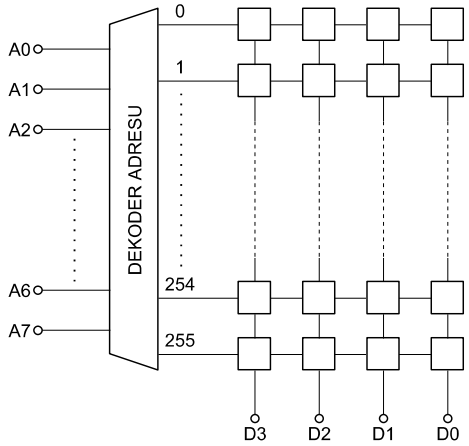
Programowalne układy logiczne (PLD)

Układy PLD – informacje ogólne

Pierwsze układy programowalne były pamięciami PROM. Pamięci ROM można używać do realizacji funkcji Boole'owskich jako generator funkcji logicznych.

Weźmy sobie pamięć ROM 256×4 (256 komórek 4 bitowych). Pamięć taką organizuje się poprzez 8 linii adresowych („wchodzące do pamięci") (2^8=256) oraz 4 linie danych („wychodzące z pamięci"):



Linie adresowe są liniami wejściowymi matrycy. Te linie są wejściami do dekodera adresu. Dekoder jest układem kombinacyjnym, który na wejściu ma liczbę podaną w naturalnym kodzie binarnym, a wyjście działa na zasadzie „1 z n" (pobudza jedno ze swoich wyprowadzeń). Każda z linii pobudza jedno słowo 4-bitowe.

Linie poszczególnych bitów (biegnące pionowo) są liniami wyjściowymi. Bity pojawiające się na nich tworzą słowo wyjściowe odczytywane z pamięci.

Idea pracy pamięci ROM polega na podaniu adresu, który w dekoderze zostanie przetworzone w kod „1 z n", pobudzając jedną z komórek pamięci, a zawartość komórek pojawia się na wyjściu pamięci.

Na linie wyjściowe można spojrzeć jako na 4 funkcje Boole'owskie 8 zmiennych. Zmienne funkcji podajemy na wejścia dekodera (wejścia adresowe pamięci), następnie odczytywany jest 1 bit odpowiadający danej kombinacji pobudzenia (8-bitowej kombinacji wejściowej) i ten bit jest wyprowadzany na któreś z wyjść.

Wyjście D_i jest funkcją sygnałów adresowych A0 – A7:

$$D_i = f(A0, A1, \ldots A6, A7)$$

Gdy programujemy pamięć, to wpisujemy w 256 komórek 1-bitowych zera i jedynki tak, jak występują one w tabeli prawdy.

Potem podajemy słowo wejściowe na wejścia adresowe, odczytywana jest wartość „0" lub „1" i pojawia się ona na konkretnym wyjściu. Programując pamięć ROM tabelą prawdy konkretnej funkcji uzyskujemy programowany generator funkcji Boole'owskiej.

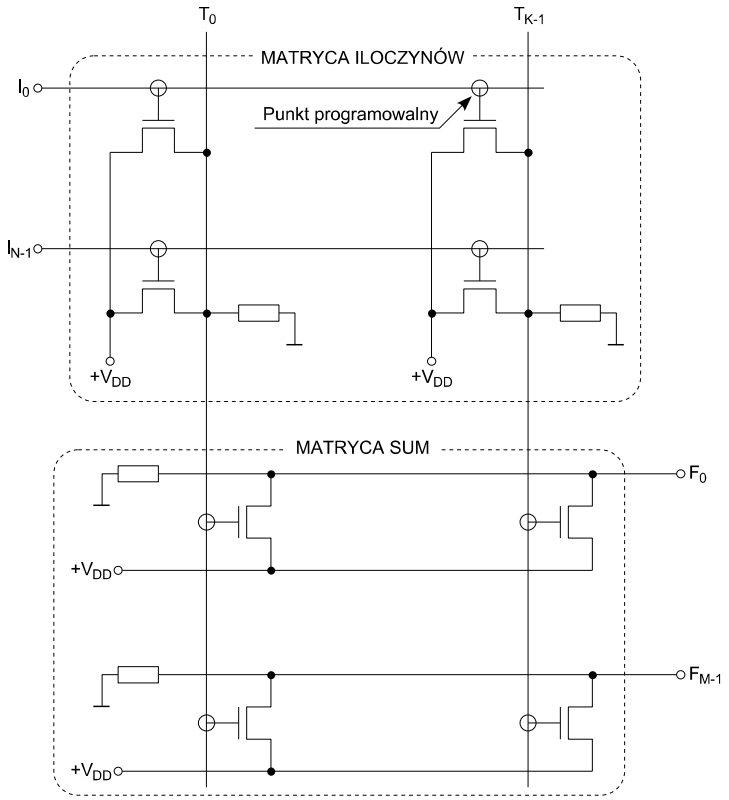
Pamięci ROM stworzone były by spełniać zadania pamięci stałej w systemach mikroprocesorowych, ale można było je wykorzystać jako pierwsze programowalne generatory funkcji.

Takie podejście ma wady. Mała elastyczność – na danym module pamięci zrealizujemy funkcje określonej liczby zmiennych (w naszym przypadku tylko 8 zmiennych). Jeżeli chcemy zrealizować funkcję 4 zmiennych (będzie miała 16 wartości), to na bity A0 – A3 podajemy słowo wejściowe, a na pozostałe „0". Blokuje to dostęp do pozostałych linii pamięci ROM, co zwyczajnie ogranicza pojemność wszystkich funkcji generowanych w danym module. Inną wadą jest wolna praca, szczególnie na początkach swojego istnienia (lata 70-te).

Matryca programowalna PLD

Niedługo po pamięciach pojawiły się prawdziwe układy programowalne. Oparte były na matrycy programowalnej PLD. Koncepcja matryc nie zmieniła się. Matryca pracuje na poziomie tranzystorów, dobrze się scala i można uzyskać duże gęstości upakowania. Dobrze przekłada się to na strukturę krzemową. Regularność i pow-

tarzalność położenia komórek, oraz połączeń ułatwia projektowanie struktury i pozwala na duże upakowanie.



Matryca składa się z dwóch części. Zaczynamy od wejść (*I*). *N* oznacza ilość wejść matrycy. Linie wejściowe krzyżują się z liniami termów (*T*). Termy są sygnałami wewnętrznymi matrycy. *K* oznacza ilość termów. Skrzyżowanie każdej linii wejściowej i linii termu opiera się na tranzystorze CMOS podłączonym między linią termu a linią napięcia zasilającego $+V_{DD}$. Ponadto każda linia termu jest poprzez rezystor połączona z masą. Bramka tranzystora jest zasilana przez linię wejściową poprzez punkt programowalny. Ten punkt może realizować połączenie, albo rozwarcie. W pierwszych rozwiązaniach punkty programowalne były realizowane w postaci bezpieczników, które przepalało się w procesie programowania. Można było w ten sposób zadecydować które z linii sygnałów wejściowych są dołączone do bramek tranzystorów.

W dolnej części matrycy koncepcja jest powielona z tą różnicą, że sygnałami wejściowymi są linie termów. Linie wyjściowe (*F*) wychodzą na zewnątrz układu. *M* określa liczbę wyjść układu. Linia termu przez punkt programowalny steruje bramką tranzystora MOS. Tranzystor zapięty jest pomiędzy linią wyjściową a linią zasilającą $+V_{DD}$. Linia wyjściowa jest połączona poprzez rezystor z masą.

W analizie pracy tego układu zakładamy, że tranzystory MOS zastosowane w matrycy otwierają się „zerem" logicznym. Tranzystory są normalnie rozwarte. Do zastosowania w modelu nadają się najlepiej tranzystory MOS z kanałem typu P (PMOS) normalnie wyłączone. W praktyce stosuje się tranzystory z kanałem typu N z powodu większej ruchliwości elektronów.

Jak działa matryca? Stan linii termu jest ustalany w górnej części matrycy. Wybrane (podczas procesu programowania) przez nas sygnały wejściowe *I* są dołączone do tranzystorów. Jeżeli na danej linii *I* znalazło się „zero" logiczne, to tranzystor otwiera się i linia termu zwiera się z linią zasilającą $+V_{DD}$ dostarczając na linię termu „jedynkę" logiczną.

Jeśli na wszystkich liniach wejściowych będą „jedynki", to żaden z tranzystorów się nie otworzy. Rezystor będzie ściągał napięcie termów do wartości zerowej.

Wartość *i*-tego termu będzie wynosiła „zero" logiczne wtedy i tylko wtedy gdy na wszystkich dołączonych wejściach będzie „jedynka" logiczna. Jeżeli na którymkolwiek z wejść pojawi się „zero" logiczne, nastąpi otwarcie tranzystora (zwarcie linii termu z linią zasilającą $+V_{DD}$) co powoduje ustawienie się „jedynki" logicznej na linii termu.

Rezystor ma na celu zabezpieczyć układ przed zwarciem między linią zasilającą a masą układu. Jego wartość powinna być znacznie większa od rezystancji kanału otwartego tranzystora.

Górna oraz dolna część matrycy realizują funkcję NAND. Dla górnej części matrycy realizowana jest funkcja:

$$T_X = NAND(\alpha_{X,0} + I_0; \alpha_{X,1} + I_1; \ldots; \alpha_{X,N-1} + I_{N-1})$$

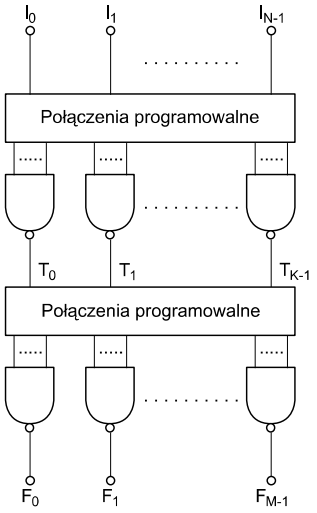
$$\alpha_{ij} = \begin{cases} 1 - \text{gdy punkt progr. rozwarty} \\ 0 - \text{gdy punkt progr. Zwarty} \end{cases}$$

Dla części dolnej:

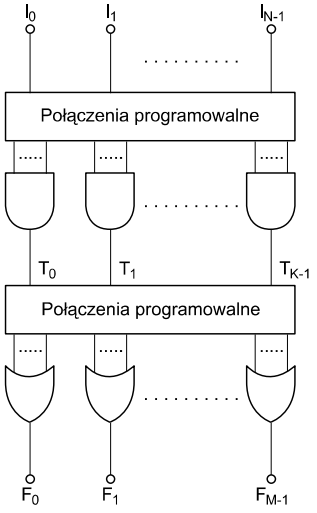
$$F_X = NAND(\beta_{X,0} + I_0; \beta_{X,1} + I_1; \ldots; \beta_{X,K-1} + I_{K-1})$$

$$\beta_{ij} = \begin{cases} 1 - \text{gdy punkt progr. rozwarty} \\ 0 - \text{gdy punkt progr. zwarty} \end{cases}$$

Nie rysuje się schematów układów PLD na poziomie tranzystorów. Można zastosować uproszczenie:



Z praw de Morgana dwuwarstwową strukturę NAND'ów można zastąpić strukturą AND oraz OR:



Z tego to powodu górną matrycę układu nazywamy matrycą iloczynów a dolną – matrycą sum.