

Zespół Szkół Technicznych im. J. i J. niadeckich w Grudzi dzu



Laboratorium Elektryczne . Pracownia
Automatyki i Robotyki (s.48)

Instrukcja Laboratoryjna:
15. UK/ ADY CYFROWE (A, B)

Opracował mgr in . Marcin Jabło ski

Cz A: Projektowanie i symulacja komputerowa układów logicznych

Cz B: Badania laboratoryjne wybranych układów cyfrowych

Spis treści

I. Podstawowe zagadnienia z teorii układów cyfrowych

1. Wstęp
2. Funkcje logiczne
3. Bramki logiczne
4. Projektowanie układów logicznych
 - 4.1 Podstawowe prawa algebry Boole'a
 - 4.2 Układanie równań boolowskich i ich przekształcanie do postaci minimalnej
 - 4.3 Realizacja układowa funkcji boolowskiej na bramkach AND, OR, NOT
 - 4.4 Realizacja układowa funkcji boolowskiej na bramkach NAND
5. Liczby i kody binarne
6. Przegląd technologii układów cyfrowych scalonych
7. Omówienie układów cyfrowych zastosowanych w wiczeniu
 - 7.1 Przerzutniki
 - 7.2 Licznik i dzielnik częstotliwości
 - 7.3 Multiplexer
 - 7.4 Wskaźniki półprzewodnikowe i układy sterujące nimi

II. Wykonanie wiczenia

Cz A. Projektowanie i symulacja komputerowa układów logicznych

Cz B. Badania laboratoryjne wybranych układów cyfrowych

1. Określenie typu bramki na podstawie tablicy prawdy
2. Pomiar charakterystyki przejściowej bramki NAND z wykorzystaniem układu UCY 7400
2. Badanie przerzutnika SR
3. Wykorzystanie bramek NAND do budowy generatora fali prostokątnej
5. Badanie dzielnika częstotliwości
6. Badanie multiplexera
7. Badanie wskaźnika 7-segmentowego i dekodera sterującego
8. Badanie licznika o programowanej długości cyklu z wyświetlaniem liczby impulsów

III. Literatura

IV. Pytania kontrolne

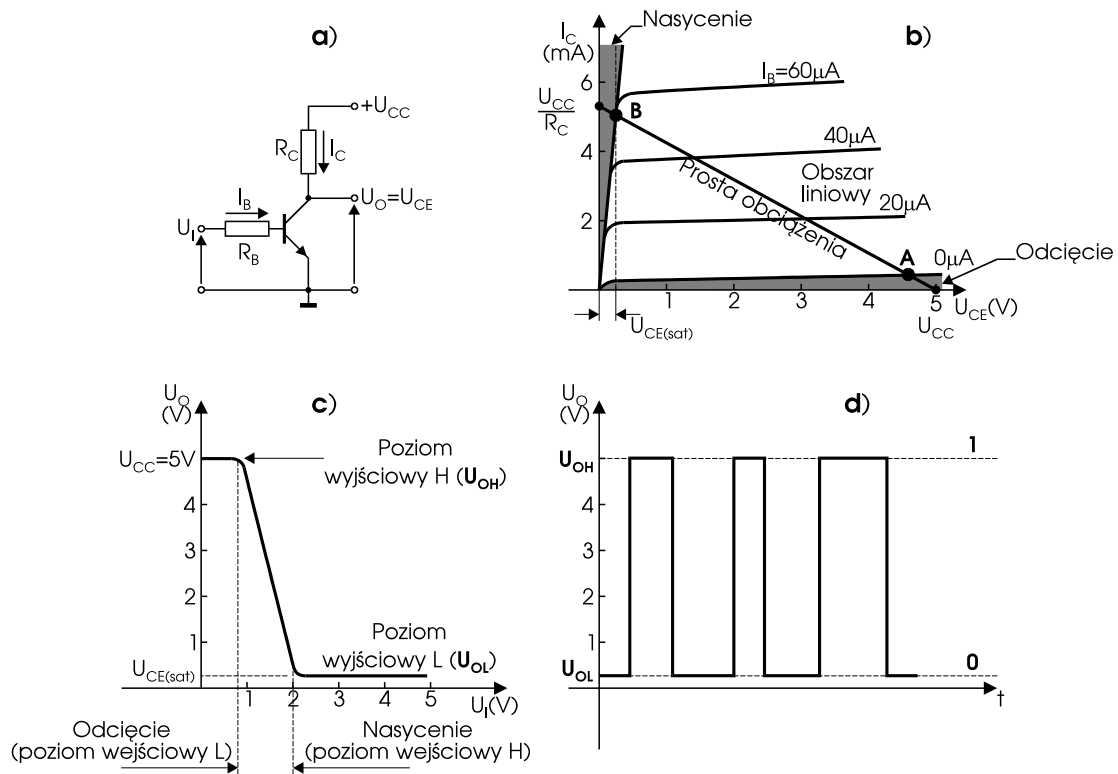
I. Podstawowe zagadnienia z teorii układów cyfrowych

1. Wstęp

Muzyka na płytach fonograficznych jest zapisana w formie kanałiku o zmiennym urze bieniu. Ruch igły prowadzonej przez kanałik odbywa się w sposób *ciężki* ale ze zmieniającą się amplitudą i częstotliwici - odzwierciedlając zapisane frazy muzyczne. Tak zarejestrowany sygnał może przybrać praktycznie dowolne wartości z pewnego szerokiego zakresu i nazywany jest *sygnałem analogowym*.

W przeciwieństwie, muzyka na dyskach kompaktowych jest zapisana w formie pewnego wzoru składającego się z płaskich obszarów oraz dziur, które albo odbijają światło albo nie. Tak zarejestrowany sygnał nazywany jest *sygnałem dyskretnym* albo *sygnałem cyfrowym*. Sygnał cyfrowy przybiera wyłącznie dwie wartości.

Elektroniczne układy cyfrowe służą do przetwarzania sygnałów cyfrowych. Ich podstawowym elementem jest tranzystor pracujący jako przełącznik. Działanie takiego tranzystora ma charakter impulsowy - znajduje się on, na przemian, albo w *stanie włączenia* - ang. *ON* (tranzystor nasycony), albo w *stanie wyłączenia* - ang. *OFF* (tranzystor odcięty). **Rys. 1.1** przedstawia pracę tranzystora w układzie przełącznika dwustanowego.



Rys. 1.1 Tranzystor jako przełącznik: a) schemat układu, b) obszary pracy tranzystora, c) charakterystyka przejściowa, d) sygnał cyfrowy.

Gdy do wejścia układu z **rys. 1.1** jest doprowadzone napięcie $U_i < 0,7V$, czyli niski poziom napięcia, wówczas tranzystor **nie przewodzi** (jest odcięty - punkt pracy **A**) i na jego kolektorze występuje napięcie bliskie U_{CC} , czyli wysoki poziom napięcia - U_{OH} . Jeżeli natomiast napięcie wejściowe U_i ma odpowiednio dużą wartość, czyli na wejściu występuje wysoki poziom napięcia, to tranzystor **przewodzi** (jest nasycony - punkt pracy **B**) i na jego kolektorze występuje napięcie nasycenia $U_{CE(sat)}$, czyli niski poziom napięcia - U_{OL} . Tranzystor w tym układzie pracuje jak typowy przełącznik dwustanowy (*OFF* - *ON*) którego kontaktami są końcówki emitera i kolektora. Rezystancja pomiędzy kolektorem i emiterem zmienia się, przy przejściu od stanu wyłączenia do stanu włączenia, od wartości powyżej $5M\Omega$ do wartości poniżej 50Ω . Z przebiegu charakterystyki przejściowej układu wyraźnie widać odwrócenie poziomów napięcia na wyjściu w stosunku do wejścia układu. Układ o takim rodzaju pracy jest nazywany **inwerterem**. Inwerter stanowi najbardziej elementarny komponent scalonych układów cyfrowych.

Wysoki poziom napięcia wyjściowego U_{OH} (tranzystor wyłączony) odpowiada stanowi logicznemu **HIGH** albo **1**, natomiast niski poziom napięcia wyjściowego U_{OL} (tranzystor włączony) odpowiada stanowi logicznemu **LOW** albo **0**. Wartości napięć wyjściowych, przedstawionych na

charakterystyce przejściowej, odpowiadającej stanowi logicznemu **1** oraz **0** (około 5V oraz około 0,2V) są typowe dla zdecydowanej większości układów cyfrowych.

Obecnie, w coraz szerszym zakresie, **technika analogowa** jest zastępowana przez **technikę cyfrową**, szczególnie w tych urządzeniach (systemach), gdzie wszelkiego rodzaju dane muszą być zgromadzone w pamięci. Zaletą systemów cyfrowych jest zdolność do przetwarzania danych wejściowych (np. przez dokonanie obliczeń), do podejmowania decyzji (np. przez wypracowanie sygnałów sterowania) oraz do wyświetlania komunikatów w postaci liczb i/lub liter. **Przetwarzanie informacji** jest bardzo znaczącym składowym wszytkich gałęzi nauki i techniki.

Układy cyfrowe potrafią przetwarzać sygnały cyfrowe w sposób prosty i jednocześnie niezwykle szybko. Automatyka, robotyka, komputery, telekomunikacja - to dziedziny techniki w których obserwujemy najbardziej gwałtowny rozwój, głównie dzięki stosowaniu coraz nowszych generacji układów cyfrowych, z każdym razem szybszych i o większym stopniu scalenia. Pojedynczy chip 64-bitowego mikroprocesora *Pentium* zawiera 3,2 miliona tranzystorów i może być taktowany zegarem o częstotliwości 100MHz. Układy o tej skali scalenia nazywają się układami **VLSI** (ang. *Very Large Scale Integration*).

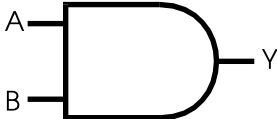
2. Funkcje logiczne

Teoria układów cyfrowych, traktowanych jako układy logicznego działania, opiera się na dwuelementowej **algebrze Boole'a**. Zastosowanie zasad i praw tej algebry umożliwia wykonanie analitycznej syntezy skomplikowanych funkcji logicznych i zbudowanie odpowiadających im schematów logicznych układów cyfrowych. Przykładowo, procesor jest sprężyną (układem) realizacją swojej własnej listy rozkazów.

Funkcja logiczna nazywa się funkcją, której argumenty (zmienne logiczne) oraz sama funkcja mogą przybierać tylko jedną z dwóch wartości, np. **0** (inaczej ang. *FALSE* - czyli **Fałszywe**) lub **1** (inaczej ang. *TRUE* - czyli **Prawdziwe**). Argumenty funkcji oznaczamy zazwyczaj literami **A**, **B**, **C**, ... Wartości argumentów funkcji logicznej odpowiadają **stanom wejściowym** układu cyfrowego, a wartości samej funkcji - **stanom wyjściowym** tego układu.

Funkcja logiczna może być zadana za pomocą **opisu słownego**, tablicy wartości funkcji - tzw. **tablicy prawdy** (ang. *truth table*), analitycznie w postaci **wyrażenia algebraicznego** (równania boolowskiego) lub graficznie za pomocą **symboli logicznych**. Sposoby wyrażania funkcji logicznych przedstawiono w **tablicy 2.1**.

Tablica 2.1 Sposoby wyrażania funkcji logicznych

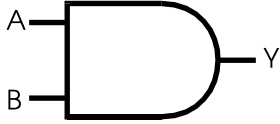
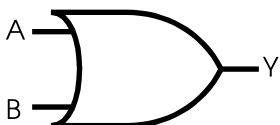
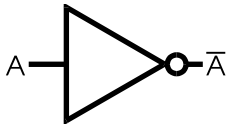
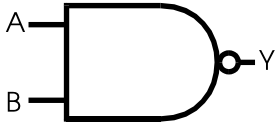
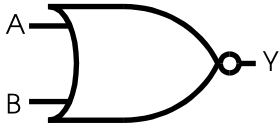
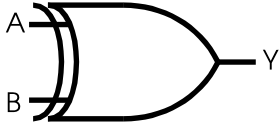
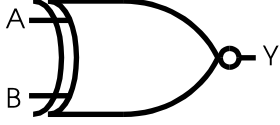
Opis słowny	Iloczyn logiczny argumentów A i B jest równy 1 , gdy A=1 i B=1															
<p>Tablica prawdy - Truth table</p> <p>Zawiera wszystkie kombinacje możliwych stanów wejściowych układu logicznego i odpowiadające im stany wyjściowe tego układu. Liczba stanów układu jest równa 2^n, gdzie n jest liczbą wejściowych linii.</p>	<table border="1"> <thead> <tr> <th>A</th> <th>B</th> <th>Y</th> </tr> </thead> <tbody> <tr> <td>0</td> <td>0</td> <td>0</td> </tr> <tr> <td>0</td> <td>1</td> <td>0</td> </tr> <tr> <td>1</td> <td>0</td> <td>0</td> </tr> <tr> <td>1</td> <td>1</td> <td>1</td> </tr> </tbody> </table>	A	B	Y	0	0	0	0	1	0	1	0	0	1	1	1
A	B	Y														
0	0	0														
0	1	0														
1	0	0														
1	1	1														
<p>Wyrażenie algebraiczne</p> <p>(czytaj: A i B równa się Y)</p>	$A \cdot B = Y$															
<p>Symbol logiczny</p> <p>(nazwa skrótowa - AND)</p>																

Podstawowymi funkcjami logicznymi są: **AND** (iloczyn logiczny), **OR** (suma logiczna) oraz **NOT** (negacja). Za ich pomocą można opisać dowolnie skomplikowany układ logiczny.

3. Bramki logiczne

Bramki logiczne - ang. *gates* (nazywane tak e *funktorami logicznymi*) s najprostszymi ukłdami cyfrowymi realizuj cymi elementarne funkcje logiczne. S one do budowy ukłdów logicznych o wi kszej z ciono ci. Podstawowe bramki logiczne, ich nazwy, symbole graficzne, opis algebraiczny oraz tablice prawdy przedstawiono w **tablicy 3.1**.

Tablica 3.1 Podstawowe bramki logiczne

FUNKCJA LOGICZNA	SYMBOL LOGICZNY	WYRA ENIE ALGEBRAICZNE	TRUTH TABLE		
			Inputs		Output
			A	B	Y
AND		$A \cdot B = Y$	0	0	0
			0	1	0
			1	0	0
			1	1	1
OR		$A + B = Y$	0	0	0
			0	1	1
			1	0	1
			1	1	1
NOT (Inverter)		$A = \bar{A}$	0		1
			1		0
NAND		$\overline{A \cdot B} = Y$	0	0	1
			0	1	1
			1	0	1
			1	1	0
NOR		$\overline{A + B} = Y$	0	0	1
			0	1	0
			1	0	0
			1	1	0
XOR		$A \oplus B = Y$	0	0	0
			0	1	1
			1	0	1
			1	1	0
XNOR		$\overline{A \oplus B} = Y$	0	0	1
			0	1	0
			1	0	0
			1	1	1

Bramki AND, OR, NOT tworz tzw. *funkcjonalnie pe y* zestaw elementów. Oznacza to, e mo na z nich zbudowa dowolnie z ciony ukłd logiczny.

Za pomoc wy cycznie bramek NAND albo wy cycznie bramek NOR mo na tak e zrealizowa dowolnie z cion funkcj , w tym równie funkcje podstawowe AND, OR, NOT. Z tego powodu mówimy, e bramki NAND i NOR oddzielnie tworz tzw. *minimalny* zestaw *funkcjonalnie pe y*. W pierwszej chwili mo e si wydawa , e stosowanie funktorów NAND lub NOR do realizacji prostych funkcji **iloczynu**, **sumy** i **negacji** jest niepotrzebne i tylko komplikuje postawione zadanie. W praktyce okazuje si , e zalety stosowania jednoelementowego zbioru do realizacji dowolnej funkcji logicznej s bardzo du e. Przegl daj c katalogi producentów ukłdów cyfrowych mo na z ctwo ci zauwa y ,

W najszerszym zakresie ofert stanowi bramki **NAND**, gdyż są one najczęściej stosowane przez użytkowników (można powiedzieć, że bramka **NAND** jest bramką uniwersalną).

Bramki **XOR** oraz **XNOR**, ze względu na swój specyficzny sposób do budowy układów arytmetycznych (sumatory binarne) oraz układów detekcji błędów przesyłu informacji cyfrowej (układy z kontrolą bitu parzystości).

Bramki, jako układy cyfrowe o niezbyt złożonej strukturze, należą do układów o małej skali scalenia, tzw. **SSI** (ang. *Small Scale Integration*).

4. Projektowanie układów logicznych

Jednym z najbardziej fascynujących aspektów **elektroniki cyfrowej** jest możliwość konstrukcji układów o prostej "mentalności", ale zdolnych do przetwarzania informacji z niezwykłą szybkością (w porównaniu z człowiekiem) produkcyjnie. Typowy komputer potrafi wykonać tysiąc dodawek 10-cio pozycyjnych liczb w ciągu sekundy.

Zagadnienie projektowania układów logicznych sprowadza się do utworzenia schematu układu, zawierającego odpowiednio połączonych bramek logicznych, zdolnego do zrealizowania po danej funkcji logicznej.

Proces projektowania rozpoczyna się od **opisu słownego** funkcji logicznej lub od **tablicy prawdy**. W kolejnych etapach projektant określa funkcję logiczną w postaci **wyrażenia algebraicznego** (równania boolowskiego), przekształca je do postaci **minimalnej** (najprostszej), a następnie, bezpośrednio z uproszczonego równania rysuje **schemat układu** używając bramek **AND**, **OR**, **NOT**. W praktyce taki schemat wymaga najczęściej **przekształcenia** w realizację układów złożonych wyłącznie z bramek **NAND**. Przedstawiony sposób postępowania odnosi się do projektowania niezbyt złożonych układów logicznych.

4.1 Podstawowe prawa algebry Boole'a

Prawie 100 lat przed pierwszym cyfrowym komputerem, George Boole, angielski matematyk (1815-1864), sformułował matematyczne reguły analizy zdań logicznych (zdania mogą być wyłącznie **prawdziwe/fałszywe**, *true/false*). Określenie **TRUE** i **FALSE** stanowi, historycznie biorąc, pierwszy rodzaj klasyfikacji dwustanowej. **Algebra Boole'a** umożliwia manipulowanie binarnymi zmiennymi **1**, **0** za pomocą związków logicznych **AND**, **OR**, **NOT**. Spośród wielu praw algebry Boole'a podstawowe znaczenie w zastosowaniu do teorii układów cyfrowych mają prawa i to samo ci przedstawione w **tablicy 4.1**.

Tablica 4.1 Podstawowe prawa i to samo ci Algebry Boole'a		Wzór odniesiony do iloczynu logicznego	Wzór odniesiony do sumy logicznej
Prawa	przemienności	$A \cdot B = B \cdot A$	$A + B = B + A$
	łączności	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$	$A + (B + C) = (A + B) + C$
	rozdzielczości	$A \cdot (B + C) = A \cdot B + A \cdot C$	$A + B \cdot C = (A + B) \cdot (A + C)$
	De Morgana	$\overline{A \cdot B \cdot \dots} = \overline{A} + \overline{B} + \dots$	$\overline{A + B + \dots} = \overline{A} \cdot \overline{B} \cdot \dots$
To samo ci	podstawowe	$A \cdot 0 = 0$ $A \cdot 1 = A$ $A \cdot \overline{A} = 0$ $A \cdot A = A$	$A + 1 = 1$ $A + 0 = A$ $A + \overline{A} = 1$ $A + A = A$
	dotatkowe	$A \cdot (\overline{A} + B) = A \cdot B$ $A + \overline{A} \cdot B = A + B$ $(A + B) \cdot (\overline{A} + \overline{B}) = \overline{A \cdot B}$	$A + \overline{A} \cdot B = A + B$ $A \cdot (\overline{A} + B) = A \cdot B$ $A \cdot B + \overline{A} \cdot \overline{B} = \overline{A + B}$

Symbolem "." oznacza si operacj iloczynu logicznego **AND**, symbolem "+" oznacza si operacj sumy logicznej **OR**, symbolem "-" oznacza si operacj negacji **NOT**.

Zale no ci przedstawione w **tablicy 4.1** wykorzystuje si przy przekszta ceniu wyra e opisuj cych z one funkcje logiczne o wielu zmiennych w celu otrzymania ich w mo liwie najprostszej postaci ko cowej, a co za tym idzie, prostszej realizacji uk adowej. Proces ten jest okre lany jako *minimalizacja funkcji logicznej*.

4.2 Uk adanie równa boolowskich i ich przekszta cenie do postaci minimalnej

Postawienie zadania zbudowania uk adu logicznego polega najcz ciej na **opisie sownym**, od którego przechodzi si nast pnie do okre lenia **tablicy prawdy**. Na podstawie tablicy prawdy mo na w sposób rutynowy u c y **równania boolowskie** opisuj ce w sposób algebraiczny dzia anie logiczne uk adu. Post powanie to zostanie zilustrowane w formie przyk adu.

PRZYK/ AD: W samolocie, dla zwi kszenia poziomu ufno ci, zastosowano potrójne systemy czujników. Zaprojektowa uk ad logiczny uruchamiaj cy automatycznego pilota wy cnie wtedy, gdy co najmniej dwa z tych systemów s aktywne. *Tekst powy szy jest typowym opisem sownym pewnego zadania logicznego.*

Systemy czujników oznaczymy literami A, B, C. S to **zmienne** funkcji logicznej czyli **wej cia** uk adu logicznego. Stanowi aktywnemu czujnika przypisujemy poziom **1 (Prawdziwy - TRUE)**, w przeciwnym wypadku - poziom **0 (Fa czywy - FALSE)**.

Sygna uruchamiaj cy automatycznego pilota oznaczymy liter Y. Jest to **warto** funkcji logicznej czyli **wyj cie** uk adu logicznego. Sygna wi uruchomienia automatycznego pilota przypisujemy poziom **1 (Prawdziwy - TRUE)**, w przeciwnym wypadku - poziom **0 (Fa czywy - FALSE)**.

Inputs			Output
A	B	C	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Okre lamy **tablic** **prawdy** uk adu logicznego wypisuj c wszystkie mo liwe kombinacje stanów wej i odpowiadaj ce im stany wyj cia (zgodnie z logik zawart w opisie sownym).

Ka d funkcj logiczn mo na przedstawi w postaci roz c onej na tzw. **sk adniki jedyinki**. Posta uzyskana w wyniku rozk adu na **sk adniki jedyinki** jest nazywana **postaci kanoniczn sumy**.

Kanoniczn posta sumy mo na otrzyma na podstawie **tablicy prawdy**, bior c pod uwag jedynie te wiersze, dla których **Y=1** i przypisuj c warto ciom **1** - zmienne niezanegowane, a warto ciom **0** - negacje zmiennych. Post puj c zgodnie z powy sz

recept otrzymamy dla naszej tablicy prawdy **równanie boolowskie**, które w sposób algebraiczny opisuje zadany problem logiczny:

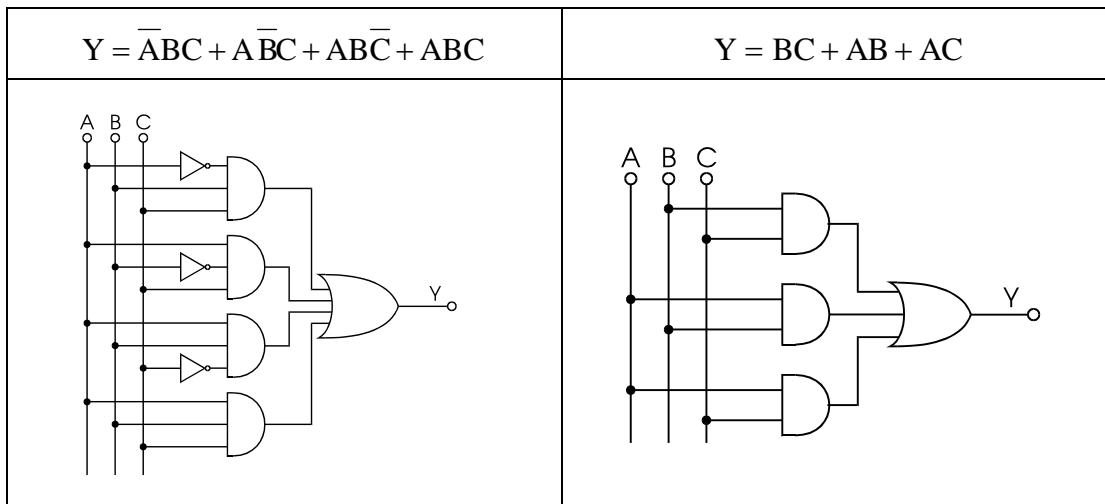
$$Y = \bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C + ABC$$

Równanie to poddamy teraz procesowi **minimalizacji funkcji logicznej**. Minimalizacja funkcji logicznej polega na takim przekszta ceniu postaci kanonicznej funkcji, zgodnie z zasadami **algebry Boole'a**, aby uzyska mo liwie najprostszy jej zapis. Ka de uproszczenie równania boolowskiego umo liwia ctwiejsz realizacj uk adów funkcji - przy u yciu mniejszej liczby elementarnych bramek logicznych. *Metoda przekszta cania wyra e algebraicznych jest metod intuicyjn i, istniej równie inne metody minimalizacji funkcji logicznej (sformalizowane), o których nie b dziemy tutaj mówi . Spróbujmy przekszta ci nasze równanie stosuj c prawa i to samo ci algebry Boole'a:*

$$\begin{aligned} Y &= \bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C + ABC = BC(\bar{A} + A) + A\bar{B}\bar{C} + A\bar{B}C = BC + A\bar{B}\bar{C} + A\bar{B}C = \\ &= B(C + A\bar{C}) + A\bar{B}\bar{C} = B(C + A) + A\bar{B}\bar{C} = BC + AB + A\bar{B}\bar{C} = BC + A(B + \bar{B}\bar{C}) = \\ &= BC + A(B + C) = BC + AB + AC \end{aligned}$$

4.3 Realizacja układowa funkcji boolowskiej na bramkach AND, OR, NOT

Dysponując równaniem boolowskim opisującym logikę układu można bezpośrednio narysować schemat układu wykorzystując do tego celu bramki podstawowe **AND**, **OR**, **NOT**. Na **rys. 4.1** przedstawiono dwie realizacje układowe dotyczące naszego przykładu, które narysowano na podstawie równania pełnego i zminimalizowanego.



Rys. 4.1 Realizacje układowe równań boolowskich z zastosowaniem bramek **AND**, **OR**, **NOT**

Jak łatwo zauważyć, schemat układu wykonany na podstawie funkcji uproszczonej zawiera mniej bramek i jego realizacja praktyczna jest na pewno prostsza i tańsza.

4.4 Realizacja układowa funkcji boolowskiej na bramkach NAND

Bramki **NAND** stosowane są do budowy układów cyfrowych częściej niż bramki **AND**, **OR**, **NOT**. Są one bowiem pod względem układowym lepsze, zapewniając stały poziom sygnału wyjściowego. Ponadto bramki **NAND** są **bramkami uniwersalnymi**, gdyż przy ich pomocy można zrealizować dowolny system cyfrowy. Dla wykazania, że każda funkcja boolowska może być zrealizowana za pomocą bramek **NAND** wystarczy wykazać, że operacje logiczne **AND**, **OR**, **NOT** dają się zrealizować przez układy zawierające wyłącznie bramki **NAND**. W **tablicy 4.2** przedstawiono **NAND**-owe realizacje układowe operacji **AND**, **OR**, **NOT** wraz z wyrażeniami algebraicznymi opisującymi przejście sygnału cyfrowego od wejścia układu do jego wyjścia.

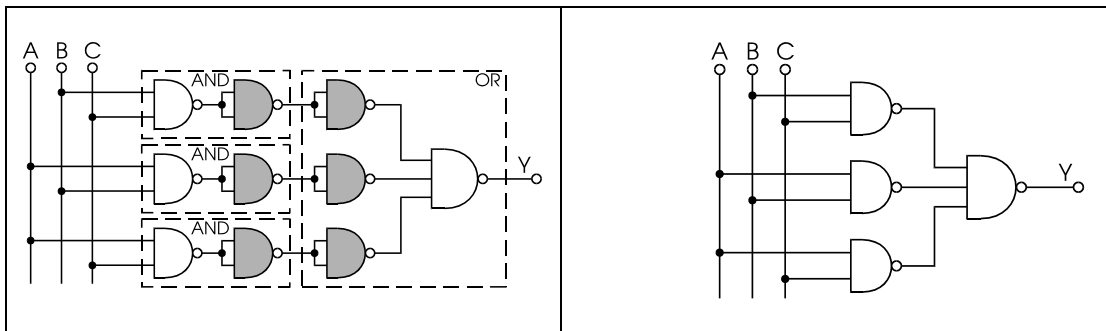
Tablica 4.2 Funkcje logiczne **AND**, **OR**, **NOT** w realizacji układowej z bramek **NAND**

AND	OR	NOT (inverter)
$\overline{\overline{A \cdot B}} = A \cdot B$	$\overline{\overline{A \cdot B}} = \overline{\overline{A + B}} = A + B$	$\overline{\overline{A \cdot A}} = \overline{A}$

Aby zrealizować funkcję boolowską za pomocą bramek **NAND**, zastosować można na prosty **metodę przekształcania schematów logicznych**. Metoda ta wymaga co prawda narysowania dwóch schematów logicznych przed otrzymaniem właściwego schematu realizacji **NAND**-owej, lecz jest ona w sumie bardzo prosta:

1. Na podstawie danego równania boolowskiego rysujemy schemat logiczny zawierający bramki **AND**, **OR**, **NOT**.
2. Rysujemy drugi schemat logiczny, w którym każda z bramek **AND**, **OR**, **NOT** zastąpimy przez układ równoważny, zbudowany z bramek **NAND**, zgodnie z **tablic 4.2**.
3. Usuwamy ze schematu wszystkie pary kaskadowo połączonych **inwerterów** (gdy jak wiadomo $\overline{\overline{A}} = A$). Otrzymany schemat jest poszukiwaną realizacją typu **NAND**.

Dla naszego przykładu przekształcimy zminimalizowaną wersję schematu logicznego z **rys. 4.1** w realizację **NAND-ową**. Przekształcenie to przedstawiono na **rys. 4.2**. Na rysunku tym zacięziono kaskadowo połączone inwertery, które ostatecznie zostały usunięte.



Rys. 4.2 Realizacja układowa z zastosowaniem bramek NAND

5. Liczby i kody binarne

Liczby binarne

W **systemie dziesiętnym** liczba jest reprezentowana przez *wartość* i *pozycję* cyfry. O dziesiętnym systemie liczbowym mówimy, iż posiada *podstawę 10*, gdyż używa się w nim 10 cyfr, a *wagi* pozycji cyfr są równe potęgom 10. Przykładowo, liczba **503,14** w zapisie dziesiętnym oznacza:

$$(503,14)_{10} = 5 \times 10^2 + 0 \times 10^1 + 3 \times 10^0 + 1 \times 10^{-1} + 4 \times 10^{-2}$$

Innym systemem liczbowym jest **system binarny**. Cyfry w tym systemie przybierają dwie możliwe wartości **1** i **0**. System posiada *podstawę 2*, a *wagi* pozycji cyfr są równe potęgom 2. Przykładowo, liczba **11010,11** w zapisie binarnym oznacza:

$$(11010,11)_2 = 1 \times 2^4 + 1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 + 1 \times 2^{-1} + 1 \times 2^{-2} = (26,75)_{10}$$

Jak łatwo zauważyć, **binarny system** zapisu liczb odpowiada w sposób naturalny zmiennym występującym w **układach cyfrowych**. Systemy cyfrowe (np. komputery) wykonują wszelkie operacje logiczne i arytmetyczne wyłącznie na poziomie zapisu binarnego.

W technice cyfrowej, szczególnie przy pracy z komputerem (np. przy programowaniu) duże znaczenie praktyczne posiada jeszcze jeden system zapisu liczb, tzw. **system heksadecymalny** (o podstawie 16). Zaletą systemu heksadecymalnego jest czterokrotne skrócenie zapisu, z natury bardzo długich liczb binarnych. System heksadecymalny używa 16 cyfr: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F. Przykładowo, liczba **B65F** w zapisie heksadecymalnym oznacza:

$$(B65F)_{16} = 11 \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + 15 \times 16^0 = (46687)_{10}$$

Tablica 5.1 Systemy liczbowe

Decimal	Binary	Hexadecimal
0	0	0
1	1	1
2	10	2
3	11	3
4	100	4
5	101	5
6	110	6
7	111	7
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F
16	10000	10
17	10001	11
18	10010	12
19	10011	13
20	10100	14

Zestawienie pierwszych dwudziestu liczb zapisanych w omówionych systemach liczbowych przedstawiono w **Tablicy 5.1**.

Konwersja z systemu binarnego na heksadecymalny jest łatwa do przeprowadzenia, jeśli podzieli liczbę binarną na grupy liczące po cztery cyfry, zaczynając od cyfry o najmniejszej wadze, np:

$$\begin{array}{cccc}
 2 & C & 6 & B \\
 (. & 10 & 1100 & 0110 & 1011)_2 = (2C6B)_{16} = (11371)_{10}
 \end{array}$$

$$\begin{array}{cc}
 F & F \\
 (1111 & 1111)_2 = (FF)_{16} = (255)_{10}
 \end{array}$$

$$\begin{array}{c}
 F \\
 (1111)_2 = (F)_{16} = (15)_{10}
 \end{array}$$

Bit, Bajt, Słowo

Pojedyncza cyfra binarna jest nazywana **bitem** (ang. *bit*). Informacja w systemach cyfrowych ma charakter uporządkowany i jest reprezentowana przez sekwencję bitów. Sekwencja 8-bitowa jest nazywana **bajtem** (ang. *byte*). Liczba bitów w sekwencji **danych** (ang. *data*) przetwarzanych przez określony typ komputera jest nazywana **słowem** (ang. *word*). Komputery przetwarzają **dane** w **słowach** 8, 16, 32, 64-bitowych. Najbardziej znaczący (o największej wadze) bit w słowie określa się jako **MSB** (ang. *Most Significant Bit*), a najmniej znaczący bit - jako **LSB** (ang. *Least Significant Bit*). Przykładowe słowo 8-bitowe z zaznaczonymi wagami bitów przedstawiono poniżej.

2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
1	1	0	1	0	0	1	0
MSB				LSB			

Warto zauważyć, że za pomocą słowa 8-bitowego można wyrazić $2^8 = 256$ różnych informacji.

Kody binarne

Dla wygody komunikacji między światem cyfrowym (np. komputerem), który akceptuje wyłącznie liczby binarne, a człowiekiem, przyzwyczajonym raczej do obcowania z liczbami dziesiętnymi stosuje się rozmaite sposoby wzajemnej konwersji tych dwóch rodzajów zapisów.

Jednym z najprostszych sposobów jest **kodowanie** każdej cyfry dziesiętnej poprzez zastąpienie jej 4-bitową reprezentacją binarną. Ponieważ słowo 4-bitowe ma $2^4 = 16$ kombinacji zerojedynkowych, zatem spośród nich nie jest wykorzystywanych do kodowania (do zakodowania 10 cyfr od 0 do 9 wystarczy 10 kombinacji). Stwarza to możliwość **binarnego kodowania dziesięciu cyfr systemu dziesiętnego** na wiele różnych sposobów. Liczba możliwych **kodów BCD** (ang. *Binary-Coded Decimals*), wynikająca z zastosowania 4 bitów jest olbrzymia. Jednak praktyczne zastosowanie znalazło tylko kilka, spośród których jednym z najbardziej popularnych jest **kod 8421** zwany powszechnie **kodem BCD** (patrz **Tablica 5.2**).

Zwróćmy uwagę, że lewa kolumna tablicy zawiera **cyfry** dziesiętne a **nie liczby**! Poniżej zamieszczono przykłady **liczb** dziesiętnych w kodzie **BCD**:

Tablica 5.2 Kod BCD

Dec. digit	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

$$(6)_{10} = (0110)_{\text{BCD}}$$

$$(13)_{10} = (0001\ 0011)_{\text{BCD}}$$

$$(369)_{10} = (0011\ 0110\ 1001)_{\text{BCD}}$$

Pamiętajmy, że kod BCD koduje inaczej liczby dziesiętne niż tzw. *naturalny kod binarny* (przedstawiony w Tablicy 5.1). Tylko liczby z zakresu od 0 do 9 mają ten sam zapis. Poniżej przedstawiono te same liczby dziesiętne, jak w powyższych przykładach, zapisane w *naturalnym kodzie binarnym*.

$$(6)_{10} = (0110)_2$$

$$(13)_{10} = (1101)_2$$

$$(369)_{10} = (101110001)_2$$

Innymi kodami są **kody alfanumeryczne** umożliwiające komunikację człowieka z systemem cyfrowym za pośrednictwem klawiatury. Standardowym kodem alfanumerycznym jest **kod ASCII** (ang. *American Standard Code for Information Interchange*). Pełny zbiór zawiera 128 znaków kodowanych symbolami 7-bitowymi (zauważ, że $2^7 = 128$). Przykładowo, kod **111 1111** oznacza klawisz **Del** (ang. *delete* - kasuj).

6. Przegląd technologii układów cyfrowych scalonych

Pod względem konstrukcyjno-technologicznym wszystkie układy cyfrowe scalone można podzielić ogólnie na:

- **bipolarne**, w których podstawowymi elementami są tranzystory bipolarne,
- **unipolarne**, nazywane również **układami MOS**, w których podstawowymi elementami są tranzystory MOS.

Spośród wielu dostępnych *klas (technik) układowych* najważniejsze znaczenie mają:

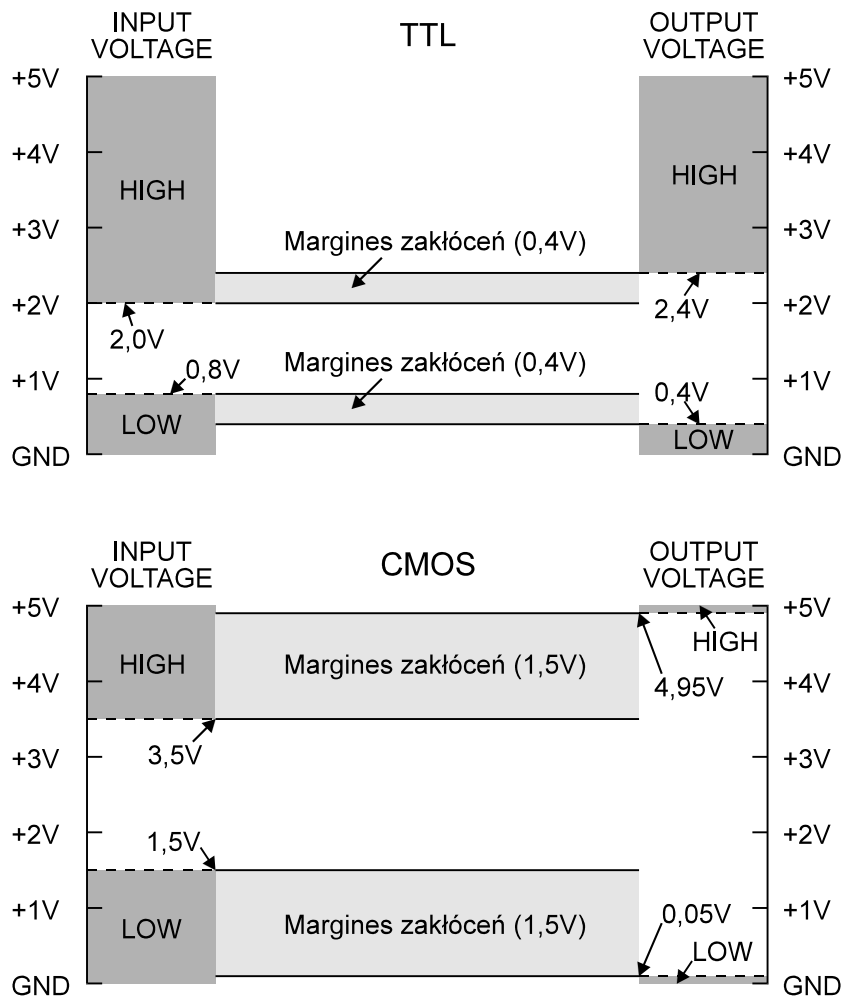
- **układy TTL** (ang. *Transistor-Transistor Logic*), które są **układami bipolarnymi**,
- **układy CMOS** (ang. *Complementary Metal Oxide Semiconductor*), które są **układami unipolarnymi** - z tranzystorami p-MOS oraz n-MOS.

Podstawowe parametry techniczne układów cyfrowych

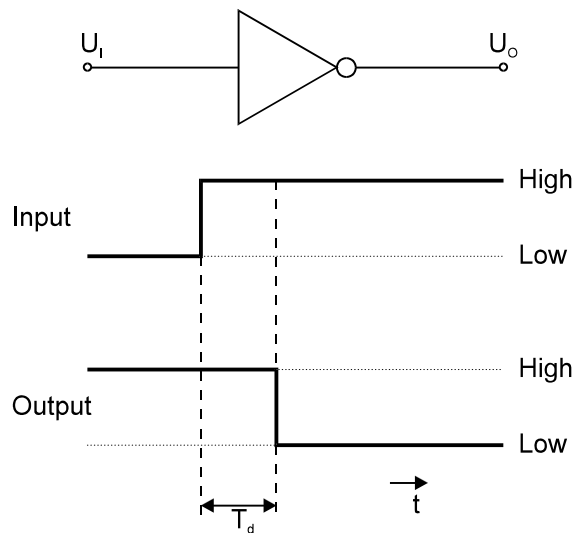
★ **Obciążalność logiczna bramki (N)** - maksymalna liczba bramek, jaka może być równolegle sterowana z wyjścia pojedynczej bramki.

★ **Napięcia poziomów logicznych (HIGH, LOW)** - zakresy napięć wejściowych oraz wyjściowych, które układ realizuje jako **gwarantowany stan 1** oraz **gwarantowany stan 0**.

★ **Marginesy zakłóceń (U_{NM})** - określa dopuszczalną wartość napięcia sygnału zakłócającego, nie powodując jeszcze **nieprawidłowej** pracy układu. Interpretację poziomów logicznych i marginesów zakłóceń podano na **rys. 6.1**.



Rys. 6.1 Napięcia poziomów logicznych HIGH, LOW i marginesy zakłóceń dla układów TTL i CMOS
 ★ **Czas propagacji (T_d)** - określa czas **opóźnienia** odpowiedzi układu na sygnał sterujący i jest podstawowym miarą szybkości działania układu cyfrowego.

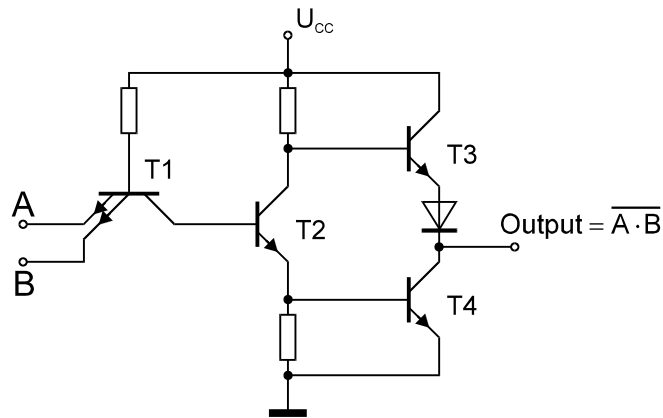


Rys. 6.2 Czas propagacji dla układu inwertera

★ **Moc strat na bramk (P_d)** - określa moc **pobieraną** przez układ z źródła zasilania.

Układ TTL

Podstawowy, dwuwęzłowy bramek NAND przedstawiono na rys. 6.3.



Rys. 6.3 Bramka NAND (TTL). Tranzystor T1 jest wieloemiterowym tranzystorem npn.

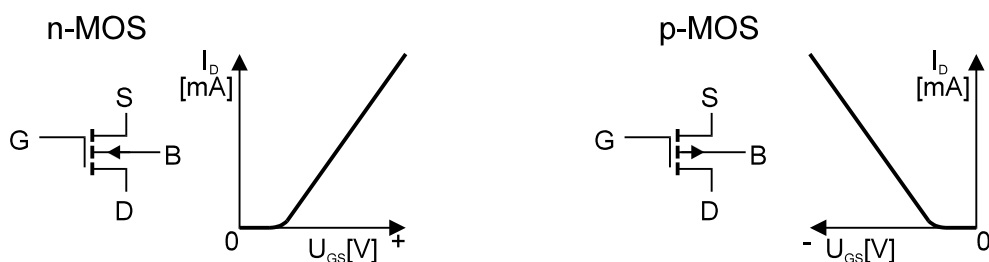
Przyłączenie jakiegokolwiek wejścia A, B lub obydwu do masy (co oznacza stan wejścia równy 0) powoduje **wyłączenie** tranzystorów T2 i T4, gdy napięcie na **bazie** tranzystora T1 nie przekracza wartości $0,3V + 0,7V = 1V$ i jest niewystarczające do spolaryzowania złącza **baza-kolektor** tranzystora T1, złącza **baza-emiter** tranzystora T2 oraz złącza **baza-emiter** tranzystora T4 w stan przewodzenia (musiałoby być równe co najmniej $0,7V + 0,7V + 0,7V = 2,1V$). Gdy T2 jest **wyłączony**, wtedy T3 jest **włączony** i napięcie na wyjściu układu osiąga wartość $5V - 0,3V - 0,7V = 4V$, co oznacza stan wyjścia układu równy 1.

UWAGA: Napięcie **0,7V** jest napięciem na przewodzącym złączu p-n, a napięcie **0,3V** jest napięciem $U_{CE(sat)}$, a więc napięciem pomiędzy kolektorem i emiterym nasyconego tranzystora.

Gdy oba wejścia A, B są na poziomie 1, wtedy złącza **baza-emiter** tranzystora T1 są spolaryzowane zaporowo - co powoduje, że napięcie na bazie tranzystora T1 może osiągnąć wartość 2,1V i jest wystarczające do wprowadzenia tranzystorów T2 i T4 w stan przewodzenia. Napięcie na kolektorze tranzystora T2 (o wartości 1,4V) jest niewystarczające do spolaryzowania tranzystora T3 w stan przewodzenia (ze względu na obecność diody) i tranzystor T3 jest **wyłączony**. Napięcie na wyjściu układu wynosi 0,3V i jest równe napięciu $U_{CE(sat)}$ tranzystora T4 - co oznacza stan 0 na wyjściu układu.

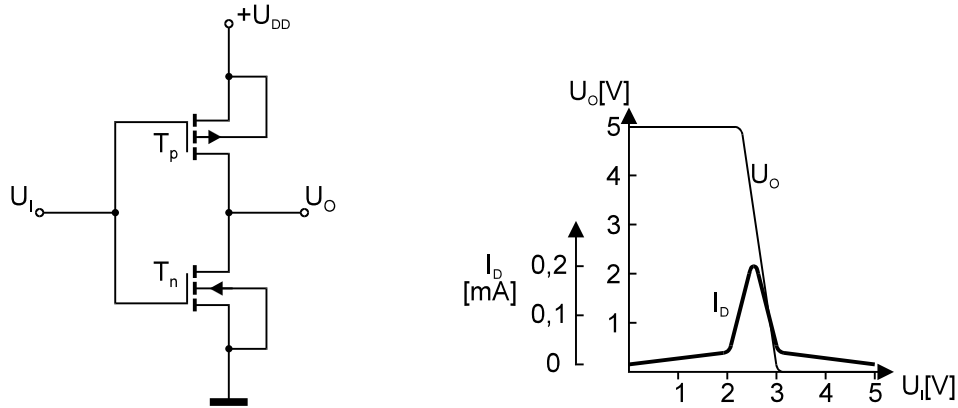
Układ CMOS

Układy CMOS zawierają komplementarne pary **wzbogacanych** ("normalnie wyłączonych") tranzystorów n-MOS oraz p-MOS. Symbole graficzne oraz charakterystyki tych tranzystorów pokazano na rys. 6.4.



Rys. 6.4 Charakterystyki wzboganych tranzystorów MOS

Zauważ, że przy napięciu $U_{GS} = 0$ dany tranzystor nie przewodzi. Podstawowym układem CMOS jest układ **inwertera** przedstawiony na rys. 6.5.



Rys. 6.5 Inwerter CMOS i jego charakterystyka przejściowa

Zauważ, że dodatni biegun napięcia zasilającego $+U_{DD}$ jest przyłączony do **ródła S** tranzystora p-MOS (oznaczonego symbolem T_p), podczas gdy **ródło S** tranzystora n-MOS (oznaczonego symbolem T_n) jest na masie. Końcówki **drenów D** obu tranzystorów są połączone razem i tworzą wyjście układu.

Gdy napięcie wejściowe $U_i = U_{GS}$ jest równe **0**, wtedy tranzystor T_n jest **wyłączony**. Dla tranzystora T_p napięcie $U_{GS} = U_i - U_{DD} \cong -U_{DD}$, a zatem T_p jest **włączony**. Napięcie wyjściowe jest równe napięciu U_{DD} , co oznacza stan **1**.

Gdy napięcie wejściowe jest dostatecznie duże (dodatnie), co oznacza stan **1**, wtedy tranzystor T_n jest **włączony**, a tranzystor T_p jest **wyłączony**. Napięcie wyjściowe jest praktycznie równe **0V**, co oznacza stan **0**.

Podstawowym zaletą układów CMOS jest **znikomy pobór mocy**, który występuje wyłącznie podczas przełączania. Ilustruje to rys. 6.5.

Porównanie układów TTL i układów CMOS

Wśród układów TTL i układów CMOS występuje wiele "rodzin" układowych charakteryzujących się różnymi parametrami technicznymi, z których najważniejsze to czas propagacji, moc strat na bramki oraz napięcie zasilania, które ma podstawowe znaczenie przy porównaniu układów CMOS z układami TTL. Porównanie układów CMOS i TTL ilustruje **tablica 6.1**.

Tablica 6.1 Charakterystyczne parametry "rodzin" CMOS i TTL

	T_d [ns]	P_d [mW/Bramk] przy 1MHz	$U_{CC/DD}$ [V]	
			Min	Max
CMOS				
4000B	50	0,5	3	18
74HC	15	0,5	2	6
74HCT	15	0,5	4,5	5,5
74AC	5	0,1	2	6
74ACT	5	0,1	4,5	5,5
TTL				
74LS	10	2	4,75	5,25
74S	3	20	4,75	5,25

74ALS	4	1	4,5	5,5
74AS	1,5	7	4,5	5,5

OBJA NIENIA:

- HC - *High-speed CMOS logic,*
- HCT - *High-speed CMOS logic (z poziomami logicznymi TTL),*
- AC - *Advanced CMOS logic,*
- ACT - *Advanced CMOS logic (z poziomami logicznymi TTL),*
- LS - *Low-power Schottky TTL logic,*
- S - *Schottky TTL logic,*
- ALS - *Advanced Low-power Schottky TTL logic,*
- AS - *Advanced Schottky TTL logic.*

UWAGI:

★CMOS-owe rodziny 74HCT oraz 74ACT mają te same **napięcia zasilania i poziomy logiczne** jak rodziny TTL. Wszystkie rodziny oznaczone **74.....** mają te same **kody numerowe** określające **funkcje** danego układu oraz identyczny **rozkład wyprowadzeń**. Umożliwia to **łączenie** układów CMOS i TTL **między sobą** (są one układami **kompatybilne**).

★Gdy wymagany jest **niski pobór mocy i rozszerzony zakres napięć logicznych**, wtedy stosujemy układy CMOS rodziny 4000B. Są one szczególnie **niekorzystnie** wykorzystywane przy stosowaniu **zasilania baterijnego**.

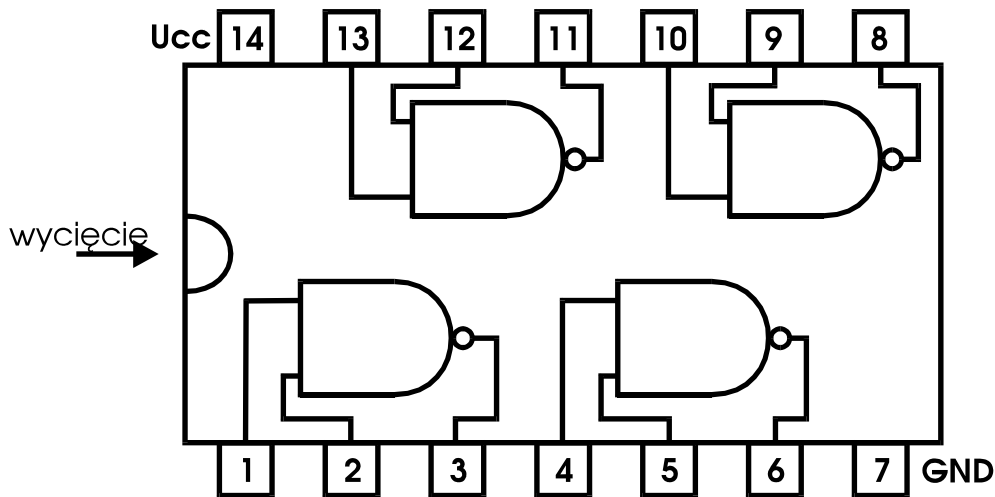
★Największe **gęstości upakowania** elementów w chip-ach osiągnięte w układach CMOS, szczególnie w układach skali **LSI** oraz **VLSI** są układami CMOS.

★Układy CMOS charakteryzują się **niskim poborem mocy**, ale są stosunkowo **wolniejsze** od układów TTL, chociaż - układy rodziny 74ACT ustępują w szybkości tylko najszybszym układom TTL tej rodziny 74AS.

★Układy CMOS charakteryzują się znacznie **większym marginesem bezpieczeństwa** niż układy TTL.

Schemat wyprowadzeń układu cyfrowego stosowany przez producentów

Na **rys. 6.6** przedstawiono przykład pokazujący schemat wyprowadzeń układu UCY 7400, który zawiera **cztery dwuwęzłowe bramki NAND**. Wyciągnięte w obudowie jednoznacznie określają kolejność wyprowadzeń.

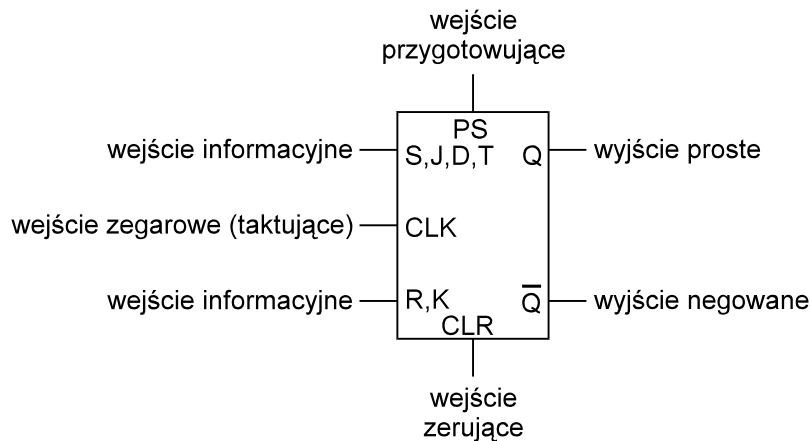


Rys. 6.6 Widok z góry wyprowadze układu UCY 7400

7. Omówienie układów cyfrowych zastosowanych w wiczeniu

7.1 Przerzutniki

Przerzutnikiem (ang. *Flip-Flop*) jest nazywany układ, charakteryzujący się istnieniem dwóch stanów równowagi trwałej (**1** albo **0**), przy czym dla przejścia z jednego stanu do drugiego jest konieczne doprowadzenie sygnału zewnętrznego (informacji). Ponieważ przerzutnik pamięta jeden bit informacji może być traktowany jako **jednobitowa komórka pamięci**. Ogólny symbol graficzny przerzutnika podano na rys. 7.1.



Rys. 7.1 Ogólny symbol graficzny przerzutnika

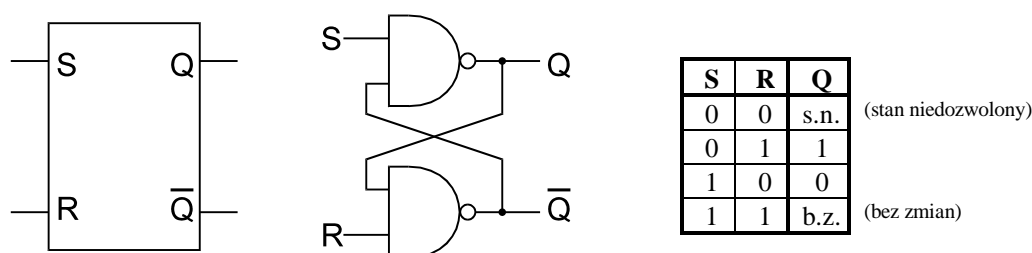
Przerzutnik ma pewną liczbę wejść i z reguły dwa wyjścia. Stan logiczny wyjścia **Q** uważa się za **stan przerzutnika**. Nazwy literowe wejść informacyjnych (**SR**, **JK**, **D**, **T**) określają jednocześnie nazwy rodzajów przerzutników (*przerzutnik SR*, *przerzutnik JK*, *przerzutnik D*, *przerzutnik T*). Wejście zegarowe **CLK** (ang. *Clock*) służy do podawania sygnałów taktujących, które narzucają synchroniczny tryb pracy układu. Wejście przygotowujące **PS** (ang. *Preset*) oraz wejście zerujące **CLR** (ang. *Clear*) służy do ustalenia stanu przerzutnika niezależnie od stanu wejść informacyjnych oraz stanu wejścia zegarowego.

Przerzutniki mogą być asynchroniczne i synchroniczne. **Przerzutniki asynchroniczne** pracują bez sygnału zegarowego, a stan przerzutnika ustala się bezpośrednio w wyniku zmiany stanu wejścia.

Przerzutniki synchroniczne pracują z udziałem sygnału zegarowego, a stan wejściowy jest przekazywany na wyjście w chwilach wystąpienia narastającego lub opadającego zbicia sygnału zegarowego (zazwyczaj wykorzystuje się zbicie opadające).

Przerzutnik SR (asynchroniczny)

Przerzutnik SR (ang. *Set*, *Reset*) jest najprostszym układem przerzutnika bistabilnego. Symbol graficzny, realizacja NAND-owa oraz tablica prawdy przerzutnika SR podano na **rys. 7.2**.

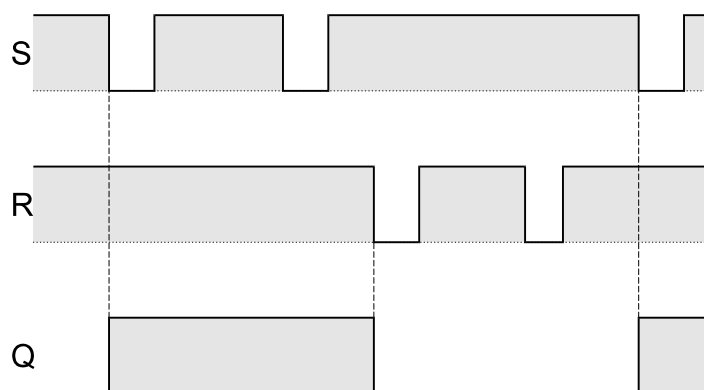


Rys. 7.2 Przerzutnik SR (symbol graficzny, realizacja NAND-owa, tablica prawdy)

Normalnym **stanem spoczynkowym** (stanem pamiętania) przerzutnika jest stan obu wejść równy **1** ($S=1$, $R=1$) podczas którego stan przerzutnika nie zmienia się (przerzutnik pamięta swój stan poprzedni).

- Podanie $S=0$ powoduje przejście Q do stanu **1**.
- Podanie $R=0$ powoduje powrót Q do stanu **0**.
- Gdy oba wejścia są równe 0 ($S=0$, $R=0$) to oba wyjścia przechodzą do stanu **1**, czego unika się w normalnej pracy przerzutnika (*stan niedozwolony!*).

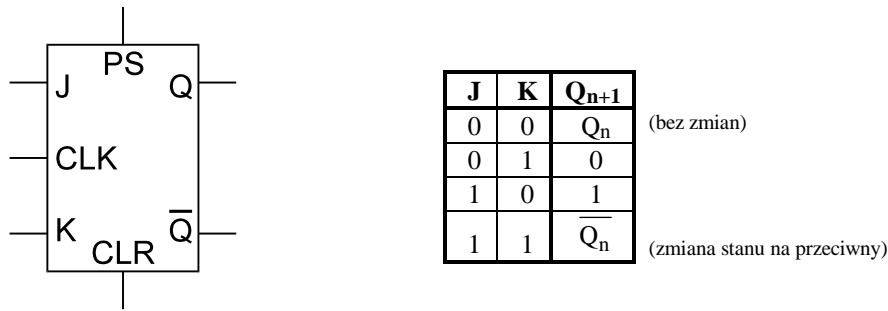
Wykres czasowy pracy przerzutnika pokazuje **rys. 7.3**.



Rys. 7.3 Wykres czasowy przerzutnika SR

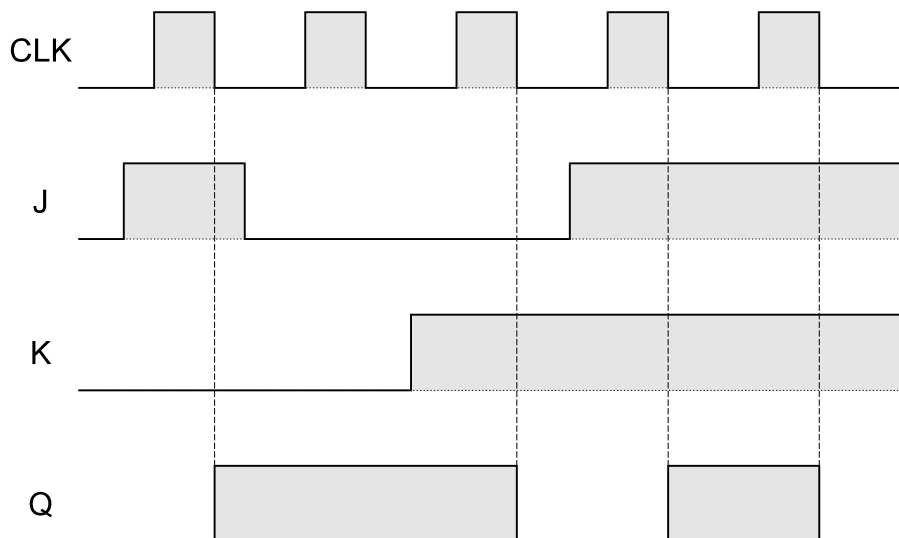
Przerzutnik JK (synchroniczny)

Przerzutnik JK jest najbardziej rozpowszechnionym układem przerzutnikowym techniki cyfrowej. Jest on traktowany jako **układ uniwersalny**. Przez zastosowanie odpowiednich połączeń zewnętrznych można z niego utworzyć inny rodzaj przerzutnika, np. **SR**(syn.), **D**, **T**. Symbol graficzny oraz tablica prawdy przerzutnika JK przedstawiono na **rys. 7.4**.



Rys. 7.4 Przerzutnik JK (symbol graficzny, tablica prawdy)

Przerzutnik JK *nie ma stanów niedozwolonych!* **Zmiana stanu** przerzutnika JK następuje w chwilach wyznaczonych przez opadające zbocze **impulsu zegarowego**. Stan logiczny wyjścia **Q** w umownym czasie t_{n+1} (t.j. po przyjęciu impulsu zegarowego) zależy od stanów **J, K, Q** w czasie t_n (t.j. przed przyjęciem impulsu zegarowego). W przedziale czasu między impulsami zegarowymi przerzutnik **nie zmienia stanu** nawet, gdy zmieniają się stany wejść **J, K**. Wykres czasowy pracy przerzutnika JK przedstawia rys. 7.5.



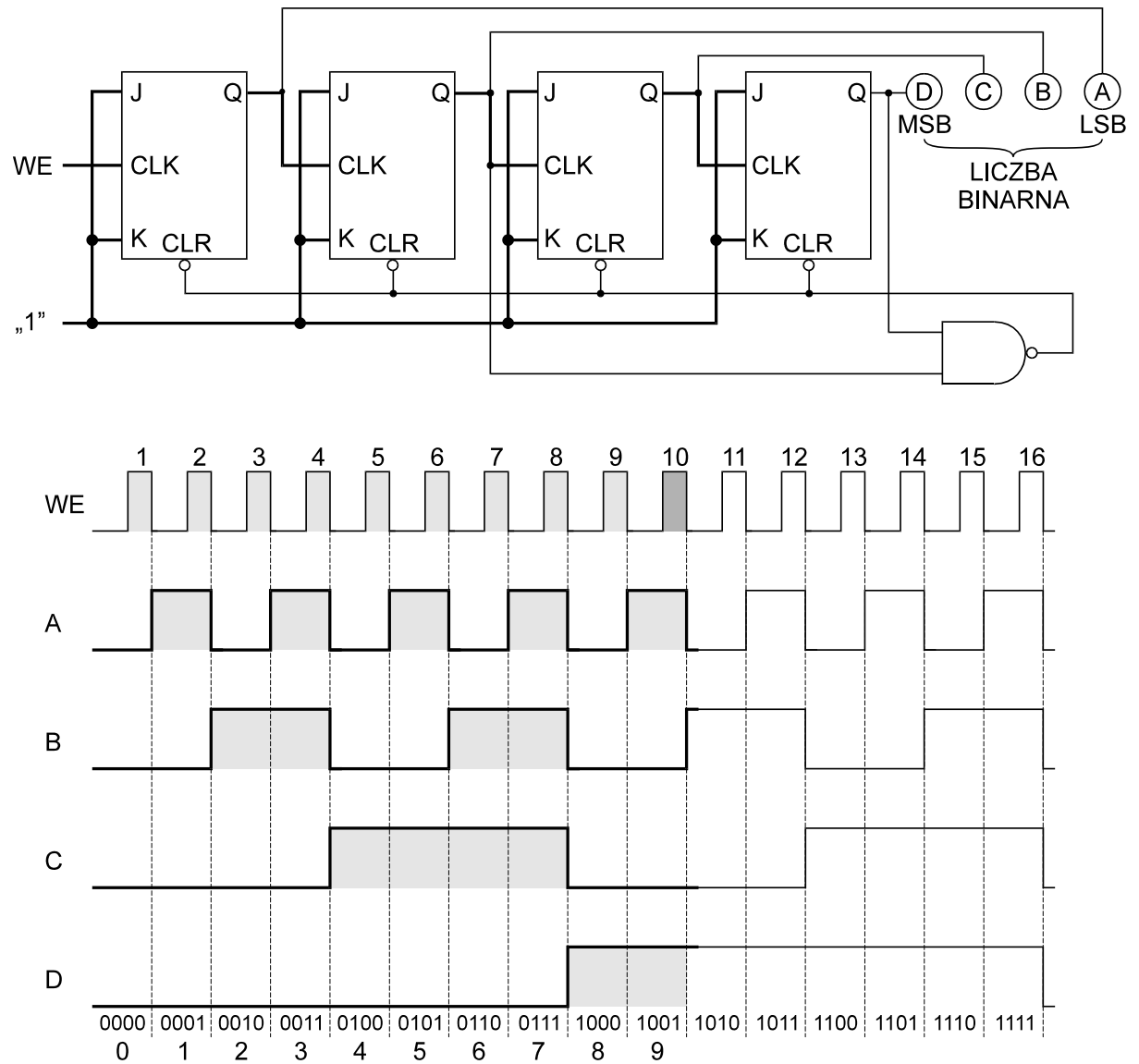
Rys. 7.5 Wykres czasowy przerzutnika JK

UWAGA: Warto zauważyć, że gdy **oba wejścia J, K** mają stan **1**, to przerzutnik JK zmienia swój stan na **przeciwny** za każdym razem, gdy tylko pojawi się impuls zegarowy. Właściwość wykorzystuje się do budowy **liczników i dzielników częstotliwości**. Przerzutniki JK pracujące z jednym sygnałem informacyjnym doprowadzonym do połączonych ze sobą wejść **J, K** nazywają się **przerzutnikami T**.

7.2 Licznik i dzielnik częstotliwości

Licznik jest układem cyfrowym służącym do zliczania i zapamiętania liczby impulsów. Liczniki buduje się z przerzutników JK pracujących w układzie przerzutnika T. Gdy licznik składa się z **n** przerzutników to tzw. **pojemność licznika (P)**, czyli maksymalna liczba impulsów, które licznik jest w stanie zliczyć, wynosi $P=2^n$. Liczbę **n** nazywamy **decyfry licznika**. Zapamiętanie licznika kończy cykl pracy licznika, po czym wraca on do stanu początkowego. **Decyfry cyklu licznika (S)** nazywamy liczbą wyróżnialnych stanów logicznych, przez które licznik przechodzi cyklicznie. Jeżeli $S \leq 2^n$ to mówimy, że jest to **licznik modulo-S** (np. licznik **modulo-10** jest licznikiem dziesiętnym, tzw. dekad

licząc, przy czym $n=4$ oraz $P=2^4=16$). Schemat i zasady działania 4-bitowego licznika asynchronicznego ze skróconym cyklem pracy (licznik modulo-10) ilustruje rys. 7.6.



Rys. 7.6 Licznik modulo-10

Jak łatwo zauważyć, odpowiednio połączona bramka NAND (na wejściach ma dwie jedynki z liczby 1010) służy do zerowania licznika po każdym cyklu liczenia od 0 do 9.

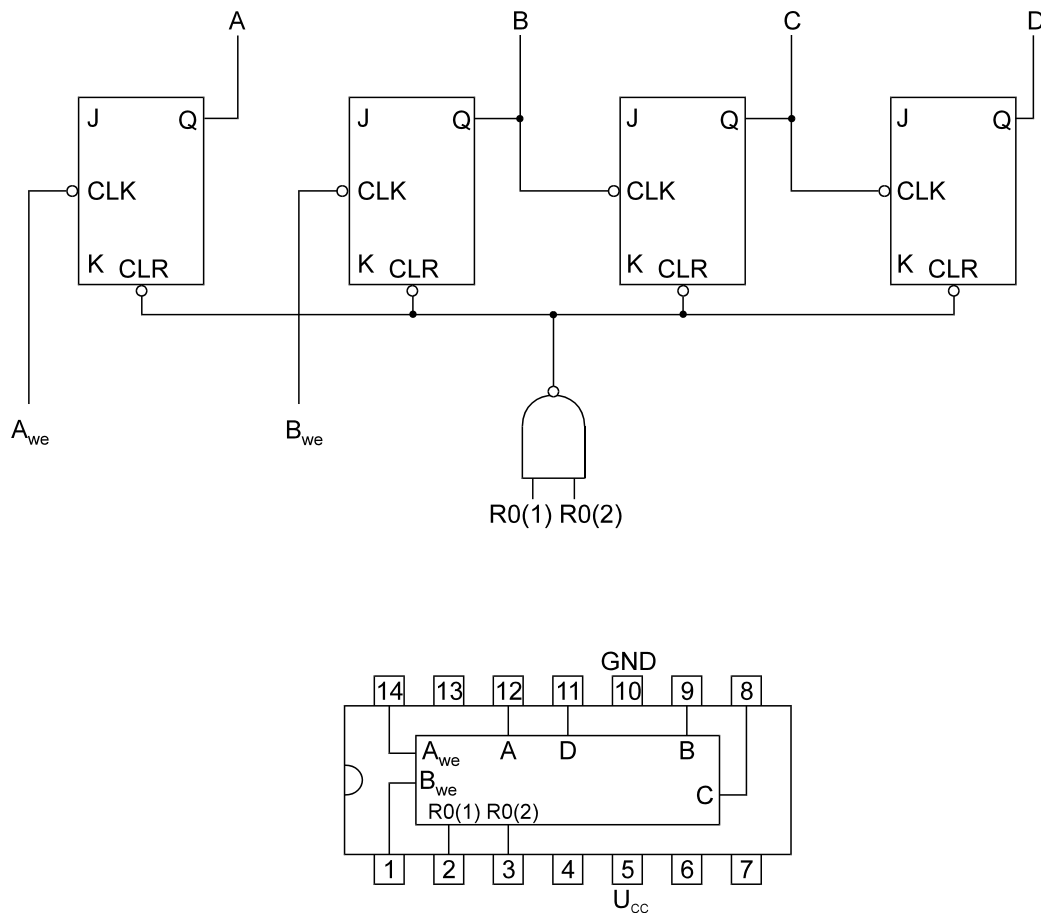
Dzielnikiem częstotliwości nazywamy układ, który daje wyjściowy impuls co N impulsów wejściowych (N jest liczbą całkowitą). Wtedy mamy:

$$f_{wy} = \frac{f_{we}}{N}$$

Zauważmy, że licznik z rys.7.6 może być traktowany jako **dzielnik częstotliwości**, gdy jego wyjścia A, B, C, D dzielą częstotliwość impulsów wejściowych odpowiednio w stosunku 1:2, 1:4, 1:8, 1:16.

Licznik UCY 7493

Licznik UCY 7493 jest scalonym asynchronicznym licznikiem składającym się z przerzutników JK połączonych w ten sposób, że tworzy liczniki zliczające **modulo-2** i **modulo-8**. Wyjście A nie jest połączone wewnętrznie z wejściem następnego przerzutnika. Licznik ma dwa bramkowane wejścia zerujące **R0(1)** i **R0(2)**. Schemat logiczny oraz widok wyprowadzeń układu UCY 7493 przedstawiono na rys. 7.7.



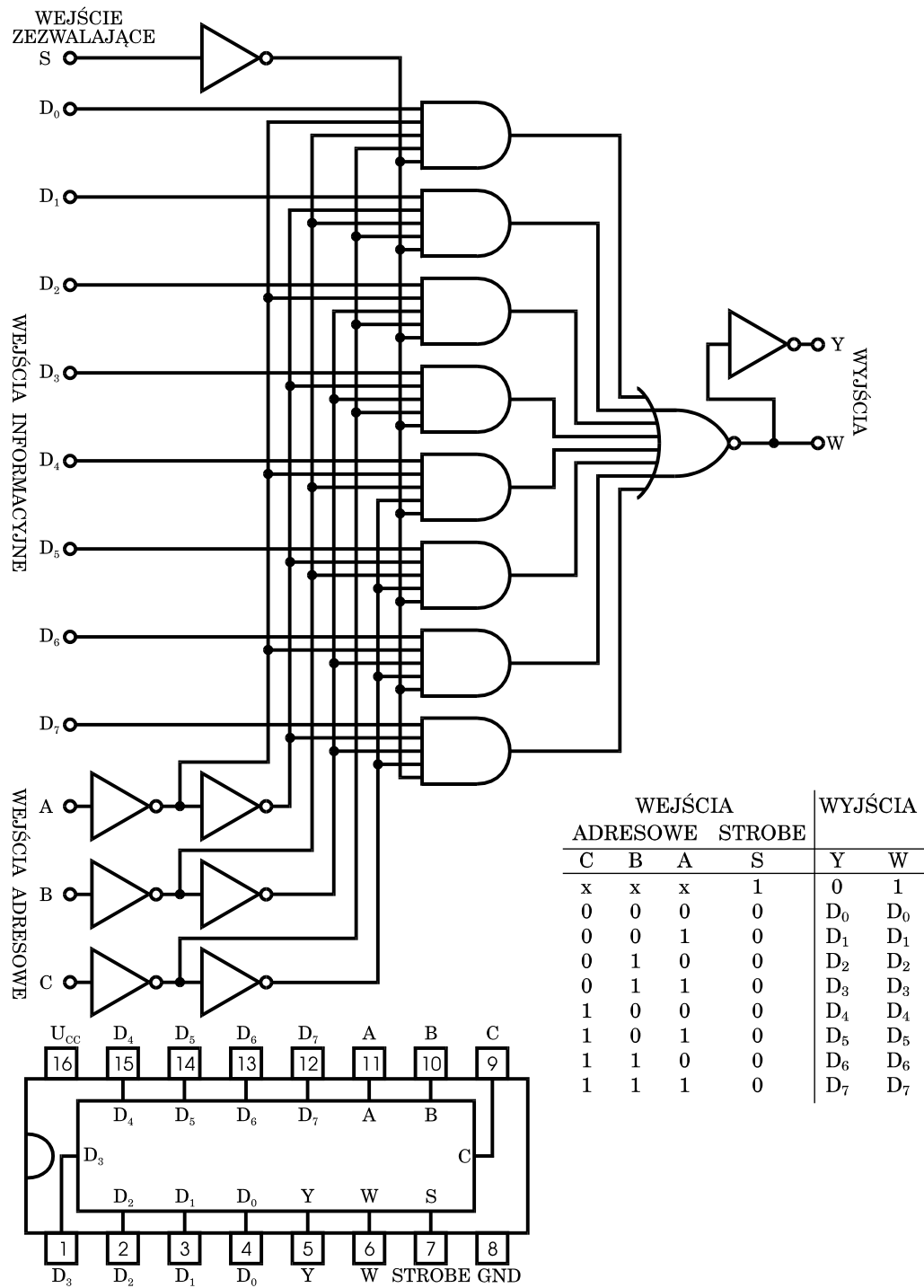
Rys. 7.7 Schemat logiczny i widok wyprowadzeń układu UCY 7493

7.3 Multiplexer

W systemach cyfrowych często istnieje potrzeba przesyłania **selektywnie wybranej** informacji binarnej, a to zmiany sposobu przesyłania z **równoległego (N bitów - N linii)** na **szeregowy (N bitów - kolejno w jednej linii)**. Realizacją tego zadania umożliwia multiplexer. Multiplexer cyfrowy przekazuje sygnały cyfrowe z **jednego z N wejść** wybranego adresem na **pojedyncze wyjście** (tj. linię przesyłową).

Multiplekser UCY 74151

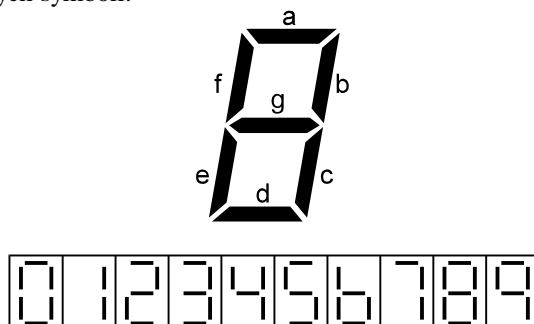
Na rys. 7.8 przedstawiono schemat logiczny, tablicę prawdy oraz widok wyprowadze multipleksera UCY 74151. Układ ma 8 wejść informacyjnych (D_0 do D_7), 3 wejścia adresowe (A , B , C), 1 wejście zezwalające - ang. *Strobe* (S) oraz wyjście proste (Y) i wyjście negowane (W). Na wyjściu Y pojawi się stan 1 wówczas, gdy wejście informacyjne, wybrane odpowiednią kombinacją stanów na wejściach adresowych, jest w stanie 1, a wejście zezwalające jest w stanie 0, czyli zezwolenia.



Rys. 7.8 Schemat logiczny i widok wyprowadze układu UCY 74151

7.4 Wskaźniki półprzewodnikowe i układy sterujące nimi

Powszechnie stosuje się wskaźniki z **diodek elektroluminescencyjnych** i **ciekrych krystalicznych**. Znajdują one zastosowanie przede wszystkim jako **wskaźniki alfanumeryczne**. Najczęściej mają budowę **segmentową** lub **mozaikową**. Wskaźnik 7-segmentowy, pokazany na **rys. 7.9**, umożliwia przedstawienie 10 cyfr i 10 liter. Wskaźniki te są wytwarzane jako jedno- lub wielopozycyjne (w jednej obudowie), przy tym często zawierają dodatkowe elementy, służące do wyświetlania kropki, dwukropka, a także innych symboli.



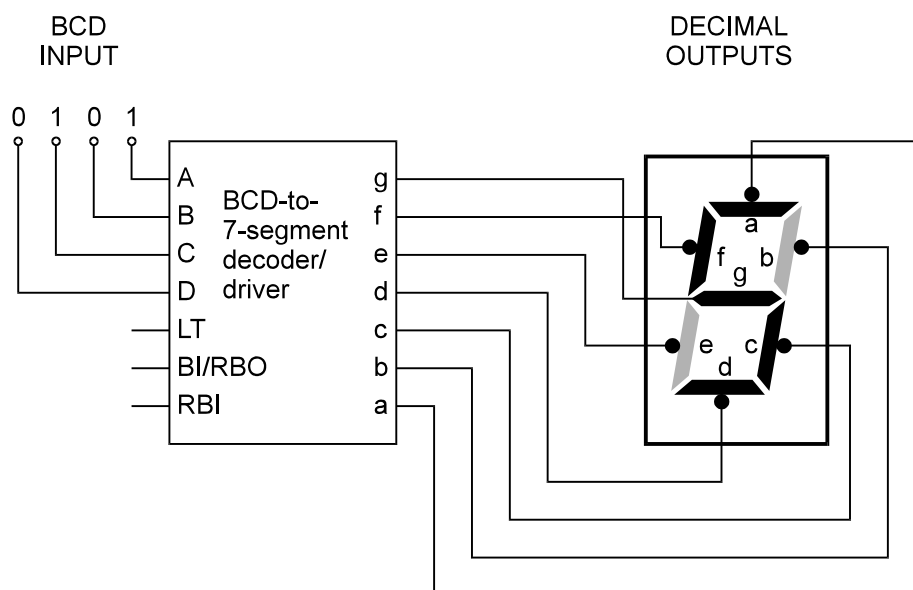
Rys. 7.9 Wskaźnik 7-segmentowy

Najważniejszymi **zaletami wskaźników z diodek elektroluminescencyjnych** są ich **duża niezawodność** i **trwałość** (szacowana na 114 lat), **niskie napięcie zasilania** (ok. 2V) oraz **możliwość uzyskania różnych barw świecenia**. Zasadniczą wadą jest **duży pobór prądu** (kilka do kilkudziesięciu mA na segment).

Wskaźniki ciekrych krystalicznych mają **najniższy pobór mocy** ze wszystkich rodzajów wskaźników, lecz są **niestety także w różnicach najwolniejsze w działaniu**. Ponadto wymagają **świetlenia tym silniejszego, im bardziej kontrastowy obraz znaku pragnie się uzyskać**.

Dekoder UCY 7447

Do sterowania wskaźnikami 7-segmentowymi używa się specjalnych **układów cyfrowych**, tzw. **dekoderów sterujących** (ang. *decoder/driver*). Dekoder sterujący przetwarza kod BCD(8421) na kod wskaźnika 7-segmentowego, powodując świecenie odpowiednich segmentów wskaźnika. Działanie, w połączeniu ze wskaźnikiem 7-segmentowym, **dekodera UCY 7447** ilustruje **rys. 7.10**.



Dodatkowe wejścia sterujące **LT**, **BI/RBO**, **RBI** mają następujące funkcje:

- LT** - gdy jest w stanie **0** - rozświetla wszystkie segmenty,
- BI/RBO** - gdy jest w stanie **0** - wygasza wszystkie segmenty,
- RBI** - gdy jest w stanie **0** - wygasza wszystkie segmenty wyjątkowo gdy pokazują cyfrę 0.

II. Wykonanie wiczenia

Cz A Projektowanie i symulacja komputerowa układów logicznych

Zaprojektować schematy logiczne poniżej przedstawionych funkcji albo zagadnień logicznych. Schematy logiczne powinny być zrealizowane w dwóch wersjach:

- a) przy użyciu bramek **AND**, **OR**, **NOT**,
- b) przy użyciu wyłącznie bramek **NAND**.

Przy projektowaniu należy stosować się do reguł przedstawionych w części teoretycznej instrukcji (patrz p. 4.2, 4.3, 4.4) określając w kolejnych krokach projektu: tablice prawdy, równanie boolowskie, zminimalizowane równanie boolowskie, realizacje układowe w wersji a) i w wersji b). Następnie, należy dokonać symulacji komputerowej zaprojektowanych schematów logicznych i wykonać sprawdzenie poprawności ich działania (wykorzystując mechanizmy programu) poprzez zadanie wybranych kombinacji stanów wejściowych. Stan wyjściowy układu porównać z tabelicą prawdy funkcji logicznej.

ZADANIE 1. Zaprojektować schematy logiczne funkcji XOR oraz XNOR (tablice prawdy tych funkcji zawarto w tabelicy 3.1).

ZADANIE 2. Zaprojektować schemat logiczny funkcji realizującej 3-wejciową operację NAND

ZADANIE 3. Dokonać minimalizacji funkcji logicznych, a następnie zaprojektować schematy logiczne dla minimalnych postaci tych funkcji.

- a) $Y = ABCD + A\overline{BCD}$
- b) $Y = AB + \overline{AC} + A\overline{BC}(AB + C)$
- c) $Y = A + B(\overline{C + DE})$

ZADANIE 4. Zaprojektować system logiczny pozwalający na start silnika samochodowego gdy **kluczyk** jest w pozycji ON i **pas kierowcy** jest zapięty i **pas pasera** obok kierowcy jest zapięty lub **czujnik ciśnienia** nie wykrył pasera.

ZADANIE 5. Zaprojektować system logiczny uruchamiający alarm zainstalowany w domu, gdy czujnik alarmu jest w pozycji ON i drzwi frontowe lub drzwi tylne lub okno są otwarte lub czujnik wykrył ruch.

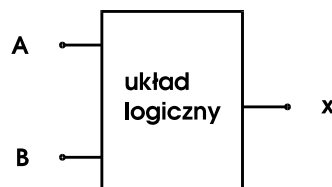
ZADANIE 6. Dokonać symulacji komputerowej przerzutnika RS w realizacji NAND-owej i sprawdzić jego działanie poprzez zadawanie odpowiednich stanów wejściowych i badanie stanu wyjściowego.

UWAGA: W sprawozdaniu należy zamieścić wszystkie kroki cyklu projektowania schematów logicznych wybranych zadań logicznych oraz wydruki komputerowe zrealizowanych symulacji schematów logicznych.

Cz. B Badania laboratoryjne wybranych układów cyfrowych

1. Określenie typu bramki na podstawie tablicy prawdy

Dla każdej z czterech nieznanymi bramek (wybieranych kolejno przyciskami 1, 2, 3, 4) należy określić **tablicę prawdy** oraz dokonać **identyfikacji** bramki. Stany wyjściowe (X) każdej bramki należy określić za pomocą próbnika stanów logicznych dla wszystkich kombinacji stanów wejściowych. Sygnały wejściowe 0 lub 1 należy zadawać na wejścia A, B za pomocą przycisków oznaczonych A, B. Wyniki pomiarów zamieścić w tablicach prawdy. Na podstawie tablicy prawdy określić rodzaj bramki i narysować jej symbol graficzny.



zadawanie sygnali		wybór układu logicznego			
1	0	1	2	3	4
A	B				

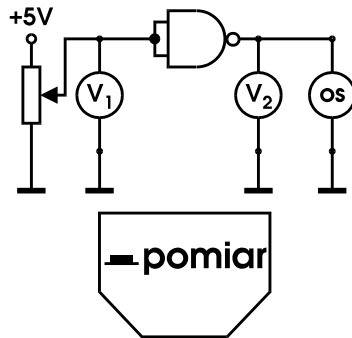
1	2	3	4																																																												
<table border="1" style="width: 100%; text-align: center;"> <thead> <tr><th>A</th><th>B</th><th>X</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td></td></tr> <tr><td>0</td><td>1</td><td></td></tr> <tr><td>1</td><td>0</td><td></td></tr> <tr><td>1</td><td>1</td><td></td></tr> </tbody> </table>	A	B	X	0	0		0	1		1	0		1	1		<table border="1" style="width: 100%; text-align: center;"> <thead> <tr><th>A</th><th>B</th><th>X</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td></td></tr> <tr><td>0</td><td>1</td><td></td></tr> <tr><td>1</td><td>0</td><td></td></tr> <tr><td>1</td><td>1</td><td></td></tr> </tbody> </table>	A	B	X	0	0		0	1		1	0		1	1		<table border="1" style="width: 100%; text-align: center;"> <thead> <tr><th>A</th><th>B</th><th>X</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td></td></tr> <tr><td>0</td><td>1</td><td></td></tr> <tr><td>1</td><td>0</td><td></td></tr> <tr><td>1</td><td>1</td><td></td></tr> </tbody> </table>	A	B	X	0	0		0	1		1	0		1	1		<table border="1" style="width: 100%; text-align: center;"> <thead> <tr><th>A</th><th>B</th><th>X</th></tr> </thead> <tbody> <tr><td>0</td><td>0</td><td></td></tr> <tr><td>0</td><td>1</td><td></td></tr> <tr><td>1</td><td>0</td><td></td></tr> <tr><td>1</td><td>1</td><td></td></tr> </tbody> </table>	A	B	X	0	0		0	1		1	0		1	1	
A	B	X																																																													
0	0																																																														
0	1																																																														
1	0																																																														
1	1																																																														
A	B	X																																																													
0	0																																																														
0	1																																																														
1	0																																																														
1	1																																																														
A	B	X																																																													
0	0																																																														
0	1																																																														
1	0																																																														
1	1																																																														
A	B	X																																																													
0	0																																																														
0	1																																																														
1	0																																																														
1	1																																																														

2. Pomiar charakterystyki przejściowej bramki NAND z wykorzystaniem układu UCY 7400

Należy wyznaczyć **charakterystyk przejściową** $U_o = f(U_i)$ bramki. Wcisnąć przycisk POMIAR. Napięcie wejściowe U_i regulować, za pomocą potencjometru, w zakresie od 0V do 5V. Napięcie wyjściowe U_o pomierzyć za pomocą oscyloskopu lub miernika o dużej oporności wejściowej. Wyniki pomiarów zamieścić w tabeli.

Na podstawie wyników pomiarów narysować **charakterystyk przejściową bramki NAND**.

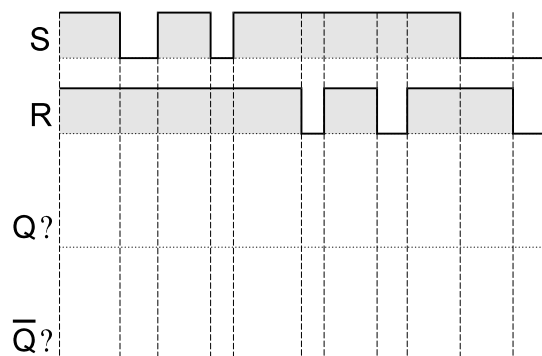
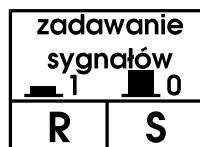
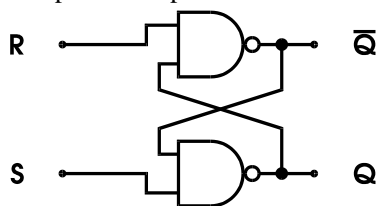
Na podstawie charakterystyki przejściowej określić **napięcia poziomów logicznych** dla stanu **HIGH** oraz **LOW** na wejściu i wyjściu bramki.



U_i [V]	U_o [V]
0,0	
0,5	
0,8	
1,1	
1,3	
1,6	
2,0	
5,0	

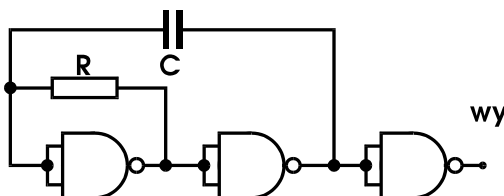
3. Badanie przerzutnika SR

Na wejścia **S** i **R** przerzutnika należy podać (za pomocą przycisków S, R) odpowiednie **sekwencje** sygnałów logicznych (patrz rysunek). Stan wyjścia **Q** i \bar{Q} określić za pomocą próbnika stanów logicznych. **Uzupełnić** rysunek wykresami poziomów logicznych na wyjściu **Q** i \bar{Q} przerzutnika. Wyniki pomiarów porównać z **tablicą prawdy** przerzutnika **SR** przedstawioną na **rys. 7.2**.



4. Wykorzystanie bramek NAND do budowy generatora fali prostokątnej

Do zbudowania generatora fali prostokątnej wykorzystano trzy bramki NAND układu **UCY 7400** (patrz rysunek). Za pomocą oscyloskopu pomierzyć częstotliwość generowanego sygnału dla trzech wartości pojemności **C**. Wyniki pomiarów zamieścić w tabeli.

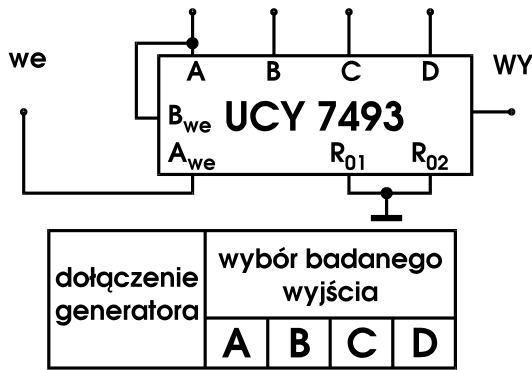


C	f [Hz]
C1	
C2	
C3	

wybór pojemności C		
1	2	3

5. Badanie dzielnika częstotliwości

WE	f_{WE}	f_{WY}	f_{WE}/f_{WY}
A			
B			
C			
D			



Dzielnik częstotliwości zbudowano z wykorzystaniem układu UCY 7493 (patrz rysunek). Na wejście A_{we} układu dołączy, za pomocą przycisku, sygnał z wyjścia generatora fali prostokątnej. Za pomocą oscyloskopu pomierzysz częstotliwość sygnałów na wszystkich wyjściach (A, B, C, D) układu. Wybór wyjścia dokonuje się za pomocą przycisków oznaczonych A, B, C, D. Dla każdego z wyjść (A, B, C, D) dzielnika obliczysz stosunek częstotliwości sygnału wyjściowego do sygnału wejściowego. Wyniki umieści w tabeli. Wykaza prawidłowo działanie dzielnika.

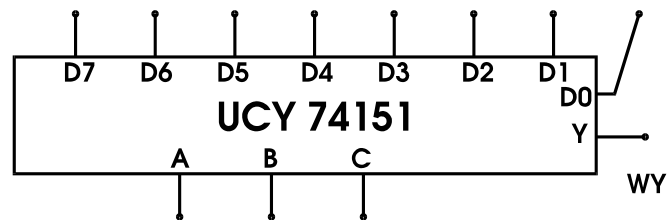
6. Badanie multipleksera

W wiczeniu wykorzystano 8-wyjściowy multipleksers UCY 74151 (patrz rysunek). Wcisnąć jeden z przycisków (dowolny), oznaczonych D₀, D₁, D₂, ..., D₇, powodujemy doprowadzenie sygnału z wyjścia dzielnika częstotliwości (badanego w poprzednim punkcie) do określonego wejścia informacyjnego (D₀, D₁, D₂, ..., D₇) multipleksera. Stan wyjścia (Y) układu obserwujemy za pomocą oscyloskopu.

Ciąg impulsów z dzielnika podanych na określone linie informacyjne multipleksera może pojawić się na jego linii wyjściowej **wyłącznie** wtedy, gdy na **liniach adresowych** (A, B, C) pojawi się, odpowiadający tej linii informacyjnej, właściwy adres oraz gdy **wejście zezwalające** (STROBE) jest w stanie 0 (przycisk STROBE jest wciśnięty).

Adres jest **kodowany binarnie** (liczbą binarną CBA) za pomocą przycisków oznaczonych A, B, C. Wciśnięcie przycisku oznacza wystawienie 1 na linii adresowej (np. dla linii informacyjnej D₅, należy na linie adresowej wystawić **słowo 101**, co oznacza wciśnięcie przycisków C oraz A).

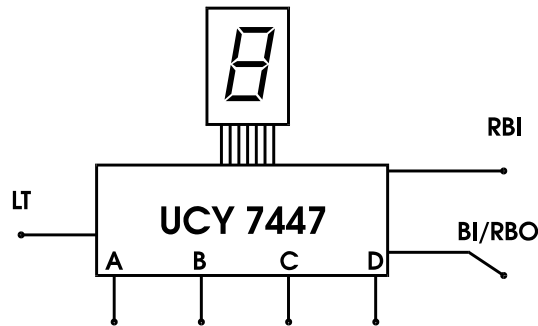
Należy zbadać zachowanie multipleksera, gdy adres jest podany **będnie** lub gdy wejście zezwalające (STROBE) jest w stanie 1. Wykaza prawidłowo działanie multipleksera odnosząc się do jego tablicy prawdy zamieszczonej na rys. 7.8.



doprowadzenie sygnału do wejścia								strob.	wybór adresu		
D ₇	D ₆	D ₅	D ₄	D ₃	D ₂	D ₁	D ₀		A	B	C

7. Badanie wskaźnika 7-segmentowego i dekodera sterującego

W wiczeniu wykorzystano wskaźnik CQZP12, który sterowany jest za pomocą dekodera sterującego UCY 7447 (patrz rysunek). Należy sprawdzić prawidłowość działania układu wskaźnikowego przez zadawanie, za pomocą przycisków oznaczonych A, B, C, D, różnych cyfr dziesiętnych (od 0 do 9) w kodzie BCD(8421) na wejścia A, B, C, D dekodera (słowo binarne - DCBA). Wciśnięcie przycisku oznacza wystawienie 1 na określone wejście.



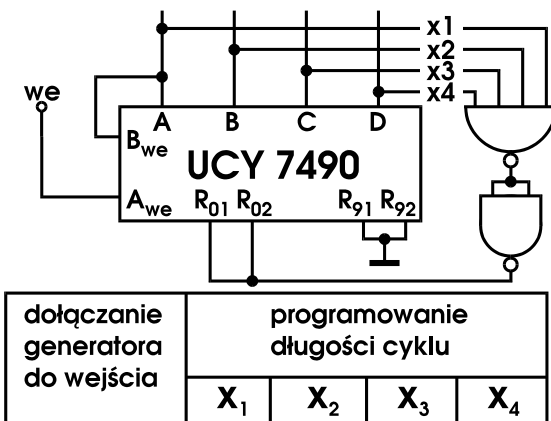
■ badanie elem. wsk. ▬ dot. elem. do liczn. cykl.	zadawanie sygnałów					
	A	B	C	D	RBI	LT

Sprawdź kolejno działanie wejść **LT**, **RBI**, **RBO** przez podanie 0 logicznego (masy) na te wejścia. **UWAGA!** - aby podać 0 na wejście **LT** oraz **RBI** należy wcisnąć odpowiedni przycisk oznaczony **LT**, **RBI**. Natomiast, aby podać 0 na wejście **RBO** należy gniazdo oznaczone **RBO** połączyć przewodem z gniazdem masowym. **Nie zwiera do masy gniazda oznaczone LT, RBI!**

Zamieć uwagi odnośnie funkcjonowania wskaźnika cyfrowego z dekoderm sterującym.

8. Badanie licznika o programowanej długości cyklu z wywietlaniem liczby impulsów

Do zbudowania licznika o programowanej długości cyklu wykorzystano asynchroniczny licznik dziesiętny UCY 7490 (patrz rysunek). Na wejście A_{we} licznika należy podać (wcisnąć odpowiedni przycisk) impulsy wytwarzane w generatorze fali prostokątnej.



UWAGA - impulsy prostokątne dołączane do wejścia A_{we} licznika są impulsami, które przeszły przez układ dzielnika częstotliwości oraz przez multiplekser. Należy uważać, aby multiplekser był właściwie zaadresowany, gdyż w przeciwnym przypadku licznik UCY 7490 nie otrzyma żadnego impulsu do zliczenia. **Dobrym praktycznym sposobem jest kontrolowanie obecności impulsów za pomocą oscyloskopu!**

W układzie zastosowano sprzężenie zerujące stan licznika zbudowane z dwóch bramek NAND. Programowanie długości cyklu licznika polega na dołączeniu odpowiednich

wyjść A, B, C, D licznika z wejściami 4-wejściowej bramki NAND. Dołączanie odbywa się przez wciśnięcie przycisków oznaczonych X_1, X_2, X_3, X_4 . Stan wyjść A, B, C, D licznika określa w kodzie binarnym, za pomocą słowa DCBA, liczb zliczonych impulsów. Jeżeli chcemy, aby po zliczeniu określonej liczby impulsów licznik zerował się, należy te wyjścia, które są w stanie 1 po zliczeniu zadanej liczby impulsów podłączyć do wejścia 4-wejściowej bramki NAND.

PRZYKŁAD: Licznik ma liczyć "do pięciu" (licznik modulo-5). Binarnym przedstawieniem liczby dziesiętnej 5 jest 4-bitowe słowo DCBA (0101). Należy podłączyć wyjście C oraz A do wejścia bramki NAND.

Do wyjść licznika należy podłączyć, za pomocą odpowiedniego przycisku, wskaźnik 7-segmentowy ułatwiający obserwację sposobu zliczania impulsów. Dobrym sposobem zliczanych

impulsów (przez dobór pojemności C generatora oraz przez dobór stopnia podziału dzielnika) tak, aby można było swobodnie obserwować zmiany cyfr we wskaźniku.

Zamieć uwagi odnośnie funkcjonowania całego układu.

UWAGA: W sprawozdaniu należy zamieścić wyniki pomiarów, wykresy, obliczenia oraz wnioski i uwagi dotyczące poszczególnych punktów ćwiczenia.

III. Literatura

1. R. Wirko, M. Rusek, W. Marciniak: Układy scalone w pytaniach i odpowiedziach, WNT, 1987
2. M. Morris Mano: Projektowanie systemów logicznych maszyn cyfrowych, WNT, 1975
3. R. L. Tokheim: Digital electronics, GLENCOE (McGraw-Hill), USA, 1994
4. A. J. Diefenderfer, B. E. Holton: Principles of electronic instrumentation, SAUNDERS COLLEGE PUBLISHING, USA, 1994

IV. Pytania kontrolne

1. Omówi działanie tranzystora jako elementu przełączającego.
2. Omówi podstawowe funkcje logiczne i sposoby przedstawiania funkcji logicznych.
3. Omówi podstawowe bramki logiczne.
4. Przedstawi metody projektowania układów logicznych:
 - tablica prawdy
 - układowanie równań boolowskich,
 - realizacja układowa AND, OR, NOT,
 - realizacja układowa NAND.
5. Omówi system liczbowy binarny i heksadecymalny oraz wzajemną konwersję liczb zapisanych w obu systemach.
6. Omówi sposób kodowania liczb dziesiętnych w kodzie BCD(8421).
7. Co oznaczają terminy: bit, bajt, słowo?
8. Co oznaczają skróty: LSB, MSB?
9. Omówi podstawowe parametry techniczne scalonych układów cyfrowych.
10. Omówi pracę bramki NAND TTL.
11. Omówi pracę inwertera CMOS.
12. Porównaj układy TTL i CMOS.
13. Omówi przerzutnik SR.
14. Omówi przerzutnik JK.
15. Omówi budowę i działanie licznika asynchronicznego.
16. Wyjaśnij terminy: pojemność licznika, długość licznika, długość cyklu licznika, licznik modulo- S .

17. Omówi działanie multipleksera UCY 74151.

18. Omówi wskaźnik 7-segmentowy oraz działanie dekodera sterującego UCY 7447.