Mnożenie dwóch liczb binarnych.

				1	1	0	1
x				1	0	1	1
				1	1	0	1
			1	1	0	1	
		0	0	0	0		
	1	1	0	1			
1	0	0	0	1	1	1	1

$(1101)_2 = (13)_{10}$

$(13)_{10} \text{ x } (11)_{10} = (143)_{10} = (10001111)_2$

$(1011)_2 = (11)_{10}$

Mnożenie dwóch liczb binarnych.

$$
\begin{array}{c|c|c|c|c|c|c|c|c|}
 & & & & & 1 & 1 & 1 & 1 \\
\times & & & & & 1 & 1 & 0 & 1 \\
\hline
 & & & & & 1 & 1 & 1 & 1 \\
 & & & & 0 & 0 & 0 & 0 & \\
 & & & 1 & 1 & 1 & 1 & & \\
 & & 1 & 1 & 1 & 1 & & & \\
\hline
 1 & & 1 & 0 & 0 & 0 & 0 & 1 & 1 \\
\end{array}
$$

Dzielenie liczb binarnych.

```
        1 1 0 1
----------------
1 0 0 1 0 0 0 1 / 1 0 1 1
  1 0 1 1
----------
    1 1 1 0
    1 0 1 1
    --------
        1 1 0 1
        1 0 1 1
        --------
            1 0
```

$(10010001)_2 = (145)_{10}$

$(1011)_2 = (11)_{10}$

$(145)_{10} / (11)_{10} = (13)_{10} + (2)_{10}$

Odejmowanie dwóch liczb n-bitowych $M - N$, w systemie o podstawie p może być wykonane w następujący sposób:

Do odjemnej M należy dodać uzupełnienie do p odjemnika N:

$$M + p^n - N = M - N + p^n$$

Jeżeli $M \geq N$, w wyrażeniu $M + p^n - N = M - N + p^n$ należy odjąć p^n, co daje wynik $M - N$.

Niech $M = (1987)_{10}$ i $N = (1958)_{10}$. W tym przypadku $p = 10$, $n = 4$, $M > N$, a różnicę ($M - N$) otrzymujemy w następujący sposób:

$$\begin{array}{lr|l}
M = & & 1987 \\
p^n - N = 10^4 - 1958 = & + & 8042 \\
\hline
\text{suma=} & 1 & 0029 \\
p^n = 10^4 & -1 & 0000 \\
\hline
\text{wynik=} & 0 & 0029
\end{array}$$

Niech $M = (1958)_{10}$ i $N = (1958)_{10}$. W tym przypadku $p = 10$, $n = 4$, $M = N$, a różnicę $M - N$ otrzymujemy w następujący sposób:

$$\begin{array}{lr:l}
M = & & 1958 \\
p^n - N = 10^4 - 1958 = & + & 8042 \\
\hline
\text{suma=} & 1 & 0000 \\
p^n = 10^4 & -1 & 0000 \\
\hline
\text{wynik=} & 0 & 0000
\end{array}$$

Odejmowanie dwóch liczb n-bitowych $M - N$, w systemie o podstawie p może być wykonane w następujący sposób:

Do odjemnej M należy dodać uzupełnienie do p odjemnika N:

$$M + p^n - N = M - N + p^n$$

Jeżeli $M \geq N$, w wyrażeniu $M + p^n - N = M - N + p^n$ należy odjąć p^n, co daje wynik $M - N$.

Przypadek: $M < N$. Wyrażenie $M + p^n - N$ można zapisać w postaci:

$$M + p^n - N = p^n - (N - M)$$

będącej uzupełnieniem do p różnicy $(N - M)$.

Biorąc uzupełnienie do p wyrażenia $p^n - (N - M)$ i poprzedzając je znakiem minus otrzymujemy:

$$-\{p^n - [p^n - (N - M)]\} = -(N - M) = M - N$$

Niech $M = (1958)_{10}$ i $N = (1987)_{10}$. W tym przypadku $p = 10$, $n = 4$, $M < N$, a różnicę $(M - N)$ otrzymujemy w następujący sposób:

$$
\begin{array}{lr}
M = & 1958 \\
p^n - N = 10^4 - 1987 = & +8013 \\
\hline
\text{suma}= & 9971
\end{array}
$$

$$-(p^n - \text{suma}) = -(10^4 - 9971) = -29$$

Niech $M = (11001)_2$ i $N = (1010)_2$. W tym przypadku $p = 2$, $n = 5$, $M > N$, a różnicę $(M - N)$ otrzymujemy w następujący sposób:

$$
\begin{array}{lr|l}
M = & & 11001 \\
p^n - N = 2^5 - 1010 = & + & 10110 \\
\hline
\text{suma}= & 1 & 01111 \\
p^n = 2^5 & -1 & 00000 \\
\hline
\text{wynik}= & 0 & 01111
\end{array}
$$

Niech $M = (1010)_2$ i $N = (1010)_2$. W tym przypadku $p = 2$, $n = 5$, $M = N$, a różnicę $(M - N)$ otrzymujemy w następujący sposób:

$$
\begin{array}{lr}
M = & 1 \vdots 1010 \\
p^n - N = 2^4 - 1010 = & +\vdots 0110 \\
\hline
\text{suma}= & 1 \vdots 0000 \\
p^n = 2^4 & -1 \vdots 0000 \\
\hline
\text{wynik}= & 0 \vdots 0000
\end{array}
$$

Niech $M = (1010)_2$ i $N = (11001)_2$. W tym przypadku $p = 2$, $n = 5$, $M < N$, a różnicę $(M - N)$ otrzymujemy w następujący sposób:

$$
\begin{array}{lr}
M = & 01010 \\
p^n - N = 2^5 - 11001 = & +00111 \\
\hline
\text{suma}= & 10001
\end{array}
$$

$$-(p^n - \text{suma}) = -(2^5 - 10001) = -01111$$

Jeżeli dodając dwie *n*-bitowe liczby bez znaku otrzymujemy wynik na n + 1 bitach, to mówimy, że wystąpił nadmiar (przepełnienie) Pojęcie nadmiaru (*overflow*) wyjaśnimy na następującym przykładzie:

$$\begin{array}{r r|l}
150 & & 1\,0\,0\,1\,0\,1\,1\,0 \\
+\,190 & & 1\,0\,1\,1\,1\,1\,1\,0 \\
\hline
340 & 1 & 0\,1\,0\,1\,0\,1\,0\,0
\end{array}$$

W przykładzie tym przyjęty format danych obejmuje 8 bitów (liczby bez znaku). Zakres liczb bez znaku zapisanych na ośmiu bitach obejmuje liczby od 0 do 255. Wynik dodawania 150 + 190 = 340 wykracza poza ten zakres. Bit przeniesienia z najbardziej znaczącej pozycji wykracza poza przyjęty format danych i jest bitem nadmiaru.

W przypadku liczb ze znakiem nadmiar może wystąpić tylko wtedy, kiedy dodajemy dwie liczby dodatnie lub ujemne. Ilustrują to następujące przykłady:

50		0	0 1 1 0 0 1 0
+ 90		0	1 0 1 1 0 1 0
140		1	0 0 0 1 1 0 0

- 50		1	1 0 0 1 1 1 0
- 90		1	0 1 0 0 1 1 0
- 140	1	0	1 1 1 0 1 0 0

W przykładzie tym przyjęty format danych obejmuje 8 bitów (liczby ze znakiem). Najbardziej znaczący bit jest bitem znaku. Zakres liczb ze znakiem zapisanych na ośmiu bitach obejmuje liczby od +127 do −128. Widzimy, że zapisany na ośmiu bitach wynik dodawania liczb dodatnich 50 + 90 = 140 ma znak ujemny (1 na bicie znaku). Błąd ten spowodowany jest tym, że liczba 50 + 90 = 140 wykracza poza zakres od +127 do −128 (nadmiar).

Widzimy również, że zapisany na ośmiu bitach wynik dodawania liczb ujemnych −50 + (−90) = −140 ma znak dodatni (0 na bicie znaku). Błąd ten spowodowany jest tym, że liczba −50 + (−90) = −140 wykracza poza zakres od +127 do −128 (nadmiar).

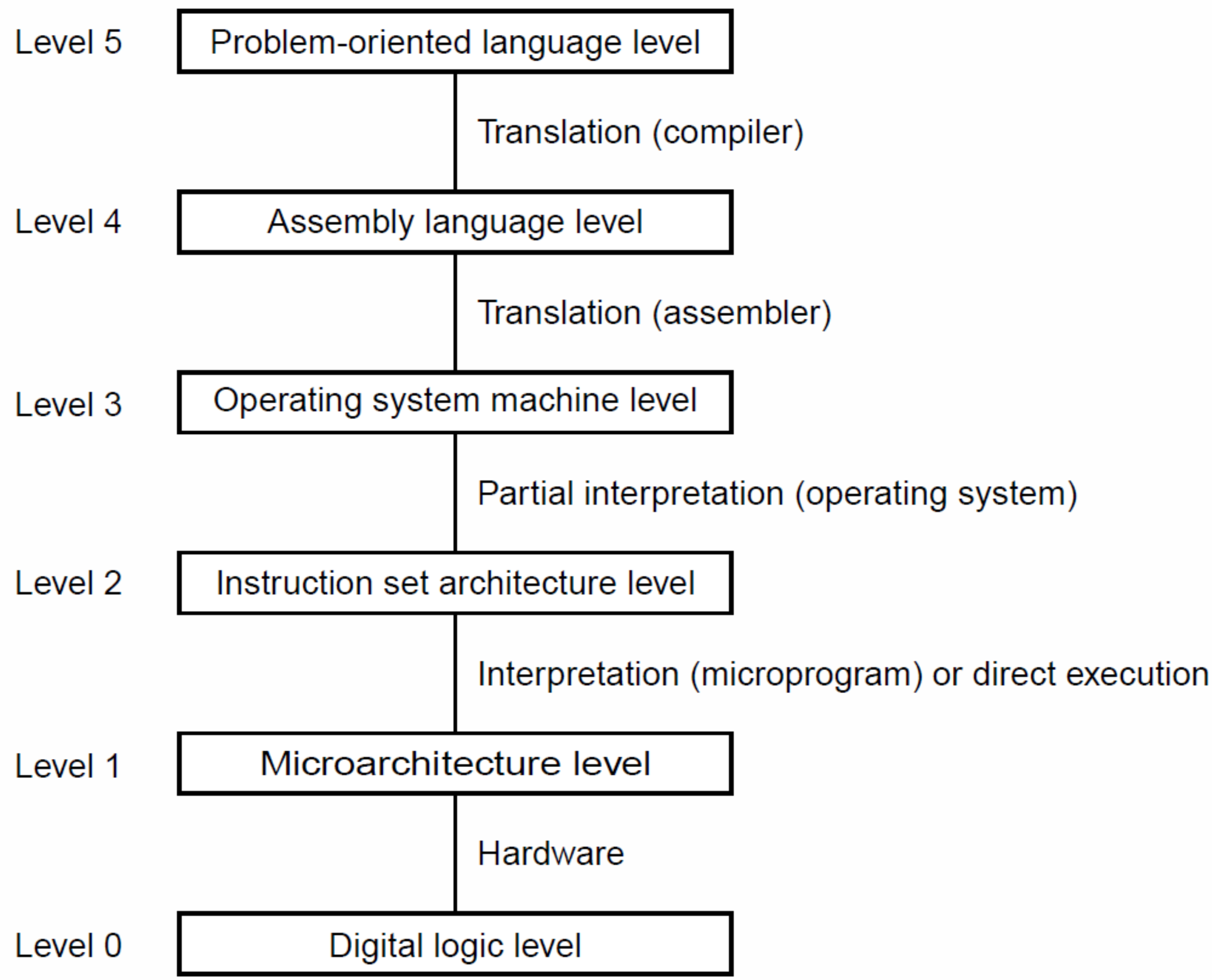


Figure 1-2. A six-level computer. The support method for each level is supported is indicated below it (along with the name of the supporting program).

A. Tanenbaum, Structured Computer Organization, Sixth edition, Pearson, 2013.

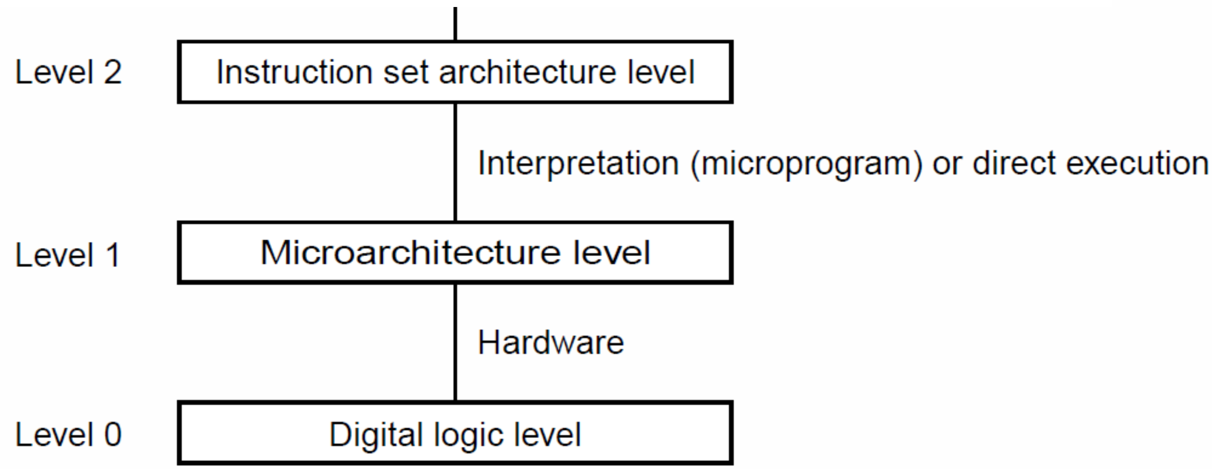


Figure 1-2. A six-level computer. The support method for each level is supported is indicated below it (along with the name of the supporting program).

A. Tanenbaum, Structured Computer Organization, Sixth edition, Pearson, 2013.

Morris Mano, Charles R. Kime, Tom Martin, Logic and computer design fundamentals, Pearson, Fiftth edition, 2016.

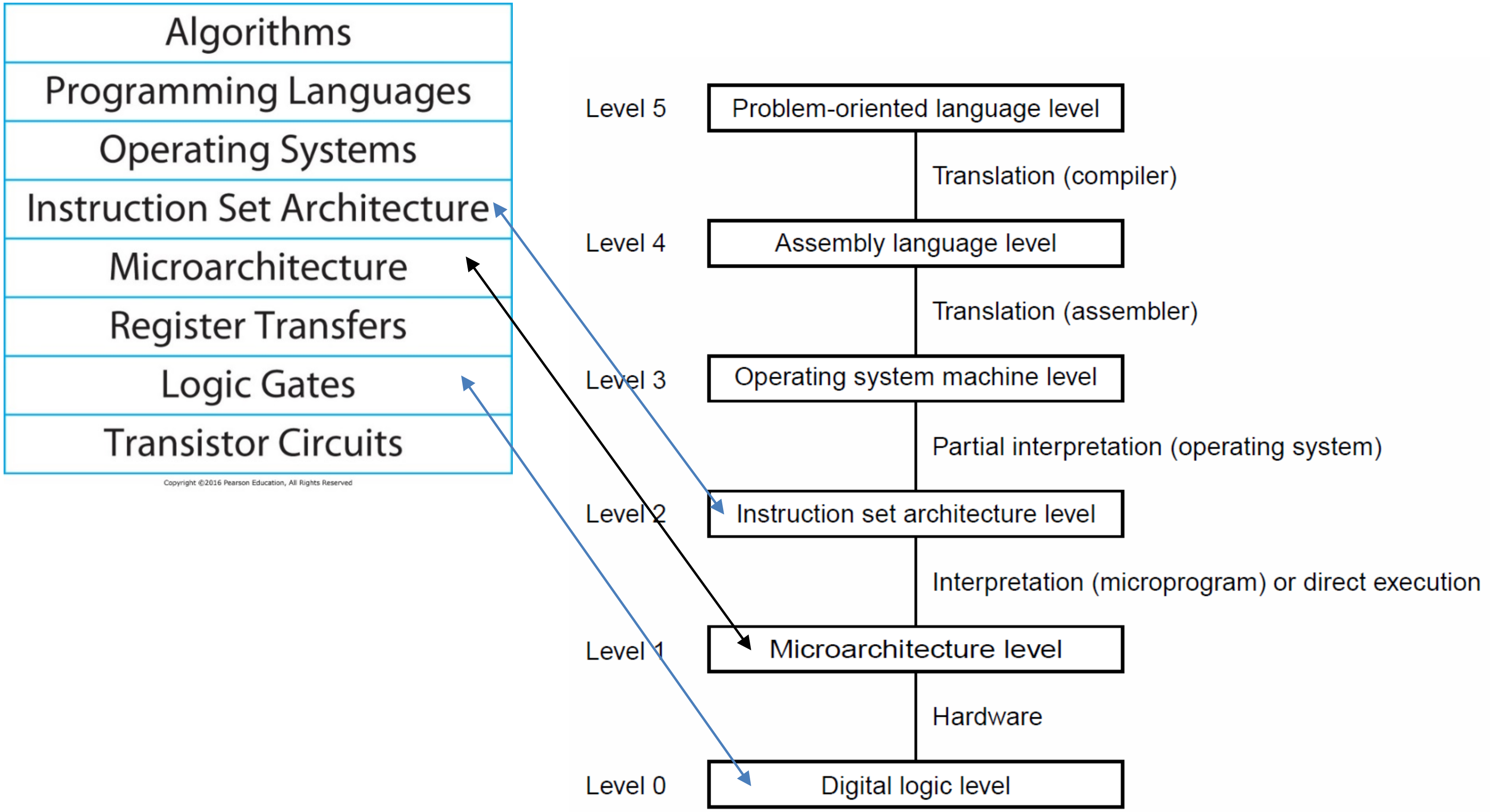
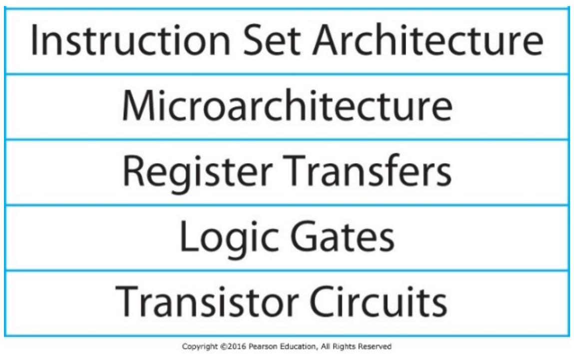


Figure 1-2. A six-level computer. The support method for each level is supported is indicated below it (along with the name of the supporting program).

A. Tanenbaum, Structured Computer Organization, Sixth edition, Pearson, 2013.

Morris Mano, Charles R. Kime, Tom Martin, Logic and computer design fundamentals, Pearson, Fiftth edition, 2016.



Dwuelementowa algebra Boole’a.

Dwuelementowa algebra Boole’a jest działem matematyki wykorzystywanym do projektowania i analizy układów cyfrowych. Zmienne boolowskie (logiczne) mogą przyjmować wartości tylko ze zbioru {0, 1}. W algebrze Boole’a do przedstawienia iloczynu logicznego (AND) zmiennych X i Y będziemy stosować zapis $X \cdot Y = XY$. Do przedstawienia sumy logicznej (OR) zmiennych X i Y będziemy stosować zapis $X + Y$. Negację zmiennej X będziemy zapisywać jako $\overline{X}$ lub X’.

Poniżej są przedstawione (za pomocą tablicy prawdy) definicje następujących funkcji logicznych: AND, OR i NOT.

AND

X	Y	$X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

OR

X	Y	$X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

NOT

X	$\overline{X}$
0	1
1	0

TABLE 2-1
Truth Tables for the Three Basic Logical Operations

AND			OR			NOT	
X	Y	$Z = X \cdot Y$	X	Y	$Z = X + Y$	X	$Z = \overline{X}$
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

M. Morris Mano, Charles R. Kime, Tom Martin, Logic and computer design fundamentals, Pearson, Fiftth edition, 2016.

AND			OR			NOT	
x	y	$x \cdot y$	x	y	$x + y$	x	x'
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

M. Morris Mano, Michael D. Ciletti, Digital Design, With an Introduction to the Verilog HDL, VHDL, and SystemVerilog, Pearson, 6th Edition, 2018.

1.	$x + 0 = x$	2.	$x \cdot 1 = x$
3.	$x + 1 = 1$	4.	$x \cdot 0 = 0$
5.	$x + x = x$	6.	$x \cdot x = x$
7.	$x + \overline{x} = 1$	8.	$x \cdot \overline{x} = 0$
9.	$\overline{\overline{x}} = x$		
10.	$x + y = y + x$	11.	$x \cdot y = y \cdot x$
12.	$x + (y + z) = (x + y) + z$	13.	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$
14.	$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$	15.	$x + (y \cdot z) = (x + y) \cdot (x + z)$
16.	$\overline{x + y} = \overline{x} \cdot \overline{y}$	17.	$\overline{x \cdot y} = \overline{x} + \overline{y}$

The duality principle

1.	$x + 0 = x$	2.	$x \cdot 1 = x$
3.	$x + 1 = 1$	4.	$x \cdot 0 = 0$
5.	$x + x = x$	6.	$x \cdot x = x$
7.	$x + \overline{x} = 1$	8.	$x \cdot \overline{x} = 0$
9.	$\overline{\overline{x}} = x$		
10.	$x + y = y + x$	11.	$x \cdot y = y \cdot x$
12.	$x + (y + z) = (x + y) + z$	13.	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$
14.	$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$	15.	$x + (y \cdot z) = (x + y) \cdot (x + z)$
16.	$\overline{x + y} = \overline{x} \cdot \overline{y}$	17.	$\overline{x \cdot y} = \overline{x} + \overline{y}$

Prawa de Morgana (pozycje:16 i 17 w tablicy powyżej).

$$\overline{x_1 + x_2 + \ldots + x_n} = \overline{x}_1 \cdot \overline{x}_2 \cdot \cdots \cdot \overline{x}_n$$

$$\overline{x_1 \cdot x_2 \cdot \cdots \cdot x_n} = \overline{x}_1 + \overline{x}_2 + \ldots + \overline{x}_n$$

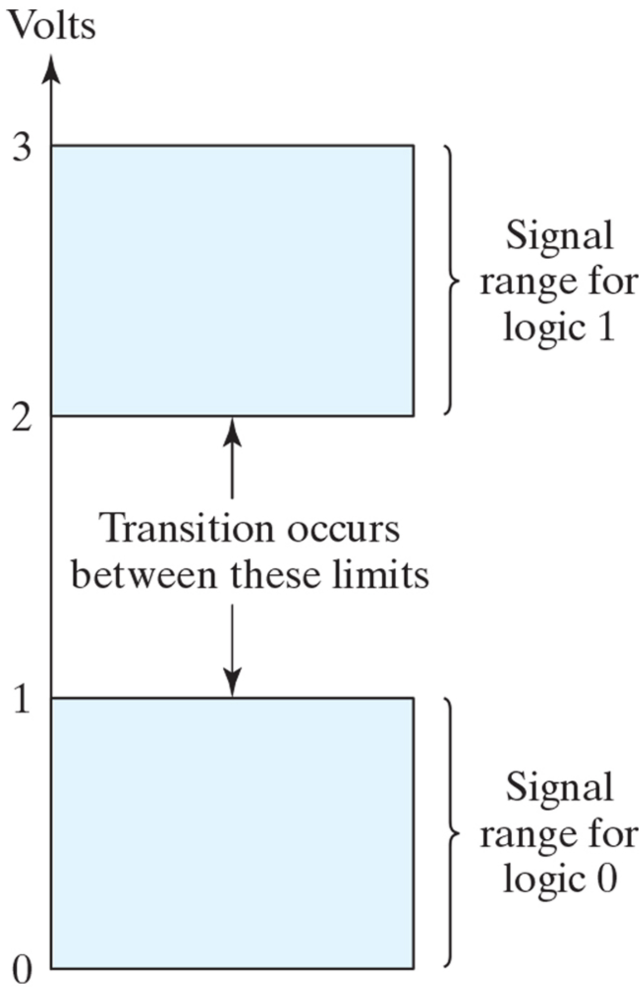
1.	$x + 0 = x$	2.	$x \cdot 1 = x$
3.	$x + 1 = 1$	4.	$x \cdot 0 = 0$
5.	$x + x = x$	6.	$x \cdot x = x$
7.	$x + \overline{x} = 1$	8.	$x \cdot \overline{x} = 0$
9.	$\overline{\overline{x}} = x$		
10.	$x + y = y + x$	11.	$x \cdot y = y \cdot x$
12.	$x + (y + z) = (x + y) + z$	13.	$x \cdot (y \cdot z) = (x \cdot y) \cdot z$
14.	$x \cdot (y + z) = (x \cdot y) + (x \cdot z)$	15.	$x + (y \cdot z) = (x + y) \cdot (x + z)$
16.	$\overline{x + y} = \overline{x} \cdot \overline{y}$	17.	$\overline{x \cdot y} = \overline{x} + \overline{y}$

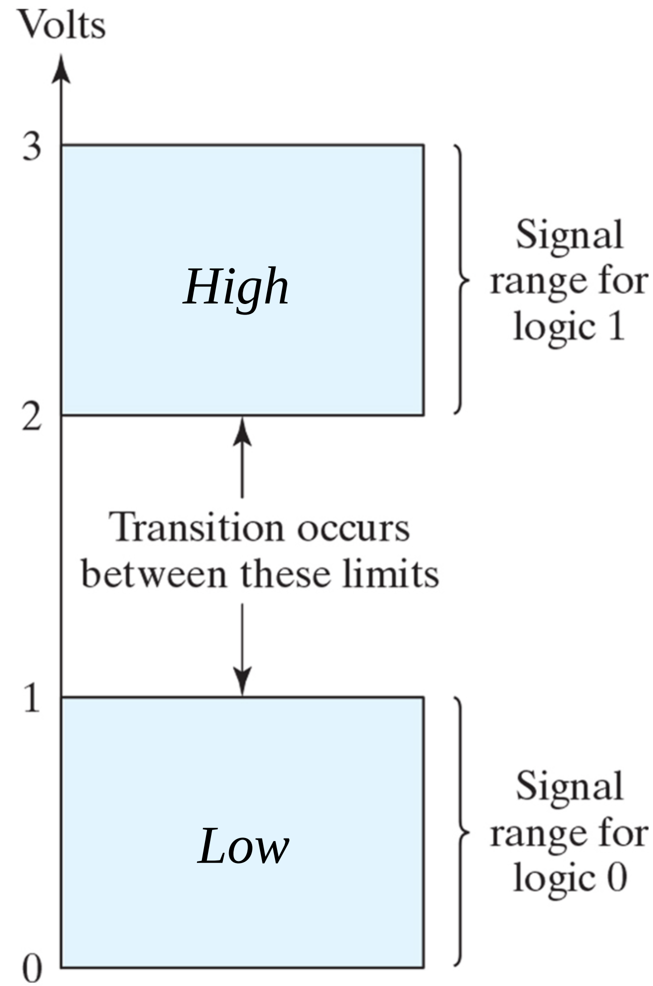
TABLE 2-6
Basic Identities of Boolean Algebra Podstawowe tożsamości algebry boolowskiej

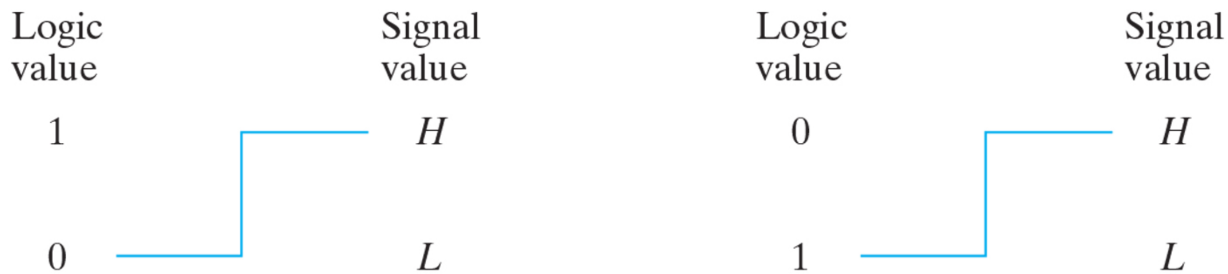
1.	$X + 0 = X$	2.	$X \cdot 1 = X$		
3.	$X + 1 = 1$	4.	$X \cdot 0 = 0$		
5.	$X + X = X$	6.	$X \cdot X = X$		
7.	$X + \overline{X} = 1$	8.	$X \cdot \overline{X} = 0$		
9.	$\overline{\overline{X}} = X$				
10.	$X + Y = Y + X$	11.	$XY = YX$	Commutative	Przemienność
12.	$X + (Y + Z) = (X + Y) + Z$	13.	$X(YZ) = (XY)Z$	Associative	Łączność
14.	$X(Y + Z) = XY + XZ$	15.	$X + YZ = (X + Y)(X + Z)$	Distributive	Rozdzielność
16.	$\overline{X + Y} = \overline{X} \cdot \overline{Y}$	17.	$\overline{X \cdot Y} = \overline{X} + \overline{Y}$	DeMorgan's	Prawa de Morgana

Morris Mano, Charles R. Kime, Tom Martin, Logic and computer design fundamentals, Pearson, Fiftth edition, 2016.

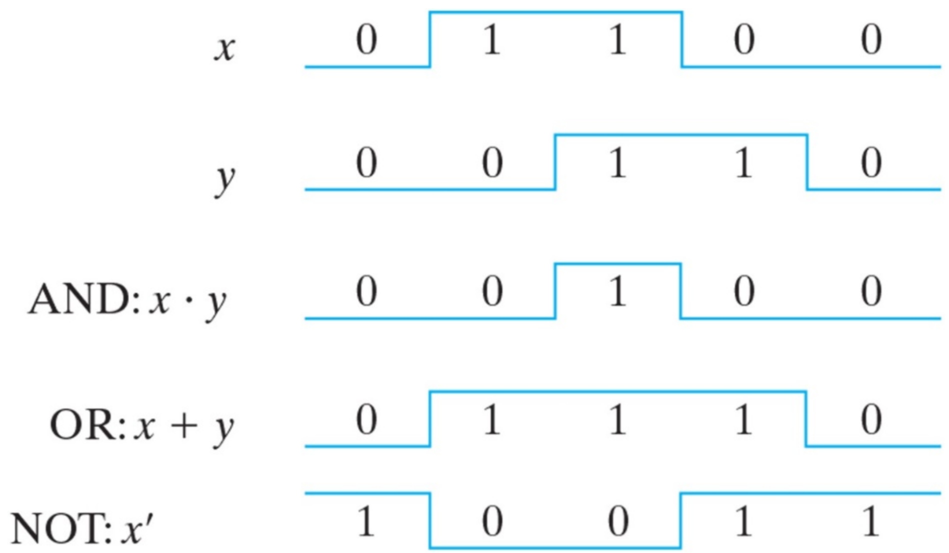
M. Morris Mano, Michael D. Ciletti, Digital design, Sixth edition, Pearson, 2018.





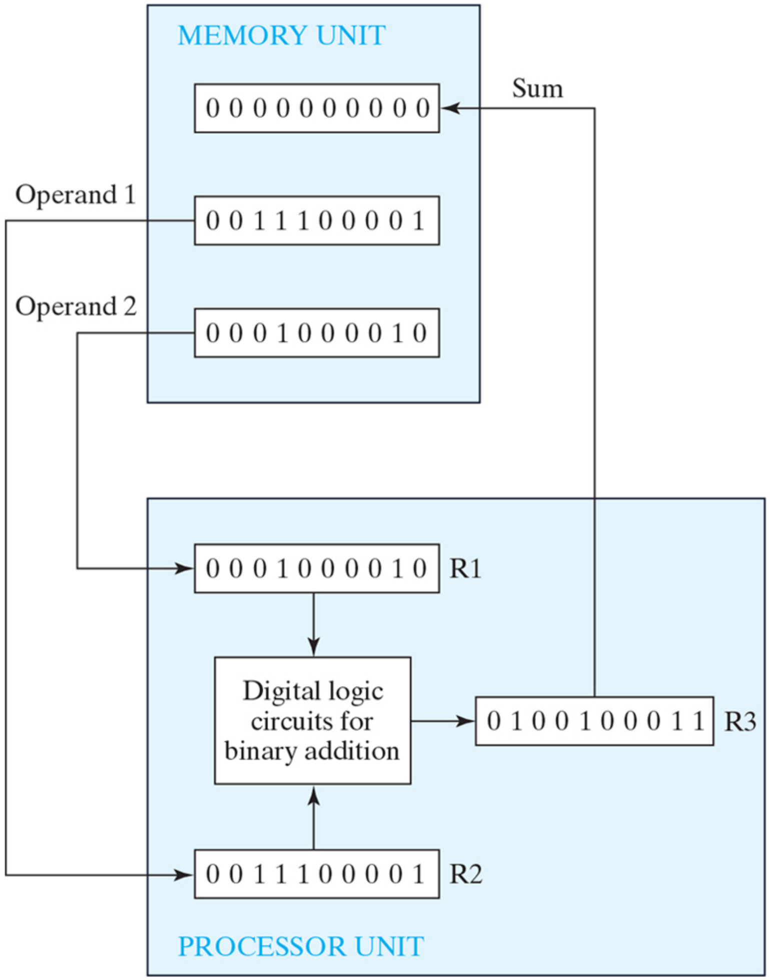


(a) Positive logic (b) Negative logic



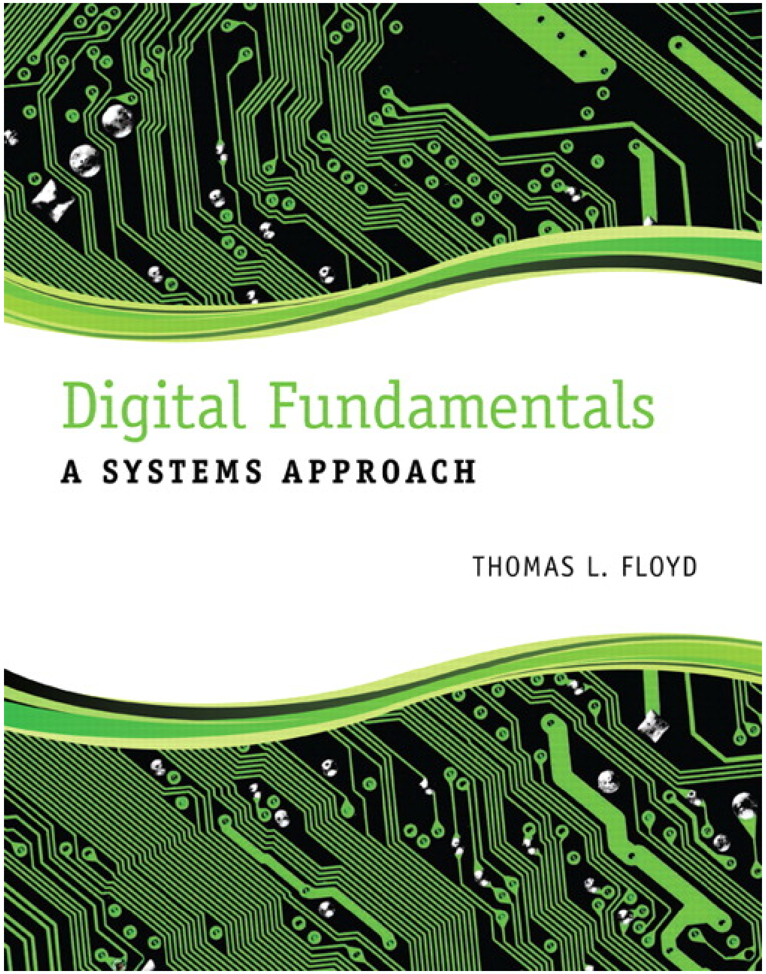
M. Morris Mano, Michael D. Ciletti, Digital design, Sixth edition, Pearson, 2018.

Przykład przetwarzania binarnego.



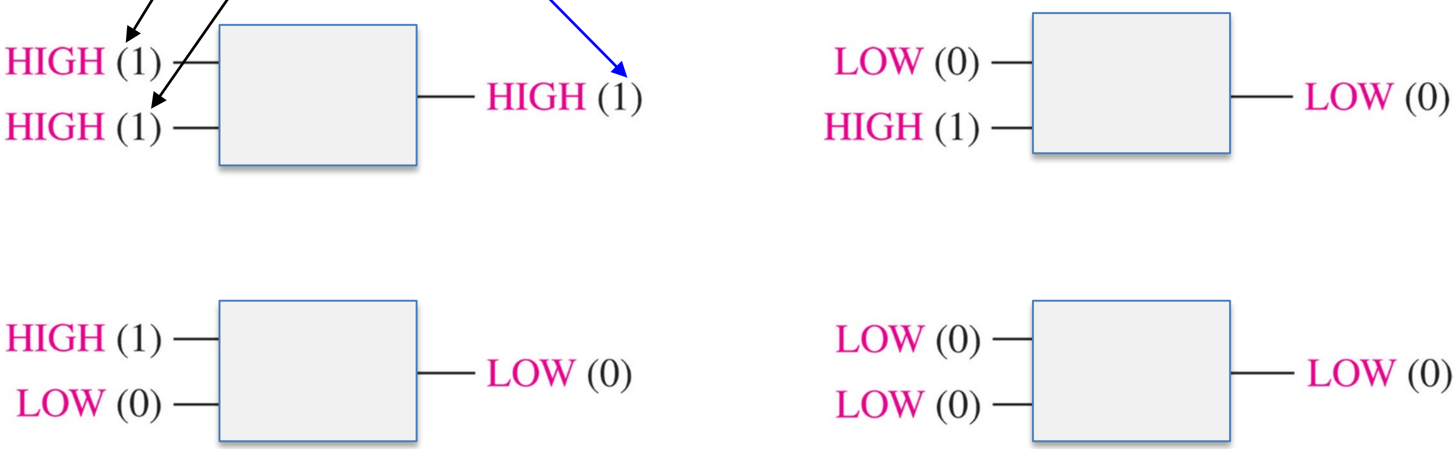
M. Morris Mano, Michael D. Ciletti, Digital design, Sixth edition, Pearson, 2018.

Thomas L. Floyd, Digital Fundamentals, A Systems Approach, Pearson, 2013.



AND

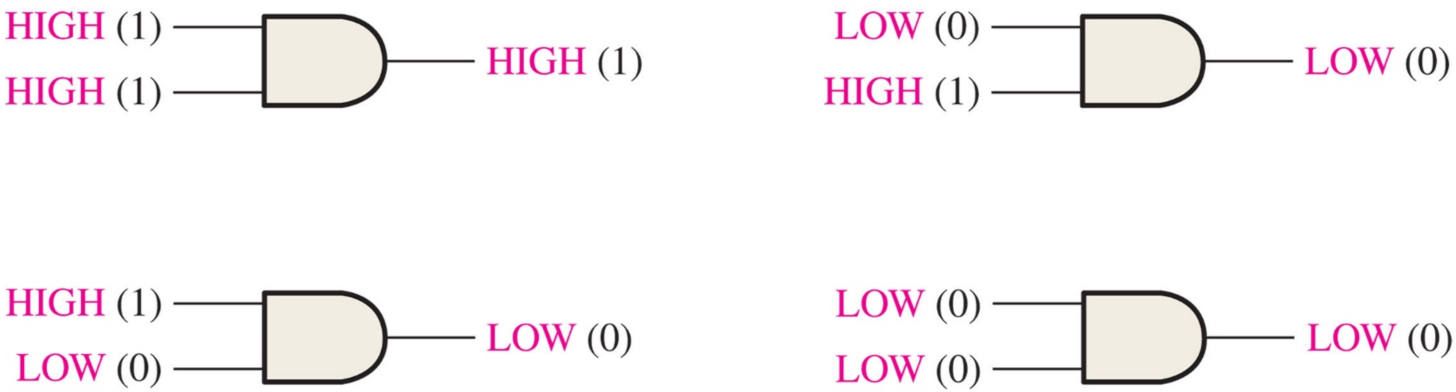
X	Y	$X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

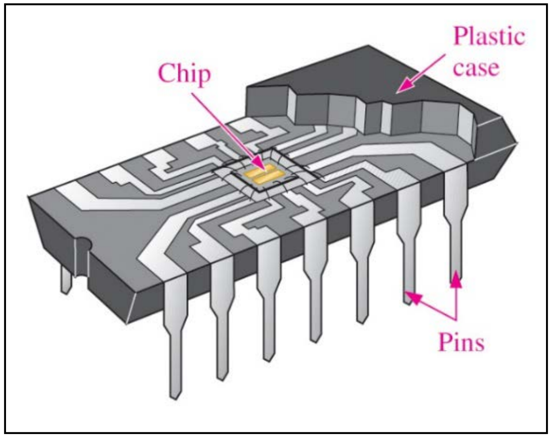


Thomas L. Floyd, Digital Fundamentals, A Systems Approach, Pearson, 2013.

AND

X	Y	$X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1



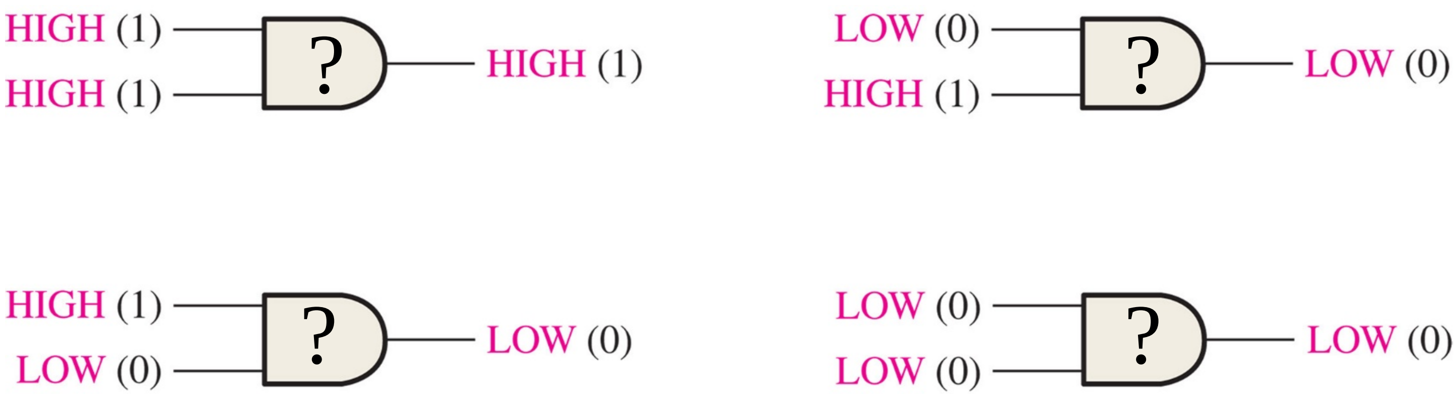


Thomas L. Floyd, Digital Fundamentals,

A Systems Approach, Pearson, 2013.

AND

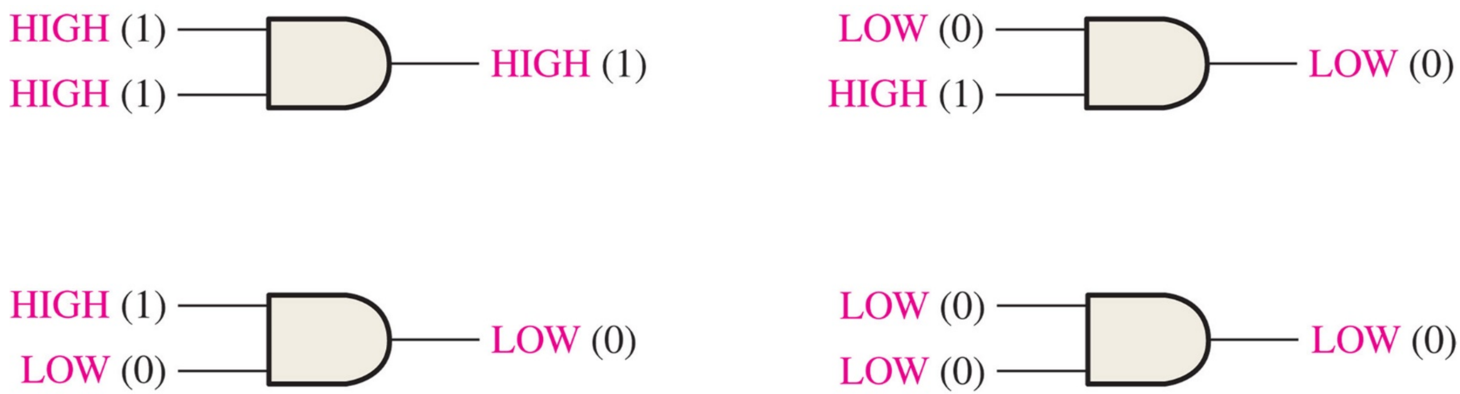
X	Y	$X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1

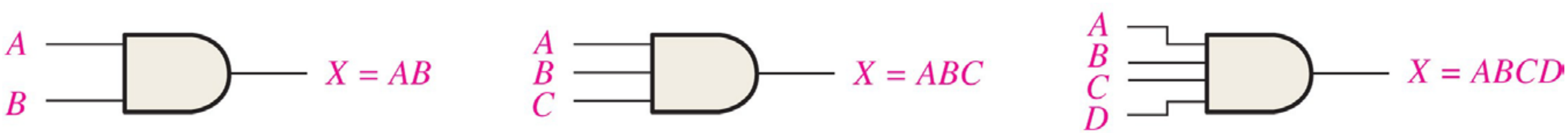


Thomas L. Floyd, Digital Fundamentals, A Systems Approach, Pearson, 2013.

AND

X	Y	$X \cdot Y$
0	0	0
0	1	0
1	0	0
1	1	1



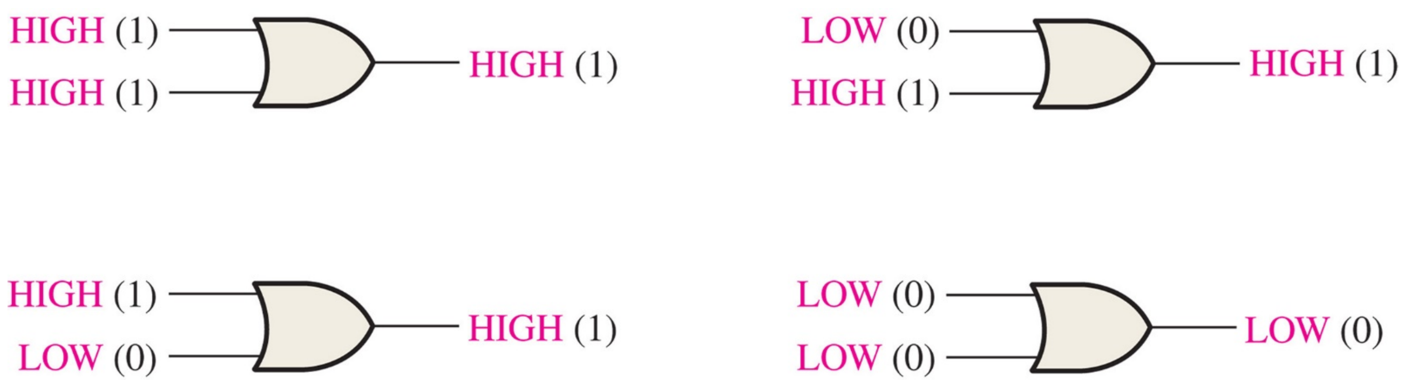


(a) (b) (c)

Thomas L. Floyd, Digital Fundamentals, A Systems Approach, Pearson, 2013.

OR

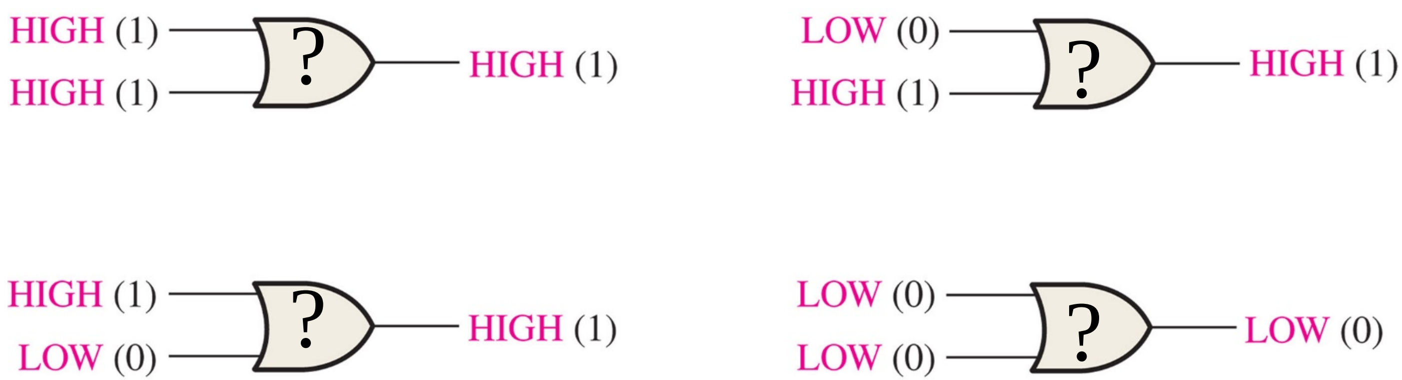
X	Y	$X + Y$
0	0	0
0	1	1
1	0	1
1	1	1



Thomas L. Floyd, Digital Fundamentals, A Systems Approach, Pearson, 2013.

OR

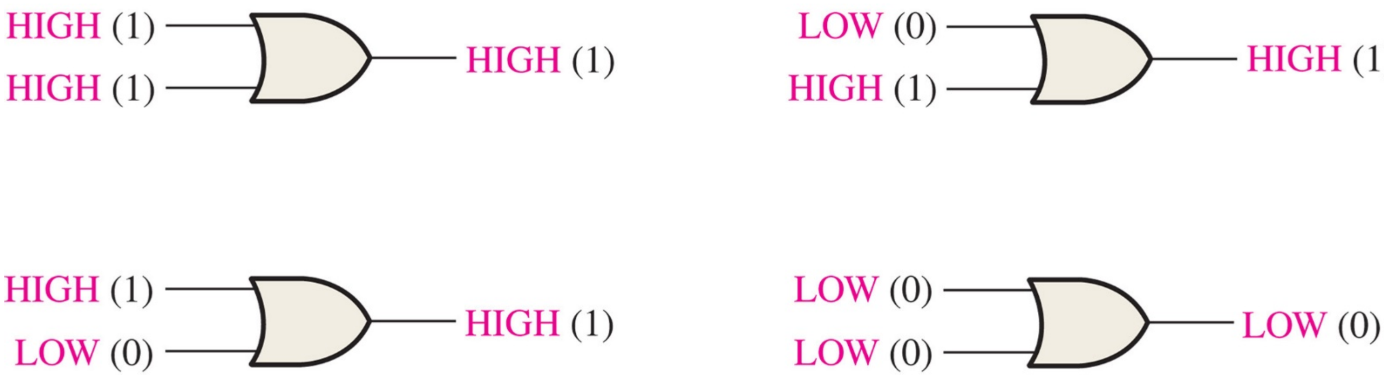
X	Y	$X + Y$
0	0	0
0	1	1
1	0	1
1	1	1

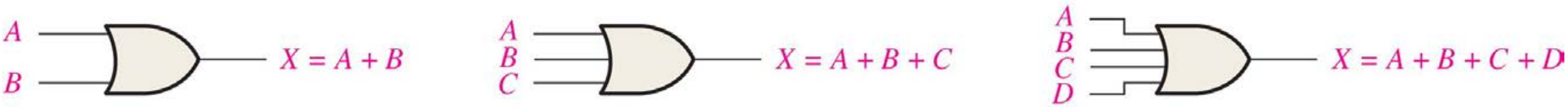


Thomas L. Floyd, Digital Fundamentals, A Systems Approach, Pearson, 2013.

OR

X	Y	$X+Y$
0	0	0
0	1	1
1	0	1
1	1	1





(a) (b) (c)

Thomas L. Floyd, Digital Fundamentals, A Systems Approach, Pearson, 2013.

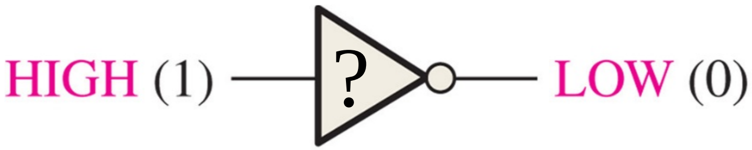
NOT

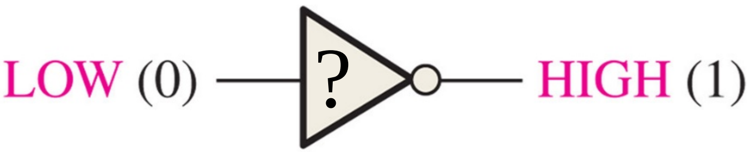
X	$\overline{X}$
0	1
1	0

Thomas L. Floyd, Digital Fundamentals, A Systems Approach, Pearson, 2013.

NOT

X	$\overline{X}$
0	1
1	0





Thomas L. Floyd, Digital Fundamentals, A Systems Approach, Pearson, 2013.