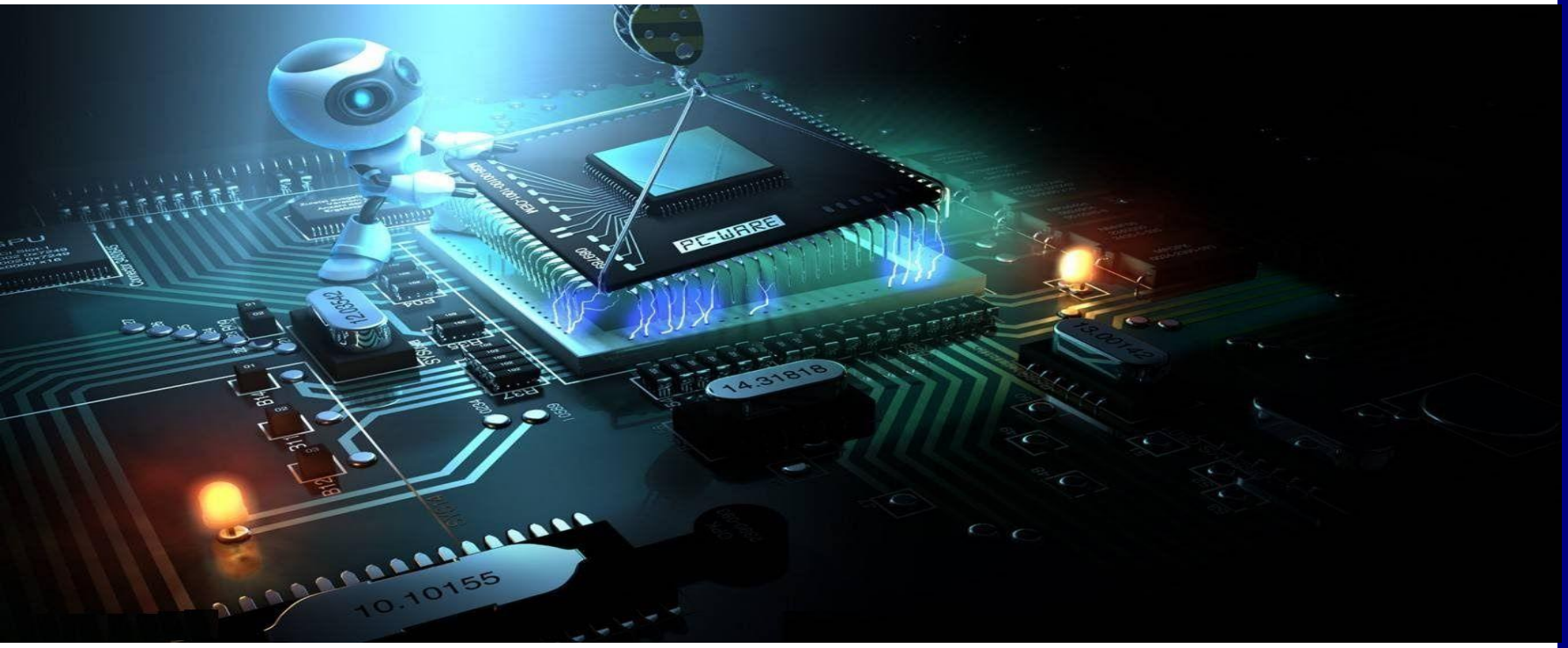
جامعة القاهرة الجديدة التكنولوجية

Fall 2023

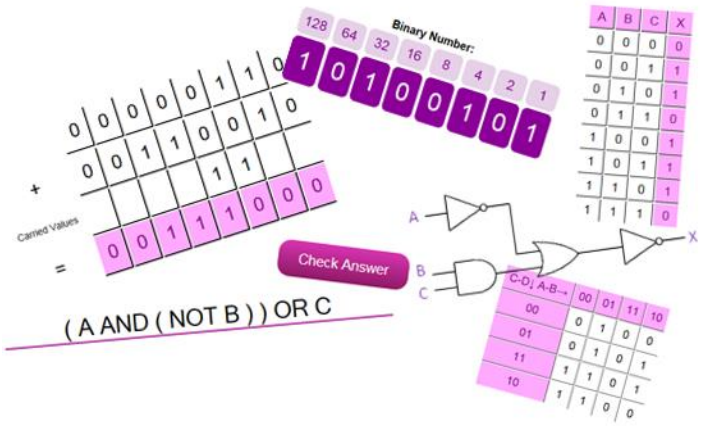
(9)



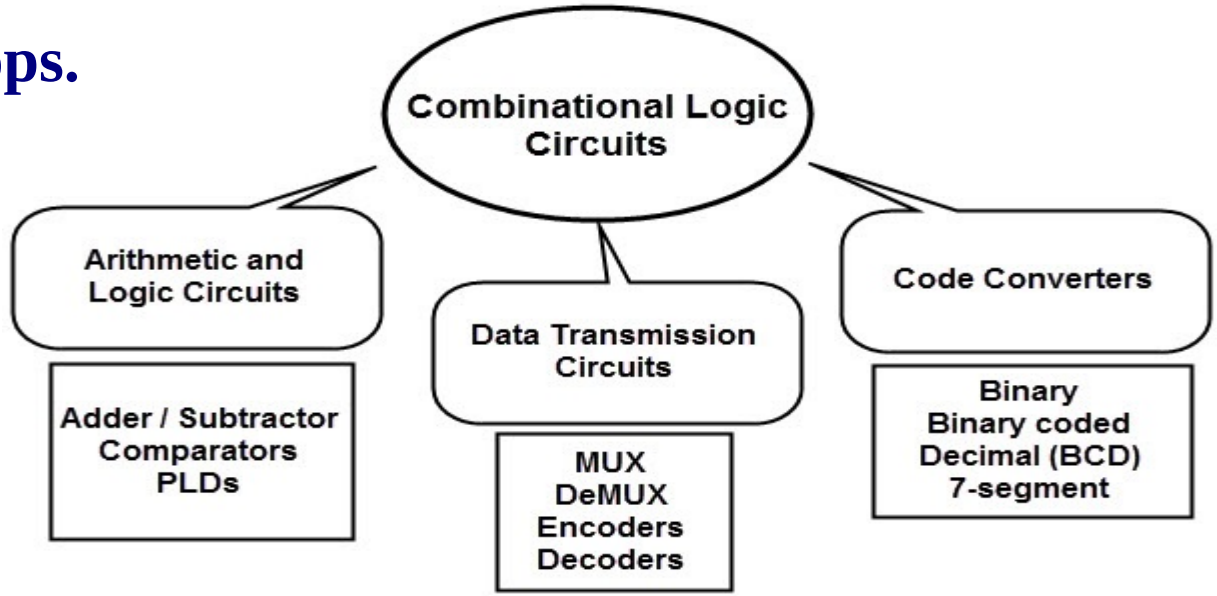
Digital Engineering

Dr. Hatem Yousry

Agenda



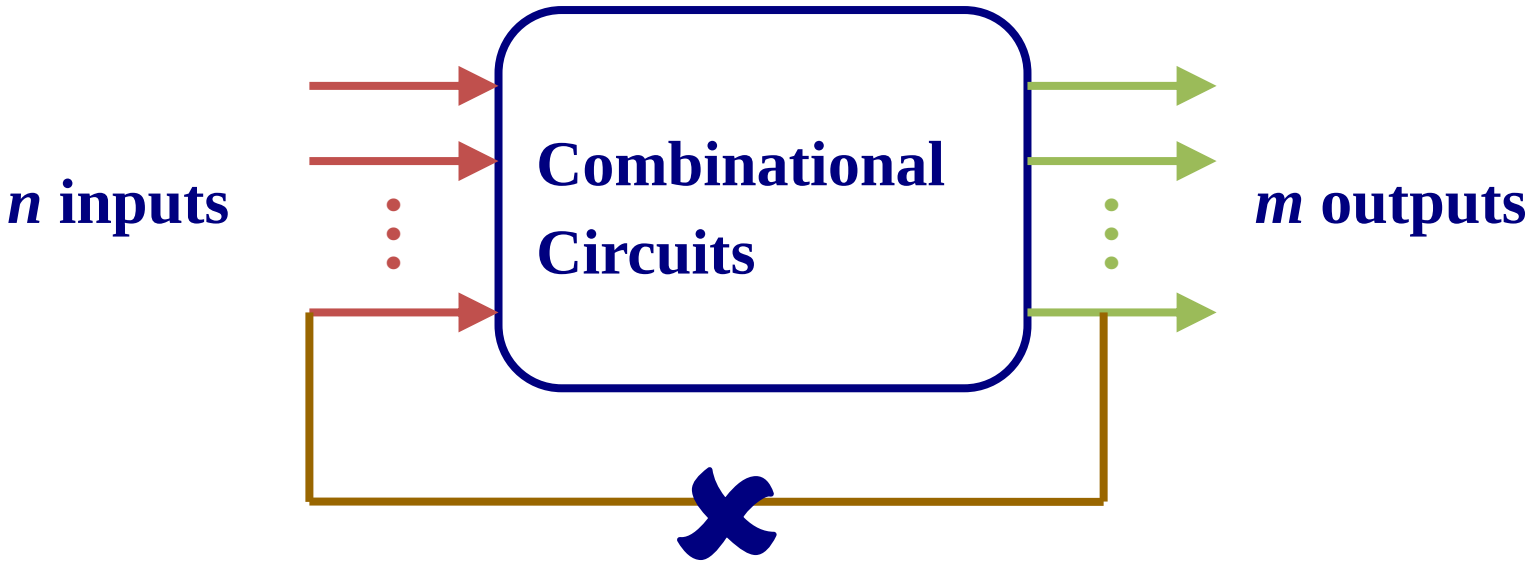
- **Combinational Circuits.**
  - **Decoders.**
  - **Binary Coded Decimal (BCD) codes.**
  - **Multiplexers.**
- **Sequential Circuits**
  - **Flip-Flops.**



Combinational Circuits

- **Output is function of input only**

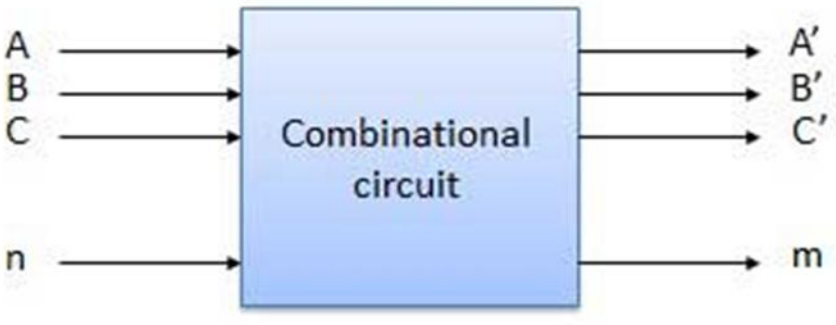
  i.e. **no feedback**

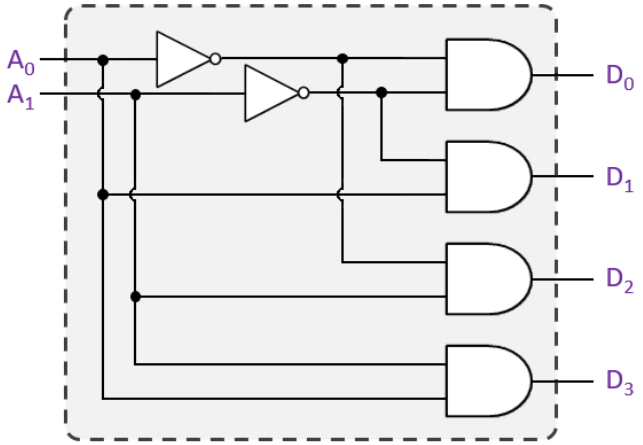


When input changes, output may change (after a delay)

Combinational Circuits

- Combinational circuits are used in a wide variety **applications** including calculators, digital measuring techniques, computers, digital processing, automatic control of machines, industrial processing, digital communications, etc.
- Combinational logic is used in computer circuits **to perform Boolean algebra on input signals and on stored data.** Practical computer circuits normally contain a mixture of combinational and sequential logic.



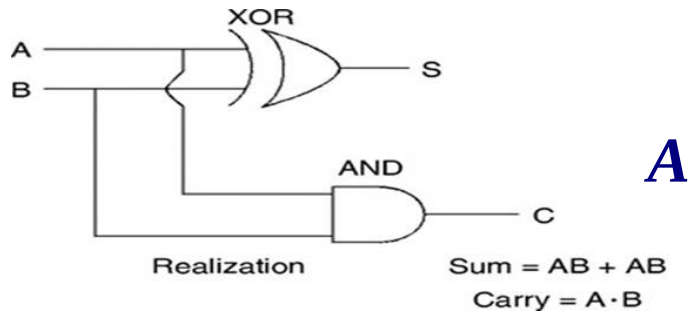


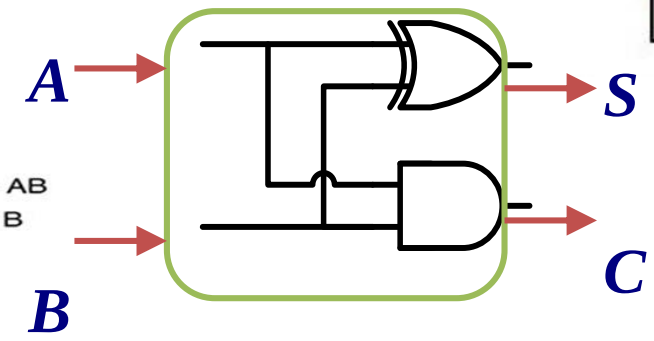
Reverse Engineering

A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

- Get the Logic Operation.
- Define the inputs and outputs.
- Construct the truth table of the logic function.
- Use K-Map based on Minterm (SOP) to get the logic function.
- Design the Combinational Circuits.

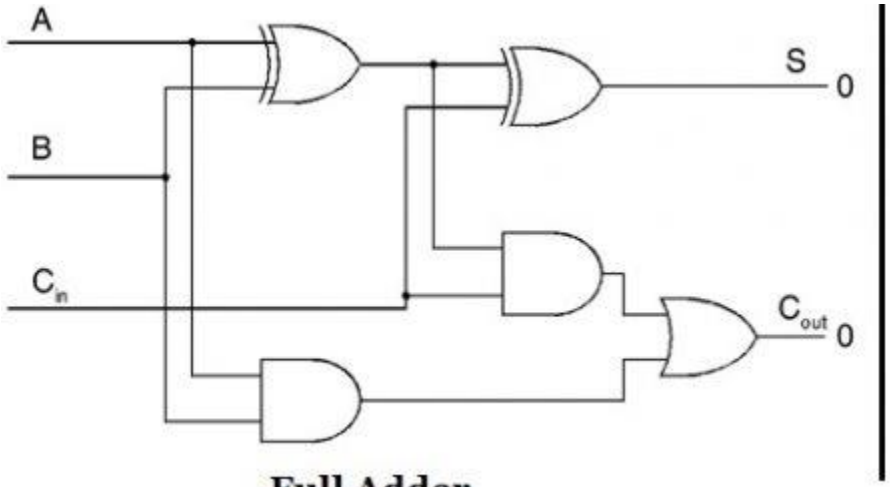
A \ B	0	1
0	0	1
1	1	0



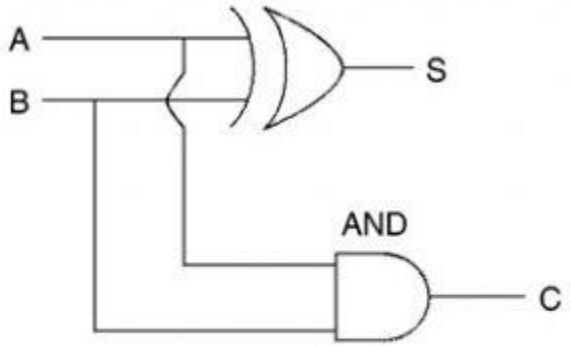


Binary Adder

- The binary adder is a combinational circuit that can perform **summation of the input binary numbers**. The most common or basic arithmetic operation is **the addition of binary digits**.
- A combination circuit which performs the additions **of two bits is a called a half adder while that performs the addition of three bits is a full adder.**



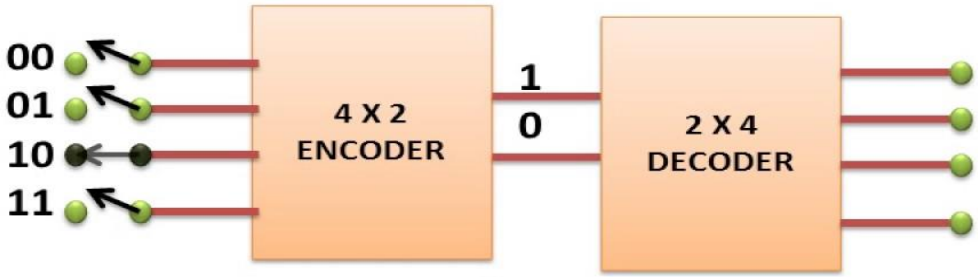
Full Adder



Half Adder

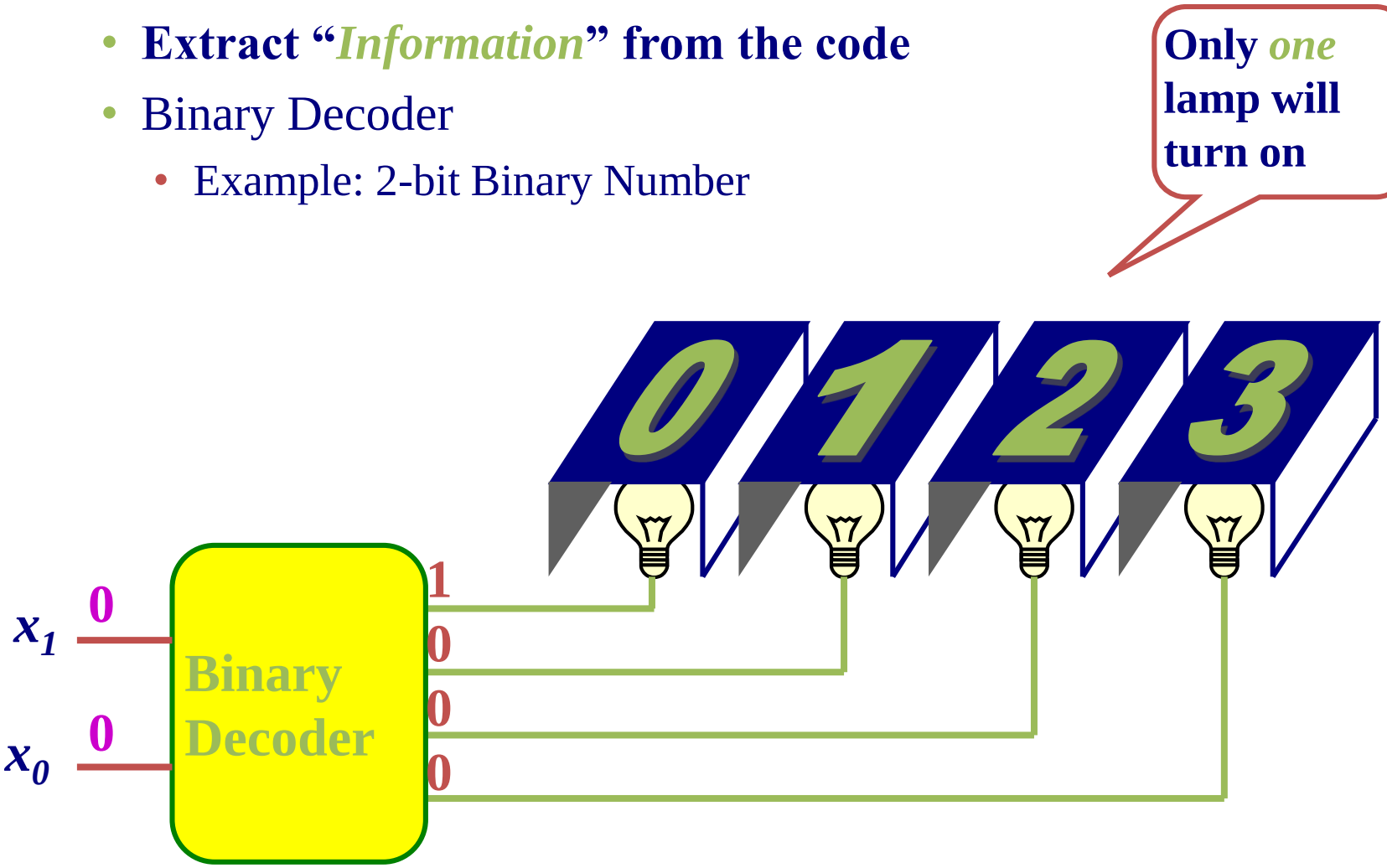
Decoder and Encoder

- A Decoder is a combinational circuit that converts binary information from n input lines to 2^n unique output lines. Apart from the Input lines, a decoder may also have an Enable input line.
- An Encoder is a combinational circuit that performs the reverse operation of Decoder. It has maximum of 2^n input lines and 'n' output lines, hence it encodes the information from 2^n inputs into an n-bit code. It will produce a binary code equivalent to the input, which is active High. Therefore, the encoder encodes 2^n input lines with 'n' bits.



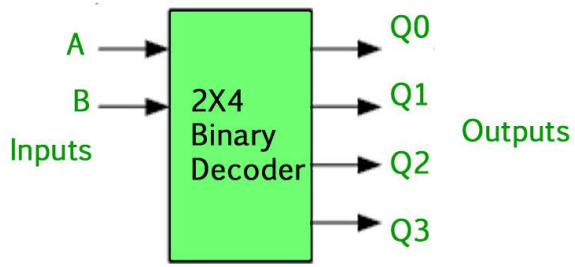
Decoders

- **Extract "*Information*" from the code**
- Binary Decoder
  - Example: 2-bit Binary Number



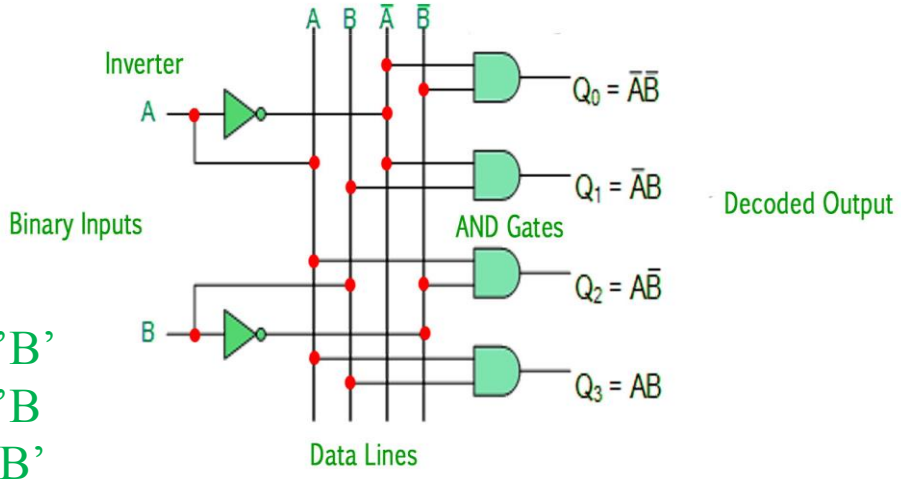
2-to-4 Line Decoder

- The 2-to-4 line binary decoder depicted above consists of an array of **four AND gates**.
- The 2 binary inputs labeled A and B are decoded into one of 4 outputs, hence the description of a 2-to-4 binary decoder. Each output represents one of the minterms of the 2 input variables, (each output = a minterm).



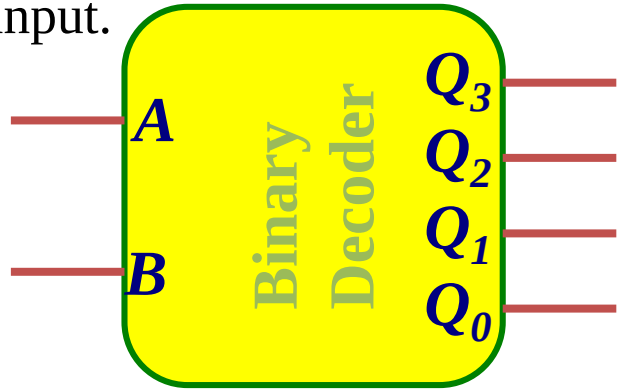
A	B	Q0	Q1	Q2	Q3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Qo=A'B'
Q1=A'B
Q2=AB'
Q3=AB

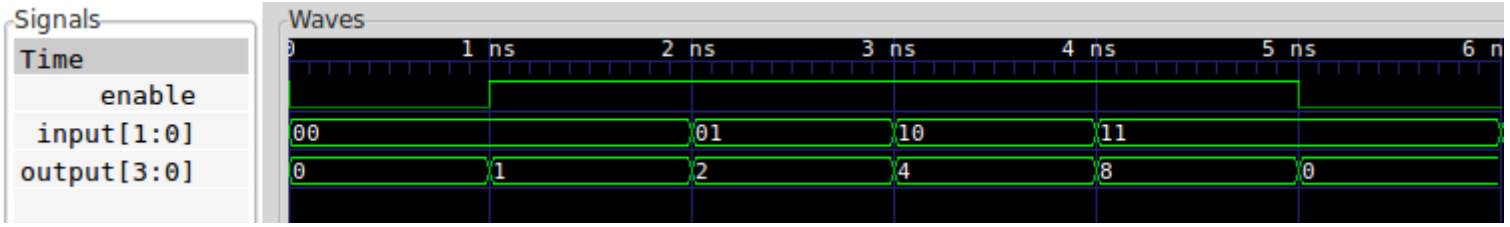


Decoders

- 2-to-4 Line Decoder
- The binary inputs A and B determine which output line from Q0 to Q3 is “HIGH” at logic level “1” while the remaining outputs are held “LOW” at logic “0” so **only one output can be active (HIGH) at any one time.** Therefore, whichever output line is “HIGH” identifies the binary code present at the input, in other words, it “decodes” the binary input.



A B	Q_3 Q_2 Q_1 Q_0
0 0	0 0 0 1
0 1	0 0 1 0
1 0	0 1 0 0
1 1	1 0 0 0



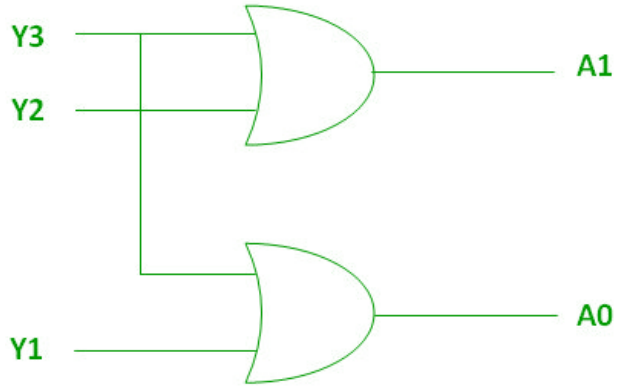
4 : 2 Encoder

- The 4 to 2 Encoder consists of four inputs Y3, Y2, Y1 & Y0 and two outputs A1 & A0. At any time, only one of these 4 inputs can be '1' in order to get the respective binary code at the output.
- The Truth table of 4 to 2 encoder is as follows :

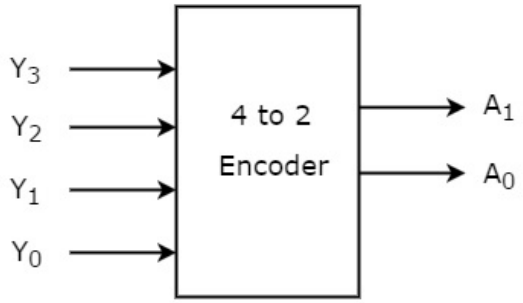
INPUTS				OUTPUTS	
Y3	Y2	Y1	Y0	A1	A0
0	0	0	1	0	0
0	0	1	0	0	1
0	1	0	0	1	0
1	0	0	0	1	1

$A1 = Y3 + Y2$

$A0 = Y3 + Y1$

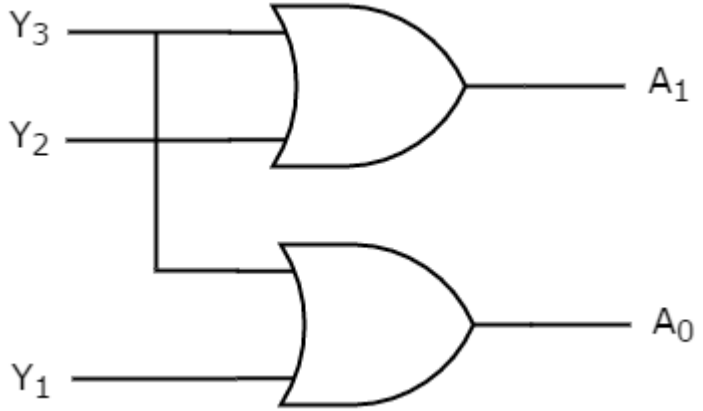


4 : 2 Encoder



- In 4 to 2 line encoder, there are total of four inputs, i.e., Y0, Y1, Y2, and Y3, and two outputs, i.e., A0 and A1.
- In 4-input lines, **one input-line is set to true at a time** to get the respective binary code in the output side.

INPUTS				OUTPUTS	
Y_3	Y_2	Y_1	Y_0	A_1	A_0
1	0	0	0	0	0
0	1	0	0	0	1
0	0	1	0	1	0
0	0	0	1	1	1

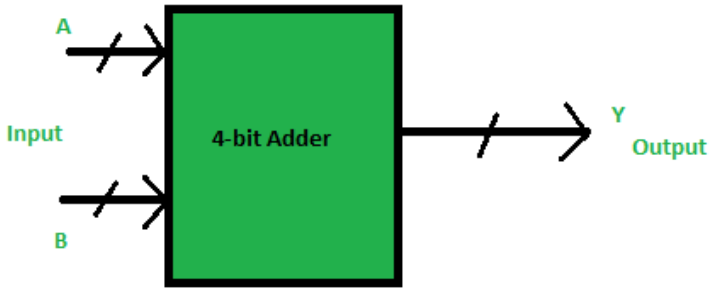


$A1 = Y3 + Y2$

$A0 = Y3 + Y1$

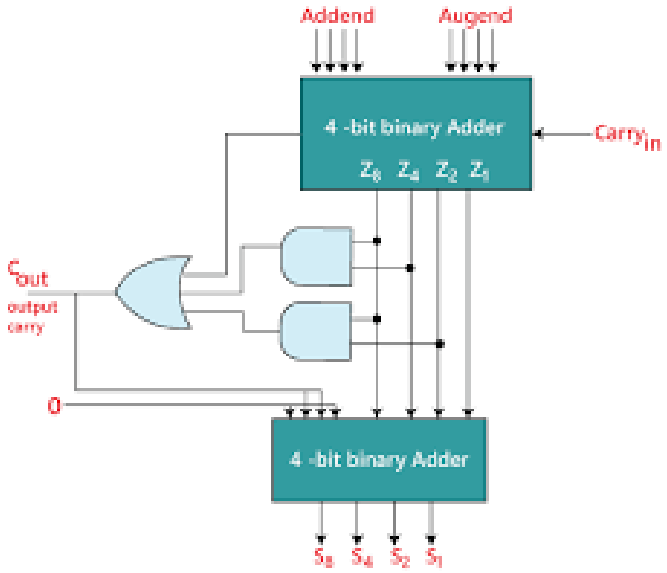
BCD Adder in Digital Logic

- BCD stand for **binary coded decimal**.
- Suppose, we have two 4-bit numbers A and B. The value of A and B can varies from 0(0000 in binary) to 9(1001 in binary) because we are considering decimal numbers.
- The output will varies from **0 to 18, if we are not considering the carry from the previous sum.** But if we are considering the carry, then the maximum value of output will be 19 (i.e. 9+9+1 = 19).
- When we are simply adding A and B, then we get the binary sum. Here, to get the output in BCD form, we will use BCD Adder.



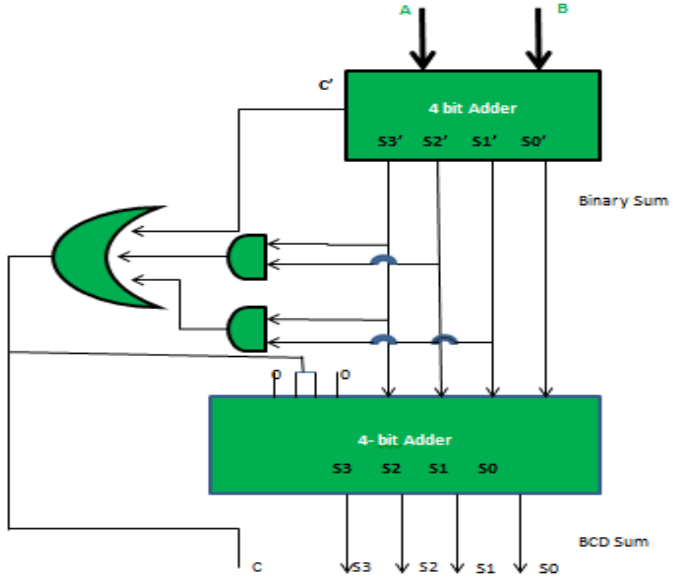
BCD Example 1:

- **Input :**
- **A = 0111 B = 1000**
- **Output :**
- **Y = 1 0101**
- **Explanation:**
- We are adding A(=7) and B(=8).
- The value of binary sum will be 1111(=**15**).
- But, the BCD sum will be **1 0101**,
- where 1 is 0001 in binary and 5 is 0101 in binary.



Black diagram of a BCD adder

BCD Example 2:



- **Input :**
- **A = 0101 B = 1001**
- **Output :**
- **Y = 1 0100**
- **Explanation:**
- We are adding A(=5) and B(=9).
- The value of binary sum will be 1110(=**14**).
- But, the BCD sum will be **1 0100,**
- where 1 is 0001 in binary and 4 is 0100 in binary.

BCD Implementation

- Note – If the sum of two number is less than or equal to 9, then the value of BCD sum and binary sum will be same otherwise they will differ by **6**(**0110** in binary).
- Now, lets move to the table and find out the logic when we are going to add "0110".
- We are adding **"0110" (=6)** only to the second half of the table.

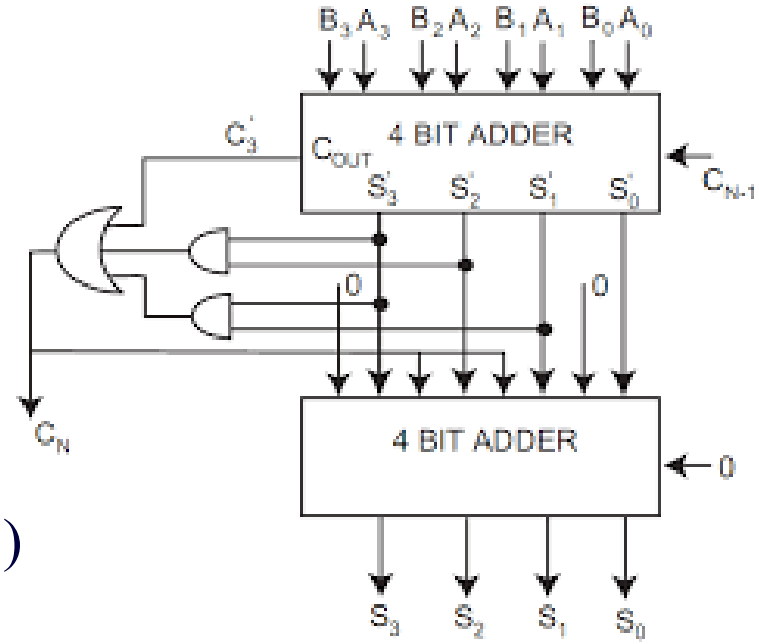
Decimal	Binary Sum					BCD Sum				
	C'	S3'	S2'	S1'	S0'	C	S3	S2	S1	S0
0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	1	0	0	0	0	1
2	0	0	0	1	0	0	0	0	1	0
3	0	0	0	1	1	0	0	0	1	1
4	0	0	1	0	0	0	0	1	0	0
5	0	0	1	0	1	0	0	1	0	1
6	0	0	1	1	0	0	0	1	1	0
7	0	0	1	1	1	0	0	1	1	1
8	0	1	0	0	0	0	1	0	0	0
9	0	1	0	0	1	0	1	0	0	1
10	0	1	0	1	0	1	0	0	0	0
11	0	1	0	1	1	1	0	0	0	1
12	0	1	1	0	0	1	0	0	1	0
13	0	1	1	0	1	1	0	0	1	1
14	0	1	1	1	0	1	0	1	0	0
15	0	1	1	1	1	1	0	1	0	1
16	1	0	0	0	0	1	0	1	1	0
17	1	0	0	0	1	1	0	1	1	1
18	1	0	0	1	0	1	1	0	0	0
19	1	0	0	1	1	1	1	0	0	1

BCD Implementation

- The conditions are:

- If C’ = 1 (Satisfies 16-19)
- If S3′.S2′ = 1 (Satisfies 12-15)
- If S3′.S1′ = 1 (Satisfies 10 and 11)
- So, our logic is

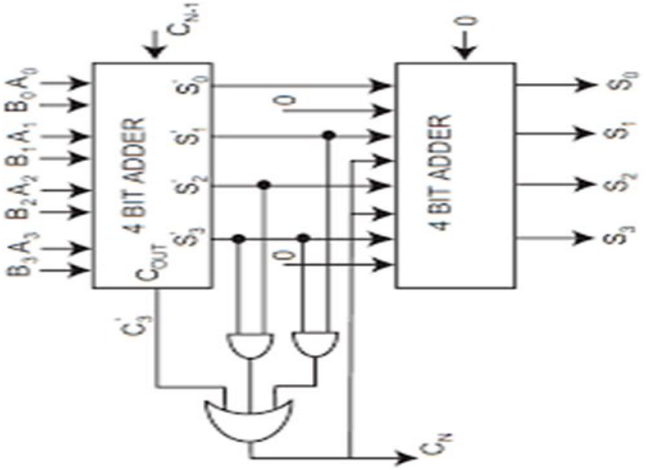
- C' + S3'.S2' + S3'.S1' = 1

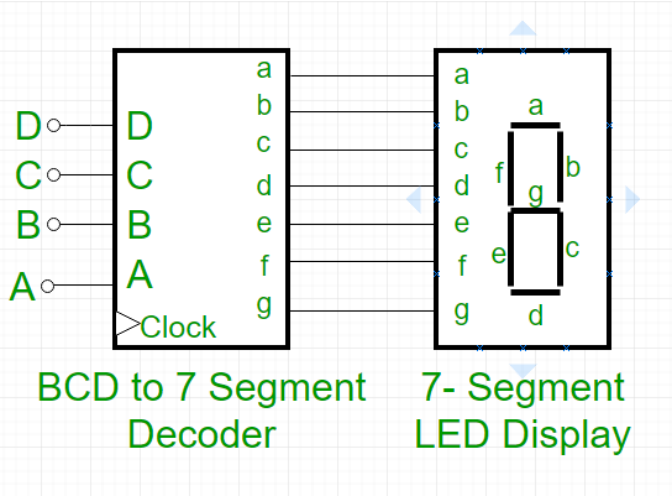


Binary Sum						BCD Sum						Decimal
K	Z_8	Z_4	Z_2	Z_1		C	S_8	S_4	S_2	S_1		
0	0	0	0	0	SAME CODE	0	0	0	0	0		0
0	0	0	0	1		0	0	0	0	1		1
0	0	0	1	0		0	0	0	1	0		2
-	-	-	-	-		-	-	-	-	-		-
-	-	-	-	-		-	-	-	-	-		-
-	-	-	-	-		-	-	-	-	-		-
-	-	-	-	-		-	-	-	-	-		-
0	1	0	0	0		0	1	0	0	0		8
0	1	0	0	1		0	1	0	0	1		9
10 to 19 Binary and BCD codes are not the same												
0	1	0	1	0		1	0	0	0	0		10
0	1	0	1	1		1	0	0	0	1		11
0	1	1	0	0		1	0	0	1	0		12
0	1	1	0	1		1	0	0	1	1		13
0	1	1	1	0		1	0	1	0	0		14
0	1	1	1	1		1	0	1	0	1		15
1	0	0	0	0		1	0	1	1	0		16
1	0	0	0	1		1	0	1	1	1		17
1	0	0	1	0		1	1	0	0	0		18
1	0	0	1	1		1	1	0	0	1		19

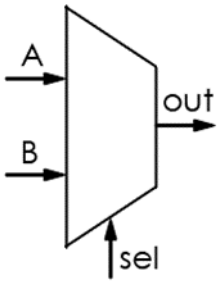
BCD to 7 Segment Decoder

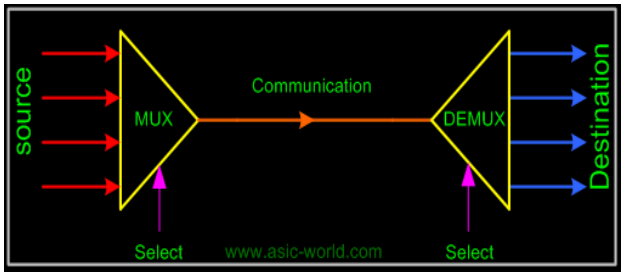
- In Binary Coded Decimal (BCD) encoding scheme each of the decimal numbers(0-9) is represented by its equivalent binary pattern(which is generally of 4-bits).
- Whereas, Seven segment display is an electronic device which consists of seven Light Emitting Diodes (LEDs) arranged in a some definite pattern (common cathode or common anode type), which is used to display Hexadecimal numerals(in this case decimal numbers, as input is BCD i.e., 0-9).



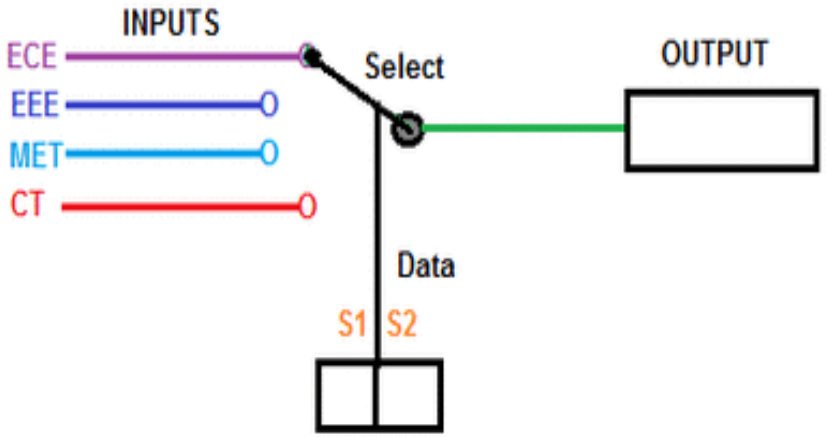


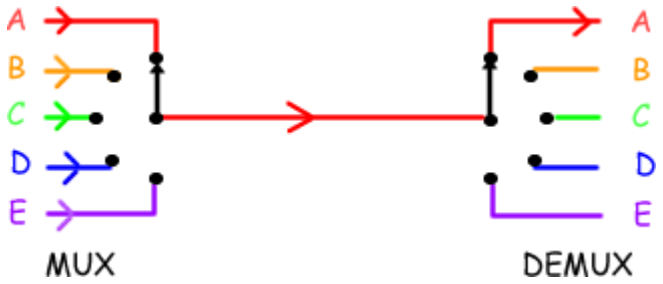
Multiplexers



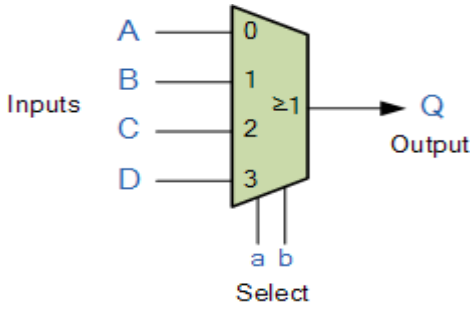


- A **multiplexer** (or mux)
- **a data selector**, is a device that selects between several analog or digital input signals and forwards the selected input to a single output line. **The selection is directed a separate set of digital inputs known as select lines.**

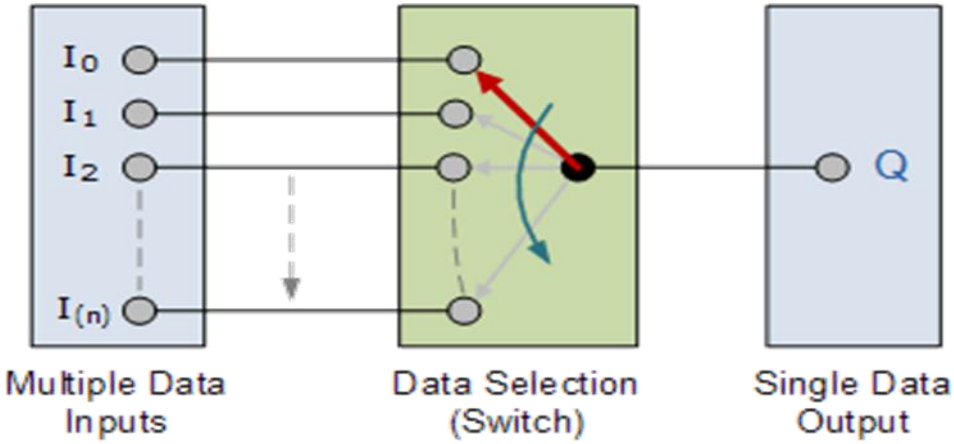




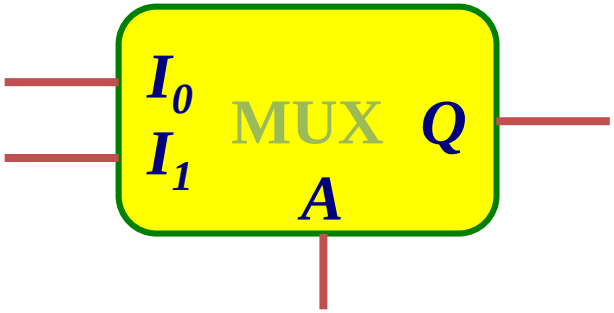
The Multiplexer



- The multiplexer is a combinational logic circuit designed **to switch one of several input lines to a single common output line.**
- The multiplexer, shortened to "MUX" or "MPX", is a combinational logic circuit designed to switch one of several input lines through to a single common output line by the application of a control signal.
- Multiplexers operate like very fast acting multiple position rotary switches connecting or controlling multiple input lines called "channels" one at a time to the output.



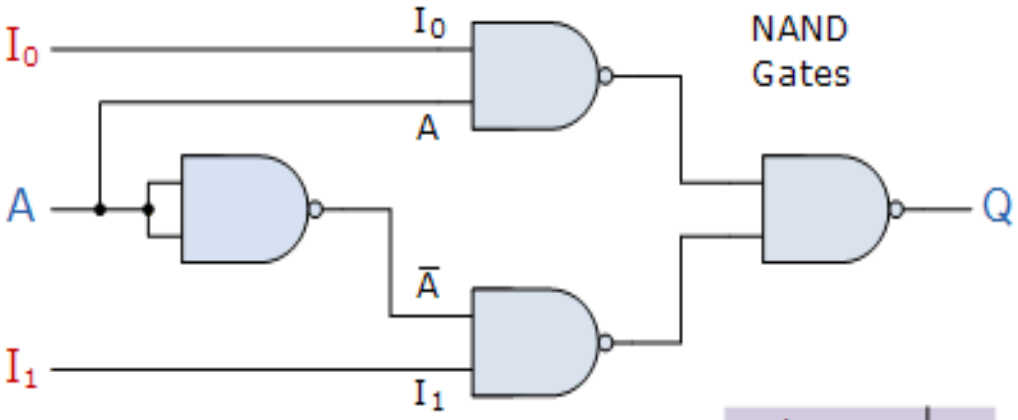
Multiplexers

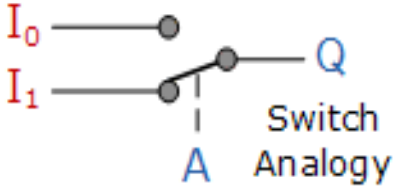


- 2-to-1 MUX

The input A of this simple 2-1 line multiplexer circuit constructed from standard NAND gates acts to control which input (I0 or I1) gets passed to the output at Q.

So by the application of either a logic "0" or a logic "1" at A we can select the appropriate input, I0 or I1 with the circuit acting a bit like a single pole double throw (SPDT) switch.





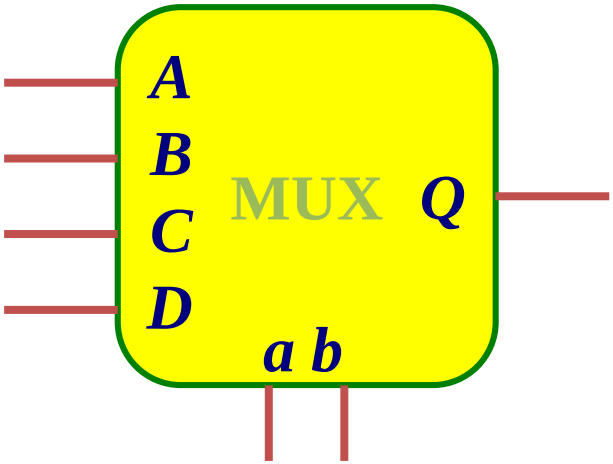
Truth Table

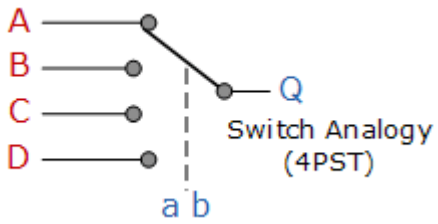
Inputs			
A	I_1	I_0	Q
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

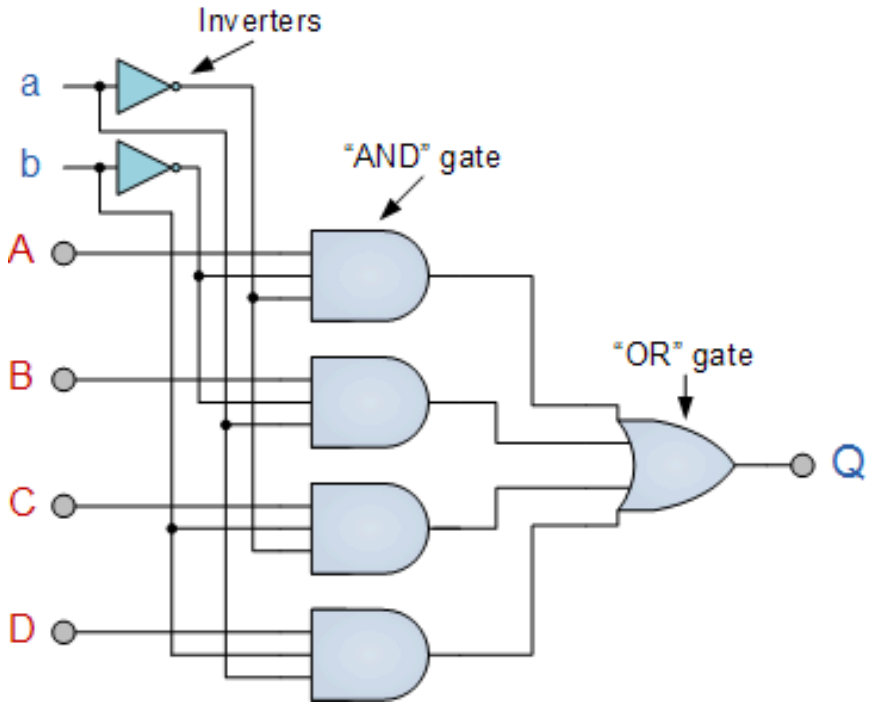
$Q = A'.I1 + A.I0$

Multiplexers

- 4-to-1 MUX





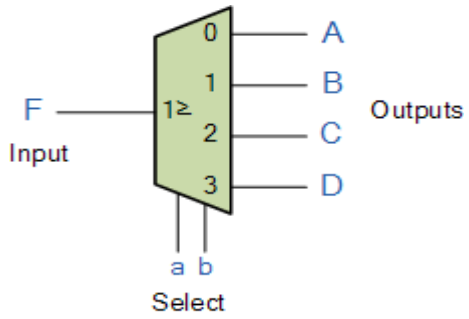


Truth Table

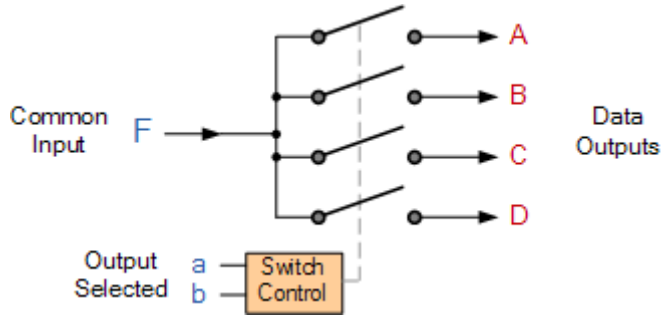
Select		Inputs				
b	a	D	C	B	A	Q
0	0	x	x	x	1	1
0	1	x	x	1	x	1
1	0	x	1	x	x	1
1	1	1	x	x	x	1

Q = **a'b**'A + a**b'**B + **a'**bC + abD

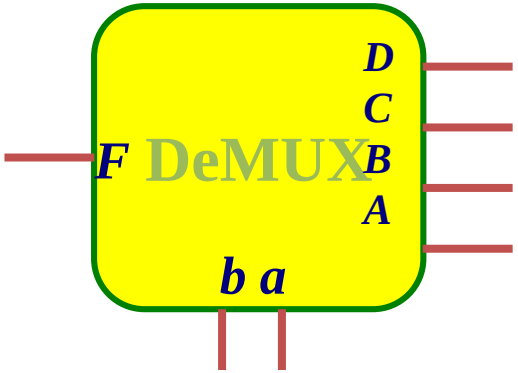
De-multiplexer



- The demultiplexer is a combinational logic circuit designed to switch one common input line to one of several separate output line.
- The data distributor, known more commonly as a Demultiplexer or **"Demux"** for short, is the exact opposite of the Multiplexer we saw in the previous tutorial.
- The demultiplexer takes one single input data line and then switches it to any one of a number of individual output lines one at a time. The demultiplexer converts a serial data signal at the input to a parallel data at its output lines as shown below.



1-to-4 Channel De-multiplexer

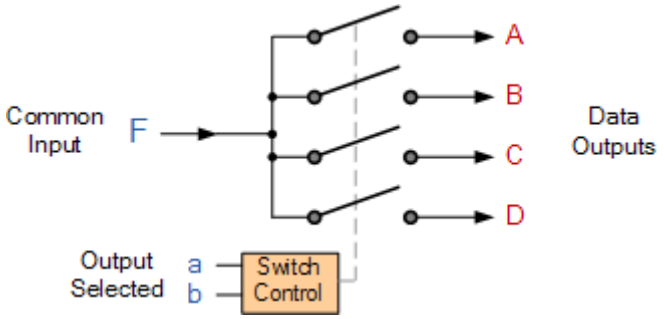


The Boolean expression for this 1-to-4 **Demultiplexer** above with outputs A to D and data select lines a, b is given as:

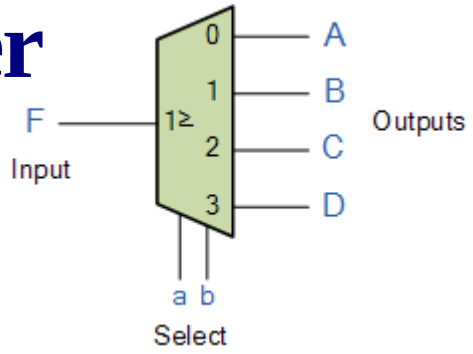
F = **a'b'**A + a**b'**B + **a'**bC + abD

The function of the **Demultiplexer** is to switch one common data input line to any one of the 4 output data lines A to D.
As with the multiplexer the individual solid state switches are selected by the binary input address code on the output select pins "a" and "b" as shown.

Output Select		Data Output Selected
b	a	
0	0	A
0	1	B
1	0	C
1	1	D



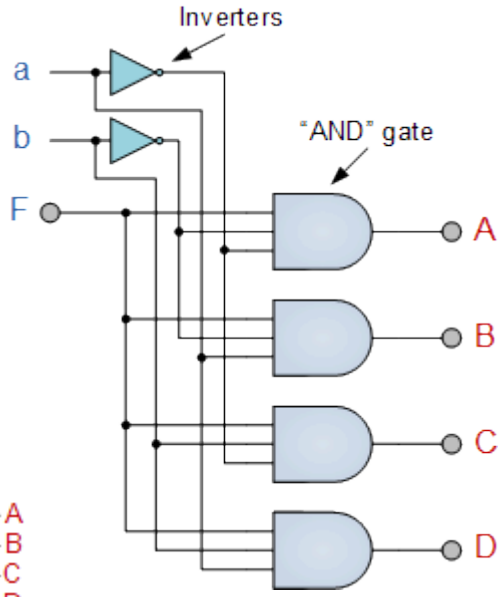
1-to-4 Channel De-multiplexer

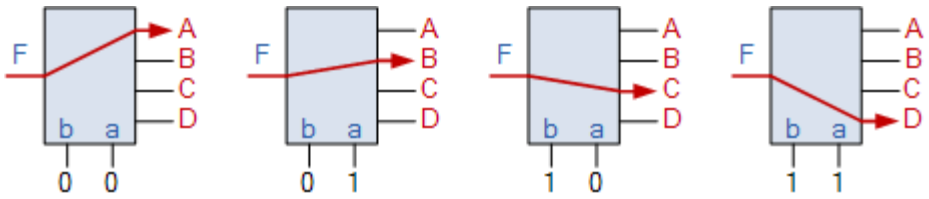


- **F = a'b'A + ab'B + a'bC + abD**

- The function of the Demultiplexer is to switch one common data input line to any one of the 4 output data lines A to D.

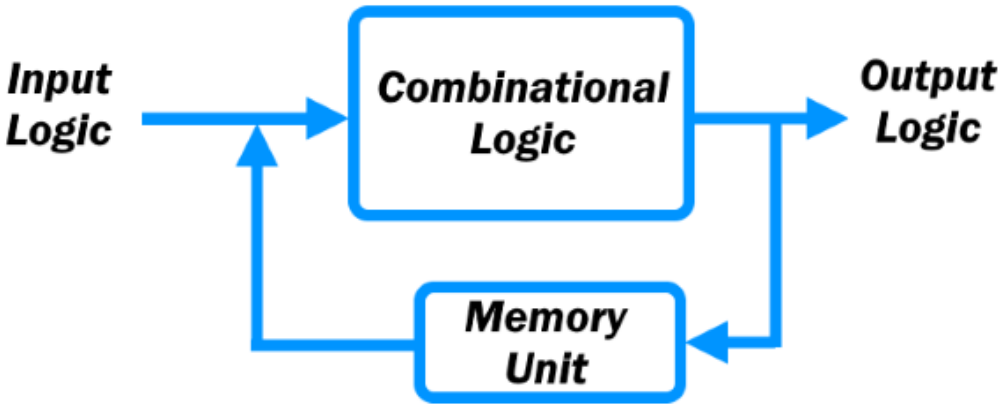
Output Select		Data Output Selected
b	a	
0	0	A
0	1	B
1	0	C
1	1	D





Combinational and Sequential Circuits

- **Combinational circuits** are defined as the time independent **circuits** which do not depends upon previous inputs to generate any output are termed as **combinational circuits**.
- **Sequential circuits** are those which are dependent on clock cycles and depends on present as well as past inputs to generate any output.

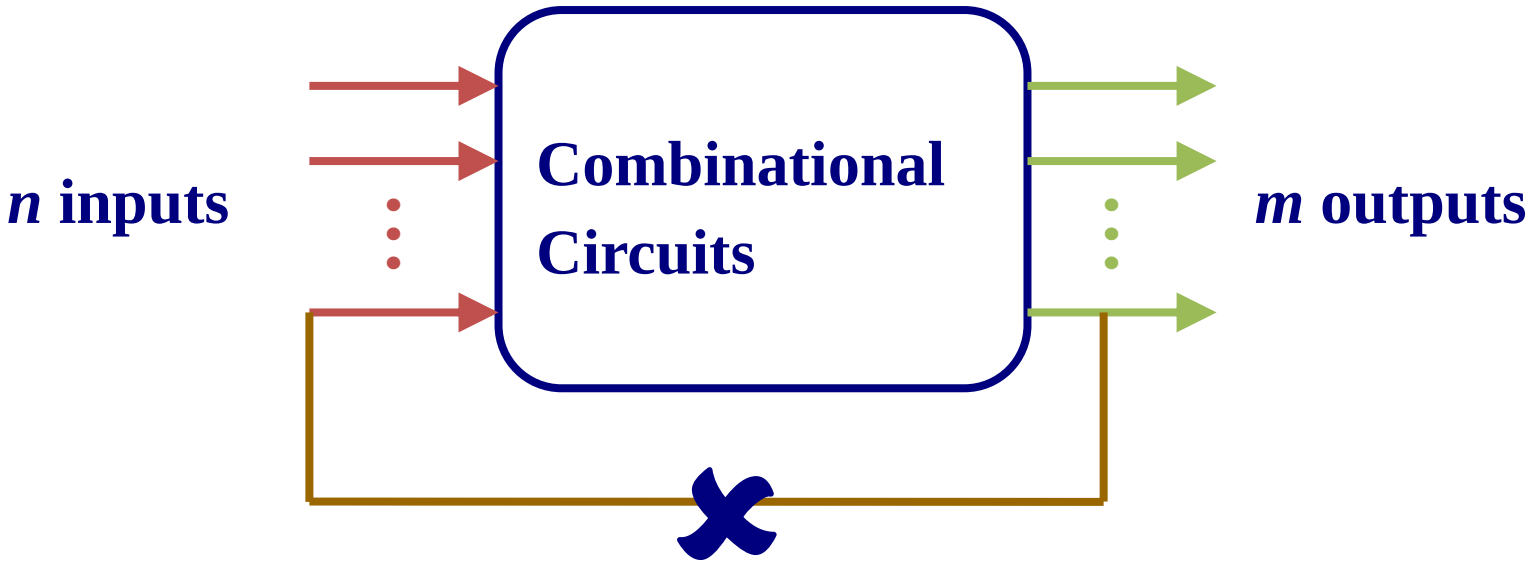


Sequential Logic Circuit

Combinational Circuits

- **Output is function of input only**

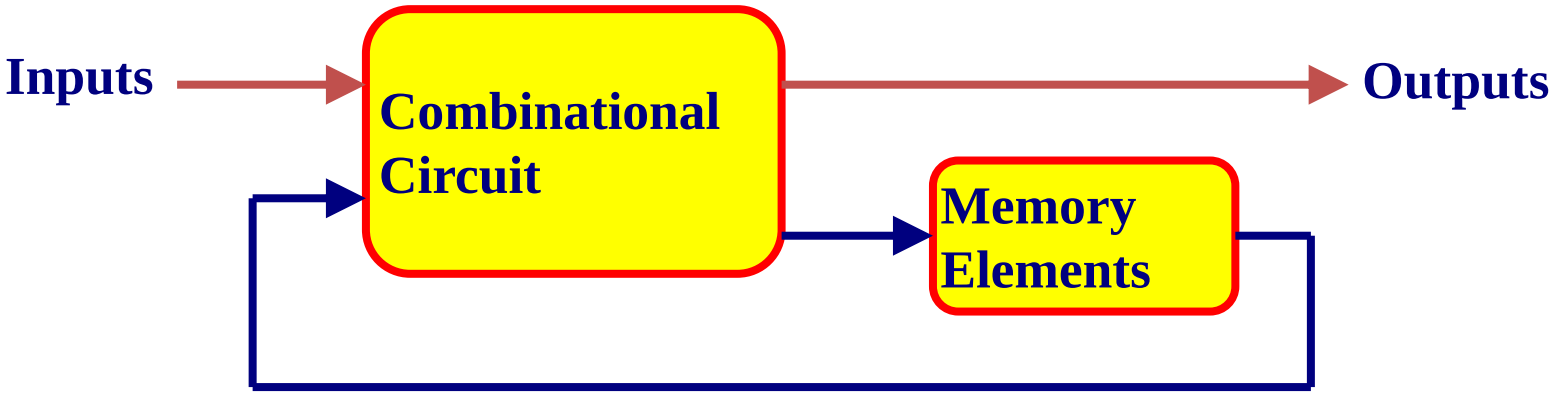
  i.e. **no feedback**



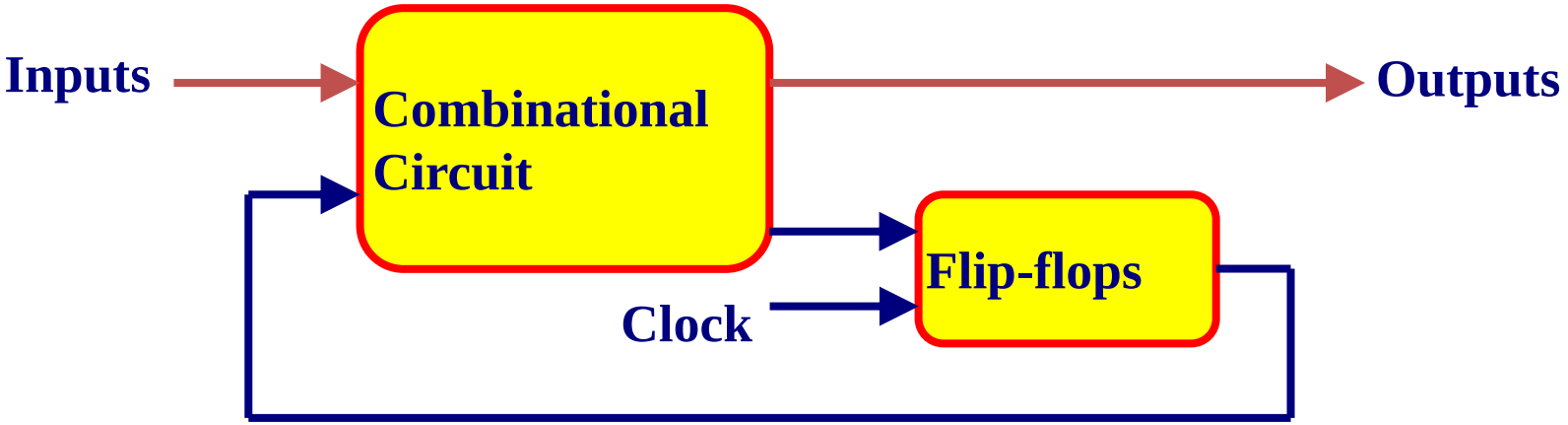
When input changes, output may change (after a delay)

Sequential Circuits

- Asynchronous

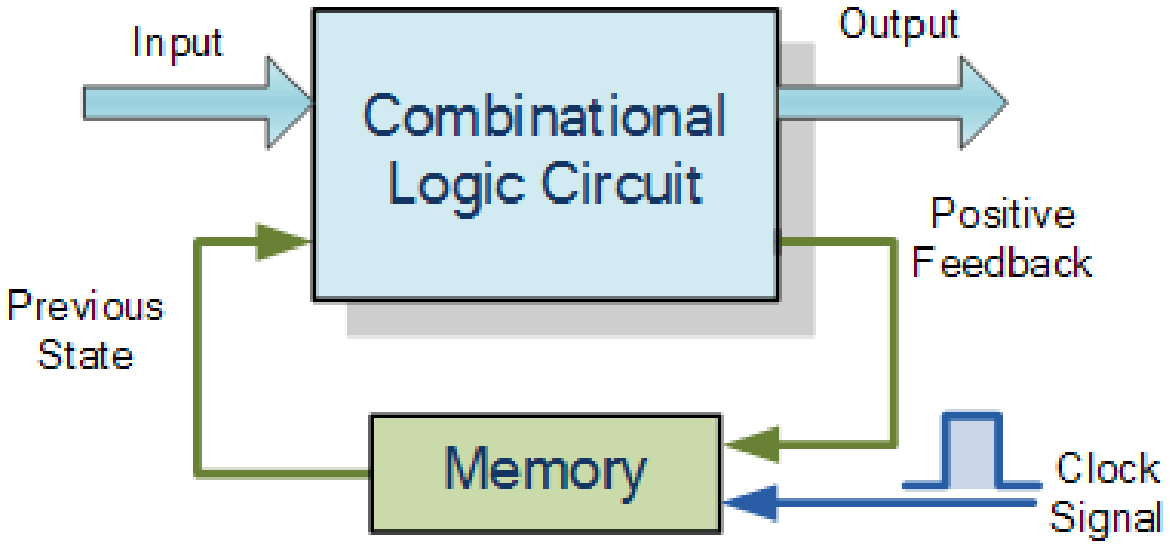


- Synchronous

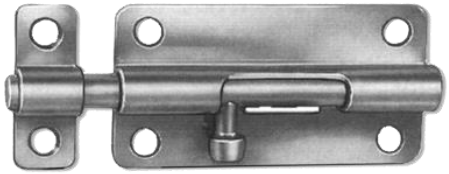


Sequential Circuits

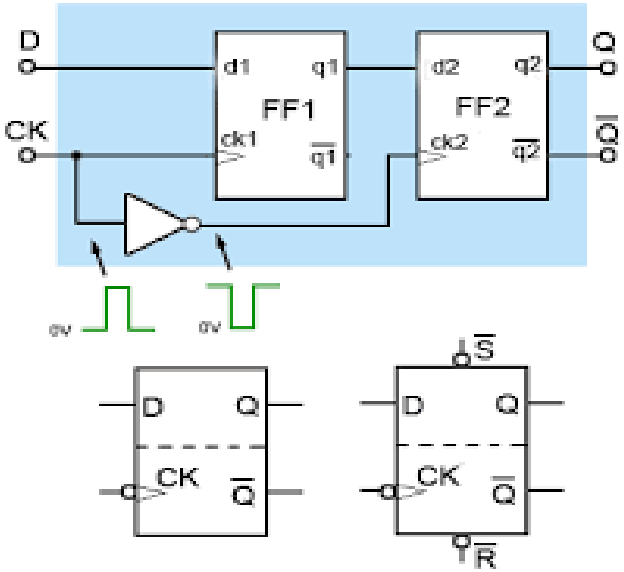
- A Sequential circuit combinational logic circuit that consists of inputs variable (X), logic gates (Computational circuit), and output variable (Z). Combinational circuit produces an output based on input variable only, **but Sequential circuit produces an output based on current input and previous input variables.**

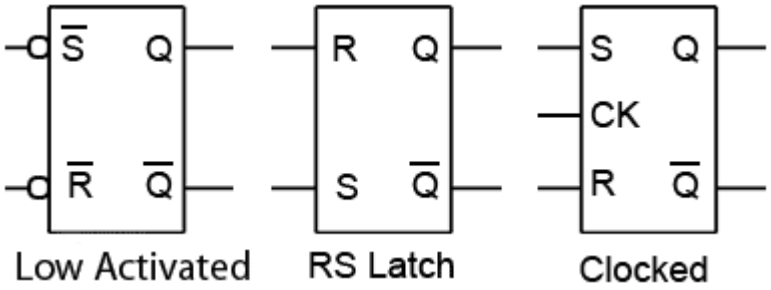


Latches

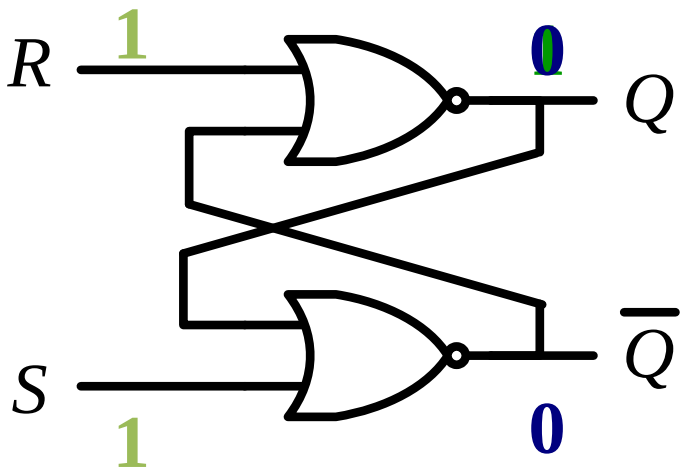


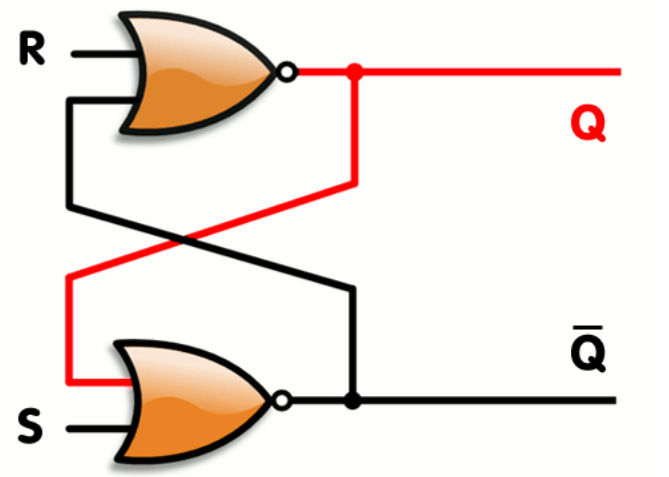
- Latches are **basic storage elements that operate with signal levels (rather than signal transitions).**
- Latches controlled by a **clock transition are flip-flops**. Latches are level-sensitive devices.
- Latches are useful for the design of the asynchronous sequential circuit.





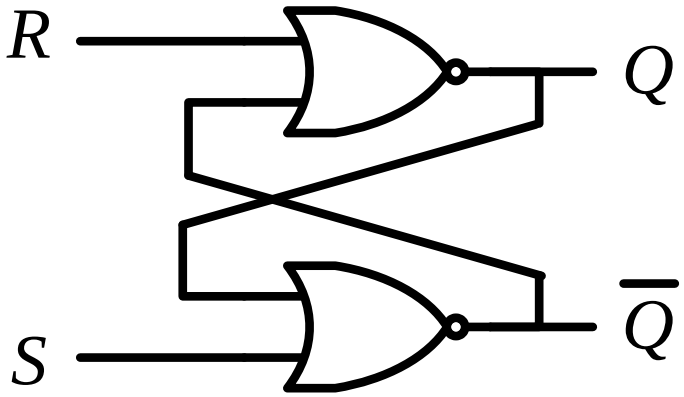
SR Latch



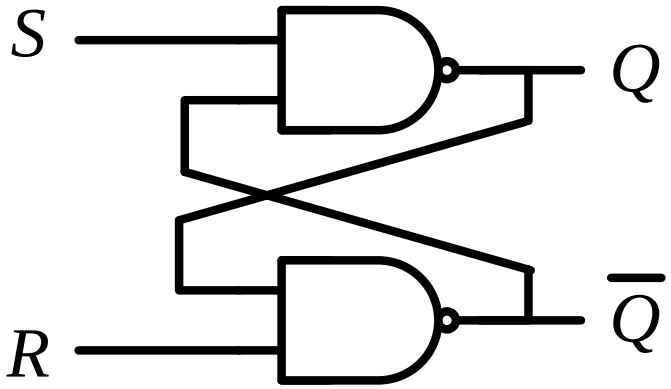


S	R	Q_0	Q	Q'	
0	0	0	0	1	$Q = Q_0$
0	0	1	1	0	
0	1	0	0	1	$Q = 0$
0	1	1	0	1	
1	0	0	1	0	$Q = 1$
1	0	1	1	0	
1	1	0	0	0	$Q = Q'$
1	1	1	0	0	$Q = Q'$

SR Latch

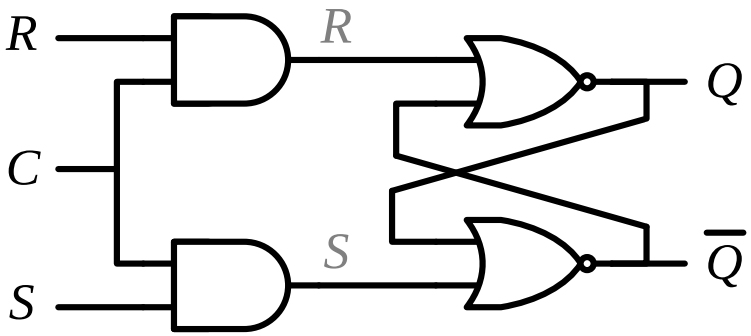


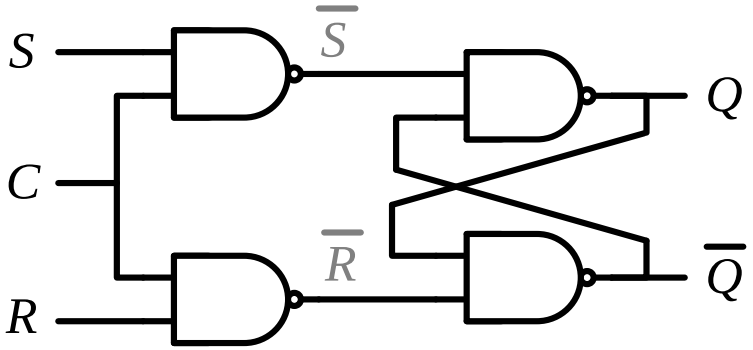
S	R	Q	
0	0	Q_0	No change
0	1	0	Reset
1	0	1	Set
1	1	$Q=Q'=0$	Invalid



S	R	Q	
0	0	$Q=Q'=1$	Invalid
0	1	1	Set
1	0	0	Reset
1	1	Q_0	No change

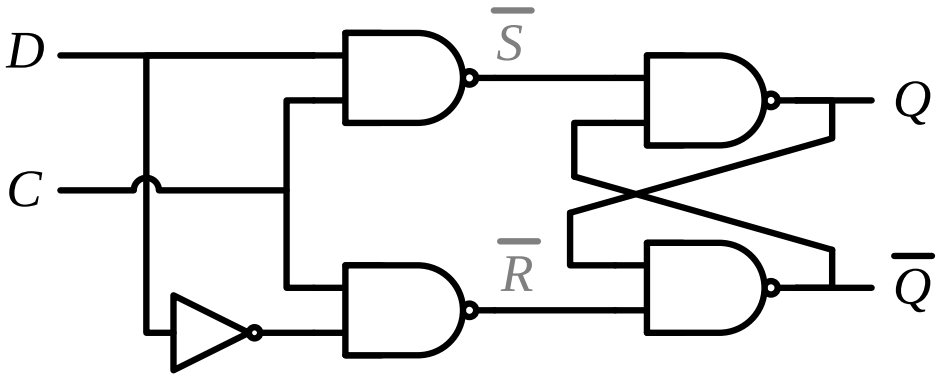
Controlled Latches





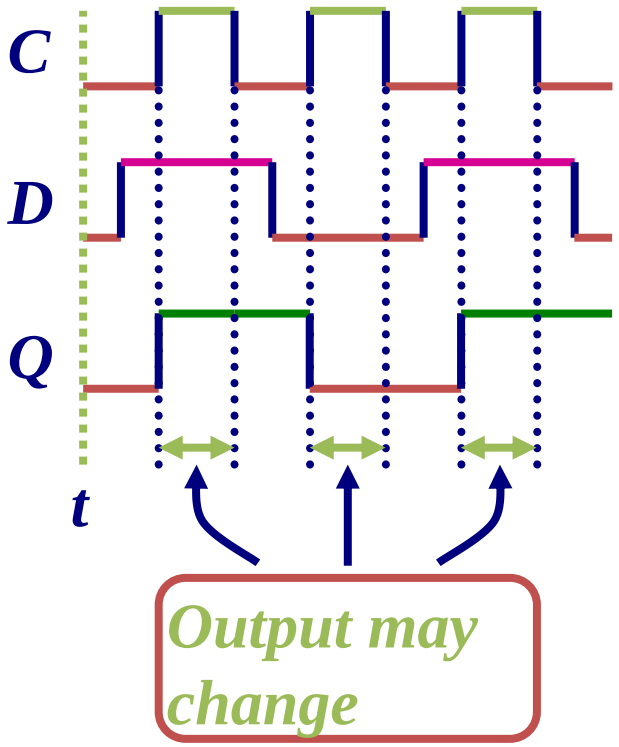
C S R	Q	
0 x x	Q_0	No change
1 0 0	Q_0	No change
1 0 1	0	Reset
1 1 0	1	Set
1 1 1	Q=Q'	Invalid

D Latch (D = Data)

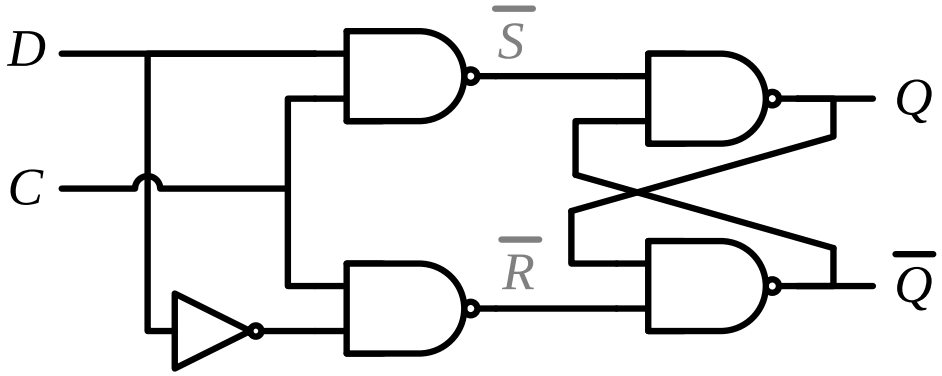


C	D	Q	
0	x	Q_0	No change
1	0	0	Reset
1	1	1	Set

Timing Diagram

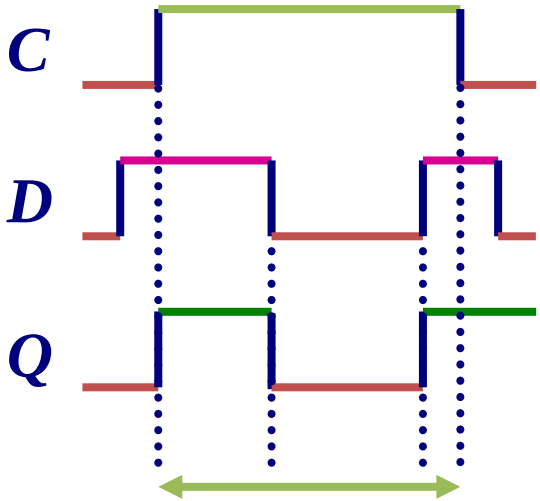


D Latch (D = Data)



C	D	Q	
0	x	Q_0	No change
1	0	0	Reset
1	1	1	Set

Timing Diagram



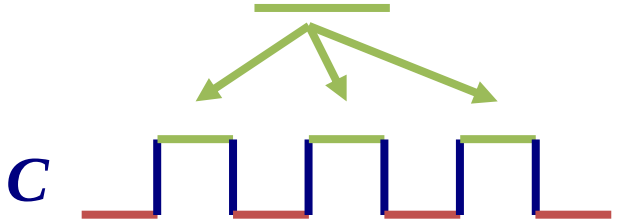
Output may change

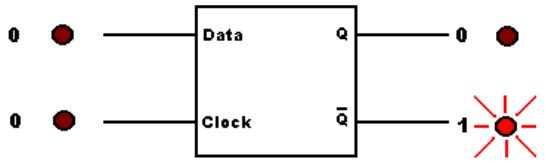
Flip-Flops

- **A flip flop is an electronic circuit with two stable states that can be used to store binary data.**
- The stored data can be changed by applying varying inputs.
- Flip-flops and latches are fundamental building blocks of digital electronics systems used in computers, communications, and many other types of systems.
- Flip-flops and latches are used as data storage elements.
- It is the basic storage element in sequential logic. But first, let's clarify the difference between a latch and a flip-flop.

Flip-Flops

- Controlled latches are level-triggered



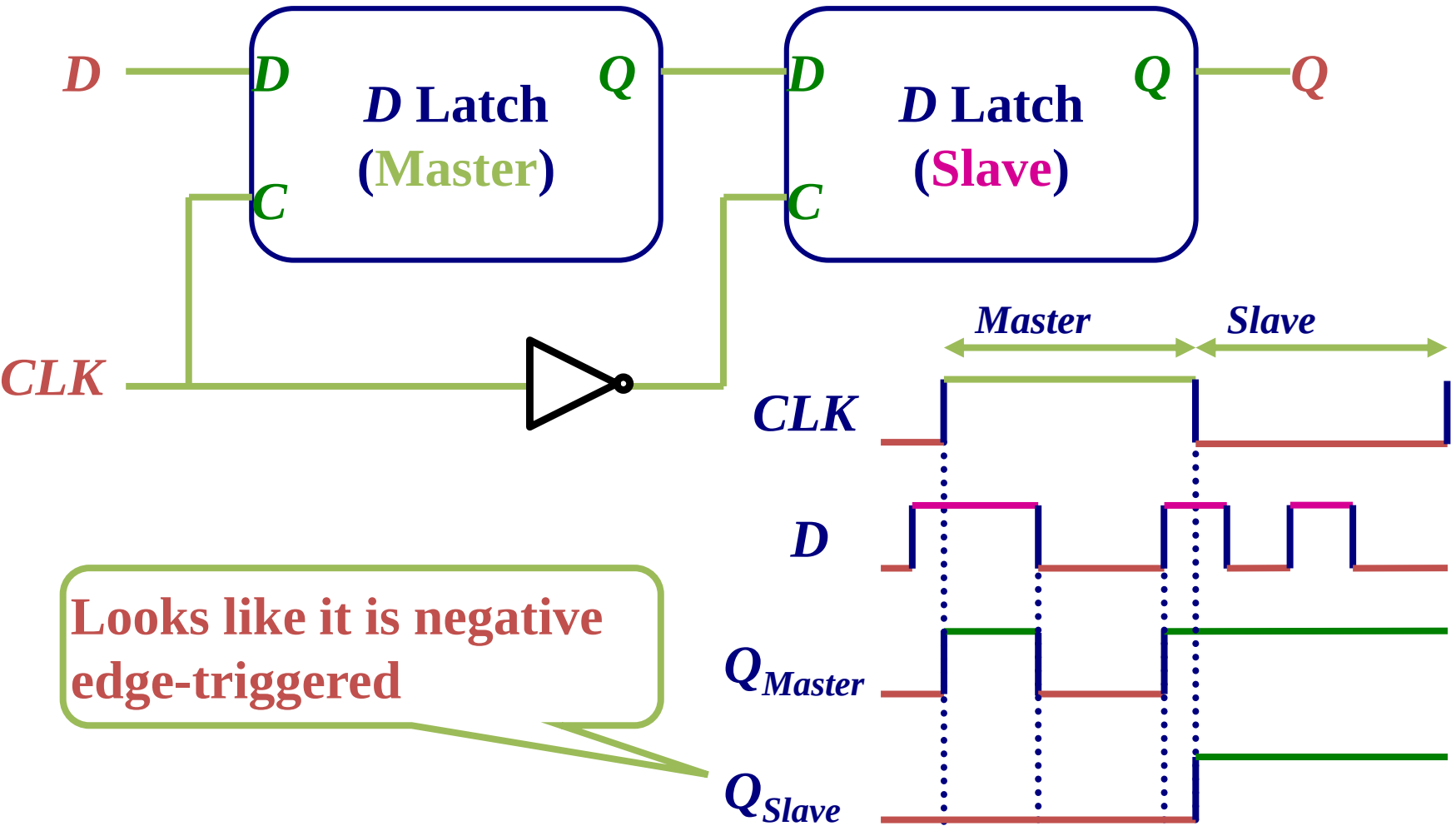


- Flip-Flops are edge-triggered

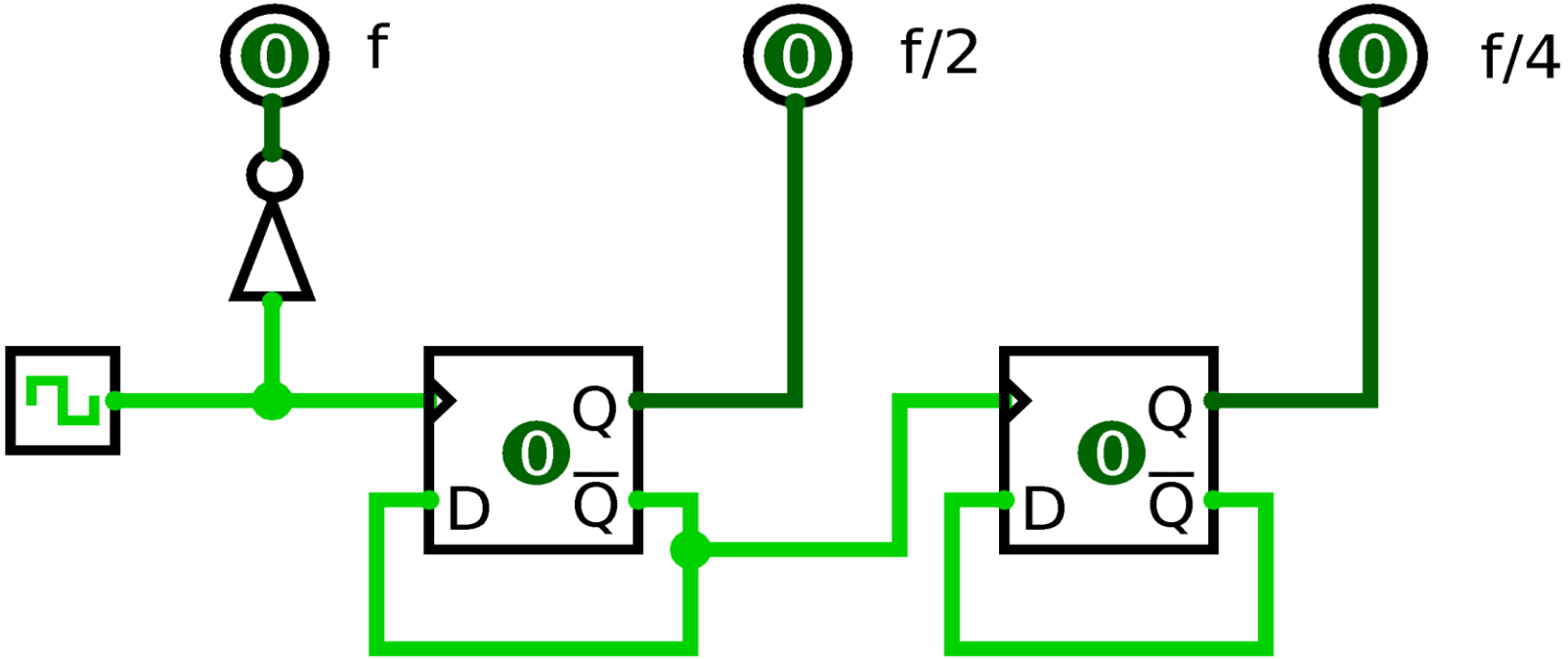
Positive Edge

Negative Edge

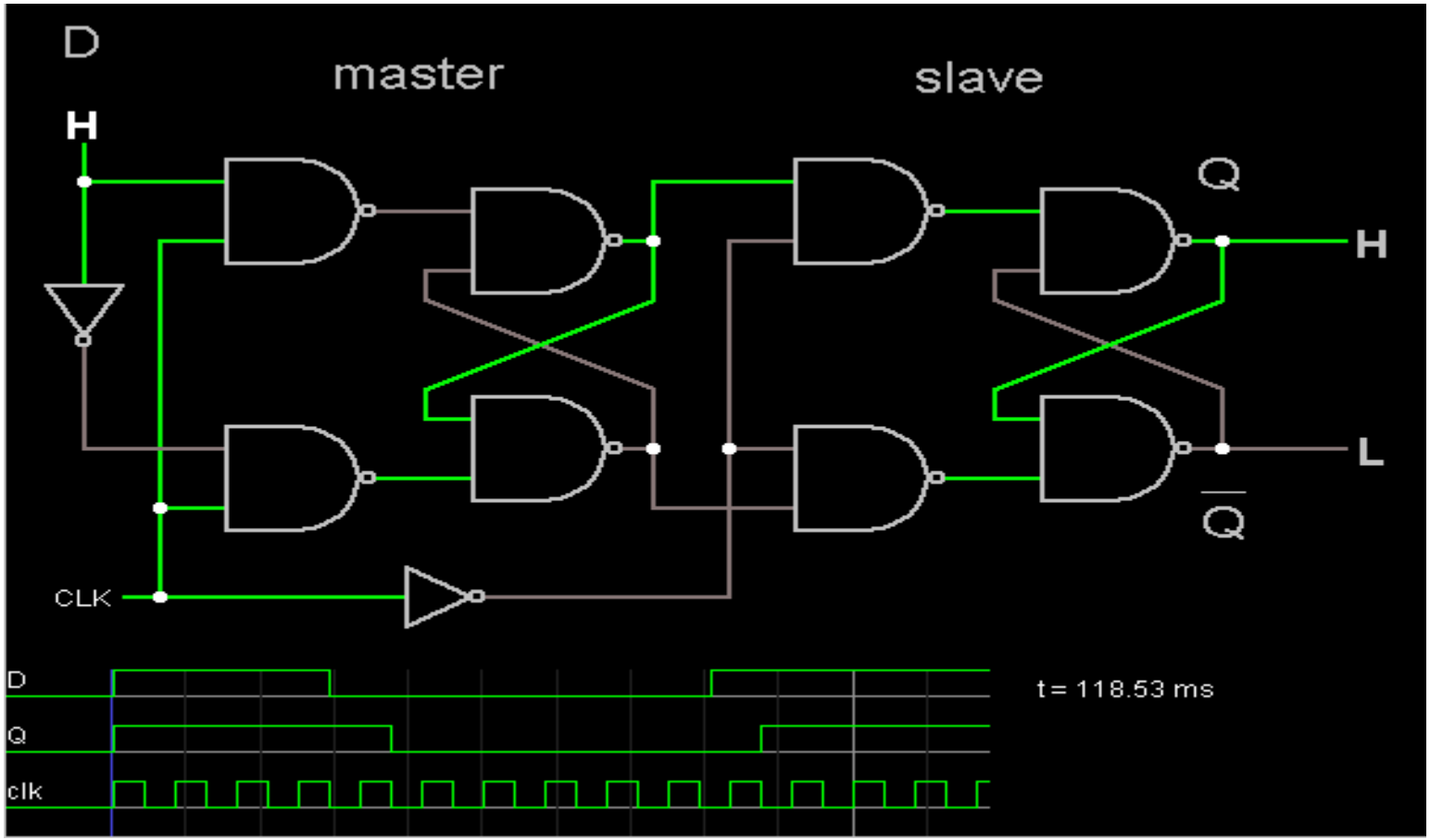
Master-Slave D Flip-Flop



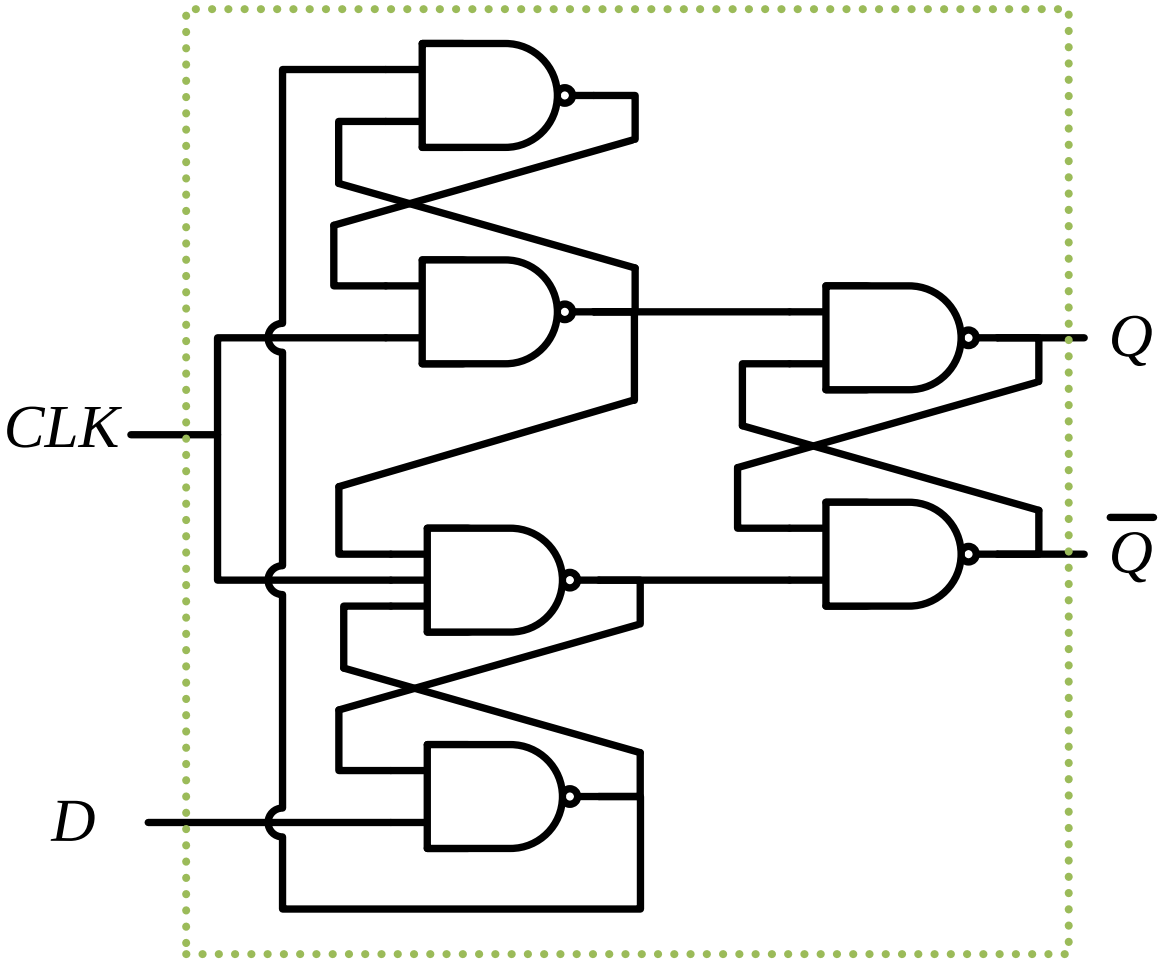
Master-Slave D Flip-Flop

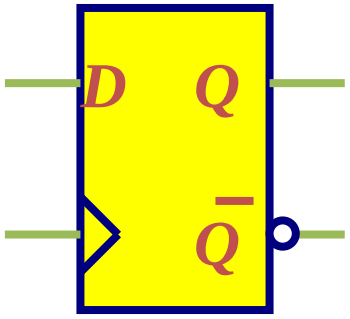


Master-Slave D Flip-Flop

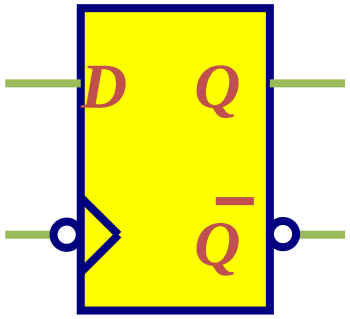


Edge-Triggered D Flip-Flop





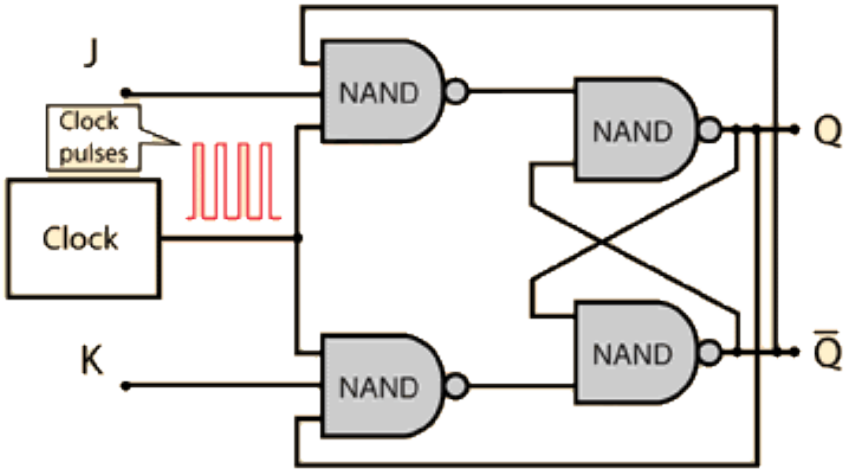
Positive Edge



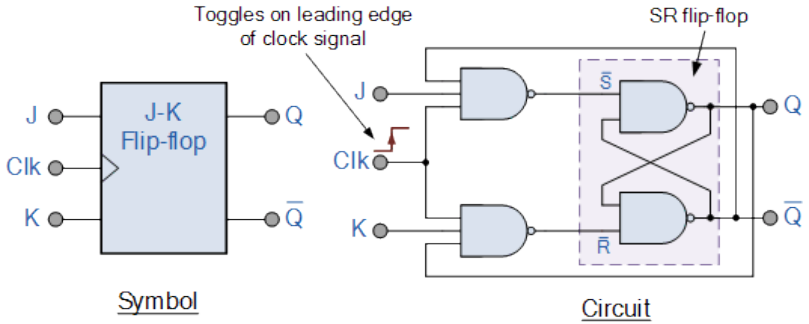
Negative Edge

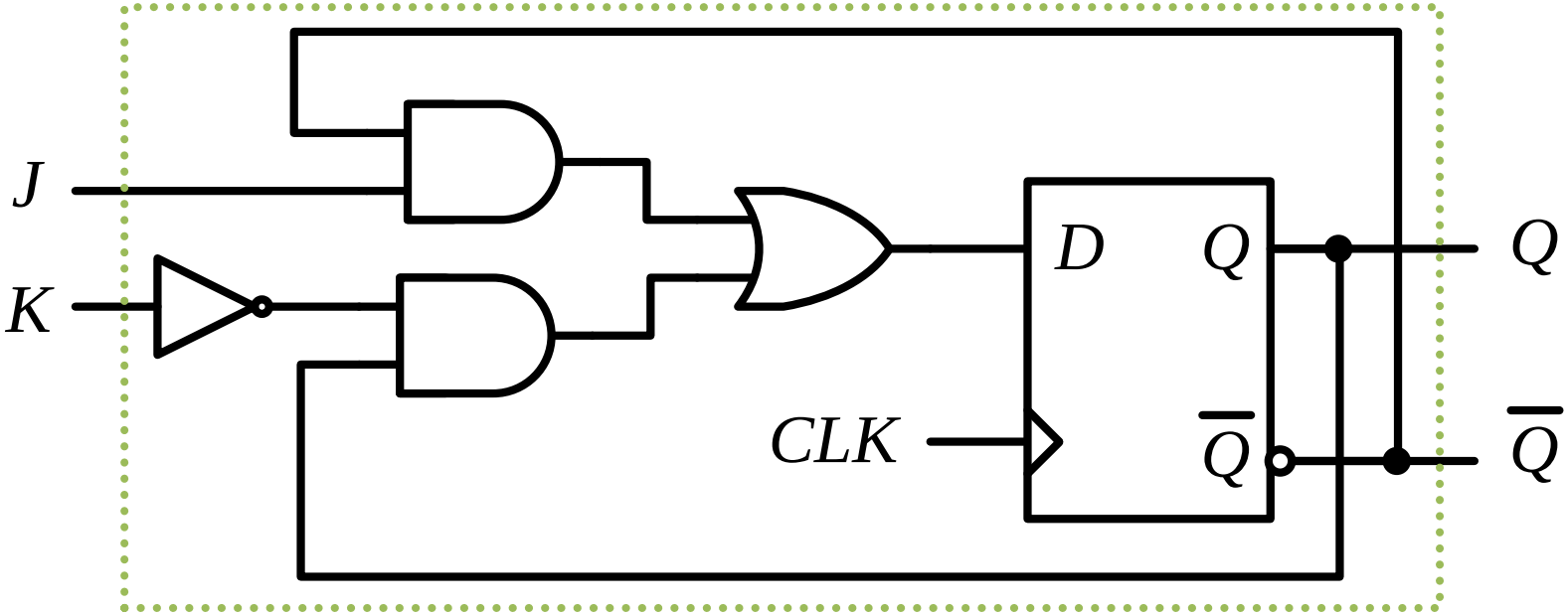
JK Flip-Flop

- The JK flip flop is basically a gated **SR flip-flop with the addition of a clock input circuitry** that prevents the illegal or invalid output condition that can occur when both inputs S and R are equal to logic level “1”.

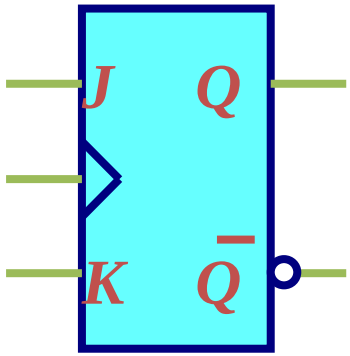


JK Flip-Flop



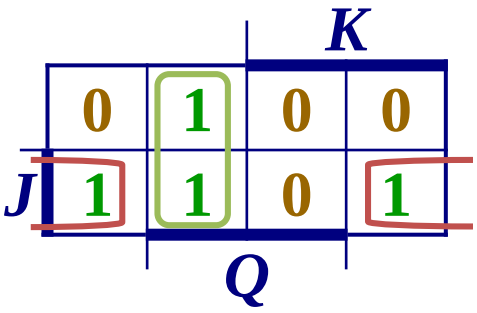


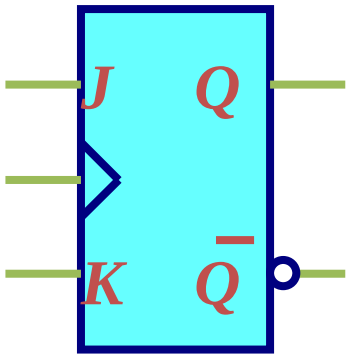
$$D = JQ' + K'Q$$



JK Flip-Flop Characteristic Equations

- Analysis / Derivation





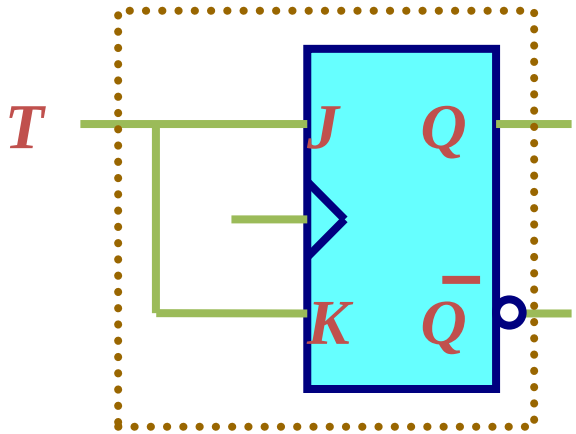
J	K	Q(t)	Q(t+1)	
0	0	0	0	No change
0	0	1	1	
0	1	0	0	Reset
0	1	1	0	
1	0	0	1	Set
1	0	1	1	
1	1	0	1	Toggle
1	1	1	0	

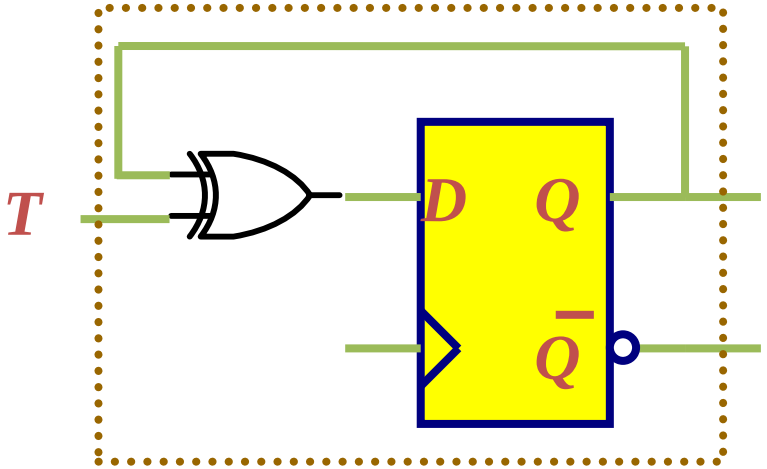
$Q(t+1) = JQ' + K'Q$

T Flip-Flop

- A T flip flop is like JK flip-flop.
- These are basically a single input version of JK flip flop.
- This **modified form of JK flip-flop is obtained by connecting both inputs J and K together.**
- This flip-flop has only one input along with the clock input.

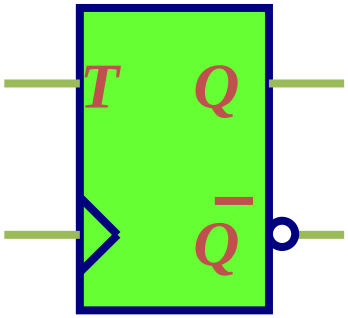
T Flip-Flop



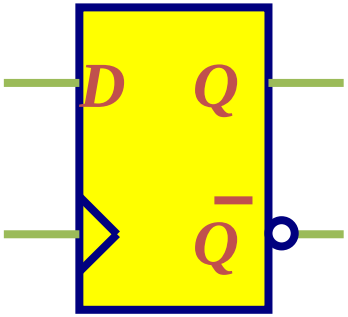


$D = JQ' + K'Q$

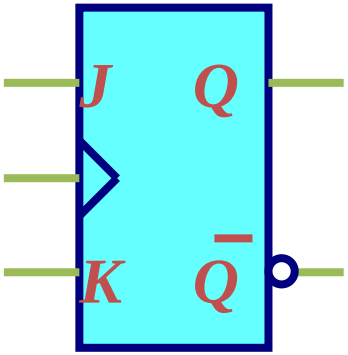
$D = TQ' + T'Q = T \oplus Q$



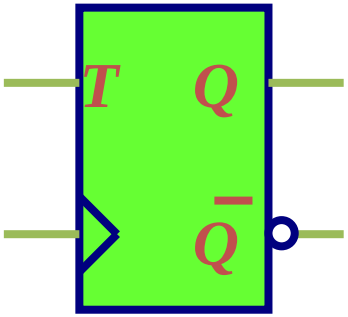
Flip-Flop Characteristic Tables



D	Q(t+1)	
0	0	Reset
1	1	Set

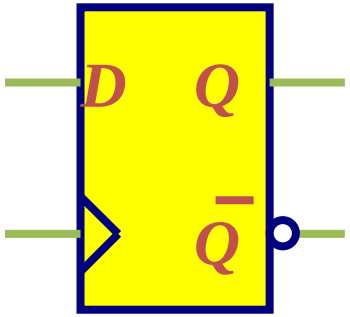


J	K	Q(t+1)	
0	0	Q(t)	No change
0	1	0	Reset
1	0	1	Set
1	1	Q'(t)	Toggle



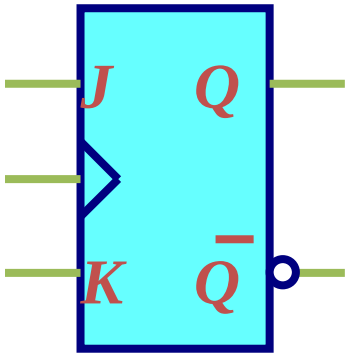
T	Q(t+1)	
0	Q(t)	No change
1	Q'(t)	Toggle

Flip-Flop Characteristic Equations



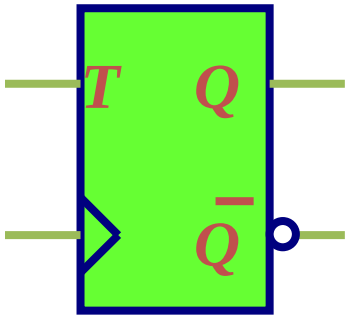
D	Q(t+1)
0	0
1	1

$Q(t+1) = D$



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

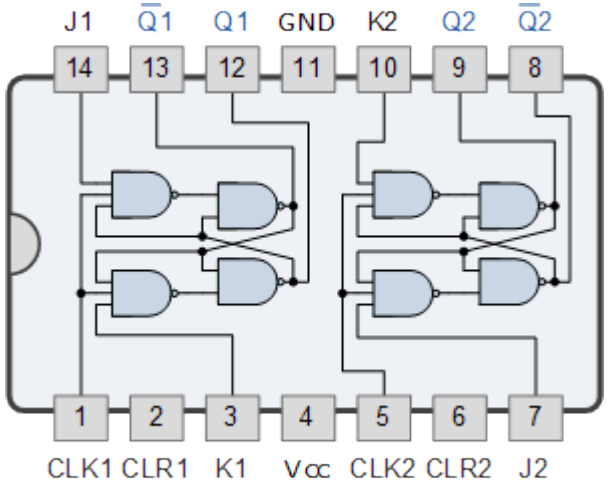
$Q(t+1) = JQ' + K'Q$

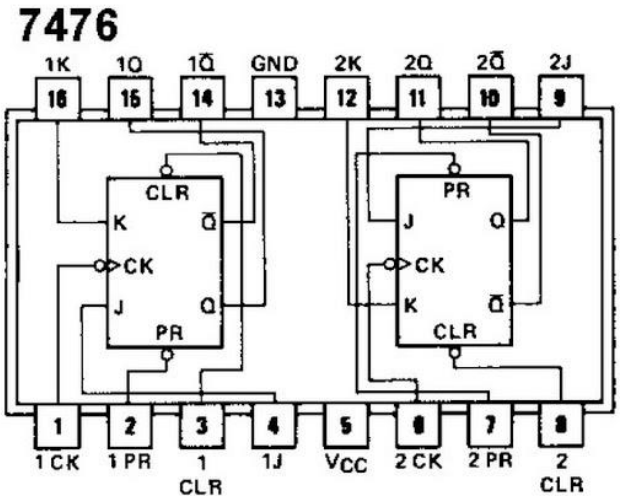


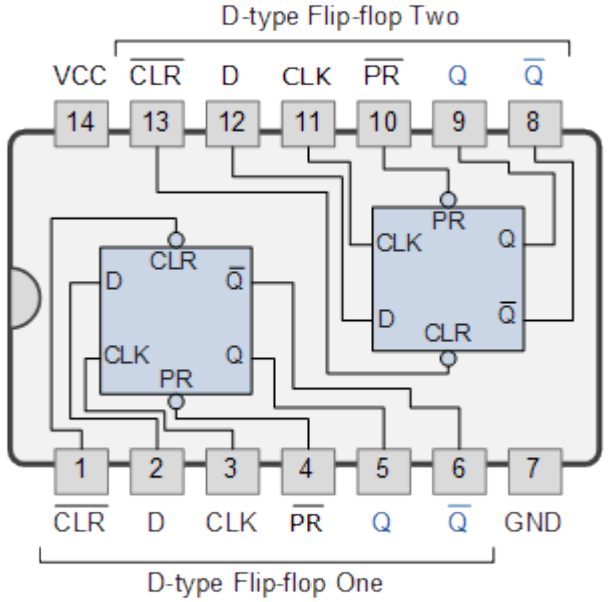
T	Q(t+1)
0	Q(t)
1	Q'(t)

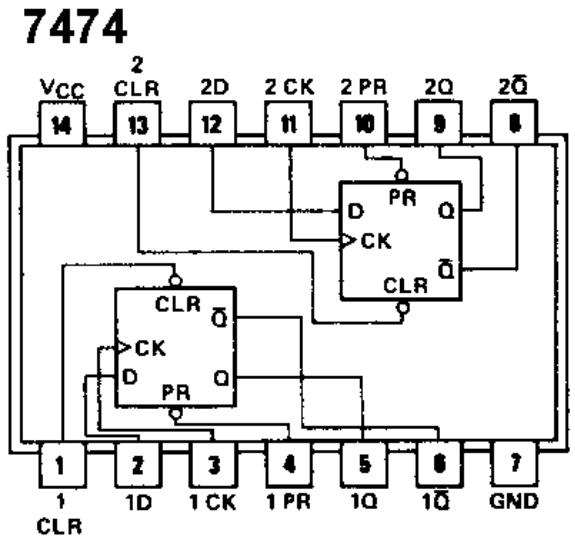
$Q(t+1) = T \oplus Q$

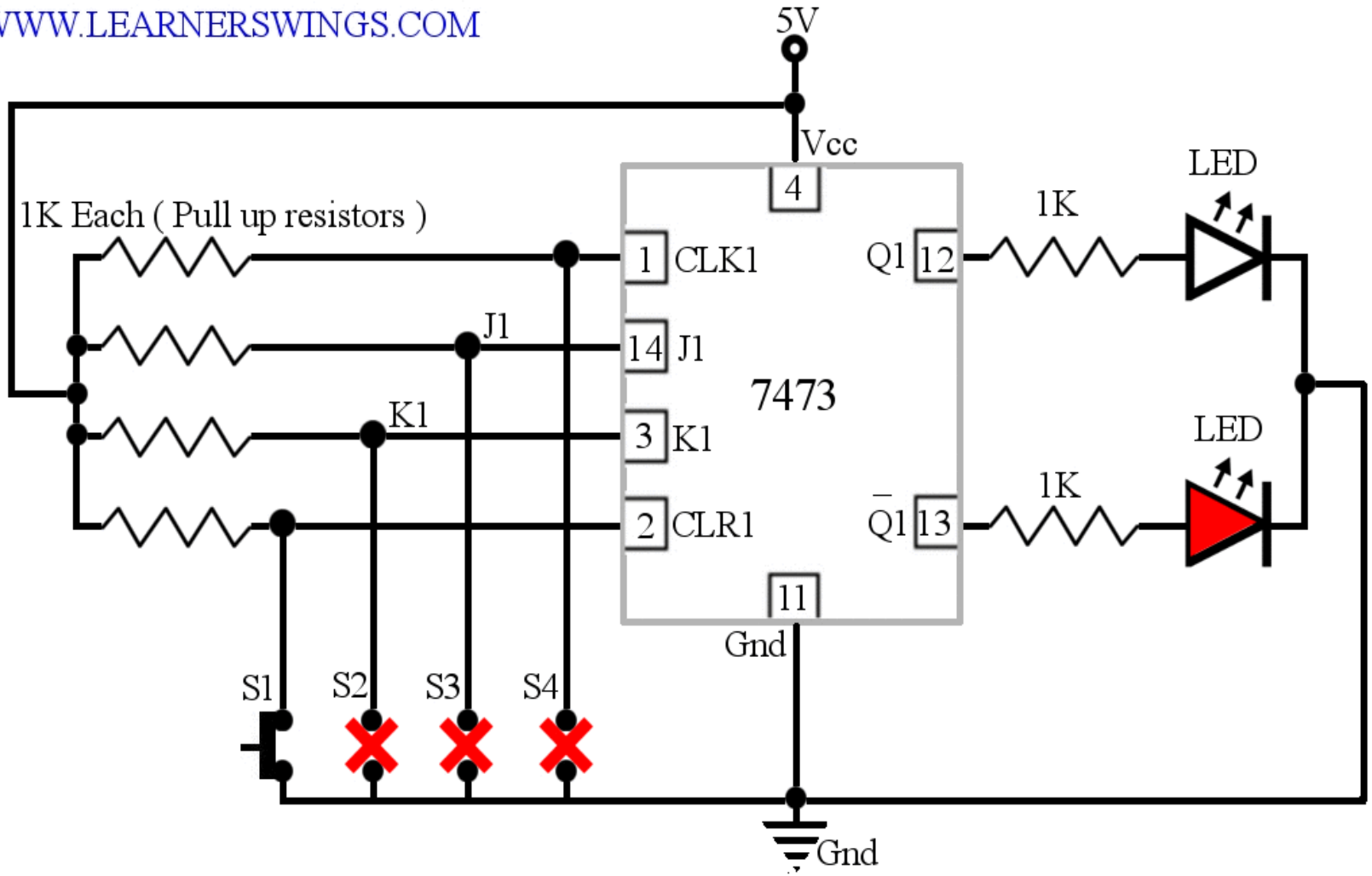
Hardware ICs



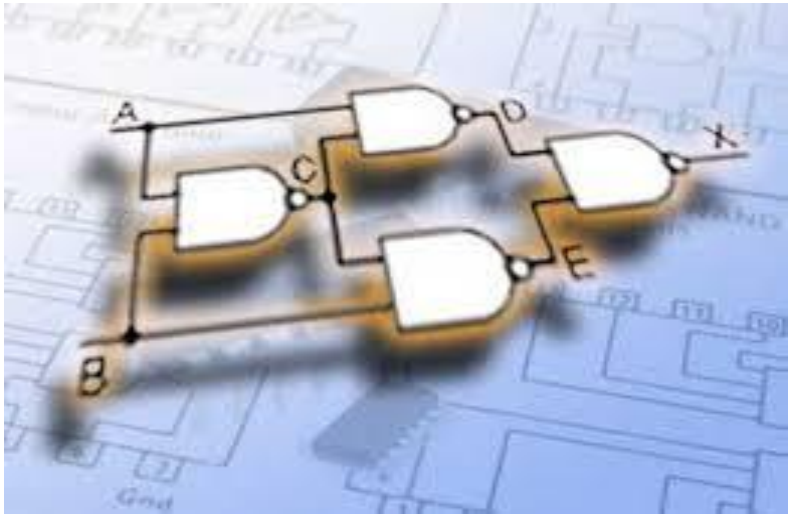


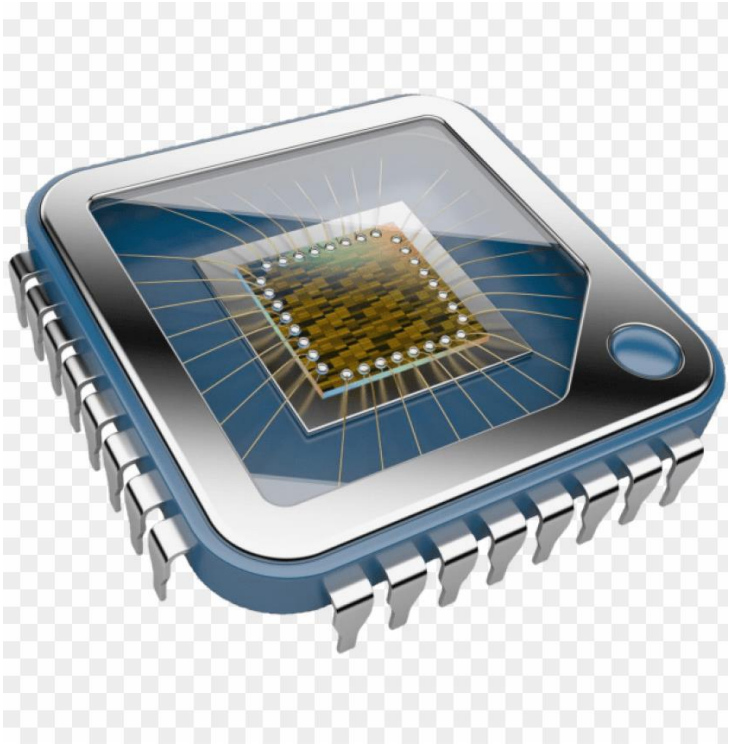






Thank You





Dr. Hatem Yousry
E-mail: Hyousry@nctu.edu.eg