ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 1994
Seria: ELEKTRONIKA z. 3 Nr kol. 1254

Andrzej KURKOWSKI
Edward HRYNKIEWICZ
Instytut Elektroniki
Politechnika Śląska

EMULATOR LOGICZNY UKŁADÓW PLD

Streszczenie. W artykule analizowano różne rozwiązania konstrukcyjne emulatora logicznego układów programowalnych (PLD). Opisano model emulatora zbudowany na elementach FPGA firmy Xilinx oraz przedstawiono sposób wykorzystania zasobów układów FPGA na potrzeby emulacji.

Logic emulator of PLDs

Summary. Different possibilities of constructing PLD logic emulator are analysed in this paper. A model of the emulator built on the basis of XILINX FPGA is described as well as the manner of utilization of FPGA resources for emulation purposes.

Логический эмулятор прогаммных логических схем

Резюме. В стаье анализировано разные возможности построения логического емулятора прогаммных логических схем (ПЛС). Представлено модель емулятора создан на прогаммных логических матрицах фирмы КСИЛИНКС а также использование ресурсами этих матриц для требований емуляции.

1. WPROWADZENIE

W projektowaniu urządzeń techniki cyfrowej stosuje się dwie metody realizacji algorytmów pracy - sprzętową i programową. Sprzętowa realizacja opiera się głównie na układach cyfrowych o małym stopniu scalenia SSI/MSI, które są powszechnie produkowane i prezentowane w katalogach. Metoda ta przy wprowadzaniu jakichkolwiek modyfikacji wiąże się z koniecznością bezpośredniej ingerencji w strukturę urządzenia i dlatego wybiera się ją wszędzie tam, gdzie złożoność algorytmów pracy jest stosunkowo nieduża. W przeciwnym przypadku stosuje się realizację programową. Nowoczesne układy zbudowane na podstawie programowej metody realizacji cechuje modułowa budowa i duża elastyczność funkcjonalna, a modyfikacja algorytmu pracy polega jedynie na zmianie oprogramowania sterującego.

Nowe podejście do konstrukcji urządzeń cyfrowych opiera się na układach ASIC (*Application Specific Integrated Circuits*). Jeden specjalizowany układ scalony umożliwia realizację całego, nawet bardzo złożonego systemu cyfrowego, a budowane z nich urządzenia stają się tańsze w produkcji, pobierają mniej energii, szybciej pracują i mają mniejsze wymiary.

Oddzielną grupę układów ASIC stanowią układy PLD (*Programmable Logic Device*), w których końcową strukturę połączeń może zaprogramować i zrealizować użytkownik. Na przestrzeni ostatnich lat nietrudno zaobserwować dynamiczny wzrost wykorzystania układów logiki programowalnej w projektowaniu i konstrukcji urządzeń cyfrowych.

Łatwo wskazać dwie zasadnicze przyczyny wzrostu popularności układów PLD: wzrastający postęp technologii produkcji układów scalonych oraz masowe pojawienie się komputerów osobistych jako taniego oraz łatwo dostępnego środka umożliwiającego ich efektywniejsze wykorzystanie. Postępowi technologicznemu należy zawdzięczać powstanie wielu nowych typów układów programowalnych o różnorodnych strukturach - od stosunkowo prostych, poprzez dość złożone, aż do bardzo rozbudowanych. Gwałtownej poprawie uległy także ich parametry użytkowe: szybkość, pobór mocy, niezawodność, pojawiły się układy reprogramowalne. Wszystko to powoduje, że praktycznie każdy projekt można zrealizować dobierając odpowiednie typy układów PLD [5][7][8].

Masowe pojawienie się komputerów osobistych - drugi z przedstawionych czynników wzrostu popularności układów PLD - w zasadniczy sposób zmieniło proces projektowania urządzeń cyfrowych na podstawie elementów programowalnych. Początkowo projektowano układy "ręcznie", posługując się tablicami przepaleń, co było czasochłonne i dość niewygodne. Wraz z upowszechnieniem się mikrokomputerów osobistych powstała cała rzesza programów wspomagających projektowanie urządzeń cyfrowych na podstawie układów logiki programowalnej, jak również programatorów układów PLD współpracujących z nimi. Zasadniczo zmieniło to proces projektowania - praktycznie nie odchodząc od komputera można opracować projekt, przesymulować jego działanie i otrzymać zaprogramowany układ [8].

Współczesny projektant urządzeń cyfrowych dysponuje dziś już nie tylko bramkami i przerzutnikami, lecz technologia oferuje mu moduły o wyższym stopniu złożoności. Projektowanie w takiej sytuacji początkowo miało charakter czysto intuicyjny (poparty pomysłowością i rutyną profesjonalisty). Klasyczne już dziś metody syntezy układów cyfrowych dostarczają jedynie narzędzi do projektowania urządzeń zbudowanych tylko z elementów typu bramka, przerzutnik. Pewne etapy tej syntezy, jak np. minimalizacja funkcji boolowskich, kodowanie stanów, są przy bardziej złożonych projektach bardzo czasochłonne oraz opierają się na złożonym i uciążliwym rachunku. Poza tym, tak złożonych urządzeń, jak na przykład komputer osobisty, nie można opisać grafem przejść. Rzecz jasna nieporozumieniem byłoby sądzić, że cała klasyczna teoria automatów straciła na znaczeniu. W obliczu nowych możliwości układów scalonych część zagadnień po prostu przestała być aktualna, a pozostałą część tak zmodyfikowano, że powstały generalnie odmienne algorytmy nawet dla takich zadań jak minimalizacja funkcji boolowskich [4],[9].

Istnienie układów logiki programowalnej zmienia charakter procesu projektowania w taki sposób, że przede wszystkim przesuwa środek ciężkości procesu syntezy na abstrakcyjny opis działania układu. Przy bardziej złożonych projektach konstruktor może stopniowo tracić kontrolę nad projektem. Wbudowane w programy symulatory zazwyczaj do działania wymagają podania wektorów testowych, co dodatkowo komplikuje projekt.

Cennym wyposażeniem konstruktora układów cyfrowych, wykorzystującego w swojej pracy układy PLD, wydaje się być, prócz programatora, sprzętowy emulator układów PLD, który umożliwi zweryfikowanie poprawności projektu przed ostatecznym zaprogramowaniem układu.

2. WYBÓR KONCEPCJI UKŁADU

Przed przystąpieniem do wyboru koncepcji układu obiektywnie sprecyzowano, jakich funkcji może oczekiwać przyszły użytkownik od takiego urządzenia.

Zdecydowanie najbardziej pożądaną cechą jest możliwość emulacji jak największej liczby układów PLD. Umożliwi to swobodny dobór elementów do projektów bez sugerowania się możliwościami emulatora. Drugą, nie mniej ważną, cechą jest emulacja w czasie rzeczywistym, pozwalająca nie tylko na statyczne sprawdzenie poprawności logicznej projektu, lecz również na dynamiczne testowanie współpracy układu PLD z otoczeniem.

Są to bez wątpienia dwa najważniejsze atrybuty, zdecydowanie podnoszące atrakcyjność całego urządzenia. Dlatego należy tak zaprojektować emulator, aby spełniał oba powyższe wymagania.

Po sformułowaniu podstawowych założeń projektowych można dokonać wyboru koncepcji urządzenia. W tym celu przeanalizowano nasuwające się cztery rozwiązania:

2.1. Emulacja sprzętowo-programowa

Idea takiego rozwiązania opiera się na wykorzystaniu wysoce specjalizowanego układu sterującego, skonstruowanego np. na bazie mikroprocesora bądź mikrokontrolera jednoukładowego. Emulacja polegałaby na analizie sygnałów wejściowych i na ich podstawie sterowaniem aktywnością linii wyjściowych. Odpowiedni program sterujący, inny dla każdego projektu, każdorazowo musiałby zostać umieszczony w pamięci programu mikroprocesora. Byłoby to stosunkowo prostym zadaniem, pod warunkiem że emulowany układ PLD realizowałby wyłącznie funkcje kombinacyjne. W przypadku układów sekwencyjnych zadanie znacznie się komplikuje, wymaga bowiem opracowania skomplikowanego i bardzo zróżnicowanego algorytmu pracy. Przy wielkiej różnorodności produkowanych układów może okazać się to zadaniem wręcz niemożliwym do wykonania.

2.2. Zastosowanie pamięci RAM

Rozwiązanie polegałoby na zastąpieniem układu PLD pamięcią RAM, w której przechowywana byłaby tablica prawdy elementu. Odpowiednim wypełnieniem pamięci miałby zajmować się mikrokomputer bądź też specjalizowany układ sterujący. Takie rozwiązanie wymusza jednak zastosowanie pamięci o bardzo wielkiej pojemności (najpopularniejsze z wykorzystywanych elementów PLD to układy wielowejściowe) oraz o bardzo dużej szybkości działania (emulacja w czasie rzeczywistym).

2.3. Wykorzystanie układu E^2PLD

Idea takiego rozwiązania jest bardzo podobna do poprzedniej koncepcji, lecz zamiast pamięci wykorzystano tu inny układ PLD pełniący podobną rolę. Oczywiście musi być to stosunkowo skomplikowany i elektrycznie reprogramowalny element (inspiracją był układ XL78C800 firmy EXEL [6]). Ogólna idea pomysłu polega na zaprojektowaniu zestawu progra-

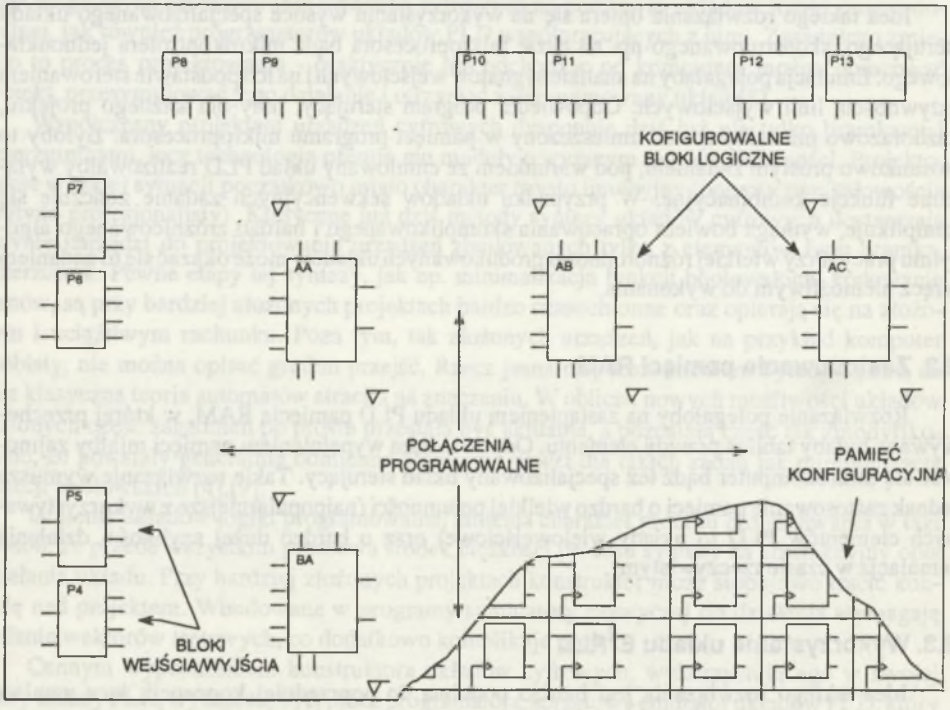
mator - emulator. Emulacja w takim urządzeniu przebiegałaby dwuetapowo. Najpierw należałoby zaprogramować element E^2PLD w taki sposób, aby pod względem zgodności wyprowadzeń i realizowalnych funkcji pełnił rolę emulowanego układu, a następnie udostępnić jego wyprowadzenia urządzeniu, w którym dokonywana jest emulacja.

Koniecznym wymogiem do realizacji tego rozwiązania jest posiadanie szczegółowych informacji dotyczących programowania danego układu. Niestety, informacje te są niechętnie ujawniane przez producentów i rzadko pojawiają się w katalogach firmowych. Bardzo utrudnia to wybór odpowiedniego elementu E^2PLD do urządzenia. Dodatkowo ilość przeprogramowań takich elementów jest liczbą skończoną. Z czasem spowoduje to konieczność wymiany - rozwiązanie stosunkowo kosztowne i kłopotliwe.

2.4. Zastosowanie układu FPGA firmy Xilinx

Jest to, w pewnym sensie, rozwinięcie poprzedniego pomysłu wynikające głównie z faktu, że elementy FPGA firmy Xilinx zasadniczo odbiegają od pozostałych układów PLD (według niektórych źródeł układy FPGA nie są zaliczane do PLD, [1][2][3]).

Przede wszystkim elementów tych nie programuje się w tradycyjnie pojęty sposób. Programowanie polega na samodzielnym odczycie przez element struktury połączeń z zewnętrznej pamięci ROM. Zupełnie inna jest również wewnętrzna architektura układu.



Rys. 1. Struktura wewnętrzna układu FPGA firmy Xilinx
Fig. 1. The structure of Xilinx FPGA

W stosunku do poprzedniej koncepcji rozwiązanie to posiada następujące zalety:

- Ilość przeprogramowań jest nieograniczona (nie istnieje taki parametr).
- Nie jest wymagana budowa programatora.

Natomiast w stosunku do pozostałych:

- Układy FPGA są elementami bardzo szybkimi - istnieje zatem realna szansa, że emulator zbudowany na bazie takiego elementu będzie pracował w czasie rzeczywistym [1].
- Inna architektura wewnętrzna umożliwi łatwą adaptację dla nowo powstających układów PLD (umożliwi wręcz emulację projektów opartych na strukturach własnych pomysłów).

Pomimo szeregu zalet takie rozwiązanie nie jest pozbawione wad:

- Do projektowania wymagany jest specjalizowany program, który jest stosunkowo drogi i do działania wymaga rozbudowanego systemu komputerowego [3].
- Koniecznym warunkiem do zbudowania emulatora jest posiadanie szczegółowych danych dotyczących sposobu programowania (konfigurowania) układu, które stanowią ścisłą tajemnicę producenta i nie zostały nigdzie opublikowane.

Z przedstawionych koncepcji jasno wynika, że skonstruowanie emulatora spełniającego przedstawione wcześniej założenia nie jest sprawą prostą. Zdecydowanie najwięcej zalet reprezentuje koncepcja bazująca na wykorzystaniu układu FPGA i dlatego dalsze rozważania ograniczono do tej koncepcji.

Współpraca i sposób połączenia z urządzeniem nadrzędnym

Emulator powinien współpracować z nadrzędnym urządzeniem zewnętrznym, które dostarczy mu niezbędnych do działania danych. Ponieważ wszystkie narzędzia programowe pomocne w trakcie projektowania, jak również sam programator, współpracują z komputerem osobistym, najbardziej celowe wydaje się wykonanie urządzenia również z nim współpracującego. Emulator powinien łączyć się z komputerem wykorzystując do tego celu standardowe złącza. Wykonanie urządzenia w postaci karty rozszerzającej możliwości funkcjonalne komputera należy raczej odrzucić z uwagi na utrudnioną możliwość przenoszenia. Pozostaje zatem łącze równoległe (Centronics) lub szeregowe (RS 232C). Jednakże łącze równoległe jest jednokierunkowe i zazwyczaj do niego podłączona jest drukarka. Wprowadzanie nowego dwukierunkowego standardu transmisji równoległej (Bitronics) przebiega zbyt wolno. Dlatego w projektowanym urządzeniu korzystniej jest zastosować łącze szeregowe (powszechnie przyjęto komputer osobisty wyposażać w dwa takie złącza, z których jedno jest wolne).

Format danych wejściowych

Wśród producentów układów logiki programowalnej przyjęty został ujednolicony sposób opisu połączeń. Wyrażany jest za pomocą kombinacji zer i jedynek jednoznacznie określających stan wewnętrznej architektury układu po zaprogramowaniu. Ten rodzaj opisu stanowi obecnie światowy standard obowiązujący dla wszystkich układów logiki programowalnej, a jego wyrazem jest zbiór w formacie JEDEC. Praktycznie wszystkie programy wspomagające projektowanie układów PLD potrafią wytworzyć zbiór JEDEC, który jest także zbiorem wejściowym dla

programatora [8]. Celowe wydaje się więc przyjęcie właśnie takiego formatu danych wejściowych dla emulatora.

Pozwoli to na przerzucenie pewnych skomplikowanych operacji (jak np. minimalizacja funkcji logicznych) na dowolny program wspomagający programowanie układów PLD.

Funkcje dodatkowe

Zdecydowanie bardzo pożyteczną funkcją (często zaniedbywaną w emulatorach pamięci EPROM) jest możliwość prezentacji stanu poszczególnych wyprowadzeń w trakcie pracy układu. Powszechnie wiadomo, że odpowiednie, wizualne przedstawianie danych znacznie podnosi ich czytelność. Ułatwi i przyśpieszy to proces testowania, jak również umożliwi szybsze wykrywanie błędów działania. Z tą opcją wiąże się kolejna funkcja dodatkowa: możliwość wyznaczenia wektorów testowych. Większość profesjonalnych programatorów posiada opcję testowania aktualnie programowanych układów PLD za pomocą dołączonych do zbioru JEDEC wektorów testowych. Przedstawienie stanu poszczególnych wyprowadzeń emulowanego elementu np. w postaci przebiegów czasowych umożliwi łatwe tworzenie wektorów i ich późniejsze dopisanie do zbioru JEDEC.

Ponieważ wykorzystany układ FPGA jest bardzo rozbudowany i może służyć do budowy systemów cyfrowych o złożoności przekraczającej 6000 bramek logicznych. Prócz układów PLD będzie mógł emulować również i inne układy cyfrowe (włącznie z samym sobą). Z tego też powodu warto rozszerzyć zakres emulowanych elementów o standardowe układy cyfrowe SSI/MSI.

Zalety takich rozwiązań są praktycznie bezdyskusyjne, dlatego zdecydowano się uwzględnić je w projektowanym urządzeniu.

3. SPOSÓB PRZYGOTOWANIA DANYCH DLA UKŁADU FPGA

W trakcie projektowania rozważono następujące koncepcje:

Wykorzystanie systemu XACT

Takie rozwiązanie polegałoby na stworzeniu, opierając się na danych zawartych w zbiorze JEDEC, listy połączeń w formacie XNF (Xilinx Netlist Format), która stanowi podstawowy zbiór wejściowym dla systemu XACT. System XACT jest zestawem specjalizowanych programów służących do wspomagania projektowania urządzeń cyfrowych na bazie układów FPGA firmy Xilinx [1][2][3]. Uzyskany w wyniku działania systemu zbiór wyjściowy byłby zbiorem opisującym konkretny projekt w strukturze układu FPGA. Rola pozostałej części oprogramowania sprowadzałaby się do:

- przygotowania odpowiedniej listy połączeń dla systemu XACT,
- wysłania zbioru wynikowego do układu FPGA,
- odpowiedniej prezentacji wyników analizy stanu wyprowadzeń emulowanego układu.

Zasadniczą wadą takiego rozwiązania jest konieczność wykorzystywania odrębnego, wysoce skomplikowanego programu. Od potencjalnego użytkownika emulatora wymagana więc będzie znajomość, przynajmniej w podstawowym stopniu, systemu XACT.

Kolejną wadą jest bardzo długi czas działania systemu przy bardziej skomplikowanych projektach. Dla przykładu tworzenie odpowiednio zaawansowanego urządzenia (216 wykorzystanych bloków CLB) trwało około 30 godzin. Praktycznie dyskwalifikuje to budowę emulatora jako urządzenia przydatnego w procesie projektowania układów PLD.

Należy również wspomnieć, że takie rozwiązanie niesie ze sobą konieczność rozpoznania zbioru XNF. Jest to jedyny zbiór wykorzystywany przez system XACT, którego format firma Xilinx udostępniła. Trudno natomiast jednoznacznie stwierdzić, czy został on udostępniony wyłącznie dla firm tworzących oprogramowanie wspomagające projektowanie układów PLD, czy też wszystkim zainteresowanym.

Stworzenie własnego programu obsługi elementu FPGA

Metoda ta pozbawiona jest wad poprzedniej koncepcji. Polega ona na stworzeniu własnego programu tworzącego zbiór konfiguracyjny dla układu FPGA. Tylko taki sposób umożliwi obsługę emulatora bez konieczności używania systemu XACT. Wiąże się jednak z koniecznością poznania znaczenia bitów konfiguracyjnych w pamięci układu FPGA.

Z przedstawionych powyżej rozważań wynika, że zbudowanie emulatora na bazie układu FPGA nie jest możliwe bez dodatkowych badań nad zbiorami opisującymi projekt. Zdecydowanie najwięcej zalet posiada koncepcja bazująca na stworzeniu własnego oprogramowania sterującego elementem FPGA. Jest to również rozwiązanie najbardziej pracochłonne, lecz zapewniające osiągnięcie najlepszych rezultatów, gdyż poznanie formatu zbioru XNF przyniesie stosunkowo niewielkie korzyści - zbiór ten wykorzystywany jest wyłącznie przez system XACT. Zbiór z danymi konfiguracyjnymi w postaci wejściowej dla programatora pamięci EPROM jest zbiorem w powszechnie znanym formacie (INTEL EXTENDED, [10]) oraz zawiera wszystkie informacje niezbędne do skonfigurowania układu FPGA. Ponieważ nie wiadomo, co dzieje się z przeczytanymi przez układ FPGA danymi, można w pierwszym przybliżeniu stwierdzić, że zbiór ten zawiera bezpośrednie odwzorowanie wewnętrznej pamięci konfiguracyjnej. Poznanie znaczenia poszczególnych bitów tej pamięci pozwoli na całkowite uniezależnienie się od systemu XACT. Umożliwi to pełne wykorzystanie najważniejszej (zdaniem autorów) cechy układów FPGA, jaką jest możliwość dynamicznej zmiany konfiguracji w trakcie pracy układu. Bez tych wiadomości korzystanie z tej właściwości jest tylko symboliczne, wymaga bowiem stosowania odpowiednio pojemnych zewnętrznych pamięci i wcześniej przygotowanych projektów. W żaden sposób nie jest natomiast możliwa budowa urządzeń cyfrowych o modyfikowanym na bieżąco algorytmie pracy.

Ponieważ rozmiar pamięci konfiguracyjnej pojedynczego układu FPGA jest znaczny (kilka kB), poznanie funkcji tak wielkiej ilości danych jest zadaniem, którego realizacja może trwać kilkanaście miesięcy. Dlatego przyjęto, że w zbiorze wzorcowym modyfikowana będzie tylko funkcja logiczna realizowana przez poszczególne bloki, a wzajemne ich połączenia pozostaną bez zmian. Zawęża to proces analizy do bitów odpowiedzialnych tylko za konfigurację bloku.

Przyjęto zatem następującą strategię:

- za pomocą systemu XACT należy stworzyć odpowiednią bibliotekę zawierającą wszystkie elementy układu PLD - tak stworzony zbiór będzie stanowił wzorzec dla późniejszych modyfikacji,
- odpowiednio napisany program obsługi na podstawie zbioru JEDEC dokona zmian w zbiorze wzorcowym,
- powstały w wyniku powyższych operacji zbiór będzie właściwym zbiorem konfiguracyjnym układu FPGA.

Zaznaczyć jednak trzeba, że rola i znaczenie poszczególnych bitów w pamięci konfiguracyjnej układu znane są tylko producentowi. Dane te nie zostały udostępnione żadnemu ze znanych producentów oprogramowania wspomagającego projektowanie układów PLD. W rezultacie tego takie programy, jak ABEL-FPGA lub COMPASS, które według ulotek reklamowych umożliwiają projektowanie układów FPGA firmy Xilinx, potrafią jedynie wytworzyć listę połączeń w formacie XNF. Posiadanie systemu XACT jest zatem niezbędne dla jakichkolwiek prac projektowych opartych na elementach FPGA firmy Xilinx.

Należy również zdawać sobie sprawę, że producent ściśle strzeże swoich tajemnic. Możliwe jest zatem występowanie pewnych określonych bitów, nie mających żadnego wpływu na konfigurację, a wprowadzonych jedynie dla odwrócenia uwagi. Bez znajomości struktury wewnętrznej układu jest to trudne do zauważenia.

Analizę przeprowadzono w sposób następujący:

- stworzono projekt, który angażował wyłącznie jeden blok CLB o ściśle określonej konfiguracji; dla takiego projektu stworzono zbiór wzorcowy w formacie INTEL EXTENDED,
- stworzono szereg zbiorów (około 500 dla jednego bloku), układów o odpowiednio zmodyfikowanej konfiguracji i dla każdego z nich generowano zbiór w ww. formacie,
- dokonywano porównania tak spreparowanych zbiorów ze zbiorem wzorcowym i na podstawie obserwacji otrzymanych wyników wyróżniono poszczególne bity odpowiedzialne za konfigurację danego bloku.

W związku z dużą liczbą operacji, jakie należy wykonać zarówno podczas inicjacji elementu FPGA, jak i podczas analizy, wygodnie będzie wykonać emulator na bazie mikroprocesora. Wykorzystanie stosunkowo złożonego automatu sekwencyjnego jest wprawdzie możliwe, lecz ze względów ekonomicznych nieopłacalne. Schemat ideowy urządzenia należy również w pewnym stopniu podporządkować elementowi FPGA stosując wskazówki i zalecenia producenta [1]. Omawianie konkretnych rozwiązań mija się jednak z celem artykułu.

4. OPROGRAMOWANIE STERUJĄCE PRACĄ EMULATORA

Oprogramowanie sterujące pracą emulatora układów PLD powinno składać się z dwóch zasadniczych części:

- program pracy emulatora (tylko w przypadku wykorzystania mikrokomputera jednoukładowego do sterowania elementarnymi funkcjami urządzenia - rozwiązanie polecane),
- program sterujący (dla komputera PC), umożliwiający sterowanie wszystkimi dostępnymi funkcjami oraz odpowiednie przetworzenie i przesyłanie danych niezbędnych do prawidłowego funkcjonowania urządzenia.

Program pracy emulatora

Program pracy emulatora powinien zostać napisany w języku asembler i zawierać następujące procedury:

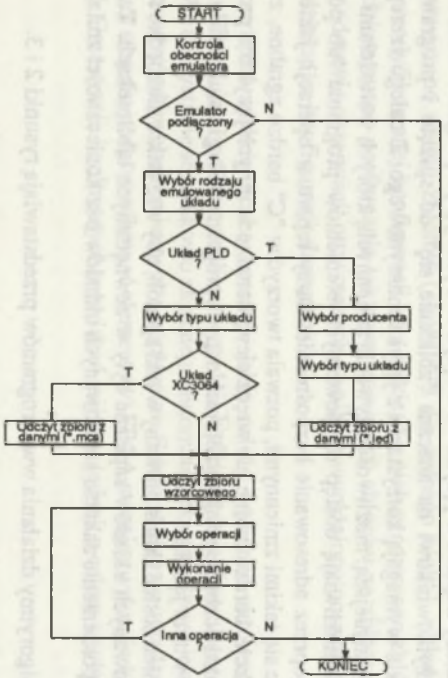
- obsługi portu szeregowego,
- obsługi układu FPGA i rejestrów symulacyjnych,
- przyjmowania, dekodowania oraz wykonywania funkcji zlecanych przez urządzenie nadrzędne,
- testu niektórych bloków funkcjonalnych układu po załączeniu zasilania.

Program sterujący emulatorem

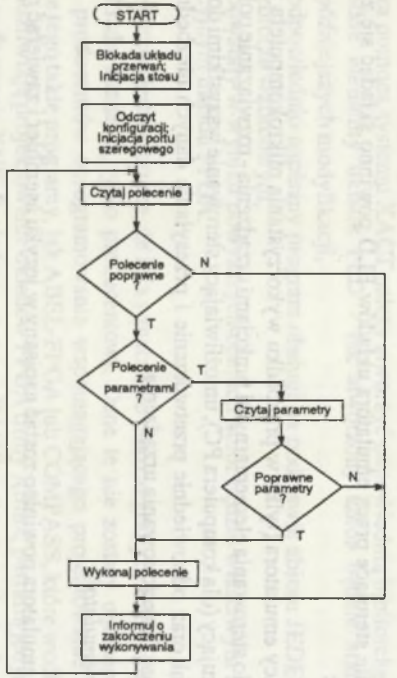
Z dostępnych kompilatorów języków wysokiego poziomu (Turbo Pascal, C/C^{++}, Basic) program sterujący należy napisać w języku "C". Język ten, jako język programowania strukturalnego, pozwala użytkownikowi na znaczne "zbliżenie się" do sprzętu i programowanie operacji, które zwykle wymagają korzystania z języka maszynowego. Z całego szeregu zalet wymienić trzeba najważniejszą: możliwość używania zmiennych, których wartościami są adresy. Zmienne takie umożliwiają dostęp do dowolnych elementów programu lub po prostu komórek pamięci poprzez adresowanie bezpośrednie. Duży repertuar operacji, jakie można wykonać posługując się takimi zmiennymi, pozwala tworzyć w "C" bardzo zgrabne, zwięzłe i pomysłowe konstrukcje, bardzo trudne lub wręcz niewykonalne w innych językach programowania. Z uwagi na dużą liczbę operacji na zbiorach danych program obsługi emulatora wskazane jest napisać właśnie w tym języku.

Program powinien zostać skonstruowany w taki sposób, aby wszystkie niezbędne informacje dotyczące emulowanych układów zapisane były w zewnętrznych bibliotekach. Takie rozwiązanie umożliwi poszerzenie zakresu emulowanych układów bez konieczności zmiany kodu źródłowego.

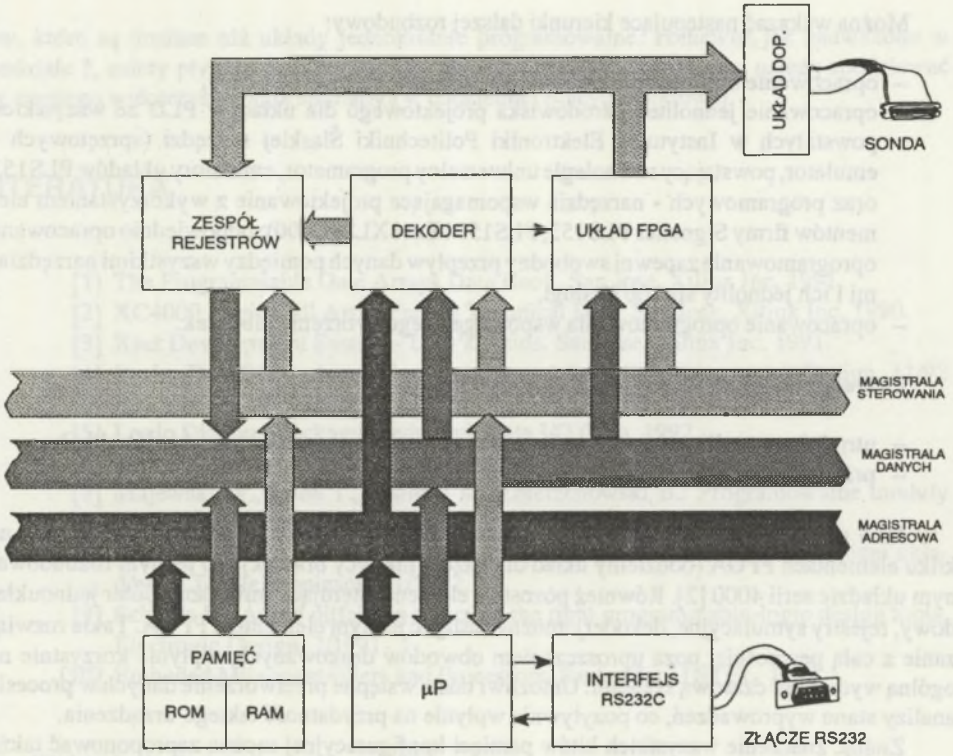
Przykładowe algorytmy działania ww. programów przedstawiają rysunki 2 i 3.



Rys. 2. Schemat blokowy programu sterującego emulatorem
Fig. 2. Block diagram of emulator controlling program for host computer



Rys. 3. Schemat blokowy programu pracy emulatora
Fig. 3. Block diagram of the emulator program operation



Rys. 4. Schemat blokowy emulatora
Fig. 4. Emulator block diagram

5. PODSUMOWANIE

Emulator logiczny układów PLD zbudowany zgodnie z przedstawioną koncepcją został uruchomiony i przetestowany. Przyjęta koncepcja urządzenia okazała się słuszna w praktyce - układ poprawnie realizował wszystkie założone funkcje. Przeprowadzone próby emulacji dla przedstawicieli większości popularnych rodzajów układów PLD potwierdziły słuszność przyjętych założeń. Schemat blokowy wykonanego urządzenia przedstawia rys. 4.

Również oprogramowanie sterujące działaniem urządzenia opracowano według wcześniejszych założeń (rozdział 4).

Komunikacja z użytkownikiem zrealizowana została poprzez opcje wybierane z rozwijalnych okien menu głównego. Poprzez zablokowanie niektórych opcji wymuszono określoną kolejność ich wykonywania. Dla przykładu nie można wykonać emulacji, jeżeli wcześniej nie zostanie wybrany rodzaj emulowanego elementu. Pewną inspiracją co do celowości zamieszczania konkretnych opcji było oprogramowane programatorów układów PLD oraz symulatorów pamięci EPROM.

Można wskazać następujące kierunki dalszej rozbudowy:

- opracowanie bibliotek dla pozostałych układów cyfrowych i PLD,
- opracowanie jednolitego środowiska projektowego dla układów PLD ze wszystkich powstałych w Instytucie Elektroniki Politechniki Śląskiej narzędzi (sprzętowych - emulator, powstający równolegle uniwersalny programator, emulatory układów PLS153 oraz programowych - narzędzia wspomagające projektowanie z wykorzystaniem elementów firmy Signetics PLS153, PLS159 i Exel XL78C800); odpowiednio opracowane oprogramowanie zapewni swobodny przepływ danych pomiędzy wszystkimi narzędziami i ich jednolity sposób obsługi,
- opracowanie oprogramowania wspomagającego tworzenie bibliotek.

Za wadę emulatora można uznać ograniczenia:

- utrudnione możliwości adaptacji na potrzeby nowo powstających układów PLD,
- pracochłonny proces tworzenia zbiorów bibliotecznych.

Jako alternatywne rozwiązanie należy rozpatrzyć wykonanie emulatora opierając się na kilku elementach FPGA (oddzielny układ dla każdej matrycy bramek) lub jednym rozbudowanym układzie serii 4000 [2]. Również pozostałe elementy sterujące (mikrokomputer jednoukładowy, rejestry symulacyjne, dekoder) można zastąpić jednym elementem FPGA. Takie rozwiązanie z całą pewnością, poza uproszczeniem obwodów drukowanych, wpłynie korzystnie na ogólną wydajność czasową systemu. Umożliwi także wstępne przetworzenie danych w procesie analizy stanu wyprowadzeń, co pozytywnie wpłynie na przydatność takiego urządzenia.

Znając znaczenie wszystkich bitów pamięci konfiguracyjnej można zaproponować także rozwiązanie polegające na samodzielnym konfigurowaniu układu FPGA według informacji zawartych w zbiorze JEDEC.

Jeżeli udałoby się poznać znaczenie pozostałych bitów pamięci konfiguracyjnej, możliwa do realizacji stałaby się koncepcja polegająca na takim skonfigurowaniu elementu FPGA, aby pełnił on takie same funkcje jak emulowany układ. Znając wartości opóźnień wnoszonych przez bloki i poszczególne połączenia programowalne można będzie tak sterować procesem rozmieszczania bloków i prowadzenia połączeń w układzie FPGA, aby pod względem logicznym i funkcjonalnym pełnił on rolę emulowanego układu.

Przyszłe konstrukcje warto także rozbudować o niektóre standardowe funkcje spotykane w analizatorach stanów logicznych (różne częstotliwości próbkowania, analiza sterowana zdarzeniami, ustawianie progów wyzwalania itp.). Warto także przewidzieć możliwość zadawania stanów wejściowych z klawiatury komputera. W takim wykonaniu emulator stałby się symulatorem logicznym.

Wykonany emulator układów PLD można uznać za użyteczne urządzenie w pracowni elektronicznej. Daje on możliwość łatwego i szybkiego sprawdzenia (pod względem zachowania logicznego) projektów układów cyfrowych tworzonych samodzielnie przez użytkownika lub za pomocą specjalizowanych programów komputerowych. Możliwość emulacji, wymagających specjalnego programatora układów PLD, jest przydatna do uruchamiania, testowania lub napraw urządzeń cyfrowych. Emulator taki w pełni uwidacznia swoje zalety, zwłaszcza podczas uruchamiania urządzenia, zwalnia bowiem z konieczności stosowania reprogramowalnych elemen-

tów, które są droższe niż układy jednokrotnie programowalne. Ponieważ, jak zauważono w rozdziale 1, zalety płynące z wykorzystania układów PLD są bardzo duże, należy spodziewać się częstego wykorzystywania emulatora w codziennej pracy projektowej.

LITERATURA

[1] The Programmable Gate Arrays Data Book. San Jose, Xilinx Inc. 1991.
[2] XC4000 Logic Cell Array Family Technical Data. San Jose, Xilinx Inc. 1990.
[3] Xact Development System - User's Guide. San Jose, Xilinx Inc. 1991.
[4] Burky D.: FPGA advances cut delays, and flexibility, Electronics Design, 11/92 San Jose, Xilinx Inc. 1991.
[5] Logic Diagram Package. Redmond, Data I/O Corp. 1992.
[6] ERASIC Multi-Level E2PLDs. San Jose, EXEL Microelectronics, 1988.
[7] Majewski W., Łuba T., Jasiński K., Zbierzchowski B.: Programowalne moduły logiczne w syntezie układów cyfrowych. WKiŁ, Warszawa 1992.
[8] Kania D., Szreter J.: Projektowanie urządzeń cyfrowych z wykorzystaniem układów PLD, Elektronizacja 11/1992.
[9] Schulze B.: Avoid pitfalls in selecting the right programmable-logic design tools, Electronic Design, 7/1991.
[10] Embeded Microcontrollers and Processors. Santa Clara INTEL, 1992.

Wpłynęło do Redakcji w czerwcu 1994 r.

Abstract

The paper contains a short review of possible realizations of logic emulators of programmable logical devices. The basic technical requirements for such an emulator and their critical examination are described. It is shown that a very simple but unconventional and useful structure of PLD logic emulator may be designed if RAM based FPGA is used for this purpose. A model of such an emulator was built on the basis of XILINX FPGA. The paper presents a unique way of utilizing FPGA resources for the emulation purposes as well as for building of devices of which the algorithm of operation can be changed.