# Ćwiczenie 27(A, B)

## UKŁADY CYFROWE

**Część A:** ***Projektowanie i symulacja komputerowa układów logicznych***

**Część B:** ***Badania laboratoryjne wybranych układów cyfrowych***

# Spis treści

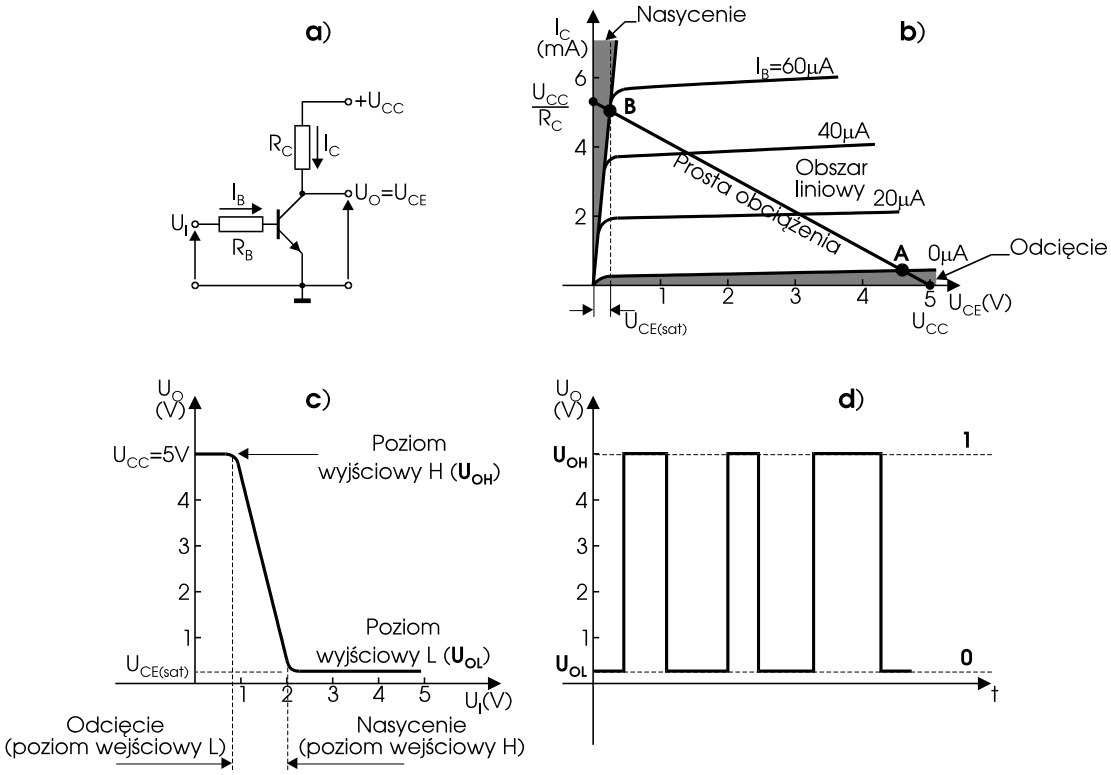
# I. Podstawowe zagadnienia z teorii układów cyfrowych

# 1. Wstęp

Muzyka na płytach fonograficznych jest zapisana w formie kanaliku o zmiennym urzeźbieniu. Ruch igły prowadzonej przez kanalik odbywa się w sposób ***ciągły*** ale ze zmieniającą się amplitudą i częstotliwością - odzwierciedlając zapisane frazy muzyczne. Tak zarejestrowany sygnał może przybierać 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* 5MΩ *do wartości poniżej* 50Ω*. Z przebiegu charakterystyki przejściowej układu wyraźnie widać odwrócenie poziomów napięć 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 $\mathbf{U_{OH}}$ (tranzystor wyłączony) odpowiada stanowi logicznemu **HIGH** albo **1**, natomiast niski poziom napięcia wyjściowego $\mathbf{U_{OL}}$ (tranzystor włączony)

odpowiada stanowi logicznemu **LOW** albo **0**. Wartości napięć wyjściowych, przedstawionych na charakterystyce przejściowej, odpowiadające 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 składnikiem wszystkich 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, za 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. ***V****ery* ***L****arge* ***S****cale* ***I****ntegration).*

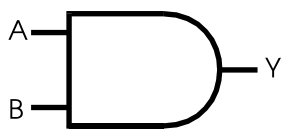
# 2.Funkcje logiczne

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 sprzętową (układową) realizacją swojej własnej listy rozkazów.

**Funkcją logiczną** nazywa się funkcję, której argumenty (zmienne logiczne) oraz sama funkcja mogą przybierać tylko jedną z dwu 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 wejść** układu cyfrowego, a wartości samej funkcji - **stanom wyjść** 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

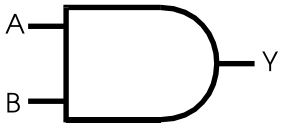
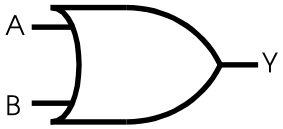
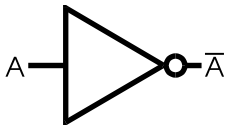
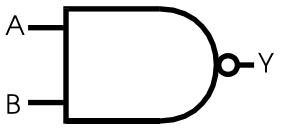
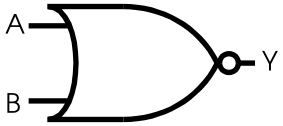
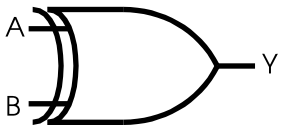
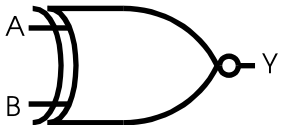
<table>
<tr><td>Opis słowny</td><td>Iloczyn logiczny argumentów A i B jest równy 1, gdy A=1 i B=1</td></tr>
<tr><td>Tablica prawdy - Truth table<br>Zawiera wszystkie kombinacje możliwych stanów wejść układu logicznego i odpowiadające im stany wyjścia tego układu. Liczba stanów układu jest równa $2^n$, gdzie n jest liczbą wejść układu.</td><td><table><tr><th>A</th><th>B</th><th>Y</th></tr><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></table></td></tr>
<tr><td>Wyrażenie algebraiczne<br>(czytaj: A i B równa się Y)</td><td>A • B = Y</td></tr>
<tr><td>Symbol logiczny<br>(nazwa skrótowa - AND)</td><td> 
</td></tr>
</table>

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 także *funktorami logicznymi*) są najprostszymi układami cyfrowymi realizującymi elementarne funkcje logiczne. Służą one do budowy układów logicznych o większej złożoności. 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 | WYRAŻENIE ALGEBRAICZNE | TRUTH TABLE | | |
|---|---|---|---|---|---|
| | | | Inputs | | Output |
| | | | A | B | Y |
| **AND** |  | $A \cdot B = Y$ | 0<br>0<br>1<br>1 | 0<br>1<br>0<br>1 | 0<br>0<br>0<br>1 |
| **OR** |  | $A + B = Y$ | 0<br>0<br>1<br>1 | 0<br>1<br>0<br>1 | 0<br>1<br>1<br>1 |
| **NOT** *(Inverter)* |  | $A = \overline{A}$ | 0<br>1 | | 1<br>0 |
| **NAND** |  | $\overline{A \cdot B} = Y$ | 0<br>0<br>1<br>1 | 0<br>1<br>0<br>1 | 1<br>1<br>1<br>0 |
| **NOR** |  | $\overline{A + B} = Y$ | 0<br>0<br>1<br>1 | 0<br>1<br>0<br>1 | 1<br>0<br>0<br>0 |
| **XOR** |  | $A \oplus B = Y$ | 0<br>0<br>1<br>1 | 0<br>1<br>0<br>1 | 0<br>1<br>1<br>0 |
| **XNOR** |  | $\overline{A \oplus B} = Y$ | 0<br>0<br>1<br>1 | 0<br>1<br>0<br>1 | 1<br>0<br>0<br>1 |

Bramki **AND**, **OR**, **NOT** tworzą tzw. ***funkcjonalnie pełny*** zestaw elementów. Oznacza to, że można z nich zbudować dowolnie złożony układ logiczny.

Za pomocą wyłącznie bramek **NAND** albo wyłącznie bramek **NOR** można także zrealizować dowolnie złożoną funkcję, w tym również funkcje podstawowe **AND**, **OR**, **NOT**. Z tego powodu mówimy, że bramki **NAND** i **NOR** oddzielnie tworzą tzw. ***minimalny*** zestaw ***funkcjonalnie pełny***. W pierwszej chwili może się wydawać, że stosowanie funktorów **NAND** lub **NOR** do realizacji prostych funkcji **iloczynu**, **sumy** i **negacji** jest niepotrzebne i tylko komplikuje postawione zadanie. W praktyce okazuje się, że zalety stosowania jednoelementowego zbioru do realizacji dowolnej funkcji logicznej są bardzo duże. Przeglądając katalogi producentów układów cyfrowych można z łatwością zauważyć, że najszerszą ofertę stanowią bramki **NAND**, gdyż są one najchętniej stosowane przez użytkowników (można powiedzieć, że bramka **NAND** jest bramką uniwersalną).

Bramki **XOR** oraz **XNOR**, ze względu na swoją specyfikę, służą 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. ***S**mall **S**cale **I**ntegration)*.

# 4. Projektowanie układów logicznych

Jednym z najbardziej fascynujących aspektów **elektroniki cyfrowej** jest możliwość konstrukcji układów o prostej "mentalności", ale zdolnych do przetwarzania informacji z niezwykle dużą (w porównaniu z człowiekiem) prędkością. Typowy komputer potrafi wykonać tysiące **dodawań** 10-cio pozycyjnych liczb w ciągu sekundy.

Zagadnienie projektowania układów logicznych sprowadza się do utworzenia schematu układu, zawierającego odpowiednio połączone bramki logiczne, zdolnego do zrealizowania pożądanej 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 bramki **AND**, **OR**, **NOT**. W praktyce taki schemat wymaga najczęściej **przekształcenia** w realizację układową złożoną 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*). Określenia ***TRUE*** i ***FALSE*** stanowią, 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 tożsamości przedstawione w **tablicy 4.1**.

**Tablica 4.1** Podstawowe prawa i tożsamości 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 | $\overline{A \cdot B \cdot \ldots} = \overline{A} + \overline{B} + \ldots$ | $\overline{A + B + \ldots} = \overline{A} \cdot \overline{B} \cdot \ldots$ |
| **Tożsamości** | podstawowe | $A \cdot 0 = 0$<br>$A \cdot 1 = A$<br>$A \cdot \overline{A} = A$<br>$A \cdot \overline{A} = 0$ | $A + 1 = 1$<br>$A + 0 = A$<br>$A + \overline{A} = A$<br>$A + \overline{A} = 1$ |
| | dodatkowe | $A \cdot (A + B) = A$<br>$A + \overline{A} \cdot B = A + B$<br>$(A + B) \cdot (\overline{A} + B) = B$ | $A + \overline{A} \cdot B = A$<br>$A \cdot (\overline{A} + B) = A \cdot B$<br>$A \cdot B + \overline{A} \cdot B = B$ |

Symbolem "$\cdot$" oznacza się operację iloczynu logicznego **AND**, symbolem "**+**" oznacza się operację sumy logicznej **OR**, symbolem "–" oznacza się operację negacji **NOT**.

Zależności przedstawione w **tablicy 4.1** wykorzystuje się przy przekształcaniu wyrażeń opisujących złożone funkcje logiczne o wielu zmiennych w celu otrzymania ich w możliwie

najprostszej postaci końcowej, a co za tym idzie, prostszej realizacji układowej. Proces ten jest określany jako *minimalizacja funkcji logicznej*.

## 4.2 Układanie równań boolowskich i ich przekształcanie do postaci minimalnej

Postawienie zadania zbudowania układu logicznego polega najczęściej na **opisie słownym**, od którego przechodzi się następnie do określenia **tablicy prawdy**. Na podstawie tablicy prawdy można w sposób rutynowy ułożyć **równania boolowskie** opisujące w sposób algebraiczny działanie logiczne układu. Postępowanie to zostanie zilustrowane w formie przykładu.

> **PRZYKŁAD:** W samolocie, dla zwiększenia poziomu ufności, zastosowano potrójne systemy czujników. Zaprojektować układ logiczny uruchamiający automatycznego pilota wyłącznie wtedy, gdy co najmniej dwa z tych systemów są aktywne. *Tekst powyższy jest typowym **opisem słownym** pewnego zadania logicznego.*

Systemy czujników oznaczmy literami A, B, C. Są to **zmienne** funkcji logicznej czyli **wejścia** układu logicznego. Stanowi aktywnemu czujnika przypisujemy poziom **1** (***Prawdziwy*** - *TRUE*), w przeciwnym wypadku - poziom **0** (***Fałszywy*** - *FALSE*).

Sygnał uruchamiający automatycznego pilota oznaczmy literą Y. Jest to **wartość** funkcji logicznej czyli **wyjście** układu logicznego. Sygnałowi uruchomienia automatycznego pilota przypisujemy poziom **1** (***Prawdziwy*** - *TRUE*), w przeciwnym wypadku - poziom **0** (***Fałszywy*** - *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** |

Określamy **tablicę prawdy** układu logicznego wypisując wszystkie możliwe kombinacje stanów wejść i odpowiadające im stany wyjścia (zgodnie z logiką zawartą w opisie słownym).

Każdą funkcję logiczną można przedstawić w postaci rozłożonej na tzw. ***składniki jedynki***. Postać uzyskana w wyniku rozkładu na *składniki jedynki* jest nazywana **postacią kanoniczną sumy**.

Kanoniczną postać sumy można otrzymać na podstawie **tablicy prawdy**, biorąc pod uwagę jedynie te wiersze, dla których **Y=1** i przypisując wartościom **1** - zmienne niezanegowane, a wartościom **0** - negacje zmiennych. Postępując zgodnie z powyższą receptą otrzymamy dla naszej tablicy prawdy **równanie boolowskie**, które w sposób algebraiczny opisuje zadany problem logiczny:

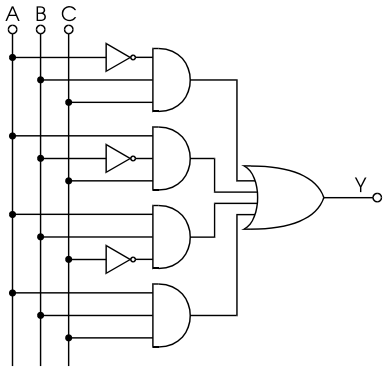
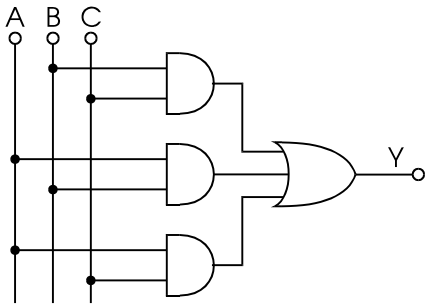
$$Y = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$$

Równanie to poddamy teraz procesowi **minimalizacji funkcji logicznej**. Minimalizacja funkcji logicznej polega na takim przekształceniu postaci kanonicznej funkcji, zgodnie z zasadami **algebry Boole'a**, aby uzyskać możliwie najprostszy jej zapis. Każde uproszczenie równania boolowskiego umożliwia łatwiejszą realizację układową funkcji - przy użyciu mniejszej liczby elementarnych bramek logicznych. *Metoda przekształcania wyrażeń algebraicznych jest metodą intuicyjną, istnieją również inne metody minimalizacji funkcji logicznej (sformalizowane), o których nie będziemy tutaj mówić*. Spróbujmy przekształcić nasze równanie stosując prawa i tożsamości algebry Boole'a:

$$Y = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC = BC(\overline{A} + A) + A\overline{B}C + AB\overline{C} = BC + A\overline{B}C + AB\overline{C} =$$
$$= B(C + A\overline{C}) + A\overline{B}C = B(C + A) + A\overline{B}C = BC + AB + A\overline{B}C = BC + A(B + \overline{B}C) =$$
$$= BC + A(B + C) = BC + AB + AC$$

## 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.

| $Y = \overline{A}BC + A\overline{B}C + AB\overline{C} + ABC$ | $Y = BC + AB + AC$ |
|---|---|
|  |  |



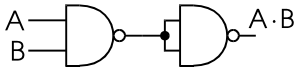
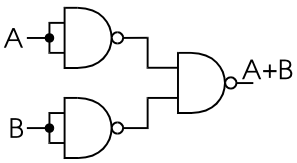
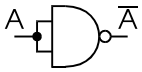
**Rys. 4.1** Realizacje układowe równań boolowskich z zastosowaniem bramek **AND, OR**, **NOT**

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 stałość poziomów 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)* |
|---|---|---|
|  |  |  |
| $\overline{\overline{A \cdot B}} = A \cdot B$ | $\overline{\overline{A} \cdot \overline{B}} = \overline{\overline{A + B}} = A + B$ | $\overline{A \cdot A} = \overline{A}$ |



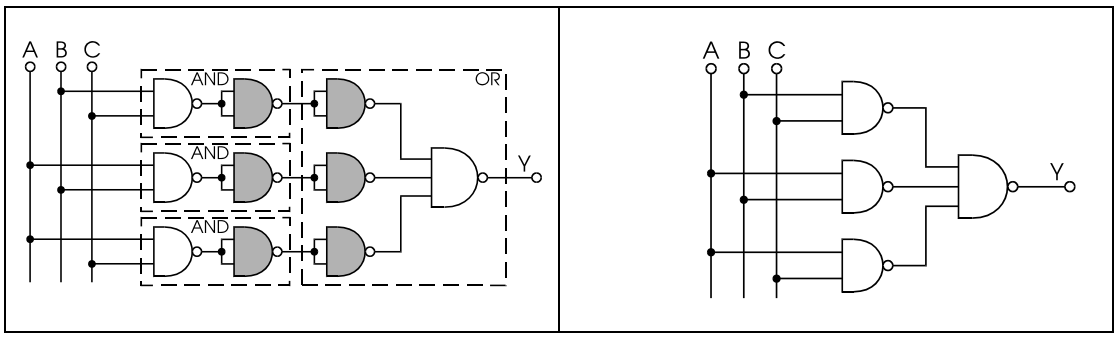
Aby zrealizować funkcję boolowską za pomocą bramek **NAND**, zastosować można prostą **metodę przekształcania schematów logicznych**. Metoda ta wymaga co prawda narysowania dwu schematów logicznych przed otrzymaniem właściwego schematu realizacji **NAND**-owej, lecz jest ona w sumie bardzo prosta:

**1.** Na podstawie danego równania boolowskiego rysujemy schemat logiczny zawierający bramki **AND**, **OR**, **NOT**.

**2.** Rysujemy drugi schemat logiczny, w którym każą z bramek **AND**, **OR**, **NOT** zastępujemy przez układ równoważny, zbudowany z bramek **NAND**, zgodnie z **tablicą 4.2**.

**3.** Usuwamy ze schematu wszystkie pary kaskadowo połączonych **inwerterów** (gdyż 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 zacieniowano 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:

$$(503,14)_{10} = 5\times10^{2} + 0\times10^{1} + 3\times10^{0} + 1\times10^{-1} + 4\times10^{-2}$$

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 sa równe potęgom 2. Przykładowo, liczba **11010,11** w zapisie binarnym oznacza:

$$(11010,11)_{2} = 1\times2^{4} + 1\times2^{3} + 0\times2^{2} + 1\times2^{1} + 0\times2^{0} + 1\times2^{-1} + 1\times2^{-2} = (26,75)_{10}$$

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:

$$(\mathrm{B65F})_{16} = 11\times16^{3} + 6\times16^{2} + 5\times16^{1} + 15\times16^{0} = (46687)_{10}$$

**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 podzielić liczbę binarną na grupy liczące po cztery cyfry, zaczynając od cyfry o najmniejszej wadze, np:

$$\overset{\boldsymbol{2}}{(..10}\ \overset{\boldsymbol{C}}{1100}\ \overset{\boldsymbol{6}}{0110}\ \overset{\boldsymbol{B}}{1011})_2 = (2C6B)_{16} = (11371)_{10}$$

$$\overset{\boldsymbol{F}}{(1111}\ \overset{\boldsymbol{F}}{1111})_2 = (FF)_{16} = (255)_{10}$$

$$\overset{\boldsymbol{F}}{(1111)}_2 = (F)_{16} = (15)_{10}$$

## 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. ***M**ost **S**ignificant **B**it*), a najmniej znaczący bit - jako **LSB** (ang. ***L**east **S**ignificant **B**it*). 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 dwu 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 sześć 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. ***B**inary-**C**oded **D**ecimals*), 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 |

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

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

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

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. ***A**merican **S**tandard **C**ode for **I**nformation **I**nterchange*). Pełny zbiór zawiera 128 znaków kodowanych słowami 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:

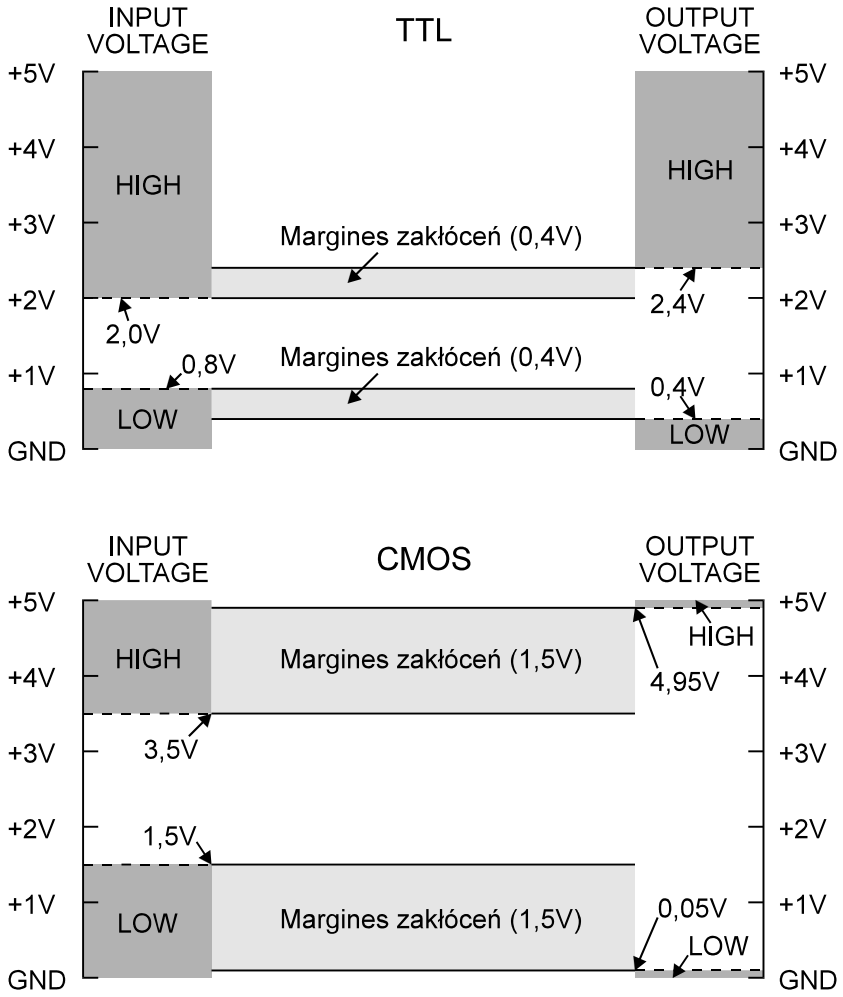
- **układy TTL** (ang. ***T**ransistor-**T**ransistor **L**ogic*), które są **układami bipolarnymi**,
- **układy CMOS** (ang. ***C**omplementary **M**etal **O**xide **S**emiconductor*), które są **układami unipolarnymi** - z tranzystorami p-MOS oraz n-MOS.

## Podstawowe parametry techniczne układów cyfrowych

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

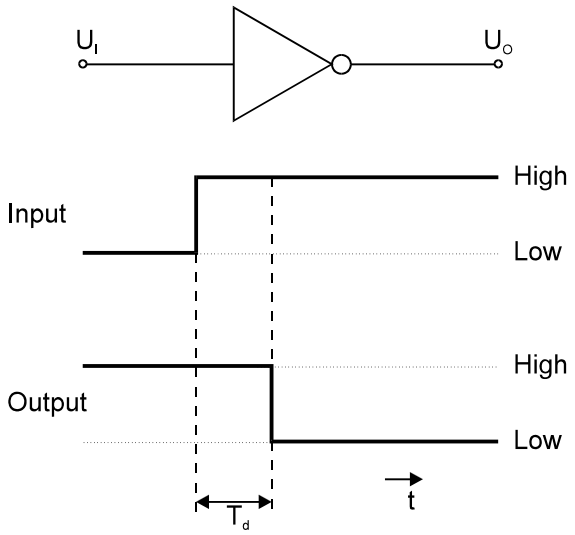
★**Napięcia poziomów logicznych (HIGH, LOW)** - zakresy napięć wejściowych oraz wyjściowych, które układ realizuje jako **gwarantowany** stan **1** oraz **gwarantowany** stan **0**.

★**Margines zakłóceń ($U_{NM}$)** - określa dopuszczalną wartość napięcia sygnału zakłócającego, nie powodującego 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 podstawową miarą szybkości działania układu cyfrowego.

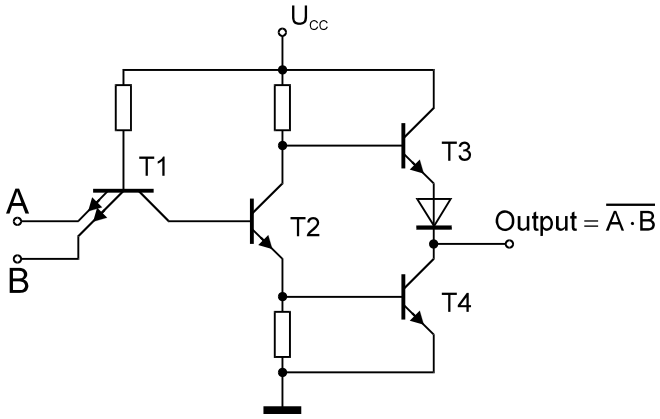



**Rys. 6.2** Czas propagacji dla układu inwertera

★**Moc strat na bramkę ($P_d$)** - określa moc **pobieraną** przez układ ze źródła zasilania.

## Układy TTL

Podstawową, dwuwejściową bramkę NAND przedstawiono na **rys. 6.3**.




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

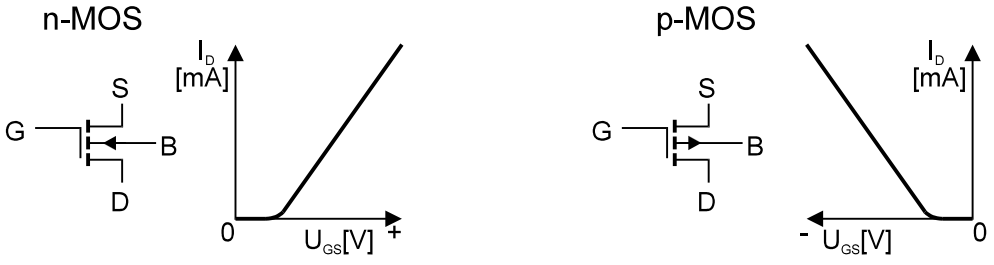
Przyłączenie jakiegokolwiek wejścia A, B lub obydwu do masy (co oznacza stan wejść równy **0**) powoduje **wyłączenie** tranzystora **T2** i **T4**, gdyż napięcie na **bazie** tranzystora **T1** nie przekracza wartości 0,3V+0,7V=1V i jest niewystarczające do spolaryzowania złącza **baza-kolektor** tranzystora **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 emiterem nasyconego tranzystora.

Gdy oba wejścia A, B są na poziomie **1**, wtedy złącza **baza-emitery** 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 **włączenia**. 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 wyjściowe osiąga wartość 0,3V i jest równe napięciu $U_{CE(sat)}$ tranzystora T4 - co oznacza stan **0** na wyjściu układu.

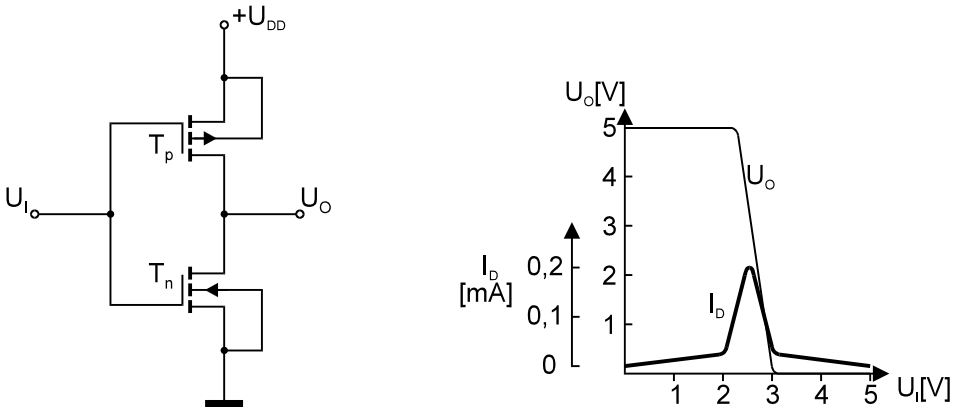
## Układy CMOS

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 wzbogacanych tranzystorów MOS

Zauważ, że przy napięciu $U_{GS} = 0$ żaden 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 **źródła S** tranzystora p-MOS (oznaczonego symbolem $T_p$), podczas gdy **źródło 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**.

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

## Porównanie układówTTL i układów CMOS

Wewnątrz układów TTL i układów CMOS występuje wiele **"rodzin"** układowych charakteryzujących się różnymi parametrami technicznymi, z których najważniejsze to czas propagacji, moc strat na bramkę oraz napięcie zasilania, które ma podstawowe znaczenie przy łączeniu 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/Bramkę] 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 |

**OBJAŚNIENIA:**

HC - ***H**igh-speed **C**MOS logic*,
HCT - ***H**igh-speed **C**MOS logic (z poziomami logicznymi **T**TL)*,
AC - ***A**dvanced **C**MOS logic*,
ACT - ***A**dvanced **C**MOS logic (z poziomami logicznymi **T**TL)*,
LS - ***L**ow-power **S**chottky TTL logic*,
S - ***S**chottky TTL logic*,
ALS - ***A**dvanced **L**ow-power **S**chottky TTL logic*,
AS - ***A**dvanced **S**chottky 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 **funkcję** danego układu oraz identyczny **rozkład wyprowadzeń**. Umożliwia to **łączenie** układów CMOS i TTL **między sobą** (są one układowo **kompatybilne**).

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

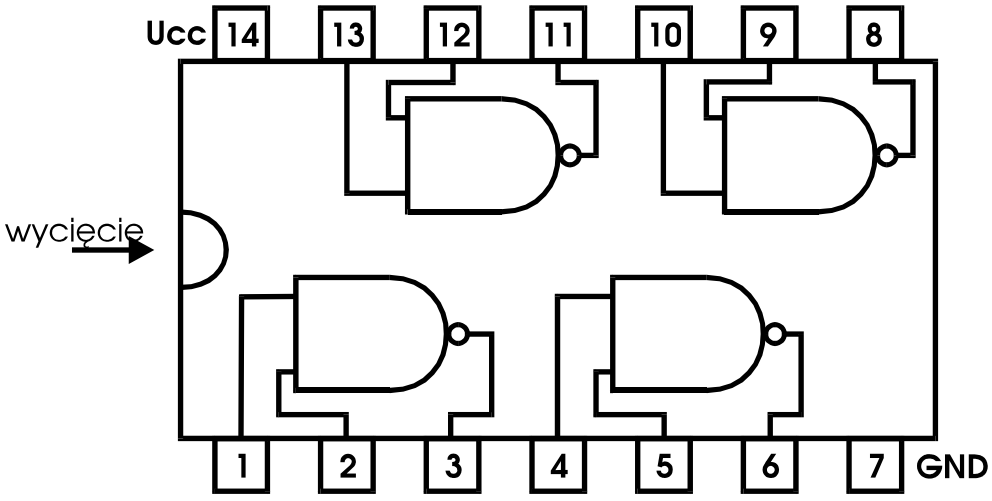
★Największą **gęstość upakowania** elementów w chip-ie osiąga się w układach CMOS, stąd układy skali **LSI** oraz **VLSI** są układami CMOS.

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

★Układy CMOS charakteryzują się znacznie **większym marginesem zakłóceń** 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 dwuwejściowe bramki NAND**. Wycięcie w obudowie jednoznacznie określa kolejność wyprowadzeń.

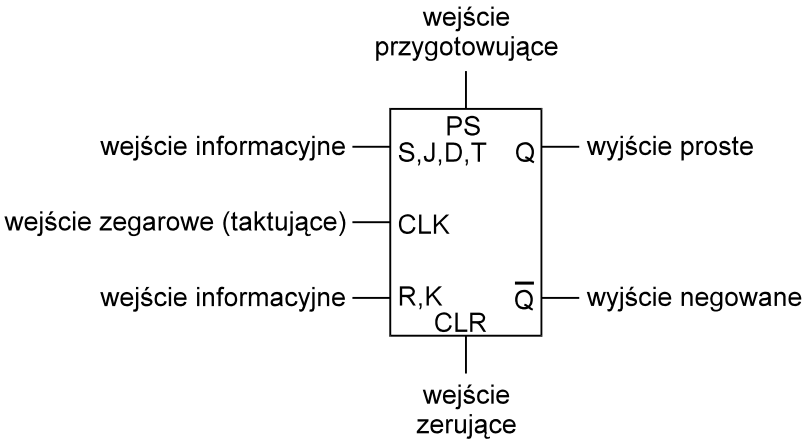



**Rys. 6.6** Widok z góry wyprowadzeń układu UCY 7400

# 7. Omówienie układów cyfrowych zastosowanych w ćwiczeniu

## 7.1 Przerzutniki

**Przerzutnikiem** (ang. *Flip-Flop*) jest nazywamy 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

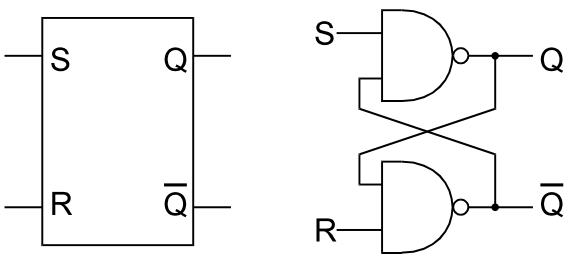
Przerzutnik ma pewną liczbę wejść i z reguły dwa wyjścia. Stan logiczny wyjścia **Q** uważa się za **stan przerzutnika**. Nazwy literowe wejść informacyjnych (**SR**, **JK**, **D**, **T**) określają jednocześnie nazwy rodzajów przerzutników (*przerzutnik* **SR**, *przerzutnik* **JK**, *przerzutnik* **D**, *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*) służą 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 wejść.

**Przerzutniki synchroniczne** pracują z udziałem sygnału zegarowego, a stan wejść informacyjnych jest przekazywany na wyjście w chwilach występowania narastającego lub opadającego zbocza sygnału zegarowego (zazwyczaj wykorzystuje się zbocze opadające).

### <u>Przerzutnik SR</u> (asynchroniczny)

**Przerzutnik SR** (ang. ***S**et*, ***R**eset*) jest najprostszym układem przerzutnika bistabilnego. Symbol graficzny, realizację NAND-ową oraz tablicę prawdy przerzutnika SR podano na **rys. 7.2**.



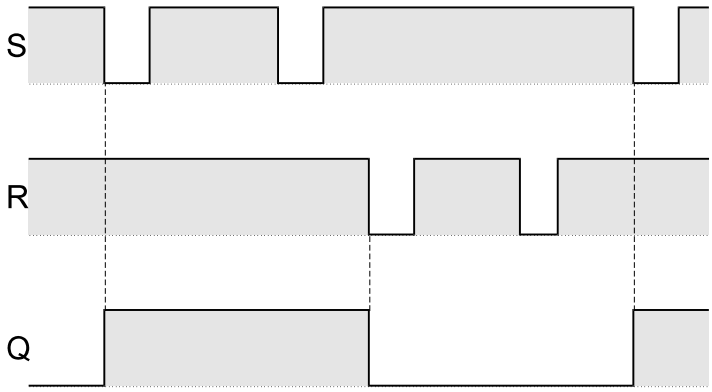

| S | R | Q | |
|---|---|---|---|
| 0 | 0 | s.n. | (stan niedozwolony) |
| 0 | 1 | 1 | |
| 1 | 0 | 0 | |
| 1 | 1 | b.z. | (bez zmian) |

**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!***).

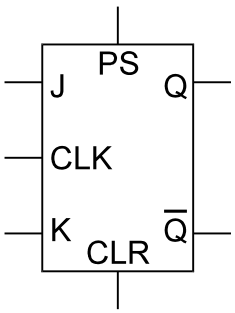
Wykres czasowy pracy przerzutnika pokazuje **rys. 7.3**.




**Rys. 7.3** Wykres czasowy przerzutnika SR

## **Przerzutnik JK** (synchroniczny)

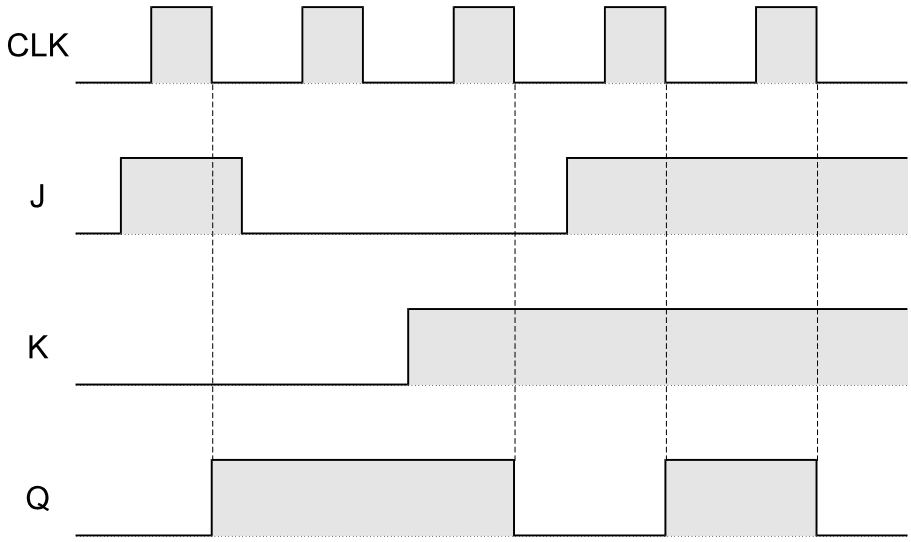
**Przerzutnik JK** jest najbardziej rozpowszechnionym układem przerzutnikowym techniki cyfrowej. Jest on traktowany jako **układ uniwersalny**. Przez zastosowanie odpowiednich połączeń zewnętrznych można z niego utworzyć inny rodzaj przerzutnika, np. **SR**(syn.), **D**, **T**. Symbol graficzny oraz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




| J | K | $Q_{n+1}$ | |
|---|---|---|---|
| 0 | 0 | $Q_n$ | (bez zmian) |
| 0 | 1 | 0 | |
| 1 | 0 | 1 | |
| 1 | 1 | $\overline{Q_n}$ | (zmiana stanu na przeciwny) |

**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 $\mathbf{t_{n+1}}$ (t.j. po przyjściu impulsu zegarowego) zależy od stanów **J**, **K**, **Q** w czasie $\mathbf{t_n}$ (t.j. przed przyjściem 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**.

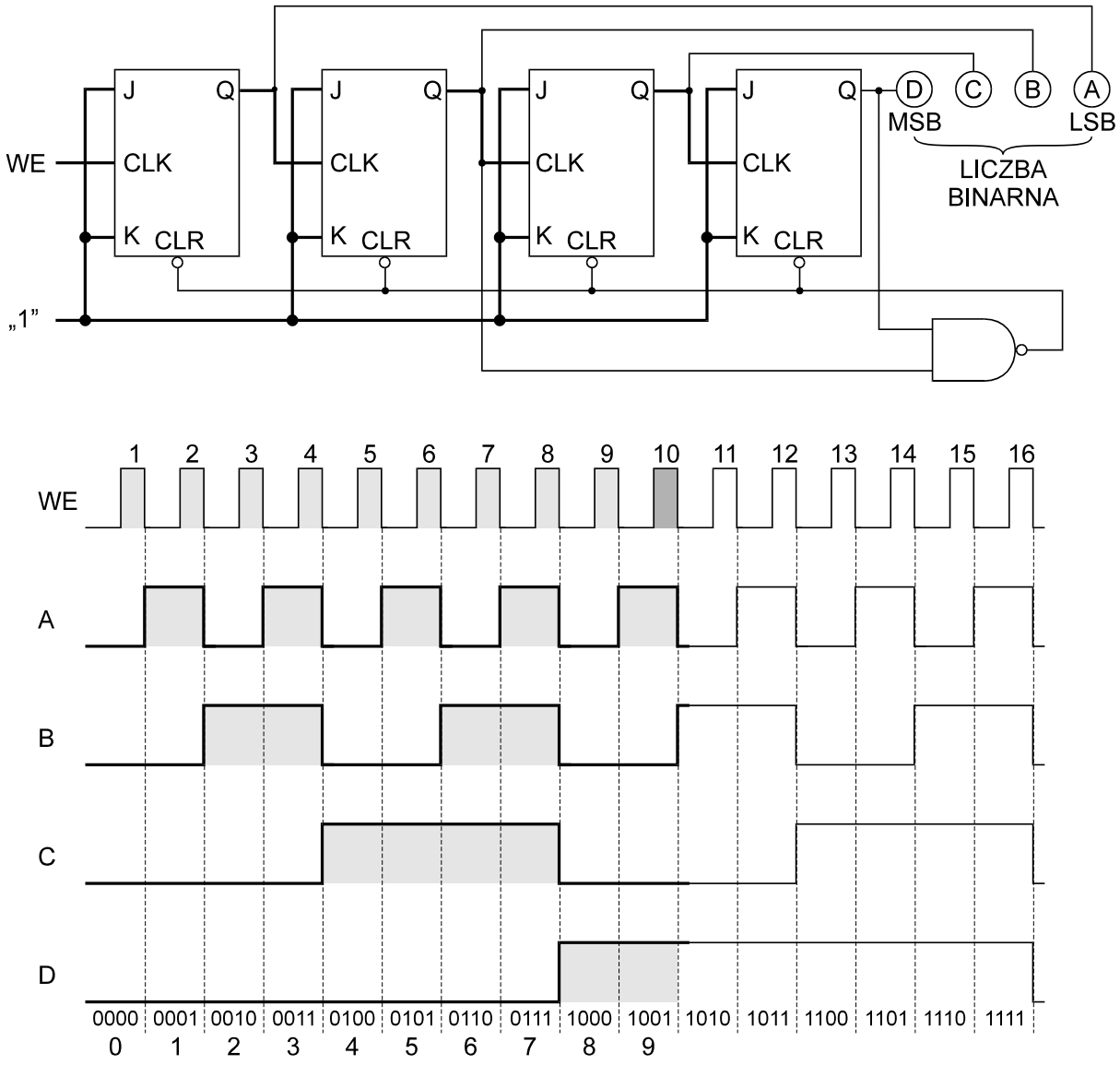



**Rys. 7.5** Wykres czasowy przerzutnika JK

**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 tę 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 $\mathbf{P=2^n}$. Liczbę **n** nazywamy **długością licznika**. Zapełnienie licznika kończy cykl pracy licznika, po czym wraca on do stanu początkowego. **Długością cyklu licznika** (**S**) nazywamy liczbę wyróżnialnych stanów logicznych, przez które licznik przechodzi cyklicznie. Jeżeli $\mathbf{S\leq 2^n}$ to mówimy, że jest to **licznik modulo-S** (np. licznik **modulo-10** jest licznikiem dziesiętnym, tzw. dekadą liczącą, przy czym **n=4** oraz $\mathbf{P=2^4=16}$). Schemat i zasadę 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 **1**0**1**0) służy do zerowania licznika po każdym cyklu liczenia od **0** do **9**.

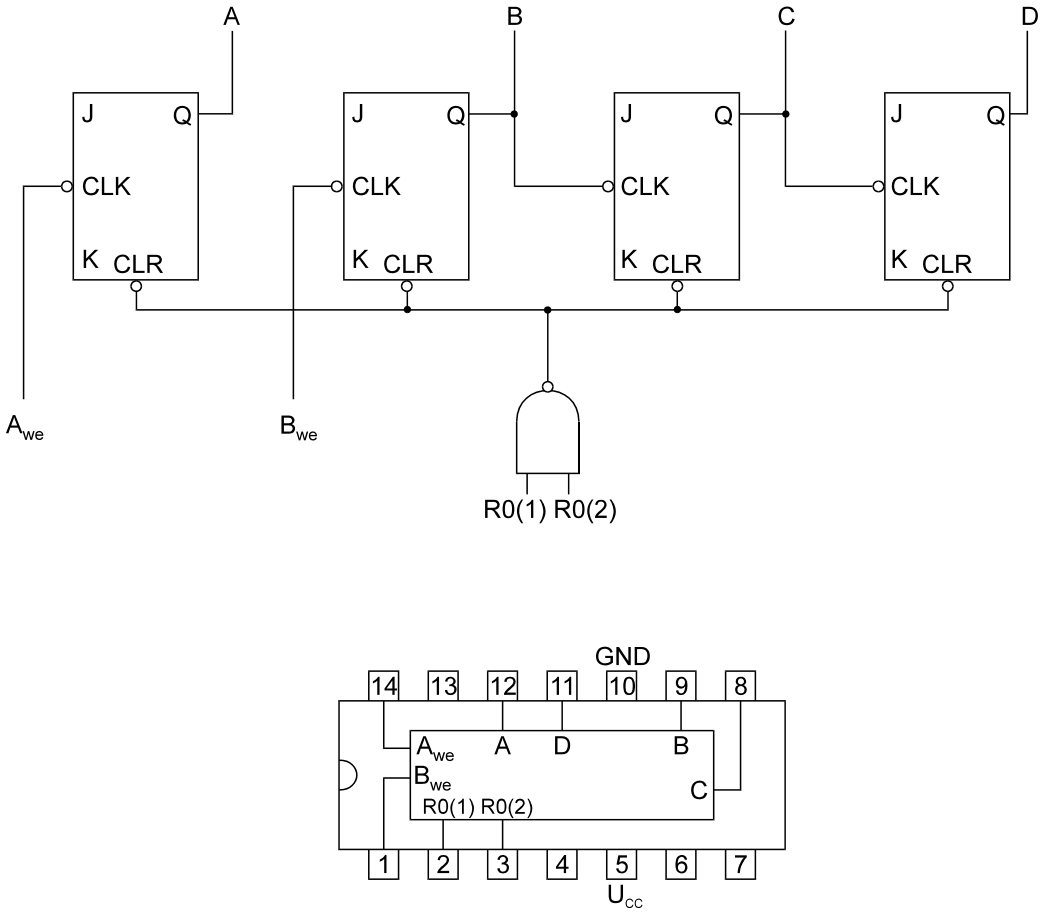
**Dzielnikiem częstotliwości** nazywamy układ dający impuls co **N** impulsów wejściowych (N jest liczbą całkowitą). Wtedy mamy:

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

Zauważmy, że **licznik** z **rys.7.6** może być traktowany jako **dzielnik częstotliwości**, gdyż 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 składajacym się z przerzutników JK połączonych w ten sposób, że tworzą 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**.



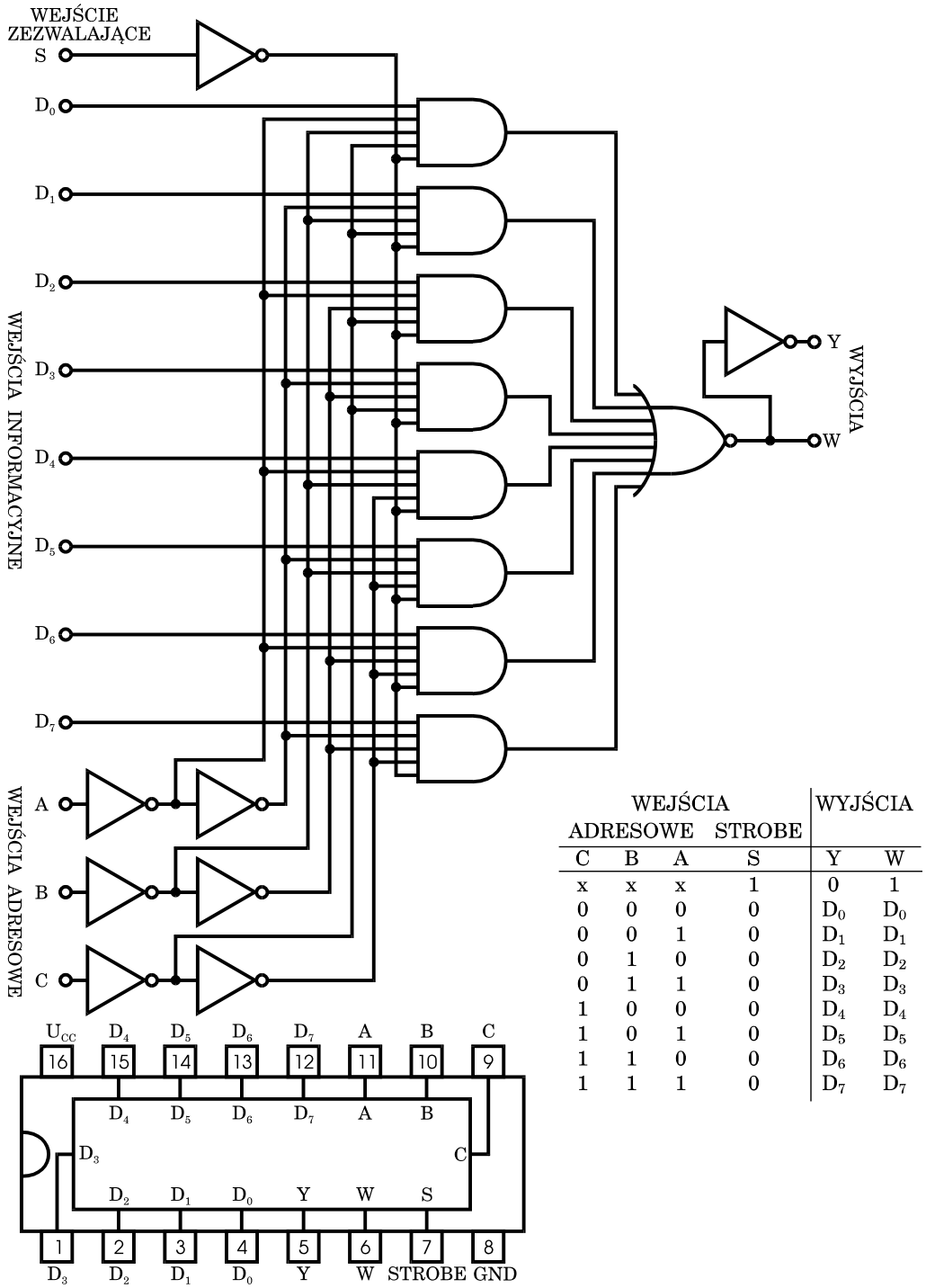

**Rys. 7.7** Schemat logiczny i widok wyprowadzeń układu UCY 7493

## 7.3 Multiplekser

W systemach cyfrowych często istnieje potrzeba przesyłania **selektywnie wybranej** informacji binarnej bądź też zmiany sposobu przesyłania z **równoległego** (**N bitów** - **N linii**) na **szeregowy** (**N bitów** - kolejno w **jednej linii**). Realizację tego zadania umożliwia multiplekser. Multiplekser cyfrowy przekazuje sygnał cyfrowy 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 wyprowadzeń **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.

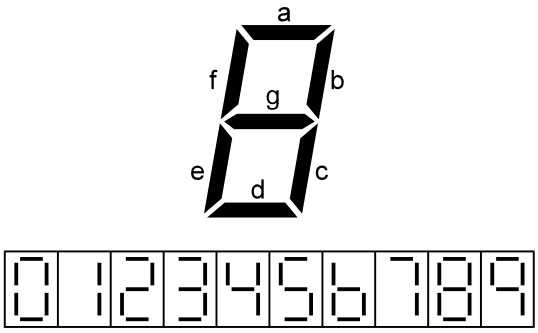



| WEJŚCIA ADRESOWE | | | WEJŚCIA STROBE | WYJŚCIA | |
|---|---|---|---|---|---|
| C | B | A | S | Y | W |
| x | x | x | 1 | 0 | 1 |
| 0 | 0 | 0 | 0 | $D_0$ | $D_0$ |
| 0 | 0 | 1 | 0 | $D_1$ | $D_1$ |
| 0 | 1 | 0 | 0 | $D_2$ | $D_2$ |
| 0 | 1 | 1 | 0 | $D_3$ | $D_3$ |
| 1 | 0 | 0 | 0 | $D_4$ | $D_4$ |
| 1 | 0 | 1 | 0 | $D_5$ | $D_5$ |
| 1 | 1 | 0 | 0 | $D_6$ | $D_6$ |
| 1 | 1 | 1 | 0 | $D_7$ | $D_7$ |

**Rys. 7.8** Schemat logiczny i widok wyprowadzeń układu UCY 74151

## 7.4 Wskaźniki półprzewodnikowe i układy sterujące nimi

Powszechnie stosuje się wskaźniki z **diod elektroluminescencyjnych** i **ciekłokrystaliczne**. 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.



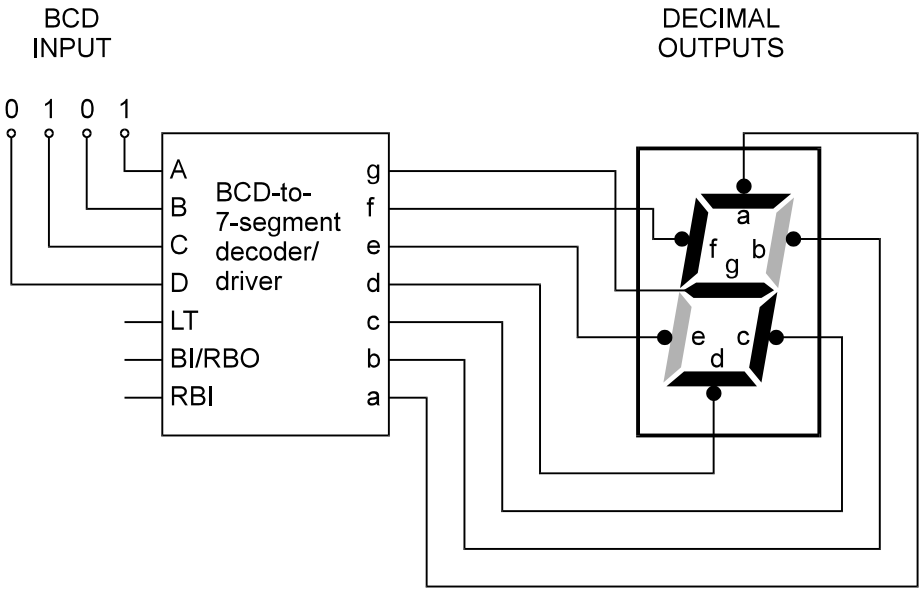

**Rys. 7.9** Wskaźnik 7-segmentowy

Najważniejszymi **zaletami wskaźników z diod elektroluminescencyjnych** są 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 ciekłokrystaliczne** mają najniższy pobór mocy ze wszystkich rodzajów wskaźników, lecz są niestety także wśród nich najwolniejsze w działaniu. Ponadto wymagają oświetlenia tym silniejszego, im bardziej kontrastowy obraz znaku pragnie się uzyskać.

## Dekoder UCY 7447

Do sterowania wskaźników 7-segmentowych 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**.




**Rys. 7.10** Działnie układu UCY 7447

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 wyłącznie gdy pokazują cyfrę 0.

# II. Wykonanie ćwiczenia

## Część 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: tablicę 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 wejść**. **Stan wyjścia** układu porównać z **tablicą prawdy** funkcji logicznej.

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

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

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

a) $Y = ABCD + A\overline{B}CD$

b) $Y = AB + \overline{AC} + A\overline{B}C(AB + C)$

c) $Y = A + B(\overline{C + \overline{DE}})$

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

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

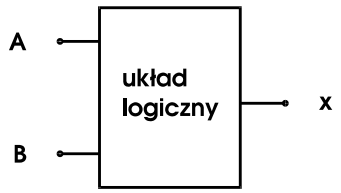
**ZADANIE 6.** Dokonać symulacji komputerowej przerzutnika **RS** w realizacji NAND-owej i sprawdzić jego działanie poprzez zadawanie odpowiednich stanów wejść i badanie stanu wyjść.

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

# Część B Badania laboratoryjne wybranych układów cyfrowych

## 1. Określenie typu bramki na podstawie tablicy prawdy

Dla każdej z czterech nieznanych bramek (wybieranych kolejno przyciskami 1, 2, 3, 4) należy określić **tablicę prawdy** oraz dokonać **identyfikacji** bramki. Stany wyjść (**X**) każdej bramki należy określić za pomocą próbnika stanów logicznych dla wszystkich kombinacji stanów wejść. 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 sygnałów 1 0 | | wybór układu logicznego | | | |
|---|---|---|---|---|---|
| A | B | 1 | 2 | 3 | 4 |

**1**

| A | B | X |
|---|---|---|
| 0 | 0 | |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | |

**2**

| A | B | X |
|---|---|---|
| 0 | 0 | |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | |

**3**

| A | B | X |
|---|---|---|
| 0 | 0 | |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | |

**4**

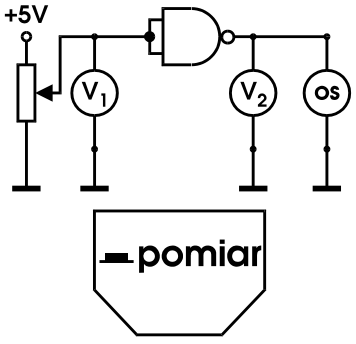
| A | B | X |
|---|---|---|
| 0 | 0 | |
| 0 | 1 | |
| 1 | 0 | |
| 1 | 1 | |

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

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

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

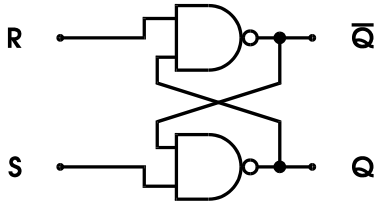
Na podstawie charakterystyki przejściowej określić **napięcia poziomów logicznych** dla stanu **HIGH** oraz **LOW** na wejściu i wyjściu bramki.

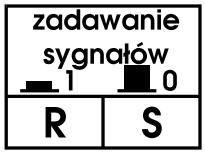


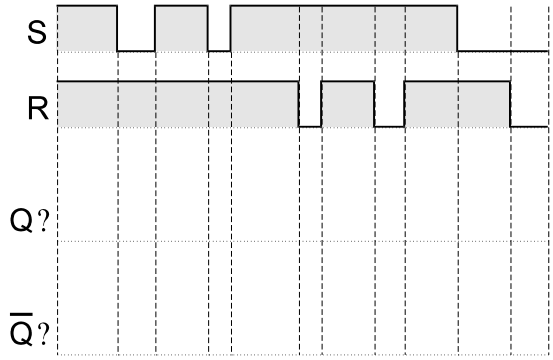

| $\mathbf{U_I}$ [V] | $\mathbf{U_O}$ [V] |
|---|---|
| 0,0 | |
| 0,5 | |
| 0,8 | |
| 1,1 | |
| 1,3 | |
| 1,6 | |
| 2,0 | |
| 5,0 | |

# 3. Badanie przerzutnika SR

Na wejścia **S** i **R** przerzutnika należy podać (za pomocą przycisków S, R) odpowiednią **sekwencję** sygnałów logicznych (patrz rysunek). Stan wyjścia **Q** i $\overline{\mathbf{Q}}$ określić za pomocą próbnika stanów logicznych. **Uzupełnić rysunek** wykreślając poziomy logiczne na wyjściu **Q** i $\overline{\mathbf{Q}}$ przerzutnika. Wyniki pomiarów porównać z **tablicą prawdy** przerzutnika **SR** przedstawioną na **rys. 7.2**.

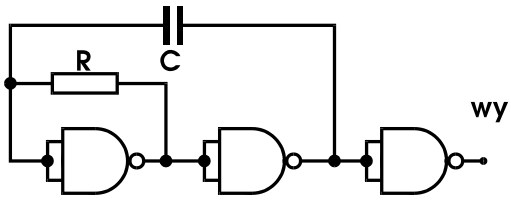



# 4. Wykorzystanie bramek NAND do budowy generatora fali prostokątnej

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.

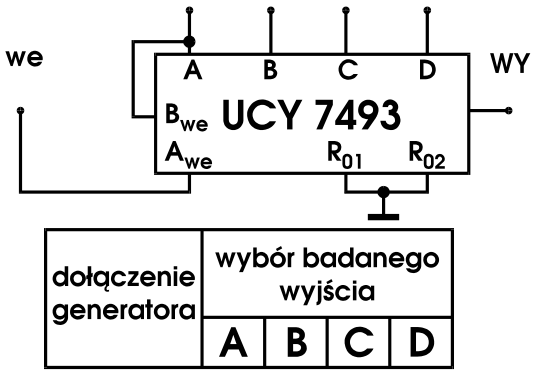




| **C** | **f** [Hz] |
|---|---|
| C1<br>C2<br>C3 | |

# 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 $\mathbf{A_{WE}}$ układu dołączyć, za pomocą przycisku, sygnał z wyjścia **generatora fali prostokątnej**. Za pomocą oscyloskopu pomierzyć 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 obliczyć stosunek częstotliwości sygnału wyjściowego do sygnału wejściowego. Wyniki umieścić w tabeli. Wykazać prawidłowość działania dzielnika.

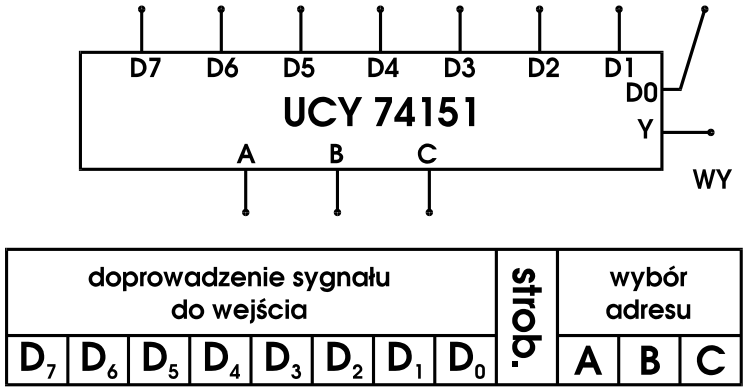
# 6. Badanie multipleksera

W ćwiczeniu wykorzystano **8-wejściowy multiplekser UCY 74151** (patrz rysunek). Wciskając jeden z przycisków (dowolny), oznaczonych $D_0$, $D_1$, $D_2$, ... , $D_7$, powodujemy doprowadzenie sygnału z wyjścia **dzielnika częstotliwości** (badanego w poprzednim punkcie) do określonego **wejścia informacyjnego** (**$D_0$**, **$D_1$**, **$D_2$**, ... , **$D_7$**) multipleksera. **Stan wyjścia** (**Y**) układu obserwujemy za pomocą oscyloskopu.

Ciąg impulsów z dzielnika podanych na określoną linię informacyjną 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).

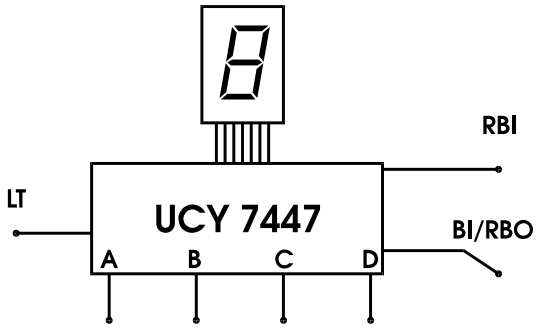
**Adres** jest **kodowany binarnie** (liczbą binarną **CBA**) za pomocą przycisków oznaczonych A, B, C. Wciśnięcie przycisku oznacza wystawienie **1** na linię adresową (np. dla linii informacyjnej **$D_5$**, należy na linie adresowe wystawić **słowo 101**, co oznacza wciśnięcie przycisków C oraz A).

Należy zbadać zachowanie multipleksera, gdy adres jest podany błędnie lub gdy wejście zezwalające (STROBE) jest w stanie **1**. Wykazać prawidłowość działania multipleksera odnosząc się do jego tablicy prawdy zamieszczonej na **rys. 7.8**.




# 7. Badanie wskaźnika 7-segmentowego i dekodera sterującego

W ćwiczeniu 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.



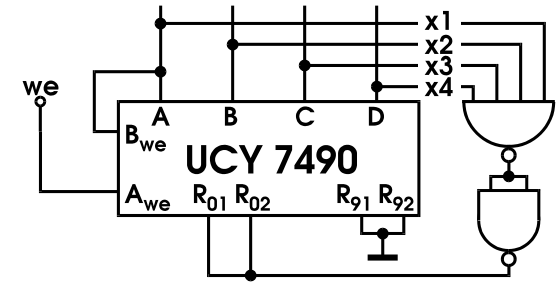

| badanie elem. wsk. / doł. elem. do liczn. cykl. | zadawanie sygnałów | | | | | |
|---|---|---|---|---|---|---|
| | A | B | C | D | RBI | LT |

Sprawdzić 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 zwierać do masy gniazd oznaczonych LT, RBI!***

Zamieścić uwagi odnośnie funkcjonowania wskaźnika cyfrowego z dekoderem sterującym.

# 8. Badanie licznika o programowanej długości cyklu z wyświetlaniem liczby impulsów

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




| dołączanie generatora do wejścia | programowanie długości cyklu | | | |
|---|---|---|---|---|
| | $X_1$ | $X_2$ | $X_3$ | $X_4$ |

**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. ***Dobrą praktyką 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, liczbę zliczonych impulsów. Jeżeli chcemy, aby po zliczeniu określonej liczby impulsów licznik zerował się, należy te wyjścia, które są wstanie **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 D**C**B**A** (0**1**0**1**). 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. Dobrać częstotliwość zliczanych impulsów (przez dobór **pojemności C** generatora oraz przez dobór **stopnia podziału** dzielnika) tak, aby można było swobodnie obserwować zmiany cyfr we wskaźniku.

Zamieścić 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. Ćwirko, 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. Omówić działanie tranzystora jako elementu przełączającego.

2. Omówić podstawowe funkcje logiczne i sposoby przedstawiania funkcji logicznych.

3. Omówić podstawowe bramki logiczne.

4. Przedstawić metodykę projektowania układów logicznych:

   - tablica prawdy

   - układanie równań boolowskich,

   - realizacja układowa AND, OR, NOT,

   - realizacja układowa NAND.

5. Omówić system liczbowy binarny i heksadecymalny oraz wzajemną konwersję liczb zapisanych w obu systemach.

6. Omówić sposób kodowania liczb dziesiętnych w kodzie BCD(8421).

7. Co oznaczają terminy: bit, bajt, słowo?

8. Co oznaczają skróty: LSB, MSB?

9. Omówić podstawowe parametry techniczne scalonych układów cyfrowych.

10. Omówić pracę bramki NAND TTL.

11. Omówić pracę inwertera CMOS.

12. Porównać układy TTL i CMOS.

13. Omówić przerzutnik SR.

14. Omówić przerzutnik JK.

15. Omówić budowę i działanie licznika asynchronicznego.

16. Wyjaśnić terminy: pojemność licznika, długość licznika, długość cyklu licznika, licznik modulo-S.

17. Omówić działanie multipleksera UCY 74151.

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