# Digital Circuits Design

Communication Engineering Department

**Lecturer 1**

## Karnaugh map

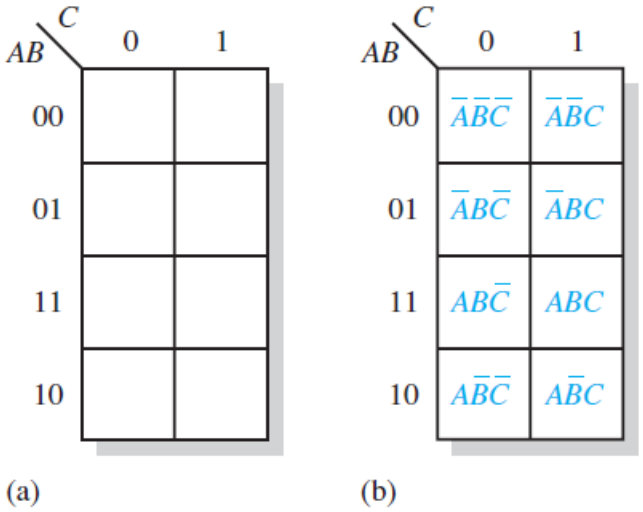
- The Karnaugh Map
  1- The 3-Variable Karnaugh Map
  2- The 4-Variable Karnaugh Map
  3- Five variables K-Map

## The Karnaugh Map

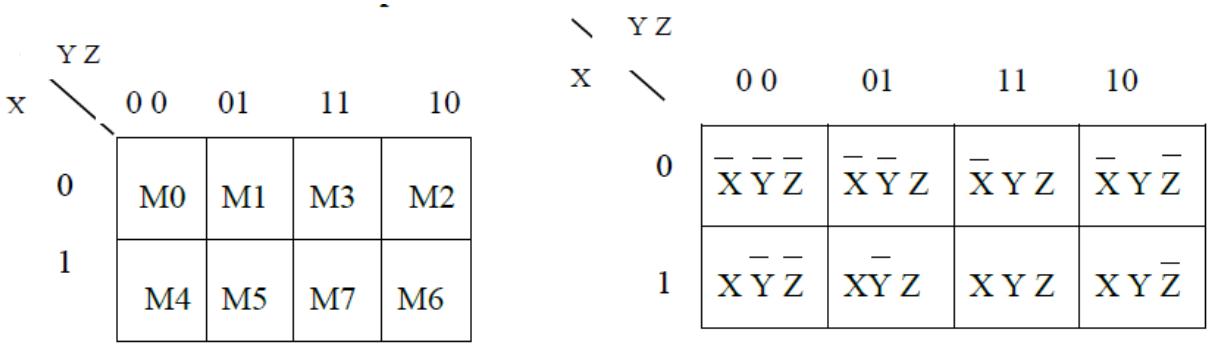
A Karnaugh map provides a systematic method for simplifying Boolean expressions and, if properly used, will produce the simplest SOP or POS expression possible, known as the minimum expression. As you have seen, the effectiveness of algebraic simplification depends on your familiarity with all the laws, rules, and theorems of Boolean algebra and on your ability to apply them. The Karnaugh map, on the other hand, provides a "cookbook" method for simplification.

### 1- The 3-Variable Karnaugh Map

The 3-variable Karnaugh map is an array of eight cells, as shown in Figure (a). In this case, A, B, and C are used for the variables although other letters could be used. Binary values of A and B are along the left side (notice the sequence) and the values of C are across the top. The value of a given cell is the binary values of A and B at the left in the same row combined with the value of C at the top in the same column. For example, the cell in the upper left corner has a binary value of 000 and the cell in the lower right corner has a binary value of 101. Figure (b) shows the standard product terms that are represented by each cell in the Karnaugh map.




(a) (b)



## 2- The 4-Variable Karnaugh Map

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | | | | |
| 01 | | | | |
| 11 | | | | |
| 10 | | | | |

(a)

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | $\bar{A}\bar{B}\bar{C}\bar{D}$ | $\bar{A}\bar{B}\bar{C}D$ | $\bar{A}\bar{B}CD$ | $\bar{A}\bar{B}C\bar{D}$ |
| 01 | $\bar{A}B\bar{C}\bar{D}$ | $\bar{A}B\bar{C}D$ | $\bar{A}BCD$ | $\bar{A}BC\bar{D}$ |
| 11 | $AB\bar{C}\bar{D}$ | $AB\bar{C}D$ | $ABCD$ | $ABC\bar{D}$ |
| 10 | $A\bar{B}\bar{C}\bar{D}$ | $A\bar{B}\bar{C}D$ | $A\bar{B}CD$ | $A\bar{B}C\bar{D}$ |

(b)

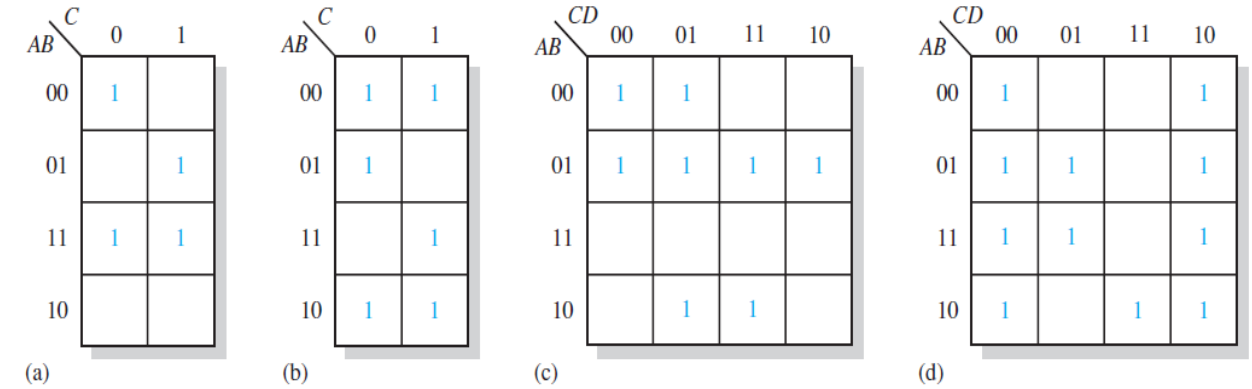
| XY \ ZW | 0 0 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 1 | 3 | 2 |
| 01 | 4 | 5 | 7 | 6 |
| 11 | 12 | 13 | 15 | 14 |
| 10 | 8 | 9 | 11 | 10 |

## 3- Five variables K-Map

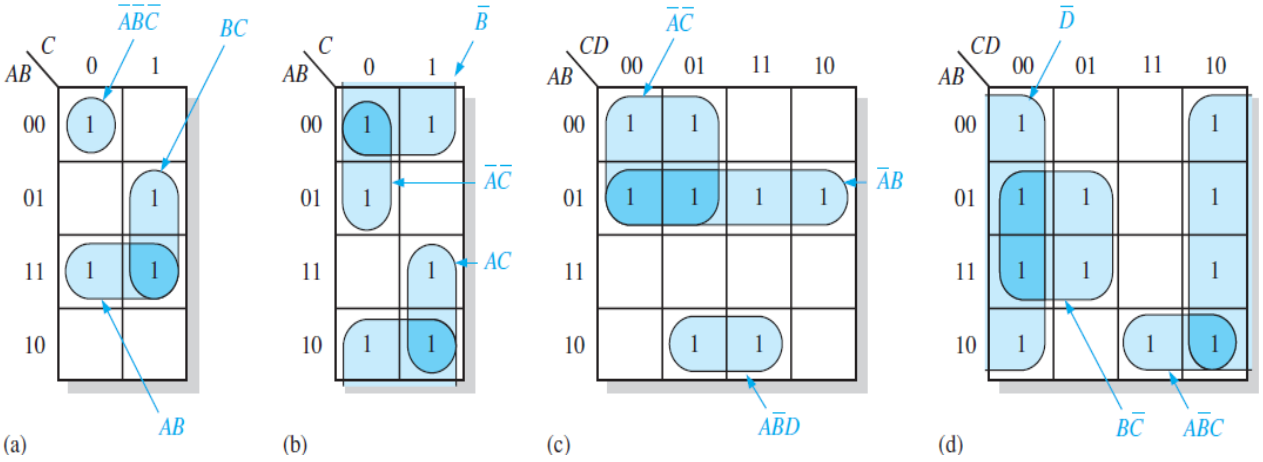
| AB \ CDE | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | 0 | 1 | 3 | 2 | 6 | 7 | 5 | 4 |
| 01 | 8 | 9 | 11 | 10 | 14 | 15 | 13 | 12 |
| 11 | 24 | 25 | 27 | 26 | 30 | 31 | 29 | 28 |
| 10 | 16 | 17 | 19 | 18 | 22 | 23 | 21 | 20 |

## Example 1:

Group the 1s in each of the Karnaugh maps in Figure




Sol:




**(a)** $AB + BC + \bar{A}\bar{B}\bar{C}$

**(b)** $\bar{B} + \bar{A}\bar{C} + AC$

**(c)** $\bar{A}B + \bar{A}\bar{C} + A\bar{B}D$

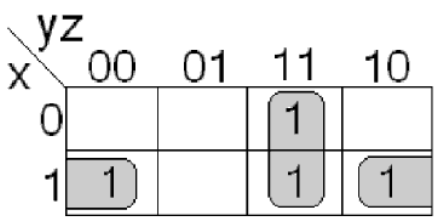
**(d)** $\bar{D} + A\bar{B}C + B\bar{C}$

## Example 2:

Simplify the following Boolean function

F (x, y, z) = ∑ (3, 4, 6, 7)

Sol:



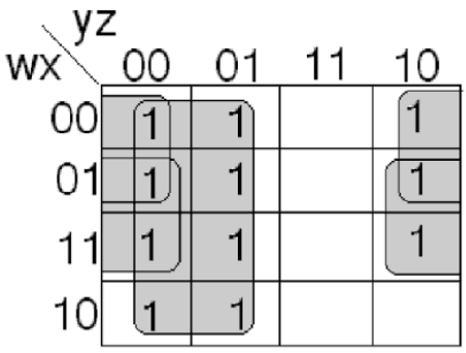

F (x, y, z) = $x\bar{z} + yz$

## Example 3:

Simplify the following Boolean function

F (w, x, y, z) = ∑ (0, 1, 2, 4, 5, 6, 8, 9, 12, 13, 14)

Sol:




$$F(w, x, y, z) = \bar{y} + \bar{w}\bar{z} + x\bar{z}$$

## Example 4:

Simplify the following Boolean function

F (A, B, C, D, E) = Σ (0, 2, 4, 6, 9, 11, 13, 15, 17, 21, 25, 27, 29, 31)

Sol:

| AB \ CDE | 000 | 001 | 011 | 010 | 110 | 111 | 101 | 100 |
|---|---|---|---|---|---|---|---|---|
| 00 | 1 | | | 1 | 1 | | | 1 |
| 01 | | 1 | 1 | | | 1 | 1 | |
| 11 | | 1 | 1 | | | 1 | 1 | |
| 10 | | 1 | | | | | 1 | |

$$F(A,B,C,D) = \bar{A}\,\bar{B}\,\bar{E} + B\,E + A\,\bar{D}\,E$$

### *Don't- Care Condition*

Sometimes a function table or map contains entries for which it is known:

- The input values for the minterm will never occur, or
- The output value for the minterm is not used

In these cases, the output value need not be defined, Instead, the output value is defined as a "don't care" these values are:

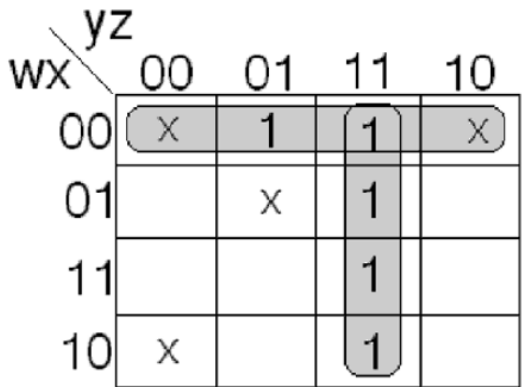
1 - Placing "don't cares" ( an "x" entry) in the function table or map,
2 – These values used in simplification with F and $\bar{F}$.
3 – These values may be not used in simplification.

**Example 5:**

Simplify the following Boolean function

F (w, x, y, z) = Σ (1, 3, 7, 11, 15)
Which has Don't Care conditions
d (w, x, y, z) = Σ (0, 2, 5, 8)

Sol:




F (w, x, y, z) = $\bar{w}\bar{x} + yz$

NOTE

Determine the minimum product term for each group.

**(a)** For a 3-variable map:

**(1)** A 1-cell group yields a 3-variable product term
**(2)** A 2-cell group yields a 2-variable product term
**(3)** A 4-cell group yields a 1-variable term
**(4)** An 8-cell group yields a value of 1 for the expression

**(b)** For a 4-variable map:

**(1)** A 1-cell group yields a 4-variable product term
**(2)** A 2-cell group yields a 3-variable product term
**(3)** A 4-cell group yields a 2-variable product term
**(4)** An 8-cell group yields a 1-variable term
**(5)** A 16-cell group yields a value of 1 for the expression

# Digital Circuits Design

Communication Engineering Department


**Lecturer 2**

## Karnaugh map 2

- **Karnaugh Map Simplification of SOP Expressions**
- **Karnaugh Map Simplification of POS Expressions:**

## Karnaugh Map Simplification of SOP Expressions:

The process that results in an expression containing the fewest possible terms with the fewest possible variables is called **minimization**. After an SOP expression has been mapped,
a minimum SOP expression is obtained by grouping the 1s and determining the minimum SOP expression from the map.

### Grouping the 1 s:

1. A group must contain either 1, 2, 4, 8, or 16 cells, which are all powers of two. In the case of a 3-variable map, $2^3 = 8$ cells are the maximum group.
2. Each cell in a group must be adjacent to one or more cells in that same group, but all cells in the group do not have to be adjacent to each other.
3. Always include the largest possible number of 1s in a group in accordance with rule 1.
4. Each 1 on the map must be included in at least one group. The 1s already in a group can be included in another group as long as the overlapping groups include noncommon 1s.

**Example 1:**

Map the following SOP expression on Karnaugh map : $\bar{A} + A\bar{B} + AB\bar{C}$
Sol:

| $\bar{A}$ | $+ A\bar{B}$ | $+ AB\bar{C}$ |
|---|---|---|
| 000 | 100 | 110 |
| 001 | 101 | |
| 010 | | |
| 011 | | |

| $AB$ \ $C$ | 0 | 1 |
|---|---|---|
| 00 | 1 | 1 |
| 01 | 1 | 1 |
| 11 | 1 | |
| 10 | 1 | 1 |

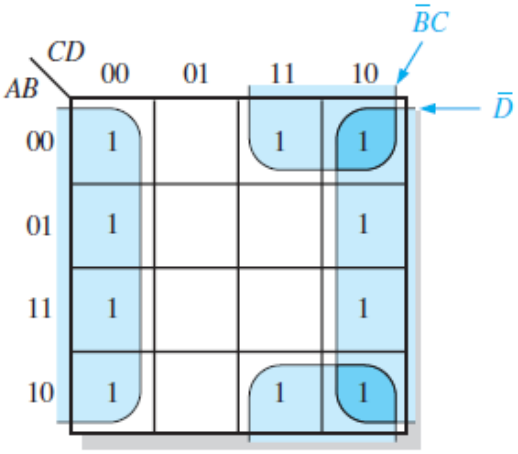
**Example 2:**

Use a Karnaugh map to minimize the following SOP expression:

$$\overline{B}\,\overline{C}\,\overline{D} + \overline{A}B\overline{C}\,\overline{D} + AB\overline{C}\,\overline{D} + \overline{A}\,\overline{B}CD + A\overline{B}CD + \overline{A}\,\overline{B}C\overline{D} + \overline{A}BC\overline{D} + ABC\overline{D} + A\overline{B}C\overline{D}$$

**Solution**

The first term $\overline{B}\,\overline{C}\,\overline{D}$ must be expanded into $A\overline{B}\,\overline{C}\,\overline{D}$ and $\overline{A}\,\overline{B}\,\overline{C}\,\overline{D}$ to get the standard SOP expression, which is then mapped; the cells are grouped as shown in Figure



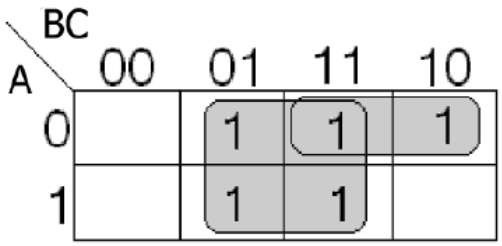

$$\overline{D} + \overline{B}C$$

**Example 3:**

Given the Boolean function:

F (A, B, C) = $\bar{A}C + \bar{A}B + A\bar{B}C + BC$

1- Express it in Sum Of Minterms
2- Find the minimal Sum Of products expression




1- F (A, B, C) = $\sum$ (1, 2, 3, 5, 7)
2- F (A, B, C) = C+ $\bar{A}B$

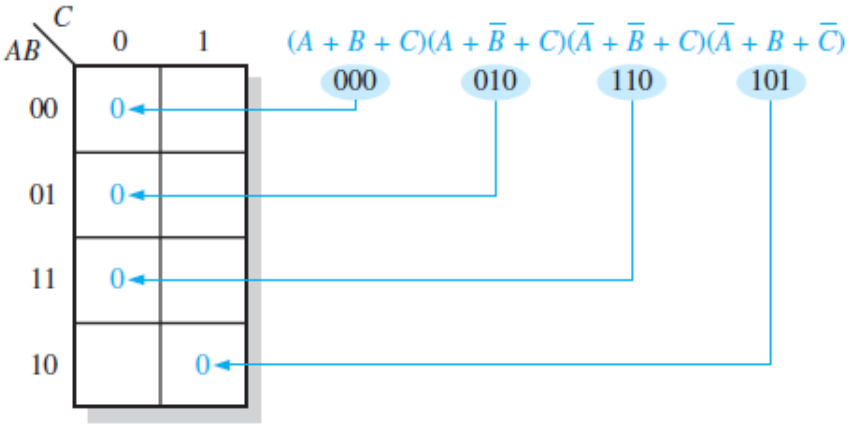
## Karnaugh Map Simplification of POS Expressions:

For a POS expression in standard form, a 0 is placed on the Karnaugh map for each sum term in the expression. Each 0 is placed in a cell corresponding to the value of a sum term.

For example, for the sum term $A + \bar{B} + C$, a 0 goes in the 010 cell on a 3-variable map. When a POS expression is completely mapped, there will be a number of 0s on the Karnaugh map equal to the number of sum terms in the standard POS expression. The cells that do not have a 0 are the cells for which the expression is 1. Usually, when working with POS expressions, the 1s are left off. The following steps and the illustration in Figure show the mapping process.

**Step 1:** Determine the binary value of each sum term in the standard POS expression. This is the binary value that makes the term equal to 0.

**Step 2:** As each sum term is evaluated, place a 0 on the Karnaugh map in the corresponding cell.




**Example 4:**

Obtain the simplified expression in Product of Sums

F (A, B, C, D) = Π (0, 1, 2, 3, 4, 10, 11)

Sol:

| AB \ CD | 0 0 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | | | |
| 11 | | | | |
| 10 | | | 0 | 0 |

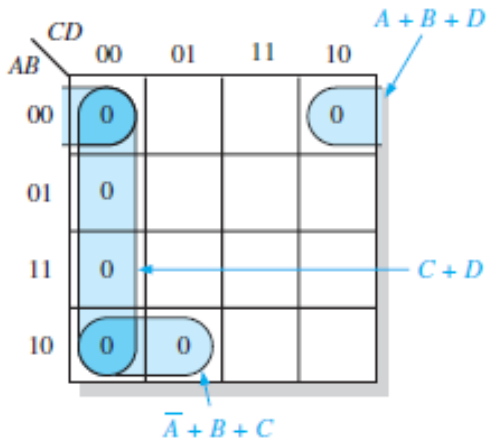
$$F = (A + B)\,(A + C + D)\,(B + \bar{C})$$

## Example 5:

Use a Karnaugh map to minimize the following POS expression:

$$(B + C + D)(A + B + \overline{C} + D)(\overline{A} + B + C + \overline{D})(A + \overline{B} + C + D)(\overline{A} + \overline{B} + C + D)$$

Sol:




$$(C + D)(A + B + D)(\overline{A} + B + C)$$

## Example 6:

simplify the Boolean function F in 1 – Sum of Products 2 – Product of Sums

F (X, Y, Z, W) = Σ (1, 3, 7, 11, 15)  d (X, Y, Z, W) = Σ (0, 2, 5)

Sol:

1- Sum of Products

| x y \ z w | 0 0 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | X | 1 | 1 | X |
| 01 | | X | 1 | |
| 11 | | | 1 | |
| 10 | | | 1 | |

$$F\ (X,Y,Z,W) = Z\,W + \overline{X}\,\overline{Y}$$

2 – Product of Sums

| x y \ z w | 0 0 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | X | | | X |
| 01 | 0 | X | | 0 |
| 11 | 0 | 0 | | 0 |
| 10 | 0 | 0 | | 0 |

$$F\ (X,Y,Z,W) = W\ (\overline{X} + Z)$$

# Digital Circuits Design

Communication Engineering Department


**Lecturer 3**

## Design of combinational CIRCUITS

**Functions of combinational logic (Adder, Comparator)**

- **Half and Full Adders**
- **The Half-Adder**
- **The Full-Adder**
- **Four-Bit Parallel Adders**
- **comparator**

- A ***combinational circuit*** is one where the **output at any time depends only on the present combination of inputs at that point of time** with **total disregard to the past state of the inputs**.

✓ The **logic gate** is the most basic building block of combinational logic.

✓ Logical function performed by a combinational circuit is fully defined by a set **of Boolean expressions.**

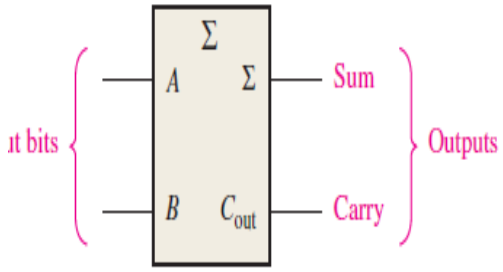
- ***Sequential logic circuits*** comprises both **logic gates** and **memory elements** such as flip-flops.

✓ Owing to the presence of memory elements, the output in a sequential circuit depends upon not only the present but also the past state of inputs.

## Half and Full Adders

### 1- The Half-Adder

$$0 + 0 = 0$$
$$0 + 1 = 1$$
$$1 + 0 = 1$$
$$1 + 1 = 10$$



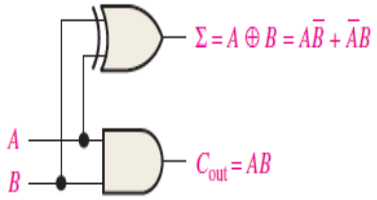

The operations are performed by a logic circuit called a **half-adder**.

**The half-adder accepts two binary digits on its inputs and produces two binary digits on its outputs—a sum bit and a carry bit.**

Half-adder truth table.

| $A$ | $B$ | $C_{out}$ | $\Sigma$ |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 0 |

$\Sigma$ = sum
$C_{out}$ = output carry




Half-adder logic diagram.

From the operation of the half-adder as stated in Table, expressions can be derived for the sum and the output carry as functions of the inputs. Notice that the output carry ($C_{out}$) is a 1 only when both $A$ and $B$ are 1s; therefore, $C$out can be expressed as the AND of the input variables.

$$C_{out} = AB$$

Now observe that the sum output (∑) is a 1 only if the input variables, $A$ and $B$, are not equal. The sum can therefore be expressed as the exclusive-OR of the input variables.

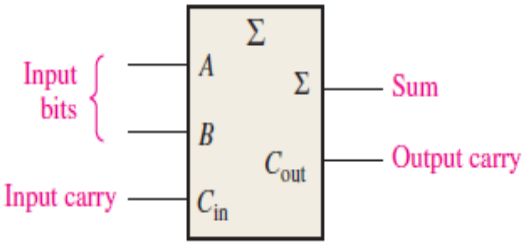
$$\Sigma = A \oplus B$$

the logic implementation required for the half-adder function can be developed. The output carry is produced with an AND gate with $A$ and $B$ on the inputs, and the sum output is generated with an exclusive-OR gate.

## 2- The Full-Adder

The second category of adder is the full-adder.

**The full-adder accepts two input bits and an input carry and generates a sum output and an output carry.**




The basic difference between a full-adder and a half-adder is that the full-adder accepts an input carry.

The full-adder must add the two input bits and the input carry. From the half-adder you know that the sum of the input bits *A* and *B* is the exclusive-OR of those two variables, $A \oplus B$. For the input carry ($C_{in}$) to be added to the input bits, it must be exclusive-Ored with $A \oplus B$, yielding the equation for the sum output

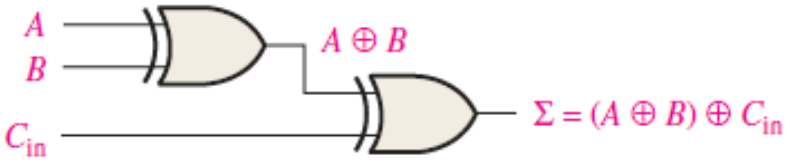
$$\Sigma = (A \oplus B) \oplus C_{in}$$

This means that to implement the full-adder sum function, two 2-input exclusive-OR gates can be used. The first must generate the term $A \oplus B$, and the second has as its inputs the output of the first XOR gate and the input carry.

Full-adder truth table.

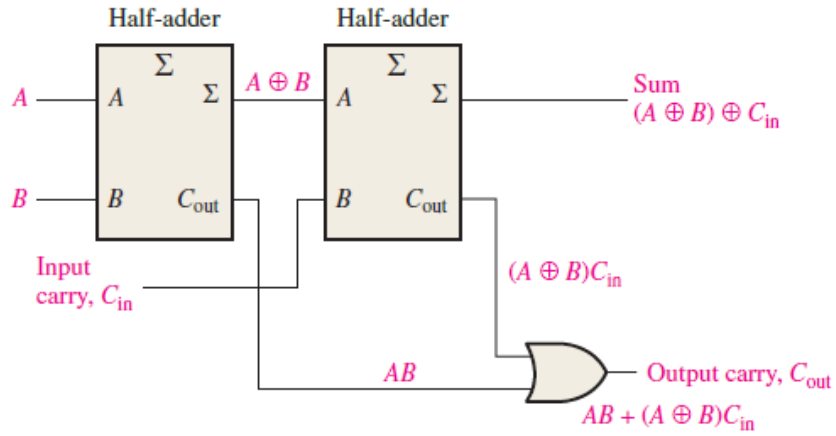
| $A$ | $B$ | $C_{in}$ | $C_{out}$ | $\Sigma$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 |

$C_{in}$ = input carry, sometimes designated as *CI*
$C_{out}$ = output carry, sometimes designated as *CO*
$\Sigma$ = sum

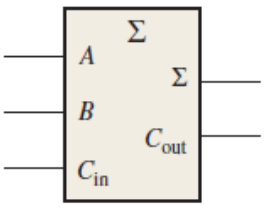



The output carry is a 1 when both inputs to the first XOR gate are 1s or when both inputs to the second XOR gate are 1s. The output carry of the full-adder is therefore produced by input A ANDed with input B and A⊕B ANDed with $C_{in}$. These two terms are ORed. This function is implemented and combined with the sum logic to form a complete full-adder circuit.

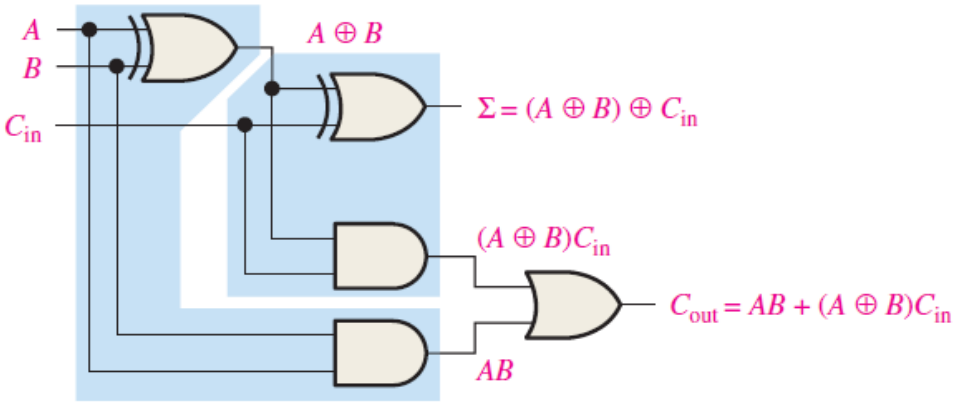
$$C_{out} = AB + (A \oplus B)C_{in}$$



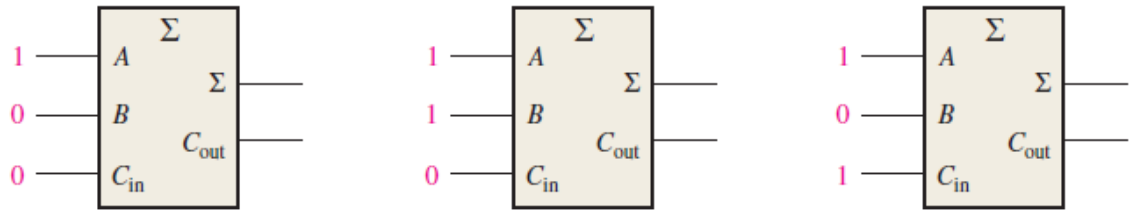

(a) Arrangement of two half-adders to form a full-adder




(b) Full-adder logic symbol




**Example: determine the outputs for the inputs shown.**




(a) (b) (c)

## Solution

(a) The input bits are $A = 1, B = 0$, and $C_{in} = 0$.

$$1 + 0 + 0 = 1 \text{ with no carry}$$

Therefore, $\Sigma = \mathbf{1}$ and $C_{out} = \mathbf{0}$.

(b) The input bits are $A = 1, B = 1$, and $C_{in} = 0$.

$$1 + 1 + 0 = 0 \text{ with a carry of 1}$$

Therefore, $\Sigma = \mathbf{0}$ and $C_{out} = \mathbf{1}$.

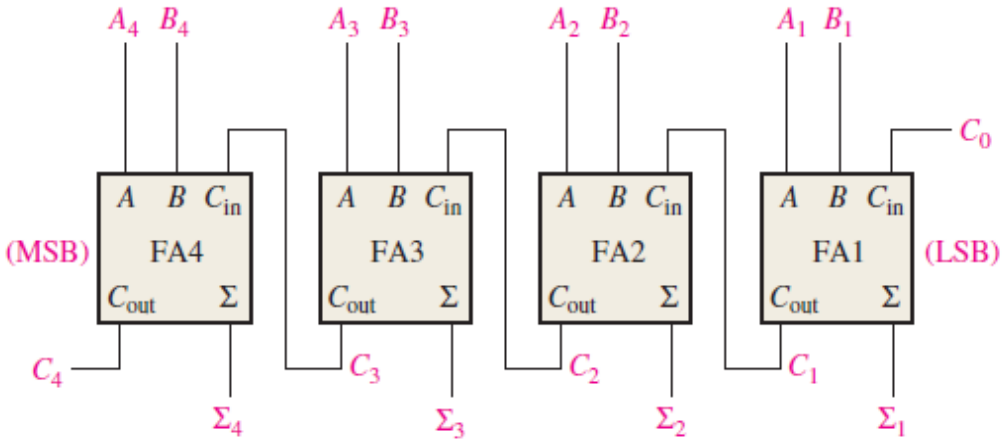
(c) The input bits are $A = 1, B = 0$, and $C_{in} = 1$.

$$1 + 0 + 1 = 0 \text{ with a carry of 1}$$

Therefore, $\Sigma = \mathbf{0}$ and $C_{out} = \mathbf{1}$.

## 3- Four-Bit Parallel Adders

A group of four bits is called a **nibble**. A basic 4-bit parallel adder is implemented with four full-adder stages as shown in Figure. Again, the LSBs (A1 and B1) in each number being added go into the right-most full-adder; the higher-order bits are applied as shown to the successively higher-order adders, with the MSBs (A4 and B4) in each number being applied to the left-most full-adder. The carry output of each adder is connected to the carry input of the next higher-order adder as indicated. These are called internal carries.




**Example:** Use the 4-bit parallel adder to find the sum and output carry for the addition of the following two 4-bit numbers if the input carry (Cn-1) is 0:

$A_4A_3A_2A_1 = 1100$ and $B_4B_3B_2B_1 = 1100$

### Solution

For $n = 1$: $A_1 = 0$, $B_1 = 0$, and $C_{n-1} = 0$. From the 1st row of the table,

$$\Sigma_1 = \mathbf{0} \quad \text{and} \quad C_1 = 0$$

For $n = 2$: $A_2 = 0$, $B_2 = 0$, and $C_{n-1} = 0$. From the 1st row of the table,

$$\Sigma_2 = \mathbf{0} \quad \text{and} \quad C_2 = 0$$

For $n = 3$: $A_3 = 1$, $B_3 = 1$, and $C_{n-1} = 0$. From the 4th row of the table,

$$\Sigma_3 = \mathbf{0} \quad \text{and} \quad C_3 = 1$$

For $n = 4$: $A_4 = 1$, $B_4 = 1$, and $C_{n-1} = 1$. From the last row of the table,

$$\Sigma_4 = \mathbf{1} \quad \text{and} \quad C_4 = \mathbf{1}$$

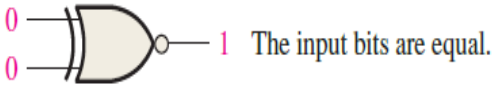
$C_4$ becomes the output carry; the sum of 1100 and 1100 is 11000.

## comparator

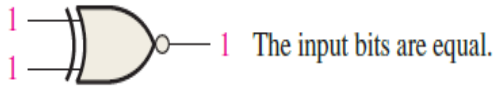
The basic function of a comparator is to compare the magnitudes of two binary quantities to determine the relationship of those quantities. In its simplest form, a comparator circuit determines whether two numbers are equal.

✓ Equality

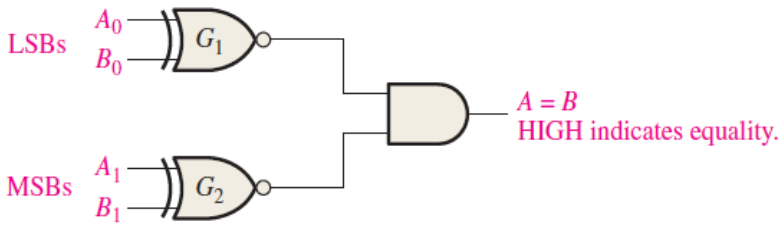
As you learned, the exclusive-NOR gate can be used as a basic comparator because its output is a 0 if the two input bits are not equal and a 1 if the input bits are equal.






In order to compare binary numbers containing two bits each, an additional exclusive-NOR gate is necessary. The two least significant bits (LSBs) of the two numbers are compared by gate G1, and the two most significant bits (MSBs) are compared by gate G2, as shown in Figure. If the two numbers are equal, their corresponding bits are the same, and the output of each exclusive-NOR gate is a 1. If the corresponding sets of bits are not equal, a 0 occurs on that exclusive-NOR gate output.

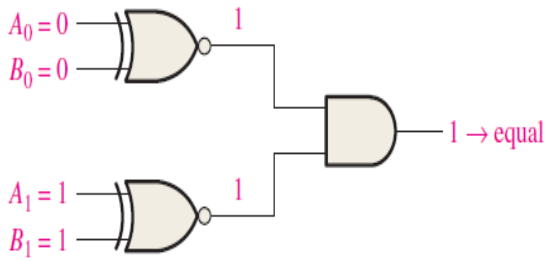



General format: Binary number $A \rightarrow A_1A_0$
Binary number $B \rightarrow B_1B_0$

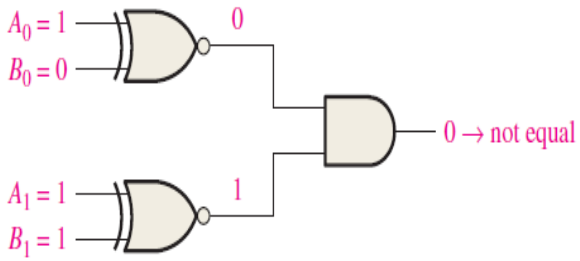
**Example:**

Apply each of the following sets of binary numbers to the comparator inputs in Figure and determine the output by following the logic levels through the circuit.

(a) 10 and 10 (b) 11 and 10




(a)

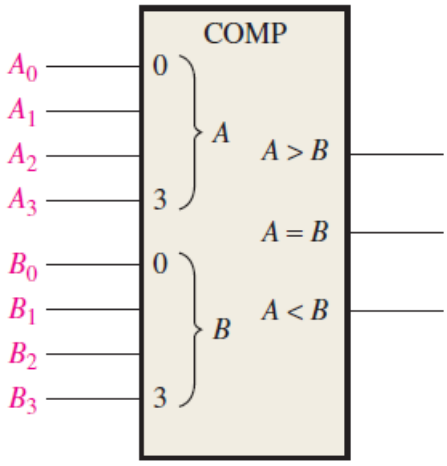



(b)

✓ Inequality

In addition to the equality output, fixed-function comparators can provide additional outputs that indicate which of the two binary numbers being compared is the larger. That is, there is an output that indicates when number $A$ is greater than number $B$ ($A > B$) and an output that indicates when number $A$ is less than number $B$ ($A < B$), as shown in the logic symbol for a 4-bit comparator. To determine an inequality of binary numbers $A$ and $B$, you first examine the highestorder bit in each number. The following conditions are possible:

**1.** If $A_3 = 1$ and $B_3 = 0$, number $A$ is greater than number $B$.
**2.** If $A_3 = 0$ and $B_3 = 1$, number $A$ is less than number $B$.
**3.** If $A_3 = B_3$, then you must examine the next lower bit position for an inequality.




Example:

Determine the $A = B$, $A > B$, and $A < B$ outputs for the input numbers shown on the comparator in Figure 6–22.

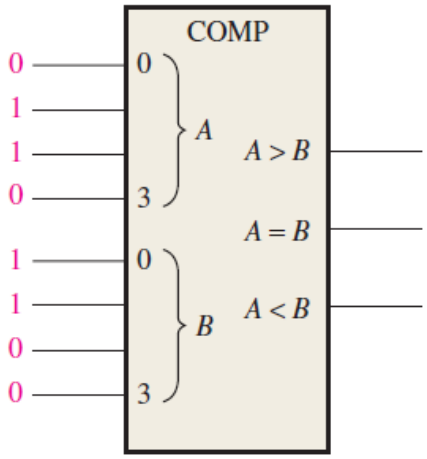



**FIGURE 6–22**

**Solution**

The number on the $A$ inputs is 0110 and the number on the $B$ inputs is 0011. The **$A > B$ output is HIGH and the other outputs are LOW.**

# Digital Circuits Design

Communication Engineering Department


**Lecturer 4**

## Design of Sequential Circuits

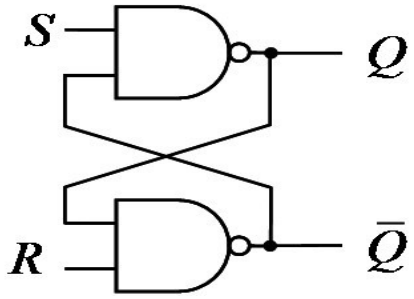
- R-S Flip-flop
- J-K flip-flop
- D flip-flop
- T flip-flop

# FLIP-FLOPS

Flip Flops are binary storage elements come under the category of **Sequential circuits**. A flip flop stores a binary digit either 0 or 1. In other words, it has two stable states namely HIGH and LOW. HIGH state stores 1 and LOW state stores 0.

## R-S Flip-flop:

The R-S flip-flop can be realized using NAND gates. The circuit diagram and the truth table of RS flip-flop are shown below.

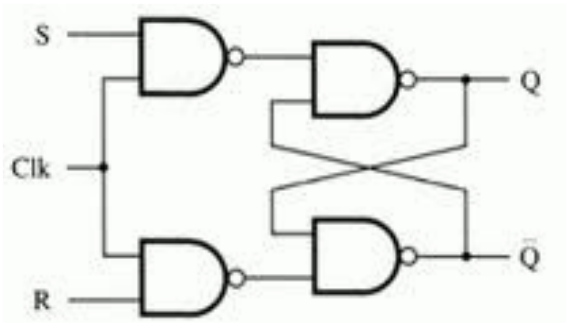



| R | S | Q | Comment |
|---|---|---|---|
| 0 | 0 | * | Race Condition |
| 0 | 1 | 1 | SET |
| 1 | 0 | 0 | RESET |
| 1 | 1 | Q' | Previous State |

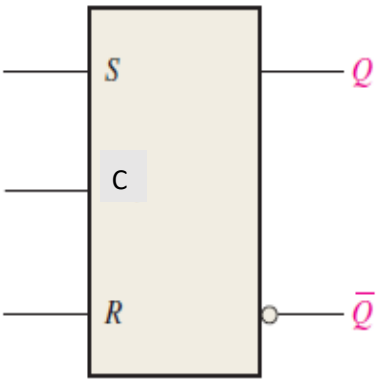
A 0 on any input to NAND gate will make its output high.
when R=0 and S=0, the output of the flip-flop becomes unpredictable. This state is sometimes referred to as a forbidden state or race condition.
When R=0 and S=1, the output Q of the flip-flop is set to 1. This state is called set state.
When R=1 and S=0, the output Q of the flip-flop is reset to 0. This state is called reset state. When R=1 and S=1, the output Q of the flip-flop will remain in its previous state. In other words, the output of the flip-flop will not change.

## Clocked R-S or S-R flip-flop:

The circuit diagram and the truth table of a S-R flip-flop are shown below.



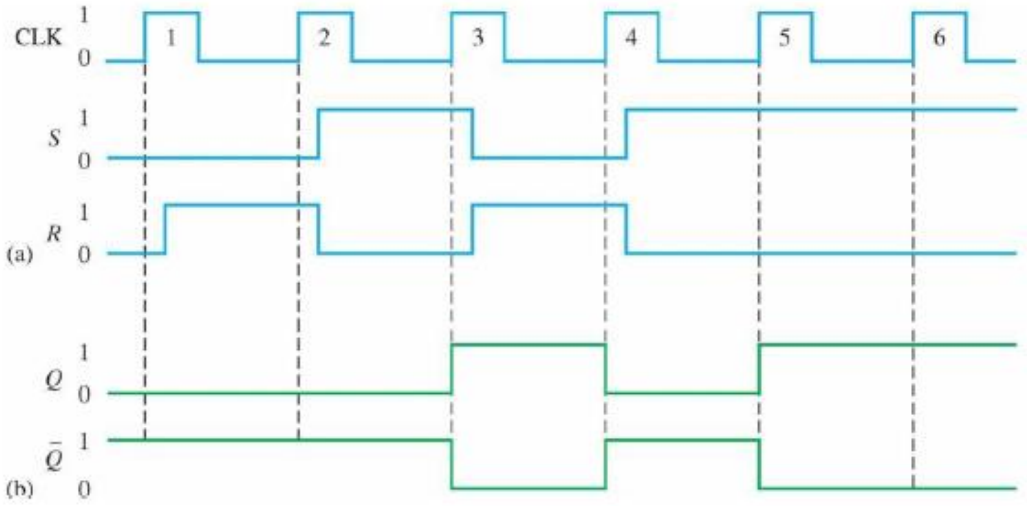

(a) Logic diagram




(b) Logic symbol

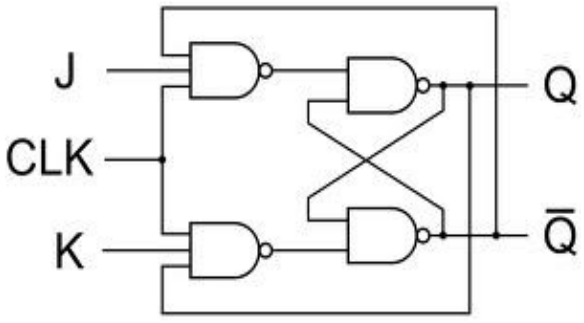
| Clock | R | S | Q | Comment |
|---|---|---|---|---|
| 0 | X | X | NC | No change |
| 1 | 0 | 0 | NC | No Change |
| 1 | 0 | 1 | 1 | SET |
| 1 | 1 | 0 | 0 | RESET |
| 1 | 1 | 1 | * | Impredictable(Racing) |

When the CLK input is 0, the output of the flip-flop will not change irrespective of the inputs at S and R. The flip-flop is disabled when the clock is 0.
When the CLK input is 1, the output of the flip-flop will change according to the S and R inputs. The flip-flop is enabled when the clock is 1. The response of the flip-flop for various S and R inputs, when the CLK is 1, is as follows.
When R=0 and S=0, the output of the flip-flop will not change.
When R=0 and S=1, the output Q of theflip-flop is set to 1.
When R=1 and S=0, the output Q of the flip-flop is reset to 0.
When R=1 and S=1, the output Q of the flip-flop becomes unpredictable.



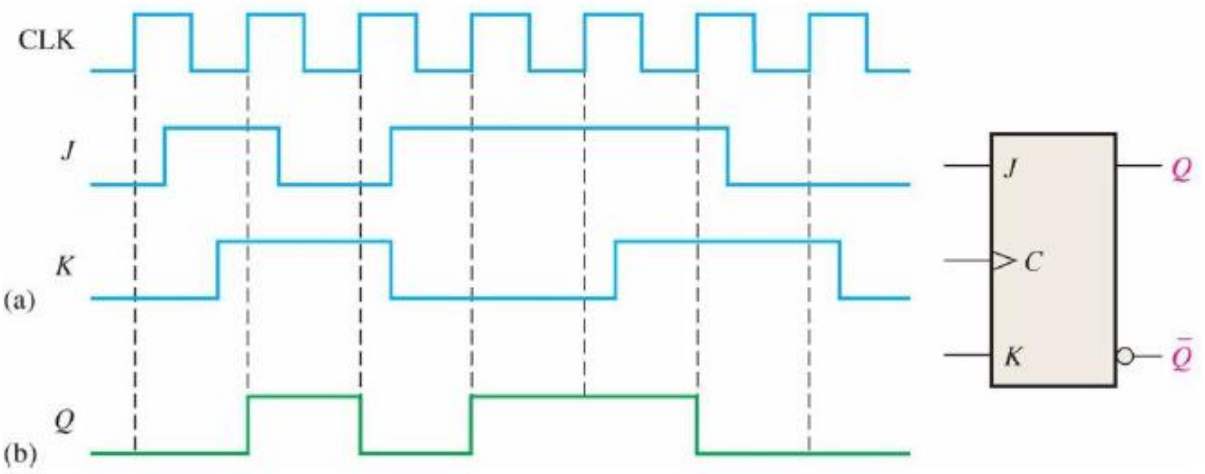

## J-K flip-flop:

The circuit diagram and the truth table of a J-K flip-flop are shown below.



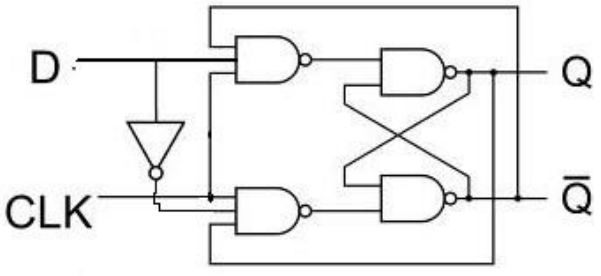

| CLK | J | K | Q | Comment |
|---|---|---|---|---|
| 0 | X | X | NC | No Change |
| ↑ | 0 | 0 | NC | No Change |
| ↑ | 0 | 1 | 0 | RESET |
| ↑ | 1 | 0 | 1 | SET |
| ↑ | 1 | 1 | * | TOGGLE |

When the CLK input is 0, the output of the flip-flop will not change irrespective of the inputs at J and K. The flip-flop is disabled when the clock is 0. When the CLK input is 1, the output of the flip-flop will change according to the J and K inputs. The flip-flop is enabled when the clock is 1. The response of the flip-flop for various J and K inputs, when the CLK is 1, is as follows.
When J=0 and K=0, the output of the flip-flop will not change.
When J=0 and K=1, the output Q of the flip-flop is set to 1.
When J=1 and K=0, the output Q of the flip-flop is reset to 0.
When J=1 andK=1, the clock pulse switches the output of the flip-flop to its complement state. This state is also referred to as toggle state. There is no forbidden or unpredictable state in a J K flip-flop.



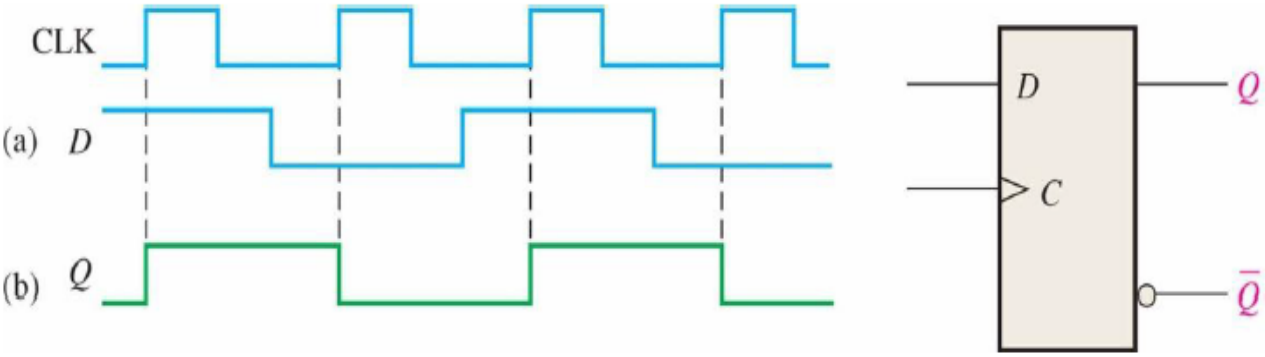

## D flip-flop:

The circuit diagram and the truth table of a D flip-flop are shown below.



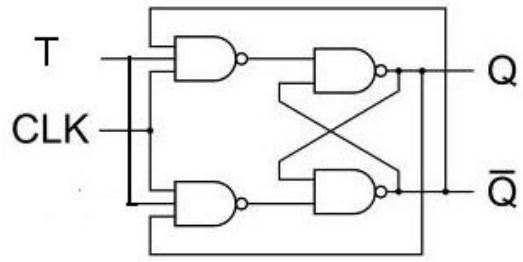

| CLK | D | Q | Comment |
|---|---|---|---|
| 0 | X | NC | No Change |
| ↑ | 0 | 0 | RESET |
| ↑ | 1 | 1 | SET |

When the CLK input is 0, the output of the flip-flop will not change irrespective of the input D. The flip-flop is disabled when the clock is 0.
When the CLK input is 1, the output of the flip-flop will change according to the D input. The flip-flop is enabled when the clock is 1. The response of the flipflop for various D input, when the CLK is 1, is as follows.
When D=0, the output Q of the flip-flop become 0 (reset).
When D=1, the output Q of the flip-flop become 1 (set). There is no forbidden or unpredictable state in a D flip-flop. A D flip-flop is sometimes referred to as a data flip flop.




## T flip-flop:

The circuit diagram and the truth table of a T flip-flop are shown below.




| CLK | T | Q | Comment |
|---|---|---|---|
| 0 | X | NC | No Change |
| ↑ | 0 | NC | No Change |
| ↑ | 1 | 1 | Toggle |

A T flip-flop can be realized from a J-K flip-flop when inputs J and K are combined to provide a single input designated as T. When the CLK input is 0, the output of the flip-flop will not change irrespective of the input T. The flip-flop is disabled when the clock is 0. When the CLK input is 1, the output of the flip-flop will change according to the T input. The flip-flop is enabled when the clock is 1.

The response of the flip-flop for various T input, when the CLK is 1, is as follows.

When T=0, the output Q of the flip-flop will remain unchanged

When T=1, the clock pulse switches the output of the flip-flop to its complement (toggle) state. A T flip-flop is sometimes referred to as a toggle flip-flop.