Politechnika Poznańska

Instytut Automatyki, Robotyki i Inżynierii Informatycznej

Zakład Sterowania i Elektroniki Przemysłowej

Układy kombinacyjne

Technika cyfrowa i mikroprocesorowa

Materiały do zajęć laboratoryjnych

dr inż. Dominik Łuczak

Dominik.Luczak@put.poznan.pl

I. CEL

WIEDZY

Celem zajęć jest zapoznanie z:

- algebrą Boole'a,
- zapisem funkcji logicznych za pomocą opisu: słownego, algebraicznego oraz z wykorzystaniem matrycy logicznej (tablica prawdy),
- zapisem funkcji logicznych z wykorzystaniem tablic Karnaugha,
- minimalizacją funkcji logicznych,
- implementacją układów kombinacyjnych za pomocą bramek logicznych oraz języka wysokiego poziomu.

UMIEJĘTNOŚCI

Celem zajęć jest nabycie umiejętności pozwalających na programową i sprzętową realizację układów kombinacyjnych:

- minimalizacją funkcji logicznych,
- implementacją układów kombinacyjnych z wykorzystaniem języka wysokiego poziomu,
- implementacją sprzętowa układów kombinacyjnych z wykorzystaniem bramek logicznych (SSI – ang. small scale integration),
- implementacją sprzętowa układów kombinacyjnych z wykorzystaniem układów średniej skali integracji (MSI – ang. medium-scale integration),
- implementacją sprzętowa układów kombinacyjnych z wykorzystaniem układów dużej skali integracji (LSI ang. large scale integration).

KOMPETENCJI SPOŁECZNYCH

Celem zajęć jest kształtowanie właściwych postaw programistycznych:

- tworzenie kodu programu w sposób czytelny,
- stosowanie dyrektyw preprocesora oraz makroinstrukcji,
- stosowanie stosownych komentarzy,
- minimalizacja zużycia zasobów systemowych,
- weryfikacja poprawności działania systemu.

II. POLECENIA KOŃCOWE

Dokończ zadania nie zrealizowane w trakcie zajęć. Przygotuj raport z przeprowadzonych zajęć. Jeden raport może być przygotowany przez maksymalnie 2 osoby. W projekcie zamieść:

1. Opis rozwiązania problemu oraz jego implementację.
2. Rysunki lub fotografie ilustrujące poprawne działanie programu.
3. Opis działania własnych programów. Udokumentuj przeprowadzenie testów funkcjonalnych z wykorzystaniem symulatora.
4. Ważne instrukcje programu należy zaopatrzyć stosownym komentarzem. Wszystkie przygotowane funkcje oraz zmienne globalne należy zaopatrzyć komentarzem zgodnym z składnią generatora dokumentacji (*Doxygen*).

Jako załącznik dołącz spakowane kody programów. Dla każdego zadania przygotuj osobny projekt. Raport należy przygotować i przekazać do kolejnego spotkania. Raporty przekazywane później nie będą oceniane.

Na ocenę raportu będą miały wpływ następujące elementy (łącznie 5 punktów):

1. spełnienie wymogów redakcyjnych (1p),
2. wykonanie, udokumentowanie oraz opis wykonanych zadań (2p),
3. zastosowanie prawidłowego warsztatu programistycznego (2p).

III. SCENARIUSZ DO ZAJĘĆ

a) ŚRODKI DYDAKTYCZNE

Sprzętowe:
- komputer,
- zestaw edukacyjny z mikrokontrolerem ADuC831.

Programowe:
- symulatora układów logicznych (Circuit Maker),
- zintegrowane środowisko programistyczne języka C dla mikrokontrolerów (Keil uVision).

b) PRZEBIEG ZAJĘĆ

1. Wygeneruj ręcznie funkcję Boolowską zgodnie z algorytmem opisanym w punkcie IV.b) i zapisz ją na kartce papieru.
2. Wykonaj minimalizację funkcji Boolowskiej (ręcznie na papierze):
    a. Zapisz funkcję Boolowską za pomocą wyrażeń algebraicznych.
    b. Zapisz minimalną postać dysjunkcyjną.
    c. Zapisz minimalną postać koniunkcyjną.
3. Zrealizuj funkcję dla systemu mikroprocesorowego (Keil uVision):
    a. Zapisz funkcję w języku wysokiego poziomu.
    b. Przetestuj działanie programu w symulatorze.
4. Zrealizuj funkcję sprzętowo (Circuit Maker):
    a. W układzie małej skali integracji SSI:
        i. Zapisz funkcję w tylko za pomocą bramek NAND,
        ii. Zapisz funkcję w tylko za pomocą bramek NOR,
        iii. Zapisz funkcję w tylko za pomocą dowolnych bramek,
    b. W układzie średniej skali integracji MSI:
        i. MUX(2)-NAND,
        ii. NAND-DMUX,
        iii. MUX-NAND-DMUX,
    c. Dużej skali integracji LSI wykorzystując układ pamięci RAM.
    d. Przetestuj działanie układu.

IV. PRZYGOTOWANIE DO ZAJĘĆ

a) ZAPOZNANIE Z PRZEPISAMI BHP

Wszystkie informacje dotyczące instrukcji BHP laboratorium są zamieszczone w sali laboratoryjnej oraz u prowadzącego zajęcia. Wszystkie nieścisłości należy wyjaśnić z prowadzącym laboratorium. Wymagane jest zaznajomienie i zastosowanie do regulaminu.

Na zajęcia należy przyjść przygotowanym zgodnie z tematem zajęć. Obowiązuje również materiał ze wszystkich odbytych zajęć.

b) ALGORYTM WYGENEROWANIA FUNKCJI BOOLOWSKIEJ

Wejście: nazwisko, imię.

Wyjście: funkcja przełączająca f(x_1, x_2, x_3, x_4).

Metoda:

1) Do tabeli Karnaugha dla funkcji czterech zmiennych wpisz 16 liter alfabetu polskiego w sposób określony w tabeli Tab. 1.
2) Do każdej komórki tabeli Karnaugha:
   a) Wpisz „1" , jeżeli zawarta w niej litera występuje w nazwisku lub imieniu,
   b) Wpisz „0" , w przeciwnym przypadku.
3) Wartości wpisane do tabeli w kroku 2 określają funkcję przełączającą f(x_1, x_2, x_3, x_4).

Tab. 1 Generacja danych – tabela Karnaugha

		0	1	3	2	
		00	01	11	10	x_3x_4
0	00	A	F	G	M	
1	01	B	C	H	O	
3	11	Ć	D	J	K	
2	10	P	E	L	R	
	x_1x_2					

c) MINIMALIZACJA FUNKCJI LOGICZNEJ

Funkcja logiczna (boolowska) *y* może zostać zapisana w postaci tabeli prawdy, która każdej kombinacji wejść przyporządkowuje jedną wartość logiczną. Przykładową funkcję logiczną o czterech wejściach przedstawiono w Tab. 2. Funkcja zostanie użyta w dalszych przykładach. Zmieniając adresację w taki sposób by adres zmieniał się tylko na jednej pozycji bitu uzyskano tablicę Karnaugh (Tab. 3). Użyta w przykładzie funkcja logiczna przyjmuje wartość logiczną 1-prawda dla adresów {0, 1, 6, 8, 9, 10, 12, 14}, natomiast wartość logiczną 0-fałsz dla adresów {2, 3, 4, 5, 7, 11, 13, 15}.

Tab. 2 Przykładowa funkcja logiczna – zapis naturalny

		0	1	2	3	
		00	01	10	11	x_3x_4
0	00	1	1	0	0	
1	01	0	0	1	0	
2	10	1	1	1	0	
3	11	1	0	1	0	
	x_1x_2					

Tab. 3 Przykładowa funkcja logiczna zapisana w tabeli Karnaugh

		0	1	3	2	
		00	01	11	10	x_3x_4
0	00	1	1	0	0	
1	01	0	0	0	1	
3	11	1	0	0	1	
2	10	1	1	0	1	
	x_1x_2					

POSTAĆ DYSJUNKCYJNA

Funkcję logiczną można zapisać za pomocą postaci dysjunkcyjnej – zapis „jedynek" (patrz Tab. 4). Bezpośrednia realizacja takiego zapisu za pomocą bramek logicznych wymaga zastosowania dużej liczby bramek logicznych. W celu redukcji liczby wyrażeń algebraicznych (liczby potrzebnych bramek do realizacji) dokonuje się minimalizacji. Postać dysjunkcyjna (1) zapisana bezpośrednio na podstawie danych z

Tab. 3 składa się z ośmiu składowych {0, 1, 6, 8, 9, 10, 12, 14}.

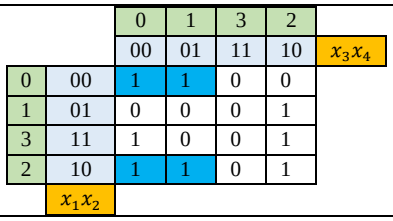
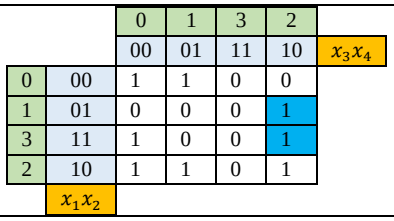
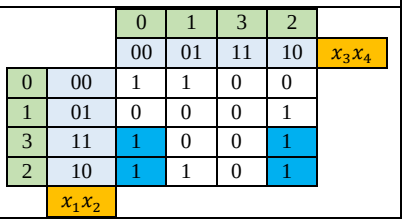
Tab. 4 Realizacja funkcji logicznej względem "1"

		0	1	3	2	
		00	01	11	10	x_3x_4
0	00	1	1	0	0	
1	01	0	0	0	1	
3	11	1	0	0	1	
2	10	1	1	0	1	
	x_1x_2					

$$y = y_{PD} = \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot \bar{x}_4 + \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot x_4 + \bar{x}_1 \cdot x_2 \cdot x_3 \cdot \bar{x}_4 + \\ + x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot \bar{x}_4 + x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot x_4 + x_1 \cdot \bar{x}_2 \cdot x_3 \cdot \bar{x}_4 + \\ + x_1 \cdot x_2 \cdot \bar{x}_3 \cdot \bar{x}_4 + x_1 \cdot x_2 \cdot x_3 \cdot \bar{x}_4 \tag{1}$$

Minimalna postać dysjunkcyjna (2) składa się z trzech składowych przedstawionych w Tab. 5. W tabeli przedstawiono wyniki cząstkowe oraz zaznaczono, których „jedynek" dotyczyła operacja sklejania.

Tab. 5 Składowe MPD

	Składowa pierwsza	Składowa druga	Składowa trzecia
Zapis matematyczny	$\bar{x}_2 \cdot \bar{x}_3$	$x_2 \cdot x_3 \cdot \bar{x}_4$	$x_1 \cdot \bar{x}_4$
Sklejane jedynki			

$$y = y_{MPD} = \bar{x}_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot (\bar{x}_4 + x_4) + (\bar{x}_1 + x_1) \cdot x_2 \cdot x_3 \cdot \bar{x}_4 + x_1 \cdot x_2 \cdot (\bar{x}_3 + x_3) \cdot \bar{x}_4 + \\ + x_1 \cdot \bar{x}_2 \cdot \bar{x}_3 \cdot (\bar{x}_4 + x_4) + x_1 \cdot \bar{x}_2 \cdot (\bar{x}_3 + x_3) \cdot \bar{x}_4 = \\ = (\bar{x}_1 + x_1) \cdot \bar{x}_2 \cdot \bar{x}_3 + x_2 \cdot x_3 \cdot \bar{x}_4 + x_1 \cdot (\bar{x}_2 + x_2) \cdot \bar{x}_4 = \\ = \bar{x}_2 \cdot \bar{x}_3 + x_2 \cdot x_3 \cdot \bar{x}_4 + x_1 \cdot \bar{x}_4 \tag{2}$$

POSTAĆ KONIUNKCYJNA

Funkcję logiczną można zapisać za pomocą postaci koniunkcyjnej – zapis „zer" (patrz Tab. 6). Postać koniunkcyjna (3) zapisana bezpośrednio na podstawie danych z Tab. 6 składa się z ośmiu składowych {2, 3, 4, 5, 7, 11, 13, 15}.

Tab. 6 Realizacja funkcji logicznej względem "0"

		0	1	3	2	
		00	01	11	10	x_3x_4
0	00	1	1	0	0	
1	01	0	0	0	1	
3	11	1	0	0	1	
2	10	1	1	0	1	
	x_1x_2					

$$y = y_{PK} = (x_1 + x_2 + \bar{x}_3 + x_4)\cdot(x_1 + x_2 + \bar{x}_3 + \bar{x}_4)\cdot(x_1 + \bar{x}_2 + x_3 + x_4)\cdot$$
$$\cdot(x_1 + \bar{x}_2 + x_3 + \bar{x}_4)\cdot(x_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4)\cdot(\bar{x}_1 + x_2 + \bar{x}_3 + \bar{x}_4)\cdot \quad (3)$$
$$\cdot(\bar{x}_1 + \bar{x}_2 + x_3 + \bar{x}_4)\cdot(\bar{x}_1 + \bar{x}_2 + \bar{x}_3 + \bar{x}_4)$$

Minimalna postać koniunkcyjna (4) składa się z czterech składowych przedstawionych w Tab. 7. W tabeli przedstawiono wynik cząstkowy oraz zaznaczono, których „zer" dotyczyła operacja sklejania.

Tab. 7 Składowe MPK

<table>
<tr><td></td><td>Składowa pierwsza</td><td>Składowa druga</td></tr>
<tr><td>Zapis matematyczny</td><td>$(x_1 + x_2 + \bar{x}_3)$</td><td>$(x_1 + \bar{x}_2 + x_3)$</td></tr>
<tr><td>Sklejane jedynki</td><td><table><tr><td></td><td></td><td>0</td><td>1</td><td>3</td><td>2</td><td></td></tr><tr><td></td><td></td><td>00</td><td>01</td><td>11</td><td>10</td><td>x_3x_4</td></tr><tr><td>0</td><td>00</td><td>1</td><td>1</td><td>0</td><td>0</td><td></td></tr><tr><td>1</td><td>01</td><td>0</td><td>0</td><td>0</td><td>1</td><td></td></tr><tr><td>3</td><td>11</td><td>1</td><td>0</td><td>0</td><td>1</td><td></td></tr><tr><td>2</td><td>10</td><td>1</td><td>1</td><td>0</td><td>1</td><td></td></tr><tr><td></td><td>x_1x_2</td><td></td><td></td><td></td><td></td><td></td></tr></table></td><td><table><tr><td></td><td></td><td>0</td><td>1</td><td>3</td><td>2</td><td></td></tr><tr><td></td><td></td><td>00</td><td>01</td><td>11</td><td>10</td><td>x_3x_4</td></tr><tr><td>0</td><td>00</td><td>1</td><td>1</td><td>0</td><td>0</td><td></td></tr><tr><td>1</td><td>01</td><td>0</td><td>0</td><td>0</td><td>1</td><td></td></tr><tr><td>3</td><td>11</td><td>1</td><td>0</td><td>0</td><td>1</td><td></td></tr><tr><td>2</td><td>10</td><td>1</td><td>1</td><td>0</td><td>1</td><td></td></tr><tr><td></td><td>x_1x_2</td><td></td><td></td><td></td><td></td><td></td></tr></table></td></tr>
<tr><td></td><td>Składowa trzecia</td><td>Składowa czwarta</td></tr>
<tr><td>Zapis matematyczny</td><td>$(\bar{x}_2 + \bar{x}_4)$</td><td>$(\bar{x}_3 + \bar{x}_4)$</td></tr>
<tr><td>Sklejane jedynki</td><td><table><tr><td></td><td></td><td>0</td><td>1</td><td>3</td><td>2</td><td></td></tr><tr><td></td><td></td><td>00</td><td>01</td><td>11</td><td>10</td><td>x_3x_4</td></tr><tr><td>0</td><td>00</td><td>1</td><td>1</td><td>0</td><td>0</td><td></td></tr><tr><td>1</td><td>01</td><td>0</td><td>0</td><td>0</td><td>1</td><td></td></tr><tr><td>3</td><td>11</td><td>1</td><td>0</td><td>0</td><td>1</td><td></td></tr><tr><td>2</td><td>10</td><td>1</td><td>1</td><td>0</td><td>1</td><td></td></tr><tr><td></td><td>x_1x_2</td><td></td><td></td><td></td><td></td><td></td></tr></table></td><td><table><tr><td></td><td></td><td>0</td><td>1</td><td>3</td><td>2</td><td></td></tr><tr><td></td><td></td><td>00</td><td>01</td><td>11</td><td>10</td><td>x_3x_4</td></tr><tr><td>0</td><td>00</td><td>1</td><td>1</td><td>0</td><td>0</td><td></td></tr><tr><td>1</td><td>01</td><td>0</td><td>0</td><td>0</td><td>1</td><td></td></tr><tr><td>3</td><td>11</td><td>1</td><td>0</td><td>0</td><td>1</td><td></td></tr><tr><td>2</td><td>10</td><td>1</td><td>1</td><td>0</td><td>1</td><td></td></tr><tr><td></td><td>x_1x_2</td><td></td><td></td><td></td><td></td><td></td></tr></table></td></tr>
</table>

$$y = y_{MPK} = (x_1 + x_2 + \bar{x}_3)\cdot(x_1 + \bar{x}_2 + x_3)\cdot(\bar{x}_2 + \bar{x}_4)\cdot(\bar{x}_3 + \bar{x}_4) \quad (4)$$

d) PROGRAMOWA REALIZACJA FUNKCJI LOGICZNEJ W JĘZYKU C

W celu realizacji funkcji logicznej w systemie mikroprocesorowym należy w pierwszej kolejności określić piny reprezentujące x_1, x_2, x_3, x_4 oraz y. W kolejnym kroku należy bezpośrednio zapisać wyrażenie wykorzystując operatory logiczne języka C (alternatywa - ||, koniunkcja - &&, negacja - !). W efekcie zapisu równania (2) uzyskano kod przedstawiony na Listing 1.

Listing 1 Implementacja wyrażenia logicznego

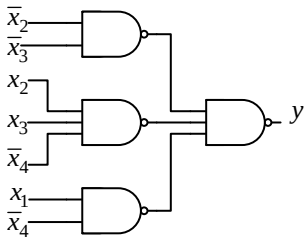
```
/**
 * @author Dominik Luczak
 * @brief Combinational logic.
 */
#include "aduc831.h"          //Definitions of ADuC831 registers name
#include "stdint.h"           //Standard integers
#include "stdfloat.h"         //Standard float
#include "IO.h"               //Input/output definitions

#define X1 BitCheck(P3,3)
#define X2 BitCheck(P3,2)
#define X3 BitCheck(P3,1)
#define X4 BitCheck(P3,0)
#define Yset() BitSet(P3,7)
#define Yreset() BitClear(P3,7)
/**
* Main function
*/
void main(void)
{
          while(1)
          {
                    if( (!X2 && !X3) || (X2 && X3 && !X4) || (X1 && !X4))
                    {
                              Yset();
                    }else{
                              Yreset();
                    }
          }
}
```

e) SPRZĘTOWA REALIZACJA FUNKCJI LOGICZNEJ W UKŁADZIE MAŁEJ SKALI INTEGRACJI Z WYKORZYSTANIEM BRAMEK NAND

Naturalnym sposobem realizacji funkcji logicznej zapisanej w postaci dysjunkcyjnej jest wykorzystanie tylko bramek NAND. Wyrażenie logiczne zapisane za pomocą postaci dysjunkcyjnej jest przekształcane, wykorzystując prawo de Morgana do postaci (5). Ogólnie realizacja funkcji logicznej z wykorzystaniem bramek NAND składa się z dwóch warstw: 1) każdy zbiór iloczynowy literałów odpowiada jednej bramce NAND o liczbie wejść równej liczbie literałów, 2) w warstwie drugiej zawsze występuje jedna bramka NAND (tzw. NAND zbierający), na której wejścia wprowadzane są wszystkie wyjścia z warstwy poprzedniej. Wykonaną realizację dla przykładowej funkcji logicznej *y* przedstawiono na Rys. 1.

$$\begin{aligned} y = y_{MPD} &= \overline{x}_2 \cdot \overline{x}_3 + x_2 \cdot x_3 \cdot \overline{x}_4 + x_1 \cdot \overline{x}_4 = \\ &= \overline{\overline{x}_2 \cdot \overline{x}_3 + x_2 \cdot x_3 \cdot \overline{x}_4 + x_1 \cdot \overline{x}_4} = \\ &= \overline{\overline{\overline{x}_2 \cdot \overline{x}_3} \cdot \overline{x_2 \cdot x_3 \cdot \overline{x}_4} \cdot \overline{x_1 \cdot \overline{x}_4}} \end{aligned} \tag{5}$$

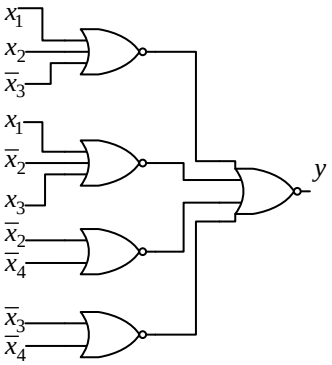


Rys. 1 Realizacja funkcji logicznej z wykorzystaniem bramek NAND

Z WYKORZYSTANIEM BRAMEK NOR

Naturalnym sposobem realizacji funkcji logicznej zapisanej w postaci koniunkcyjnej jest wykorzystanie tylko bramek NOR. Wyrażenie logiczne zapisane za pomocą postaci koniunkcyjnej jest przekształcane, wykorzystując prawo de Morgana do postaci (6). Ogólnie realizacja funkcji logicznej z wykorzystaniem bramek NOR składa się z dwóch warstw: 1) każdy zbiór iloczynowy literałów odpowiada jednej bramce NOR o liczbie wejść równej liczbie literałów, 2) w warstwie drugiej zawsze występuje jedna bramka NOR (tzw. NOR zbierający), na której wejścia wprowadzane są wszystkie wyjścia z warstwy poprzedniej. Wykonaną realizację dla przykładowej funkcji logicznej *y* przedstawiono na Rys. 2.

$$\begin{aligned} y = y_{MPK} &= (x_1 + x_2 + \overline{x}_3)\cdot(x_1 + \overline{x}_2 + x_3)\cdot(\overline{x}_2 + \overline{x}_4)\cdot(\overline{x}_3 + \overline{x}_4) = \\ &= \overline{\overline{(x_1 + x_2 + \overline{x}_3)\cdot(x_1 + \overline{x}_2 + x_3)\cdot(\overline{x}_2 + \overline{x}_4)\cdot(\overline{x}_3 + \overline{x}_4)}} = \\ &= \overline{\overline{(x_1 + x_2 + \overline{x}_3)} + \overline{(x_1 + \overline{x}_2 + x_3)} + \overline{(\overline{x}_2 + \overline{x}_4)} + \overline{(\overline{x}_3 + \overline{x}_4)}} \end{aligned} \tag{6}$$



Rys. 2 Realizacji funkcji logicznej z wykorzystaniem bramek NOR

f) SPRZĘTOWA REALIZACJA FUNKCJI LOGICZNEJ W UKŁADZIE ŚREDNIEJ SKALI INTEGRACJI

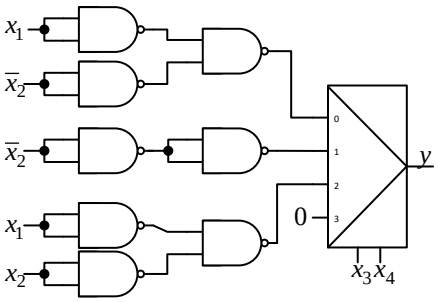
STRUKTURA MUX(2)-NAND

Multiplekser jest układem kombinacyjnym, który służy do wyboru jednego z kilku dostępnych sygnałów wejściowych i przekazania go na wyjście danych. Sygnał wejściowy danych wybierany jest za pomocą wejść adresowych multipleksera. Liczba wejść multipleksera jest zazwyczaj równa $2^{liczba\ wejsc\ adresowych}$. Zastosowanie multipleksera do realizacji funkcji logicznej pozwala na rozłożenie dużej tablicy prawdy na mniejsze tablice o zredukowanej liczbie zmiennych. W przykładzie wykorzystano zmienną x_3 oraz x_4 jako wejście adresowe multipleksera o dwóch wejściach adresowych. W efekcie początkowa tablica prawdy Tab. 3 została rozłożona na cztery tablice z dwoma zmiennymi x_1 oraz x_2. Każdą z nowo powstałych tablic należy zminimalizować oraz zaimplementować. Wyjście każdej funkcji „resztkowej" stanowi wejście do multipleksera. Należy

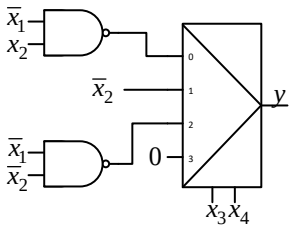
zwrócić szczególną uwagę na sposób adresowania w tabeli Karnaugh by nie pomylić kolejności adresowania multipleksera. Wynik minimalizacji każdej z funkcji resztkowych przedstawiono w Tab. 8. Realizując cząstkowe funkcje logiczne tylko z wykorzystaniem bramek NAND w podstawowym podejściu (bez uwzględnienia redukcji wyrażeń) uzyskano system przedstawiony na Rys. 3. Po przekształceniu opisu do prostszej postaci uzyskano system przedstawiony na Rys. 4, który wykorzystuje mniejszą liczbę bramek logicznych.

Tab. 8 Minimalizacja funkcji resztkowych przy strukturze MUX(2)-NAND

<table>
<tr><td></td><td>Składowa zerowa $x_3x_4 = 00$</td><td>Składowa pierwsza $x_3x_4 = 01$</td></tr>
<tr><td></td><td></td><td></td></tr>
<tr><td>Zapis matematyczny</td><td>$\overline{\overline{x_1 + \overline{x}_2}} = \overline{\overline{x}_1 \cdot x_2}$</td><td>$\overline{x}_2$</td></tr>
<tr><td>Sklejane jedynki</td><td><table><tr><td></td><td></td><td>0</td></tr><tr><td></td><td></td><td>00</td><td>x_3x_4</td></tr><tr><td>0</td><td>00</td><td>1</td></tr><tr><td>1</td><td>01</td><td>0</td></tr><tr><td>3</td><td>11</td><td>1</td></tr><tr><td>2</td><td>10</td><td>1</td></tr><tr><td></td><td>x_1x_2</td><td></td></tr></table></td><td><table><tr><td></td><td></td><td>1</td></tr><tr><td></td><td></td><td>01</td><td>x_3x_4</td></tr><tr><td>0</td><td>00</td><td>1</td></tr><tr><td>1</td><td>01</td><td>0</td></tr><tr><td>3</td><td>11</td><td>0</td></tr><tr><td>2</td><td>10</td><td>1</td></tr><tr><td></td><td>x_1x_2</td><td></td></tr></table></td></tr>
<tr><td></td><td>Składowa druga $x_3x_4 = 10$</td><td>Składowa trzecia $x_3x_4 = 11$</td></tr>
<tr><td>Zapis matematyczny</td><td>$\overline{\overline{x_1 + x_2}} = \overline{\overline{x}_1 \cdot \overline{x}_2}$</td><td>0</td></tr>
<tr><td>Sklejane jedynki</td><td><table><tr><td></td><td></td><td>2</td></tr><tr><td></td><td></td><td>10</td><td>x_3x_4</td></tr><tr><td>0</td><td>00</td><td>0</td></tr><tr><td>1</td><td>01</td><td>1</td></tr><tr><td>3</td><td>11</td><td>1</td></tr><tr><td>2</td><td>10</td><td>1</td></tr><tr><td></td><td>x_1x_2</td><td></td></tr></table></td><td><table><tr><td></td><td></td><td>3</td></tr><tr><td></td><td></td><td>11</td><td>x_3x_4</td></tr><tr><td>0</td><td>00</td><td>0</td></tr><tr><td>1</td><td>01</td><td>0</td></tr><tr><td>3</td><td>11</td><td>0</td></tr><tr><td>2</td><td>10</td><td>0</td></tr><tr><td></td><td>x_1x_2</td><td></td></tr></table></td></tr>
</table>



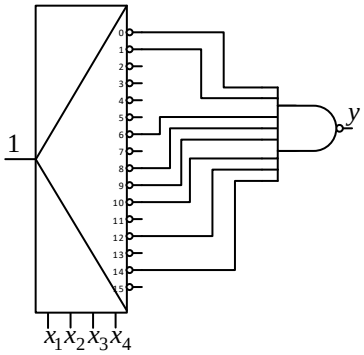
Rys. 3 Realizacja funkcji logicznej z wykorzystaniem struktury MUX-NAND – bezpośrednio wg algorytmu



Rys. 4 Realizacja funkcji logicznej z wykorzystaniem struktury MUX-NAND – po uproszczeniu

STRUKTURA NAND-DMUX

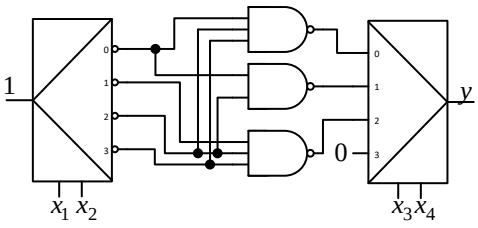
Kolejną strukturą umożliwiającą realizację funkcji logicznej jest struktura NAND-DMUX (wykorzystuje ona małą liczbę bramek logicznych). Demultiplekser jest układem kombinacyjnym, który posiada jedno wejście danych przepisywane na wybrane wyście danych określone przez wejścia adresowe. W tej realizacji wszystkie zmienne funkcji logicznej są podłączone pod wejścia adresowe, natomiast wejście danych jest ustawione na wartość logiczną 1. Dodatkowo wszystkie wyjścia danych są zanegowane. Wyjścia danych demultipleksera, których adresy odpowiadają wartościom logicznym z tabeli prawdy realizowanej funkcji logicznej stanowią wejścia bramki NAND. Wyjście bramki NAND określa wartość funkcji dla podanych wartości x_1, x_2, x_3, x_4. Realizację funkcji logicznej z przykładu (Tab. 2) przedstawia Rys. 5. Wadą przedstawionej struktury jest ograniczona liczby wejść adresowych demultipleksera oraz ograniczona liczby wejść bramki NAND.



Rys. 5 Realizacja funkcji logicznej z wykorzystaniem struktury NAND-DMUX

STRUKTURA MUX-NAND-DMUX

Kolejna struktura MUX-NAND-DMUX wykorzystuje zalety dwóch poprzednich struktur. Układ multipleksera rozkłada tablicę prawdy na mniejsze tablice prawdy, które są realizowane za pomocą struktury NAND-DMUX. Zastosowany demultiplekser realizujący funkcje cząstkowe wymaga mniejszej liczby wejść adresowych oraz bramki NAND z mniejszą liczbą wejść. Wszystkie zmienne funkcji logicznej muszą być użyte jako wejścia adresowe. Optymalny wybór rozdziału wejść adresowych pomiędzy układ multipleksera i demultipleksera wykracza poza zakres przewidzianego materiału dydaktycznego. Realizację funkcji logicznej z przykładu (Tab. 2) przedstawia Rys. 6.



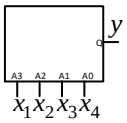
Rys. 6 Realizacja funkcji logicznej z wykorzystaniem struktury MUX-NAND-DMUX

g) SPRZĘTOWA REALIZACJA FUNKCJI LOGICZNEJ W UKŁADZIE DUŻEJ SKALI INTEGRACJI

Realizacja funkcji logicznej w układzie dużej skali integracji wymaga zastosowania pamięci nieulotnej. Liczba wejść adresowych pamięci musi odpowiadać liczbie zmiennych logicznych. Następnie uzupełniane są poszczególne komórki pamięci dla każdego możliwego adresu. W przykładzie (Tab. 2) występują cztery zmienne, zatem należy uzupełnić 16 wartości funkcji logicznej dla każdego adresu. Sprowadza się to do przepisania Tab. 2 do Tab. 9, gdzie Q oznacza wartość funkcji logicznej (wartość zapisanej komórki pamięci). Sprzętową realizację przedstawiono na Rys. 7, w której pominięto układ programowania pamięci.

Tab. 9 Zapis pamięci

Nr	A3	A2	A1	A0	Q
0	0	0	0	0	1
1	0	0	0	1	1
2	0	0	1	0	0
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	1
7	0	1	1	1	0
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	0



Rys. 7 Realizacja funkcji logicznej z wykorzystaniem pamięci nieulotnej