فصل سوم

حداقل‌سازی در سطح گیت

اهمیت ساده‌سازی عبارات جبری و در نتیجه ساده‌سازی وکاهش میزان پیچیدگی سیستم‌های دیجیتال مبتنی بر گیت‌های منطقی دیجیتال ؟.

یکی از روش‌های ساده‌سازی **استفاده از قوانین و اتحادهای جبری** (جبر بول) است گر چه عمدتاً کارایی این روش محدود است زیرا قوانین و اتحادهای آن محدود است.

یک روش ساده‌سازی دیگر استفاده از «**روش نقشه**[1]» است؛ روش نقشه معادلِ گرافیکیِ جدول صحت/درستی است. نام دیگر آن، «**جدول/نقشه‌ی کارنو**[2]» یا «نقشه‌ی K[3]» است. هدف از جدول کارنو رسیدن به عبارات جبریِ معادل با کمترین تعداد جملات و کمترین تعداد لیترال‌ها در هر جمله است.

نقشه‌ی دو متغیره:

قالب کلی:

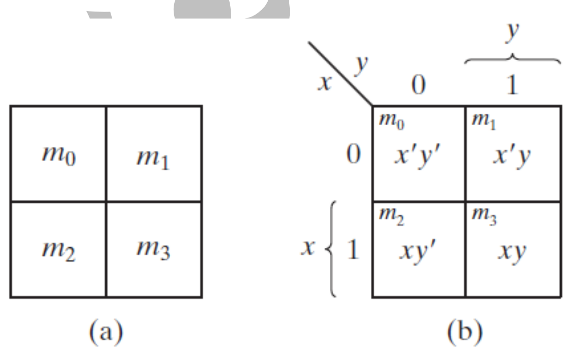
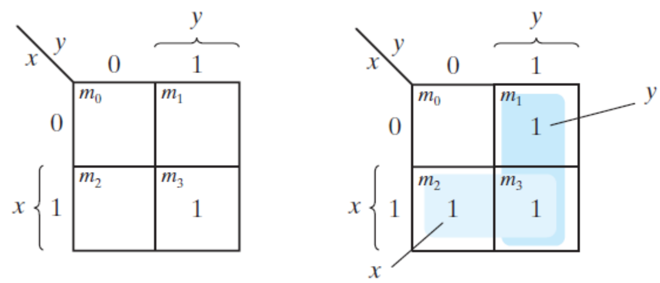


FIGURE 3.1
Two-variable K-map

مثال مربوط به نحوه‌ی نمایش یک تابع به کمک جدول کارنو:

[1] Map Method
[2] Karnaugh map
[3] K-map



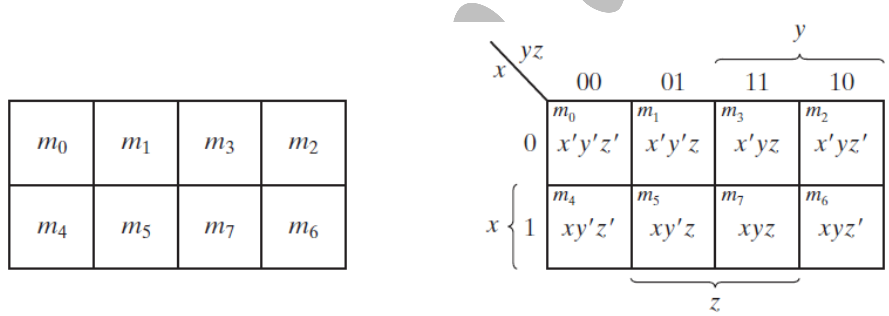
(a) xy (b) $x + y$

FIGURE 3.2
Representation of functions in the map

$$m_1 + m_2 + m_3 = x'y + xy' + xy = x + y$$

نقشه‌ی سه‌متغیره:

قالب کلی: از کد گری استفاده می‌شود.



(a) (b)

FIGURE 3.3
Three-variable K-map

به دلیل استفاده از کد گری، هر دو جمله‌ی مربوط به دو خانه‌ی مجاور هم تنها در یک متغیر اختلاف دارند که یکی از این جملات شامل خود متغیر و جمله‌ی دیگر شامل متغیرِ پریم‌دار است؛ لذا به سادگی ملاحظه می‌شود که جمع عبارات جبری دو خانه به تنها یک جمله و آن هم شامل تنها دو متغیر ساده می‌شود. برای مثال در قالب کلی اخیر داریم:

$$m_5 + m_7 = xy'z + xyz = xz(y' + y) = xz$$

مثال:

EXAMPLE 3.1

Simplify the Boolean function

$$F(x, y, z) = \Sigma(2, 3, 4, 5)$$

First, a 1 is marked in each minterm square that represents the function. This is shown in Fig. 3.4, in which the squares for minterms 010, 011, 100, and 101 are marked with 1's. The next step is to find possible adjacent squares. These are indicated in the map by two shaded rectangles, each enclosing two 1's. The upper right rectangle represents the area enclosed by $x'y$. This area is determined by observing that the two-square area is in row 0, corresponding to x', and the last two columns, corresponding to y. Similarly, the lower left rectangle represents the product term xy'. (The second row represents x and the two left columns represent y'.) The sum of four minterms can be replaced by a sum of

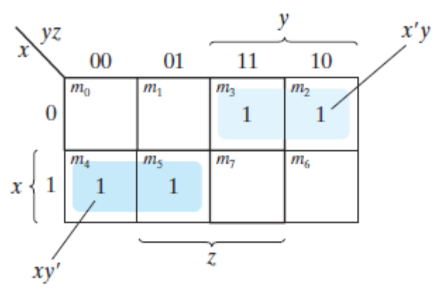


FIGURE 3.4
Map for Example 3.1, $F(x, y, z) = \Sigma(2, 3, 4, 5) = x'y + xy'$

only two product terms. The logical sum of these two product terms gives the simplified expression

$$F = x'y + xy'$$

«مجاور بودنِ» دو خانه از جدول لزوماً به معنای مجاور بودن فیزیکیِ آن دو خانه نیست بلکه تغییر تنها یک بیت در کدهای متناظر آن دو خانه، مجاور بودن را تعیین می‌کند. در قالب کلی (شکل ۳ـ۳) خانه‌های m_0 و m_2 و یا خانه‌های m_4 و m_6 نیز با هم مجاور هستند. زیرا:

$$m_0 + m_2 = x'y'z' + x'yz' = x'z'(y' + y) = x'z'$$
$$m_4 + m_6 = xy'z' + xyz' = xz' + (y' + y) = xz'$$

EXAMPLE 3.2

Simplify the Boolean function

$$F(x, y, z) = \Sigma(3, 4, 6, 7)$$

The map for this function is shown in Fig. 3.5. There are four squares marked with 1's, one for each minterm of the function. Two adjacent squares are combined in the third column to give a two-literal term yz. The remaining two squares with 1's are also adjacent by the new definition. These two squares, when combined, give the two-literal term xz'. The simplified function then becomes

$$F = yz + xz'$$

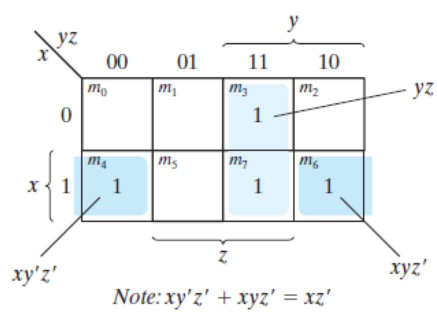


FIGURE 3.5
Map for Example 3.2, $F(x, y, z) = \Sigma(3, 4, 6, 7) = yz + xz'$

مجموعه‌ی خانه‌های مجاور می‌تواند شامل بیش از دو (البته شامل تعداد توان صحیحی از ۲، مثلاً ۴ یا ۸) خانه‌ی مجاور باشد. برای مثال در قالب کلی، خانه‌های m_0، m_2، m_4 و m_6 با هم مجاور بوده و جمع آنها عبارت ساده شده‌ی زیر را نتیجه می‌دهد:

$$\begin{aligned} m_0 + m_2 + m_4 + m_6 &= x'y'z' + x'yz' + xy'z' + xyz' \\ &= x'z'(y' + y) + xz'(y' + y) \\ &= x'z' + xz' = z'(x' + x) = z' \end{aligned}$$

نکته‌ی بعدی این است که دو مجموعه‌ی مختلف از خانه‌های مجاور می‌توانند به منظور رسیدن به ساده‌سازی بیشتر، شامل یک یا چند خانه‌ی مشترک باشند زیرا میزان ساده شدن عبارت نهایی مربوط به هر گروه/مجموعه از خانه‌های مجاور بستگی به تعداد خانه‌های موجود در هر گروه دارد؛ بنابراین در تشکیل گروه‌ها به دنبال تشکیل گروه‌هایی با بیشترین تعداد خانه‌های مجاور باید بود.

نکته‌ی بعدی این که در هر گروه از خانه‌های مجاور، می‌توان محیط آن گروه (که به شکل یم مربع یا مستطیل است) را پیمود و یک دور کامل زد؛ در این کار، شرط حرکت از یک خانه به خانه مجاور (در جهت افقی یا عمودی) این است که کد متنار با آن دو خانه، تنها در یک بیت با هم اختلاف داشته باشند (ـ م).

EXAMPLE 3.3

Simplify the Boolean function

$$F(x, y, z) = \Sigma(0, 2, 4, 5, 6)$$

The map for F is shown in Fig. 3.6. First, we combine the four adjacent squares in the first and last columns to give the single literal term z'. The remaining single square, representing minterm 5, is combined with an adjacent square that has already been used once. This is not only permissible, but rather desirable, because the two adjacent squares give the two-literal term xy' and the single square represents the three-literal minterm $xy'z$. The simplified function is

$$F = z' + xy'$$

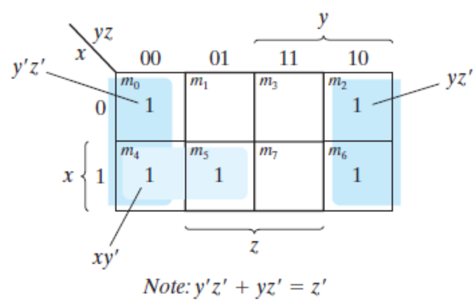


FIGURE 3.6
Map for Example 3.3, $F(x, y, z) = \Sigma(0, 2, 4, 5, 6) = z' + xy'$

برخی اوقات ممکن است تابع به صورت مجموع مینترم‌ها داده نشود. در چنین مواقعی جدول کارنو می‌تواند به ما کمک کند تابع را برحسب مجموع مینترم‌ها توصیف و سپس آن را ساده‌سازی کنیم.

EXAMPLE 3.4

For the Boolean function

$$F = A'C + A'B + AB'C + BC$$

(a) Express this function as a sum of minterms.
(b) Find the minimal sum-of-products expression.

Note that F is a sum of products. Three product terms in the expression have two literals and are represented in a three-variable map by two squares each. The two squares corresponding to the first term, $A'C$, are found in Fig. 3.7 from the coincidence of A' (first row) and C (two middle columns) to give squares 001 and 011. Note that, in marking 1's in the squares, it is possible to find a 1 already placed there from a preceding term. This happens with the second term, $A'B$, which has 1's in squares 011 and 010. Square 011 is common with the first term, $A'C$, though, so only one 1 is marked in it. Continuing in this fashion, we determine that the term $AB'C$ belongs in square 101, corresponding to minterm 5, and the term BC has two 1's in squares 011 and 111. The function has a total of five minterms, as indicated by the five 1's in the map of Fig. 3.7. The minterms are read directly from the map to be 1, 2, 3, 5, and 7. The function can be expressed in sum-of-minterms form as

$$F(A, B, C) = \Sigma(1, 2, 3, 5, 7)$$

The sum-of-products expression, as originally given, has too many terms. It can be simplified, as shown in the map, to an expression with only two terms:

$$F = C + A'B$$

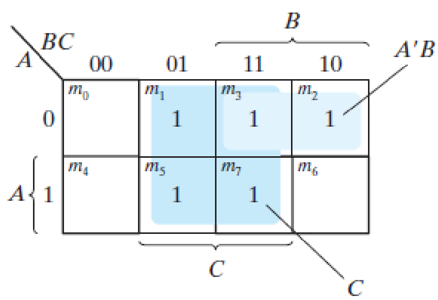


FIGURE 3.7
Map of Example 3.4, $A'C + A'B + AB'C + BC = C + A'B$

نقشه‌ی چهارمتغیره

قالب کلی:

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

(a)

$wx \backslash yz$	00	01	11	10
00	m_0 $w'x'y'z'$	m_1 $w'x'y'z$	m_3 $w'x'yz$	m_2 $w'x'yz'$
01	m_4 $w'xy'z'$	m_5 $w'xy'z$	m_7 $w'xyz$	m_6 $w'xyz'$
11	m_{12} $wxy'z'$	m_{13} $wxy'z$	m_{15} $wxyz$	m_{14} $wxyz'$
10	m_8 $wx'y'z'$	m_9 $wx'y'z$	m_{11} $wx'yz$	m_{10} $wx'yz'$

y (columns 11, 10); x (rows 01, 11); w (rows 11, 10); z (columns 01, 11)

(b)

FIGURE 3.8
Four-variable map

EXAMPLE 3.5

Simplify the Boolean function

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

Since the function has four variables, a four-variable map must be used. The minterms listed in the sum are marked by 1's in the map of Fig. 3.9. Eight adjacent squares marked with 1's can be combined to form the one literal term y'. The remaining three 1's on the right cannot be combined to give a simplified term; they must be combined as two or four adjacent squares. The larger the number of squares combined, the smaller is the number of literals in the term. In this example, the top two 1's on the right are combined with the top two 1's on the left to give the term $w'z'$. Note that it is permissible to use the same square more than once. We are now left with a square marked by 1 in the third row and fourth column (square 1110). Instead of taking this square alone (which will give a term with four literals), we combine it with squares already used to form an area of four adjacent squares. These squares make up the two middle rows and the two end columns, giving the term xz'. The simplified function is

$$F = y' + w'z' + xz'$$

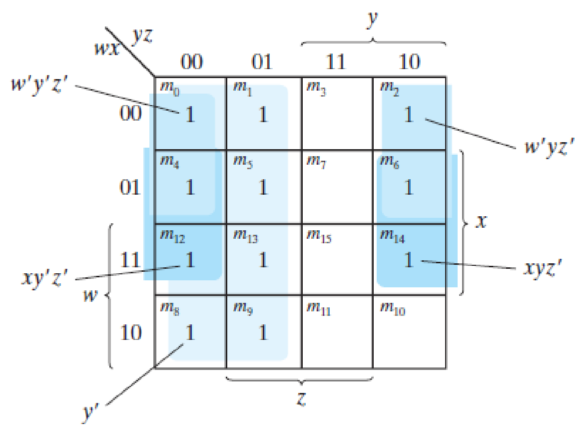


FIGURE 3.9
Map for Example 3.5, $F(w, x, y, z) = \Sigma(0,1, 2, 4, 5, 6, 8, 9, 12, 13, 14) = y' + w'z' + xz'$

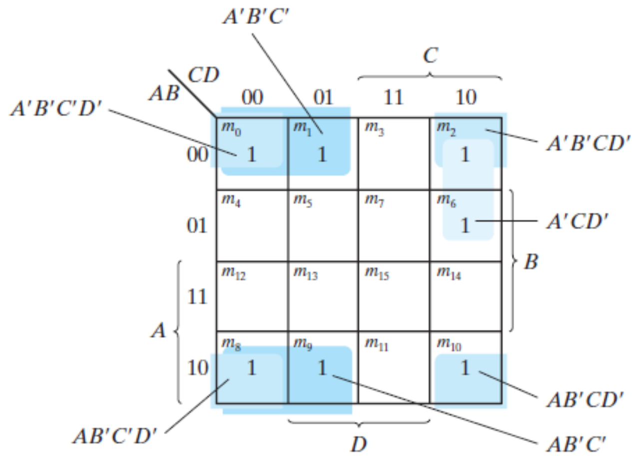
■

EXAMPLE 3.6

Simplify the Boolean function

$$F = A'B'C' + B'CD' + A'BCD' + AB'C'$$

The area in the map covered by this function consists of the squares marked with 1's in Fig. 3.10. The function has four variables and, as expressed, consists of three terms with



Note: $A'B'C'D' + A'B'CD' = A'B'D'$
$AB'C'D' + AB'CD' = AB'D'$
$A'B'D' + AB'D' = B'D'$
$A'B'C' + AB'C' = B'C'$

FIGURE 3.10
Map for Example 3.6, $A'B'C' + B'CD' + A'BCD' + AB'C' = B'D' + B'C' + A'CD'$

three literals each and one term with four literals. Each term with three literals is represented in the map by two squares. For example, $A'B'C'$ is represented in squares 0000 and 0001. The function can be simplified in the map by taking the 1's in the four corners to give the term $B'D'$. This is possible because these four squares are adjacent when the map is drawn in a surface with top and bottom edges, as well as left and right edges, touching one another. The two left-hand 1's in the top row are combined with the two 1's in the bottom row to give the term $B'C'$. The remaining 1 may be combined in a two-square area to give the term $A'CD'$. The simplified function is

$$F = B'D' + B'C' + A'CD'$$

■

موجب اصلی و موجب اصلی اساسی

جهت ساده‌سازی ابتدا بزرگترین گروههایی که هیچ اشتراکی با هم ندارند (به نام موجب‌های اصلی اساسی[1]) را شناسایی و عبارتهای متناظر هرکدام را پس از ساده‌سازی بنویسید؛ سپس، برای هر یک یا چند خانه‌ای از جدول که مقدار ۱ داشته و در هیچ‌یک از موجب‌های اصلی اساسی قرار نگرفته، بزرگترین گروه را پیدا کنید (به نام موجب اصلی[2]) و عبارت ساده شده‌ی متناظر را بنویسید. برای هر یک از چنین خانه‌هایی ممکن است چندین موجب اصلی مختلف پیدا کنید؛ بنابراین،

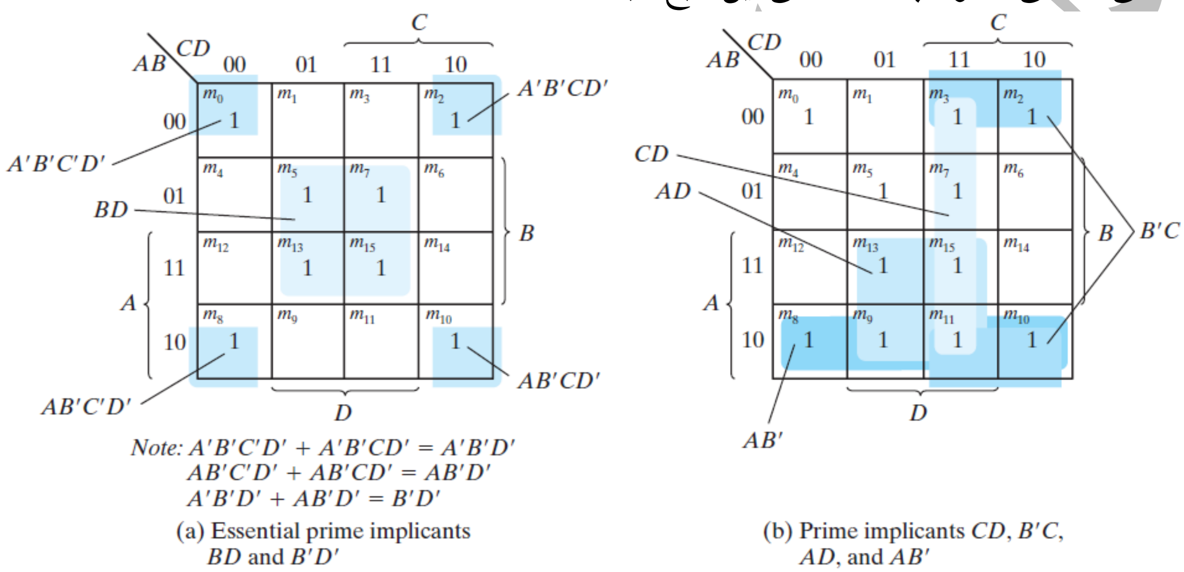
[1] Essential Prime Implicants
[2] Prime Implicant

نتیجه این که عبارتهای مربوط به موجب‌های اصلی اساسی فقط یک بار در عبارت ساده شده‌ی نهایی ظاهر می‌شوند اما عبارتهای متعدد و البته معادل متعددی می‌توان برای موجب های اصلی پیدا کرد که باعث می‌شوند یک تابع داده شده دارای عبارتهای ساده شده‌ی متعددی باشد.

مثال: تابع زیر را در نظر بگیرید.

$$F(A, B, C, D) = \Sigma(0, 2, 3, 5, 7, 8, 9, 10, 11, 13, 15)$$

موجب‌های اصلی اساسی و موجب‌های اصلی این تابع عبارتند از:



(a) Essential prime implicants BD and $B'D'$

(b) Prime implicants CD, $B'C$, AD, and AB'

FIGURE 3.11
Simplification using prime implicants

بنابراین، عبارت ساده شده‌ی تابع داده شده را به روشهای مختلفی مانند زیر می‌توان نوشت:

$$\begin{aligned} F &= BD + B'D' + CD + AD \\ &= BD + B'D' + CD + AB' \\ &= BD + B'D' + B'C + AD \\ &= BD + B'D' + B'C + AB' \end{aligned}$$

در هر یک از چهار عبارت فوق، اولین دو جمله در تمام چهار عبارت، یکسان و مشترک بوده و مربوط به موجب‌های اصلی اساسی هستند.

نقشه‌ی پنج‌ـ‌متغیره و بیشتر

وقتی تعداد متغیرهای تابع از چهار متغیر بیشتر می‌شود، روش گرافیکی جدول کارنو به مراتب مشکل‌تر می‌شود لذا چندان مفید و کاربردی نبوده و لذا این حالتها در این جا در نظر گرفته نمی‌شوند.

ساده‌سازی به کمک ضرب حاصل‌جمع‌ها

تاکنون در روش جدول کارنو، خروجیِ ساده شده به شکل جمع حاصل‌ضرب‌ها بود. می‌توان عبارت ساده شده را به شکل ضرب حاصل‌جمع‌ها نیز به دست آورد. برای این کار کافی است گروه‌بندی‌ها را روی خانه‌هایی که مقدار صفر دارند، انجام دهیم تا بدین ترتیب، به جای تابع F، متمم تابع یعنی F’ را به دست آوریم. حالا به کمک تئوری دمورگان، خود تابع F را به دست می‌آوریم که به شکل ضرب حاصل‌جمع‌ها خواهد بود.

EXAMPLE 3.7

Simplify the following Boolean function into (a) sum-of-products form and (b) product-of-sums form:

$$F(A, B, C, D) = \Sigma(0, 1, 2, 5, 8, 9, 10)$$

The 1's marked in the map of Fig. 3.12 represent all the minterms of the function. The squares marked with 0's represent the minterms not included in *F* and therefore denote the complement of *F*. Combining the squares with 1's gives the simplified function in sum-of-products form:

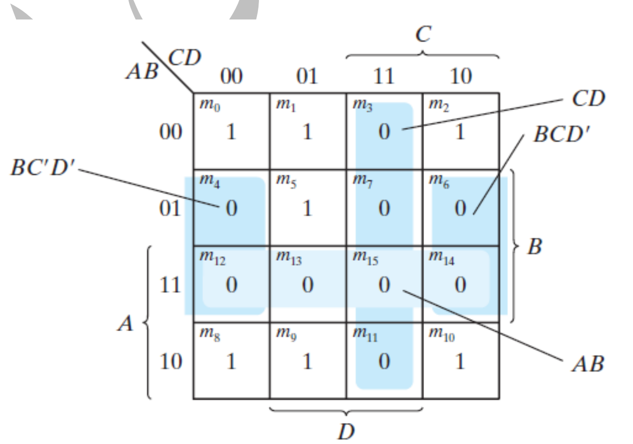
(a) $F = B'D' + B'C' + A'C'D$

If the squares marked with 0's are combined, as shown in the diagram, we obtain the simplified complemented function:

$$F' = AB + CD + BD'$$

Applying DeMorgan's theorem (by taking the dual and complementing each literal as described in Section 2.4), we obtain the simplified function in product-of-sums form:

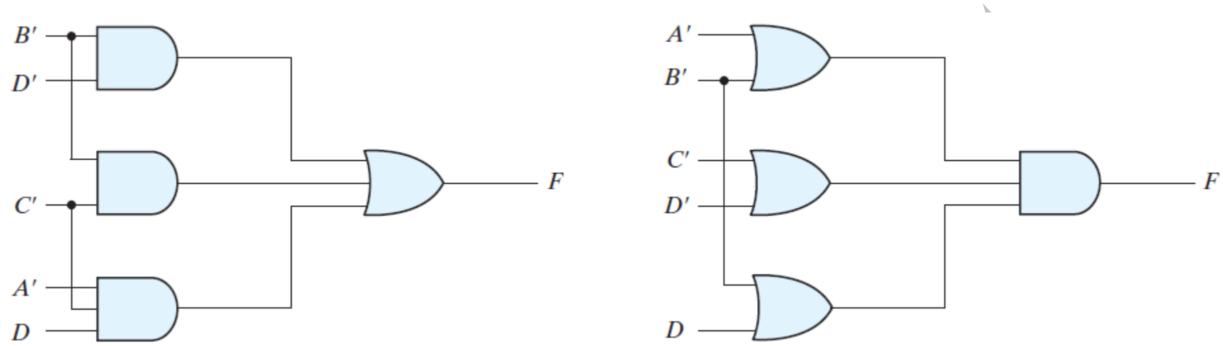
(b) $F = (A' + B')(C' + D')(B' + D)$



Note: $BC'D' + BCD' = BD'$

FIGURE 3.12
Map for Example 3.7, $F(A, B, C, D) = \Sigma(0,1, 2, 5, 8, 9,10) = B'D' + B'C' + A'C'D = (A' + B')(C' + D')(B' + D)$

پیاده‌سازی تابع اخیر به دو شکل ضرب حاصل‌جمع‌ها و جمع حاصل‌ضرب‌ها مطابق شکل زیر است که به هر دو «پیاده‌سازی دو سطحی» گفته شده و یک روال عمومی و کلی برای پیاده‌سازی توابع است (در مداراتی مانند PAL و PLA و GAL).



(a) $F = B'D' + B'C' + A'C'D$ (b) $F = (A' + B')(C' + D')(B' + D)$

FIGURE 3.13
Gate implementations of the function of Example 3.7

ساده‌سازی تابعی که از ابتدا برحسب ماکسترمها توصیف شده است:

مثال: فرض کنید تابع F به صورت زیر داده شده باشد.

Table 3.1
Truth Table of Function F

x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

برای توصیف F برحسب مینترم‌ها از خانه‌هایی که مقدار ۱ دارند و برای توصیف F برحسب ماکسترمها از خانه‌هایی که مقدار صفر دارند، استفاده می‌کنیم. بنابراین:

$$F(x, y, z) = \Sigma(1, 3, 4, 6)$$
$$F(x, y, z) = \Pi(0, 2, 5, 7)$$

جدول کارنوی این تابع به صورت زیر است:

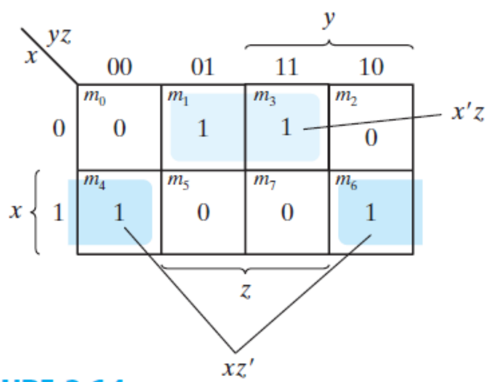


FIGURE 3.14
Map for the function of Table 3.1

اگر ساده‌سازی را برحسب خانه‌هایی که مقدار ۱ دارند انجام دهیم، عبارت ساده شده بر حسب جمع حاصل‌ضرب‌ها خواهد بود:

$$F = x'z + xz'$$

اگر ساده‌سازی را برحسب خانه‌هایی که مقدار صفر دارند انجام دهیم، عبارت ساده شده بر حسب ضرب حاصل جمع‌ها خواهد بود:

$$F' = xz + x'z'$$
$$F = (x' + z')(x + z)$$

اگر از ابتدا تابع را به صورت ضرب حاصل‌جمع‌ها داده باشند، ابتدا با متمم‌گیری متمم آن تابع را برحسب جمع حاصل‌ضرب‌ها به دست می‌آوریم. برای مثال اگر داشته باشیم

$$F = (A' + B' + C')(B + D)$$

آن‌گاه با متمم‌گیری داریم:

$$F' = ABC + B'D'$$

حالا با تشکیل جدول کارنو، مینترم‌های F’ را پیدا کرده و در خانه‌های مربوطه به جای ۱ ، مقدار صفر قرار می‌دهیم (زیرا به جای F، متمم آن، F’ را در دست داریم). حالا بقیه‌ی مراحل ساده‌سازی مثل همیشه است.

حالات بی‌اهمیت (Don’t Care)

در برخی کاربردها، مقدار تابع به ازاء تمام ترکیبات ممکن متغیرهای ورودی مشخص و معلوم نیست (توابع ناکامل[1])؛ برای مثال، در کاربردکد دودویی چهاربیتی برای ارقام دهدهی چون تنها ارقام دهدهی 0 تا 9 مجاز هستند پس شش ترکیب ممکن از کد دودویی چهاربیتی، بلااستفاده خواهد ماند (از 1010 تا 1111). به این حالتها، حالتهای «بی‌اهمیت» گفته شده و در ساده‌تر کردن عبارت بولی تابع استفاده می‌شوند.

[1] Incompletely specified functions

EXAMPLE 3.8

Simplify the Boolean function

$$F(w, x, y, z) = \Sigma(1, 3, 7, 11, 15)$$

which has the don't-care conditions

$$d(w, x, y, z) = \Sigma(0, 2, 5)$$

The minterms of F are the variable combinations that make the function equal to 1. The minterms of d are the don't-care minterms that may be assigned either 0 or 1. The map simplification is shown in Fig. 3.15. The minterms of F are marked by 1's, those of d are marked by X's, and the remaining squares are filled with 0's. To get the simplified expression in sum-of-products form, we must include all five 1's in the map, but we may or may not include any of the X's, depending on the way the function is simplified. The term yz covers the four minterms in the third column. The remaining minterm, m_1, can be combined with minterm m_3 to give the three-literal term $w'x'z$. However, by including one or two adjacent X's we can combine four adjacent squares to give a two-literal term. In Fig. 3.15(a), don't-care minterms 0 and 2 are included with the 1's, resulting in the simplified function

$$F = yz + w'x'$$

In Fig. 3.15(b), don't-care minterm 5 is included with the 1's, and the simplified function is now

$$F = yz + w'z$$

Either one of the preceding two expressions satisfies the conditions stated for this example.

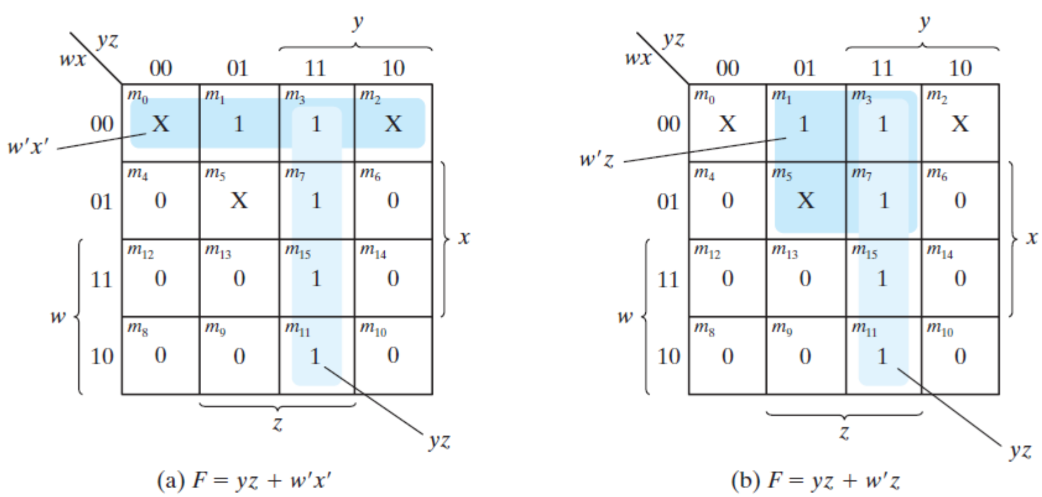


FIGURE 3.15
Example with don't-care conditions

پیاده‌سازی با گیت‌های NAND و NOR

مدارهای دیجیتال اغلب به جای AND و OR با گیت‌های NAND و NOR ساخته می‌شوند زیرا ساخت این گیت‌ها با قطعات الکترونیکی ساده‌تر بوده و به عنوان گیت‌های پایه در تمام خانواده‌های ICهای دیجیتال به کار می‌روند.

گیت NAND یک گیت جامع یا یونیورسال است به این معنا که با استفاده از تنها گیت‌های NAND می‌توان هر مدار دیجیتالی را پیاده‌سازی کرد. برای اثبات این امر کافی است نشان داده شود که هر یک از گیت‌های AND و NOT و OR را می‌توان تنها به کمک استفاده از گیت NAND پیاده‌سازی کرد.

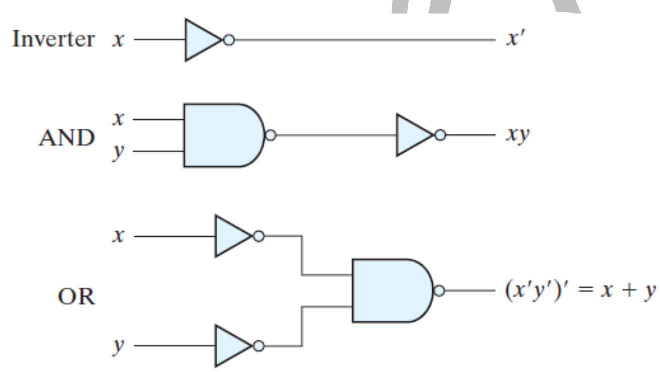


FIGURE 3.16
Logic operations with NAND gates

دو نماد یا سمبل گرافیکی گیت NAND

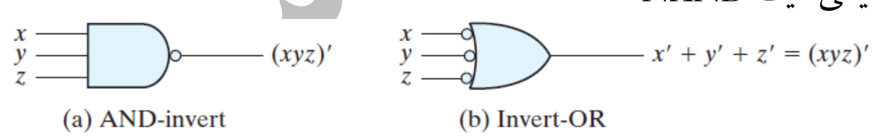


FIGURE 3.17
Two graphic symbols for a three-input NAND gate

اساس پیاده‌سازی برحسب گیت NAND در شکل زیر نشان داده شده است. در شکل زیر، هر سه مدار تابع زیر را پیاده‌سازی می‌کنند:

$$F = AB + CD$$

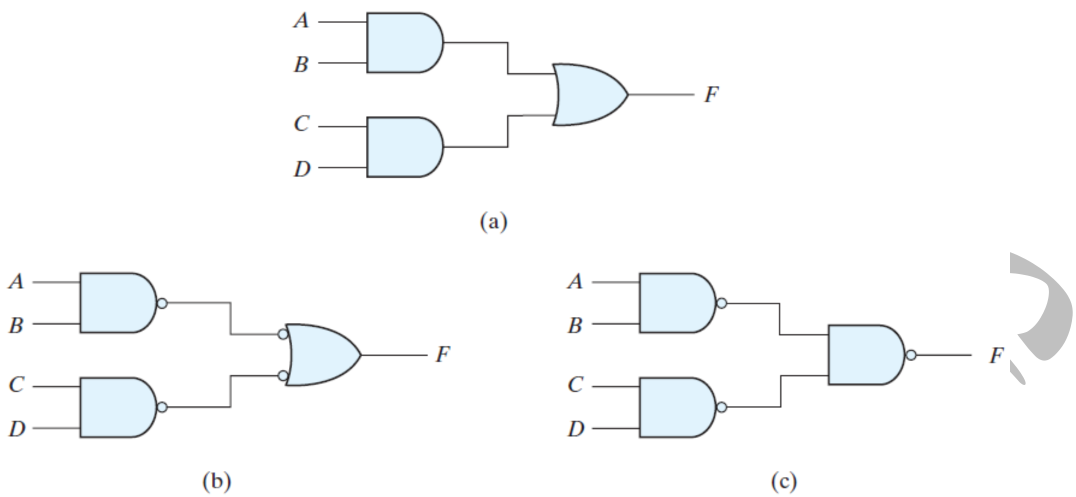


FIGURE 3.18
Three ways to implement $F = AB + CD$

مثال

EXAMPLE 3.9

Implement the following Boolean function with NAND gates:

$$F(x, y, z) = (1, 2, 3, 4, 5, 7)$$

The first step is to simplify the function into sum-of-products form. This is done by means of the map of Fig. 3.19(a), from which the simplified function is obtained:

$$F = xy' + x'y + z$$

The two-level NAND implementation is shown in Fig. 3.19(b) in mixed notation. Note that input z must have a one-input NAND gate (an inverter) to compensate for the bubble in the second-level gate. An alternative way of drawing the logic diagram is given in Fig. 3.19(c). Here, all the NAND gates are drawn with the same graphic symbol. The inverter with input z has been removed, but the input variable is complemented and denoted by z'.

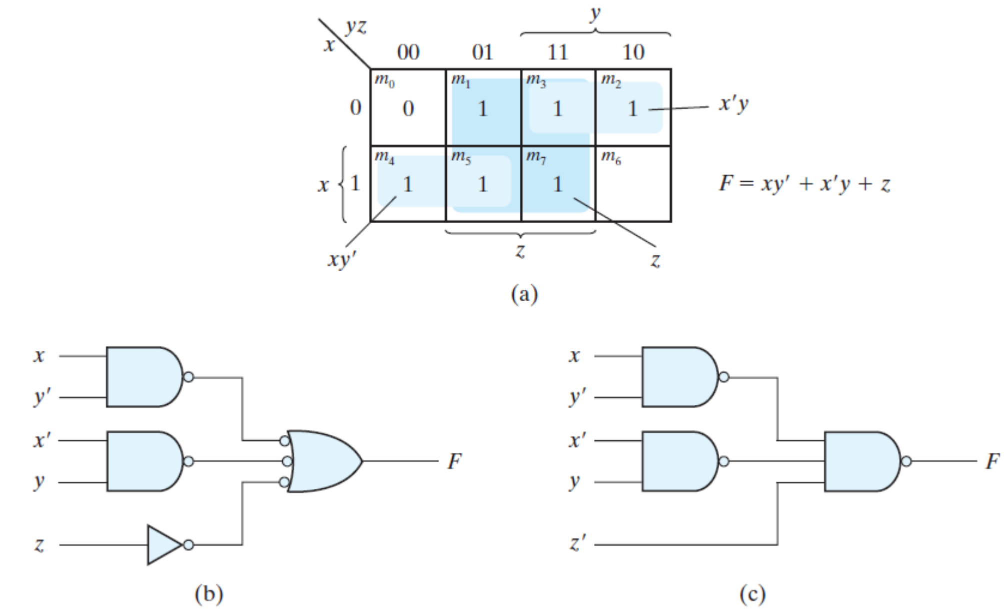


FIGURE 3.19
Solution to Example 3.9

پیاده‌سازی با گیت NOR

گیت NOR دوگان گیت NAND است. این گیت نیز یک گیت جامع (یونیورسال) است (شکل زیر).

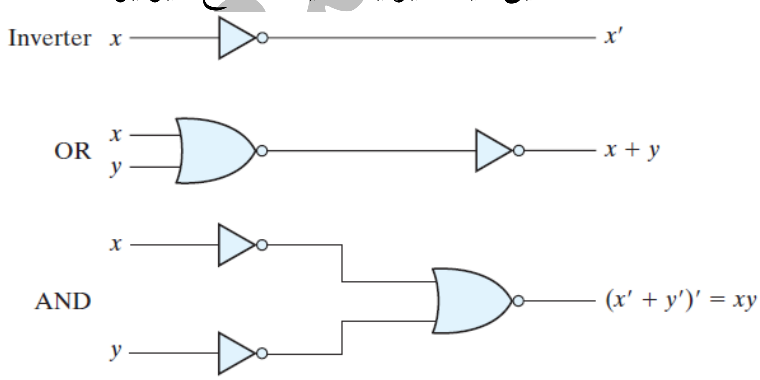


FIGURE 3.22
Logic operations with NOR gates

نمادها یا سمبل‌های گرافیکی گیت NOR

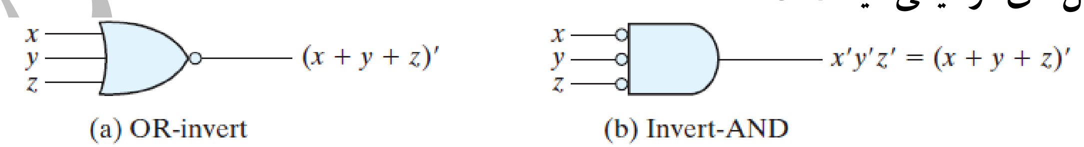


FIGURE 3.23
Two graphic symbols for the NOR gate

پیاده‌سازی دو طبقه با گیت NOR

بهتر است تابع به صورت ضرب حاصل‌جمع‌ها ساده شود تا پیاده‌سازی آن راحت‌تر شود. به خاطر دارید که عبارت ضرب حاصل‌جمع‌های ساده شده به کمک جدول کارنو با گروه‌بندی صفرها و متمم کردن نتیجه به دست می‌آید.

برای مثال، اگر تابع زیر را در نظر بگیریم،

$$F = (A + B)(C + D)E$$

به راحتی، پیاده‌سازی آن به کمک گیت NAND به صورت زیر قابل انجام است:

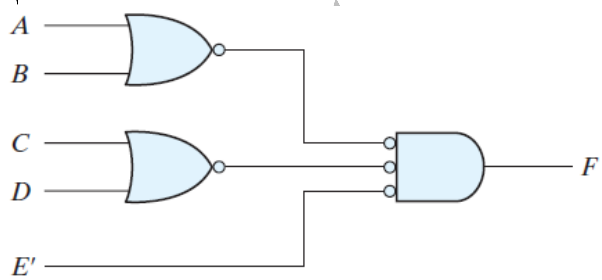
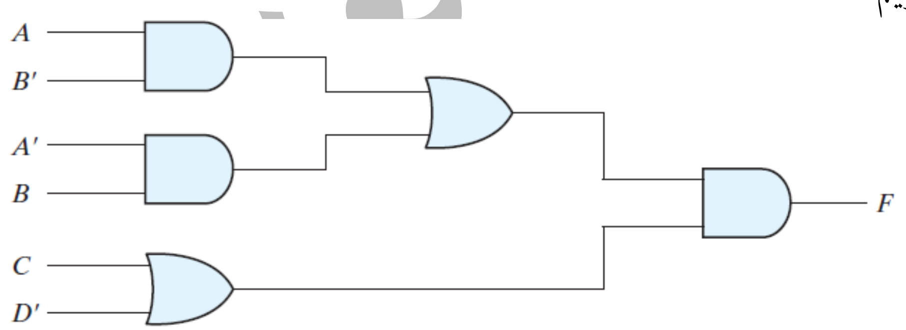


FIGURE 3.24
Implementing $F = (A + B)(C + D)E$

اما اگر تابع به شکل جمع حاصل‌ضرب‌ها داده شود، نیز می‌توان آن را به کمک گیت NOR پیاده‌سازی کرد. برای مثال اگر تابع زیر را در نظر بگیریم:



به صورت زیر می‌توان این تابع را بر حسب گیت NAND توصیف کرد:

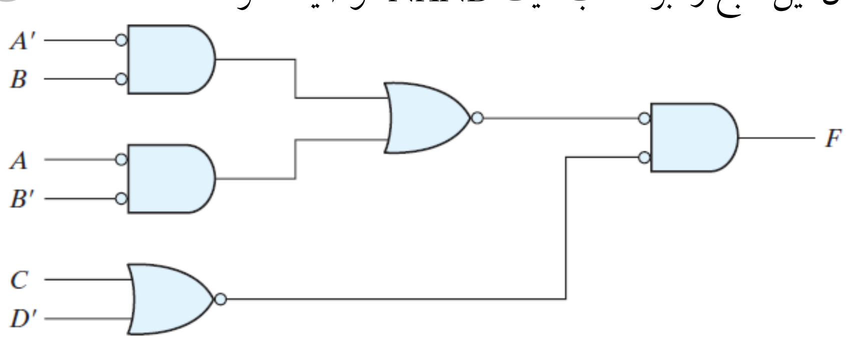


FIGURE 3.25
Implementing $F = (AB' + A'B)(C + D')$ with NOR gates

تابع OR انحصاری (XOR)

عملگر یا گیت OR انحصاری (XOR) تابع زیر را پیاده می‌کند:

$$x \oplus y = xy' + x'y$$

تابع فوق تنها زمانی مقدار ۱ می‌گیرد که فقط یکی از ورودی‌ها مقدار ۱ داشته باشد (به عبارت دیگر، اگر هر دو ورودی مقدار ۱ یا صفر داشته باشند، خروجی تابع فوق برابر صفر می‌شود)

عملگر یا گیت NOR انحصاری (XNOR) تابع زیر را پیاده‌سازی می‌کند:

$$(x \oplus y)' = xy + x'y'$$

تابع فوق زمانی مقدار ۱ می‌گیرد که هر دو ورودی همزمان و مشترکاً مقدار ۱ یا مقدار صفر داشته باشند.

دو تابع فوق متمم یکدیگر هستند:

$$(x \oplus y)' = (xy' + x'y)' = (x' + y)(x + y') = xy + x'y'$$

برخی ویژگی‌های دیگر تابع OR انحصاری به قرار زیر است:

$$x \oplus 0 = x$$
$$x \oplus 1 = x'$$
$$x \oplus x = 0$$
$$x \oplus x' = 1$$
$$x \oplus y' = x' \oplus y = (x \oplus y)'$$
$$A \oplus B = B \oplus A$$
$$(A \oplus B) \oplus C = A \oplus (B \oplus C) = A \oplus B \oplus C$$

ملاحظه می‌شود که خواص جابجایی و شرکت‌پذیری نیز برقرارند. برطبق خاصیت شرکت‌پذیری وجود یا عدم وجود پرانتز در عملگر XOR اهمیتی ندارد.

تابع XOR چند ورودی، تابعی «فرد» است به این معنا که اگر تعداد ۱ ها در ورودی‌های این تابع عددی فرد باشد، خروجی تابع مقدار ۱ و در غیر این صورت مقدار صفر به خود می‌گیرد. بنابراین، تابع XOR در تولید توازن زوج استفاده می‌شود. برای تشخیص خطا نیز از همین تابع می‌توان استفاده کرد.

جدول صحت مدار تولید توازن زوج:

Table 3.3
Even-Parity-Generator Truth Table

Three-Bit Message			Parity Bit
x	*y*	*z*	*P*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

جدول صحت چک کننده‌ی توازن زوج:

Table 3.4
Even-Parity-Checker Truth Table

Four Bits Received				Parity Error Check
x	*y*	*z*	*P*	*C*
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

مدارات تولید و چک کننده‌ی توازن زوج:

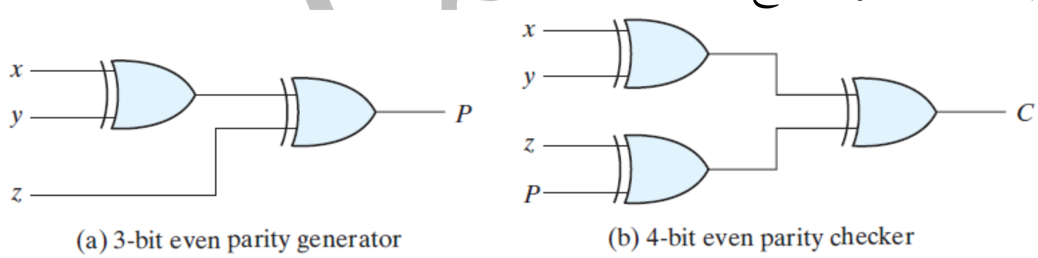


FIGURE 3.34
Logic diagram of a parity generator and checker

براي سلامتي رهبر انقلاب و تعجيل در ظهور حضرت ولي عصر (عج)

صلوات