Układy logiczne

Algebra Boole'a:

Zmienne przyjmują dwie wartości:
1- prawda (true), 0 –fałsz (false)

Podstawowe operacje:

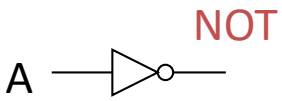
$A * B \equiv A \; \boldsymbol{AND} \; B$

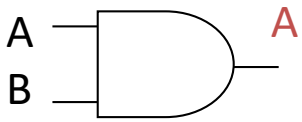
$A + B \equiv A \; \boldsymbol{OR} \; B$

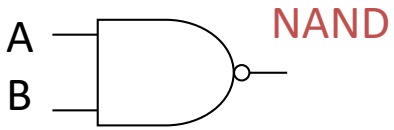
$\overline{A} \equiv \boldsymbol{NOT} \; A$

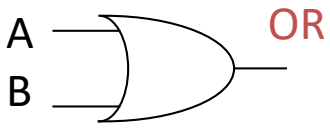
Bramki logiczne

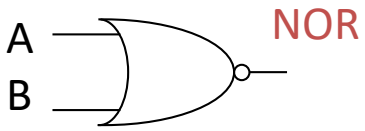
Podstawowe składniki wszystkich układów logicznych

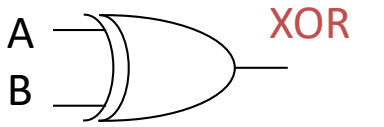


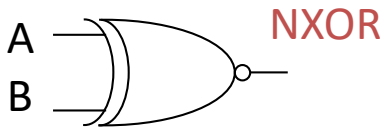












A	NOT
0	1
1	0

A	B	AND	NAND
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

A	B	OR	NOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

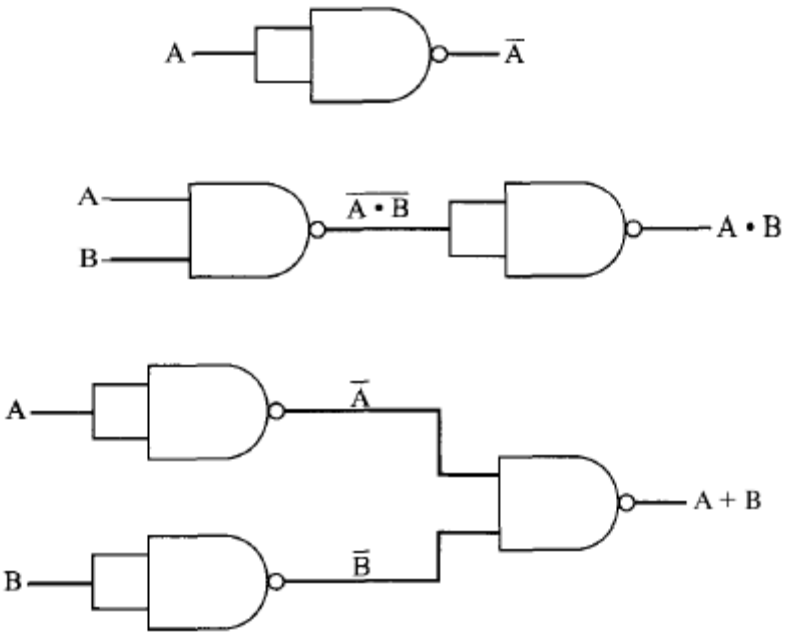
A	B	XOR	NXOR
0	0	0	1
0	1	1	0
1	0	1	0
1	1	0	1

Podstawowe tożsamości algebry Boole'a

A* B = B*A	A+B = B+A	prawo przemienności
A*(B+C) = A*B + A*C	A+(B*C) = (A+B)*(A+C)	prawo rozdzielności
1 * A = A	0 + A = A	prawo tożsamości
$A * \overline{A} = 0$	$A + \overline{A} = 1$	prawo odwrotności
0 * A = 0	1 + A = 1	
A * A = A	A + A = A	
$\overline{A * B} = \overline{A} + \overline{B}$	$\overline{A + B} = \overline{A} * \overline{B}$	tw. de Morgana

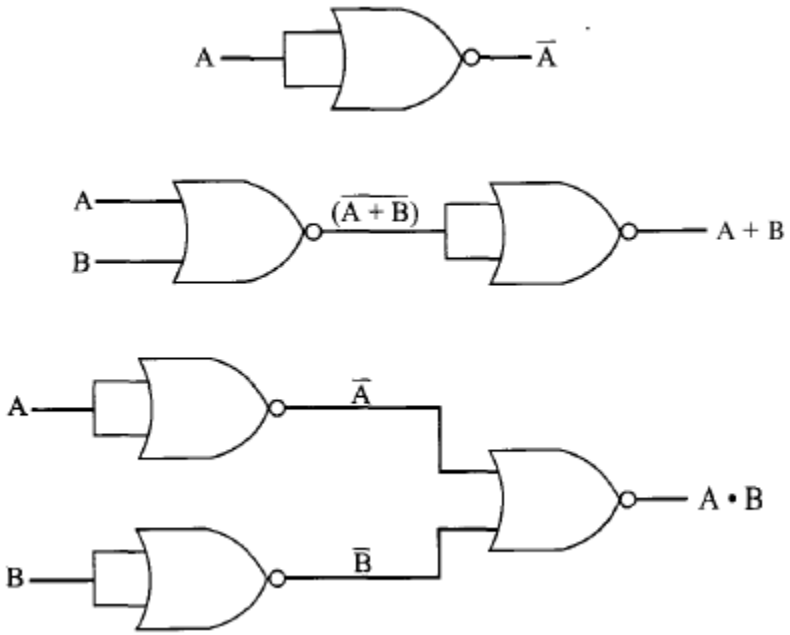
Przykłady realizacji funkcji logicznych

NAND



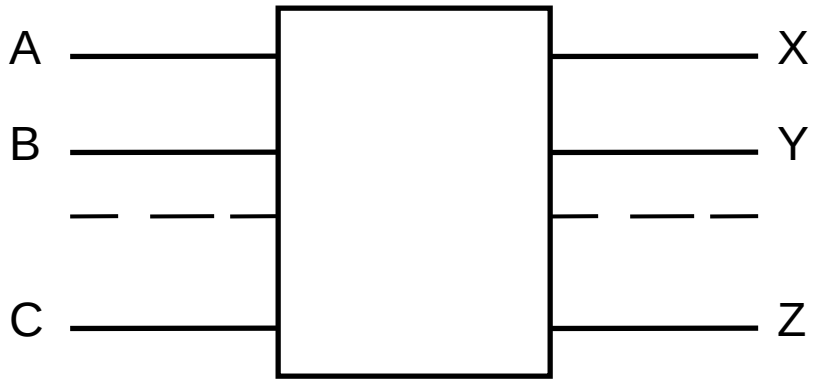
Przykłady realizacji funkcji logicznych

NOR



Układ logiczny

- elementarny blok mający jedno lub więcej wejść i jedno lub więcej wyjść. Jest on zwykle projektowany jako standardowa jednostka funkcjonalna. Zadaniem układu logicznego jest przyjmowanie standardowych sygnałów logicznych na swoich wejściach i produkowanie na wyjściach innych, również standardowych sygnałów logicznych



Ogólne oznaczenie układu logicznego

Struktura wewnętrzna układu logicznego może zawierać różne rodzaje układów przełączających.Zmienne logiczne (mające wartości 0 lub 1) są oznaczone przez *A, B, C..., X, Y, Z.*

Bloki funkcjonale

Układy kombinacyjne

Stan wyjść jest jednoznacznie określony przez stan wejść układu:

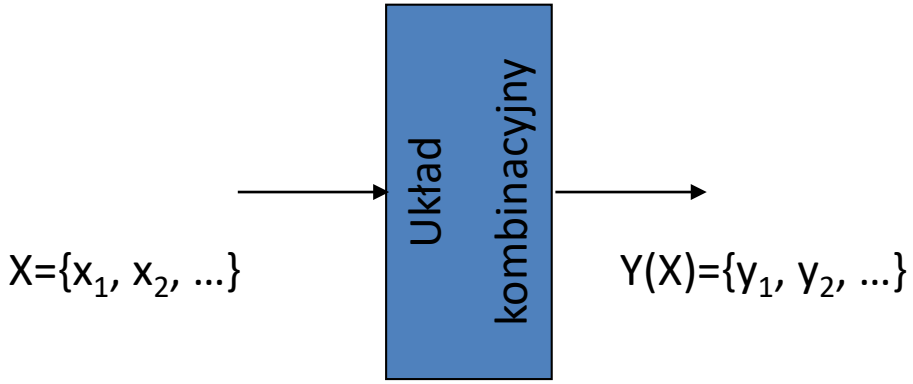
Układy sekwencyjne

Stan wyjść zależy od stanu wejść oraz od poprzednich stanów układu:

$$Y(t_n) = f(X(t_0), X(t_1), \dots, X(t_{n-1}))$$

Układy kombinacyjne

- Stan wyjść zależy tylko od stanu wejść
- Układ taki można definiować za pomocą:
  - Tablicy prawdy
  - Symbolu graficznego
  - Równania Boole'a



Układy kombinacyjne

Tablica prawdy:

Sygnały wejściowe			Sygnał wyjściowy
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	0

Układy kombinacyjne

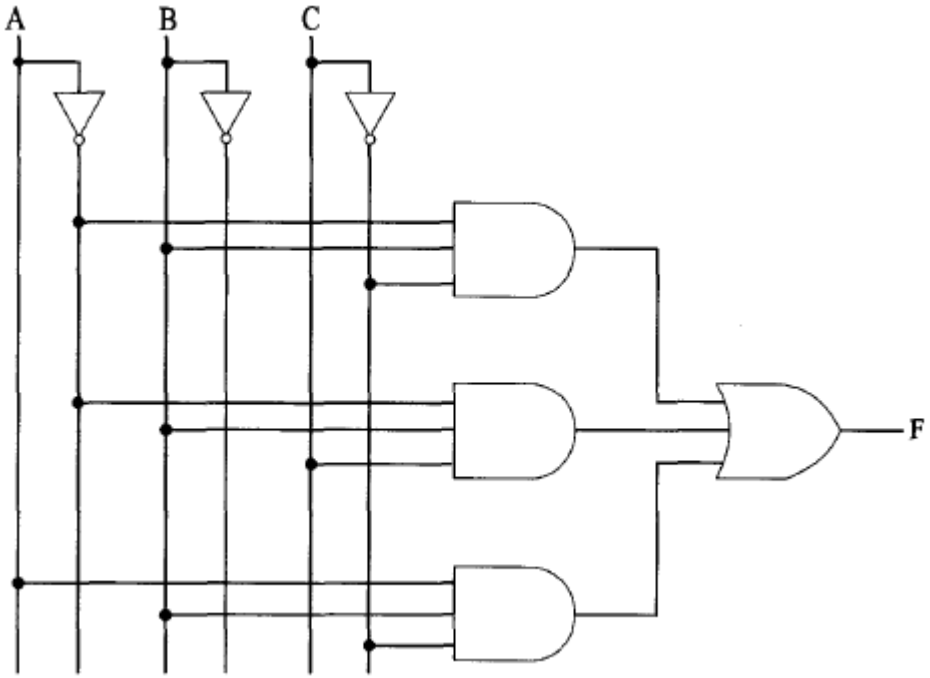
Równanie Boole’a:

Można je wyrazić jako sumę kombinacji wartości zmiennych A, B, C , dla których mamy F=1

$$F = \overline{A} * B * \overline{C} + \overline{A} * B * C + A * B * \overline{C}$$

Układy kombinacyjne

Realizacja układu za pomocą bramek AND, OR i NOT:



Metody upraszczanie układów kombinacyjnych

Mapa Karnaugha:

Kod Graya

(a)

AB	00	01	11	10
		1		1

$F=\overline{A}*B+A*\overline{B}$

(b)

A \ BC	00	01	11	10
0			1	1
1				1

$F=\overline{A}*B*C+\overline{A}*B*\overline{C}+A*B*\overline{C}$

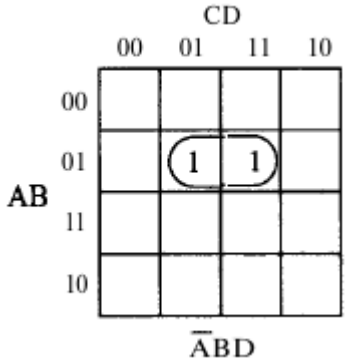
(c)

AB \ CD	00	01	11	10
00			1	
10				
11	1			
10		1		

$F=\overline{A}*\overline{B}*C*D+A*B*\overline{C}*\overline{D}+A*\overline{B}*\overline{C}*D$

Metody upraszczanie układów kombinacyjnych

Mapa Karnaugha:

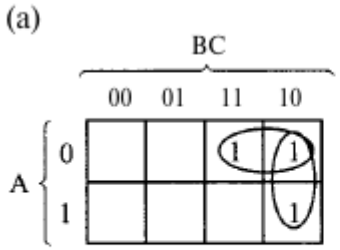


$$F = \overline{A} * B * \overline{C} * D + \overline{A} * B * C * D = \overline{A} * B * D$$

Jeśli sąsiadujące kwadraty zawierają 1, to odpowiednie iloczyny różnią się tylko jedną zmienną. W takim przypadku te iloczyny mogą być połączone przez wyeliminowanie tej zmiennej

Metody upraszczanie układów kombinacyjnych

Mapa Karnaugha:

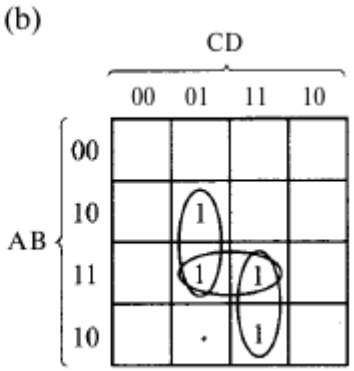


$F=\overline{A} * B + B * \overline{C}$

Gdy zakreślamy grupy, dozwolone jest użycie tej samej jedynki więcej niż jeden raz.

Metody upraszczanie układów kombinacyjnych

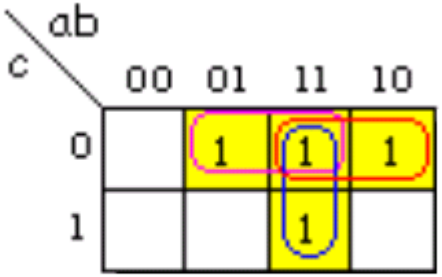
Mapa Karnaugha:



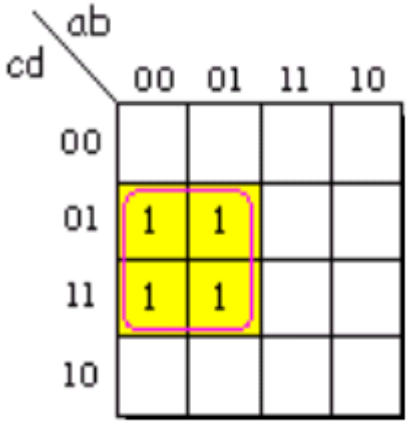
$F=A*\overline{C}*D+A*C*D$

Możemy wyeliminować dowolną grupę jedynek, która w całości nakłada się z innymi grupami

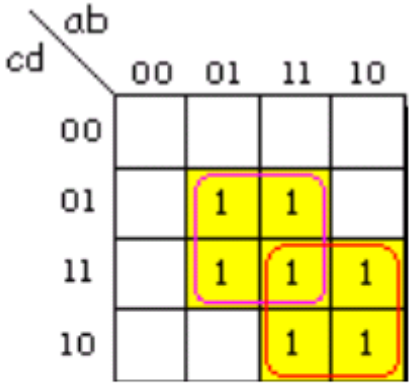
Metody upraszczanie układów kombinacyjnych



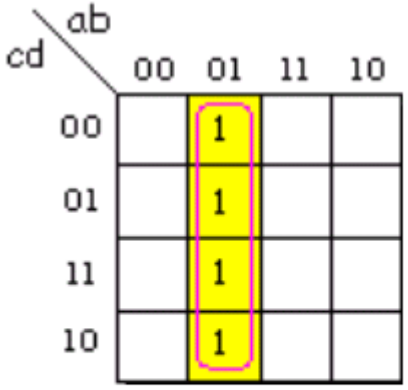
$$Y = (b * \overline{c}) + (a * \overline{c}) + (a * b)$$



$$Y = \overline{a} * d$$

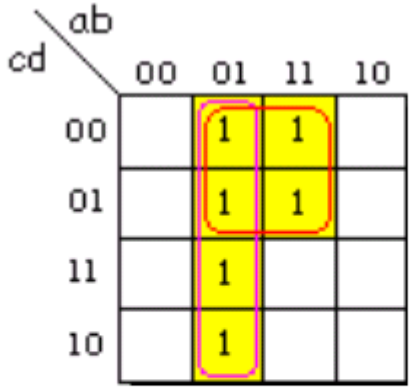


$$Y = (b * d) + (a * c)$$

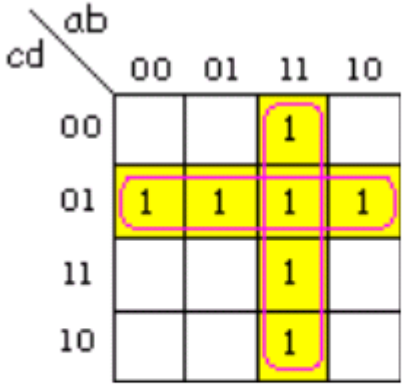


$$Y = (\overline{a} * b)$$

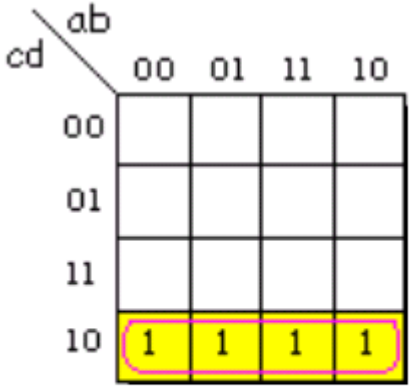
Metody upraszczanie układów kombinacyjnych



$$Y = (\overline{a} * b) + (b * \overline{c})$$

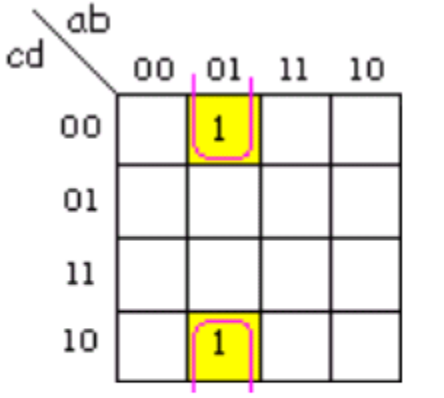


$$Y = (a * b) + (\overline{c} * d)$$

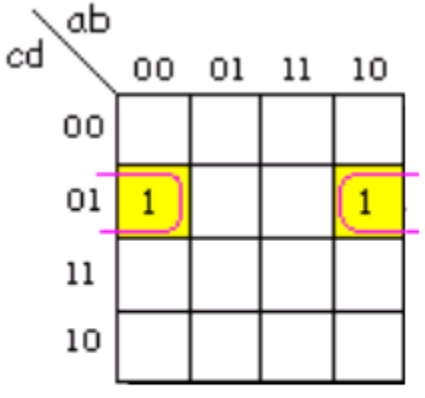


$$Y = (c * \overline{d})$$

Metody upraszczanie układów kombinacyjnych



$$Y = (\overline{a} * b * \overline{d})$$



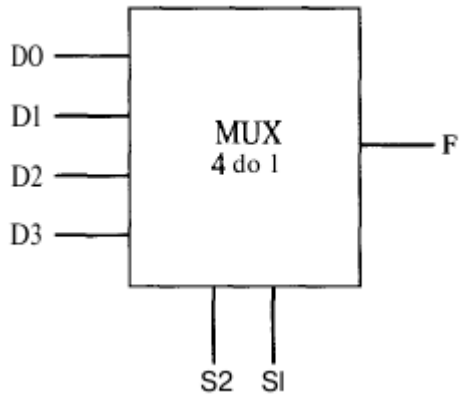
$$Y = (\overline{b} * \overline{c} * d)$$

Układy kombinacyjne

Przykładowe układy:

- Multiplekser, demultiplekser
- Koder, dekoder
- Sumator
- Komparator

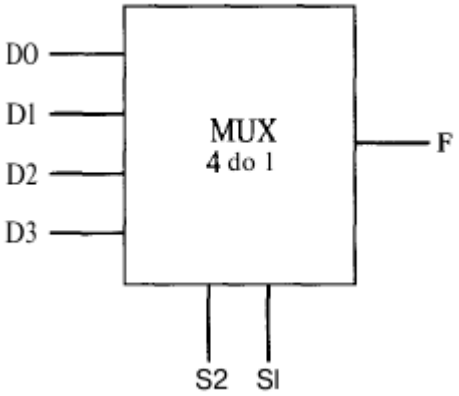
Multiplekser

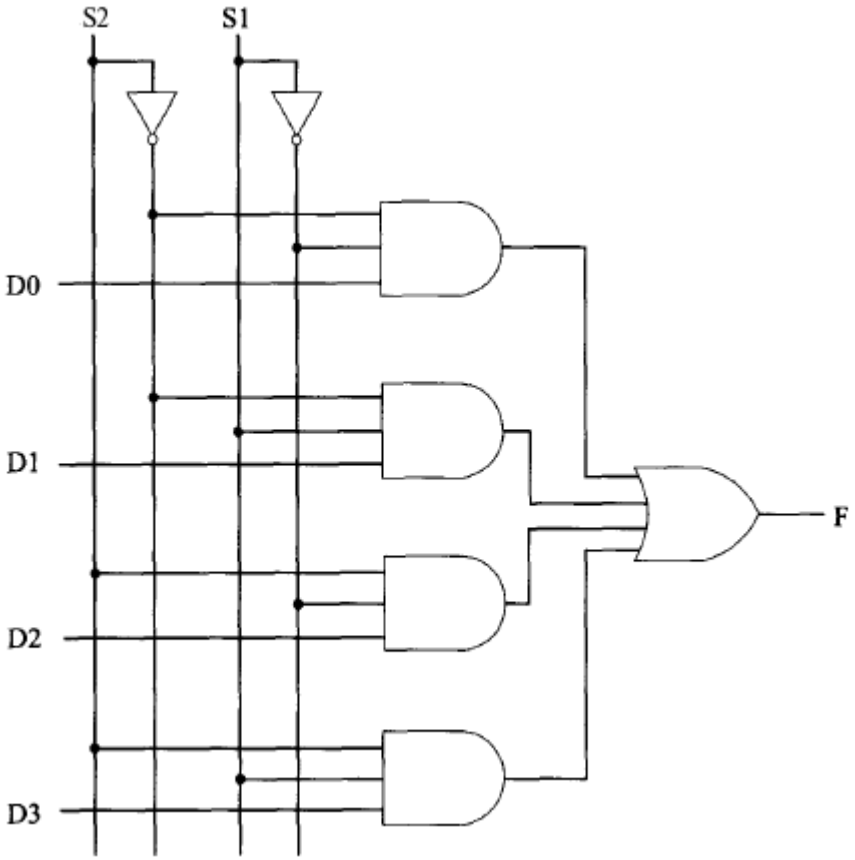


S2	S1	F
0	0	D0
0	1	D1
1	0	D2
1	1	D3

Tablica prawdy

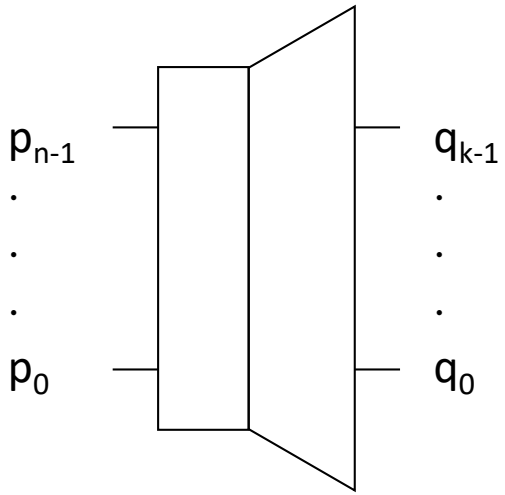
Multiplekser

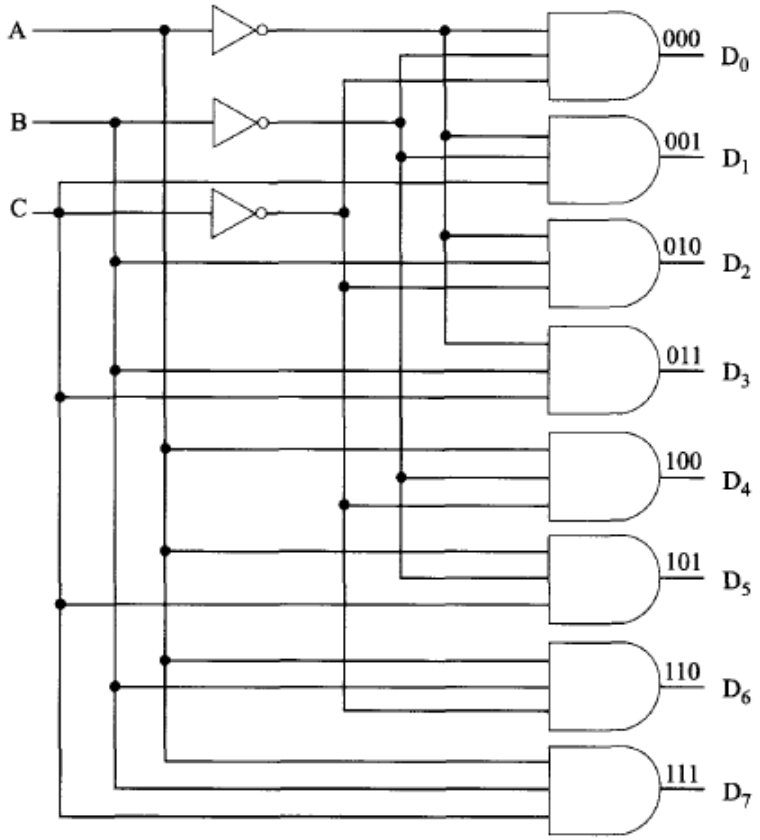




S2	S1	F
0	0	D0
0	1	D1
1	0	D2
1	1	D3

Dekodery





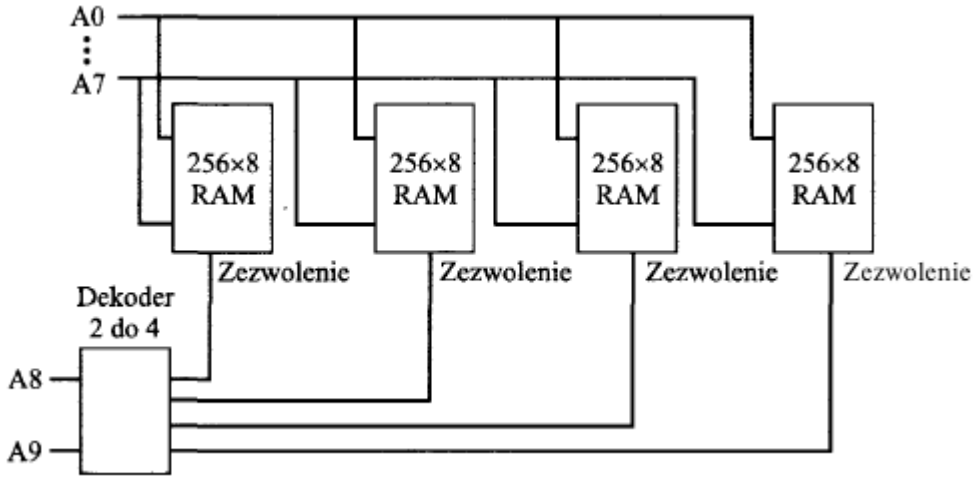
$p_{n-1} \dots p_0$ – wejścia dekodera

$q_{k-1} \dots q_0$ – wyjścia dekodera

$k=2^n$

Dekodery znajdują zastosowanie np. do dekodowania adresu

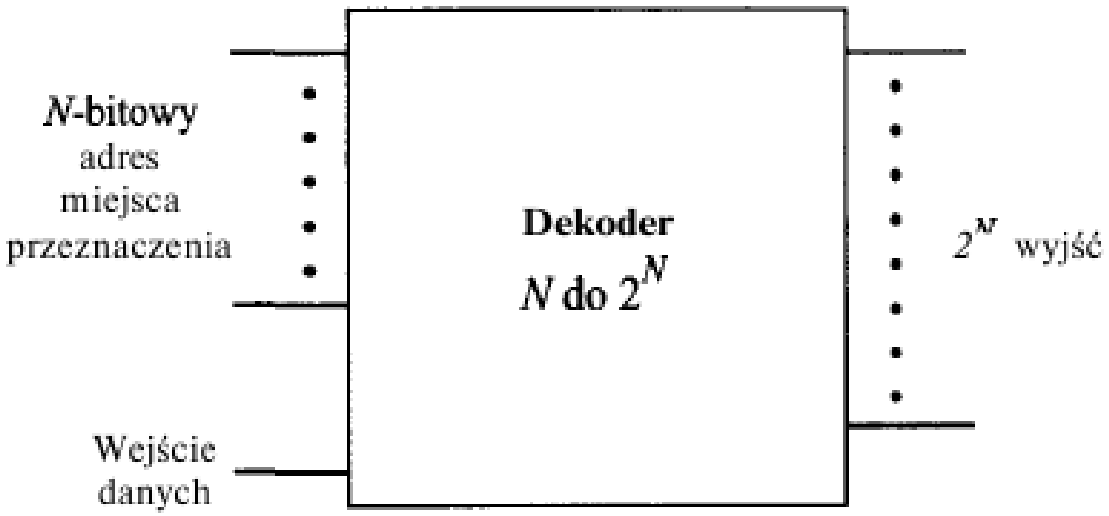
Dekodery



Chcemy zbudować 1 kilobajtową pamięć z czterech układów RAM o pojemności 256 bajtów. Przestrzeń adresową możemy podzielić następująco:

adres	układ
0000 – 00FF	0
0100 – 01FF	1
0200 – 02FF	2
0300 – 03FF	3

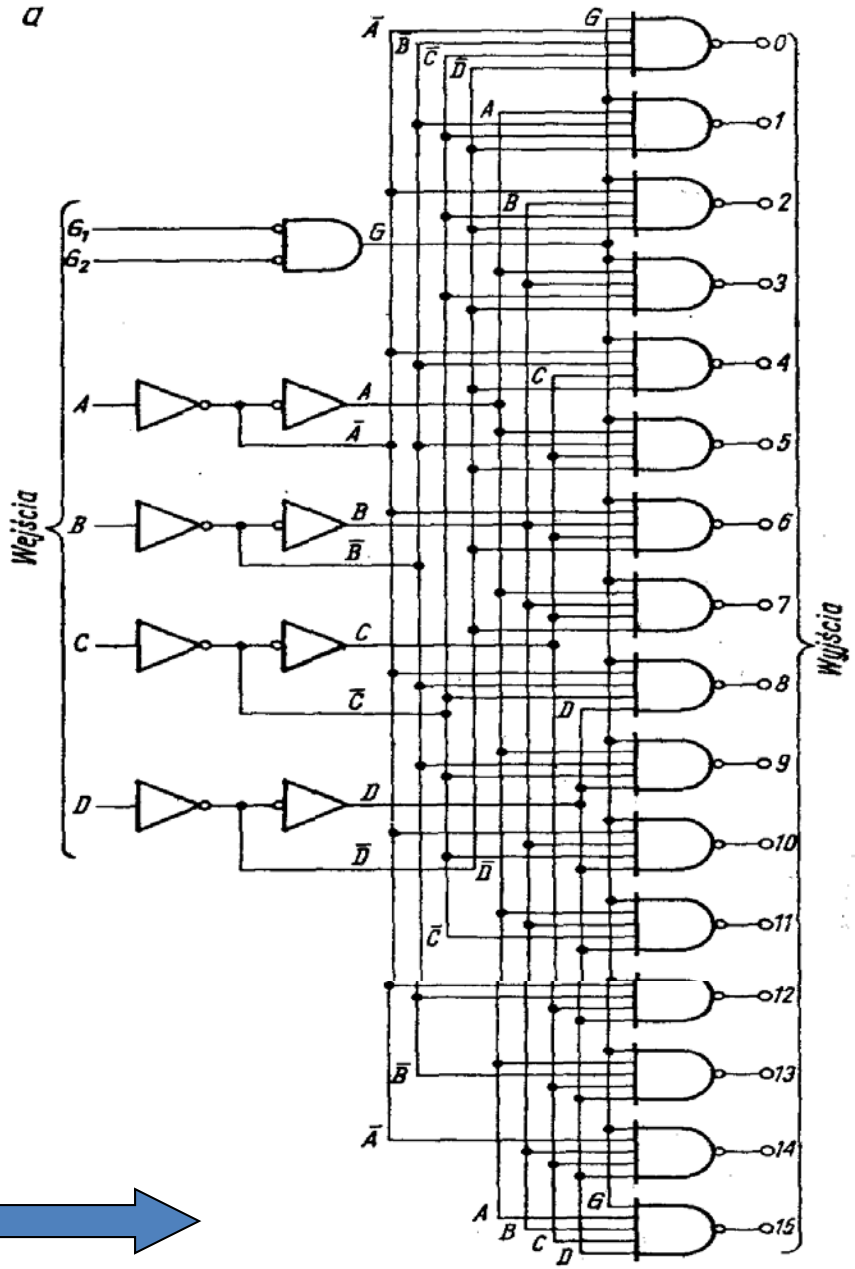
Demultiplekser



Po dodaniu jednej linii wejściowej
dekoder może służyć jako demultiplekser

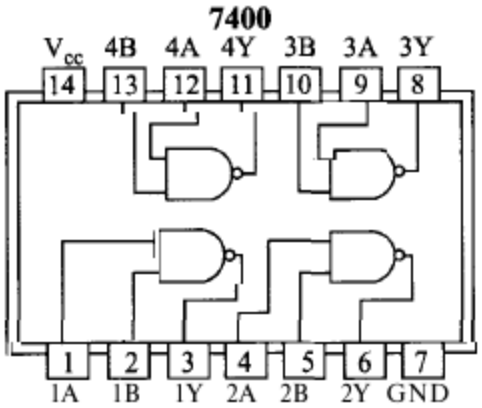
Demultiplekser 154

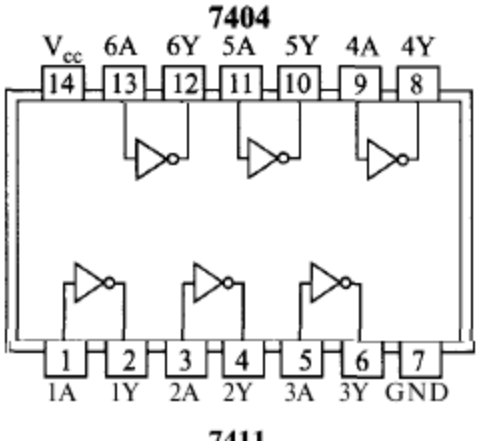
W technice TTL są produkowane demultipleksery o 16 oraz o 4 wyjściach informacyjnych i odpowiednio o 4 i 2 wejściach adresowych.
Typowym reprezentantem demultiplekserów scalonych jest układ 154 . Układ ten spełnia funkcję dekodera naturalnego 4-bitowego kodu dwójkowego na kod l z 16 i jest wyposażony w wejścia strobujące G1, i G2 z których jedno może służyć jako wejście informacyjne, a drugie jako wejście strobujące. Słowo adresowe (dekodowane) jest podawane na wejścia A, B, C i D powodując, że jedno z wyjść znajdzie się w stanie niskim, jeśli na obydwu wejściach strobujących jest poziom niski.



Schemat blokowy

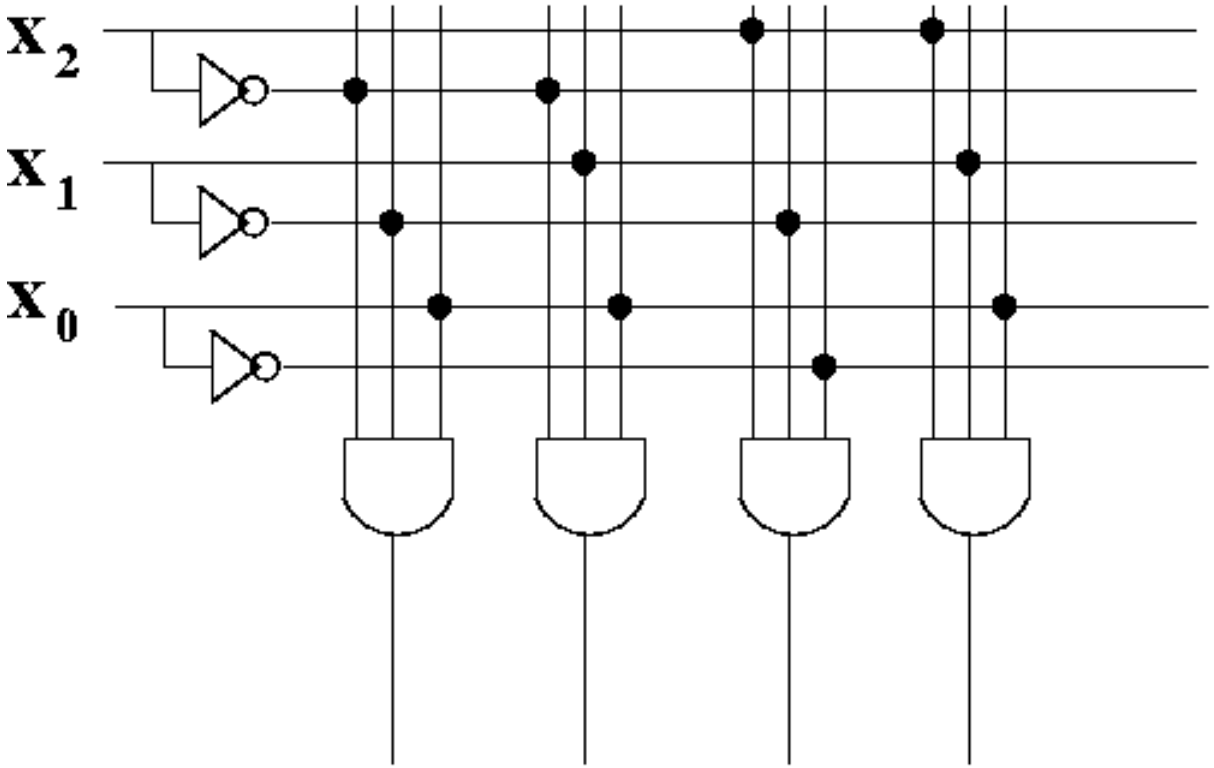
Układy małej skali integracji (SSI)





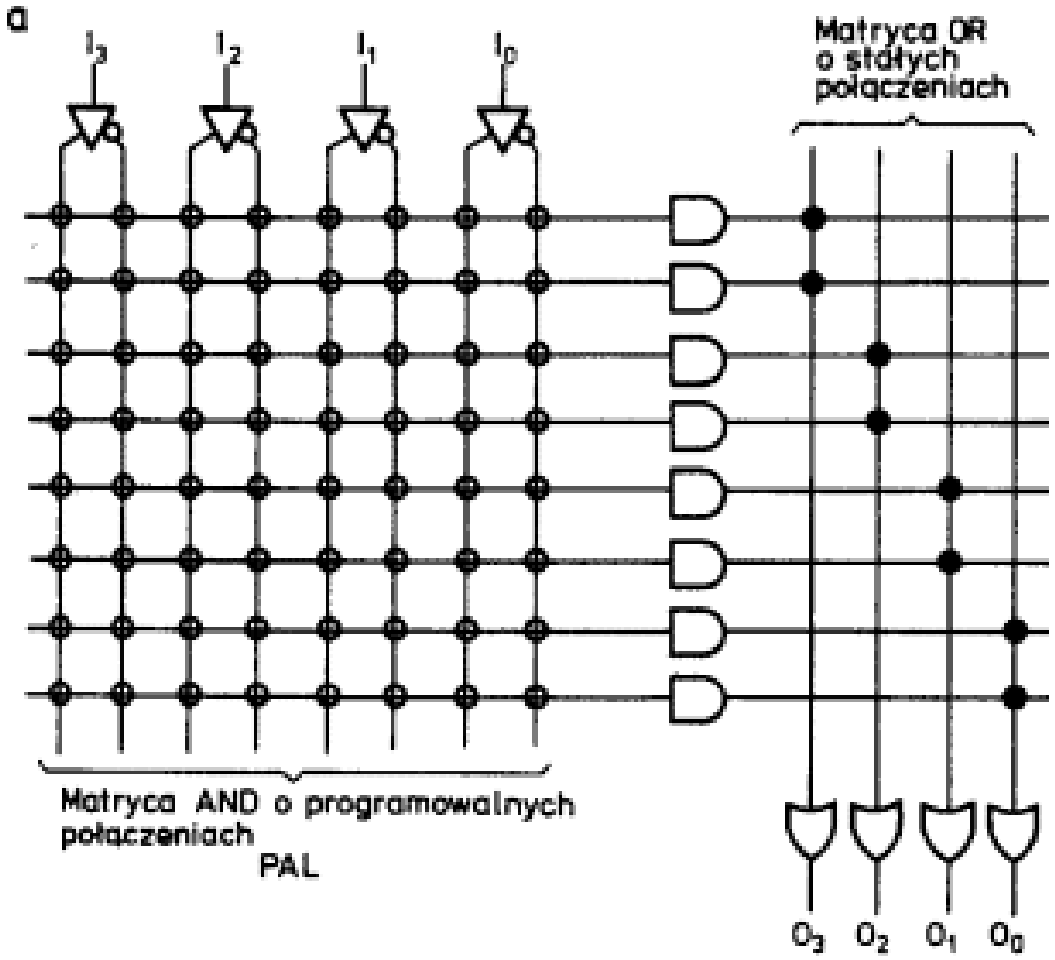
Aby zrealizować funkcję logiczną należy użyć pewną liczbe tych układów

Programowalne tablice logiczne (PLA)



Koncepcja PLA polega na tym, że dowolna funkcja Boole'a może być wyrażona na podstawie sumy iloczynów. Programowanie polega na przepalaniu zbędnych połaczeń.

Programowalne tablice logiczne (PLA)



Bezpośrednio programowalna macierz bramek, FPGA

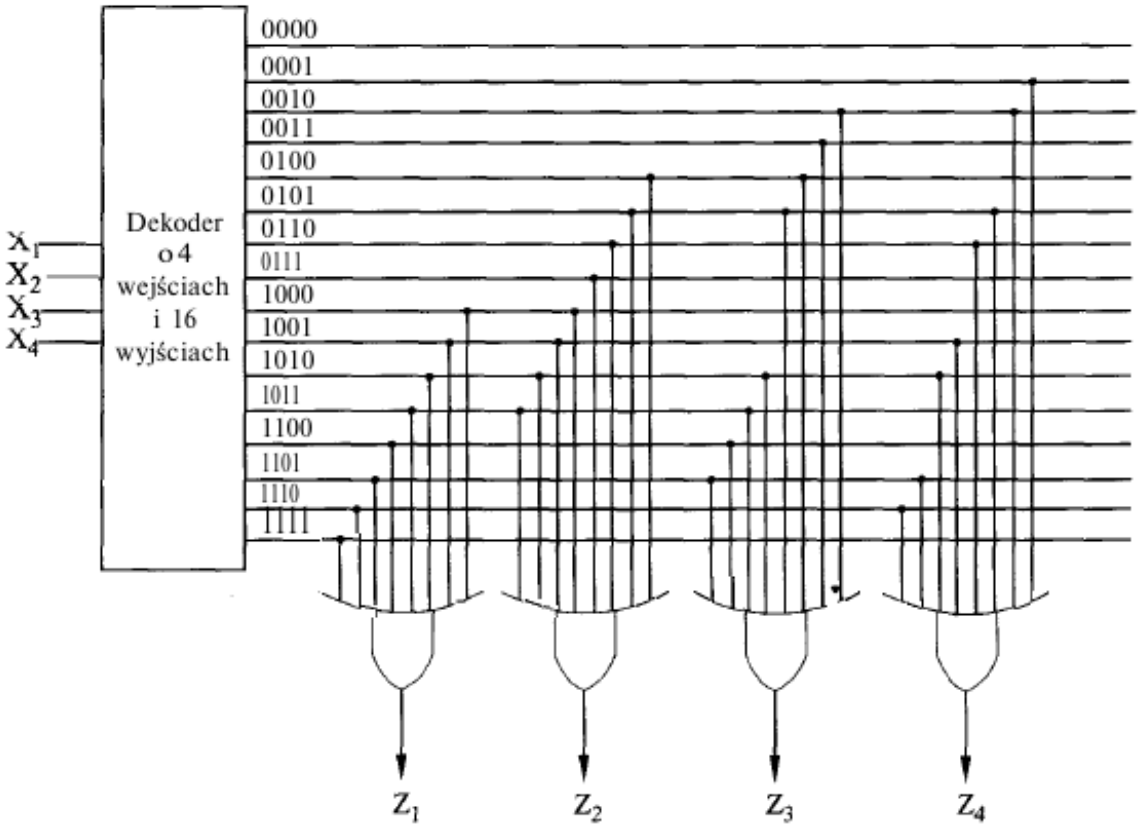
Field-Programmable Gate Array

Struktury FPGA zawierają dziesiątki tysięcy bloków logicznych o bardzo zróżnicowanej budowie

Układy FPGA używane są w cyfrowym przetwarzaniu sygnałów, w fazie prototypowej układów ASIC i w wielu innych dziedzinach.

Aby zdefiniować zachowanie układu FPGA używa się języka opisu sprzętu (np.: Verilog, VHDL)

Pamięć stała (ROM –read only memory)



Wejścia/ adresy				Wyjścia /zawartość			
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0

Informacja zawarta w pamięci ROM jest trwała. Jest ona zapisana w procesie tworzenia układu.

Sumatory

A	B	S	C
0	0	0	0
1	0	1	0
0	1	1	0
1	1	0	1

C_i	A	B	S	C_o
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

A, B – dane wejściowe
C_i – wejście przeniesienia
S – dane wyjściowe (suma)
C_o – wyjście przeniesienia

Sumatory

		A B			
		0 0	01	11	1 0
C_i	0	0	1	0	1
	1	1	0	1	0

Suma

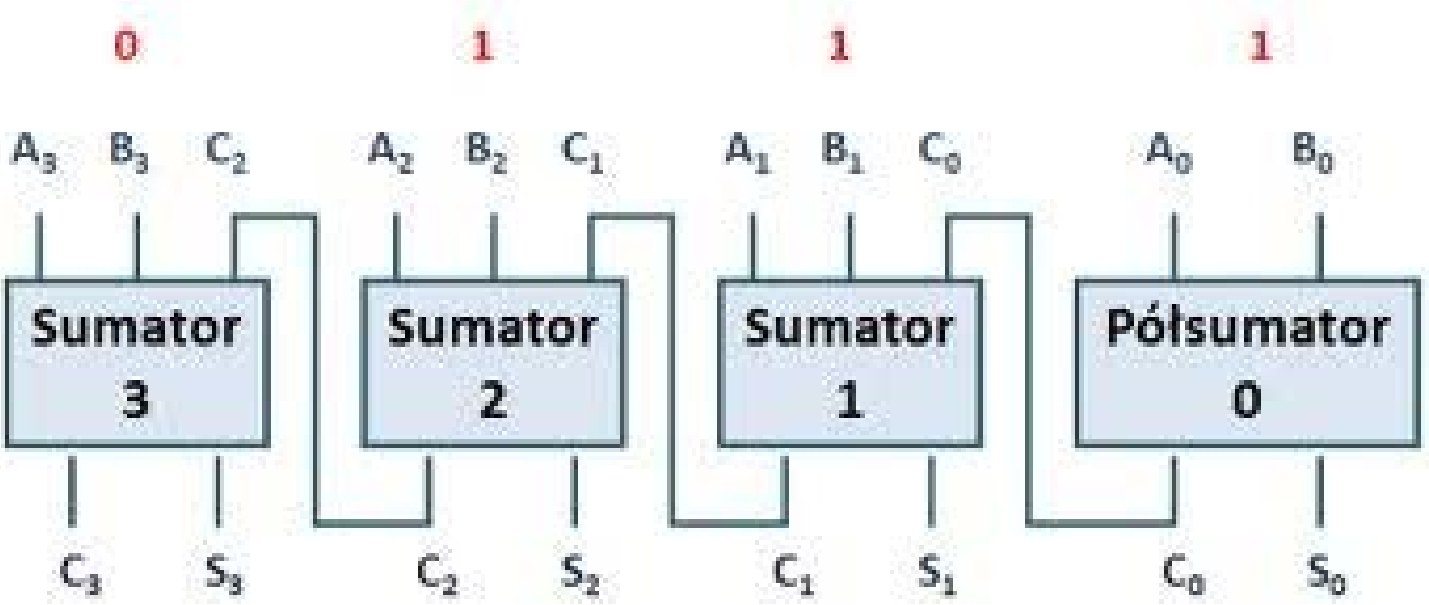
		A B			
		0 0	01	11	1 0
C_i	0	0	0	1	0
	1	0	1	1	1

Wyjście przeniesienia

$$S=\overline{A} * \overline{B} * C_i + \overline{A} * B * \overline{C_i} + A * B * C + A * \overline{B} * \overline{C_i}$$

$$C_o = CB + CA + AB$$

Sumator 4-bitowy



Sumator 32-bitowy

Można zbudować sumator dla większej ilości bitów złożony z sumatorów 1-bitowych.

Wady takiego rozwiązania: w każdym sumatorze 1-bitowym występuje opóźnienie odpowiedzi względem sygnałów wejściowych. Dla sumatora wielobitowego może być bardzo duże.

Rozwiązanie:

- Określenie wartości przeniesień bez przechodzenia przez wszystkie poprzednie stopnie
- Każdy sumator 1-bitowy działa niezależnie i opóźnienia się nie kumulują

Sumator 32-bitowy

$$C_0 = A_0 B_0 \qquad (*)$$

$$C_1 = A_1 B_1 + (A_1 + B_1) C_0 \qquad (**)$$

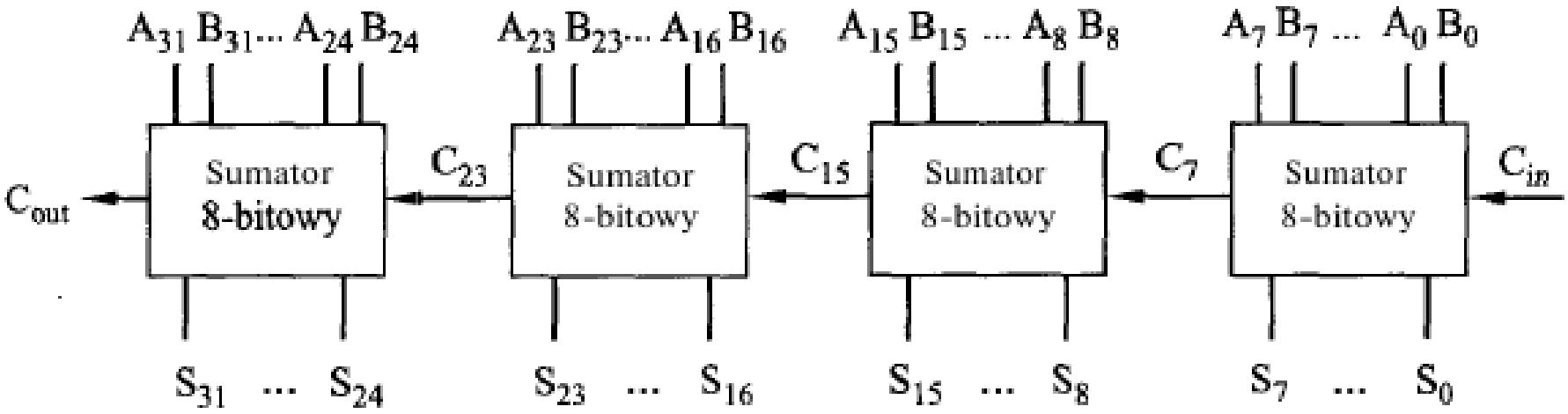
Podstawiając (*) do (**) dostajemy:

$$C_1 = A_1 B_1 + A_1 A_0 B_0 + B_1 A_0 B_0$$

Powtarzając tę procedurę dostajemy kolejne wartości przeniesień . Jednak w przypadku długich liczb to rozwiązanie staje się bardzo skomplikowane.

Sumator 32-bitowy

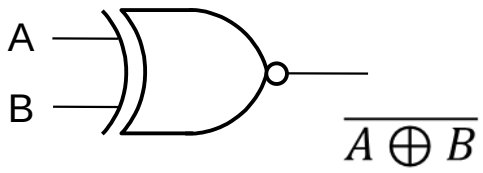
Stosuje się rozwiązania pośrednie. Np. sumator 32-bitowy można zbudować z 4 sumatorów 8-bitowych.



Komparatory

Komparatorem cyfrowym nazywamy układ służący do porównywania dwu lub więcej liczb binarnych. Najważniejsze kryteria porównawcze to A = B, A > B, A < B. Układ sprawdzający wszystkie trzy relacje nazywa się komparatorem uniwersalnym. Najprostsze komparatory umożliwiają jedynie określenie czy dwie porównywane liczby są sobie równe lub która z liczb jest większa.

Kryterium równości dwóch liczb binarnych jest identyczność wszystkich bitów.
W przypadku dwóch liczb jednobitowych A i B, informację o tym uzyskać można za pomocą funkcji **negacja EXOR**:



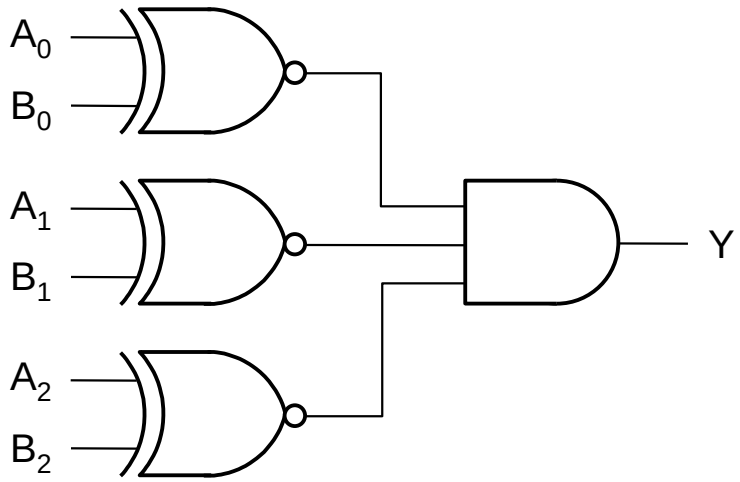
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

Wartość 1 na wyjściu sygnalizuje równość A = B.

Komparatory

Przykład komparatora równoległego 3 – bitowego

Komparator równoległy to taki układ, na którego wejścia podawane są jednocześnie wszystkie bity porównywanych liczb.



Y = 1 tylko wówczas gdy:
$A_0 = B_0$ i $A_1 = B_1$ i $A_2 = B_2$
czyli A = B.