3 Combinatorial Logic Circuits

In the previous chapter, operation and truth tables of single gates were discussed. However, in practise, single gates are seldom useful and combinations of several gates are employed for a particular application. For example, see Figure 3.1 where different gates are used to obtain the output *F*.

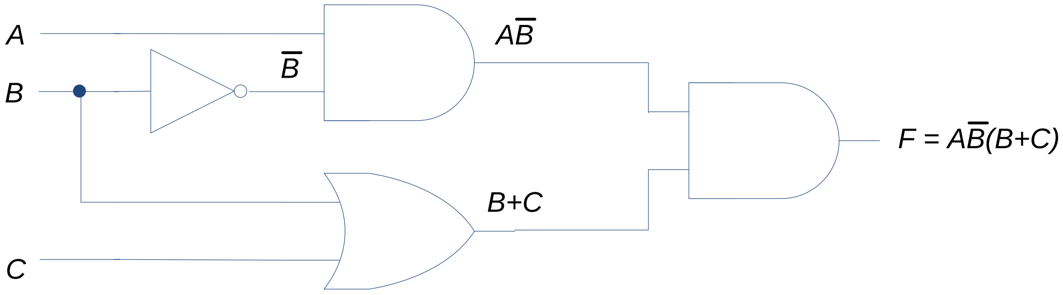


Figure 3.1: Example of combinatorial logic circuit.

3.1 Logic circuit simplification

Very often, there is the need to simplify logic circuits (whenever possible). For example, the circuit shown in Figure 3.1 requires four gates but equivalent logic output can be obtained with just two gates by simplifying the expression as follows:

$$\begin{aligned} F &= A\overline{B}(B+C) \\ &= A\overline{B}B + A\overline{B}C \qquad \text{after expanding} \\ &= A\overline{B}C\,. \end{aligned}$$

$A\overline{B}B$ is zero due to the presence of $\overline{B}B$ as shown in the truth table given in Table 3.1. The simplified circuit is given in Figure 3.2. Table 3.2 gives the truth table and it can be seen that the outputs given by expressions $F = A\overline{B}(B+C)$ and $F = A\overline{B}C$ are the same.

Table 3.1: Truth table for $A\overline{B}B$

A	*B*	$\overline{B}$	$\overline{B}B$	$A\overline{B}B$
0	0	1	0	0
0	1	0	0	0
1	0	1	0	0
1	1	0	0	0

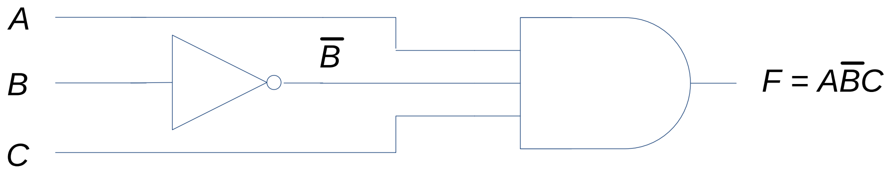


Figure 3.2: Simplified logic circuit.

Table 3.2: Truth table for $F = A\overline{B}(B+C)$ and $F = A\overline{B}C$

A	B	C	$F = A\overline{B}(B+C)$	$F = A\overline{B}C$
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	0	0
1	0	1	1	1
1	1	0	0	0
1	1	1	0	0

The above simplification may not be clear at this stage but that will be the purpose of the following sections to look into Boolean algebra that will be useful to simplify logic circuits. Not only will the simplification result in lower cost, smaller and simpler design (since fewer gates will be used), it will also reduce other complications such as overheating and propagation delay.

3.2 Boolean algebra

Basic axioms of Boolean algebra are shown in Table 3.3, while Table 3.4 shows the Boolean theorems for operation of a single variable and a constant (either 0 or1).

Boolean algebra satisfies commutative and associative laws. Therefore, the order of variables in a product or sum does not matter and the order of evaluating sub-expression in brackets does not matter. For example:

Commutative law: $A+B=B+A$ and $A\cdot B=B\cdot A$;

Associative law: $A+(B+C)=(A+B)+C=A+B+C$ and $A\cdot(B\cdot C)=(A\cdot B)\cdot C=A\cdot B\cdot C$.

Boolean algebra also satisfies the distributive law where the expression can be expanded by multiplying out the terms. For example:

Distributive law: $A\cdot(B+C)=A\cdot B+A\cdot C$.

It should be evident by now that when an expression contains AND and OR, AND operator takes precedence over OR operator. For example, $0\cdot1+1\cdot1=0+1=1$ and not $0\cdot1+1\cdot1=0\cdot1\cdot1=0$.

Table 3.3: Basic axioms of Boolean algebra

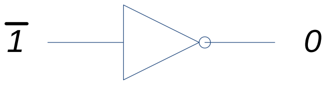
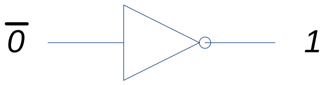
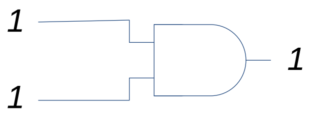
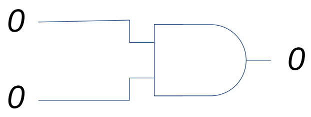
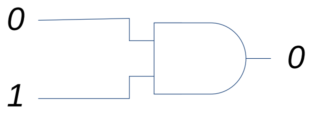
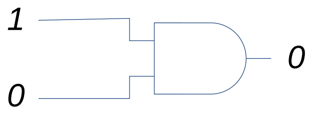
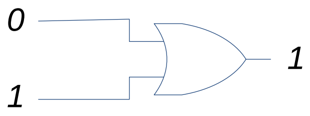
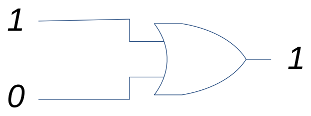
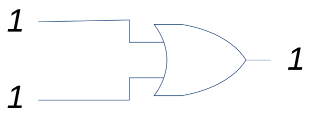
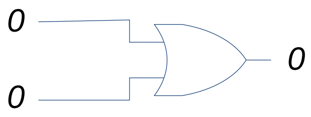
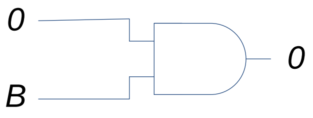
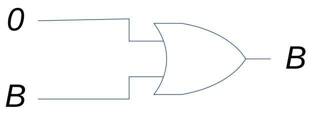
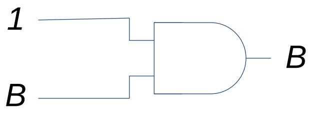
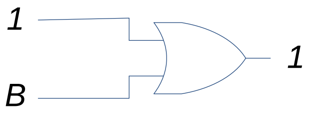
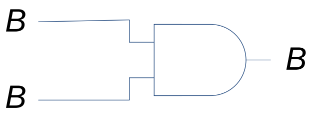
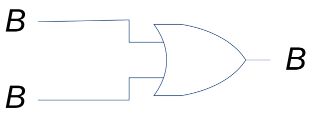
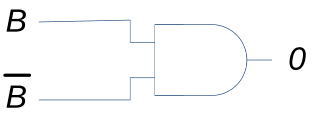
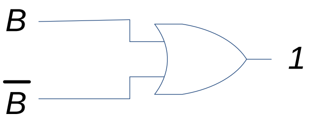
$\overline{1} = 0$		$\overline{0} = 1$	
$1 \cdot 1 = 1$		$0 \cdot 0 = 0$	
$0 \cdot 1 = 0$		$1 \cdot 0 = 0$	
$0 + 1 = 1$		$1 + 0 = 1$	
$1 + 1 = 1$		$0 + 0 = 0$	

Table 3.4: Boolean theorems for operation of a single variable and a constant

$0 \cdot B = 0$		$0 + B = B$	
$1 \cdot B = B$		$1 + B = 1$	
$B \cdot B = B$		$B + B = B$	
$B \cdot \overline{B} = 0$		$B + \overline{B} = 1$	

3.3 DeMorgan's theorem

DeMorgan's theorem is very useful to simplify expressions when they contain a bar (inversion) over more than a single variable. It states that an inverted expression can be replaced by its individual inverted variables but with AND replaced by OR and vice versa. For example:

DeMorgan's theorem: $\overline{A \cdot B} = \overline{A} + \overline{B}$ and $\overline{A + B} = \overline{A} \cdot \overline{B}$

Figure 3.3 shows the circuit equivalence using DeMorgan's theorem.

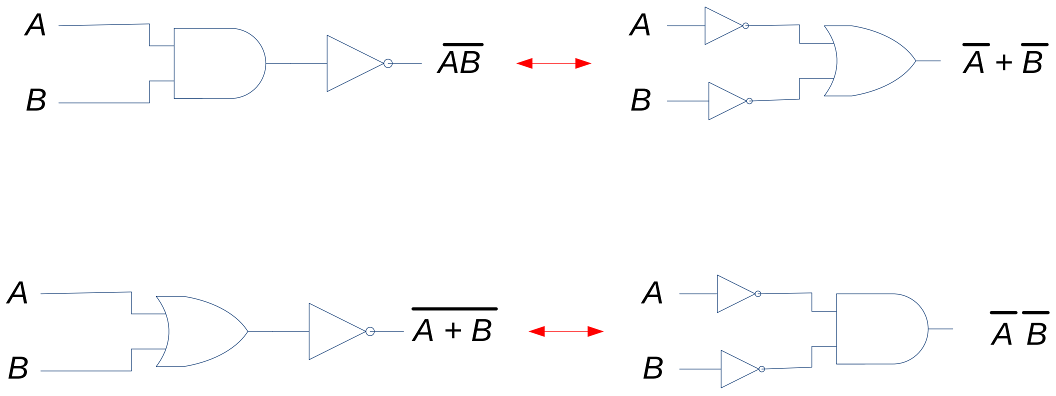


Figure 3.3: Circuit equivalence using DeMorgan's theorem.

3.3.1 Examples illustrating DeMorgan's theorem

The following examples show the usefulness of using DeMorgan's theorem. Note that from now on, the use of AND ($\cdot$) sign in the expression will be dropped for simplicity sake unless noted otherwise, so $F = A \cdot B \cdot C$ will be written as $F = ABC$..

$$\begin{aligned} F &= \overline{A + B + \overline{C}} && \text{apply DeMorgan's theorem} \\ &= \overline{A}\,\overline{B}\,\overline{\overline{C}} \\ &= \overline{A}\,\overline{B}C\,. \end{aligned}$$

$$\begin{aligned} F &= \overline{ABC + D} && \text{apply DeMorgan's theorem} \\ &= \overline{ABC}\,\overline{D} && \text{apply DeMorgan's theorem again} \\ &= \overline{D}(\overline{A} + \overline{B} + \overline{C})\,. \end{aligned}$$

$$\begin{aligned} F &= A + \overline{(B + C)D} && \text{apply DeMorgan's theorem} \\ &= A + \overline{B + C} + \overline{D} && \text{apply DeMorgan's theorem again} \\ &= A + \overline{B}\,\overline{C} + \overline{D}\,. \end{aligned}$$

3.4 More examples

In this section, several examples are given to illustrate simplification using Boolean algebra and DeMorgan's theorem:

$$\begin{aligned} F &= \overline{A + B} + \overline{A}B && \text{apply DeMorgan's theorem} \\ &= \overline{A}\,\overline{B} + \overline{A}B \\ &= \overline{A}(\overline{B} + B) && \text{see Table 3.4, } (\overline{B} + B) = 1 \\ &= \overline{A} \end{aligned}$$

$$\begin{aligned} F &= \overline{A + B + C} + A\overline{B} + \overline{B}C && \text{apply DeMorgan's theorem} \\ &= \overline{A}\,\overline{B}\,\overline{C} + \overline{B}(A + C) \\ &= \overline{B}(\overline{A}\,\overline{C} + A + C) \\ &= \overline{B}(\overline{A + C} + A + C) && \text{apply inverse of DeMorgan's theorem} \\ &= \overline{B}\,. && \text{see Table 3.4, since } (\overline{X} + X) = 1 \text{ where } X = A + C \end{aligned}$$

$$
\begin{aligned}
F &= \overline{AB} + ABC && \text{apply DeMorgan's theorem}^{6} \\
&= \overline{A} + \overline{B} + ABC && \\
&= \overline{A} + \overline{B} + C(AB) && \\
&= \overline{A} + \overline{B} + C(\overline{\overline{AB}}) && \text{apply DeMorgan's theorem again} \\
&= \overline{A} + \overline{B} + C(\overline{\overline{A} + \overline{B}}) && \text{let } X = \overline{A} + \overline{B} \\
&= X + C\overline{X} && \\
&= X + C\overline{X} + XC && \text{add } XC \text{ since } X + XC = X(1 + C) = X \\
&= X + C(\overline{X} + X) && \\
&= X + C && \text{as } X + \overline{X} = 1 \\
&= \overline{A} + \overline{B} + C && \text{replace } X = \overline{A} + \overline{B}
\end{aligned}
$$

[6] There is a simpler method to obtain the solution by letting X=AB in the first place but the shown procedure illustrates several useful simplifications.

As another example, consider the circuit diagram given in Figure 3.4 which can be simplified as

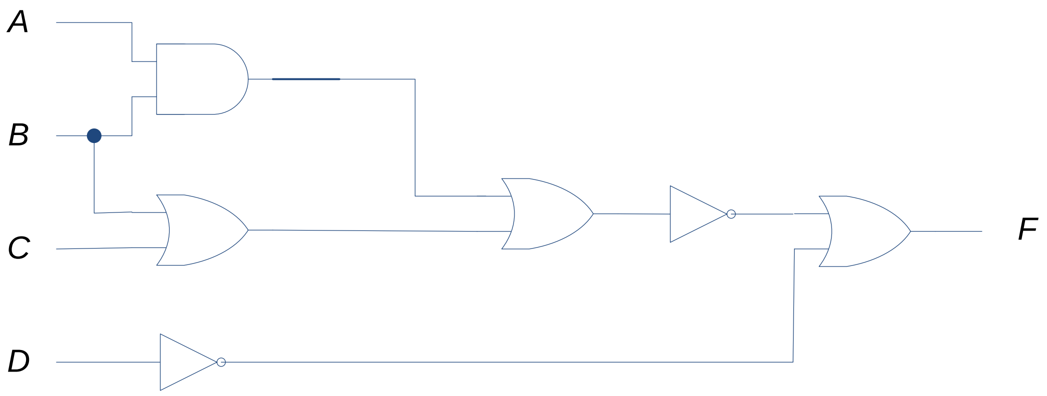


Figure 3.4: Logic circuit example for simplification.

$\overline{AB+(B+C)}+\overline{D}$	obtain the expression
$\overline{AB}\,\overline{(B+C)}+\overline{D}$	using DeMorgan's theorem on the top (outermost) invers
$(\overline{A}+\overline{B})(\overline{B}\,\overline{C})+\overline{D}$	using DeMorgan's theorem again
$\overline{A}\,\overline{B}\,\overline{C}+\overline{B}\,\overline{B}\,\overline{C}+\overline{D}$	after expanding
$\overline{A}\,\overline{B}\,\overline{C}+\overline{B}\,\overline{C}+\overline{D}$	since $\overline{B}\,\overline{B}=\overline{B}$
$\overline{B}\,\overline{C}(\overline{A}+1)+\overline{D}$	as $\overline{A}+1=1$
$\overline{B}\,\overline{C}+\overline{D}$.	

The correctness of the simplified expression can be verified by constructing a truth table and comparing the output from both expressions. The simplified logic circuit diagram is shown in Figure 3.5 where only five gates are required as opposed to six gates in the original circuit. It can be seen that there is no input *A* as its logic value does not affect the output based on the simplified expression.

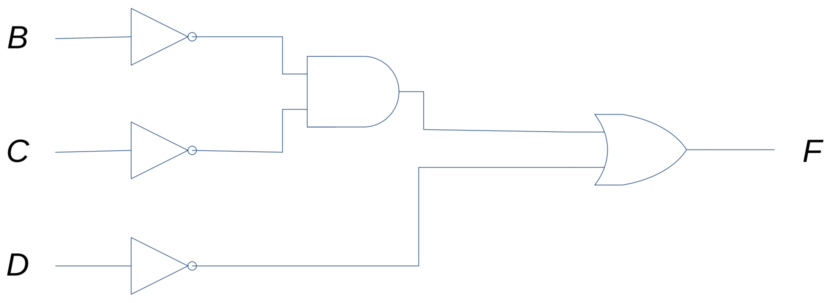


Figure 3.5: Simplified logic circuit of the example shown in Figure 3.4.

While the expression for the logic circuit shown in Figure 3.5 is simplified to single literals, it is interesting to note that another equivalent logic circuit shown in Figure 3.6 only requires four gates as $F = \overline{B}\,\overline{C} + \overline{D} = \overline{B+C} + \overline{D}$.

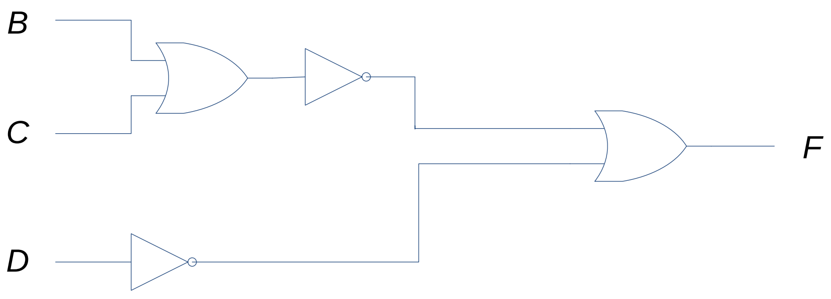


Figure 3.6: Equivalent logic circuit of the example shown in Figures 3.4 and 3.5.

If complement inputs are available, then the simplified circuit shown in Figure 3.5 will only require two gates as shown in Figure 3.7.

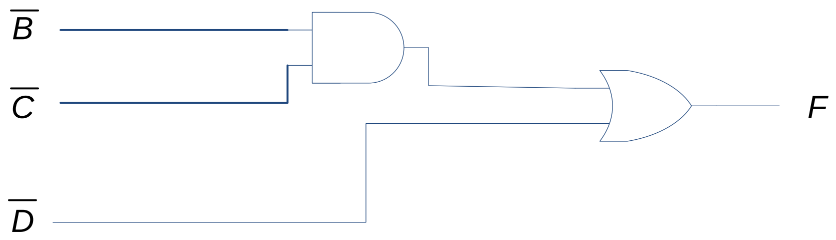


Figure 3.7: Simplified logic circuit when complement inputs are available.

3.5 XOR and XNOR gates

To conclude the chapter, it is useful to look at two more frequently used gates: Exclusive OR (XOR) and Exclusive NOR (XNOR). These gates would be useful when circuitry such as half adders and full adders are discussed in later chapters. XOR gate as shown in Figure 3.8 has algebraic representation, $F = A\overline{B} + \overline{A}B$ or more commonly written as $F = A \oplus B$.

The truth table that gives the output *F* for inputs *A* and *B* is given in Table 3.5. It can be seen that when both inputs have the same logic value, the output is LOW. The output is HIGH when the input logic values are dissimilar, i.e. one LOW and one HIGH.

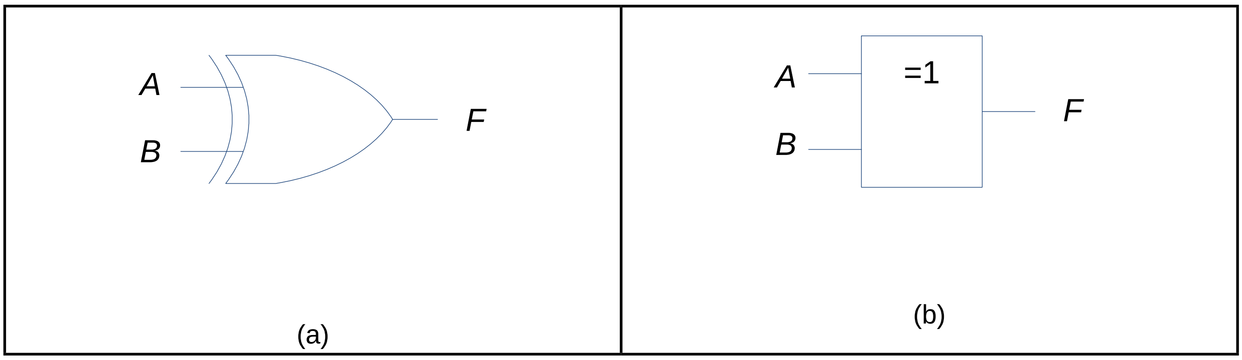


Figure 3.8: NOR gate logic symbols: (a) traditional (b) IEEE/ANSI standard.

Table 3.5: Truth table for two-input XOR gate

A	*B*	*F*
0	0	0
0	1	1
1	0	1
1	1	0

XNOR gate is simply XOR with an inversion. The gate is shown in Figure 3.9 and has algebraic representation,

$$\begin{array}{ll}
F = \overline{A\overline{B} + \overline{A}B} & \\
F = \overline{A\overline{B}}\,\overline{\overline{A}B} & \text{using DeMorgan's theorem on the top inversion} \\
F = (\overline{A} + B)(A + \overline{B}) & \text{using DeMorgan's theorem again} \\
F = \overline{A}A + B\overline{B} + AB + \overline{A}\overline{B} & \text{after expanding} \\
F = AB + \overline{A}\overline{B} & \text{expression for XNOR}
\end{array}$$

or more commonly written as $F = \overline{A \oplus B}$.

The truth table is given in Table 3.6. The output is HIGH when both inputs have the same logic value. The output is LOW when the input logic values are dissimilar, i.e. one LOW and one HIGH.

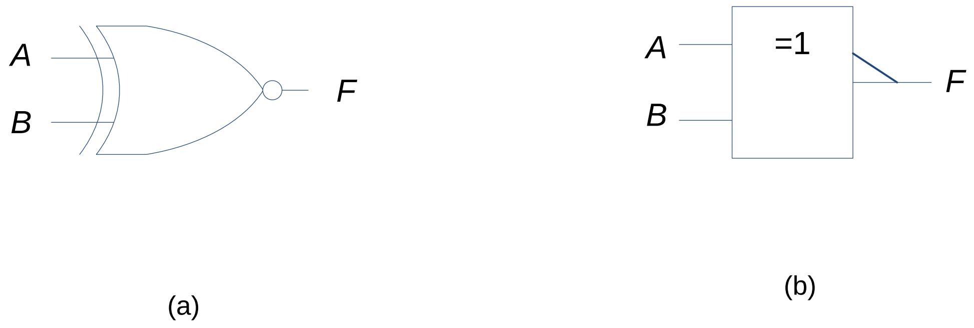


Figure 3.9: XNOR gate logic symbols: (a) traditional (b) IEEE/ANSI standard.

Table 3.6: Truth table for two-input XNOR gate

A	*B*	*F*
0	0	1
0	1	0
1	0	0
1	1	1

3.5.1 Boolean algebra for XOR operation

Table 3.7 shows the Boolean algebra for XOR operation. XOR operation is also both commutative and associative: $A \oplus B = B \oplus A$ and $A \oplus (B \oplus C) = (A \oplus B) \oplus C = A \oplus B \oplus C$.

Table 3.7: Boolean algebra for XOR operation

$A \oplus 0 = A$	$A \oplus A = 0$	$A \oplus \overline{B} = \overline{A \oplus B}$
$A \oplus 1 = \overline{A}$	$A \oplus \overline{A} = 1$	$\overline{A} \oplus B = \overline{A \oplus B}$

3.5.2 Parity checker

As mentioned earlier, XOR gates are useful when designing more advanced circuitry such as adders, but these are also used in parity checker devices. Parity checker is used to reduce errors from transmitting a binary code across a communication channel. For example, if the seven bit ASCII code for W, 1010111 (see Table 1.1) is to be transmitted, an eight parity bit is appended at the beginning of the code. This parity bit will either be 0 or 1 depending on whether even or odd parity is required. Assuming that it is even parity checker, then the total number of bits will be even. In this case, the parity bit will be 1 and code to be transmitted will be 11010111.

XOR gates can be used as even parity checker. For example, with three inputs, the expression will be $F = A \oplus B \oplus C$ and the output is HIGH if one of the inputs or all three inputs are HIGH. Similarly, for eight inputs, the output is HIGH when odd number of inputs is HIGH.

Figure 3.10 shows the logic circuit using seven two-input XOR gates where the bits representing the code are A_0, A_1,…., A_6 and the parity bit is *P*. The output F will be HIGH when odd number of inputs is HIGH. So if the code is not transmitted correctly (say resulting in odd number of 1s), then the LED will light-up to show that an error has occured. On the other hand, with correct transmission, the number of 1s will be even and the output will be low (i.e. LED will not light-up).

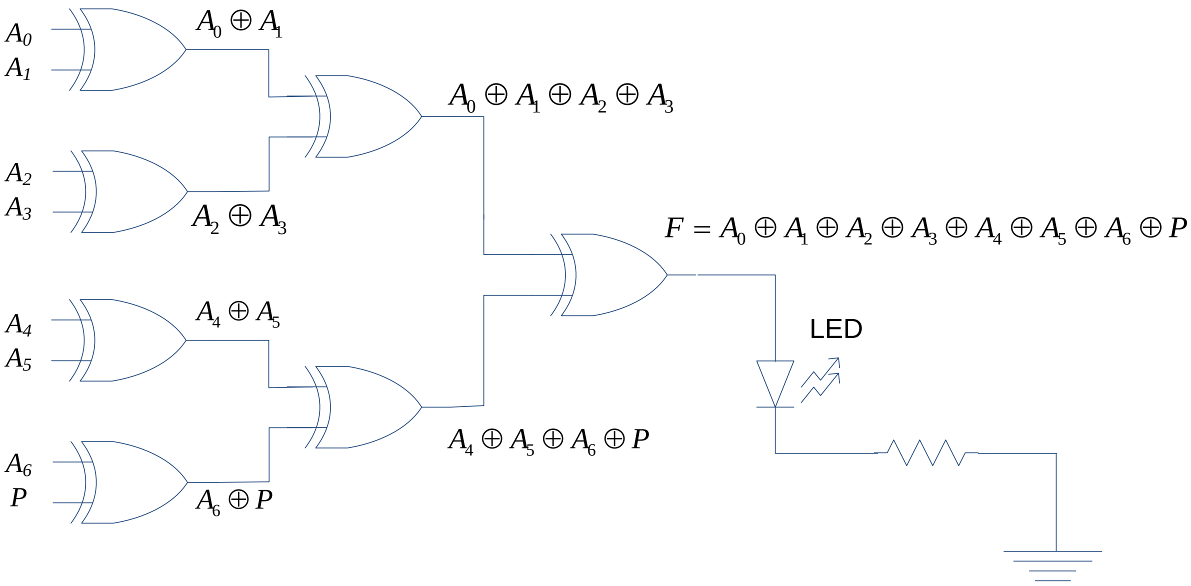

A_0
A_1
$A_0 \oplus A_1$
A_2
A_3
$A_2 \oplus A_3$
$A_0 \oplus A_1 \oplus A_2 \oplus A_3$
$F = A_0 \oplus A_1 \oplus A_2 \oplus A_3 \oplus A_4 \oplus A_5 \oplus A_6 \oplus P$
LED
A_4
A_5
$A_4 \oplus A_5$
$A_4 \oplus A_5 \oplus A_6 \oplus P$
A_6
P
$A_6 \oplus P$

Figure 3.10: XOR gate usage as even parity checker.

It should be obvious that XNOR gates can be used as odd parity checker as the output will be HIGH only when even number of inputs is HIGH.