

INSTYTUT CYBERNETYKI TECHNICZNEJ POLITECHNIKI WROCŁAWSKIEJ
ZAKŁAD SZTUCZNEJ INTELIGENCJI I AUTOMATÓW

Ćwiczenia laboratoryjne z Logiki Układów Cyfrowych

ćwiczenie 202

temat: UKŁADY KOMBINACYJNE

1. CEL ĆWICZENIA

Ćwiczenie ma na celu praktyczne zapoznanie studentów z budową, działaniem, właściwościami oraz syntezą podstawowych układów kombinacyjnych, takich jak: szyfratory, deszyfratory, transkodery, sumatory, komparatory oraz układy kontroli parzystości.

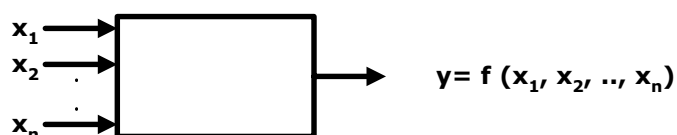
2. PROGRAM ĆWICZENIA

1. Synteza zadanych układów kombinacyjnych z uwzględnieniem zadanych parametrów.
2. Realizacja techniczna układów z zastosowaniem elementów scalonych TTL serii UCY-74.
3. Analiza zbudowanych układów.

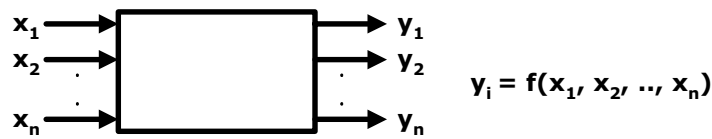
3. PROBLEMATYKA ĆWICZENIA

Układ kombinacyjny jest układem cyfrowym, w którym każda kombinacja sygnałów wejściowych jednoznacznie określa kombinację sygnałów wyjściowych. W odróżnieniu od układów sekwencyjnych układ kombinacyjny nie posiada właściwości pamiętania stanów wewnętrznych układu – jest automatem bez pamięci.

Każdy układ kombinacyjny można zbudować na wiele sposobów różniących się takimi parametrami jak: niezawodność, czas propagacji sygnałów, koszt układu, rodzaj zastosowanych elementów, obciążalności układu itp.. W najprostszym przypadku syntezę układu kombinacyjnego o n wejściach i m wyjściach można przeprowadzić przedstawiając go jako zespół m funkcji przełączających (boolowskich) jednowyjściowych. Syntezę takiego układu można wówczas ograniczyć do realizacji każdej z tych funkcji oddzielnie, co prowadzi jednak do wielu przypadkowych rozwiązań nie najlepszych ze względu na koszt i złożoność układu. Rozwiązanie korzystne uzyskuje się stosując specjalne metody syntezy, które pozwalają minimalizować liczbę elementów logicznych.



Układ kombinacyjny o jednym wyjściu



Układ kombinacyjny o wielu wyjściach

4. WIADOMOŚCI PODSTAWOWE

4.1 Wstęp

Teoria projektowania systemów cyfrowych opiera się na dziale matematyki zwanym algebrą Boole'a. Algebra Boole'a jest wygodnym narzędziem do opisu elementów dwustanowych i sieci złożonych z takich elementów. Każda ze zmiennych boolowskich może przyjmować wartość zero lub jeden. Algebra Boole'a stosowana do analizy układów cyfrowych opiera się na trzech zasadniczych funkcjach:

- iloczynnie logicznym AND,
- sumie logicznej OR,
- negacji NOT.

Funkcją boolowską (logiczną, przełączającą) nazywamy funkcję, której zmienne i ona sama przyjmują wartości ze zbioru $\{0, 1\}$.

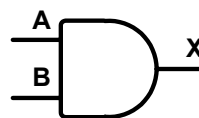
4.2 Bramki logiczne

Bramkami nazywane są kombinacyjne układy cyfrowe realizujące proste funkcje logiczne jednej lub wielu zmiennych logicznych. Zmienną logiczną nazywamy sygnał elektryczny występujący na wejściach i wyjściach układów.

Bramka AND

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

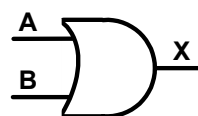
$$X = AB$$



Bramka OR

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

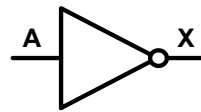
$$X = \overline{\overline{A} \overline{B}} = \overline{\overline{A+B}} = A+B$$



Bramka NOT

A	X
0	1
1	0

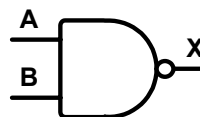
$$X = \bar{A}$$



Bramka NAND

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

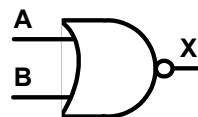
$$X = \overline{AB}$$



Bramka NOR

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

$$X = \overline{\bar{A} + \bar{B}} = \overline{A + B}$$



Bramka Ex-OR

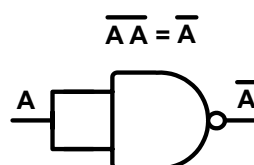
A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

$$X = \overline{A}B + A\overline{B} = A \oplus B$$

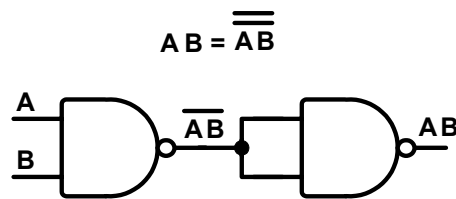
4.3 Logika NAND

Ze względu na fakt, że bramka NAND jest podstawową bramką technologii TTL często zachodzi potrzeba realizacji różnych funkcji logicznych za pomocą bramek NAND. Wszystkie operacje logiczne można wyrazić za pomocą operacji NAND.

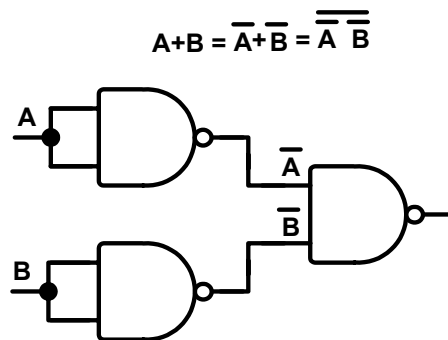
Realizacja funkcji NOT za pomocą bramek NAND (4.1)



Realizacja funkcji AND za pomocą bramek NAND (4.2)



Realizacja funkcji OR za pomocą bramek NAND (4.3)



4.3.1 Przejście z układów AND/OR/NOT na układy NAND

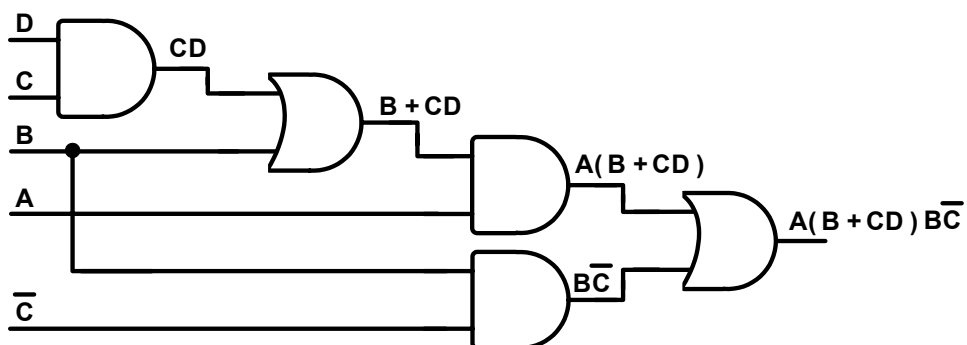
Rozpatrzmy metodę diagramów logicznych na następującym przykładzie. Należy zrealizować układ realizujący następującą funkcję:

$$F = A (B + CD) + B/C$$

krok 1

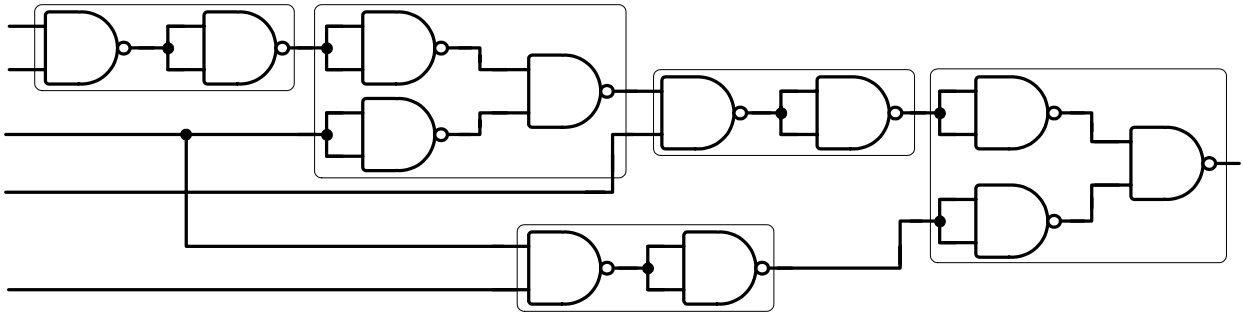
Rysujemy schemat logiczny układu z użyciem funkcyj logicznych dowolnego rodzaju (AND, OR, NOT):

$$F = A (B + CD) + B\overline{C}$$



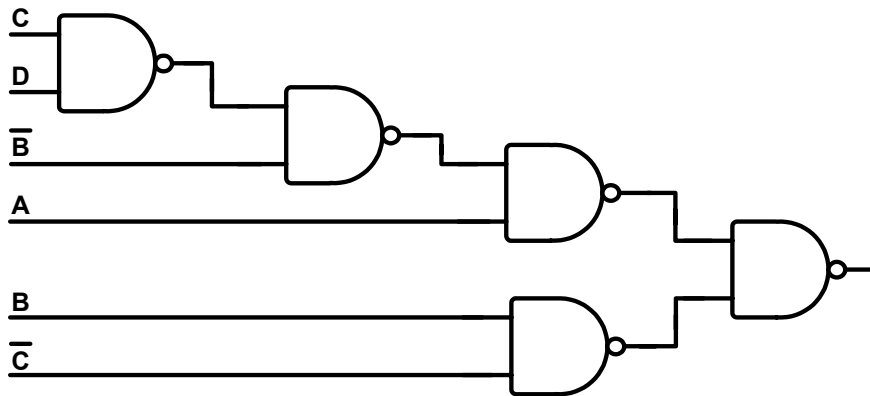
krok 2

Elementy AND, OR, NOT zastępujemy układami NAND (zgodnie z zależnościami 4.1, 4.2 i 4.3):



krok 3

Minimalizujemy układ eliminując za schematu inwertery połączone kaskadowo (zaznaczone liniami na rysunku w kroku 2)

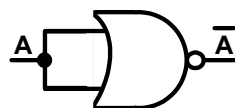


4.4. Logika NOR

Wszystkie operacje logiczne można wyrazić również za pomocą operacji NOR.

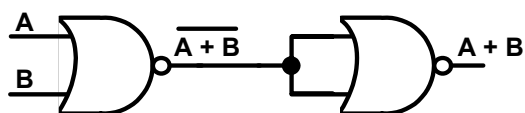
Realizacja funkcji NOT za pomocą bramek NOR (4.4)

$$A + \bar{A} = \bar{A}$$



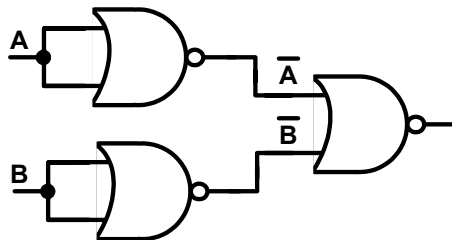
Realizacja funkcji OR za pomocą bramek NOR (4.5)

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



Realizacja funkcji AND za pomocą bramek NOR (4.6)

$$AB = \overline{\overline{A} \overline{B}} = \overline{\overline{A} + \overline{B}}$$



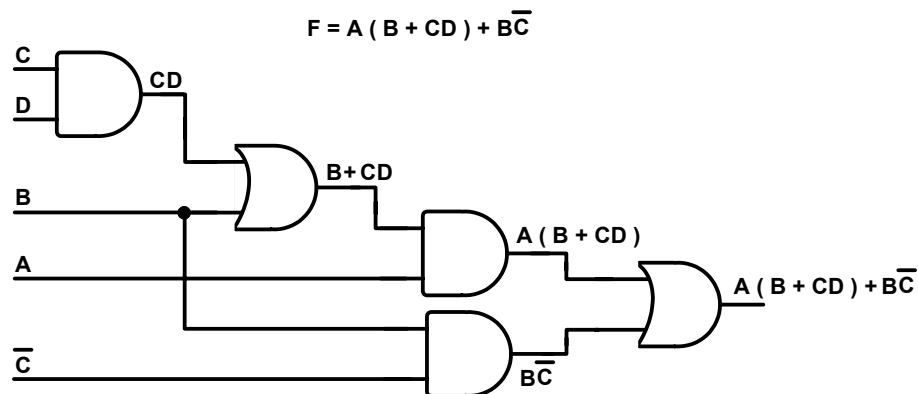
4.4.1 Przejście z układów AND/OR/NOT na układy NOR

Rozpatrzmy metodę diagramów logicznych na następującym przykładzie. Należy zrealizować układ realizujący następującą funkcję:

$$F = A (B + CD) + B/C$$

krok 1

Rysujemy schemat logiczny układu z użyciem funkcyj logicznych dowolnego rodzaju (AND, OR, NOT):



krok 2

Elementy AND, OR, NOT zastępujemy układami NOR (zgodnie z 4.4, 4.5 i 4.6):

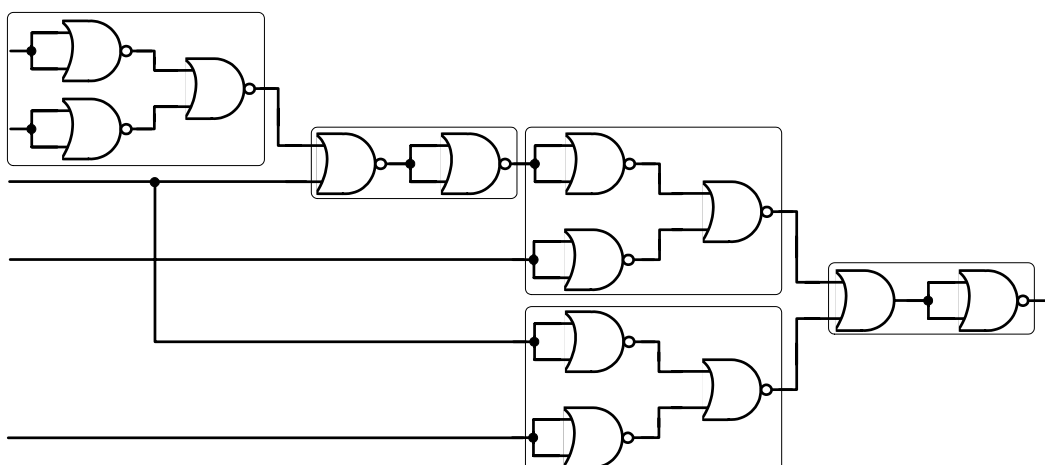


Tabela 4.5.2 Kody dwójkowo-dziesiętne

cyfra	naturalny BCD 8421	Aikena 2421	plus 3
0	0000	0000	0011
1	0001	0001	0100
2	0010	0010	0101
3	0011	0011	0110
4	0100	0100	0111
5	0101	1011	1000
6	0110	1100	1001
7	0111	1101	1010
8	1000	1110	1011
9	1001	1111	1100

Tabela 4.5.3 Kod 1 z 10

cyfra	kod 1 z 10
0	000000001
1	000000010
2	000000100
3	000001000
4	000010000
5	000100000
6	001000000
7	001000000
8	010000000
9	100000000

Tabela 4.5.4 Kod U2 (4-bitowy)

liczba	kod U2	liczba	kod U2
0	0000		
1	0001	-1	1111
2	0010	-2	1110
3	0011	-3	1101
4	0100	-4	1100
5	0101	-5	1011
6	0110	-6	1010
7	0111	-7	1001
		-8	1000

4.6. Konwertery kodów

Szyfratorami nazywamy układy służące do konwersji kodu 1 z n na określony kod wyjściowy. Szyfratory mają n wejść, z których tylko jedno jest wyróżniane w danej chwili.

Deszyfratorem nazywamy układ kombinacyjny n/m , gdzie n oznacza liczbę wejść, a m liczbę wyjść, służący do konwersji kodu wejściowego – innego niż 1 z n - na kod 1 z n . Deszyfrator nazywa się deszyfratorem pełnym, jeśli $2^n = m$ i deszyfratorem niepełnym, jeżeli $2^n > m$.

Transkoderami nazywamy układy służące do konwersji kodu dwójkowego wejściowego innego niż 1 z n na kod wyjściowy inny niż 1 z n .

Przykład 1

Rozpatrzmy transkoder realizujący konwersję 2-bitowego kodu dwójkowego na kod „plus 3”, (tabela 5).

tabela 5

A	B	x_2	x_1	x_0
0	0	0	1	1
0	1	1	0	0
1	0	1	0	1
1	1	1	1	0

$$\begin{aligned}
 x_0 &= \overline{B} \\
 x_1 &= \overline{A} \overline{B} + AB \\
 x_2 &= B + A
 \end{aligned}$$

4.7. Sumatory

Dodawanie liczb dwójkowych wykonuje się według tych samych zasad, co w przypadku liczb dziesiętnych.

Przykład 2

Rozpatrzmy półsumator, czyli układ realizujący dodawanie liczb dwójkowych jednobitowych.

Niech A i B będą składnikami dodawania, C przeniesieniem, a S sumą A i B.



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

$$C = AB$$

$$S = A\bar{B} + \bar{A}B = A \oplus B$$

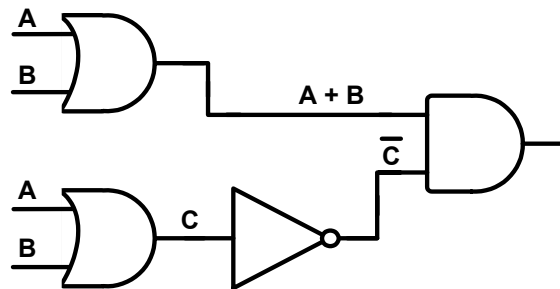
$$S = (A+B)(\bar{A}+\bar{B})$$

$$\bar{C} = \overline{AB} = \bar{A} + \bar{B}$$

$$S = (A+B)C \quad (4.7.1)$$

$$C = AB \quad (4.7.2)$$

Na podstawie 4.7.1 i 4.7.2 otrzymujemy następujący schemat logiczny



W przypadku dodawania wielobitowych liczb dwójkowych należy uwzględnić przeniesienie z pozycji sąsiedniej mniej znaczącej od rozpatrywanej.

4.8. Komparatory

Układy zwane komparatorami liczb służą do porównywania wartości dwu lub więcej liczb dwójkowych. Komparatory mogą występować w różnych postaciach. Jeżeli A i B są porównywanymi liczbami, to komparator może realizować jedną lub kilka z następujących funkcji: $A = B$, $A < B$, $A \leq B$, $A > B$, $A \geq B$.

Przykład 3

Rozpatrzmy komparator jednobitowy realizujący funkcję $A > B$.

A	B	A > B
0	0	0
0	1	0
1	0	1
1	1	0

$$(A > B) = A\bar{B}$$

4.9. Układy generowania i kontroli parzystości

Układy generowania i kontroli bitu parzystości stosuje się przy transmisji danych cyfrowych do wykrywania nieparzystej liczby błędów w kontrolowanym słowie. Parzyste liczby błędów nie są wykrywane.

Generowanie bitu parzystości polega na wytworzeniu jednego bitu i dodaniu go do słowa kodowego, będącego nośnikiem informacji. Bit parzystości przyjmuje wartość 1, jeżeli słowo kodowe zawiera nieparzysta liczbę jedynek.

Kontrola bitu parzystości polega na wytworzeniu bitu parzystości dla odebranego słowa i porównaniu go z odebranym bitem parzystości.

Przykład 4

Rozpatrzmy generator bitu parzystości dla słowa 2-bitowego.

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

$$X = A \oplus B$$

5. PRZEBIEG ĆWICZENIA

O ile prowadzący zajęcia nie poleci inaczej należy zrealizować następujące układy kombinacyjne:

1. Transkoder kodu BCD na kod „plus 3”
2. Sumator dwubitowy
3. Komparator dwubitowy realizujący funkcję $A > B$
4. Generator bitu parzystości dla słowa 4-bitowego

Realizację układów przeprowadzić według następujących punktów:

1. Określić funkcje logiczne.
2. Dokonać minimalizacji funkcji logicznych.
3. Sporządzić schemat logiczny z bramek NOT, AND, OR.
4. Dokonać przejścia na układy NAND
5. Dokonać przejścia na układy NOR
6. Zmontować układy według schematów z punktu 3, 4 i 5
7. Przeanalizować działanie i porównać poszczególne układy.

Zadania dodatkowe

1. Generator bitu parzystości dla słowa 6- i 8-bitowego.
2. Układ następnika dla liczby 4-, 6- i 8-bitowej.
3. Sumator 4-bitowy.
4. Układ zaprzeczenia liczby w kodzie U2.
5. Sumator/subtraktor w kodzie U2.
6. Szyfratory:
 - a. kod 1 z n na kod BCD,
 - b. kod 1 z n na kod 2-4-2-1.
7. Deszyfratory:
 - a. kod BCD na kod 1 z n ,
 - b. kod 2-4-2-1 na kod 1 z n .
8. Transkodery:
 - a. kod Gray'a na kod dwójkowy,
 - b. kod dwójkowy na kod Gray'a,
 - c. kod „plus 3” na kod 2-4-2-1.
9. Komparator liczb 4-, 6- i 8-bitowych

6. SPRAWOZDANIE Z ĆWICZENIA

- Sformułować rozwiązywane problemy i przedyskutować wybrany sposób rozwiązania
- Przedstawić kolejne etapy syntezy układów (tablice stanów, tablice Karnaugh'a, schematy logiczne),
- Opisać sposób i wyniki testowania zbudowanych układów,
- Sformułować wnioski

LITERATURA

1. J. Pieńkoś, J. Turczyński, Układy scalone TTL w systemach cyfrowych. WKŁ, Warszawa 1986
2. W. Sasal, Układy scalone serii UCY 64/UCY 74, Parametry i zastosowania, WKŁ, Warszawa 1985
3. J. Kazimierczak, J. Kluska, A. Kaczmarek, Podstawy teorii automatów. Laboratorium, Wydawnictwo Politechniki Rzeszowskiej, Rzeszów 1984
4. W. Krasiński, Doświadczenia z podstaw techniki cyfrowej, Politechnika Wrocławska, Wrocław 1988