Rozdział 4. Bloki funkcjonalne układów cyfrowych – Zadania

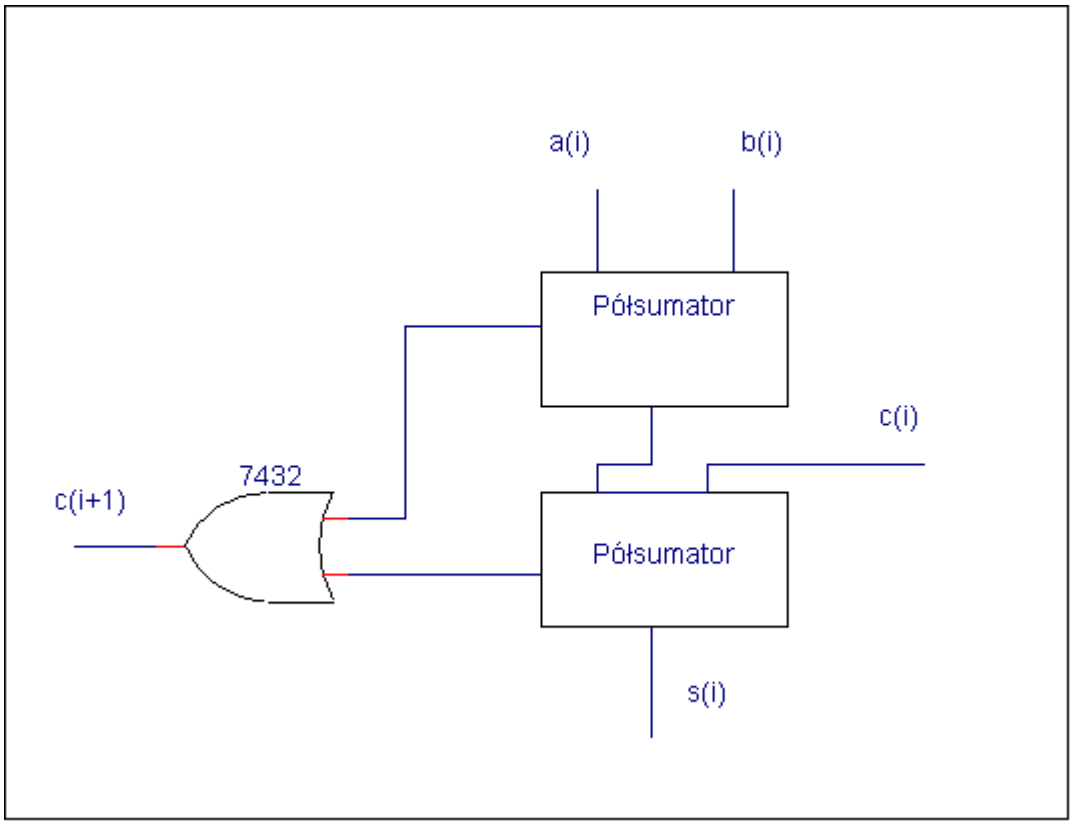
Zadanie 4.1

Zrealizować z bramek multiplekser z szerokością multipleksowanych słów $n = 1$ i $k = 2$-bitowym wejściem adresowym.

Zadanie 4.2

Mamy 2 półsumatory i bramkę sumy logicznej. Zbudować z tych układów sumator jednobitowy pełny.

Rozwiązanie:



Rys. 1 Sumator jednobitowy pełny zbudowany z półsumatorów

Zadanie 4.3

Zrealizować za pomocą multipleksera z 8-mioma wejściami informacyjnymi i trzema adresowymi następujące funkcje boolowskie:

a) $f(x_1, x_2, x_3) = (x_1\bar{x}_2 + \bar{x}_1 x_2)\bar{x}_3$

b) $f(x_1, x_2, x_3) = x_1\bar{x}_2 x_3 + x_2$

c) $f(x_1, x_2, x_3) = x_1\bar{x}_2 x_3 + x_2\bar{x}_1$

d) $f(x_1, x_2, x_3) = x_1\bar{x}_2 x_3 \oplus x_2$

e) $f(x_1, x_2, x_3) = x_1\bar{x}_2 \oplus x_3 \oplus x_2$

f) $f(x_1, x_2, x_3) = x_1 \oplus \bar{x}_2 \oplus x_3 \oplus x_2$

Zadanie 4.4

Zaprojektować translator kodu BCD na kod 7-mio segmentowy (kod sterujący wyświetlaczem siedmiosegmentowym).

Zadanie 4.5

Mamy sumator 8-mio bitowy w postaci układu scalonego (nie mamy dostępu do bitów przeniesień wewnątrz układu sumatora). Zaprojektować układ kombinacyjny ustawiający znaczniki OF (Overflow Flag), CF (carry flag), AF (Auxiliary Carry Flag), PF (Parity Flag) i ZF (Zero Flag).

Zadanie 4.6

To samo co w zadaniu 2.5 ale przy założeniu, że mamy dostęp do bitów przeniesień wewnątrz sumatora.

Zadanie 4.7

Zaprojektować translator kodu temperaturowego z 32 poziomami (kod stosowany w przetwornikach A/D typu flash) na kod NKB.

Zadanie 4.8

Zaprojektować 8 wejściowy koder priorytetowy tzn. układ podający pozycję najwyżej położonej jedynki na 8 bitowym wejściu.

Zadanie 4.9

Zaprojektować sumator 4 bitowy z multiplekserów z wejściami adresowymi 3 bitowymi.

Zadanie 4.10

Zrobić sumator 8- mio bitowy z multiplekserów o 5 bitowym wejściu adresowym.

Zadanie 4.11

Zaprojektować układ kombinacyjny zamieniający
a) 8 bitowy kod NKB na 8 bitowy kod Gray'a
b) 8 bitowy kod Gray'a na 8 bitowy kod NKB

Zadanie 4.12

Zaprojektować układ kombinacyjny obliczający sumę modulo m tzn. $y = x_1 \oplus_m x_2$, gdzie $x_1, x_2 \in Z_m$. Zaprojektować układ dla dowolnego m i dla przypadku gdy liczby $x_1, x_2 \in Z_m$ i m są reprezentowane za pomocą słów 8 bitowych.

Zadanie 4.13

Załóżmy, że mamy scalony 16 bitowy układ mnożący (wejścia są 8-bitowe, wyjście 16-bitowe) używając tego układu zaprojektować układ kombinacyjny obliczający iloczyn modulo m tzn. $y = x_1 \otimes_m x_2$, gdzie $x_1, x_2 \in Z_m$, $2 \le m < 256$ i liczby te są reprezentowane za pomocą słów 8-bitowych.

Zadanie 4.14

Znaleźć funkcję boolowską realizowaną przez układ z rys.1 i przedstawić ją w postaci normalnej dysjunkcyjnej.

Zadanie 4.15

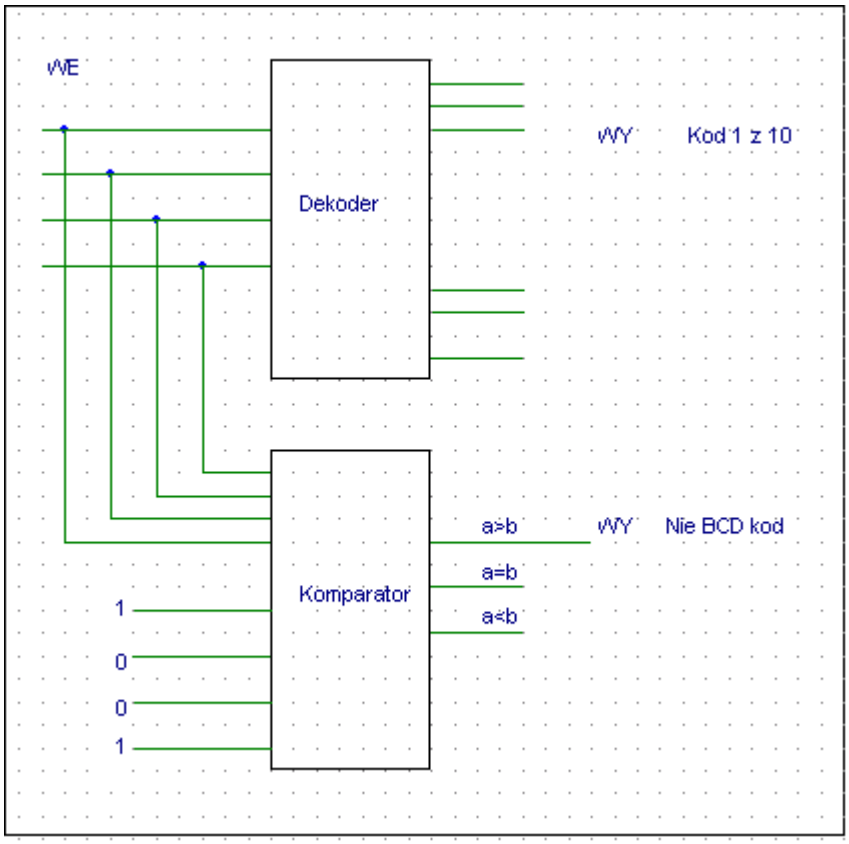
Zaprojektować układ ustawiający znacznik nadmiaru w 8 bitowym zapisie moduł znak.

Zadanie 4.16

Zaprojektować z bloków funkcjonalnych translator kodu BCD 8421 na kod BCD 1 z 10. Wyposażyć układ w układ kontroli tego czy 4 bitowe słowo wejściowe jest słowem kodowym kodu BCD 8421.

Rozwiązanie

Rozwiązanie wykorzystujące 4-bitowy dekoder i komparator pokazane jest na Rys. 2. Komparator porównuje 4-bitowe słowo wejściowe ze słowem 1001 (liczbą 9 w kodzie BCD).

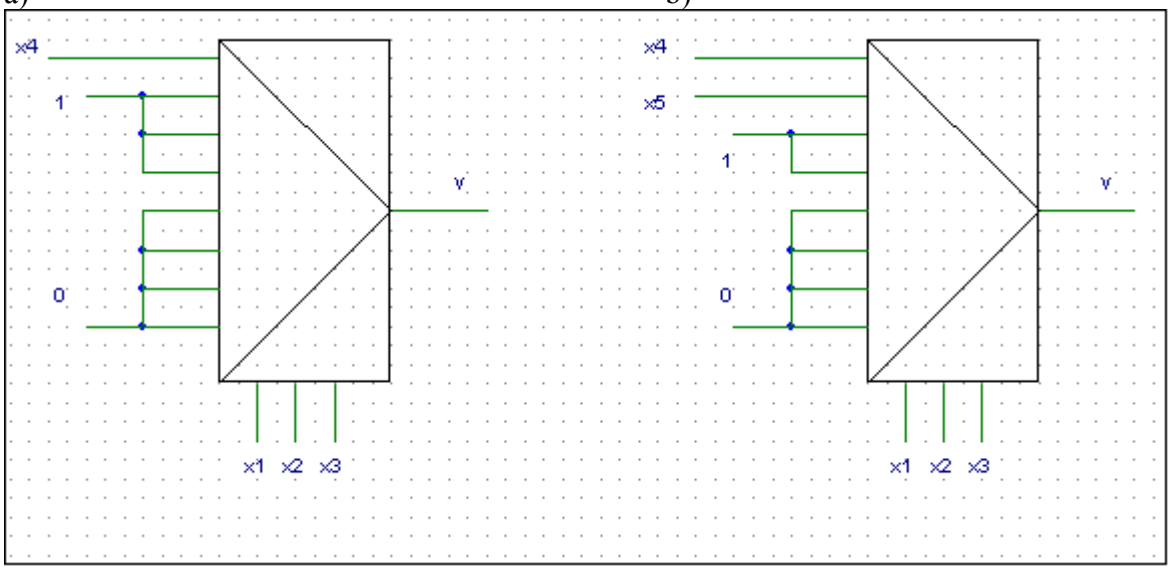


Rys. 2. Translator kodu BCD 8421 na kod BCD 1 z 10; sygnał „Nie BCD kod”=1, jeśli podane na wejście układu słowo nie jest słowem kodowym kodu BCD 8421

Zadanie 4.17

Jaką funkcję boolowską realizują układy pokazane na Rys. 3.

a) b)



Rys. 3. Multipleksery realizujące pewną funkcję boolowską

Rozwiązanie:

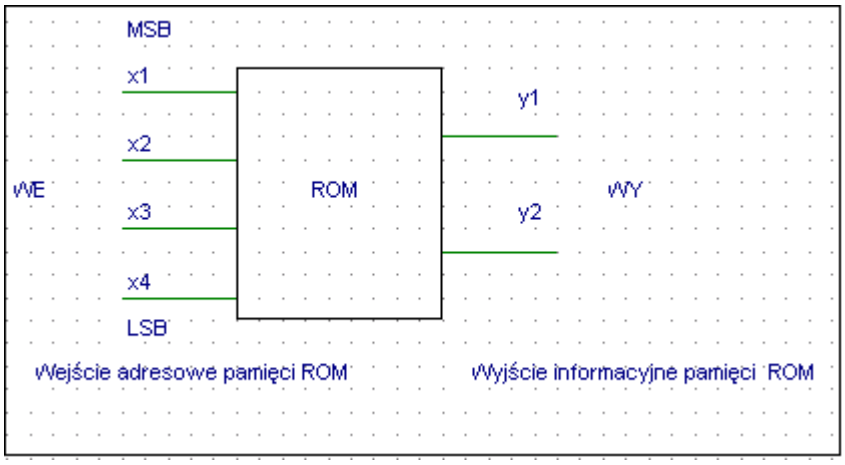
Układ z Rys. 4.2 a) realizuje funkcję:

$$y = f(x_1, x_2, x_3, x_4) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 + \overline{x_1} \cdot \overline{x_2} \cdot x_3 + \overline{x_1} \cdot x_2 \cdot \overline{x_3} + \overline{x_1} \cdot x_2 \cdot x_3$$

Układ z Rys. 4.2 b) realizuje funkcję:

$$y = f(x_1, x_2, x_3, x_4, x_5) = \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 + \overline{x_1} \cdot \overline{x_2} \cdot x_3 \cdot x_5 + \overline{x_1} \cdot x_2 \cdot \overline{x_3} + \overline{x_1} \cdot x_2 \cdot x_3$$

Zadanie 4.18



Rys. 4. Realizacja funkcji boolowskiej za pomocą pamięci ROM.

W naturalny sposób do realizacji funkcji boolowskiej można wykorzystać pamięć ROM (por. rys. 4.). Nie jest to metoda oszczędna, ale umożliwia na ogół łatwą realizację i modyfikację funkcji boolowskiej, np. w przypadku zastosowania pamięci typu flash. Podać wypełnienie pamięci ROM takie, by układ z rys. 4 generował funkcje:

$$y_1 = f_1(x_1, x_2, x_3, x_4) = x_1 \cdot x_2 \cdot x_3 \cdot x_4 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4}$$

$$y_2 = f_2(x_1, x_2, x_3, x_4) = x_1 \cdot x_2 \cdot x_3 \cdot x_4 + \overline{x_1} \cdot \overline{x_2}$$

Zastosowana pamięć ROM ma wejście adresowe 4- bitowe i wyjście informacyjne 2-bitowe.

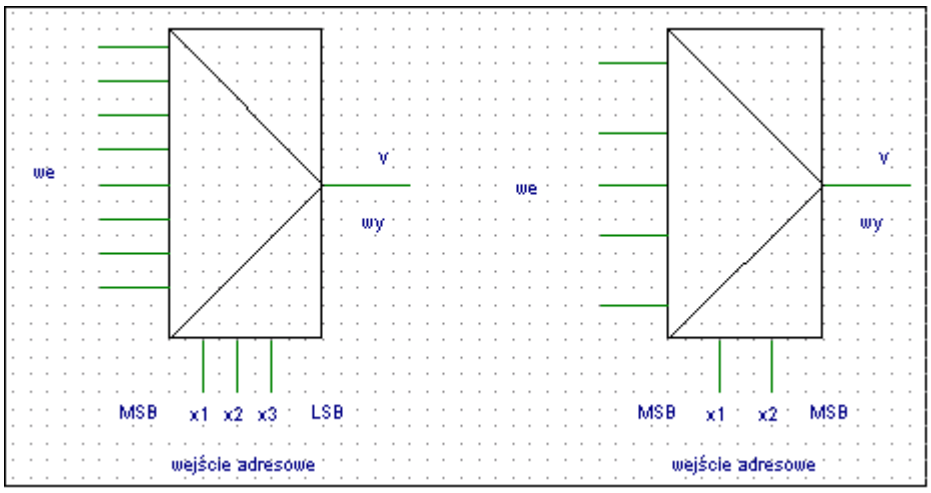
Rozwiązanie:
Zawartość pamięci ROM opisuje tabelka z Rys. 5. Jest to tabelka zawierająca jednocześnie tabelki prawdy funkcji $y_1 = f_1(x_1, x_2, x_3, x_4) = x_1 \cdot x_2 \cdot x_3 \cdot x_4 + \overline{x_1} \cdot \overline{x_2} \cdot \overline{x_3} \cdot \overline{x_4}$ i funkcji. $y_2 = f_2(x_1, x_2, x_3, x_4) = x_1 \cdot x_2 \cdot x_3 \cdot x_4 + \overline{x_1} \cdot \overline{x_2}$.

WEJŚCIA ADRESOWE PAMIĘCI ROM WYJŚCIA NFORMACYJNE

x1	x2	x3	x4	y1	y2
0	0	0	0	1	1
0	0	0	1	0	1
0	0	1	0	0	1
0	0	1	1	0	1
0	1	0	0	0	0
0	1	0	1	0	0
0	1	1	0	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	0	0	1	0	0
1	0	1	0	0	0
1	0	1	1	0	0
1	1	0	0	0	0
1	1	0	1	0	0
1	1	1	0	0	0
1	1	1	1	1	1

Rys.5 Tabelka opisująca zawartość pamięci ROM

Zadanie 4.19



Rys. 6. Multipleksery wykorzystywane do realizacji funkcji boolowskiej

Mamy dwa multipleksery jeden z 3 bitowym słowem adresowym a drugi z 2-bitowym słowem adresowym por. Rys. 6. Zrealizować za pomocą tych 2 multiplekserów funkcję boolowską $f:\{0,1\}^5 \to \{0,1\}$ zadaną wzorem

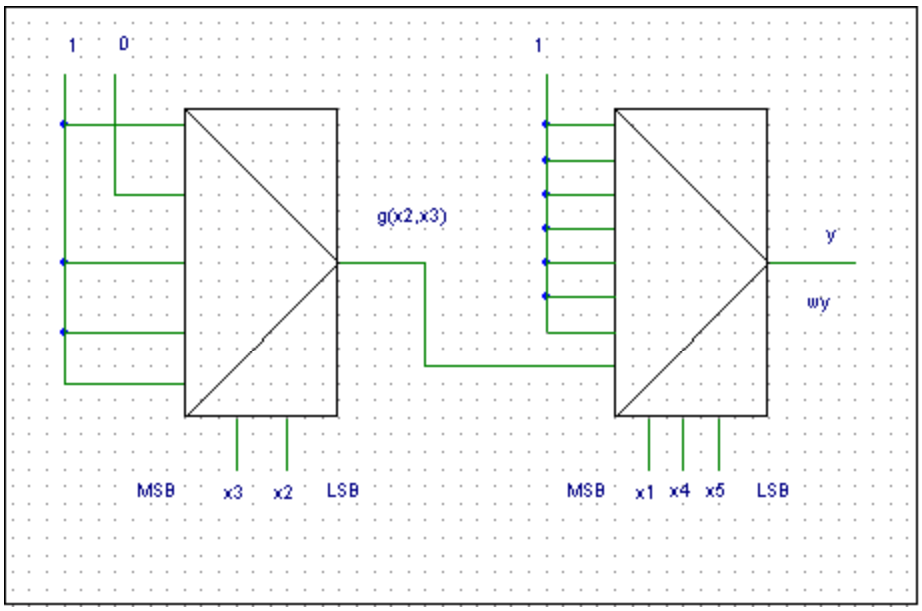
$$y = f_1(x_1, x_2, x_3, x_4, x_5) = x_1 \cdot \overline{x_2} \cdot x_4 \cdot x_5 + x_1 \cdot x_3 \cdot x_4 \cdot x_5$$

Rozwiązanie:

Wprowadźmy funkcję boolowską $g:\{0,1\}^2 \to \{0,1\}$, gdzie $g(a,b) = \overline{a} + b$, wówczas

$$y = f_1(x_1, x_2, x_3, x_4, x_5) = x_1 \cdot \overline{x_2} \cdot x_4 \cdot x_5 + x_1 \cdot x_3 \cdot x_4 \cdot x_5 = (\overline{x_2} + x_3) \cdot x_1 \cdot x_4 \cdot x_5 = g(x_2, x_3) \cdot x_1 \cdot x_4 \cdot x_5$$

Z powyższej równości wynika jak zrealizować za pomocą 2 multiplekserów z Rys. 6. funkcję $y = f_1(x_1, x_2, x_3, x_4, x_5)$. Szczegóły pokazane są na Rys. 7.



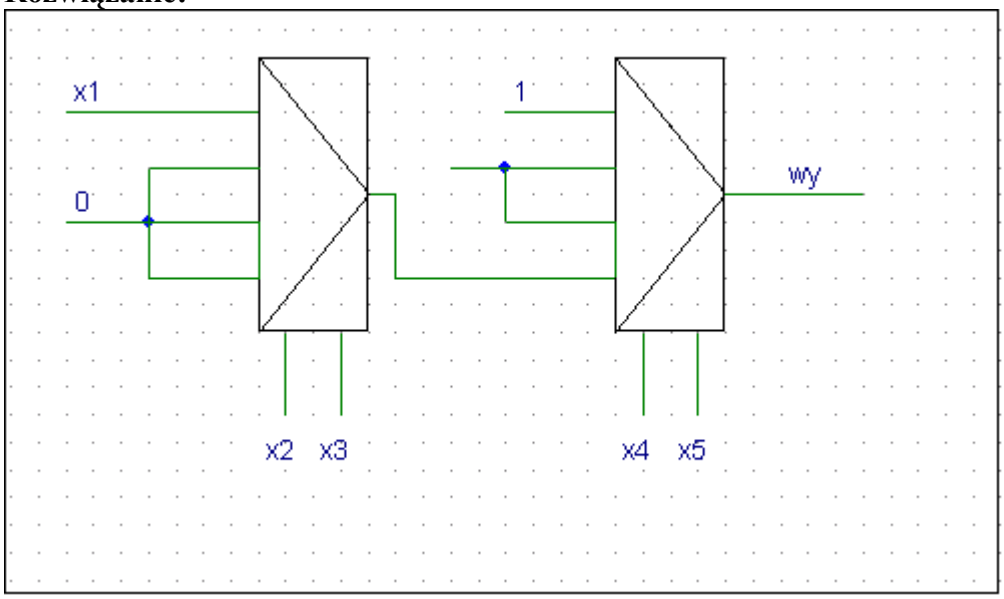
Rys. 7. Realizacja funkcji boolowskiej $f_1(x_1, x_2, x_3, x_4, x_5) = x_1 \cdot \overline{x_2} \cdot x_4 \cdot x_5 + x_1 \cdot x_3 \cdot x_4 \cdot x_5$

Zadanie 4.20

Mamy tylko dwa multipleksery z 2 bitowym słowem adresowym. Zrealizować za pomocą tych 2 multiplekserów funkcję boolowską $f:\{0,1\}^5 \to \{0,1\}$ zadaną wzorem

$$y = f_1(x_1, x_2, x_3, x_4, x_5) = x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 \cdot x_5 + \overline{x_4} \cdot \overline{x_5}$$

Rozwiązanie:

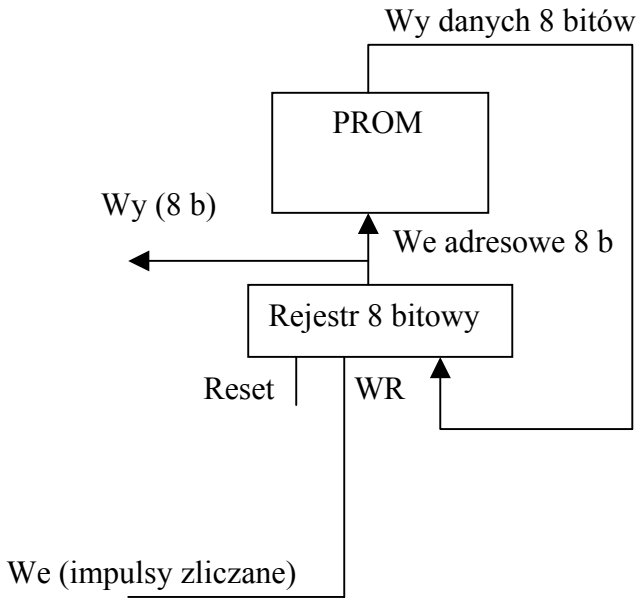


Rys. 8. Realizacja funkcji boolowskiej $y = f_1(x_1, x_2, x_3, x_4, x_5) = x_1 \cdot \overline{x_2} \cdot \overline{x_3} \cdot x_4 \cdot x_5 + \overline{x_4} \cdot \overline{x_5}$ **za pomocą 2 multiplekserów z 2 bitowymi wejściami adresowymi**

Zadanie 4.21

Mamy 8 bitowy rejestr równoległo-równoległy i pamięć PROM z 8 bitowym wejściem adresowym i 8 bitowym słowem danych. Zaprojektować 2 dekadowy licznik w kodzie BCD.

Rozwiązanie:
Na rys. 9 pokazana jest struktura licznika



Rys. 9. Dwudekadowy licznik w kodzie BCD

Zawartość pamięci PROM Czytelnik sam łatwo ustali. Początek tabelki opisującej zawartość pamięci PROM jest następujący.

PROM

Adres	Zawartość komórki	
0	0000	0001
1	0000	0010
2	0000	0011
3	0000	0100
itd		

Rys. 10 Zawartość pamięci PROM układu

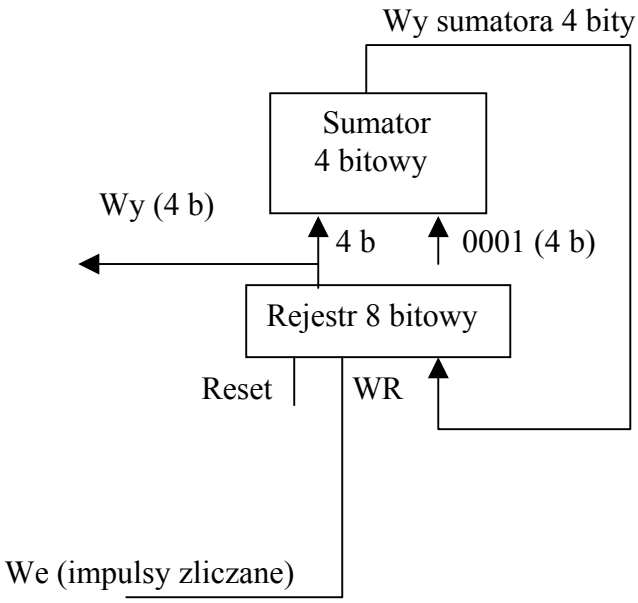
Zadanie 4.22

Zaprojektować a) licznik modulo 5 b) licznik modulo 7 c) licznik synchroniczny modulo 5 d) licznik synchroniczny modulo 7.

Zadanie 4.23

Mamy 4-bitowy rejestr i 4-bitowy sumator zaprojektować wykorzystując te układy licznik mod 16.

Rozwiązanie:
Przykładowe rozwiązanie układu licznika pokazane jest na rys. 11.

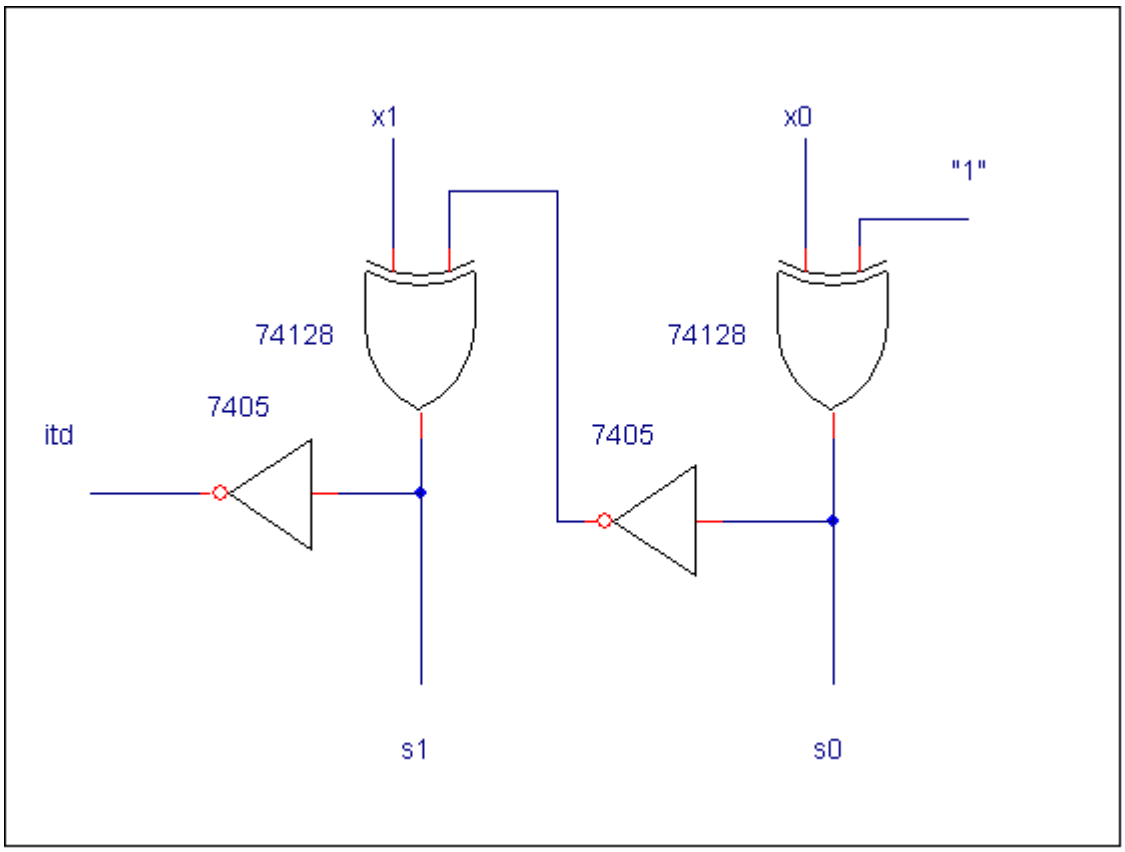


Rys. 11. Licznik modulo 16 wykorzystujący rejestr i sumator

Zadanie 4.24

Mamy jeden 4_bitowy rejestr równoległo-równoległy, 2_wejściowe bramki EXOR (ile trzeba) i inwertery. (ile trzeba). Zaprojektować wykorzystując te układy licznik mod 16.

Rozwiązanie:
Przykładowe rozwiązanie układu licznika pokazane jest na rys. 11 z tym, że zamiast sumatora stosujemy układ inkrementacji o 1 zbudowany za pomocą sum modulo 2 i inwerterów (rys. 12).



Rys. 12. Układ inkrementacji o 1

Zadanie 4.25

Zaprojektować układ określający liczbę jedynek w podawanym na wejście układu słowie.

Zadanie 4.26

Zaprojektować układ podający na 1 bitowe wyjście jedynkę dokładnie wtedy gdy w słowie wejściowym mamy k jedynek. Liczba k jest podawana na wejście programujące. Zaprojektować wersję ze słowem podawanym równolegle (8 bitów w słowie) i podawanym szeregowo (dowolna liczba bitów w słowie), k jest zapisywane za pomocą 8 bitowego słowa..

Zadanie 4.27

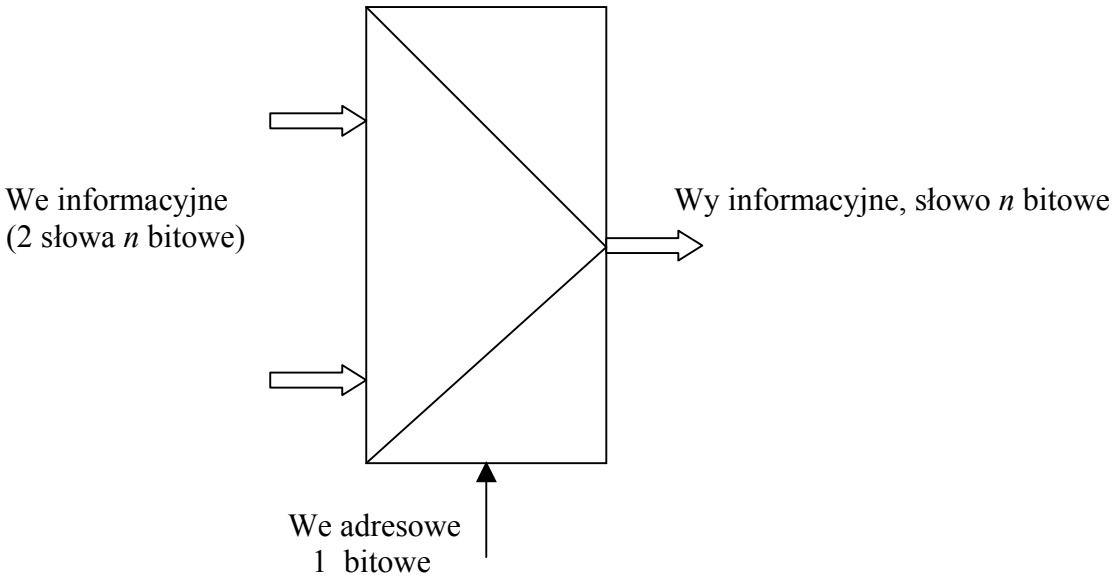
Mamy jeden 4 bitowy rejestr równoległo-równoległy i bramki EXOR 2 wejściowe (dowolną ilość) . Zaprojektować układ liczący w 4 bitowym kodzie Graya.

Zadanie 4.28

Mamy jeden 4 bitowy licznik (modulo 2^4) i bramki EXOR 2 wejściowe (dowolną ilość). Zaprojektować układ liczący w 4 bitowym kodzie Graya.

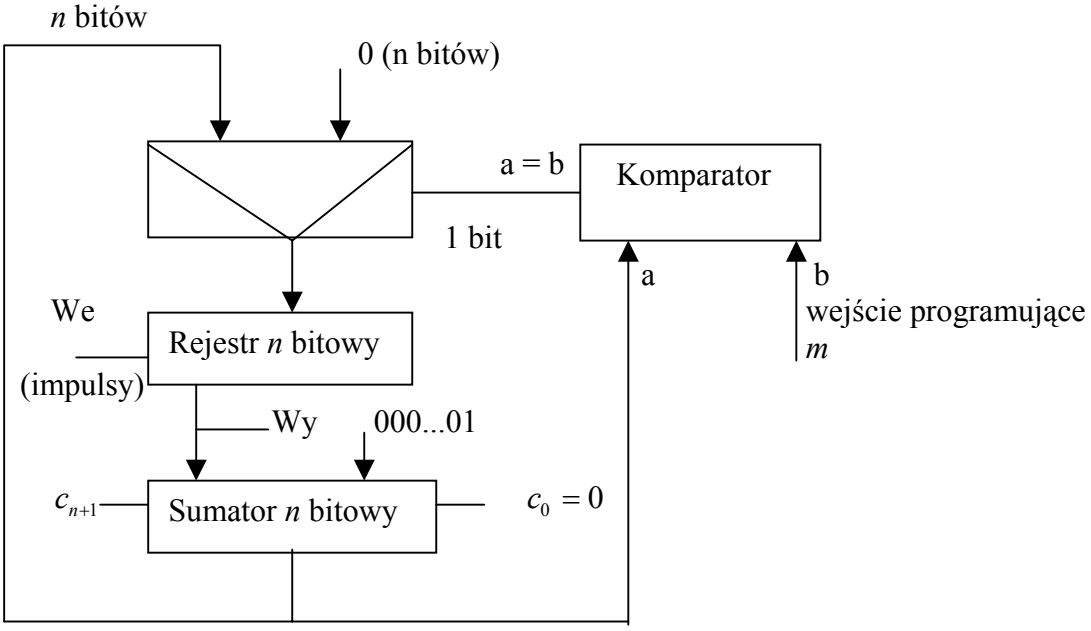
Zadanie 4.29

Zaprojektować programowany licznik synchroniczny modulo $m \le 2^n$. Dysponujemy rejestrem n bitowym, n bitowym sumatorem, n bitowym komparatorem i n bitowym multiplekserem z wejściem adresowym 1 bitowym por rys. 13.



Rys. 13. *n*-bitowy multiplekser z wejściem adresowym 1-bitowym

Rozwiązanie:
Schemat układu programowanego licznika modulo *m* pokazany jest na rys.14.



Rys.14. Układ programowanego licznika modulo *m*