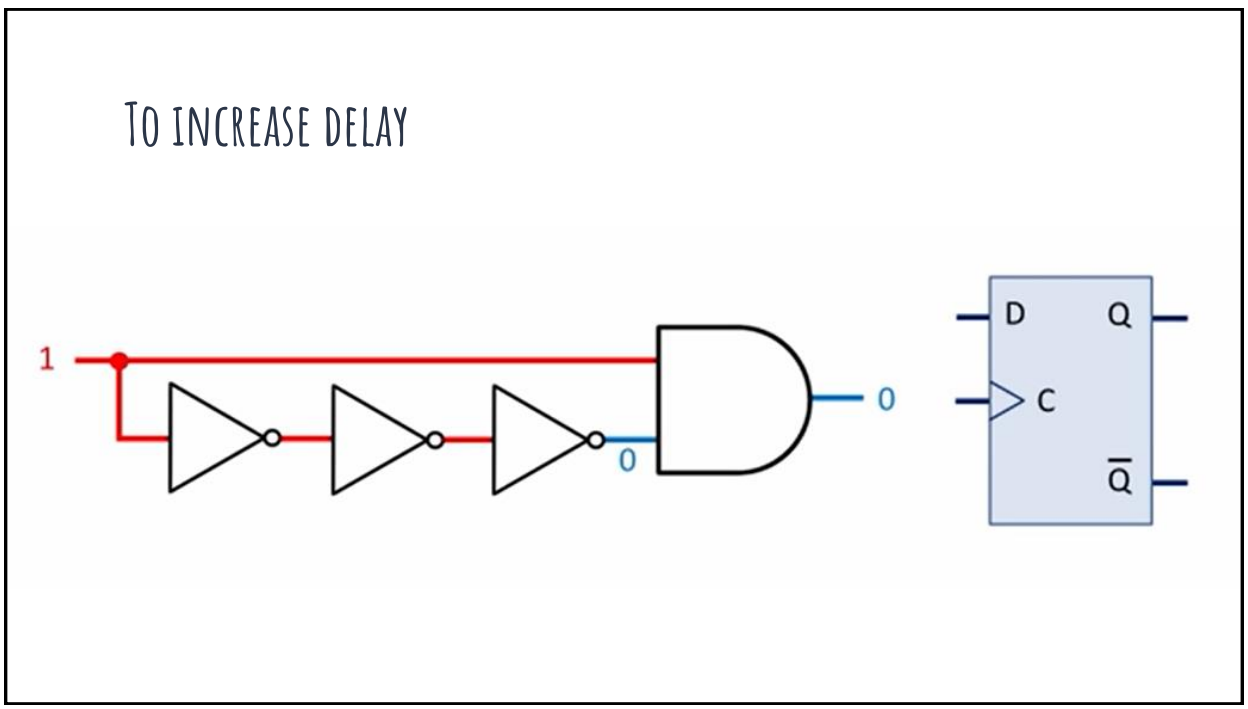


87



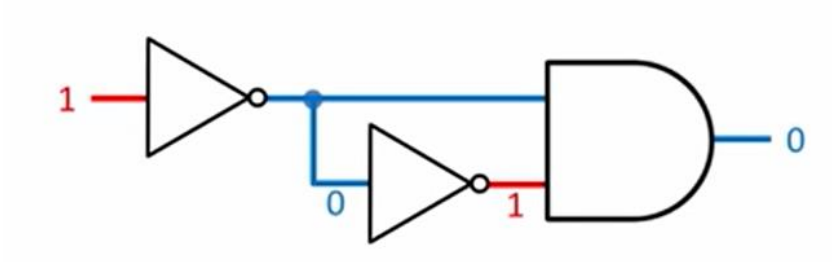
88

To increase delay



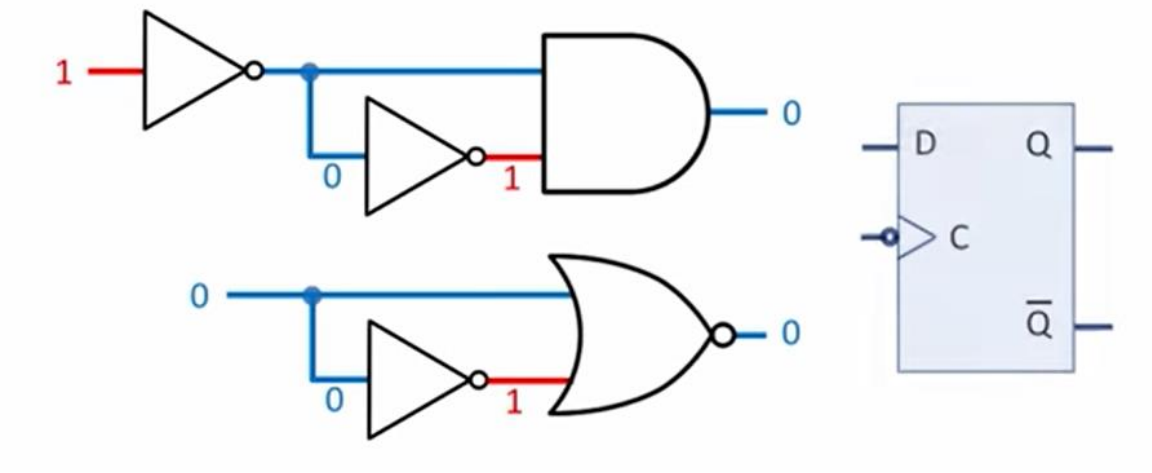
89

To create a Negative edge triggered device



90

To create a Negative edge triggered device



91

Flip Flop Symbols

(a) Positive-edge (a) Negative-edge

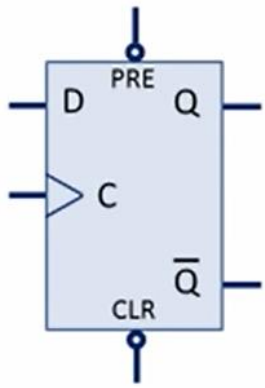
- **Triangle indicates clock**
- **Edge trigger:**
  - **No bubble at clock: positive edge triggered**
  - **Bubble at clock: negative edge triggered**

92

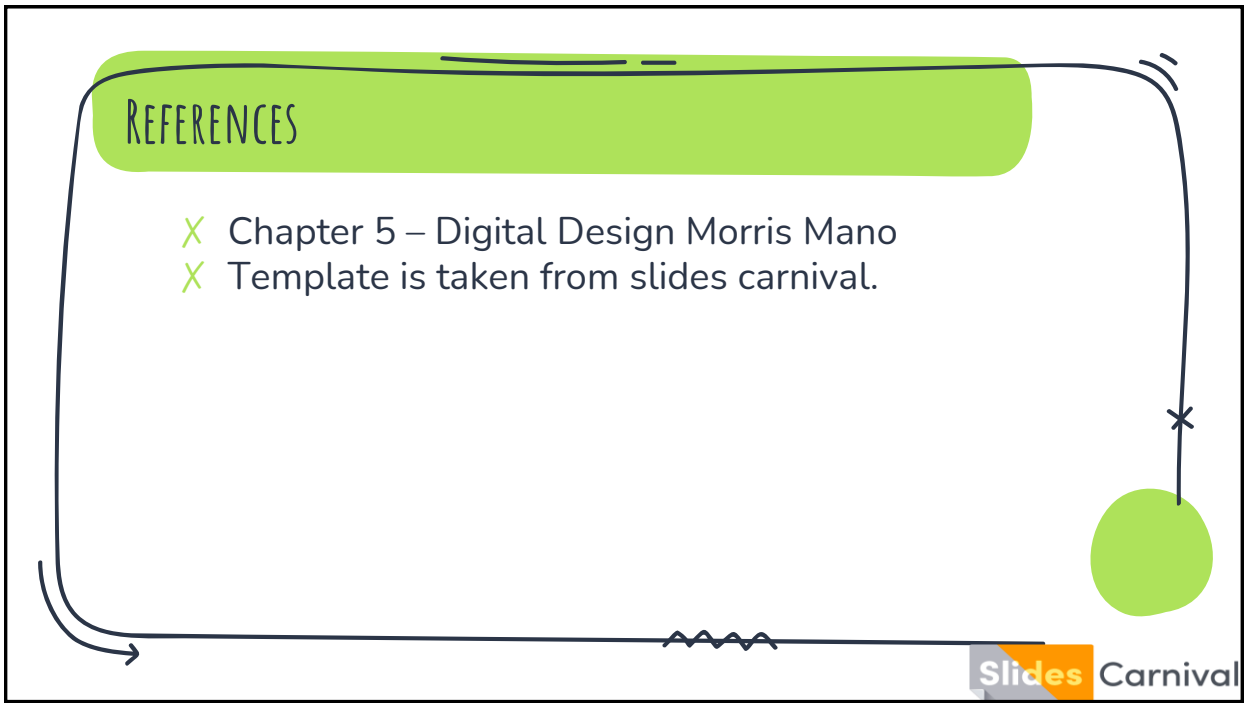
Preset and Clear – NO change when both high

Preset and Clear are asynchronous inputs.
They are used to initialize the memory to 0 or 1.

If preset is low, then q is set as 1
If clear is low, then q is cleared to 0.



93


References
Chapter 5 – Digital Design Morris Mano
Template is taken from slides carnival.
Slides Carnival

94