

2.9 DIGITAL LOGIC GATES

As Boolean functions are expressed in terms of AND, OR, and NOT operations, it is easier to implement the Boolean functions with these basic types of gates. However, for all practical purposes, it is possible to construct other types of logic gates. The following factors are to be considered for construction of other types of gates.


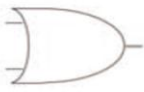
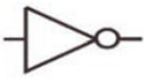
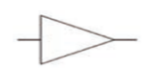




Name	Graphic Symbol	Algebraic Function	Truth Table		
			A	B	F
AND		$F = AB$	0	0	0
			0	1	0
			1	0	0
			1	1	1
OR		$F = A + B$	0	0	0
			0	1	1
			1	0	1
			1	1	1
Inverter or NOT		$F = A'$	A	F	
			0	1	
Buffer		$F = A$	A	F	
			0	0	
NAND		$F = (AB)'$	0	0	1
			0	1	1
			1	0	1
			1	1	0
NOR		$F = (A + B)'$	0	0	1
			0	1	0
			1	0	0
			1	1	0
Exclusive-OR (XOR)		$F = AB' + A'B$ $= A \oplus B$	0	0	0
			0	1	1
			1	0	1
			1	1	0
Equivalence Or Exclusive-NOR (XNOR)		$F = AB + A'B'$	0	0	1
			0	1	0
			1	0	0
			1	1	1

Figure 2-12

1. The feasibility and economy of producing the gate with physical parameters.
2. The possibility of extending to more than two inputs.
3. The basic properties of the binary operator such as commutability and associability.
4. The ability of the gate to implement the Boolean functions alone or in conjunction

with other gates.

There are eight functions—Transfer (or buffer), Complement, AND, OR, NAND, NOR, Exclusive-OR (XOR), and Equivalence (XNOR) that may be considered to be standard gates in digital design.

The transfer or buffer and complement or inverter or NOT gates are unary gates, *i.e.*, they have single input, while other logic gates have two or more inputs.

2.9.1 Extension to Multiple Inputs

A gate can be extended to have multiple inputs if its binary operation is commutative and associative. AND and OR gates are both commutative and associative.

For the AND function, $AB = BA$ -commutative

and

$(AB)C = A(BC) = ABC$. -associative

For the OR function, $A + B = B + A$ -commutative

and

$(A + B) + C = A + (B + C)$. -associative

These indicate that the gate inputs can be interchanged and these functions can be extended to three or more variables very simply as shown in Figures 2.13(a) and 2.13(b).



Figure 2-13(a)



Figure 2-13(b)

The NAND and NOR functions are the complements of AND and OR functions respectively. They are commutative, but not associative. So these functions can not be extended to multiple input variables very simply. However, these gates can be extended to multiple inputs with slightly modified functions as shown in Figures 2.14(a) and 2.14(b) below.

For NAND function, $(AB)' = (BA)'$. -commutative
 But, $((AB)'C)' \neq (A(BC)')'$. -does not follow associative property.

As $((AB)'C)' = (AB) + C'$ and
 $(A(BC)')' = A' + BC$.

Similarly, for NOR function, $((A + B)' + C)' \neq (A + (B + C)')'$.

As, $((A + B)' + C)' = (A + B)C' = AC' + BC'$.

And $(A + (B + C)')' = A'(B + C) = A'B + A'C$.

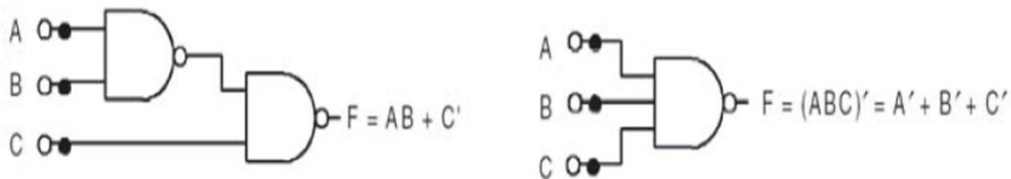


Figure 2-14(a)

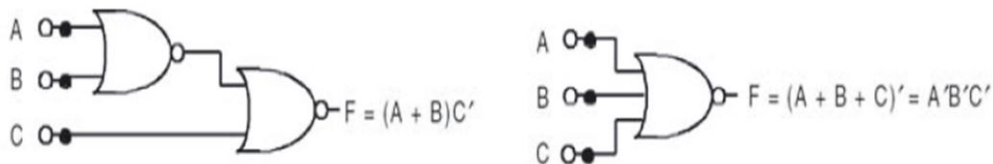


Figure 2-14(b)

The Exclusive-OR gates and equivalence gates both possess commutative and associative properties, and they can be extended to multiple input variables. For a multiple-input Ex-OR (XOR) gate output is low when even numbers of 1s are applied to the inputs, and when the number of 1s is odd the output is logic 1. Equivalence gate or XNOR gate is equivalent to XOR gate followed by NOT gate and hence its logic behavior is opposite to the XOR gate. However, multiple-input exclusive-OR and equivalence gates are uncommon in practice. Figures 2.15(a) and 2.15(b) describe the extension to multiple-input exclusive-OR and equivalence gates.



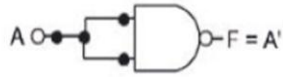
Figure 2-15(a)



Figure 2-15(b)

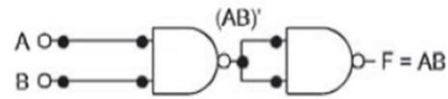
2.9.2 Universal Gates

NAND gates and NOR gates are called *universal gates* or *universal building blocks*, as any type of gates or logic functions can be implemented by these gates. Figures 2.16(a)-(e) show how various logic functions can be realized by NAND gates and Figures 2.17(a)-(d) show the realization of various logic gates by NOR gates.



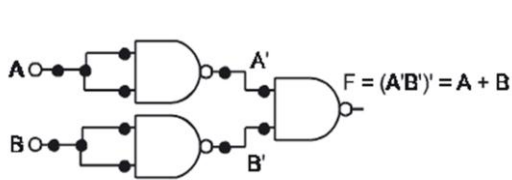
NOT function: $F = A'$

Figure 2-16(a)



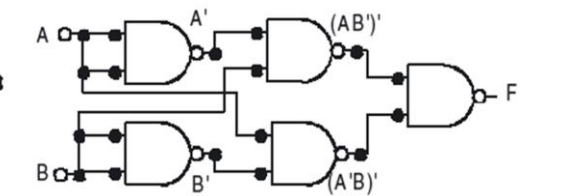
AND function: $F = AB$

Figure 2-16(b)



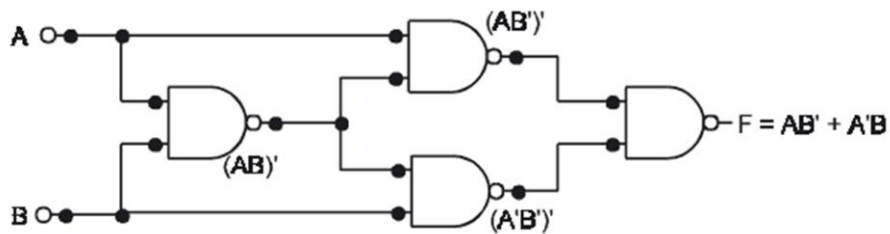
OR function: $F = A + B$

Figure 2-16(c)



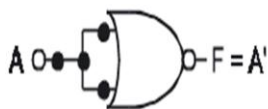
Ex-OR function: $F = ((AB)'(A'B))' = AB' + A'B$

Figure 2-16(d)



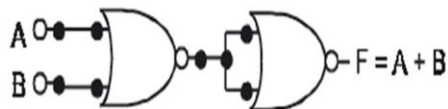
Ex-OR gate with reduced number of NAND gates

Figure 2-16(e)



NOT function: $F = A'$

Figure 2-17(a)



OR function: $F = A + B$

Figure 2-17(b)

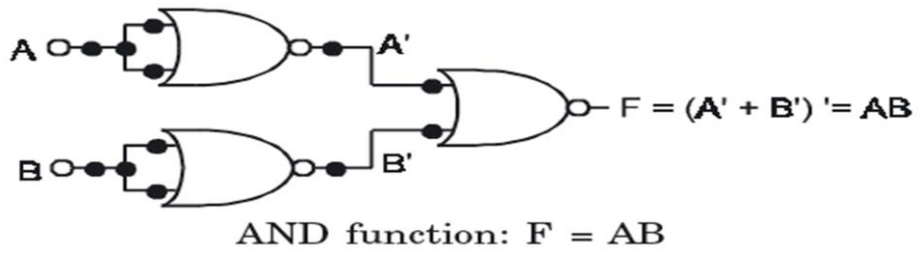
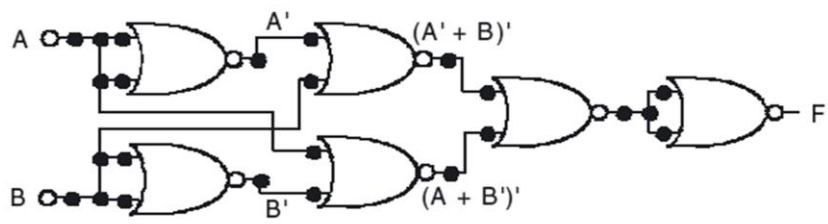


Figure 2-17(c)



Ex-OR function: $F = [(A' + B)' + (A + B)'] = AB' + A'B$

Figure 2-17(d)