ŚRODOWISKO DO SZYBKIEGO PROTOTYPOWANIA SYSTEMÓW DO DIAGNOSTYKI W CZASIE RZECZYWISTYM

Grzegorz RUBIN, Mirosław OMIELJANOWICZ
Politechnika Białostocka, Wydział Informatyki, Katedra Systemów Czasu Rzeczywistego
15-351 Białystok, ul. Wiejska 45A, gregor@ii.pb.bialystok.pl

Streszczenie

Idea szybkiego prototypowania (ang. rapid prototyping) oznacza takie prowadzenie prac projektowych i wdrożeniowych aby maksymalnie ograniczyć czas i koszty związane z wprowadzeniem nowych koncepcji i produktów zaawansowanych technologii na rynek. Prototypowanie to, ujmując lapidarnie szybkie stworzenie prototypu urządzenia, czy też programu, umożliwiającego dokonanie oceny jego funkcjonalności, przydatności i akceptowalności przez końcowego odbiorcę. Realizacja takiego podejścia w ogólnym przypadku jest bardzo trudna. Tworzone są więc systemy prototypowania dla wybranych dziedzin i gałęzi przemysłu. W przedstawionej pracy opisana została koncepcja postępowania i propozycja zestawu narzędzi szybkiego prototypowania z dziedziny cyfrowego przetwarzania sygnałów akustycznych i wibroakustycznych. Pokazano też wykorzystanie jej do stworzenia modelu i prototypu uniwersalnej architektury do prowadzenia złożonych obliczeń (w tym zaawansowanych pomiarów) w czasie rzeczywistym.

Słowa kluczowe: szybkie prototypowanie, sieci Petri, FPGA.

RAPID PROTOTYPING ENIVIRONMENT FOR DIAGNOSIS REAL TIME SYSTEMS

Summary

Rapid prototyping aims to reducing development cost via prototyping. Prototypes are built to assess whether proposed system will be acceptable to its user and whether a proposed design will provide adequate functionality and performance. A prototype is constructed prior to the system's production version to gain information that guides analysis and design. In this article is proposed RPPE (Rapid Prototyping Project Environment) for real-time embedded systems. This environment is based on modeling using new modification of Petri nets called hardware Petri nets. Implementation is made using special hardware architecture called dynamic reconfigurable internal architecture. The main advantage of proposed RPPE is that there is no need to change hardware, even if all used algorithms must be changed. There is a possibility to make all design process to be almost fully automated from modeling to working prototype.

Keywords: Rapid Prototyping, Petri nets, FPGA.

1. WPROWADZENIE

Prace prowadzone nad stworzeniem środowiska szybkiego prototypowania mają na celu redukcję kosztów związanych z przygotowaniem wersji produkcyjnych programów i urządzeń. Prototyp jest tworzony aby zdobyć wiedzę i umiejętności pozwalające na opanowanie i opracowanie efektywnego czasowo i kosztowo procesu wytworzenia wersji produkcyjnej produktu. Daje też możliwość zbadania funkcjonalności przyjętych rozwiązań i oceny zrealizowania założeń projektowych. Pracująca wersja programu czy urządzenia pozwala na sprawdzenie jej przez użytkownika końcowego oraz ustalenie jego opinii o proponowanych rozwiązaniach. Ich uwzględnienie w przygotowywanej wersji produkcyjnej stanowi klucz do sukcesu rynkowego lub pokazuje popełnione błędy i pozwala na dokonanie korekt nawet na poziomie podstawowych założeń. Szybkie prototypowanie to koncepcja, która „szybko tworzy prototyp", co oznacza, że zmiana założeń wyjściowych staje się zwykłym elementem procesu projektowego. Aby było to możliwe prototypowanie musi realizować dwa zasadnicze zadania. Pierwsze to stworzenie modelu, który ma jednoznacznie pokazać poprawność pracy na poziomie logicznym. Drugie to metoda sprawnej i bezbłędnej konstrukcji pracującego prototypu. Oba zadania powinny być realizowane w maksymalnym stopniu korzystać z komputerowego wspomagania projektowania. W prezentowanej pracy opracowano środowisko szybkiego prototypowania (RPPE – ang. Rapid Prototyping Project Environment) dla wbudowanych systemów czasu rzeczywistego (RTES ang. Real-Time Embedded Systems). Modelowanie oparto o sprzętowe sieci Petri [1] zaś wdrażanie o platformę sprzętową o dynamicznie rekonfigurowanej architekturze wewnętrznej. Dodatkowo opracowano założenia do stworzenia

w pełni automatycznego systemu implementującego sprawdzony model w układach FPGA.

2. METODA MODELOWANIA PRZY WYKORZYSTANIU ZMODYFIKOWANYCH SIECI PETRI

Pierwszym elementem składowym RPPE musi być sprawny mechanizm modelowania. Dzisiejsze metody diagnostyczne opierają się na sygnałach analogowych z czujników pomiarowych przetworzonych na informację cyfrową. Dalszy proces odbywa się na danych cyfrowych przy wykorzystaniu zaawansowanego aparatu matematycznego. Właściwe odwzorowanie zależności matematycznych za pomocą algorytmów obliczeniowych jest kluczem poprawnego działania systemu diagnostycznego. Poszukiwania i weryfikacja poprawności algorytmów to złożony proces i możliwy do szybkiego przeprowadzenia jedynie przy wykorzystaniu metod modelowania komputerowego. Biorąc pod uwagę implementację sprzętową prototypu jako metodę modelowania wykorzystano sieci Petri [4].

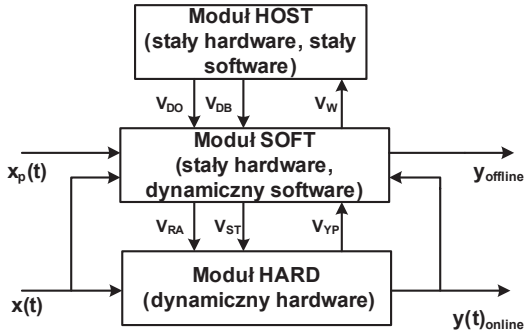
Znane na dzień dzisiejszy różne interpretacje poszerzania i modyfikacji sieci Petri pozwalają w zasadzie modelować równoległe procesy za pomocą środowiska programistycznego [2] systemów obliczeniowych (na wielu poziomach: od systemowego do mikroprogramowego). W takich podejściach należy wykluczyć krytyczne sytuacje w opracowywanych modelach. W przypadku pojawienia się dowolnej krytycznej sytuacji należy rozbudować funkcję środowiska w ten sposób, żeby w równoważnym modelu sytuacje krytyczne nie występowały. Brak takiej możliwości powoduje większą pracochłonność przy śledzeniu pracy algorytmu w innym urządzeniu, a co za tym idzie wielokrotność opracowywania i zmieniania modelu algorytmu. Uwzględniając wykonanie sprzętowe prototypu sieci Petri, powinny obowiązkowo uwzględniać występowanie krytycznych sytuacji. W celu budowy takich modeli proponuje się formalne opisanie modyfikowanych sprzętowych sieci Petri. Są to metody nie sieciowe pozwalające rozszerzać możliwości modelowanego sieciowego obiektu badań oraz przeprowadzić symulację i opracować wyniki. Proponowane rozwiązywanie pozwala na budowę adekwatnych modeli operacyjnych automatów cyfrowego przetwarzania sygnałów czasu rzeczywistego i przeprowadzić śledzenie działania różnorodnych algorytmów. W sytuacji kiedy mamy problem z realizacją jakiegoś elementu układu np. mnożenia i nie jesteśmy pewni czy operacja wykona się w założonym czasie, możemy zejść na niższy poziom detalizacji przebudowując i symulując sprzętową sieć Petri dla samego elementu. Ten proces można powtarzać, aż znajdziemy się na najniższym możliwym poziomie szczegółowości, np. bramki logicznej. Budowanie modeli dla poszczególnych elementów i zagłębianie się w kolejne poziomy szczegółowości realizowane jest za pomocą Wielopoziomowych Sprzętowych Sieci Petri [1]. Zejście na niższy poziom szczegółowości daje nam możliwość dokładniej kontroli poszczególnych sygnałów sterujących danymi elementami. Efektem eksperymentu mogą być tymczasowe diagramy zgrania wszystkich części oraz bezpośrednio cyfrowy wynik emitowanych obliczeń (czas cyklu sieciowego). Tak wygenerowany diagram czasowy pozwala na szybkie i proste zaimplementowanie zweryfikowanego algorytmu w struktury FPGA.

3. UNIWERSALNA ARCHITEKTURA IMPLENTACYJNA DRAW

Drugą podstawową składową RPPE jest część przenosząca sprawdzone logicznie algorytmy w pracujące urządzenia. Najistotniejszym elementem tego etapu jest osiągnięcie odpowiedniej mocy obliczeniowej przy minimalizacji środków finansowych oraz zautomatyzowania przejścia „od algorytmów do prototypu". Tradycyjne podejście polega na wyborze specjalistycznych rozwiązań sprzętowych dopasowanych do jednego konkretnego zagadnienia. Niestety, często się zdarza, że badania pracującego prototypu wskazują na konieczność zmian w sprzęcie. Naturalnie spowoduje to wzrost kosztów oraz czasu potrzebnego na modyfikacje. Tego typu postępowanie w oczywisty sposób nie przystaje do koncepcji szybkiego prototypowania. Proponowane rozwiązanie w/w problemu polega na zastosowaniu nowego podejścia do cech i wykonania części sprzętowej. Zaprojektowane RPPE opiera się na specjalnie do tego celu opracowanej platformie. Jest to zestaw środków sprzętowych, który dzięki odpowiednio zaplanowanym połączeniom pomiędzy nimi tworzy system klasyfikowany obecnie jako rekonfigurowalny system cyfrowego przetwarzania sygnałów. Został on nazwany platformą o dynamicznie rekonfigurowanej architekturze wewnętrznej (DRAW).

Platforma DRAW składa się z trzech modułów funkcjonalnych. Jej schemat blokowy przedstawiono na rys. 1. Realizację idei dynamicznie rekonfigurowalnej architektury wewnętrznej zapewniają moduły SOFT i HARD zaś moduł HOST stanowi klasyczny blok komunikacji z operatorem i innymi zewnętrznymi systemami.

Elastyczność wymaganą w środowisku szybkiego prototypowania zapewniają moduły SOFT i HARD. Moduł SOFT to możliwość zmiany realizowanych funkcji drogą klasycznej wymiany oprogramowania. Nową jakość wprowadza moduł HARD. Jest on oparty o logiczne podzespoły reprogramowalne typu FPGA.



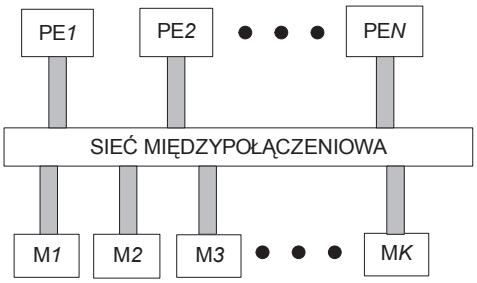
Rys. 1. Schemat blokowy platformy DRAW

Uruchomienie procesu obliczeniowego następuje na żądanie z zewnątrz (operator lub system komputerowy). W oparciu o ustawione zadania moduł HOST definiuje tryb pracy, który jest jednoznacznie określony za pomocą wektorów pracy: V_{DO} (wektor danych ogólnych) i V_{DB} (wektor danych binarnych). Wektor danych ogólnych V_{DO} zawiera informacje ogólne typu: rodzaj analizy do wykonania, zakres pomiarowy, sposób podziału na pasma, typy wyjść do których mają być przekazane wyniki. Dodatkowo definiuje wektor danych binarnych V_{DB} zawierający informacje szczegółowe typu: dynamika pomiarów, ilość pasm, ewentualnie zakresy poszczególnych pasm, rozdzielczość, ilość kanałów pomiarowych. Wektory V_{DO} i V_{DB} są przekazywane do modułu SOFT. W przypadku prostej korekty zadanych opcji wysyłany będzie tylko wektor V_{DB}. Drugą zasadniczą funkcją bloku HOST jest prezentacja rezultatów obliczeń otrzymywanych z modułu SOFT. Możliwa jest wizualizacja danych otrzymanych w postaci wektora wyników V_W za pomocą wewnętrznego układu wyświetlającego lub zewnętrznego systemu prezentacji w postaci obrazu czy wydruku. Istotną funkcją dodatkową musi być przekazywanie w postaci ciągu binarnego nowych wersji oprogramowania dla modułu SOFT i nowych architektur sprzętowych modułu HARD (wykorzystanego do tego celu wektory V_{DO} i V_{DB}). Ze względu, iż filozofia komunikacji z operatorem (bezpośredniej – za pomocą klawiatury lokalnej, pośredniej – poprzez system zewnętrzny) może być opracowana jednorazowo bez konieczności późniejszych zasadniczych modyfikacji, moduł HOST nie musi mieć zmiennej architektury i oprogramowania, czyli może być blokiem o stałym oprogramowaniu i stałym układzie sprzętowym. Dobrą propozycją układów do jego realizacji jest zastosowanie mikrokontrolera typu RISC 16/32 bitowego. Zapewni to najprostszą konstrukcję (o niskim koszcie), gdyż wystarczy jeden układ scalony zawierający oprócz mikroprocesora także układy peryferyjne, np. w postaci interfejsów komunikacyjnych, współpracy z różnymi rodzajami pamięci, bezpośredniej komunikacji z różnego typu wyświetlaczami, wewnętrzną pamięć RAM. Mają one również dużą wydajność pozwalającą na implementację wielu protokołów komunikacyjnych i dodatkową obróbkę otrzymanych wyników do postaci dopasowanej do zastosowanego układu wizualizacji.

Sedno architektury DRAW stanowią cechy funkcjonalne modułu SOFT. Po odebraniu wektorów pracy V_{DO} i V_{DB} dokonuje on autokonfiguracji oraz utworzenia wektorów konfiguracji V_{RA} i V_{ST} wysyłanych do modułu HARD, ustalając tym jego architekturę. Wektor rekonfiguracji architektury V_{RA} przekazuje informacje o architekturze wewnętrznej (programuje strukturę logicznych układów reprogramowalnych), zaś wektor stałych V_{ST} zawiera parametry pracy implementowanych struktur obliczeniowych. W zależności od zadanego trybu pracy, blok SOFT dokonuje wyboru stosowanego algorytmu zarówno użytego na potrzeby pomiaru w ramach własnego modułu jak i ustawianego w formie sprzętowej w module HARD. W sytuacji, gdy zadany tryb pracy oraz zastosowany algorytm nie wymagają dynamicznej zmiany, całość pracuje jak klasyczny statyczny system analizujący dane. Przykładem może być analiza częstotliwościowa w jednym zadanym przedziale częstotliwości. Niezbędna filtracja czy okienkowanie wykonywane jest sprzętowo w bloku HARD, zaś przetworzone dane w postaci wektora wyników pomocniczych V_{YP}, przekazane do bloku SOFT, gdzie wykonana zostanie FFT. W przypadku, gdy zadana analiza ma być prowadzona z podziałem pasmowym moduł SOFT wyznacza strukturę banku filtrów ustalając ilość wymaganych filtrów i decymacji oraz przesyłając odpowiednie wektory V_{RA} i V_{ST}. Ustalają one również z jakiego punktu przetwarzania będą przesyłane dane w postaci wektora V_{YP} (np. w celu diagnostyki poprawności pracy struktury obliczeniowej) i czy końcowe sygnały zostaną przekazane na bezpośrednie wyjście y_{online} lub powrotnie do modułu SOFT i wyprowadzone na zewnątrz w postaci $y_{offline}$ lub przekazane dalej do wizualizacji za pomocą bloku HOST. W przypadku trybu pracy dynamicznej dodatkową funkcją modułu SOFT jest śledzenie stanu sygnałów badanych $x(n)$, sygnałów z nimi skojarzonych $x_p(n)$ oraz wyników pośrednich V_{YP}. Polega ono na wyznaczaniu parametrów wymienionych wielkości wejściowych, a następnie wyliczeniu optymalnych współczynników pracy głównych algorytmów zaimplementowanych w strukturach sprzętowych HARD i przekazywanie ich za pomocą wektora V_{ST}. W efekcie takiego działania mamy dynamicznie zmieniany algorytm czasu rzeczywistego. W najbardziej zaawansowanym przypadku moduł SOFT może podjąć decyzje o zmianie lub wymianie całej struktury obliczeniowej zawartej w module HARD. Przykładem mogą być sytuacje gdy badane są sygnały wibroakustycznego zaczną wskazywać na niestabilną pracę i zachodzi konieczność przejścia do analizy w dziedzinie rzędów, czy też przestawienie ilości analizowanych pasm lub zmiana funkcji okienkowania sygnału.

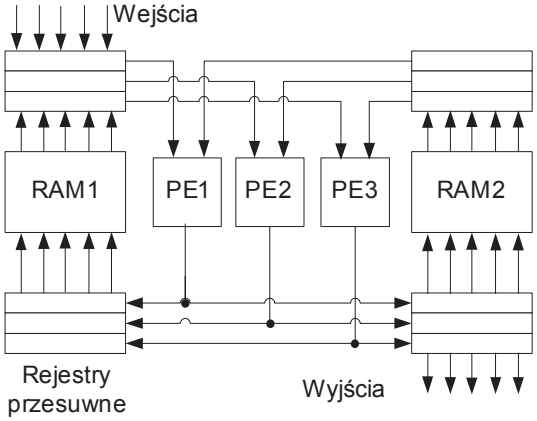
Rozpatrując blok SOFT z punktu widzenia realizacji fizycznej jedynym podzespołem nadającym się do jego konstrukcji są mikroprocesory klasy DSP. Umożliwiają one efektywną implementację algorytmów analizy sygnałów jak i algorytmów decyzyjnych. Ze względu, iż mają one niezmienną budowę wewnętrzną nie jest możliwe dopasowywanie struktury obliczeniowej do wykonywanego algorytmu ale możliwa jest stosunkowo prosta implementacja typowych zadań cyfrowego przetwarzania sygnałów (np. FFT) oraz bardzo złożonych i długich algorytmów. Zmiana sposobu pracy odbywa się drogą wymiany programu. Z jednej strony oznacza to łatwość wymiany funkcji bez konieczności zmian sprzętowych, z drugiej oznacza to jednak, że przy szerokim zestawie wykonywanych przekształceń sygnałów część z nich wykonywana jest ze zmniejszoną efektywnością. Uwzględniając złożoność funkcji wykonywanych w module SOFT należy przyjąć, że może wystąpić konieczność użycia więcej niż jednego mikroprocesora DSP. Ze względu na przyjętą koncepcję funkcjonalną bloku SOFT należy on do kategorii zespołu o stałej architekturze sprzętowej z możliwością dynamicznej zmiany oprogramowania.

Ostatnim elementem składowym architektury DRAW jest moduł HARD. Jego „kształt" jest ustawiany przez moduł SOFT, a zasadniczą funkcją efektywna realizacja algorytmów obliczeniowych czasu rzeczywistego. Efektywność ta oznacza konieczność zmiany architektury procesora obliczeniowego w zależności od zadanego algorytmu. Zgodnie z założeniem architektury DRAW zmiany takie muszą być możliwe do wykonania nawet w trakcie bieżącej pracy. Jedyną możliwość realizacji takiej koncepcji daje w chwili obecnej użycie reprogramowalnych układów logicznych typu FPGA lub CPLD. Typowe operacje o krytycznej ścieżce czasowej to filtracja z precyzyjnym podziałem pasmowym za pomocą banku filtrów, operacja splotu, przetwarzanie potokowe, część obliczeń niezbędnych przy analizie w dziedzinie rzędów. Układy programowalne umożliwiają precyzyjne dopasowanie struktury liczącej do ich specyfiki. Możliwe jest bowiem dobranie długości słowa, zmiana ilości ogniw filtracji i decymacji, pomijania niepotrzebnych w danej chwili bloków funkcjonalnych. Wewnętrzne struktury tego modułu są zorganizowane w architekturze o współdzielonej pamięci z zespołem elementów wykonawczych [4]. Architektura ze współdzieloną pamięcią pokazana na rys. 2. Posiada procesory z wieloma (K) wejściami danych, współdzieloną pamięć podzieloną na (K) oddzielnych pamięci. Używając powtarzanego schematu dostępu, każdy procesor posiada dostęp do pamięci z każdym N cyklem, w którym każdy procesor zapisuje wyniki do jednej lub wielu pamięci, czy też odczytuje równolegle maksymalnie jedną wartość z każdej pamięci. Ten prosty schemat gwarantuje brak konfliktów dostępu. System jest ściśle sprzężony tak, że wszystkie procesory mają ten sam czas dostępu do pamięci. Tego typu architektura doskonale nadaje się do zastosowań związanych z algorytmami DSP, np. algorytmy rekursywne ze skomplikowanymi zależnościami danych. Wadą tego typu architektury jest możliwość zastosowania niewielkiej liczby procesorów. Ma to związek z wąską szerokością pasma do pamięci, dlatego w takich sytuacjach stosuje się łączenie ze sobą kilku takich mini systemów w sposób równoległy lub kaskadowy.



Rys. 2. Architektura wieloprocesorowa

Takie podejście pozwala na wielokrotne i szybkie zmiany sposobu implementacji algorytmów. Działający prototyp po zebraniu informacji od użytkowników końcowych można w bardzo krótkim czasie przystosować zgodnie i ich uwagami i sugestiami. Możliwe są również korekty pozwalające na podniesienie wydajności drogą zrównoleglenia operacji (angażując więcej elementów wykonawczych) lub zasadniczej zmiany struktury połączeń elementów wykonawczych. Rysunek 3 ilustruje przykład jednego z rozwiązań balansowania struktury pomiędzy ilością pamięci a ilością elementów wykonawczych.



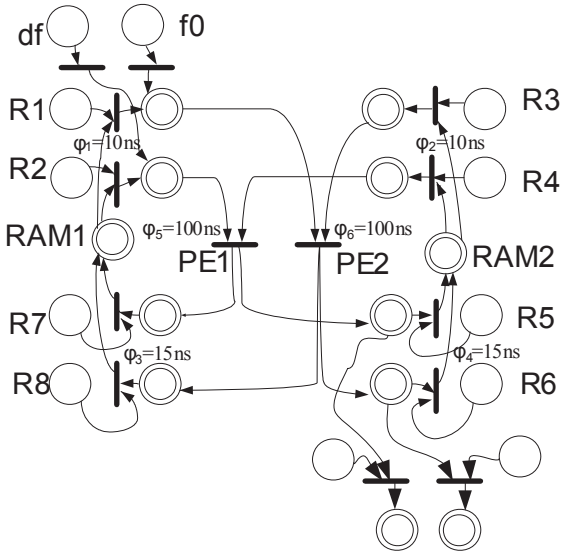
Rys. 3. Architektura obliczeniowa ze współdzieloną pamięcią

Podsumowując można stwierdzić, że zaproponowana architektura DRAW-daje możliwość szybkiej budowy prototypów urządzeń diagnostycznych pracujących w czasie rzeczywistym. Jest koncepcją otwartą pozwalająca na szybkie i daleko idące zmiany funkcjonalności bez potrzeby przeprojektowywania konstrukcji

urządzenia. Dzięki możliwości łatwego i szybkiego przestawienia realizowanych algorytmów może być cennym składnikiem środowiska szybkiego prototypownia. [7],[8].

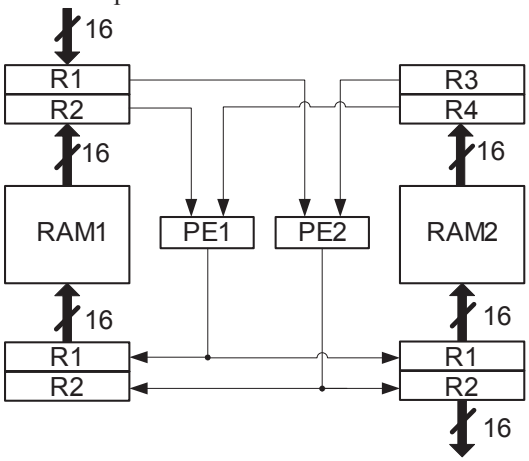
4. EKSPERYMENTALNA REALIZACJA RPPE DO DIAGNOSTYKI WIBROAKUSTYCZNEJ

W celu weryfikacji koncepcji RPPE opracowano implementację algorytmu TVDFT [5] do diagnostyki wibroakustycznej. W procesie projektowym zastosowano opracowane postępowanie szybkiego prototypowania. Modelowanie algorytmu wykonano za pomocą sprzętowych sieci Petri. Oprogramowanie modelujące zrealizowano w technologii **C++** na komputery osobiste. Narzędzie programowe skonstruowano w sposób umożliwiający wykorzystanie wielopoziomowych sieci Petri. Modelowanie projektowanych algorytmów można zacząć od ogólnego funkcjonalnego poziomu. Następnie każda ze złożonych operacji jest dzielona na mniejsze fragmenty. Końcowy stopień szczegółowości modelowania obejmuje poziom rejestrów, elementów wykonawczych (np. układów mnożących o dobieranej długości słowa) elementów sterujących oraz sekwencji czasowej sygnałów sterujących. Jest to zatem gotowy i sprawdzony logicznie projekt układu realizującego zadane algorytmy. Rysunek 4 ilustruje model sieci Petri elementu wykonawczego obliczającego składową harmoniczną sygnału wibroakustycznego. Natomiast rys. 5 jest to dopasowana zbalansowana architektura elementu wykonawczego pozwalająca na kompromis pomiędzy wykorzystaniem przestrzeni w układzie programowalnym a poborem mocy oraz szybkością pracy.

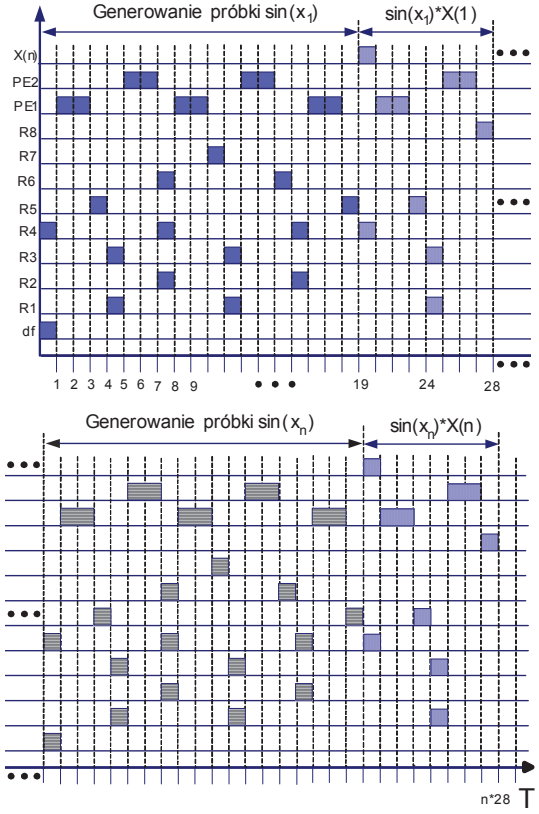


Rys. 4. Model sieci Petri jednostki wykonawczej

Diagram czasowy przedstawiony na rys. 6 przedstawia kolejność wykonywania operacji projektowanego elementu wykonawczego. Mając taki diagram zbudowanie układu w strukturze FPGA jest bardzo proste.



Rys. 5. Architektura obliczeniowa ze współdzieloną pamięcią dla TVDFT



Rys. 6. Zestawienie czasowe wykonywania poszczególnych operacji algorytmu TVDFT

Kolejny etap to implementacja sprzętowa. Bazą elementową musiał być zestaw podzespołów umożliwiający stworzenie architektury DRAW. Skonstruowanie w pełni funkcjonalnej platformy zgodnej z architekturą DRAW to zadanie bardzo złożone W związku z tym zdecydowano się na tym etapie prac na weryfikację tylko niektórych elementów. Skoncentrowano się na sprawdzeniu realizowalności koncepcji działania modułu HARD i SOFT. Jako moduł HOST wykorzystano komputer osobisty. (w oczywisty sposób pozwala realizować komunikację z operatorem oraz prezentować wyniki pomiarów i obliczeń). Jako bazę modułów SOFT

i HARD wykorzystano platformę sprzętową SignalWave [6] firmy Lyrtech. Zawiera ona połączone wewnętrzną magistralą podzespoły mikroprocesorowe Texas Instruments (DSP 6713), układy FPGA Xilinx (VirtexII - XC3000). oraz wejścia analogowo-cyfrowe. Zbiór podzespołów mikroprocesorowych tworzy moduł SOFT, zaś układów FPGA moduł HARD. Oprogramowanie dostarczone razem ze sprzętem zapewnia sprawną komunikację z komputerem osobistym. Daje ono możliwość załadowania oprogramowania użytkowego do systemu mikroprocesorowego oraz struktur wewnętrznych układów FPGA. Posiada też funkcje uruchomienia i obserwacji pracy systemu w czasie rzeczywistym. Dotychczasowe prace przy wykorzystaniu w/w platformy potwierdziły możliwość wykonania szybkiej implementacji sprzętowej struktur obliczeniowych opracowanych w wyniku modelowania z wykorzystaniem sprzętowych sieci Petri.

Doświadczenia przy implementacji algorytmu TVDFT wskazują że, wszelkie zmiany czy poprawki mogą być wykonywane bez konieczności wymiany sprzętu. Przeprowadzone próby i eksperymenty z zastosowaniem elementów architektury DRAW oraz mechanizmu współdzielonej pamięci jako jądra konstruowania specjalizowanych struktur obliczeniowych w układach programowalnych pokazały, że umożliwia jest budowa systemu RPPE do celów diagnostycznych dla wszystkich kategorii rozwiązań real-time embedded systems.

LITERATURA

[1] Rubin G, Petrovsky A.Omieljanowicz M.: *Multilevel hardware Petri nets for rapid prototyping design platform.* In the proc. of the 12th Intern. Conference Mixed design on integrated circuits and systems, MIXDES' 2005, Kraków, Poland, 20-25 June 2005, 147-152.

[2] Auguin M., Boeri F.: *A method for designing special fast systems for signal processing.* Proc. Signal processing: Theories and applications. North-Holland publishing company. 1980. P.239-244.

[3] Wanhammar L.: *DSP integrated circuits*, Academic Press, USA,1999.

[4] Peter H. Starke.: *"Sieci Petri"*, PWN, Warszawa, 1987.

[5] Zubrycki P., Rubin G., Piotrowski A., Omiljanowicz M.: *Procesor czasu rzeczywistego do prowadzenia diagnostyki w dziedzinie rzędów oparty na układach typu FPGA.* DIAG 2003', Ustroń, 2003.

[6] http://www.lyrtech.com

[7] Jones A. B., Cavallaro J. R: A *Rapid Prototyping Environment for Wireless Communication Embedded Systems.* EURASIP Journal on Applied Signal Processing, 1 May 2003, Vol.6, s. 603-614.

[8] Richards M. A.: *The Rapid Prototyping of Application Specific Signal Processors Program: Overview and Accomplishments*. Proceedings 1st Annual RASSP Conference, August 1994, s. 1-8.

Mgr inż. **Grzegorz RUBIN** jest pracownikiem Wydziału Informatyki Politechniki Białostockiej w Katedrze Systemów Czasu Rzeczywistego. Zajmuje się realizacją algorytmów cyfrowego przetwarzania sygnałów wibroakustycznych w strukturach FPGA oraz modelowaniem za pomocą sieci Petri.

Dr inż. **Mirosław OMILJANOWICZ** jest pracownikiem Wydziału informatyki Politechniki Białostockiej w Katedrze Systemów Czasu Rzeczywistego. Zajmuje się architekturą systemów komputerowych czasu rzeczywistego, implementacją algorytmów cyfrowego przetwarzania sygnałów wibroakustycznych w strukturach FPGA oraz procesorach DSP.