Fundusze Europejskie dla Rozwoju Społecznego
Rzeczpospolita Polska
Dofinansowane przez Unię Europejską

Politechnika Świętokrzyska
Wydział Elektrotechniki, Automatyki i Informatyki

Kierunek studiów:
Elektrotechnika

Jakub Piekoszewski

Materiały dydaktyczne do przedmiotu

Układy Cyfrowe

opracowane w ramach realizacji Projektu
„**Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki**”
FERS.01.05-IP.08-0234/23

Kielce, 2025

Spis treści

1. Wprowadzenie

Układy cyfrowe stanowią podstawę współczesnej elektroniki i informatyki. W przeciwieństwie do układów analogowych, które operują na sygnałach o ciągłym przebiegu, układy cyfrowe wykorzystują sygnały dyskretne – przyjmujące tylko dwa poziomy logiczne: 0 (stan niski, „fałsz") oraz 1 (stan wysoki, „prawda"). Dzięki temu możliwa jest niezawodna reprezentacja i przetwarzanie informacji w systemach komputerowych, mikrokontrolerach i urządzeniach cyfrowych.

Podstawą teoretyczną działania układów cyfrowych jest algebra Boole'a. Umożliwia ona opis i analizę zależności logicznych pomiędzy sygnałami za pomocą prostych operacji logicznych: AND (iloczyn logiczny), OR (suma logiczna) i NOT (negacja). Za pomocą tych operacji można tworzyć bardziej złożone funkcje logiczne, które stanowią matematyczny model działania bramek logicznych i całych układów cyfrowych.

Układy cyfrowe dzielimy na dwie podstawowe grupy:

- ✓ Układy kombinacyjne – w których stan wyjść zależy wyłącznie od aktualnych stanów wejść (np. sumatory, dekodery, multipleksery);
- ✓ Układy sekwencyjne – w których stan wyjść zależy nie tylko od wejść bieżących, ale również od poprzednich stanów (czyli posiadają pamięć). Przykładami są przerzutniki, liczniki oraz rejestry;

1.1. Podstawowe aksjomaty i prawa algebry Boole'a

Algebra Boole'a jest działem matematyki. Została opracowana w XIX wieku przez George'a Boole'a i stanowi podstawę logiki cyfrowej, używanej we wszystkich układach komputerowych oraz systemach cyfrowych. W algebrze Boole'a występują trzy podstawowe działania logiczne przedstawione w tabeli 1. Na tych trzech operacjach opiera się budowa bramek logicznych. Podstawowe typy bramek to: AND, OR, NOT, NAND, NOR, XOR, XNOR. Układy cyfrowe projektuje się, łącząc m.in. te elementy w większe struktury.

Tabela 1. Podstawowe operacje logiczne

Operacja logiczna	**Symbol**	**Zasada działania**
Iloczyn logiczny (koniunkcja)	$A \cdot B$	Wynik = 1 tylko, gdy $A = 1$ i $B = 1$
Suma logiczna (alternatywa)	$A + B$	Wynik = 1, gdy $A = 1$ lub $B = 1$
Negacja	$\bar{A}$	Wynik = 1, gdy $A = 0$

Każdy układ logiczny można uprościć dzięki prawom algebry Boole'a.

Najważniejsze z nich to:

1. Prawo Idempotentności:

$$A + A = A, \qquad A \cdot A = A \tag{1}$$

2. Prawo pochłaniania:

$$A + 1 = 1, \qquad A \cdot 0 = 0 \tag{2}$$

3. Prawo dopełnienia:

$$A + \overline{A} = 1, \qquad A \cdot \overline{A} = 0 \tag{3}$$

4. Prawo rozdzielności:

$$A \cdot (B + C) = A \cdot B + A \cdot C \tag{4}$$

5. Prawo De Morgana:

$$\overline{A \cdot B} = \overline{A} + \overline{B}, \qquad \overline{A + B} = \overline{A} \cdot \overline{B} \tag{5}$$

6. Prawo dominacji (neutralności):

$$A + 0 = A, \qquad A \cdot 1 = A \tag{6}$$

7. Prawo podwójnego zaprzeczenia:

$$\overline{\overline{A}} = A \tag{7}$$

1.2. Zastosowanie algebry Boole'a w praktyce

W niniejszym podrozdziale przedstawiono zestaw przykładowych zadań dotyczących podstaw algebry Boole'a. Ich celem jest utrwalenie wiadomości teoretycznych oraz nabycie umiejętności praktycznego stosowania praw i operacji logicznych. Zadania obejmują między innymi upraszczanie funkcji logicznych z wykorzystaniem

aksjomatów algebry Boole'a oraz analizę działania prostych układów logicznych. Rozwiązanie tych przykładów ma na celu lepsze zrozumienie zasady funkcjonowania układów cyfrowych oraz przygotowania do projektowania bardziej złożonych systemów logicznych.

1.2.1. Zadanie 1.1

Podać realizację bramki logicznej typu NAND o 3 wejściach z wykorzystaniem: a) elementów NAND o 2 wejściach oraz b) elementów NOR o 2 wejściach;

Rozwiązanie podpunktu a):
Punktem wyjścia do roważań jest funkcja realizowana przez bramkę logiczną typu NAND o 3 wejściach – negacja iloczynu logicznego trzech zmiennych:

$$y = \overline{a \cdot b \cdot c} \tag{8}$$

Wykorzystując prawo podwójnego zaprzeczenia (wzór nr 7) można przekształcić powyższe wyrażenie do postaci:

$$y = \overline{\overline{\overline{a \cdot b} \cdot c}} \tag{9}$$

Powstała funkcja $\overline{a \cdot b}$ jest poszukiwanym 2 wejściowym elementem typu NAND. Dla uproszczenia dalszych przekształceń można ją oznaczyć jako y_1. Równianie przyjmie wówczas postać:

$$y = \overline{\overline{y_1} \cdot c} \tag{10}$$

W kolejnym kroku powyższe równianie zostaje przekształcone z wykorzystaniem prawa idempotentności (1):

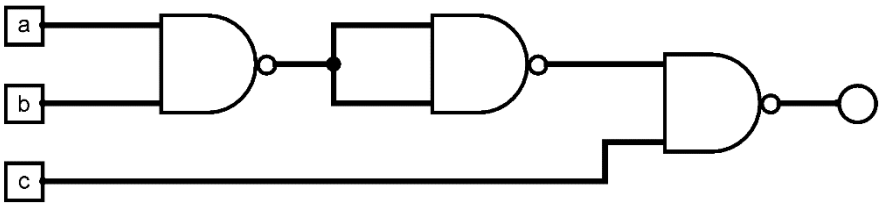
$$y = \overline{\overline{y_1 \cdot y_1} \cdot c} \tag{11}$$

Takie przekształcenie umożliwia utworzenie kolejnego elementu tupu NAND o 2 wejściach ($\overline{y_1 \cdot y_1}$), które można oznaczyć jako y_2. Równianie przyjmie wówczas postać:

$$y = \overline{y_2 \cdot c} \tag{12}$$

Jest to kolejny i zarazem ostatni element typu NAND o 2 wejściach potrzebny do realizacji układu. Rysunek 1 przedstawia realizację układową 3 wejściowej bramki logicznej typu NAND przy wykorzystaniu 2 wejściowych elementów tego samego typu.

[Tekst alternatywny. Schemat połączeń trzech dwuwejściowych bramek typu NAND.]



Rys. 1. Realizacja 3 wejściowej bramki typu NAND przy wykorzystaniu bramek typu NAND o 2 wejściach

Rozwiązanie podpunktu b):

Punktem wyjścia do roważań jest funkcja realizowana przez bramkę logiczną typu NAND o 3 wejściach – negacja iloczynu logicznego trzech zmiennych:

$$y = \overline{a \cdot b \cdot c} \tag{13}$$

Wykorzystując prawo De Morgana zaprzeczenia (wzór nr 5) można przekształcić powyższe wyrażenie do postaci:

$$y = \overline{a} + \overline{b} + \overline{c} \tag{14}$$

Wykorzystując prawo idempotentności (1) powyższe równanie można przekształcić do postaci:

$$y = \overline{a + a} + \overline{b + b} + \overline{c + c} \tag{15}$$

Powstałe funkcje $\overline{a+a}$, $\overline{b+b}$ oraz $\overline{c+c}$ są poszukiwanymi 2 wejściowym elementami typu NOR. Dla uproszczenia dalszych przekształceń można ją oznaczyć jako y_1, y_2 oraz y_3. Równianie przyjmie wówczas postać:

$$y = y_1 + y_2 + y_3 \tag{16}$$

Wykorzystując prawo podwójnego zaprzeczenia (wzór nr 7) można przekształcić powyższe wyrażenie do postaci:

$$y = \overline{\overline{y_1 + y_2}} + y_3 \tag{17}$$

Powstała funkcja $\overline{y_1 + y_2}$ jest kolejnym poszukiwanym 2 wejściowym elementem typu NOR. Dla uproszczenia dalszych przekształceń można go oznaczyć jako y_4. Równianie przyjmie wówczas postać:

$$y = \overline{y_4} + y_3 \tag{18}$$

Ponownie wykorzystując prawo idempotentności (1) powyższe równanie można przekształcić do postaci:

$$y = \overline{y_4 + y_4} + y_3 \tag{19}$$

Powstała funkcja $\overline{y_4 + y_4}$ jest kolejnym poszukiwanym 2 wejściowym elementam typu NOR. Dla uproszczenia dalszych przekształceń można go oznaczyć jako y_5. Równianie przyjmie wówczas postać:

$$y = y_5 + y_3 \tag{20}$$

W dalszym kroku wykorzystując prawo podwójnego zaprzeczenia (wzór nr 7) można przekształcić powyższe wyrażenie do postaci:

$$y = \overline{\overline{y_5 + y_3}} \tag{21}$$

Powstała funkcja $\overline{y_5 + y_3}$ jest kolejnym poszukiwanym 2 wejściowym elementem typu NOR. Dla uproszczenia dalszych przekształceń można go oznaczyć jako y_6. Równianie przyjmie wówczas postać:

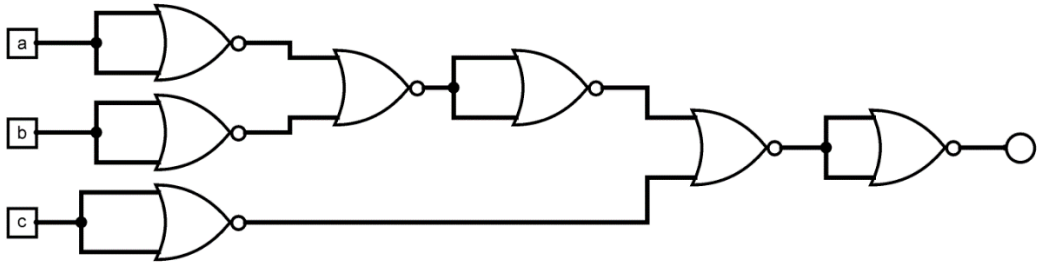
$$y = \overline{y_6} \tag{22}$$

Ponownie należy wykorzystać prawo idempotentności (1), aby przekształcić powyższe równanie do postaci:

$$y = \overline{y_6 + y_6} \tag{23}$$

Jest to kolejny i zarazem ostatni element typu NOR o 2 wejściach potrzebny do realizacji układu. Rysunek 2 przedstawia realizację układową 3 wejściowej bramki logicznej typu NAND przy wykorzystaniu 2 wejściowych elementów typu NOR.

[Tekst alternatywny. Schemat połączeń sześciu dwuwejściowych bramek typu NOR.]



Rys. 2. Realizacja 3 wejściowej bramki typu NAND przy wykorzystaniu bramek typu NOR o 2 wejściach

1.2.2. Zadanie 1.2

Podać realizację funkcji $y = a \cdot b + \overline{c} + d$ przy wykorzystaniuelementów NAND o 3 wejściach.

Rozwiązanie:
W pierwszym kroku należy zastosować prawo podwójnego zaprzeczenia (7):

$$y = \overline{\overline{a \cdot b + \overline{c} + d}} \tag{23}$$

Wprowadzenie podwójnej negacji umożliwi zastosowanie prawa De Morgana (5):

$$y = \overline{\overline{a \cdot b} \cdot \overline{\overline{c}} \cdot \overline{d}} = \overline{\overline{a \cdot b} \cdot c \cdot \overline{d}} \tag{24}$$

Wykorzystując prawo idempotentności (1) powyższe równanie można przekształcić do postaci:

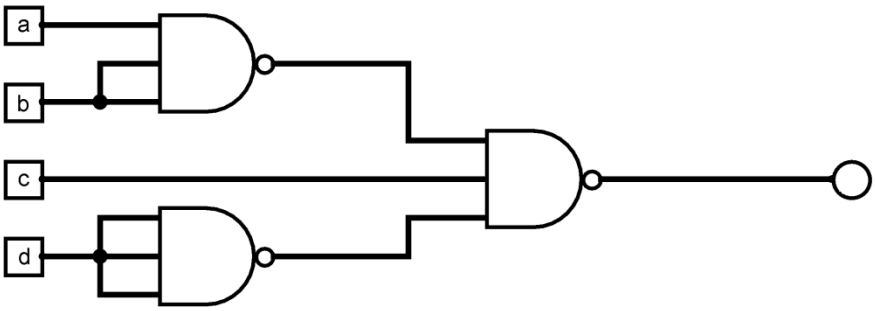
$$y = \overline{\overline{a \cdot b \cdot b} \cdot c \cdot \overline{d \cdot d \cdot d}} \tag{25}$$

Powstałe funkcje $\overline{a \cdot b \cdot b}$, oraz $\overline{d \cdot d \cdot d}$ są poszukiwanymi 3 wejściowym elementami typu NAND. Dla uproszczenia dalszych przekształceń można ją oznaczyć jako y_1 oraz y_2. Równianie przyjmie wówczas postać:

$$y = \overline{y_1 \cdot c \cdot y_2} \tag{25}$$

Jest to kolejny i zarazem ostatni element typu NAND o 3 wejściach potrzebny do realizacji układu. Rysunek 3 przedstawia realizację układową funkcji $y = a \cdot b + \overline{c} + d$ przy wykorzystaniu 3 wejściowych elementów typu NAND.

[Tekst alternatywny. Schemat połączeń trzech trójwejściowych bramek typu NAND.]



Rys. 3. Realizacja funkcji $y = a \cdot b + \overline{c} + d$ przy wykorzystaniu bramek typu NAND o 3 wejściach

Każda funkcja logiczna może zostać zapisana w postaci wyrażenia algebraicznego wykorzystującego tylko podstawowe operacje logiczne: AND, OR i NOT. Oznacza to, że algebra Boole'a umożliwia realizację dowolnej funkcji przełączającej, niezależnie od liczby wejść i stopnia złożoności. W praktyce oznacza to, że przy pomocy bramek logicznych odpowiadających tym działaniom (AND, OR, NOT) można skonstruować każdy układ cyfrowy, od prostych dekoderów po złożone procesory. W praktyce inżynierskiej wykorzystuje się również inne funkcjonalnie zupełne zestawy, np.: tylko bramki NAND lub NOR. Dzięki temu można budować całe układy z jednego rodzaju elementów, co upraszcza konstrukcję i projektowanie.

2. Układy kombinacyjne

Układy kombinacyjne stanowią jedną z podstawowych grup układów cyfrowych. Charakteryzują się tym, że stan ich wyjść w danej chwili zależy wyłącznie od aktualnych stanów wejść, a nie od wcześniejszych sygnałów czy stanów układu. Oznacza to, że nie posiadają one elementów pamiętających — wynik działania jest więc bezpośrednią funkcją wartości logicznych podanych na wejścia.

Funkcje opisujące układy kombinacyjne można zapisać w postaci funkcji przełączających, które przedstawiają zależność między wejściami a wyjściami układu. Funkcje te mogą przyjmować formę algebraiczną (np. wyrażenie w algebrze Boole'a) lub tabelaryczną (np. tabela prawdy). W praktyce projektowania układów cyfrowych dąży się do tego, aby funkcje te były możliwie najprostsze, czyli realizowały określone działanie przy użyciu minimalnej liczby bramek logicznych. Proces ten nazywamy minimalizacją funkcji logicznych. Jedną z najczęściej stosowanych metod jest metoda tablic Karnaugha, która pozwala w sposób graficzny uprościć funkcje logiczne poprzez grupowanie jednakowych wartości w tabeli odwzorowującej wszystkie możliwe kombinacje wejść.

2.2. Minimalizacja funkcji logicznych metodą tablic Karnaugha

W praktyce projektowania układów kombinacyjnych jednym z głównych celów jest upraszczanie funkcji logicznych. Złożone funkcje, opisujące zależność między wieloma wejściami a wyjściem, można uprościć tak, aby do ich realizacji potrzeba było jak najmniej bramek logicznych. Jedną z najczęściej stosowanych metod graficznych służących do tego celu jest metoda tablic Karnaugha.

Tablica Karnaugha to graficzna reprezentacja tabeli prawdy. Każda komórka tablicy odpowiada jednej kombinacji wartości wejść (czyli jednemu wierszowi tabeli prawdy) i oznaczona jest wartością funkcji logicznej — 0 lub 1. Dzięki specjalnemu rozmieszczeniu komórek (w kodzie Graya, gdzie sąsiednie komórki różnią się tylko jedną zmienną), możliwe jest łączenie grup sąsiednich jedynek w celu uproszczenia wyrażenia logicznego.

Zasada grupowania:
Podstawową zasadą metody jest grupowanie jedynek (1) w zbiory zawierające liczbę komórek równą potędze liczby 2 (1, 2, 4, 8, 16, ..). Każda taka grupa (nazywana

implikantem) odpowiada uproszczonemu fragmentowi funkcji. Grupowanie można wykonywać w poziomie, pionie, a także z zawijaniem krawędzi tablicy.
Reguły tworzenia grup:

1. Grupy mogą zawierać tylko komórki z wartością 1.
2. Liczba komórek w grupie musi być potęgą liczby 2.
3. Grupy mogą się nakładać – to często pozwala uzyskać prostsze rozwiązanie.
4. Należy tworzyć jak największe grupy, aby uzyskać maksymalne uproszczenie.
5. Komórki o wartości „-" mogą być użyte w grupowaniu, jeśli prowadzi to do prostszej postaci funkcji.

Po dokonaniu grupowania każda grupa odpowiada iloczynowi (dla funkcji w postaci sumy iloczynów, SOP – *Sum of Products*), w którym występują tylko te zmienne, które nie zmieniają swojej wartości wewnątrz grupy. Istnieje też równoważna metoda minimalizacji zer, prowadząca do postaci iloczynu sum (POS – *Product of Sums*).

Metoda tablic Karnaugha jest bardzo użyteczna przy upraszczaniu funkcji o niewielkiej liczbie zmiennych (2–6). Dla większych funkcji stosuje się metody komputerowe, takie jak algorytm Quine'a–McCluskeya.

2.2.1. Zadanie 2.1

Działanie układu cyfrowego opisane jest funkcją kanonicznej postaci sumy pięciu zmiennych:

$$y = f(x_1, \dots, x_5) = \sum [2,3,4,8,12,19,24,28,31, (11,18,20,23,26)] \qquad (26)$$

*Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki"
nr FERS.01.05-IP.08-0234/23*

Podać optymalne realizacje układu (włącznie ze schematami) wykorzystując: a) elementy NAND oraz NOT oraz b) elementy NOR oraz NOT.

Rozwiązanie podpunktu a:

W pierwszej kolejności należy uzupełnić zawartość tablicy Karnaugha odpowiednimi wartościami. Kanoniczna postać sumy ze wzoru 26 zawiera pozycje w tablicy dla których funkcja przyjmuje wartości „1”. Są to pozycje: 2, 3, 4, 8, 12…

[Tekst alternatywny. Tablica Karnaugha z pozycjami dla których wartość funkcji przyjmuje „1”.]

x_1x_2 \ $x_3x_4x_5$	000	001	011	010	110	111	101	100
00			1	1		1		
01	1		-					1
11	1			-		1		1
10			1	-		-		-

Rys. 4. Tablica Karnaugha

Kanoniczna postać sumy ze wzoru 26 zawiera również pozycje w tablicy dla których funkcja przyjmuje wartości „-”. Są to pozycje: 11, 18, 20, 23 oraz 26:

[Tekst alternatywny. Tablica Karnaugha z pozycjami dla których wartość funkcji przyjmuje „-”.]

$x_3x_4x_5$ / x_1x_2	000	001	011	010	110	111	101	100
00			1	1		1		
01	1							1
11	1					1		1
10			1					

Rys. 5. Tablica Karnaugha

Pozostałe puste pola należy uzupełnić wartościami „0”:

[Tekst alternatywny. Tablica Karnaugha z pozycjami dla których wartość funkcji przyjmuje „0”.]

$x_3x_4x_5$ / x_1x_2	000	001	011	010	110	111	101	100
00	0	0	1	1	0	1	0	0
01	1	0	-	0	0	0	0	1
11	1	0	0	-	0	1	0	1
10	0	0	1	-	0	-	0	-

Rys. 6. Tablica Karnaugha

Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki”
nr FERS.01.05-IP.08-0234/23

W celu realizacji funkcji z wykorzystaniem elementów NAND i NOT należy w powyższej tablicy połączyć w jak najliczniejsze grupy sąsiadujące „1”. Powyższa tablica, pozwala na utworzenie grup czteroelementowych.

[Tekst alternatywny. Tablica Karnaugha z zaznaczoną, kolorem czerwonym, pierwszą czteroelementową grupą sąsiadujących „1”.]

$x_1x_2x_3$ / x_1x_2	000	001	011	010	110	111	101	100
00	0	0	1	1	0	1	0	0
01	1	0	-	0	0	0	0	1
11	1	0	0	-	0	1	0	1
10	0	0	1	-	0	-	0	-

Rys. 7. Tablica Karnaugha

Kolejna najliczniejsza grupa sąsiadujących „1” jest również grupa czteroelementowa.

[Tekst alternatywny. Tablica Karnaugha z zaznaczoną, kolorem zielonym, drugą czteroelementową grupą sąsiadujących „1”.]

$x_1x_2x_3$ / x_1x_2	000	001	011	010	110	111	101	100
00	0	0	1	1	0	1	0	0
01	1	0	-	0	0	0	0	1
11	1	0	0	-	0	1	0	1
10	0	0	1	-	0	-	0	-

Rys. 8. Tablica Karnaugha

*Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki”
nr FERS.01.05-IP.08-0234/23*

Kolejna najliczniejsza grupa sąsiadujących „1”, którą można oznaczyć jest ponownie grupa czteroelementowa.

[Tekst alternatywny. Tablica Karnaugha z zaznaczoną, kolorem niebieski, trzecią czteroelementową grupą sąsiadujących „1”.]

$x_1x_2x_3$ / x_1x_2	000	001	011	010	110	111	101	100
00	0	0	1	1	0	1	0	0
01	1	0	-	0	0	0	0	1
11	1	0	0	-	0	1	0	1
10	0	0	1	-	0	-	0	-

Rys. 9. Tablica Karnaugha

Na dalszym etapie nie ma już możliwości połącznia pozostałych „1” w równie liczne grupy. Pozostaje jedynie możliwość utworzenia jednej dwuelementowej grupy zawierającej ostatnią „1”.

[Tekst alternatywny. Tablica Karnaugha z zaznaczoną, kolorem pomarańczowym, czwartą, ostatnią dwulementową grupą sąsiadujących „1”.]

$x_1x_2x_3$ / x_1x_2	000	001	011	010	110	111	101	100
00	0	0	1	1	0	1	0	0
01	1	0	-	0	0	0	0	1
11	1	0	0	-	0	1	0	1
10	0	0	1	-	0	-	0	-

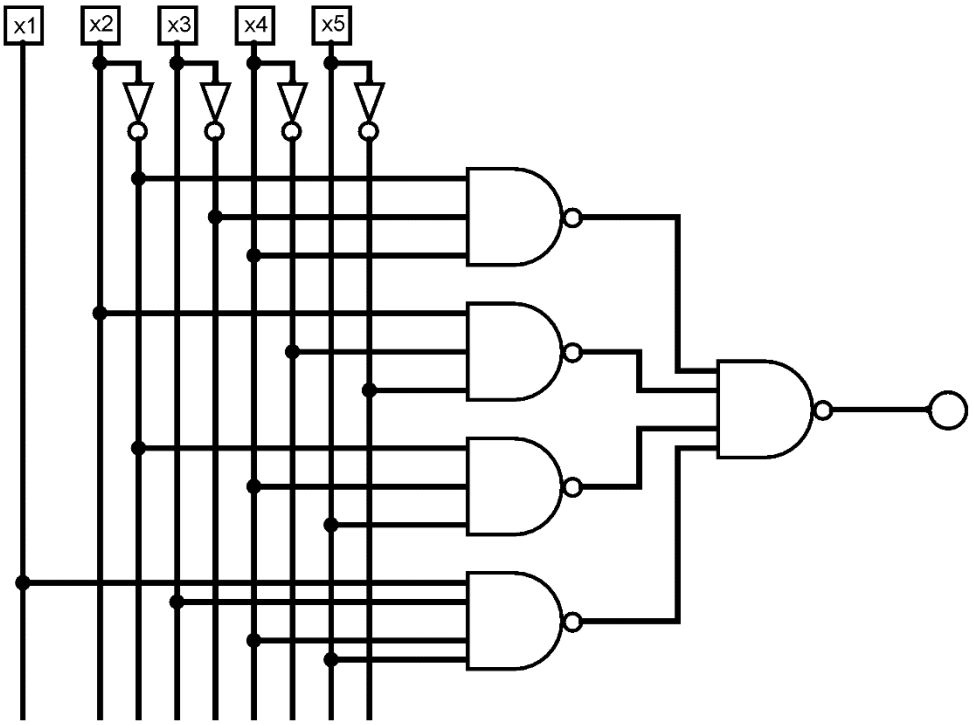
Rys. 10. Tablica Karnaugha

Na podstawie zaznaczonych grup zminimalizowana funkcja ma postać:

$$y = \overline{x_2} \cdot \overline{x_3} \cdot x_4 + x_2 \cdot \overline{x_4} \cdot \overline{x_5} + \overline{x_2} \cdot x_4 \cdot x_5 + x_1 \cdot x_3 \cdot x_4 \cdot x_5 \quad (27)$$

Powyższe równanie w łatwy sposób można przedstawić na schemacie (rysunek 11) składającym się jedynie z bramek typu NAND oraz NOT.

[Tekst alternatywny. Schemat połączeń realizowanej funkcji zawierający jedynie elementy typu NAND i NOT.]

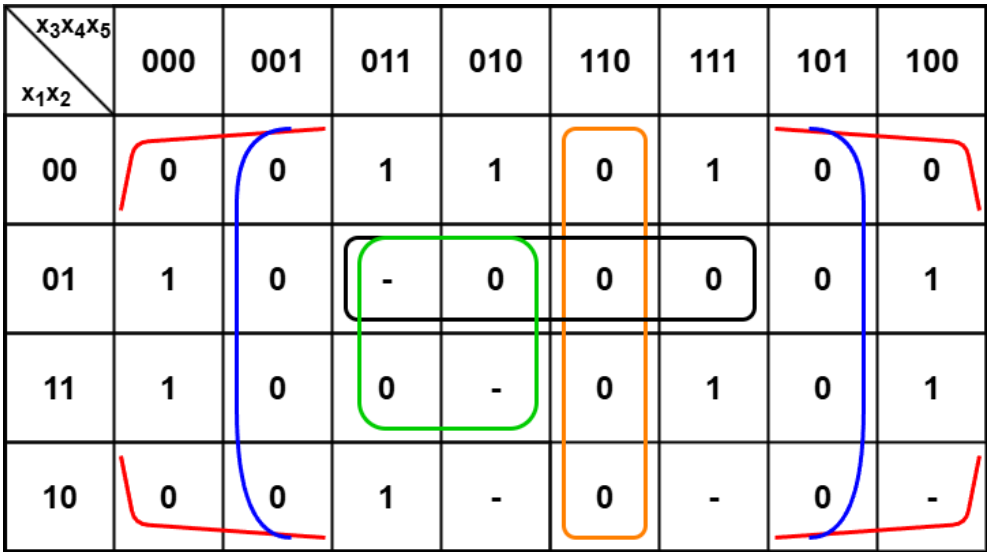


Rys. 11. Schemat układu

Rozwiązanie podpunktu b:

Korzystamy z uzupełnionej tablicy Karnaugha z rysunku 6. W celu realizacji funkcji przy wykorzystaniu elementów typu NOR i NOT, należy tym razem połączyć w jak najliczniejsze grupy, sąsiadujące ze sobą elementy przyjmujące wartość „0”.

[Tekst alternatywny. Tablica Karnaugha z zaznaczonymi wszystkimi grupami „0”].



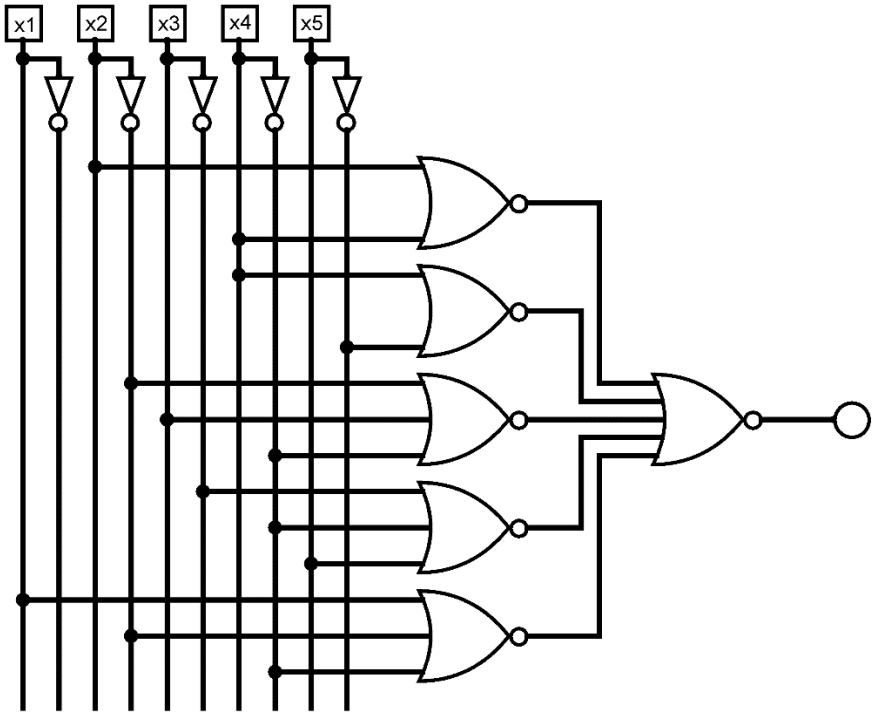
x_1x_2 \ $x_3x_4x_5$	000	001	011	010	110	111	101	100
00	0	0	1	1	0	1	0	0
01	1	0	-	0	0	0	0	1
11	1	0	0	-	0	1	0	1
10	0	0	1	-	0	-	0	-

Rys. 12. Tablica Karnaugha

Na podstawie zaznaczonych grup zminimalizowana funkcja ma postać:

$$y = (x_2 + x_4) \cdot (x_4 + \overline{x_5}) \cdot (\overline{x_2} + x_3 + \overline{x_4}) \cdot (\overline{x_3} + \overline{x_4} + x_5) \cdot (x_1 + \overline{x_2} + \overline{x_4}) \quad (28)$$

[Tekst alternatywny. Schemat połączeń realizowanej funkcji zawierający jedynie elementy typu NOR i NOT.]



Rys. 13. Schemat układu

2.3. Realizacje multiplekserowe układów kombinacyjnych

Multiplekser (ang. Multiplexer, w skrócie MUX) jest układem kombinacyjnym, który umożliwia wybór jednego z wielu sygnałów wejściowych i przekazanie go na jedno wyjście. Działa więc jak przełącznik elektroniczny sterowany sygnałami binarnymi.

Multiplekser znajduje szerokie zastosowanie w systemach cyfrowych — m.in. w układach sterujących, dekoderach, magistralach danych i realizacjach funkcji logicznych. Jedną z jego najważniejszych cech jest możliwość realizacji dowolnej funkcji logicznej przy odpowiednim podłączeniu wejść i linii sterujących.

Typowy multiplekser posiada:

- ✓ n wejść danych;
- ✓ m wejść sterujących (adresowych);
- ✓ jedno wyjście Y;

Liczba wejść danych jest związana z liczbą linii adresowych zależnością:

$$n = 2^m \tag{29}$$

2.3.1. Zadanie 2.2

Podać realizację poniższej funkcji (włącznie ze schematem) wykorzystując multiplekser o 3 wejściach adresowych oraz elementy NAND i NOT.

$$y = y = \overline{x_2} \cdot \overline{x_3} \cdot x_4 + x_2 \cdot \overline{x_4} \cdot \overline{x_5} + \overline{x_2} \cdot x_4 \cdot x_5 + x_1 \cdot x_3 \cdot x_4 \cdot x_5 \quad (30)$$

Rozwiązanie:

[Tekst alternatywny. Pierwotna postać zbioru implikantów prostych funkcji dla przykładu z zadania (wzór nr 30).]

$F^1{}_{min}$

x_1	x_2	x_3	x_4	x_5
-	0	0	1	-
-	1	-	0	0
-	0	-	1	1
0	-	0	0	0

Rys. 14. Zbiór prostych implikantów funkcji F^1min

W kolejnym kroku należy określić liczbę „-” oraz „0” dla każdego z sygnałów.

[Tekst alternatywny. Pierwotna postać zbioru implikantów prostych funkcji wraz z określoną liczbą „-” oraz „0” dla każdego z sygnałów.]

$F^1{}_{min}$

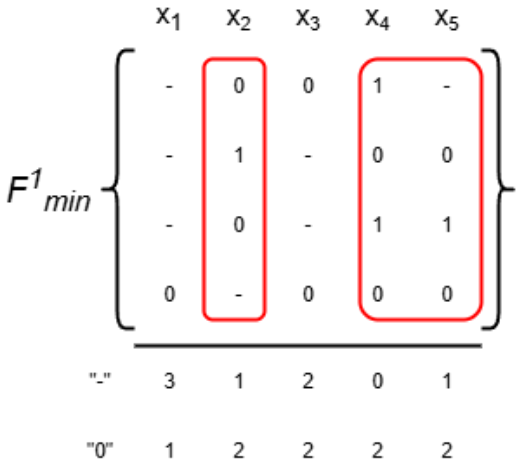
x_1	x_2	x_3	x_4	x_5
-	0	0	1	-
-	1	-	0	0
-	0	-	1	1
0	-	0	0	0
"-"	3	1	2	0	1
"0"	1	2	2	2	2

Rys. 15. Zbiór prostych implikantów funkcji F^1min

Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki” nr FERS.01.05-IP.08-0234/23

Dla optymalnego rozwiązania za wejścia adresowe wybieramy te trzy sygnały, dla których liczba kresek jest najmniejsza. W przypadku takiej samej liczby zer dla wielu z sygnałów, w drugiej kolejności wybieramy te o największej liczbie zer.

[Tekst alternatywny. Pierwotna postać zbioru implikantów prostych funkcji wraz z zaznaczonymi kolorem czerwonym sygnałami, które zostaną wykorzystane za wejścia adresowe multipleksera.]

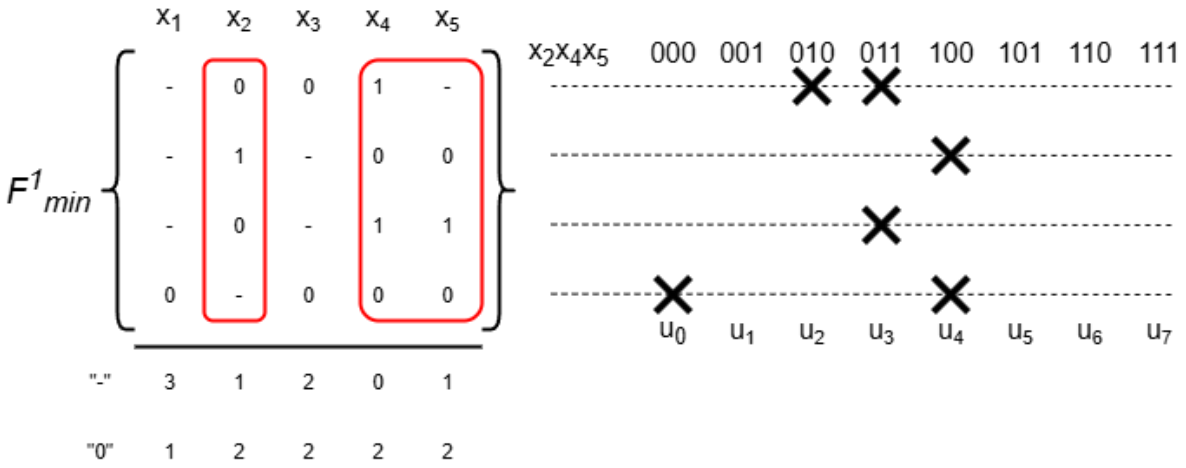


Rys. 16. Zbiór prostych implikantów funkcji F^1min

Aby uzyskać wyrażenia opisujące funkcje $u_1, u_2, u_3, \ldots, u_7$ będące kolejnymi wejściami informacyjnymi projektowanego multipleksera, należy określić które z wierszy wchodzą w skład opisów wyrażeń dla kolejnych wartości sygnałów x_2, x_4, x_5, tzn. 000, 001, 010… Zostało to zaznaczone na rysunku 17 znakami **X.**

[Tekst alternatywny. Pierwotna postać zbioru implikantów prostych funkcji wraz z naniesioną informacją (X), które z wierszy wchodzą w skład opisu wejść informacyjnych multipleksera.]

*Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki”
nr FERS.01.05-IP.08-0234/23*



Rys. 17. Zbiór prostych implikantów funkcji $F^1 min$

Na tej podstawie, postaci funkcji $u_1, u_2, u_3, \dots, u_7$ (wejść informacyjnych) są następujące (w skład ich opisu wchodzą wyłącznie zmienne nie wykorzystane za wejścia adresowe - x_1, x_3):

$$u_0 = \{0\ 0\} = \overline{x_1} \cdot \overline{x_3} \tag{31}$$

$$u_1 = \{\emptyset\} = 0 \tag{32}$$

$$u_2 = \{-\ 0\} = 1 \cdot \overline{x_3} = \overline{x_3} \tag{33}$$

$$u_3 = \begin{Bmatrix} - & 0 \\ - & - \end{Bmatrix} = 1 \cdot \overline{x_3} + 1 \cdot 1 = \overline{x_3} + 1 = 1 \tag{33}$$

$$u_4 = \begin{Bmatrix} - & - \\ 0 & 0 \end{Bmatrix} = 1 \cdot 1 + \overline{x_1} \cdot \overline{x_3} = 1 + \overline{x_1} \cdot \overline{x_3} = 1 \tag{34}$$

$$u_5 = u_6 = u_7 = \{\emptyset\} = 0 \tag{35}$$

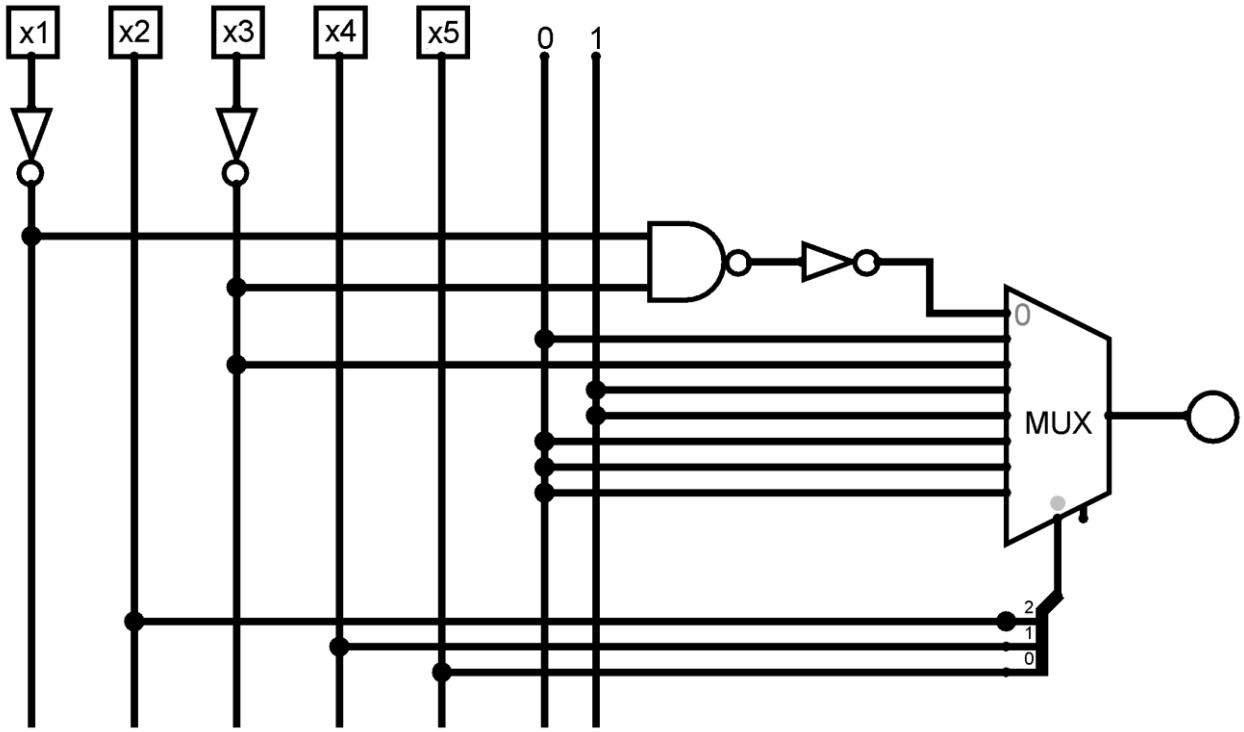
Uzyskana realizacje funkcji na bazie multipleksera o trzech wejściach adresowych można zwięźle zapisać za pomocą operatora MUX:

$$y = MUX(\overline{x_1} \cdot \overline{x_3}, 0, \overline{x_3}, 1,1,0,0,0;\ x_2, x_4, x_5) \tag{36}$$

Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki” nr FERS.01.05-IP.08-0234/23

Schemat układu:

[Tekst alternatywny. Realizacja układowa funkcji na bazie multipleksera o trzech wejściach adresowych (sygnały x_2, x_4, x_5) oraz elementach typu NAND i NOT.]



Rys. 18. Schemat układu

3. Układy sekwencyjne

Układy sekwencyjne stanowią jedną z podstawowych klas układów cyfrowych, obok układów kombinacyjnych. W odróżnieniu od nich, wyjście układu sekwencyjnego zależy nie tylko od aktualnych wartości sygnałów wejściowych, lecz także od poprzednich stanów układu, czyli od jego pamięci. Z tego powodu układy te nazywane są często układami z pamięcią lub układami dynamicznymi.

W układach sekwencyjnych istotną rolę odgrywa stan wewnętrzny, który reprezentuje historię działania układu i decyduje o jego przyszłym zachowaniu. Stan ten jest

Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki”
nr FERS.01.05-IP.08-0234/23

najczęściej przechowywany w elementach pamięciowych, takich jak przerzutniki lub rejestry, które reagują na sygnały zegarowe (clock).

Układy sekwencyjne można podzielić na dwie główne grupy:

- ✓ Układy asynchroniczne – w których zmiany stanów zachodzą natychmiast po zmianie sygnałów wejściowych, bez wspólnego sygnału synchronizującego. Charakteryzują się dużą szybkością działania, ale są bardziej podatne na błędy i trudniejsze w projektowaniu.
- ✓ Układy synchroniczne – w których zmiany stanów występują jedynie w określonych chwilach czasu, wyznaczonych przez impulsy zegarowe (clock pulses). Tego typu układy są obecnie najczęściej stosowane w praktyce, szczególnie w mikroprocesorach, automatach sterujących i systemach cyfrowych.

Podstawowym elementem układu sekwencyjnego jest przerzutnik (flip-flop), który przechowuje jeden bit informacji. Przerzutniki mogą zmieniać stan w odpowiedzi na sygnały wejściowe i impuls zegarowy, dzięki czemu możliwe jest zapamiętywanie informacji oraz realizacja bardziej złożonych funkcji logicznych zależnych od czasu.

3.1. Zadanie 3.1

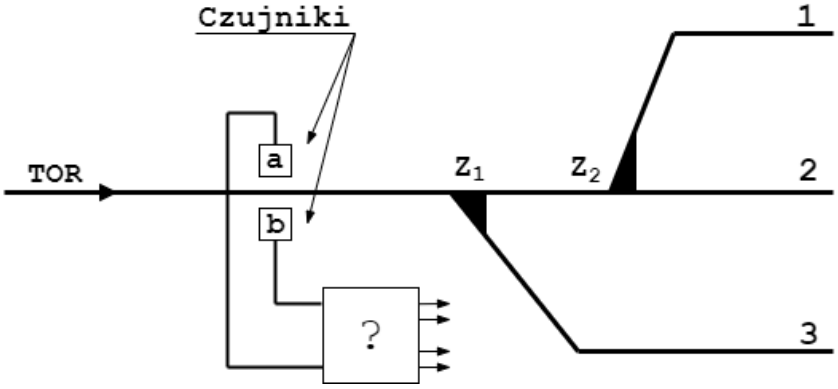
Zaprojektować układa sterowania zwrotnicami przedstawiony na rysunku 19. Z lewej strony po torze nadjeżdżają wózki cechowane kółkami a i b umieszczonymi na podwoziu. Wózek z dwoma kołkami ma wjechać na tor 2, a z jednym – odpowiednio na tor 1 albo 3. Każda zwrotnica sterowana jest dwoma sygnałami $x_i y_i$ (szczegóły dotyczące działań podejmowanych w zależności od ich wartości przedstawiono w tabeli 2). Należy uwzględnić niejednoczesność oddziaływania kołków na czujniki oraz żądanie, żeby w czasie przejazdu wózka nad czujnikami zwrotnice nie wykonywały zbędnych ruchów. Podać realizację układu, wraz ze schematem, wykorzystując elementy NAND i NOT.

Tabela 2. Możliwości sterowania zwrotnicą

Sygnały $x_i y_i$	**Podejmowane działanie**
00	zwrotnica nie zmienia położenia
01	przestawia się „na wprost” i pozostaje w tm położeniu
10	przestawia się „na bok” i pozostaje w tym położeniu
11	kombinacja zabroniona

*Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki”
nr FERS.01.05-IP.08-0234/23*

[Tekst alternatywny. Schemat działania dwóch zwrotnic umieszczonych na torze, po którym przemieszczają się wózki. Bezpośrednio przed rozjazdami umieszczono układ sterujący decydujący o kierunku jazdy.]



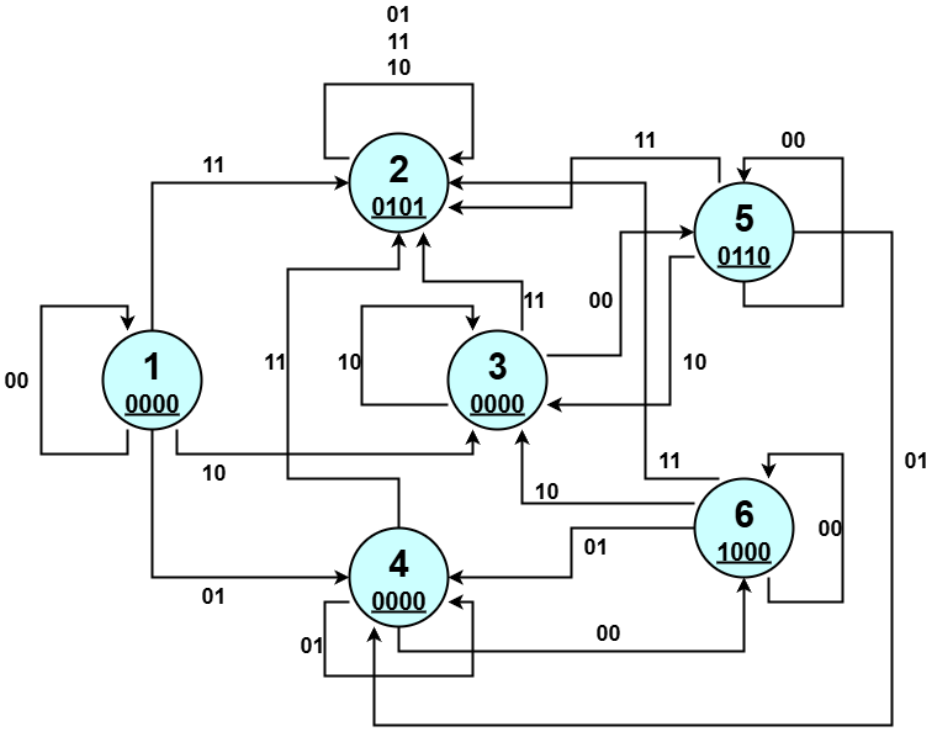
Rys. 19. Plan zwrotnic

Rozwiązanie:

W pierwszej kolejności należy zaproponować graf przejść i wyjść, który niesie pełny opis projektowanego układu, tzn. określa liczbę jego stanów wewnętrznych oraz definiuje funkcje przejść i wyjść:

[Tekst alternatywny. Pełny opis projektowanego układu. Cyframi 1,2,…6 oznaczono stany wewnętrzne, podkreślone słowa bitowe stanowią wyjścia układu, natomiast

wartości przy strzałkach – przejścia pomiędzy stanami.]



Rys. 20. Graf przejść i wyjść

Na podstawie grafu przejść i wyjść należy utworzyć pierwotna tablicę przejść i wyjść:

[Tekst alternatywny. Z lewej strony tablicy wypisane są wszystkie stany wewnętrzne automatu, a u góry wszystkie stany przejść. Prawa kolumna przedstawia stany wyjść automatu.]

A \ ab	00	01	11	10	$x_1y_1x_2y_2$
1	1	4	2	3	0000
2	1	2	2	2	0101
3	5	-	2	3	0000
4	6	4	2	-	0000
5	5	4	2	3	0110
6	6	4	2	3	1000

Rys. 21. Pierwotna tablica przejść i wyjść

Kolejnym krokiem jest kodowanie wartości pierwotnej tablicy przejść i wyjść. W związku z liczbą stanów wewnętrznych równą 6 konieczne jest zastosowanie trzy bitowych słów:

[Tekst alternatywny. Zakodowana tablica, w której poszczególne stany wewnętrzne automatu przedstawiono w postaci binarnej.]

$Q_1Q_2Q_3$ \ ab	00	01	11	10	$x_1y_1x_2y_2$
000	000	010	001	011	0000
001	000	001	001	001	0101
011	110	-	001	011	0000
010	111	010	001	-	0000
110	110	010	001	011	0110
111	111	010	001	011	1000
---	---	---	---	---	----
---	---	---	---	---	----

Rys. 22. Zakodowana tablica przejść i wyjść

Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki”
nr FERS.01.05-IP.08-0234/23

Na podstawie zakodowanej tablicy projektowanego układu wyznaczamy tablice układów wzbudzeń dla przerzutników typu D:

[Tekst alternatywny. Zakodowana tablica rozpisana na trzy niezależne tablice wzbudzeń przerzutnika typu D.]

$Q_1Q_2Q_3$ \ ab	00	01	11	10
000	0	0	0	0
001	0	0	0	0
011	1	-	0	0
010	1	0	0	-
110	1	0	0	0
111	1	0	0	0
101	---	---	---	---
100	---	---	---	---

D_1

$Q_1Q_2Q_3$ \ ab	00	01	11	10
000	0	1	0	1
001	0	0	0	0
011	1	-	0	1
010	1	1	0	-
110	1	1	0	1
111	1	1	0	1
101	---	---	---	---
100	---	---	---	---

D_2

$Q_1Q_2Q_3$ \ ab	00	01	11	10
000	0	0	1	1
001	0	1	1	1
011	0	-	1	1
010	1	0	1	-
110	0	0	1	1
111	1	0	1	1
101	---	---	---	---
100	---	---	---	---

D_3

Rys. 23. Tablice wzbudzeń dla przerzutników typu D

W celu określenia funkcji wzbudzeń każdego z przerzutników należy połączyć w jak najliczniejsze grupy, sąsiadujące ze sobą elementy przyjmujące wartość „1”:

[Tekst alternatywny. Tablice wzbudzeń z zaznaczonymi kolorami grupami „1”].

$Q_1Q_2Q_3$ \ ab	00	01	11	10
000	0	0	0	0
001	0	0	0	0
011	1	-	0	0
010	1	0	0	-
110	1	0	0	0
111	1	0	0	0
101	---	---	---	---
100	---	---	---	---

D_1

$Q_1Q_2Q_3$ \ ab	00	01	11	10
000	0	1	0	1
001	0	0	0	0
011	1	-	0	1
010	1	1	0	-
110	1	1	0	1
111	1	1	0	1
101	---	---	---	---
100	---	---	---	---

D_2

$Q_1Q_2Q_3$ \ ab	00	01	11	10
000	0	0	1	1
001	0	1	1	1
011	0	-	1	1
010	1	0	1	-
110	0	0	1	1
111	1	0	1	1
101	---	---	---	---
100	---	---	---	---

D_3

Rys. 24. Tablice wzbudzeń dla przerzutników typu D

Politechnika Świętokrzyska
Kielce University of Technology
Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki”
nr FERS.01.05-IP.08-0234/23

Na podstawie zaznaczonych grup funkcje wzbudzeń poszczególnych przerzutników typu D przyjmują postać:

$$D_1 = Q_2 \cdot \overline{a} \cdot \overline{b} \tag{37}$$

$$D_2 = Q_2 \cdot \overline{a} + Q_2 \cdot \overline{b} + \overline{Q_3} \cdot \overline{a} \cdot b + \overline{Q_3} \cdot a \cdot \overline{b} \tag{38}$$

$$D_3 = a + \overline{Q_2} \cdot Q_3 \cdot b + Q_1 \cdot Q_3 \cdot \overline{b} + \overline{Q_1} \cdot Q_2 \cdot \overline{Q_3} \cdot \overline{b} \tag{39}$$

Na podstawie zakodowanej tablicy przejść i wyjść należy także wyznaczyć kolejne tablice wyjść (x_i y_i):

[Tekst alternatywny. Tablice wyjść z zaznaczonymi kolorami grupami „1”].

Q_1Q_2 \ Q_3	0	1
00	0	0
01	0	0
11	0	1
10	-	-

x_1

Q_1Q_2 \ Q_3	0	1
00	0	1
01	0	0
11	1	0
10	-	-

y_1

Q_1Q_2 \ Q_3	0	1
00	0	0
01	0	0
11	1	0
10	-	-

x_2

Q_1Q_2 \ Q_3	0	1
00	0	1
01	0	0
11	0	0
10	-	-

y_2

Rys. 25. Tablice wyjść projektowanego układu

Na podstawie zaznaczonych grup funkcje wyjść projektowanego układu mają następującą postać:

$$x_1 = Q_1 \cdot Q_3 \tag{40}$$

$$y_1 = Q_1 \cdot \overline{Q_3} + \overline{Q_2} \cdot Q_3 \tag{41}$$

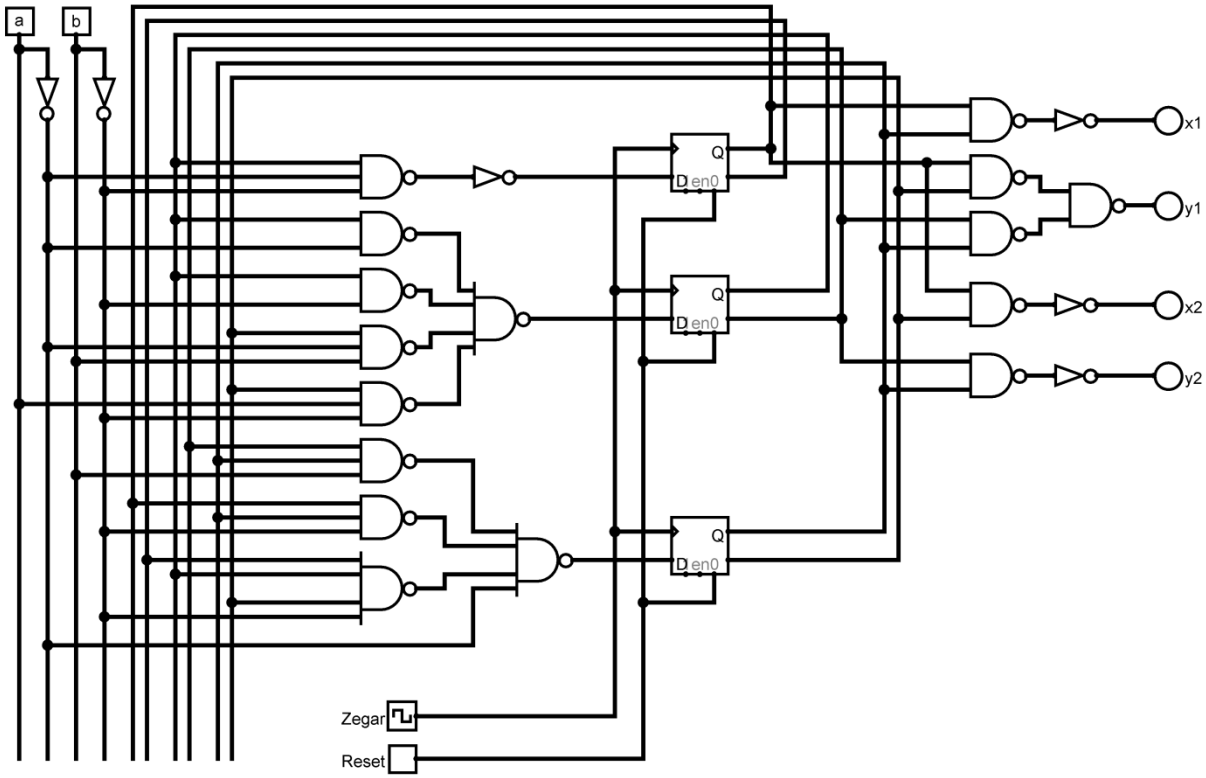
$$x_2 = Q_1 \cdot \overline{Q_3} \tag{42}$$

Projekt „Dostosowanie kształcenia w Politechnice Świętokrzyskiej do potrzeb współczesnej gospodarki” nr FERS.01.05-IP.08-0234/23

$$y_2 = \overline{Q_2} \cdot Q_3 \tag{43}$$

Schemat układu:

[Tekst alternatywny. Schemat projektowanego układu sekwencyjnego, składający się z bramek logicznych typu NAND i NOT oraz przerzutników typu D.]



Rys. 26. Schemat układu

4. Literatura

1. Gorzałczany M.B. (2002), *Układy cyfrowe – metody syntezy. Tom I: Elementy, Układy kombinacyjne*, Wydawnictwo Politechniki Świętokrzyskiej, skrypt nr 378, Kielce.
2. Gorzałczany M.B. (2002), *Układy cyfrowe – metody syntezy. Tom II: Układy sekwencyjne, Układy mikroprogramowane*, Wydawnictwo Politechniki Świętokrzyskiej, skrypt nr 392, Kielce.