Programowalna matryca logiczna

1. Wprowadzenie

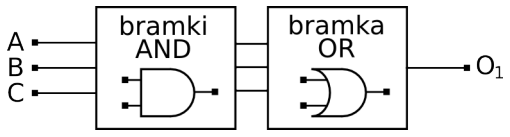
We współczesnej elektronice cyfrowej obecne są dwa trendy rozwoju[1]:

- Specjalizowane układy scalone (Application Specific Integrated Circuits - ASIC)– są to układy przeznaczone do określonych, ale zwykle bardzo wąskich zastosowań. Układy scalone tego typu projektowane są w celu wyprodukowania pewnej ilości egzemplarzy, których modyfikacja nie jest po produkcji możliwa. Zaletą układów ASIC jest niski koszt jednostkowy w przypadku produkowania dużych serii układów scalonych. Układy te są zoptymalizowane pod względem zajmowanej powierzchni krzemu i szybkości. Liczba elementów logicznych ograniczona jest do niezbędnego minimum, ścieżki łączące tranzystory są zoptymalizowane pod kątem minimalizacji poboru mocy.
- Programowalne układy logiczne (Programmable Logic Devices – PLD) – są to układy uniwersalne, które mogą zostać wykorzystane w bardzo szerokim spektrum aplikacji. Zbudowane są one z programowalnych bloków elementów logicznych (kombinacyjnych i sekwencyjnych) oraz konfigurowalnych ścieżek umożliwiających łączenie bloków logiki. Funkcjonalność tych układów określana jest przez projektanta na drodze programowania połączeń pomiędzy blokami elementów logicznych. Do zapamiętania swojej konfiguracji układy PLD używają pamięci typu SRAM (Static Access Random Memory), EEPROM (Electrically-Erasable Programmable Read-Only Memory) lub układów bezpieczników (układy antifuse-FPGA) konfigurowanych jednorazowo za pomocą przepalania bezpieczników łączących komórki elementów logicznych oraz ścieżki.

Współcześnie dostępne technologie stosowane do produkcji układów elektroniki cyfrowej typu ASIC umożliwiają budowanie układów taktowanych zegarami o częstotliwościach rzędu kilku gigaherców. Mimo tego, wiele aplikacji przemysłowych nie wymaga taktowania układów tak szybkimi zegarami i układy typu PLD w zupełności zaspokajają zapotrzebowanie inżynierów.
Proces projektowania układów elektronicznych opartych na układach PLD umożliwia wielokrotne reprogramowanie układów połączeń – jest to główna przewaga układów PLD nad układami ASIC. Dodatkowo układy reprogramowalne produkowane w dużych seriach cenowo zbliżają się do układów ASIC. Właśnie dlatego układy PLD zdobywają coraz większą popularność w praktyce inżynierskiej.

1.1.Budowa wewnętrzna układów konfigurowalnych

Najprostszą i najczęściej stosowaną strukturą PLD jest struktura złożona z matrycy bramek AND i bramki OR. przedstawiona na rys. 1.1.

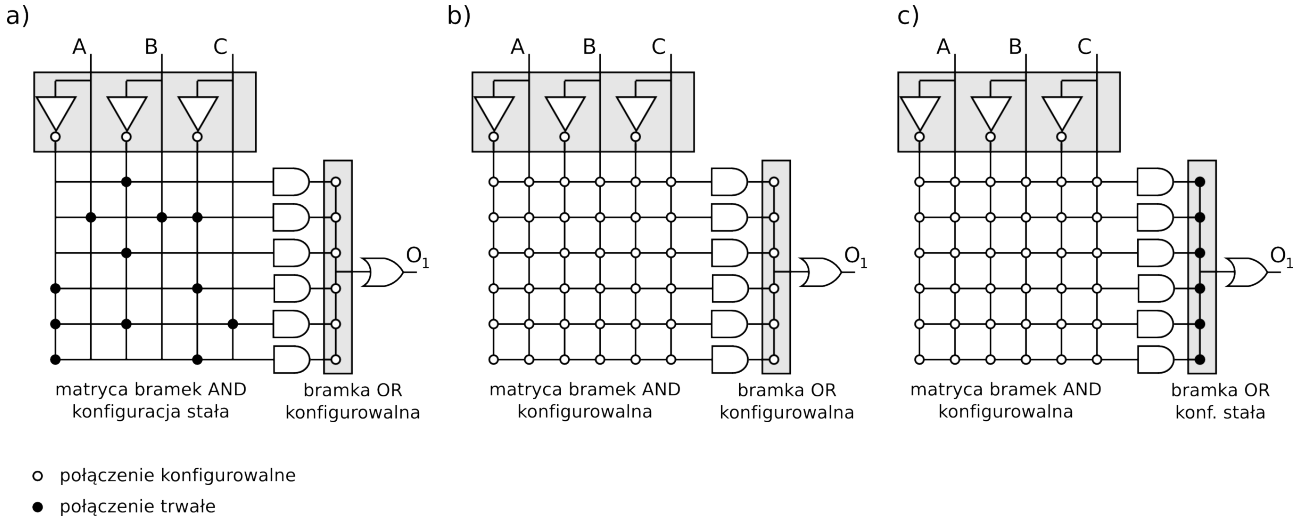


Rys. 1.1. Schemat blokowy programowalnej matrycy logicznej.

Wyróżniamy trzy typy matryc programowalnych PLD:

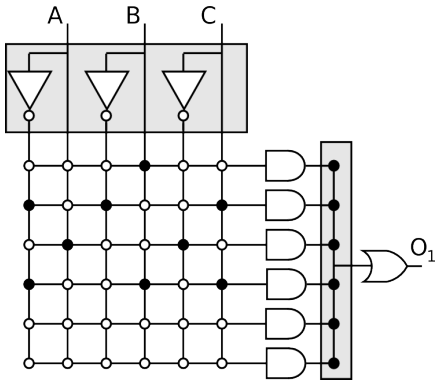
- PLA (Programmable Logic Array), w której konfigurowalna jest zarówno matryca bramek AND jak również bramek OR
- PAL (Programmable Array Logic), w której użytkownik ma wpływ tylko na połączenia w matrycy bramek AND
- ROM – w tym typie matrycy możliwa jest tylko konfiguracja wejść bramki OR.

Przykład realizacji układu przedstawionego na rys. 1.1. w architekturze ROM, PLA i PAL przedstawia rysunek 1.2.



Rys. 1.2. Realizacja praktyczna matrycy bramek PLD, a) architektura ROM, b) architektura PLA, c) architektura PAL

Układ inwerterów połączony z wejściami umożliwia uwzględnienie negacji sygnałów wejściowych w programowanych funkcjach logicznych. Kilka połączeń konfigurowalnych (oznaczonych na rys. 1.2 jako puste kropki), połączonych z pojedynczą bramką AND lub OR oznacza, że dana bramka posiada kilka konfigurowalnych wejść. Jedno połączenie konfigurowalne na schemacie oznacza jedno (możliwe do wyboru przez projektanta) wejście bramki AND lub OR.
W ćwiczeniu wykorzystywać będziemy tylko matrycę typu PAL, dlatego w dalszej części instrukcji nie będziemy się zajmować pozostałymi architekturami. Przykładowa matryca PAL posiada jedną szesciowejściową bramkę OR, jednak liczba wejść, jak również liczba bramek OR może być inna. Dokonując odpowiednich połączeń w macierzy bramek AND, programujemy dowolną funkcję będącą sumą iloczynów sygnałów wejściowych A, B C. Przykład połączeń w macierzy bramek przedstawia rys. 1.3.



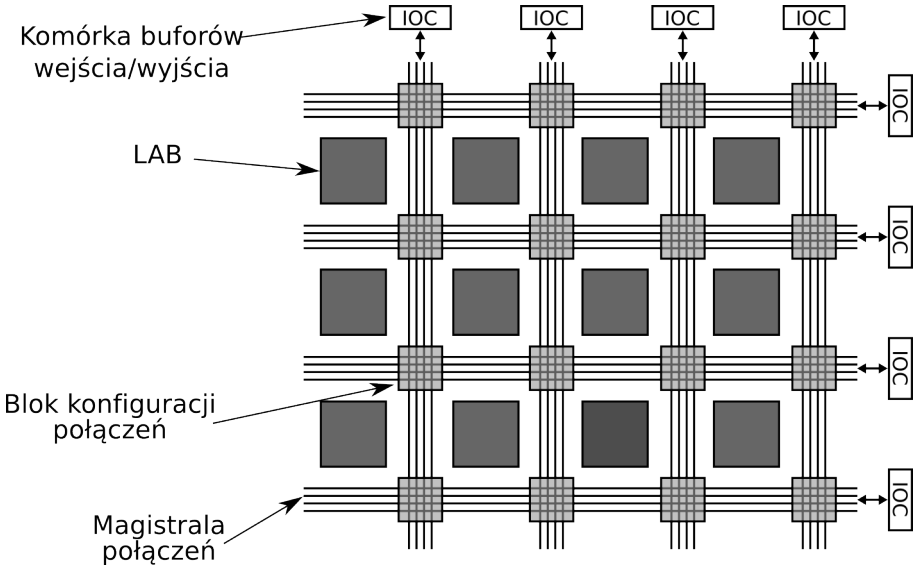
Rys. 1.3. Przykład połączeń matrycy bramek PAL.

W podanym przypadku funkcja logiczna na wyjściu bramki OR opisana jest wyrażeniem:

$$O_1 = B + \bar{A}\,\bar{B}\,C + A\,\bar{C} + \bar{A}\,B\,C$$

1.2 Zastosowania praktyczne programowalnych układów logicznych

Omówiona powyżej struktura PAL jest przykładem bloku funkcjonalnego, który wykorzystywany jest w bardziej złożonych układach programowalnych, takich jak układy FPGA (Field Programmable Gate Array). Układy FPGA są współcześnie stosowane w procesie projektowania praktycznie we wszystkich gałęziach elektroniki – właśnie dzięki możliwości wielokrotnego programowania. Przykład struktury układu FPGA przedstawia rys. 1.4.

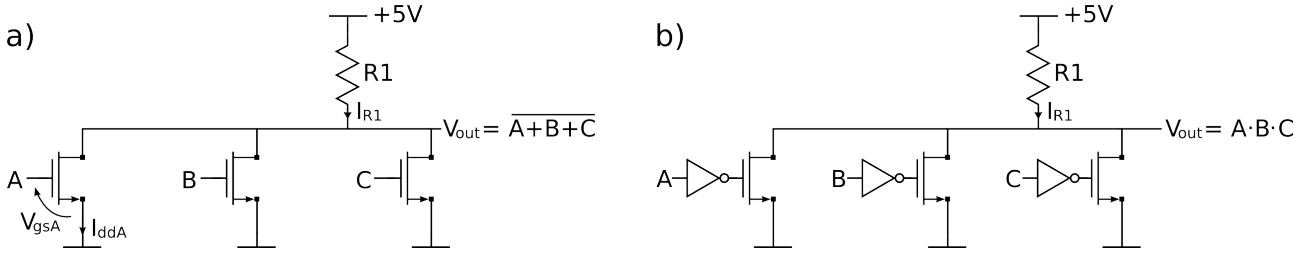


Rys. 1.4. Struktura wewnętrzna układu FPGA

Strukturę układu tworzy macierz niezależnych bloków logicznych LAB (Logic Array Block). Każdy blok logiczny zbudowany jest z pewnej liczby elementów logicznych, każdy element może realizować niezależną funkcję logiczną. Połączenia elektryczne pomiędzy blokami są realizowane za pomocą magistrali połączeń, których topologia jest konfigurowana w blokach konfiguracji połączeń. Magistrale biegną wzdłuż całego układu, dzięki czemu łączone mogą być ze sobą bloki w dowolnych lokalizacjach. Układy FPGA wyposażone są także w konfigurowalne bufory wejścia/wyjścia zwykle zawierające bramki trójstanowe. Umożliwia to używanie pojedynczej linii sygnałowej w zależności od potrzeb jako linii wejściowej, wyjściowej lub dwukieruknkowej.

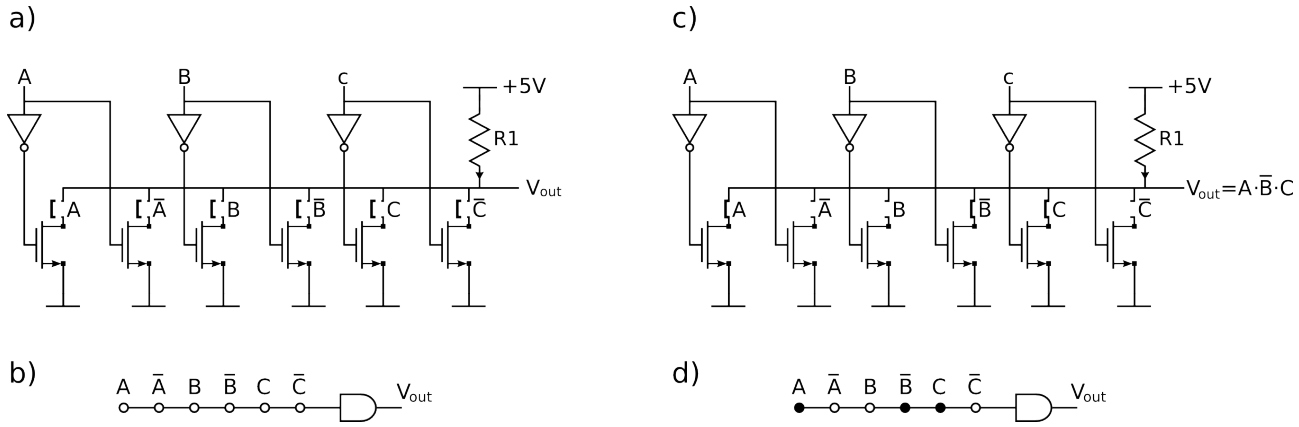
1.2. Iloczyn galwaniczny

Dla zrozumienia działania programowalnej zworkowo matrycy logicznej konieczne jest wprowadzenie pojęcia iloczynu galwanicznego (tzw. „bramka na drucie"). Przedstawione architektury cyfrowych układów konfigurowalnych zawierają w swojej strukturze bramki AND oraz bramki OR. Realizacja układowa wielowejściowej bramki wymaga zbudowania matrycy kluczy połączonych z wejściami bramki logicznej. Rozwiązanie takie wymaga zastosowania relatywnie dużej liczby tranzystorów oraz linii transmisyjnych, dlatego w praktyce w cyfrowych układach programowalnych stosuje się układ iloczynu galwanicznego.
Przykład realizacji bramek w postaci iloczynu galwanicznego przedstawia rysunek 1.5.



Rys. 1.5. 3-wejściowa bramka logiczna w architekturze iloczynu galwanicznego, a) bramka NOR, b) bramka AND

Wejścia logiczne stanowią bramki tranzystorów, połączonych równolegle, pracujących w układzie wspólnego źródła. W tej konfiguracji można traktować tranzystor NMOS jako sterowany klucz. Źródła tranzystorów połączone są z masą, dreny połączone są z rezystorem odciągającym R1 (tzw. rezystor pull-up). Jeżeli potencjał na bramkach (wejściach) tranzystorów wynosi ~0V, tranzystory pracują jako klucze rozwarte i prąd drenu nie płynie ($I_{ddA} = 0$), potencjał na wyjściu ustala się blisko górnego zasilania (logiczna „1"). Jeżeli bramka przynajmniej jednego tranzystora zostanie spolaryzowana napięciem +5V, tranzystor ten zaczyna pracować jak klucz zwarty i potencjał na wyjściu bramki ustali się blisko 0V (logiczne „0"). W ten sposób, jeżeli chociaż na jednym z wejść ustali się logiczna „1", na wyjściu układu otrzymamy logiczne „0". Układ ten działa zatem jak 3-wejściowa bramka NOR.
Jeżeli zanegujemy wejścia bramki NOR, otrzymamy bramkę AND (rys. 1.5.b.). Jeśli uzupełnimy bramkę AND w dodatkowe tranzystory oraz zworki, uzyskamy układ programowalnej bramki AND (rys. 1.6 a). Bramka taka pozwala na zaprogramowanie wybranego iloczynu z sygnałów $(A, \bar{A}, B, \bar{B}, C, \bar{C})$. Przykładowo, aby zaprogramować iloczyn $A * \bar{B} * C$ należy połączyć zworki w sposób pokazany na rysunku 1.6.c.



Rys. 1.6. a) Programowalna bramka AND – zworki rozwarte, b) zapis symboliczny, c) zaprogramowana zworkami funkcja logiczna $A * \bar{B} * C$ d) zapis symboliczny przykładowej konfiguracji bramki AND

1.3 Przykład realizacji funkcji logicznej – logika kombinacyjna

Używając metody tablic Karnaugha, można przedstawić przykład realizacji dowolnej funkcji logicznej zaimplementowanej w opisanej powyżej strukturze PAL. Rozważmy przykład 3-bitowego konwertera z kodu binarnego na kod Graya. Tabela 1. przedstawia rozważane kody wraz z reprezentacją dziesiętną.

Tabela 1. Liczby od 0 do 7 w kodzie binarnym i kodzie Gray'a

Liczba dziesiętna	Kod binarny C B A	Kod Gray'a $o_3\ o_2\ o_1$
0	0 0 0	0 0 0
1	0 0 1	0 0 1
2	0 1 0	0 1 1
3	0 1 1	0 1 0
4	1 0 0	1 1 0
5	1 0 1	1 1 1
6	1 1 0	1 0 1
7	1 1 1	1 0 0

Obliczmy funkcję logiczną dla najmłodszego bitu w kodzie Graya, czyli $\mathbf{o_1}$. Sporządzamy tablicę Karnaugha, rozdzielając zmienne wejściowe na grupy C oraz BA (rysunek 1.7).

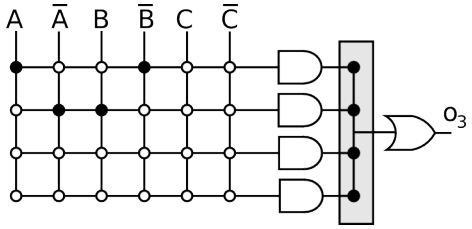
C \ BA	00	10	11	01
0	0	1	0	1
1	0	1	0	1

Rys. 1.7. Tablica Karnaugha dla najmłodszego bitu w 3-bitowym kodzie Graya.

Grupując wartości logiczne 1 w dwie grupy, możemy zapisać funkcję logiczną:

$$o_1 = A\,\bar{B} + \bar{A}\,B$$

Następnym krokiem jest wykonanie połączeń w matrycy bramki AND; połączenia te odpowiadają odpowiednio iloczynom $A\,\bar{B}$ oraz $\bar{A}\,B$, bramka OR odpowiada sumie logicznej w wyrażeniu na najmłodszy bit kodu Graya $\mathbf{o_1}$ (rysunek 1.8).

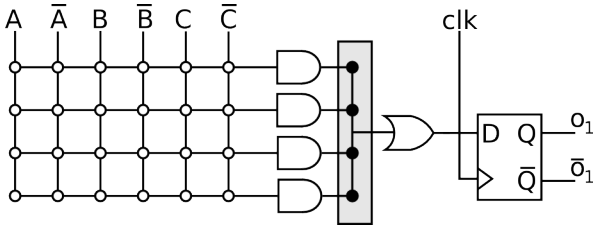


Rys. 1.8. Konfiguracja matrycy PLD dla funkcji logicznej najmłodszego bitu kodu Graya. Architektura PAL, wejścia bramki OR zwarte na stałe.

Analogicznie, sporządzając tablice prawdy dla pozostałych bitów $\mathbf{o_2}$ i $\mathbf{o_3}$, przy użyciu trzech matryc PLD otrzymalibyśmy pełny, 3-bitowy dekoder z kodu BCD na kod Graya.

1.5 Matryca PLD z rejestrem wyjściowym

Dla celów programowania układów logiki sekwencyjnej, struktury PLD mogą być wyposażone w rejestr (przerzutnik) na wyjściu. Przykład układu PLD z przerzutnikiem typu D przedstawia rysunek 1.9.



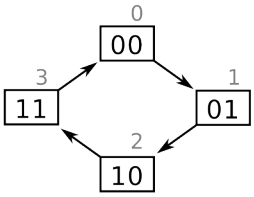
Rys. 1.9. Struktura PLD w architekturze PAL z rejestrem wyjściowym

Zastosowanie rejestru wyjściowego pozwala zapamiętać stan logiczny wyjścia matrycy bramek AND i OR. Stan logiczny zapamiętany w poprzednim cyklu zegara może również służyć jako sygnał wejściowy dla innego układu logiki kombinacyjnej lub sekwencyjnej.

1.6 Przykład realizacji funkcji logicznej – logika sekwencyjna

Przykładem zastosowania układu PLD z rejestrem wyjściowym jest licznik 2-bitowy w kodzie BCD. Licznik 2-bitowy możemy rozważać jako maszynę stanów o N=4 stanach (0,1,2,3, rysunek 1.10.a). Sporządzamy tabelę przejść przedstawiającą stan następny w funkcji stanu poprzedniego (rysunek 1.10.b).

a)

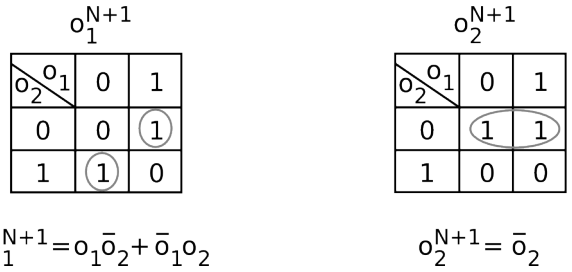


b)

Stan N o_1o_2	Stan N+1 o_1o_2
0 0	0 1
0 1	1 0
1 0	1 1
1 1	0 0

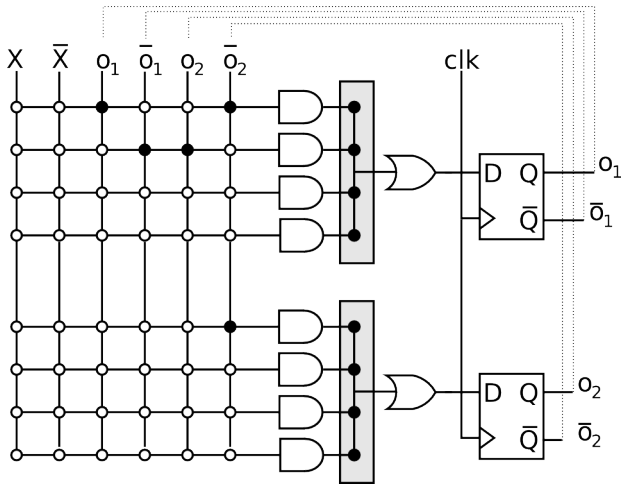
Rys. 1.10. a) Diagram przejść, b) tabela stanów licznika 2-bitowego

Używając metody tablic Karnaugha, wyznaczamy funkcje logiczne stanu następnego (N+1) w funkcji stanu poprzedniego dla wyjść o_1 i o_2 (rysunek 1.11).



Rys. 1.11. Tablice Karnaugha i odpowiadające im funkcje logiczne

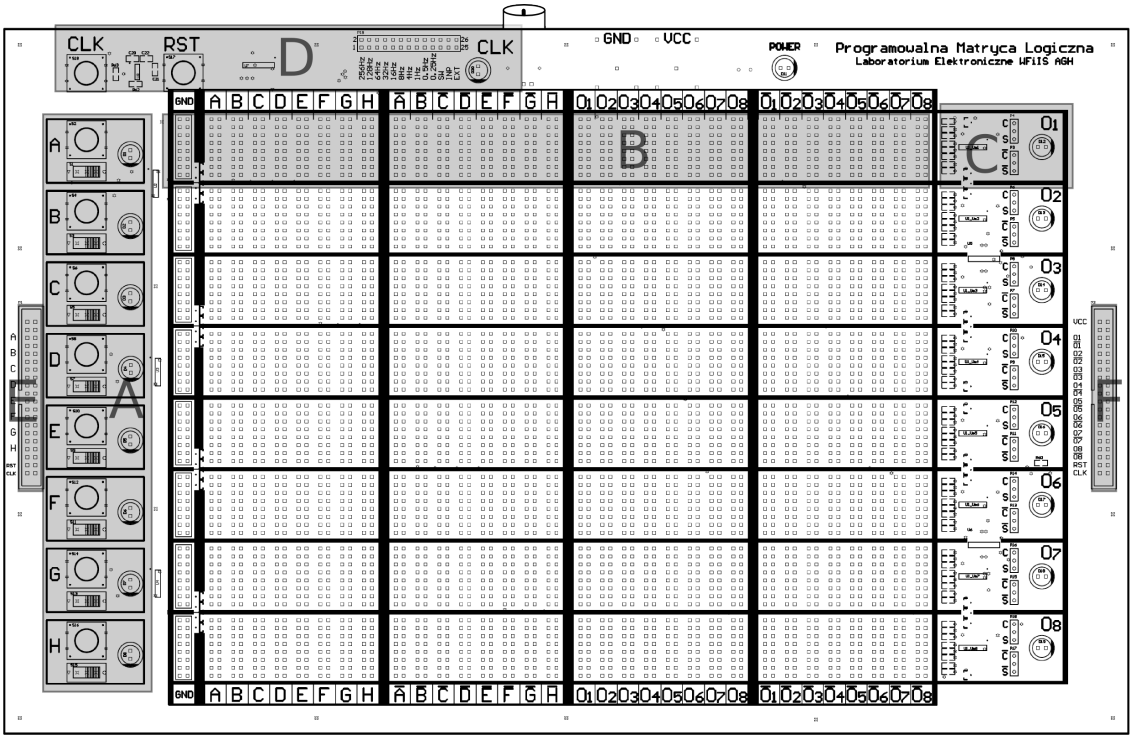
Jeżeli połączymy wyjścia przerzutników z wejściami matryc logicznych, możemy zaimplementować funkcje logiczne do układu sekwencyjnego złożonego z dwóch matryc PLD z rejestrami wyjściowymi (rysunek 1.12):



Rys. 1.12. Implementacja licznika 2-bitowego z użyciem matryc PLD

2. Budowa programowalnej matrycy logicznej wykorzystywanej w ćwiczeniu

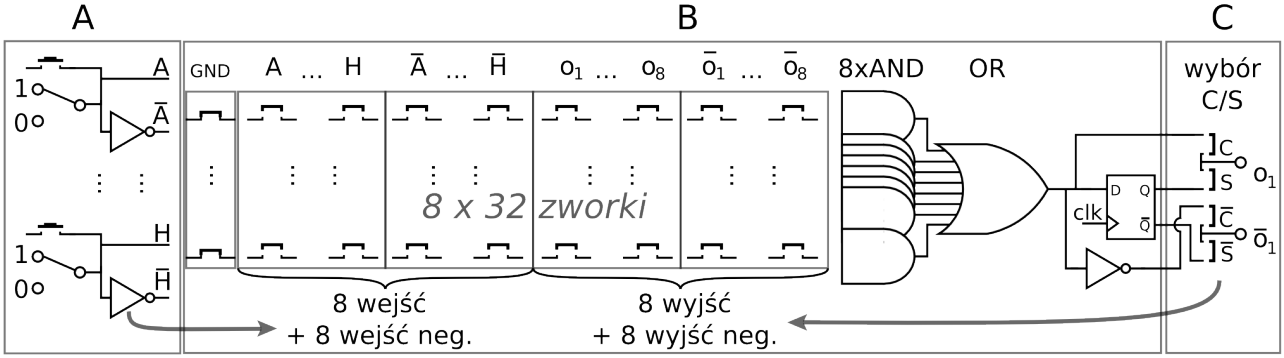
Matryca logiczna wykorzystywana w ćwiczeniu pozwala na programowanie układów logiki kombinacyjnej oraz sekwencyjnej za pomocą odpowiedniej kombinacji zworek. Budowa matrycy logicznej przedstawiona została na rysunku 2.1.



Rys. 2.1. Budowa programowalnej matrycy logicznej

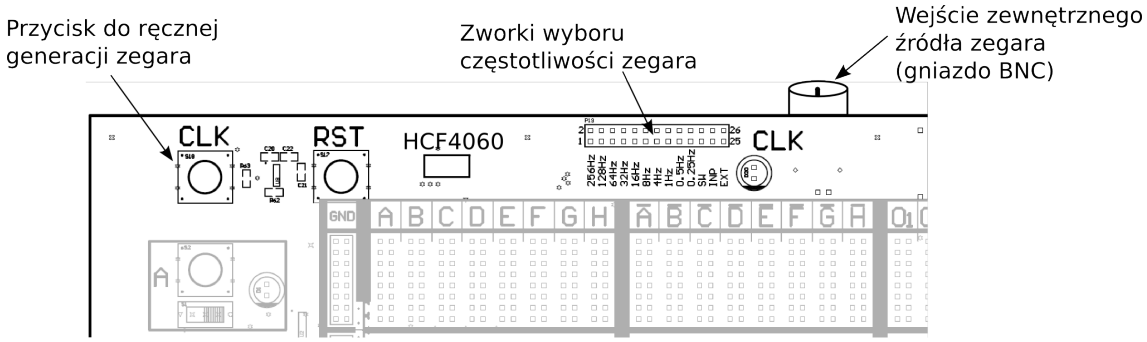
Sygnały wejściowe podawane są za pomocą przełączników (blok **A**, rys. 2.1) monostabilnych lub bistabilnych (dwupozycyjnych). Osiem sygnałów wejściowych oznaczonych zostało kolejnymi literami alfabetu od „A"do „H". Stan logiczny wejść sygnalizowany jest diodami luminescencyjnymi koloru żółtego, umiejscowionymi w bezpośrednim sąsiedztwie przełączników.

Głównym elementem funkcjonalnymi matrycy są programowalne bloki logiczne. Matryca posiada 8 bloków, pierwszy blok został zaznaczony na rys. 2.1 jako „**B**". Pojedynczy blok składa się z ośmiu 32-wejściowych bramek AND, 8-wejściowej bramki OR oraz przerzutnika typu D. Bramki AND zrealizowane zostały jako iloczyn galwaniczny (patrz punkt 1.2 instrukcji). Możliwe jest zbudowanie bramek AND czułych na sygnały wejściowe $A-H$, negacje sygnałów wejściowych $(\bar{A}-\bar{H})$, wyjścia bloków logicznych o_1-o_8 oraz negacje wyjść bloków logicznych $\bar{o}_1-\bar{o}_8$. Bramka OR zrealizowana została w postaci układu scalonego CD4078. Schemat programowalnego bloku logicznego (**B**) oraz zworek wyboru wyjścia sekwencyjnego lub kombinacyjnego (**C**) przedstawione zostały na rys. 2.2.



Rys. 2.2. Blok przełączników sygnałów wejściowych (A), programowalny blok logiczny (B), zworki wyboru wyjścia kombinacyjnego i sekwencyjnego (C).

Pojedynczy rząd pinów (poziomo) stanowi jedną bramkę AND. Sygnały, na które ma być czuła pojedyncza bramka AND wybiera się poprzez wpięcie zworki w odpowiednio oznaczonej kolumnie. Jeśli w danym bloku logicznym któraś bramka AND ma pozostać wyłączona, należy pamiętać, że wpięta powinna być w tym rzędzie zworka w kolumnie oznaczonej GND – spowoduje to wyłączenie bramki AND.
Zworki w bloku „**C**" pozwalają skonfigurować linie wejściowe $o_1 - o_8, \bar{o}_1 - \bar{o}_8$ w konfigurowalnych bramkach AND poprzez połączenie ich z wyjściami kombinacyjnymi (bezpośrednio z bramek OR) lub wyjściami sekwencyjnymi (rejestrowymi) wybranych bloków logicznych. W bloku **C** (zworki konfiguracji wyjść), „C" oznacza wyjścia kombinacyjne, „S" - wyjścia sekwencyjne. Stan logiczny wyjścia danego bloku sygnalizowany jest diodą koloru czerwonego, umiejscowioną obok zworek C/S.
Zastosowanie przerzutników na wyjściach bloków logicznych wymaga generatora sygnału zegarowego. Blok generatora sygnału zegarowego oznaczony jest na rys. 2.1. symbolem **D**. Stan logiczny sygnału zegarowego sygnalizowany jest diodą koloru niebieskiego, podpisaną „CLK". Widok szczegółowy bloku generacji zegara przedstawia rys. 2.3.



Rys. 2.3. Blok generacji sygnału zegarowego.

W programowanej zworkowo matrycy logicznej źródłem impulsów zegarowych mogą być:

1) przycisk CLK – wybór tego źródła dokonuje się poprzez zapięcie zworki oznaczonej jako „SW" w bloku generacji zegara,
2) scalony generator impulsów zegarowych HCF4060. Generator scalony pozwala na wybór częstotliwości zegara w zakresie 0.25 – 256Hz, poprzez zapięcie odpowiednio oznaczonej zworki (rys. 2.3).
3) Sygnał zewnętrzny – wybór tego źródła dokonuje się poprzez zapięcie zworki „EXT". Sygnał zegarowy pobierany jest z pinu „CLK" konektora **E** lub gniazda BNC znajdującego się po północnej stronie płytki .

W matrycy logicznej zastosowano przerzutniki z wejściem resetującym. W bloku generacji zegara przycisk „RST" pozwala zresetować wszystkie przerzutniki znajdujące się na płytce.

Programowalna matryca logiczna została wyposażona w dwa konektory do komunikacji z modułami zewnętrznymi. Sygnały wejściowe mogą być podawane z zewnątrz za pomocą konektora **E**, sygnały wyjściowe mogą być odczytywane poprzez konektor **F**.

3. Moduły rozszerzeń

Programowalna matryca logiczna wyposażona została w moduły rozszerzeń. Moduły te mają na celu ułatwienie weryfikacji zaprojektowanych układów logicznych (sekwencyjnych i kombinacyjnych) oraz obserwację praktycznego działania budowanych układów.

3.1 Wyświetlacz 7-mio segmentowy 4-ro pozycyjny

Wyświetlacz 7-mio segmentowy 4-ro pozycyjny jest modułem rozszerzającym programowalną matrycę logiczną w celu umożliwienia prezentacji wyniku działania układów logicznych. Moduł posiada 8 wejść, które za pomocą kabla taśmowego łączone są z wyjściami $o_1 - o_8$ matrycy logicznej. Wyświetlacz wyposażony jest w układ mikroprocesorowy, który umożliwia pracę w kilku trybach:

a) tryb bezpośredni – w tym trybie aktywny jest tylko pierwszy blok wyświetlacza - wejścia modułu podłączone są bezpośrednio do segmentów wyświetlacza na pierwszej pozycji, sposób połączeń oraz oznaczenia segmentów przedstawia rysunek 3.1.
b) tryb BIN-DEC – tryb ten służy do obserwacji liczby 8-mio bitowej liczby binarnej w formacie dziesiętnym (co odpowiada wartościom dziesiętnym 0-255). Po podłączeniu do matrycy programowalnej, wyjście o_1 odpowiada najmłodszemu, a o_8 – najstarszemu bitowi.
c) tryb 2x4BCD – mikrokontroler pracujacy w module wyświetlacza interpretuje bity $o_1 - o_8$ jako jedną liczbę binarną w kodzie BCD

Rys. 3.1. Schemat modułu wyświetlacza 4-ro pozycyjnego.

3.2 Przetwornik cyfrowo-analogowy R2R

Schemat przetwornika

(..to be written...)

4. Program ćwiczenia

4.1. Realizacja bramek logicznych

Korzystając z jednego wyjścia kombinacyjnego (np. O_1) zaprojektować w postaci sumy iloczynów następujące funkcje logiczne :

NOT, AND, NAND, OR, NOR, XOR, XNOR.

Funkcje te zaprogramować na matrycy logicznej, zweryfikować doświadczalnie poprawność połączeń.

Dla wybranej bramki logicznej zmierzyć czasy propagacji sygnału (przyjmując poziom 1.5V jako zmianę stanu) oraz czas narastania/opadania (10-90% amplitudy). Sygnał z generatora doprowadzić poprzez konektor wejściowy **E**, zmiany stanu logicznego na wyjściu zaobserwować podłączając oscyloskop do odpowiedniego wyjścia poprzez konektor wyjściowy **F**.

4.2. Logika kombinacyjna – dekoder kodu binarnego na wyświetlacz 7-mio segmentowy

Zbudować układ dekodujący liczby w kodzie binarnym na wyświetlacz 7-mio segmentowy prezentujący liczby w kodzie dziesiętnym. Należy użyć wejść A-D jako 4 bitów liczby binarnej oraz wyjść O_1 – O_7 jako sygnałów sterujących poszczególnymi segmentami wyświetlacza. Dla celów weryfikacji dekodera należy użyć modułu z 4-ro pozycyjnym wyświetlaczem 7-mio segmentowym, wykorzystywany będzie tylko pierwszy wyświetlacz. Moduł wyświetlacza należy ustawić w trybie pracy bezpośredniej (patrz pkt. 4.1. instrukcji).

W wersji podstawowej wyświetlacz powinien prezentować liczby dziesiętne od 0 do 9, odpowiednio dla binarnych 0000 – 1001. W wersji rozszerzonej można rozbudować funkcjonalność konwertera o wyświetlanie znaków kodu szesnastkowego, czyli oprócz powyższych cyfr, również litery od A do F dla wartości binarnych od 1010 do 1111. Schemat połączeń wyświetlacza przedstawiony został w pkt. 3.1 instrukcji.

4.3. Logika sekwencyjna - liczniki, maszyna stanów, detektor ciągu binarnego

a) Zaprojektować i zbudować przy pomocy programowalnej matrycy logicznej licznik o zadanej pojemności (np. modulo 9), liczący w kodzie binarnym. Efekt działania licznika zaobserwować, podłączając moduł wyświetlacza ustawiony w trybie

b)

4.4. Logika sekwencyjna – przetwornik cyfrowo-analogowy R2R, sterowanie silnikiem krokowym

5. Przygotowanie sprawozdania

Do przygotowania sprawozdania należy użyć zamieszczonych poniżej schematów zworek – w miejscach gdzie dokonywano zwarć należy zaznaczyć kratkę.

A) Realizacja bramek logicznych
Zaznaczyć na schemacie A (dołączonym na końcu instrukcji) konfiguracje zworek dla poszczególnych bramek. Podać czasy narastania i propagacji zmierzone na wyjściu wybranych bramek logicznych.

6. Literatura:

[1] http://www.xilinx.com/company/gettingstarted/fpgavsasic.htm#pcs
[2] Ming-Bo Lin: „Digital Systems Designs and Practices", John Wiley & Sons (Asia) Pte Ltd, Singapore 2008
[3] John F.Wakerly: „Digital Design, Principles and Practices", Pearson Education, wyd. 4, New Jersey, 2005
[4] U.Tietze, C.Schenk: „Układy półprzewodnikowe", Wydawnictwa Naukowo-Techniczne, Warszawa