Układy PLD, część 1

W serii artykułów chcielibyśmy się zająć omówieniem właściwości i możliwości oferowanych przez coraz bardziej popularne na świecie układy PLD (Programmable Logic Devices).

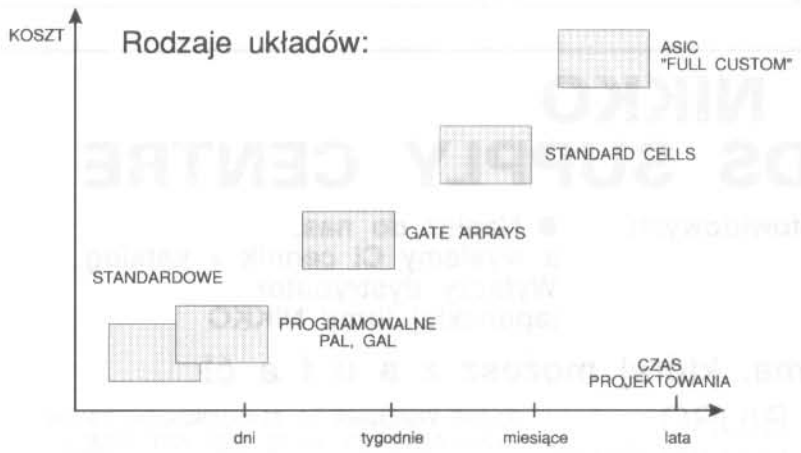
Dlaczego zajęliśmy się tym tematem? Otóż coraz bardziej popularne są różnego rodzaju odmiany układów specjalizowanych (ASIC - application specific integrated circuits) - wynika to z konieczności specjalizacji wykonywanych urządzeń w wąskich dziedzinach. Często też spotykamy się z koniecznością realizacji funkcji (często bardzo prostej), której odpowiednika nie ma w standardowych bibliotekach układów TTL lub CMOS, a wykonanie jej odpowiednika związane jest ze znacznym wzrostem komplikacji układu - co utrudnia zarówno wykonanie jak i testowanie urządzenia. Jeden układ scalony jest bardziej niezawodny od kilkunastu „kości" montowanych na dość drogich, często wielowarstwowych obwodach drukowanych, zużywa ponadto o wiele mniej energii. Bogate i stosunkowo tanie oprogramowanie doskonale wspiera działania konstruktora, dzięki czemu „wyprodukowanie" układu odpowiadającego naszym oczekiwaniom sprowadza się tylko do precyzyjnego zdefiniowania problemu, który chcemy rozwiązać.

Dodatkową zaletą układów PLD jest możliwość testowania i analizy pracy zaprojektowanego przez nas urządzenia „na sucho" - dzięki symulatorom dołączanym do pakietów oprogramowania. Postępowanie takie zapobiega wielokrotnym próbom i poprawianiu ewentualnych błędów w gotowym urządzeniu.

Największym problemem z jakim konstruktor musi się uporać przy projektowaniu omawianą przez nas techniką jest dostęp do programatorów układów PAL i GAL. Są to urządzenia dość drogie - dlatego redakcja EP zakupiła wysokiej klasy programator firmy Hi-Lo, typu ALL-03A (test programatora w jednym z najbliższych numerów EP), który będzie mógł być wykorzystany przez naszych Czytelników do realizacji własnych projektów. Uzupełnieniem tej oferty są programy kompilacyjne serii CUPL (patrz artykuł na str. 19 w tym numerze EP), w różnych wersjach - od najprostszych - z minimalnymi wymaganiami do komputera, na którym program będzie uruchamiany, lecz w pełni funkcjonalnych i bardzo tanich - po wersje profesjonalne, o dużych wymaganiach sprzętowych (komputer co najmniej 386DX) i znacznej cenie.

Podsumowując, dzięki zastosowaniu układów PLD

każdy z nas może zaprojektować i wykonać własny - oryginalny cyfrowy układ scalony



Rys. 1. Koszt i czas opracowania projektu dla różnych rodzajów układów

Jak wcześniej wspomniano, układy PLD są jedną z odmian techniki ASIC, najtańszą z nich, dzięki temu dostępną także w zastosowaniach nieprofesjonalnych.

Do rodziny układów PLD należą od dawna znane pamięci PROM, nowsze rozwiązania - układy PLA (Programmable Logic Array), PAL (Programmable Array Logic), PLS (Programmable Logic Sequencer). Kontynuacją tych rozwiązań są serie układów PGA (Programmable Gate Array), LCA (Logic Cell Array) oraz GAL (Generic Array Logic). Do tego „jednym tchem" przekazanego wykazu osiągnięć serii PLD należy dodać układy PEEL oraz MAX, są to jednak daleko zaawansowane rozwiązania, jak na razie pozostające poza zasięgiem amatorów i to nie tylko u nas w kraju. W dalszej części artykułu omówione zostaną tylko ich podstawowe właściwości.

Na **rys. 1** jest przedstawiona zależność czasu i kosztów opracowania urządzeń z zastosowaniem układów standardowych, PLD, układów Gate Arrays, Standard Cells oraz ASIC (wykres oparto na danych Lattice Semiconductor Corp. zawartych w „GAL Handbook" z 1988 r.). Warto dodać, że techniki wymagające dłuższych czasów i większych kosztów projektowania są opłacalne przy coraz dłuższych seriach produkcyjnych, gdy udział kosztów opracowania w cenie jednostkowej układu staje się znikomo mały.

Ze względu na praktyczne zalety, tj. dostępność i względnie niską cenę, szczegółowo omówione zostaną dwie rodziny układów o podobnych właściwościach:

- układy PAL wraz z programem kompilatora równań logicznych PALASM (nazwy zastrzeżone przez Advanced Micro Devices);
- układy GAL wraz z programem kompilacyjnym CUPL.

Autor nie zamierza przeprowadzić „suchego" wykładu na temat budowy wewnętrznej, metod minimalizacji funkcji logicznych, itp. i dla-

tego przedstawione zostaną praktyczne, bardzo proste rozwiązania z zastosowaniem układów PAL i GAL.

Nieco pożytecznej teorii

Teoria układów cyfrowych, a więc i PLD opiera się na algebrze dwójkowej - Boole'a - od nazwiska Georga Boole'a, angielskiego matematyka, który w 1854 roku publikacją „An Investigation of the Laws of Thought" dał podwaliny pod rozwój współczesnej logiki. Z teorii Boole'a wynika iż każdą, nawet najbardziej złożoną funkcję logiczną można przedstawić za pomocą trzech podstawowych operatorów:

- operatora sumy logicznej - jego odpowiednikiem w technice cyfrowej jest funktor OR;
- operatora iloczynu logicznego - jego odpowiednikiem jest funktor AND;
- operatora negacji - odpowiednikiem jest funktor NOT (inwerter).

Symbole graficzne wraz z tablicami prawdy przedstawia **rys. 2.**

Dopełnieniem teorii Boole'a były cztery postulaty opublikowane w 1904 roku przez E. Huntingtona (w publikacji „Sets of Independent Postulates for the Algebra Logic"), które stworzyły podstawy do tworzenia teorii działań algebraicznych opartych na teorii logiki.

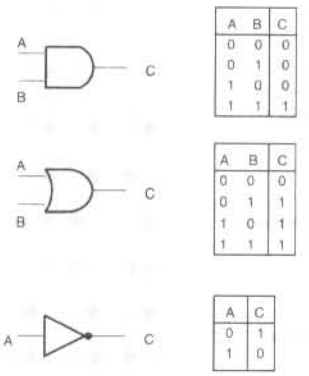
Można je wyrazić w sposób następujący:

- dla każdego działania istnieje element obojętny, tzw. moduł:
dla sumy jest to zero: x+0=x;
dla iloczynu jest to jeden: x·1=x;
- zarówno sumowanie, jak i iloczyn są przemienne:
x+y=y+x, x·y=y·x;
- działania są rozdzielne względem siebie:
(x+y)·z=(x·z)+(y·z), x+(y·z)=(x+y)·(x+z)
- warunki dopełnienia:
x+/x=1, x·/x=0

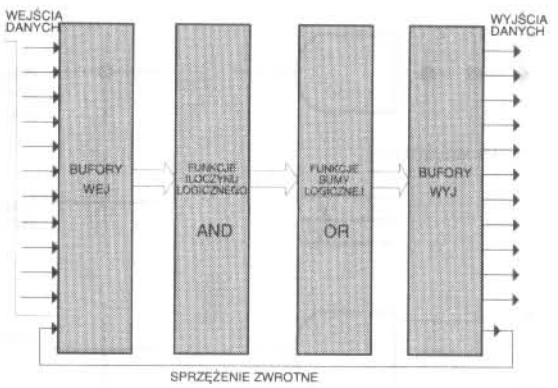
Cały czas należy pamiętać iż prawa te dotyczą liczb binarnych!

Ostatnim elementem teorii podstawowej algebry Boole'a jest zbiór praw, dotyczących działań:

- idempotentność dodawania i mnożenia:
x·x=x (dla iloczynu);
x+x=x (dla sumy);
- właściwości charakterystyczne działań:
x·0=0, x+0=x;
x+1=1, x·1=x;
/0=1,
/1=0;



Rys. 2. Symbole i tablice prawdy trzech funktorów elementarnych



Rys. 3. Schemat blokowy układu PLD

- absorpcja jednej zmiennej:
x·(x+y)=x,
x+(x·y)=x
- łączność dodawania i mnożenia:
x+(y+z)=(x+y)+z;
x·(y·z)=(x·y)·z;
- jeżeli występują następujące warunki:
x·y=y oraz x+y=y,
to prawdą jest iż:
x=y;
- prawo podwójnego zaprzeczenia:
//x=x
- i dwa prawa DeMorgan'a:
/(x+y)=/x·/y
/(x·y)=/x+/y

Symbol „·" oznacza iloczyn logiczny, „/" - negację (zaprzeczenie), natomiast „+" - sumę logiczną.

Należy pamiętać, iż w notacji stosowanej w algebrze Boole'a obowiązuje identyczna kolejność wykonywania działań (poziomy nawiasów itp.), jak w „zwykłej" algebrze.

Dla porządku należałoby jeszcze zasygnalizować istnienie tzw. tablic Karnaugh'a, rzeczy bardzo istotnej, zwłaszcza przy minimalizowaniu funkcji logicznych, lecz nie będziemy tutaj poświęcać miejsca na omówienie metod tworzenia i „obróbki" tych tablic i dlatego dla osób pragnących zgłębić teorię zamieszczamy na końcu artykułu indeks książek, które zawierają pełną informację na ten temat.

Wyjaśnienia może wymagać jeszcze jeden termin , bardzo popularny w literaturze dotyczącej układów PLD. „Term" - kojarzy się z czymś ciepłym, a oznacza po prostu elementarne wyrażenie logiczne (z ang. term - wyrażenie), opisane równaniem. Zwrot „termy wejściowe" (spolszczenie jest może niezbyt szczęśliwe) oznacza składniki funkcji wyjściowej realizowane przez połączenia zaprogramowane w matrycy połączeń (wejściowej) przez użytkownika.

Być może, a nawet na pewno cała ta dawka teorii działa zniechęcająco, ale zawarte w niej jest minimum informacji koniecznej do zrozumienia dalszych wywodów, bliższych praktyce i lutownicy.

Pewnym podsumowaniem tej przykrej dawki „wzorków" niech będzie następujące twierdzenie:

do zrealizowania dowolnej funkcji logicznej wystarczą trzy funktory: AND, OR, NOT.

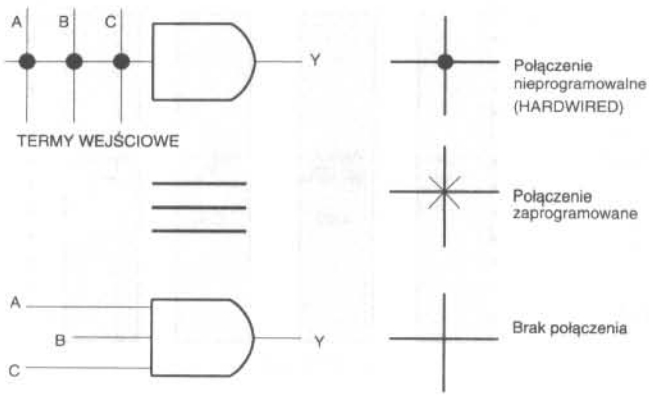
Innymi słowy. każdą funkcję można „złożyć" z większej lub mniejszej ilości elementów podstawowych (podobnie jest z kolorami!) odpowiednio połączonych.

Architektura układów PLD

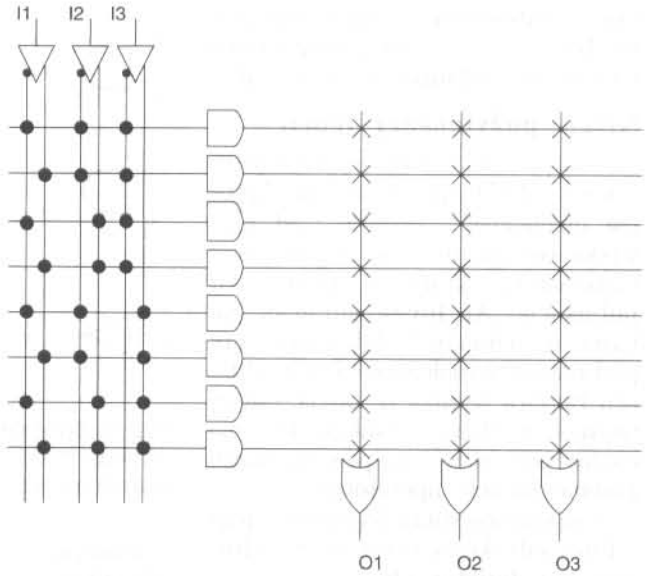
Na **rys. 3** przedstawiono schemat blokowy układu PLD, prawdziwy zarówno dla PROM, PLA, PAL, GAL. Jak można zauważyć, konstruktorzy układów PLD skrzętnie wykorzystali wniosek z wywodów matematyków i w strukturze logicznej PLD nie zastosowali innych funktorów niż podstawowe.

Na **rys. 4** podano wyjaśnienie symboliki stosowanej na wszystkich schematach.

Na **rys. 5** jest przedstawiony schemat wewnętrznej struktury logicznej układów typu PROM. Jak widać, bufory-inwertery wejściowe mają wyjścia na stałe połączone ze ściśle określonymi wejściami bramek AND (jest to tzw. Fixed Array - tablica stała, której zawartość ustala producent). Układ ten tworzy dekoder dokonujący selekcji aktywnego fragmentu matrycy (tablicy)



Rys. 4. Symbole stosowane w opisie struktury logicznej układów PLD

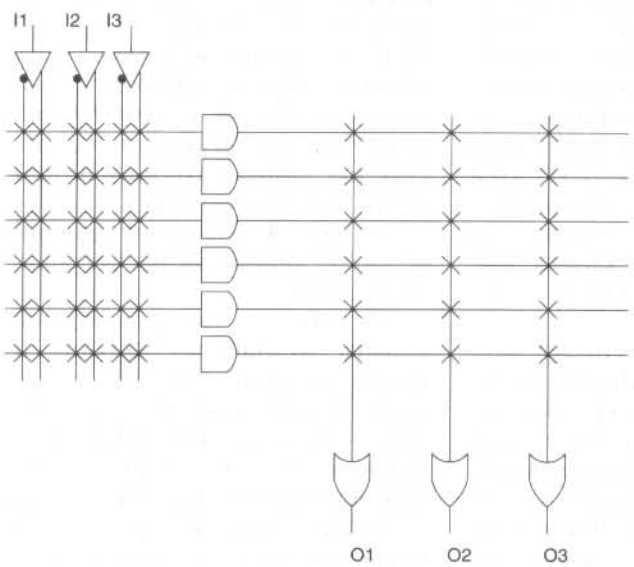


Rys. 5. Schemat struktury logicznej układów PROM

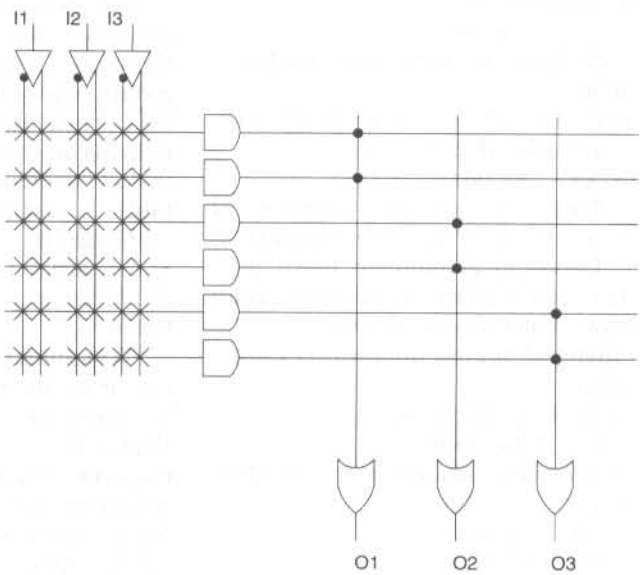
programowalnej (ang. Programmable Array). Za pomocą krzyżyków symbolicznie zaznaczno bezpieczniki (ang. fuses), których przepalenie powoduje programowanie pamięci - tu ważna uwaga - funkcję wyjściową określają połączenia, które nie zostały przepalone. W pewnej „opozycji" do techniki przepaleń jest tzw. metoda „anty-fuses". Programowanie komórki odbywa się poprzez połączenie wewnątrz struktury układu dwóch warstw przewodzących, oddzielonych od siebie pierwotnie warstwą izolacyjną. Technika taka jest bardzo rzadko stosowana w układach PROM, o wiele częściej w nowszych opracowaniach układów PLD.

Tak więc w układach klasy PROM tylko matryca OR jest programowalna. Możliwości realizowania złożonych funkcji logicznych są nieco ograniczone w tego typu układach, ponadto każda z kombinacji sygnałów wejściowych I0..In daje jedną charakterystyczną (i zaprogramowaną) odpowiedź na wyjściu, ale wymaga matrycy o rozmiarach 2^n, co czyni takie rozwiązanie stosunkowo drogim. Drugą bardzo istotną wadą jest powolność pracy tak dużych matryc (zwiększone czasy propagacji wynikają z fizycznych rozmiarów struktury układu). Tak zaprojektowana matryca wejściowa nazywana jest „fully decoded array".

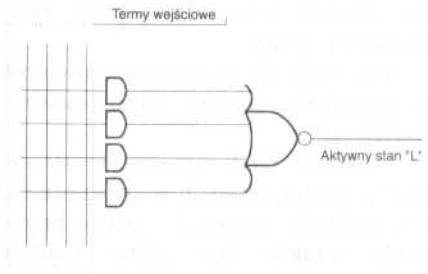
Nieco większe możliwości programowania oferuje struktura typu PLA (na **rys. 6**). Jak widać, zarówno matryca wejściowa (wejścia bramek AND) jak i wyjściowa (wejścia bramek OR) jest programowalna. Ma ona przewagę nad strukturą typu PROM, głównie ze względu na szybkość pracy - ponieważ bramki AND nie tworzą dekodera typu „1 z N", czyli nie muszą zdekodować 2^n wierszy matrycy, a tylko wybrane i niezbędne do prawidłowego przetworzenia informacji zawartej w słowie wejściowym. Dzięki temu unika się sytuacji, w których 90% matrycy nie jest wykorzystane. Dla przykładu - chcemy zaprojektować dekoder adresowy lub inny układ logiczny, na którego wejście dostarczamy 10-bitową informację (z szyny adresowej komputera), a na wyjściu otrzymujemy tylko jeden sygnał selekcji dla wybranego adresu. W układzie typu PROM z matrycy o ilości 2^{10}=1024 komórek wykorzystujemy tylko jedną! W układach PLA wskaźnik ten jest zdecy-



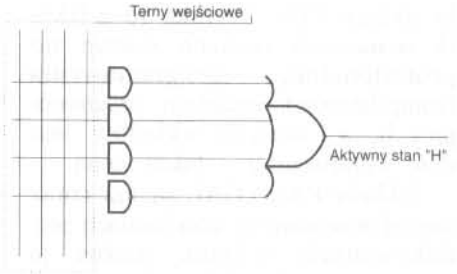
Rys. 6. Schemat struktury logicznej układów PLA



Rys. 7. Schemat struktury logicznej układów PAL



Rys. 8. Wyjście typu L



Rys. 9. Wyjście typu H

dowanie korzystniejszy, wyraźniej jest to widoczne w rozwiązaniach nieco mniej ekstremalnych od podanego przykładu.

Jeszcze inne podejście, wynikające z dokładnej analizy układów cyfrowych, zostało zastosowane w układach PAL. Schemat struktury wewnętrznej tego typu układu przedstawia **rys. 7**. Jak widać, matryca wejściowa jest w pełni programowalna, natomiast wyjściowa jest programowana przez producenta układu. Takie rozwiązanie zapewnia największą szybkość pracy i największą elastyczność budowania funkcji logicznych. Praktycznie każde równanie logiczne można zapisać jako sumę czterech lub pięciu „product-term'ów" (zaprogramowanych iloczynów logicznych), a w najprostszych układach PAL (np. PAL16L/H/P8) możliwe jest wykonanie sumowania siedmiu lub ośmiu iloczynów.

Ponieważ zamierzamy poświęcić więcej czasu układom PAL, warto omówić rodzaje wyjść stosowane w tego typu układach. Od typu wyjścia zależy typ układu (np. w serii PAL16x8, gdzie x=L, C, H, P, R, V).

Na **rys. 8** przedstawione zostało wyjście typu L (aktywne w stanie niskim), na **rys. 9** wyjście typu H (aktywne w stanie wysokim), wyjście typu R (rejestrowe) na **rys. 10**. Wyjście typu C (Complementar) jest wyjściem dualnym - typu L i H jednocześnie, oczywiście na różnych nóżkach. Wyjście typu P (Programmable) ma dzięki wewnętrznej bramce Ex-OR programowaną (poprzez przepalenie komórki bitu sterującego) polaryzację.

Oprócz stanu logicznego wyróżniającego aktywne wyjście istnieje jeszcze podział na tzw. wyjścia lub wejścia dedykowane i dwukierunkowe. Przykłady wyjść dedykowanych przedstawiają rysunki 8, 9, 10. Jak widać, tego typu wyjście jest wyłącznie jednokierunkowe, w odróżnieniu od wyjścia z **rys. 11**. które w zależności od ustawienia odpowiedniego bitu (sterującego stanem sygnału OE) mogą pracować jako wejście lub wyjście.

Budowa wewnętrzna układów PAL zmusza projektanta do odpowiedniego konstruowania równań logicznych, opisujących funkcję spełnianą przez układ (chociaż często tę pracę potrafi wykonać program kompilujący).

Istnieją dwie formy kanoniczne równań:

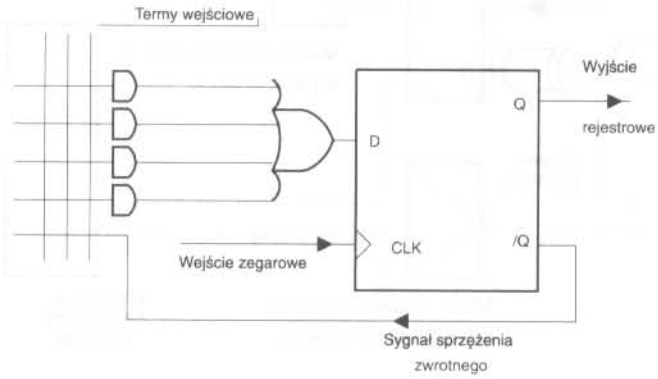
- równanie typu POS (Product of Sum) - czyli iloczyn sum logicznych;

- równanie typu SOP (Sum of Product) - suma iloczynów logicznych.

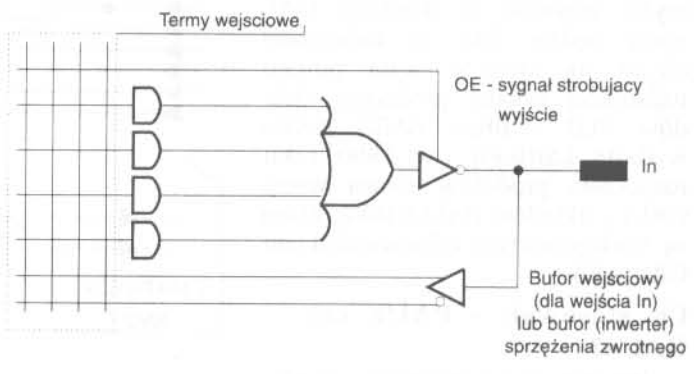
Druga forma (SOP) jest dogodniejsza dla logicznej implementacji w układach PAL, jednakże większość programów kompilujących potrafi samoczynnie przekształcić i zminimalizować podane równania do możliwie najprostszej postaci.

Jak więc wynika z tego krótkiego omówienia, programowanie układów PLD polega na odpowiednim przepalaniu wewnętrznych bezpieczników (są to subminiaturowe połączenia wykonane z platyny - przybliżony wygląd przedstawia **rys. 12**), zatem układ raz zaprogramowany nie daje możliwości zmiany realizowanych funkcji.

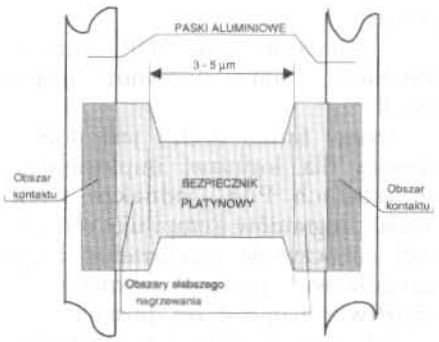
Tę niedogodność usunęła najpierw firma Altera, opracowując serię układów EPLD (Erasable PLD) typu EP310 i pochodne (EP320, 600), której programowanie wyglądało podobnie jak pamięci EPROM. Firma Cypress produkuje nieco mniej „ekstrawaganckie", bo pochodne standardu P16x8 i P22x8, układy reprogramowalne EPLD. Zamiast niszczonych bezpowrotnie bezpieczników zastosowano komórki z tranzystorami MOSFET, a ładunek zgromadzony pod bramką tranzystora w izolowanym kondensatorze zapewnia długotrwałą pracę zaprogramowanego układu. Kasowanie zawartości takiego układu odbywa się poprzez naświetlanie promieniowaniem ultrafioletowym o długości fali 2357A. Koncepcja prosta i w zasadzie dobra.... Ale dalej zaszła firma Lattice. Około 1982 roku wyprodukowała pierwszą partię układów EEPLD (Electrically Erasable PLD) - elektrycznie kasowanych



Rys. 10. Wyjście typu R



Rys. 11. Wyjście dwukierunkowe



Rys. 12. Połączenie platynowe - przepalany bezpiecznik

i programowanych, odpowiedników funkcjonalnych serii P16H/L8. Dalsze prace badawcze zaowocowały wielkim (zdaniem nie tylko autora) hitem - układami GAL (Generic Array Logic).

Przewaga tej rodziny układów nad opracowaną przez AMD serią popularnych PAL-i wynika nie tylko z faktu reprogramowalności (wielokrotne wykorzystanie), ale także z bardziej uniwersalnej budowy wewnętrznej. Uniwersalność ta została uzyskana dzięki zastosowaniu na wyjściu układu zamiast standardowej bramki OR, komórki OLMC (Output Logic Macro Cell).

Budowę takiej komórki przedstawia **rys. 13**.

Dzięki zastosowaniu multiplekserów z programowanymi stanami sygnałów na wejściach adresowych, komórka OLMC może stać się jednym, z dowolnie wybranych, rodzajem wyjścia - może to być wyjście typu L, H, R, dedykowane lub nie. To właśnie ta właściwość plus możliwość wielokrotnego przeprogramowywania dały układom GAL dość silną pozycję wśród bardzo dużej, trzeba przyznać konkurencji.

Pewnym potwierdzeniem jakości myśli zawartej w układach GAL niech będzie fakt, iż największy chyba na świecie, a na pewno najbardziej uznany producent układów PLD - firma AMD, poszła w ślady Lattice'a i w 1990 roku rozpoczęła produkcję swojej wersji GAL'i - układów PALCE16V8, które są funkcjonalnym odpowiednikiem GAL16V8.

Co stosować - PAL'e czy GAL'e?

Pytanie jest postawione w nieco przewrotny sposób, ponieważ pominięto w sferze wyboru pozostałe układy PLD. Wynika to z faktu iż w naszych realiach dostęp do profesjonalnego oprogramowania (kompilatorów), urządzeń programujących, a i samych układów jest dość ograniczony - także ceną.

Układy PAL i GAL są już coraz szerzej stosowane w urządzeniach produkowanych w kraju, można je dostać w wielu sklepach, a niezbędny hardware i software jest osiągalny w różny sposób.

Układy PAL - ogólnie rzecz ujmując są nieco mniej elastyczne od GAL'i. Nie dotyczy to nowszych wersji struktur, np. takich jak PALCE z AMD). Z kolei układy GAL są droższe, choć nie jest to jakaś kolosalna różnica w cenie (układy G16V8 są o ok. 0.5 - 1 USD droższe od P16P8 - układ PAL w wersji z programowaną polaryzacją wyjścia), lecz za to można je wykorzystywać wielokrotnie.

Ze względu na stosunkowo starą technologię (bipolarna) układy PAL pobierają dość dużo prądu z układu zasilającego. Jest to pobór rzędu 180mA przy napięciu 5V. Ograniczenie poboru mocy w tego typu układach wiąże się ze znacznym zwiększeniem czasu propagacji sygnału - w konsekwencji maksymalna częstotliwość pracy maleje.

Układy GAL są znacznie mniej „prądożerne" (typowy pobór prądu ok. 45mA) i nie tracą przy tym szybkości pracy.

Tak więc na pytanie postawione w śródtytule można odpowiedzieć następująco:

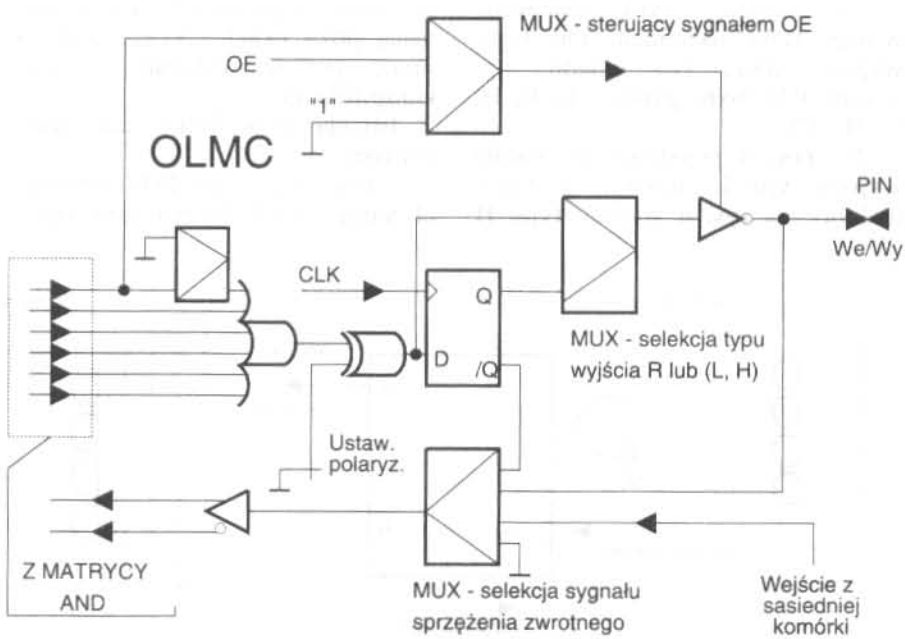
- układy GAL opłaca się stosować w prototypowych rozwiązaniach i w małych seriach produkcyjnych, a także tam gdzie bardzo ważne jest ograniczenie pobieranej przez układ mocy;
- układy PAL są znacznie lepsze do seryjnych urządzeń dokładnie przetestowanych.

Do zastosowań nieprofesjonalnych polecalibyśmy raczej serię GAL, ale ostateczne rozstrzygnięcia pozostawiamy Czytelnikom.

Piotr Zbysiński

Literatura:

1. W. Majewski, T. Łuba, K. Jasiński, B. Zbierzchowski „Programowalne moduły logiczne w syntezie układów cyfrowych", WKiŁ 1992.
2. T. Łuba, K. Jasiński, B. Zbierzchowski „Projektowanie i programowanie układów PLD", wyd. IT-PW 1989
3. Lattice Semiconductor Corp. „GAL Handbook" 1988.
4. Advanced Micro Devices „PAL Device Data Book" 1988.
5. Altera „User-Configurable Logic" 1988.
6. Monolithic Memories „PAL/PLE Device Data Book" 1990.



Rys. 13. Schemat komórki OLMC