Układy programowalne, część 2

Tę część kursu przeznaczymy na przedstawienie tajników architektur układów PLD, w tym przede wszystkim układów GAL22V10/ispGAL22V10. Informacje tutaj zawarte są niezbędną podstawą do wprawnego posługiwania się układami GAL.

Historia układów PLD (*Programmable Logic Devices*) sięga końca lat 60., kiedy to powstały pierwsze teoretyczne opracowania, w których wykazywano, że jest możliwe zbudowanie programowanego układu realizującego dowolną logiczną funkcję kombinacyjną i synchroniczną (z wykorzystaniem rejestrów). Podstawą do tych rozważań była praca *An Investigation of the Laws of Thought* George'a Boole'a z 1854 roku, w której wykazał on, że dowolną, najbardziej nawet skomplikowaną funkcję logiczną można stworzyć za pomocą trzech funktorów logicznych: AND, OR i NOT. Wystarczyło więc stworzyć układ, w którym funktory te są ze sobą połączone za pomocą programowanych połączeń, co zapewnia jego maksymalną uniwersalność.

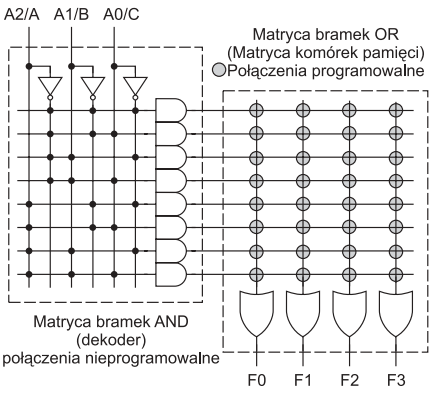
Na początku był chaos...

Jak łatwo zauważyć, możliwych sposobów połączenia funktorów ze sobą jest wiele. Historycznym przykładem układu PLD jest pamięć PROM, w której sygnały wejściowe są podawane na bramki AND o ustalonych połączeniach (tak że spełniają rolę dekodera adresowego pamięci), a ich wyjścia są połączone z programowalną matrycą bramek OR (**rys. 6**). Stosunkowo niewielka elastyczność takiego układu PLD i brak możliwości wygodnego projektowania automatów synchronicznych spowodowały, że prace badawcze trwały, a w ich wyniku powstały układy PLA (*Programmable Logic Array*). Różnią się one od pamięci wprowadzeniem programowanych połączeń w matrycy bramek AND (**rys. 7**), co spowodowało zwiększenie elastyczności tych układów, ułatwiło także optymalne wykorzystywanie ich zasobów.

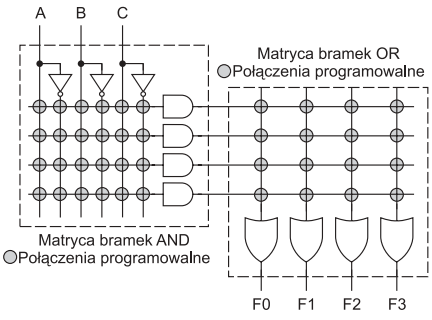
Kolejnym etapem rozwoju układów programowalnych były układy PAL (*Programmable Array Logic*), które charakteryzują się programowalną matrycą AND i skonfigurowaną na stałe (przez producenta) matrycą OR (**rys. 8**). Wprowadzono je do sprzedaży w roku 1978 jako remedium na problemy związane ze stosowaniem układów PLA: trudne przygotowywanie implementowanych projektów (nie było wtedy praktycznie żadnych narzędzi automatyzujących projektowanie!), duży pobór mocy, niewielka szybkość pracy.

Układy PLA i PAL były wykonywane w technologii bipolarnej z pamięcią konfigurującą typu PROM (jednokrotnie programowalną).

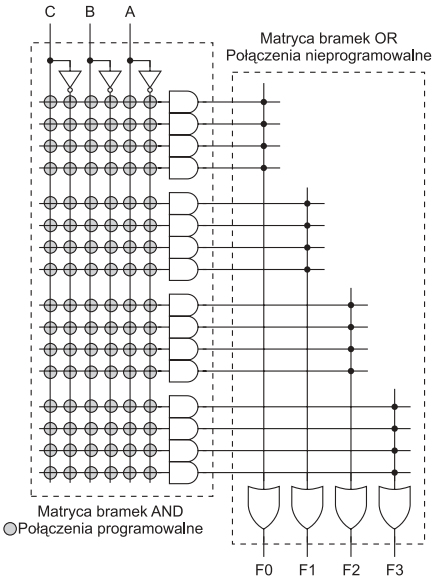
George Boole (1815-1864) w opublikowanym w 1854 roku "dziele życia" (*An Investigation of the Laws of Thought*) stworzył podstawy rozwoju techniki cyfrowej i później układów PLD.



Rys. 6. Budowa logiczna pamięci ROM/PROM



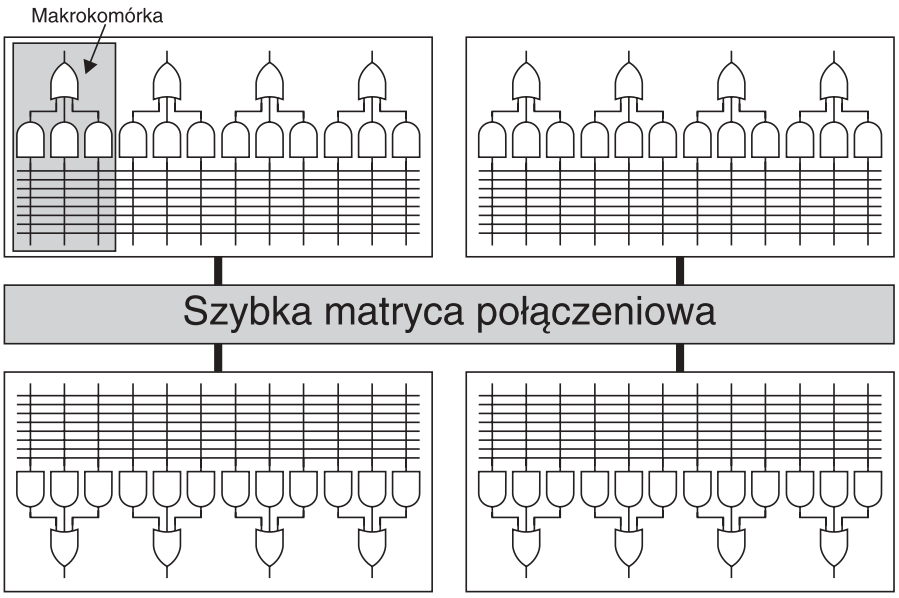
Rys. 7. Budowa logiczna układów PLA



Rys. 8. Budowa logiczna układów PAL

Bliżej współczesności...

Być może niektórzy Czytelnicy poczują się zawiedzeni, ale w tym momencie moglibyśmy przejść do omawiania architektury układów GAL22V10. Układy te powstały bowiem na bazie pomysłów z lat 70. Zaskakujące połączenie: nowoczesna (ciągle nieco awangardowa) technologia, bezpośrednio wykorzystująca pomysły z czasów - dla współczesnych - historycznych.



Rys. 9. Budowa logiczna układów CPLD

Zanim jednak przejdziemy do zgłębiania tajników układów GAL22V10, na chwilę powrócimy do historii rozwoju układów PLD, bo oferują one znacznie większe możliwości niż było to możliwe „za panowania" GAL22V10.

Ewolucja podążyła dwiema ścieżkami:

- Powiększania zasobów dostępnych w pojedynczych układach, poprzez powielanie komórek opartych na matrycach PAL. W ten sposób postały układy CPLD (*Complex Programmable Logic Devices* - **rys. 9**). Wszystkie współczesne układy CPLD wyposażono w nieulotne pamięci konfigurujące (EEPROM lub Flash), których zawartość może być wielokrotnie zmieniana.
- Zastosowaniu radykalnie odmiennej architektury, którą nazwano FPGA (*Field Programmable Gate Array* - **rys. 10**). Układy tego typu składają się z matryc jednakowych lub bardzo do siebie podobnych komórek, w których zintegrowano konfigurowalne zasoby logiczne. Połączenia pomiędzy nimi są możliwe dzięki traktom połączeniowym, których konfigurację może zmieniać użytkownik. W układach FPGA rolę pamięci konfigurującej spełnia zazwyczaj ulotna pamięć SRAM, której zawartość jest każdorazowo po włączeniu zasilania odtwarzana (dane są pobierane z zewnętrznej pamięci nieulotnej).

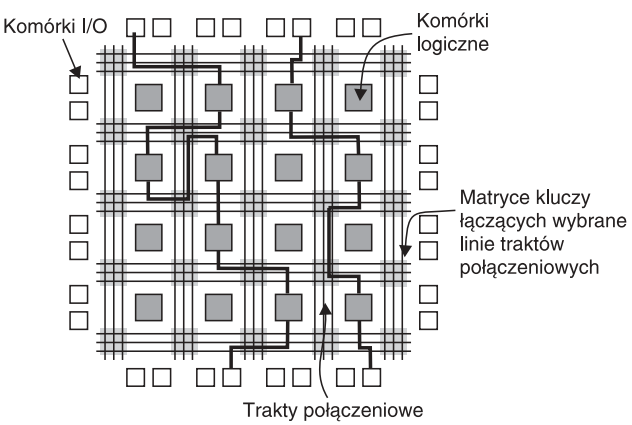
W ostatnich latach producenci wprowadzają do sprzedaży układy o nowatorskiej architekturze nazywanej SoC (*System on a Chip*) lub PSoC (*Programmable System on a Chip*), które składają się z mikroprocesora (często bardzo szybkiego) oraz dużego bloku logiki konfigurowalnej (**rys. 11**). Układy tego typu są coraz chętniej stosowane w aplikacjach, ponieważ pozwalają na budowanie w jednym układzie kompletnych urządzeń spełniających wymagania nawet bardzo zaawansowanych aplikacji.

Tab. 2. Funkcje sygnałów interfejsu JTAG

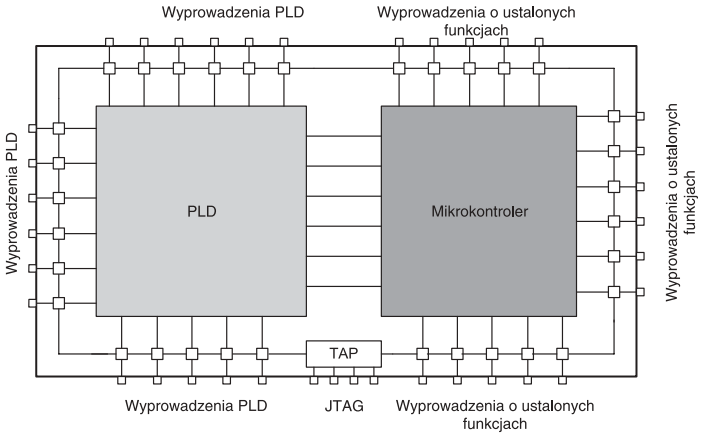
Nazwa	Opis funkcjonalny
TDI Test Data Input	Szeregowe wejście danych do konfiguracji i testowania. Dane z tego wejścia są synchronizowane narastającym zboczem sygnału zegarowego TCK.
TDO Test Data Output	Szeregowe wyjście danych wyprowadzanych z rejestru BST lub pamięci konfiguracji układu. Dane wyjściowe są synchronizowane opadającym zboczem sygnału zegarowego TCK.
TMS Test Mode Select	Wejście sterujące pracą automatu TAP zgodnie z grafem przedstawionym na rys. 13. Ustalenie wartości logicznej na tym wejściu powinno nastąpić przed narastającym zboczem sygnału zegarowego TCK.
TCK Test Clock	Wejście sygnału zegarowego, taktującego automat TAP oraz rejestr instrukcji.

ispGAL22V10 vs GAL22V10

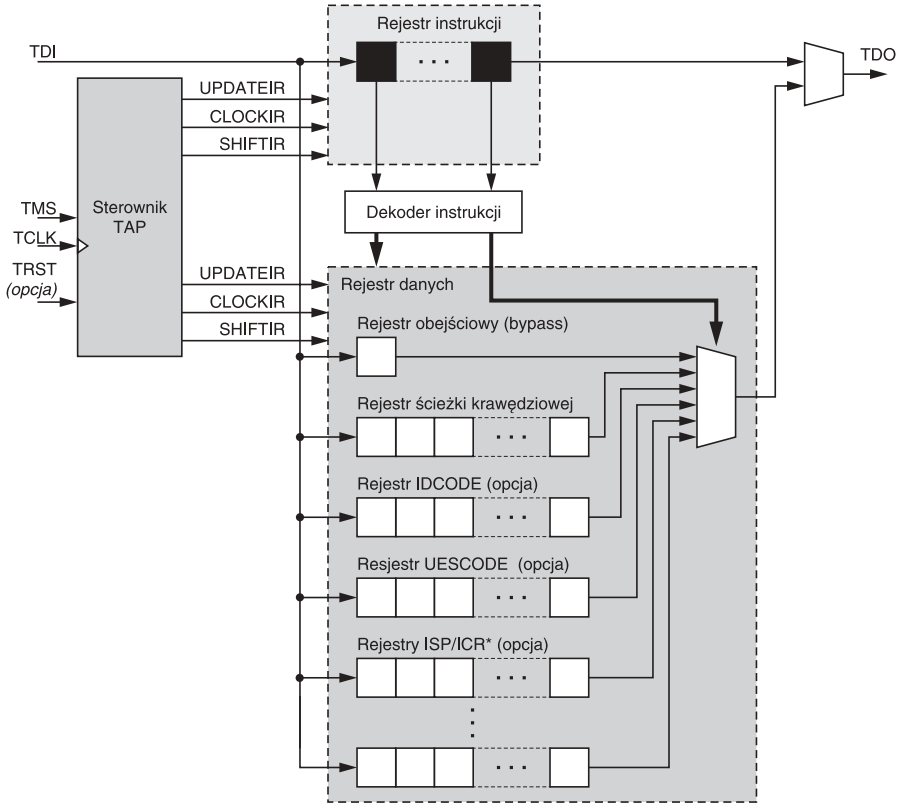
Układ ispGAL22V10 (w dowolnej wersji) jest funkcjonalnym odpowiednikiem standardowego układu GAL22V10. Kompatybilność dotyczy zarówno rozmieszczenia wyprowadzeń, jak i plików JEDEC wykorzystywanych do programowania układu.



Rys. 10. Budowa logiczna układów FPGA



Rys. 11. Budowa logiczna układów SoC



Rys. 12. Schemat blokowy jednej z wielu możliwych implementacji interfejsu JTAG

Tab. 3. Liczba programowalnych iloczynów dostępnych dla OLMC dołączonych do wyprowadzeń układu GAL22V10 (w obudowie PLCC28)

Numer wyprowadzenia I/O	Liczba iloczynów dostępnych dla OLMC
27	8
26	10
25	12
24	14
23	16
21	16
20	14
19	12
18	10
17	8

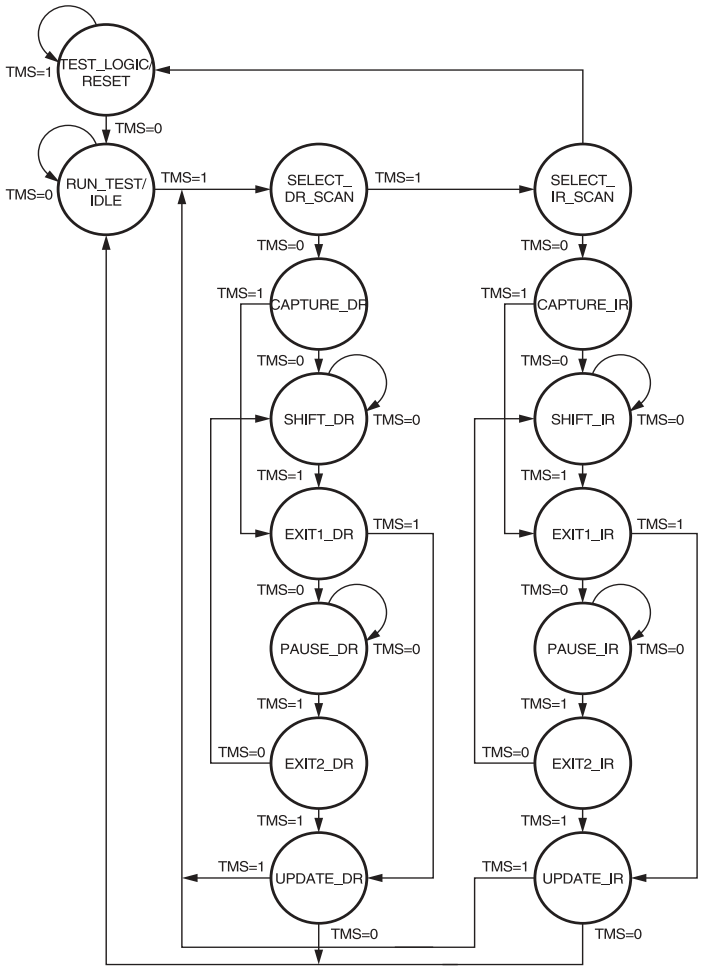
O programowaniu słów kilka

Programowanie ISP (*In System Programming*), obecnie tak modne wśród użytkowników mikrokontrolerów, jest wykorzystywane w układach CPLD i FPGA od początku lat '90. Obecnie obowiązuje jednolity standard - do programowania układów PLD jest powszechnie wykorzystywany interfejs JTAG. Niegdyś jego podstawowym zadaniem było umożliwienie testowania układów cyfrowych i połączeń między nimi, stopniowo zdobył on popularność głównie jako interfejs służący do programowania układów PLD w systemie. Obecnie JTAG jest dostępny nawet w układach o tak niewielkich zasobach logicznych jak w przypadku ispGAL22V10.

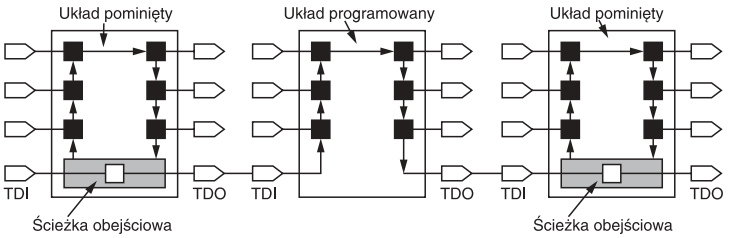
Układy wyposażone w JTAG mają wbudowane specjalne kontrolery zarządzające jego pracą (TAP - *Test Access Point* - **rys. 12**). TAP jest 16-stanowym automatem, którego cykl pracy pokazano na **rys. 13**. Przebiegiem cyklu pracy automatu TAP sterują cztery wyprowadzenia (TMS, TDI, TDO i TCK), których funkcje zestawiono w **tab. 2**. Obsługa programowania ISP została dodana do

Trwałość pamięci konfigurującej

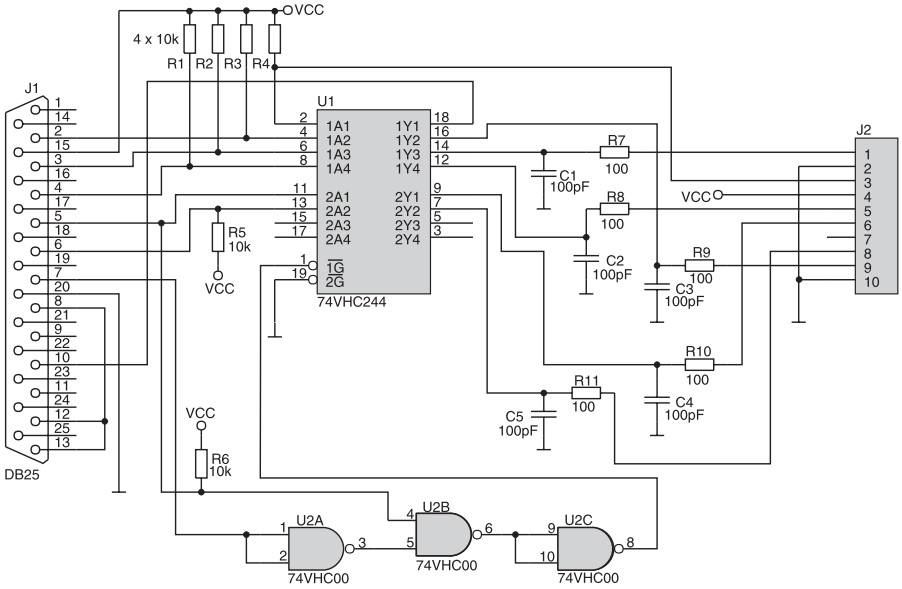
Pamięć EEPROM spełniająca w układzie ispGAL22V10 rolę pamięci konfigurującej jest duża, bowiem producent gwarantuje co najmniej 10000 cykli kasowanie-zapis.



Rys. 13. Cykl pracy automatu TAP



Rys. 14. Układy z interfejsem JTAG można łączyć w łańcuchy (na rysunku pominięto sygnały TMS i TCK, które są dostarczane do wszystkich układów jednocześnie)



Rys. 15. Schemat przykładowego programatora układów PLD firmy Lattice

standardowego JTAG-a „sztucznie" (bo, jak wspomniano, JTAG miał służyć do testowania układów po zamontowaniu w systemie), ale zostało to zrobione w taki sposób, że użytkownik tej „sztuczności" w żaden sposób nie odczuwa.

Za pomocą JTAG-a można programować zarówno pojedyncze układy (jak ma to miejsce m.in. w zestawie, który wykorzystamy podczas kursu), jak i wiele układów połączonych w łańcuch (**rys. 14**). Na rysunku, żeby nie zmniejszać jego czytelności nie narysowano sygnałów TMS i TCK, które są dostarczane równolegle do wszystkich układów wchodzących w skład łańcucha.

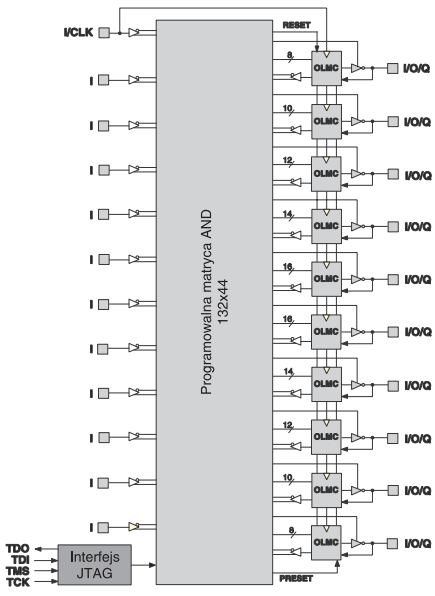
Korzystanie z JTAG-a, pomimo jego dość złożonej budowy, jest nadzwyczaj proste. Rolę programatora spełnia łatwy w wykonaniu interfejs (schemat elektryczny programatora ISP dla układów Lattice pokazano na **rys. 15**), za sterowanie jego pracą odpowiada specjalne oprogramowanie (ispVM), udostępniane przez firmę Lattice bezpłatnie. Najnowszą wersję tego programu oraz wersje dla Linuksa publikujemy także na CD-EP4/2004B. Sposób obsługi tego programu przedstawimy w jednym z kolejnych odcinków cyklu.

PAL vs GAL

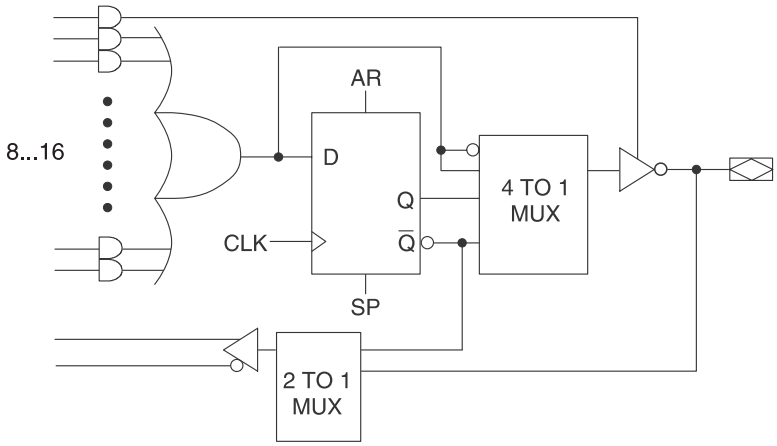
Układy GAL są reprogramowalnymi, uniwersalnymi wersjami układów PAL. Ich komórki wyjściowe są tak elastyczne, że można je skonfigurować w dowolny tryb obsługiwany przez układy PALxxR (z wyjściami rejestrowymi), PALxxH (bez inwerterów na wyjściu) i PALxxL (z inwerterami na wyjściu).

Nasz bohater: ispGAL22V10

Jak już wspomniano, architektura układu ispGAL22V10 jest bezpośrednim rozwinięciem klasycznych, bipolarnych PAL-i. Układ jest zbudowany (**rys. 16**) z 10 konfigurowalnych makrokomórek OLMC (*Output Logic Macro Cell*), na wyjściu których znajdują



Rys. 16. Architektura układów (isp)GAL22V10



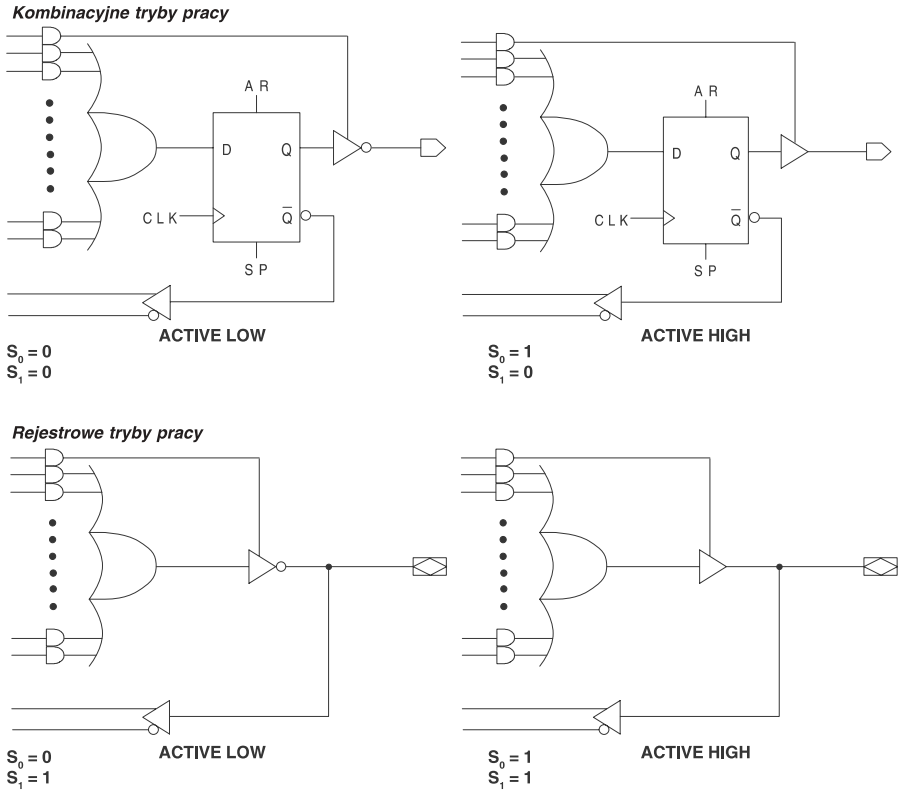
Rys. 17. Budowa komórki OLMC układu (isp)GAL22V10

się bufory trójstanowe. Sygnały na wejścia OLMC są podawane z programowanej matrycy bramek AND o organizacji 44 (liczba wejść bramek AND) x 132 (liczba bramek AND).

Budowę OLMC pokazano na **rys. 17**. W jej skład wchodzi przerzutnik D z wejściami: asynchronicznego zerowania (AR) i synchronicznego ustawiania (SP), na którego wejście D jest podawany sygnał wytwarzany zaprogramowaną przez użytkownika przez sumę iloczynów sygnałów wejściowych (oznaczonych literą I) i sygnałów podawanych zwrotnie na matrycę programowalną, wytwarzanych w innych OLMC. W zależności od lokalizacji OLMC, liczba dostępnych dla OLMC iloczynów waha się od 8 do 16 (według wzoru: „zewnętrzne" OLMC mają ich 8, następne 10, aż do 16 dla OLMC umieszczonych centralnie - **tab. 3**).

Multipleksery widoczne na rys. 17 są wykorzystywane wyłącznie w celu skonfigurowania trybu pracy OLMC (za ich konfigurację odpowiadają bezpieczniki S_0 i S_1, oddzielne dla każdej OLMC), nie można więc zmieniać ich stanu podczas pracy układu. Na **rys. 18** pokazano możliwe konfiguracje OLMC. W każdym z możliwych trybów pracy, sygnały wytwarzane w OLMC są zwrotnie przesyłane na programowalną matrycę AND, dzięki czemu funkcje logiczne realizowane w OLMC mogą być wykorzystywane przez inne OLMC (oczywiście, jeśli występuje taka konieczność). W przypadku pracy OLMC w trybie rejestrowym nie ma możliwości wykorzystania linii I/O jako wejściowej (jest na stałe skonfigurowana jako trójstanowe wyjście). Sygnał CLK dostarczany na wejścia zegarowe wszystkich przerzutników jest na stałe przypisany do wejścia I/CLK (wyprowadzenie numer 2 w obudowie PLCC28). Niestety, w układach GAL22V10 nie ma możliwości taktowania przerzutników niezależnymi sygnałami zegarowymi - w trybie rejestrowym OLMC mają do dyspozycji tylko jeden, wspólny sygnał. Ograniczenie to nie dotyczy sygnałów AR i SP, które są wytwarzane indywidualnie dla każdego przerzutnika.

Piotr Zbysiński, EP
piotr.zbysinski@ep.com.pl



Rys. 18. Możliwe sposoby skonfigurowania OLMC w układach (isp)GAL22V10