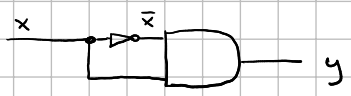
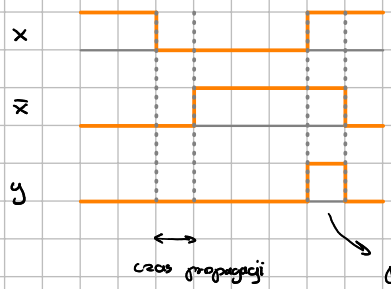
Hazard statyczny – zjawisko fizyczne



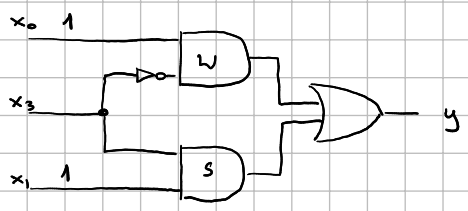
fizyczne elementy charakteryzuje czas propagacji (rzędu 1 ns)

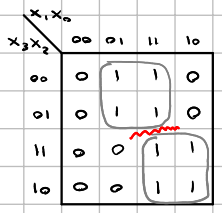


Logiczenie, układ zawsze daje na wyjściu 0

przez krótki Δt $x \cdot \bar{x}$ daje na wyjściu 1 (szpilka)

Impulsy hazardowe prawie zawsze pojawiają się w układach kombinacyjnych

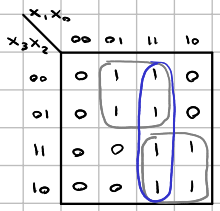




miejsca wystąpienia szpilki widać w mapie Karnaugha

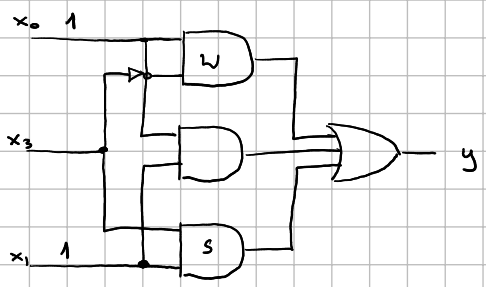
$y = \bar{x}_3 x_0 + x_3 x_1$

Rezygnacja z postaci minimalnej pozwala na uniknięcie hazardu



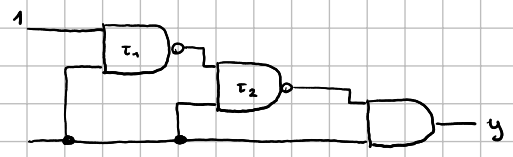
$y = \bar{x}_3 x_0 + x_3 x_1 + x_1 x_0$

dodatkowy implikant zapobiega powstawaniu hazardu



Hazard dynamiczny

ten sam sygnał dochodzi do conajmniej 3 poziomów układu



należy unikać faktoryzacji

Układy wielowyjściowe

$Y = F(X)$
$X = (x_{m-1}, \dots, x_0) \rightarrow m$
$Y = (y_{n-1}, \dots, y_0) \rightarrow n$
$F = (f_{n-1}, \dots, f_0) \rightarrow n$

$$Y = \begin{cases} y_0 = f_0(x_{m-1}, \dots, x_0) \\ y_1 = f_1(x_{m-1}, \dots, x_0) \\ \dots \\ y_{n-1} = f_{n-1}(x_{m-1}, \dots, x_0) \end{cases}$$

n funkcji, każda od tego samego argumentu

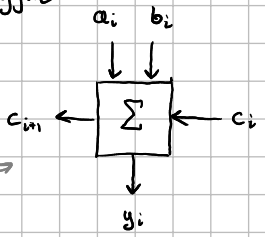
Minimalizacja łączna i rozłączna

Minimalizacja rozłączna - oddzielnie każdej z funkcji składowych

Przeprowadzając minimalizację każdej funkcji oddzielnie otrzymamy układ, który możnaby jeszcze uprościć wykorzystując implikanty powtarzające się między funkcjami (i oszczędzić bramki)

1) Minimalizacja każdej z funkcji y_i na mapie Karnaugha
2) Znalezienie wszystkich kombinacji iloczynów y_i (dla 3 → y_1y_2, y_1y_3, y_1y_3, $y_1y_2y_3$) i minimalizacja na mapie Karnaugha
3) Wpisanie wszystkich implikantów do tabeli Quine'a z podziałem na funkcje składowe
4) Znalezienie minimalnego pokrycia i zapisanie każdej z funkcji składowych

Sumowanie jednopozycyjne



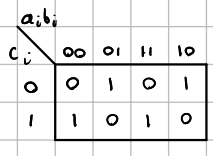
a, b – składniki sumy
c – przeniesienie
y – wynik

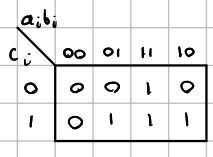
tabela prawdy → zespół funkcji

a_i	b_i	c_i	y_i	c_{i+1}
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

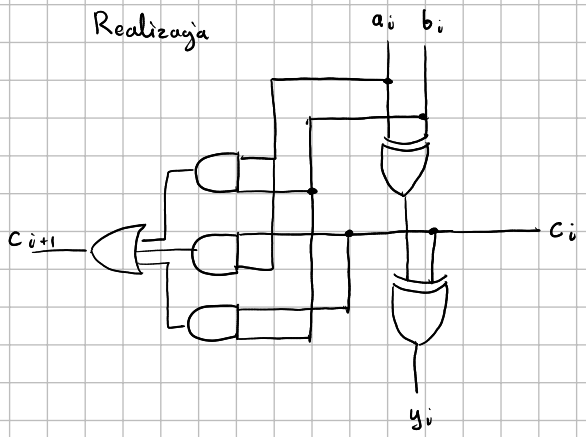
$$y_i = a_i \oplus b_i \oplus c_i$$

$$c_{i+1} = a_i b_i + a_i c_i + b_i c_i$$

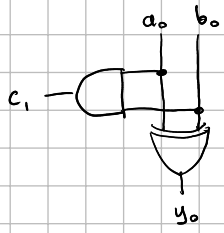




Realizacja

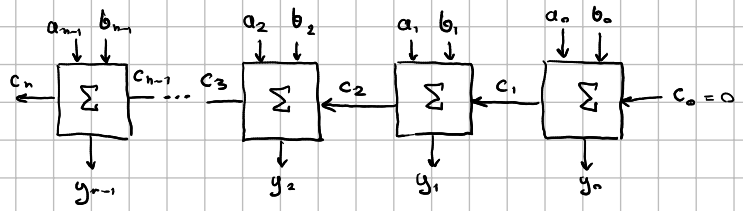


przypadek brzegowy kiedy $c_0 = 0$ można znacznie uprościć



Sumator iteracyjny n-pozycyjny

Gotowy blok wystarczy powtórzyć odpowiednią liczbę razy podając na wejścia kolejne bity i łącząc sygnały przeniesienia



Układy iteracyjne

Rozwiązujemy po kolei mniejsze problemy

Projektowanie układu iteracyjnego

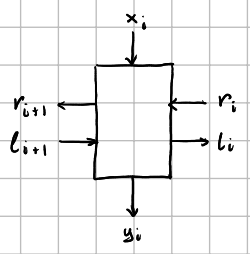
- Zdekomponować układ dla pojedynczego wyjścia – jeden blok
- Określić i wyznaczyć przeniesienia pomiędzy blokami (tabela prawdy)
- Wyznaczyć funkcję wyjść i funkcję przeniesień
- Połączyć bloki

Przykładowy układ

daje '1' na wszystkich pozycjach między sąsiednimi '1' wyjścia

IN	0	0	1	0	1	1	0	0	0	1	0	0	0
OUT	0	0	1	1	1	1	1	1	1	1	0	0	0

blok



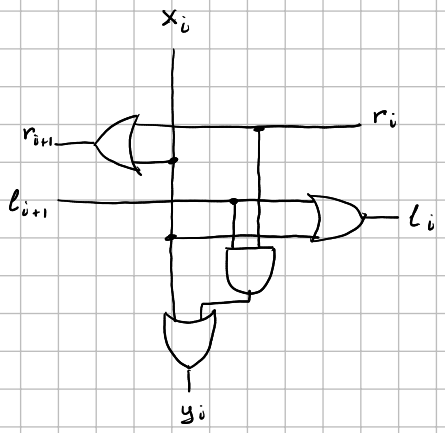
r – znajduje jedynkę z prawej strony
l – znajduje jedynkę z lewej strony

x_i	l_{i+1}	r_i	y_i	l_i	r_{i+1}
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	1	1	1
1	0	0	1	1	1
1	0	1	1	1	1
1	1	0	1	1	1
1	1	1	1	1	1

$y_i = x_i + l_{i+1} r_i$

$l_i = x_i + l_{i+1}$

$r_{i+1} = x_i + r_i$



Bloki funkcjonalne (kombinacyjne)

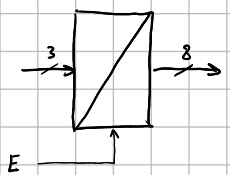
Gotowe układy realizujące często powtarzające się przydatne funkcje

Konwertery kodu (kodery i dekodery)

- kod NKB na kod Graya
- kod Graya na kod NKB
- kod NKB na kod '1 z n'

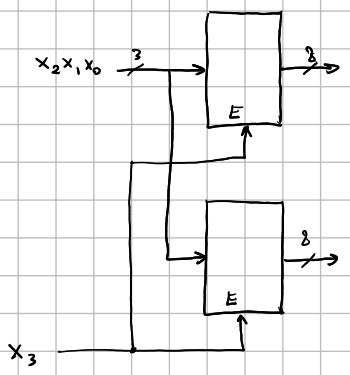
itp.

oznaczenie

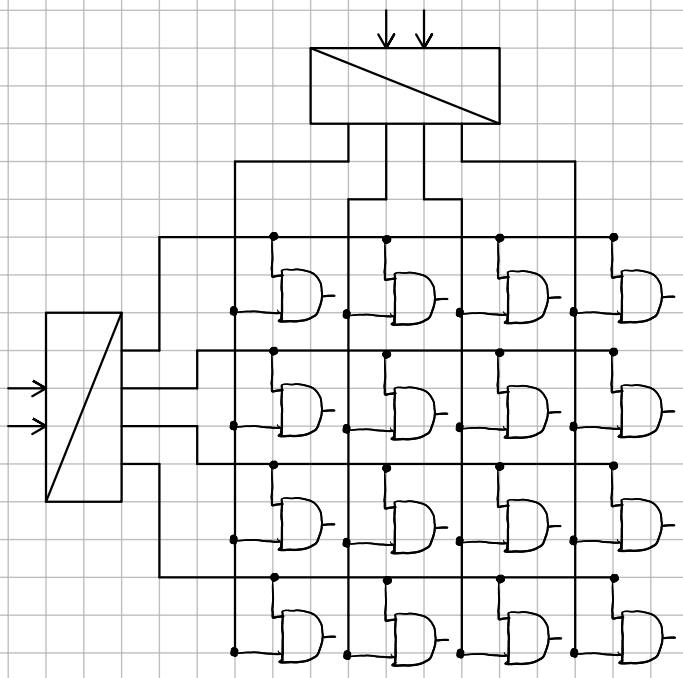


3 bity wejścia
8 bitów wyjścia
sygnał Enable (włącza / wyłącza układ)

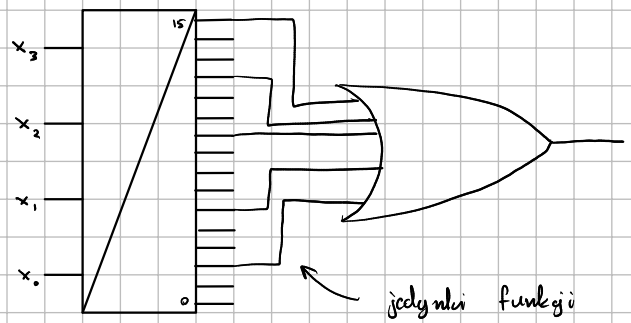
Dekoder 4 na 16 złożony z dwóch 3 na 8



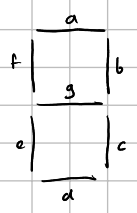
Dekoder współrzędnościowy



Realizacja funkcji logicznej 4 zmiennych na dekoderze 1 z 16



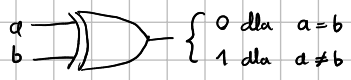
Wyświetlacz 7 segmentowy



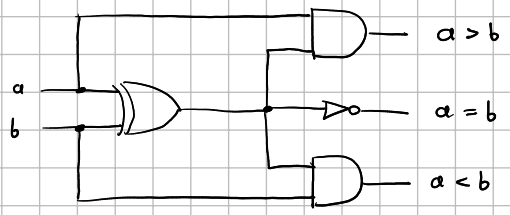
0	1	2	3
4	5	6	7
8	9	A	B
C	D	E	F

→ dekoder z 4-bitowym wejściem

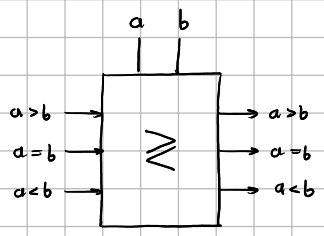
Komparator



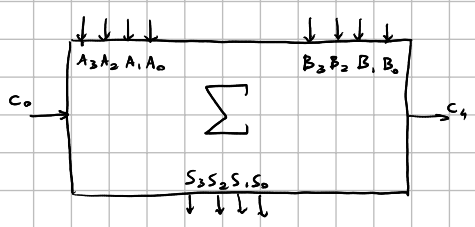
Komparator jednobitowy



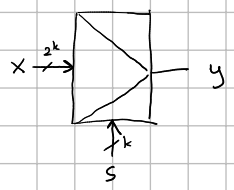
Blok iteracyjny komparatora



Sumator 4-bitowy

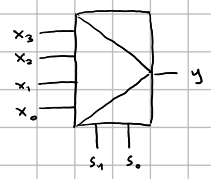


Multiplekser



Wybiera jedno z wejść (X) za pomocą sygnału sterującego (S) i wyprowadza je na wyjście (y)

multiplekser $4 \rightarrow 1$

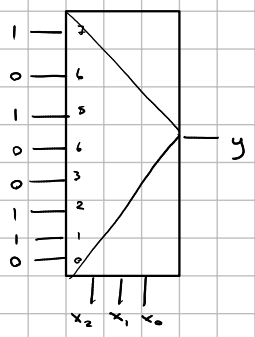


s_1	s_0	y
0	0	x_0
0	1	x_1
1	0	x_2
1	1	x_3

Realizacja funkcji logicznej 3 zmiennych na multiplekserze $8 \rightarrow 1$

x_2 \ x_1x_0	00	01	11	10
0	0	1	0	1
1	0	1	1	0

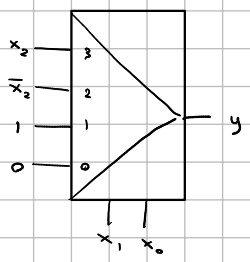
y



Realizacja funkcji logicznej 3 zmiennych na multiplekserze $4 \rightarrow 1$

x_2 \ x_1x_0	00	01	11	10
0	0	1	0	1
1	0	1	1	0

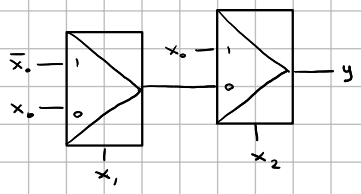
y



Realizacja funkcji logicznej 3 zmiennych na multiplekserach $2 \rightarrow 1$

x_2 \ x_1x_0	00	01	11	10
0	0	1	0	1
1	0	1	1	0

y



Na jednym MUX $2^n \rightarrow 1$
zawsze można zrealizować funkcję n - zmiennych
(stałe 0 albo 1 na wejściach informacyjnych)

Na jednym MUX $2^{n-1} \rightarrow 1$
zawsze można zrealizować funkcję n - zmiennych
(stałe 0 albo 1 lub jedna ze zmiennych na wejściach informacyjnych)

Na jednym MUX $2^{n-2} \rightarrow 1$
nie zawsze można zrealizować funkcję n - zmiennych

Wybór wyjść sterujących

1) Zliczyć literały w APN
2) Wybrać te wystąpujące najczęściej

jeśli realizacja jest możliwa to da dobrą odpowiedź

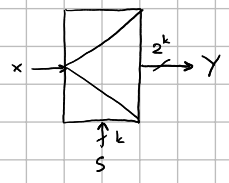
x_3x_2 \ x_1x_0	00	01	11	10
00	1	1	1	0
01	0	1	1	0
11	0	0	1	1
10	0	0	1	1

$y = x_3x_1 + \bar{x}_3x_0 + \bar{x}_3\bar{x}_2\bar{x}_1$

$x_3 \rightarrow 3$
$x_2 \rightarrow 1$
$x_1 \rightarrow 2$
$x_0 \rightarrow 1$

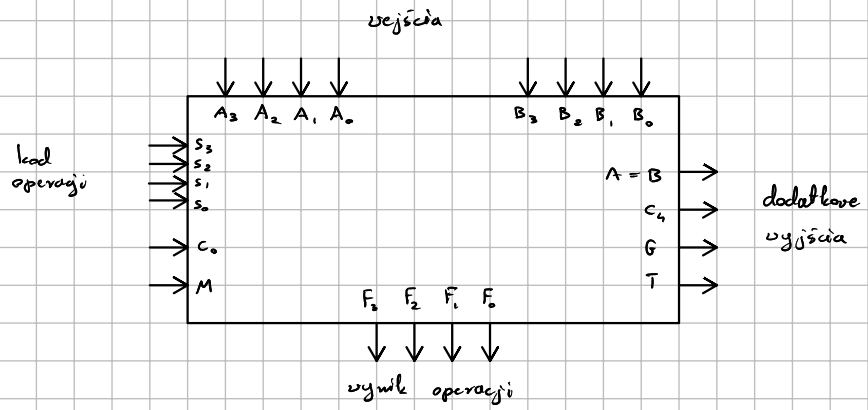
x_3x_1 na wejście sterujące

Demultiplekser

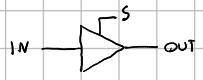


Przekazuje wejście (x) na jedno z wyjść (Y) wybrane sygnałem sterującym (S)

Jednostka arytmetyczno-logiczna (ALU)
(jest układem kombinacyjnym)

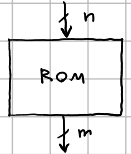


Bramka trójstanowa



Fizycznie odcina część układu na podstawie sygnału sterującego

Pamięć ROM (read-only memory)



Tablica 2^n wierszy
W każdym wierszu jest m-bitowe słowo
Adres n-bitowy wskazuje wiersz tablicy