Sylwester GARNCAREK, **Bogdan OLECH**
GRYFTEC EMBEDDED SYSTEMS, ul. Juliana Tuwima 21, 71-426 Szczecin
ZACHODNIOPOMORSKI UNIWERSYSTET TECHNOLOGICZNY W SZCZECINIE, WYDZIAŁ INFORMATYKI, ul. Żołnierska 49, 71-210 Szczecin

Integracja narzędzi przetwarzania sygnałów pakietu MATLAB/Simulink oraz środowiska CoDeveloper w projektowaniu aplikacji wbudowanych z użyciem FPGA

Mgr inż Sylwester GARNCAREK

W roku 2009 uzyskał tytuł magistra oraz inżyniera na Wydziale Informatyki Zachodniopomorskiego Uniwersytetu Technologicznego. Od roku 2004 jest zatrudniony w firmie Gryftec Embedded Systems, obecnie na stanowisku menedżera projektu. Jego główne obszary zainteresowań, to sprzętowe testery oraz systemy operacyjne dla urządzeń wbudowanych.

e-mail: sylwester.garncarek@gmail.com

Dr inż. Bogdan OLECH

Uzyskał tytuł magistra oraz inżyniera na Politechnice Szczecińskiej w roku 1986, stopień Doktora uzyskał w roku 1999 na Politechnice Warszawskiej. Obecnie jest adiunktem na Wydziale Informatyki Zachodniopomorskiego Uniwersytetu Technologicznego. Współpracuje z firmami innowacyjnymi. Jego zainteresowania naukowe, to: metodologia konstruowania technicznego, obliczenia hybrydowe, przetwarzanie sygnałów elektromagnetycznych oraz mechatronika.

e-mail: bolech@wi.zut.edu.pl

Streszczenie

Celem prezentowanej pracy była analiza wykonalności, a następnie zaprojektowanie, implementacja i przetestowanie oryginalnego oprogramowania wpierającego proces projektowania systemów wbudowanych. Chodziło o możliwość połączenia zalet oprogramowania narzędziowego szybkiego wytwarzania aplikacji rekonfigurowalnych (CoDeveloper; Impulse) z możliwościami narzędzi graficznego projektowania i obrazowania (MATLAB/Simulink). Założony cel został osiągnięty.

Słowa kluczowe: rekonfigurowalne tablice logiczne, systemy wbudowane, procesor konwergentny.

Integration of MATLAB/Simulink signal processing with CoDeveloper FPGA C based programming tool for embedded system design

Abstract

Computation platforms of embedded systems evolved, from being based on traditional microcontrollers and signal processors separately, to complex systems built with FPGA devices [1, 2]. However, complexity of hardware description languages and hardware skills requirements were the major limitations of generalization of these naturally evolved platforms in embedded systems design. The solution was to create tools that could give software developers a possibility to use the potential of programmable logic devices without considerable knowledge of hardware and hardware description languages [5]. This work presents an idea of bridging two different design tools; CoDeveloper [9] and MATLAB/Simulink [8] in context convergent processor paradigm [3, 4]. As the result, virtually new environment (Integrating Application) for development, simulation and verification of digital signal processing algorithms in embedded system domain was created. Fig. 1 shows the block diagram of Integrating Application structure in which it is possible to exchange data in real time with reconfigurable embedded system hardware being under development. Connection of these two tools allows getting additional features that support development process, and which are beyond the basic sum of features originally offered by these tools separately. Smulink models can represent Impulse C models, to mention the one characteristic.

Keywords: FPGA, embedded systems, convergent procesor.

1. Wstęp

Mikrokontrolery w systemie wbudowanym, oprócz automatyzacji zadań kontroli i sterowania, mogą zajmować się również szeroko pojętym przetwarzaniem danych, w szczególności zadaniami cyfrowego przetwarzania sygnałów (DSP – *Digital Signal Processing*). W przypadku, gdy zadania DSP stawiane przed mikrokontrolerem nie są intensywne obliczeniowo istnieje szansa, że projektanci systemu będą w stanie przeznaczyć odpowiednio dużo czasu procesora również dla zadań kontrolnych. Gdy wymagania, co do wydajności rosną pojawiają się sie problemy, których nie można rozwiązać przy użyciu danego mikrokontrolera. Rozwiązaniem może być zastosowanie drugiego mikrokontrolera, który przejmie cześć zadań. Istnieją jednak zadania DSP, którym nie będzie mógł podołać nawet wydzielony, dodatkowy mikrokontroler. Powstaje wtedy konieczność użycia procesora sygnałowego współpracującego z mikrokontrolerem [1].

Dla cyfrowego przetwarzania sygnałów procesor DSP sprawdza sie znakomicie, jednak nie jest on w stanie pracować, jako samodzielny procesor w systemie wbudowanym. Jest on zoptymalizowany do zadań matematycznych nie do kontrolowania, np. pamięci operacyjnej pod kątem zadań systemu operacyjnego [2].

Mikrokontrolery i procesory sygnałowe tworzą środowisko heterogeniczne, w którym nie ma możliwości przenoszenia kodu już na etapie działania systemu, tzn. kod jest podzielony na stałe pomiędzy procesory. Jest to istotne ograniczenie współpracy tych procesorów, które nie pozwala jednemu z procesorów przejąć części zadania drugiego, gdy jest on przeciążony. Pochodny problem z tym związany pojawia się, gdy z pewnych względów wymagania stawiane przed systemem wbudowanym wzrastają, np. pojawi sie nowsza wersja algorytmu kompresji wideo.

W dominującym aktualnie modelu wytwarzania systemów wbudowanych projektanci dzielą sie na dwa zespoły, z których każdy zajmuje sie oddzielnym aspektem systemu, w tym przypadku; sterowanie i przetwarzanie sygnałów. Zespoły te spotykają się jedynie w momencie, gdy ustalają interfejs pomiędzy tymi dwoma odmiennymi częściami systemu. Taka sytuacja uwstecznia uznane podejście w projektowaniu zakładające współbieżność działań projektowych. Jednak podział na dwa zespoły wydaje sie naturalny - jeśli wziąć pod uwagę koszt pozyskania z rynku pracy i utrzymania specjalistów o szerokim spektrum kwalifikacji; systemy operacyjne dla procesorów wbudowanych, znajomość urządzeń peryferyjnych, oraz zagadnień cyfrowego przetwarzania sygnałów.

Nie bez znaczenia jest tez koszt zakupu i utrzymania narzędzi programistycznych w przypadku dwóch różnych architektur procesorów. Na rynku wśród producentów procesorów pojawia się jednak trend do udostępniania narzędzi na zasadzie darmowego oprogramowania (*freeware*) lub za symboliczna opłatą, w porównaniu do wielkości planowanej produkcji.

Wadą stosowania dwóch procesorów jest wzrost rozmiarów obwodów drukowanych i konieczność zapewnienia szybkiej komunikacji pomiędzy tymi procesorami. Rozwiązaniem może być stosowanie dwóch lub więcej procesorów zintegrowanych w jednym układzie scalonym (np. rodzina procesorów OMAP firmy Texas Instruments), jednak z drugiej strony ogranicza to możliwość wyboru architektur do predefiniowanych przez producenta połączeń funkcjonalnych.

Prezentowana praca podejmuje próbę praktycznego wsparcia szeregu istotnych postulatów dotyczących aktualnie konstruowanych systemów wbudowanych oraz metod i narzędzi pomocnych w tym konstruowaniu. Rozważania wstępne wskazują na problem jednoczesnego występowania w aplikacjach wbudowanych zakresu typowych działań kontrolnych i sterujących wraz z działaniami charakterystycznymi dla cyfrowego przetwarzania sygnałów. Odpowiedzią na ten problem ma być procesor konwergentny, scharakteryzowany w punkcie 2.

Punkt 3 odnosi się do coraz bardziej dominującej roli technologii FPGA w konstruowaniu systemów wbudowanych. Omawia również aspekt metodologiczny wynikający ze specyfiki języków opisu, niezbędnych z uwagi na automatyzację syntezy struktur umieszczanych w FPGA.

Punkt 4 omawia realizację sprzęgu (aplikacji sprzęgającej) pozwalającej na wspieranie postulatu konwergentności architektury obliczeniowej. Jednocześnie, zapewnia poszerzenie możliwości znanych metod i narzędzi projektowania (MATLAB/Simulink oraz CoDeveloper) o nowe, niewykorzystywane dotychczas w praktyce konstruowania. Następuje przenikanie się graficznego projektowania Simulink oraz konwersji formalnego podejścia MATLAB z opisem działań na bazie języka C, charakterystycznego dla CoDeveloper.

Walor aplikacji sprzęgającej wiąże się również z tym, że zapewnia ona możliwość jednoczesnej pracy ze sprzętem, dając możliwość wymieniania bloków pomiędzy symulacją, rzeczywistym działaniem. Wnioski końcowe zawiera punk 5.

2. Procesor konwergentny

W odpowiedzi na wady stosowania dwóch różnego typu procesorów (nawet w jednym układzie scalonym) wymienione wcześniej powstała koncepcja procesora konwergentnego. Procesor konwergentny jest zoptymalizowany jednocześnie pod katem zadań kontrolnych, jak i przetwarzania sygnałów [3, 4]. Model pamięci procesorów konwergentnych zapewnia dużą elastyczność, np. procesory Blackfin pozwalają na konfigurowanie pamięci, jako korzystającej z pamięci podręcznej (jak w procesorach ogólnego przeznaczenia), jako statycznej (jak w procesorach sygnałowych) lub jako połączenie obydwu trybów tak, aby zapewnić wydajność zadaniom kontrolnym i przewidywalność czasową algorytmom DSP.

Innym przykładem architektury konwergentnej jest procesor StarCore SC140 firmy Freescale. Procesor ten łączy w sobie cztery jednostki arytmetyczno-logiczne, dwie jednostki generowania adresów i jednostkę sterowania przepływem programu. Zastosowanie powielonej liczby jednostek oraz instrukcji o zmiennej długości słowa VLIW (*Variable Lenght Instruction Word*) pozwala na osiągnięcie wysokiego stopnia równoległości obliczeń, przekłada sie to na bardzo duży wzrost wydajności.

Lista instrukcji procesora SC140 oraz wydajny kompilator optymalizujący, generujący równoległe instrukcje o wysokiej gęstości, pozwalają na budowanie oprogramowania w języku C oraz zachowanie przenoszalności kodu.

Procesor konwergentny upraszcza proces produkcji urządzenia. Programiści mogą nauczyć się jednego zestawu instrukcji oraz utrzymywać i rozwijać tylko jedno środowisko projektowe. Nie ma potrzeby tworzenia interfejsu pomiędzy dwoma procesorami. Jest tylko jeden system operacyjny, jedno API (*Application Programming Interface*), jeden zestaw sterowników, jeden sposób wyszukiwania błędów, jeden program ładujący, jeden język i kompilator.

Dzięki posiadaniu sprzętowego stosu i kontrolera przerwań z priorytetowaniem źródeł, możliwe jest użycie systemu operacyjnego. Dzięki temu, zadania kontrolne i przetwarzania sygnałów mogą być wykonywane na jednym procesorze przy jednoczesnym spełnieniu ich wymagań, takich jak reakcja w czasie rzeczywistym.

3. FPGA jako uniwersalna platforma aplikacji wbudowanych

Dynamiczny rozwój układów FPGA (bezpośrednio programowalna macierz bramek; *Field-programmable Gate Array*) oraz spadek ich cen spowodował, że są one coraz powszechniej stosowane w aplikacjach systemów wbudowanych. Rozwój ten jednak w pewnym momencie zaczął być ograniczany przez wymaganie znajomości od projektantów języków opisu sprzętu. Języki te są trudne i nie ułatwiają uzyskania odpowiednio wysokiego poziomu abstrakcji przy projektowaniu i implementacji aplikacji dla systemów wbudowanych.

Odpowiedzią na ten problem jest projektowanie i weryfikacja na poziomie systemu elektronicznego ESL (*Electronic System Level*). Głównym celem ESL, w rozważanej tutaj dziedzinie projektowania, jest umożliwienie projektantom z umiejętnościami jedynie programistycznymi, łatwej implementacji rozwiązań w układach reprogramowalnych, bez potrzeby nauki technik projektowania aplikacji sprzętowych. Założeniem jest, aby możliwe było modelowanie zachowania całych systemów przy użyciu języków wysokiego poziomu takich, jak C, C++. Większość narzędzi opartych na idei ESL korzysta z SystemC lub modyfikacji języka ANSI C, jako abstrakcyjnego języka modelowania systemów [5]. Narzędzia te, to m.in. CatapultC firmy Mentor Graphics, Handel-C firmy Celoxica, czy GreenSocs (dostępny na zasadzie otwartych źródeł). Wymienione narzędzia, to zaledwie ułamek wszystkich, dostępnych na rynku narzędzi ESL.

Aby wykorzystanie układów FPGA mogło sie upowszechnić pojawiła się konieczność zbudowania narzędzi umożliwiających automatyczną, lub wymagająca minimalnej ingerencji, konwersję z języków sekwencyjnych do języków opisu sprzętu. Warunkiem jest również zminimalizowanie koniecznej wiedzy o zagadnieniach związanych ze sprzętem, wymaganej od użytkownika. Ponieważ języki sekwencyjne nie maja możliwości wyrażania równoległości, musi sie to odbywać poprzez użycie pewnego modelu programowania, który definiuje, w jaki sposób program sekwencyjny musi zostać napisany, aby narzędzie konwersji było w stanie przekształcić kod sekwencyjny do postaci zrównoleglonej [6].

W świetle dostępności narzędzi konwersji z języków sekwencyjnych do języków opisu sprzętu oraz narzędzi wspomagania projektowania algorytmów, pojawia sie pytanie: czy możliwe jest połączenie tego typu narzędzi w taki sposób, aby suma ich możliwości dawała dodatkowe korzyści w stosunku do ich oddzielnego użycia.

4. Jednoczesne użycie MATLAB/Simulink oraz CoDeveloper

Do narzędzi ESL można również w obecnej chwili zaliczyć pakiet MATLAB/Simulink. Kojarzony początkowo z zastosowaniami akademickimi MATLAB stopniowo zwiększał swoje możliwości, powoli wkraczając w sektor przemysłu. Obecnie, MATLAB to rozbudowane narzędzie oferujące bogaty zestaw bibliotek z różnych dziedzin nauki i techniki. Pakiet MATLAB znajduje również zastosowanie w systemach wbudowanych [7]. Umożliwia on np. generowanie kodu C ze skryptów napisanych w języku MATLAB. Przykładowe wspierane architektury systemów wbudowanych to m.in. Motorola MPC555, Texas Instruments C2000, czy Infineon C166. Możliwe jest również, za pośrednictwem narzędzi takich jak Xilinx System Generator generowanie kodu w języku opisu sprzętu, na podstawie systemów modelowanych w języku MATLAB, jak również na podstawie modeli Simulink [8].

Spośród narzędzi ESL, ze względu na prostotę modelu programowania i niska cenę, na uwagę zasługuje oprogramowanie CoDeveloper firmy Impulse Accelerated Technologies. Zastosowany w nim, bazujący na ANSI C, język Impulse C opiera sie na zastosowaniu równolegle wykonujących się procesów, komunikujących się za pomocą tzw. strumieni [9]. Wadą tego narzędzia, która jest bezpośrednią konsekwencją niskiej ceny, jest ograniczo-

ność bibliotek funkcji oraz niewielkie możliwości wizualizacji modeli, czy podglądania danych przepływających pomiędzy procesami. Wada ta mogłaby być zniwelowana poprzez zastosowanie narzędzia takiego, jak MATLAB/Simulink, z którym modele te mogłyby współpracować. Zgodnie z oficjalną informacją firmy Impulse Accelerated Technologies, potwierdzoną również w bezpośrednich kontaktach, w obecnej chwili nie istnieje narzędzie umożliwiające sprzęganie w wygodny i automatyczny sposób tych dwóch środowisk. Firma proponuje ręczną ingerencję w kod celem użycia modelu Impulse C, jako całości lub części większego systemu w programie MATLAB/Simulink.

Idea taka, wykonanego w zakresie projektu, polega na takim połączeniu środowisk Impulse C i Simulink, aby możliwy był bezpośredni dostęp z poziomu Simulink do danych przepływających przez połączenia pomiędzy procesami. Jednocześnie, pożądana jest możliwość dowolnego wyboru lokalizacji każdego z procesów tak, aby mogły wykonywać się w sposób przezroczysty zarówno na komputerze PC (symulacja), jak i w docelowym systemie wbudowanym. Nie powinno mieć tu znaczenia, czy dany proces jest typu programowego, czy sprzętowego.

Warunkiem jednoczesnego użycia pakietu MATLAB/Simulink oraz modeli napisanych w języku Impulse C jest konieczność znalezienia sposobu reprezentacji elementów modelu programowania języka Impulse C w środowisku Simulink. Konieczne jest też zapewnienie komunikacji pomiędzy środowiskiem Simulink, a modelem Impulse C umożliwiającej podgląd i modyfikowalność danych przepływających pomiędzy procesami.

Aby sprzęg pomiędzy pakietem MATLAB Simulink i Impulse C był funkcjonalny musi spełniać poniższe założenia:

Przezroczystość sprzęgu - oznacza to, że sprzęg nie powinien mieć wpływu na funkcjonowanie modelu chyba, że będzie to świadome działanie użytkownika.

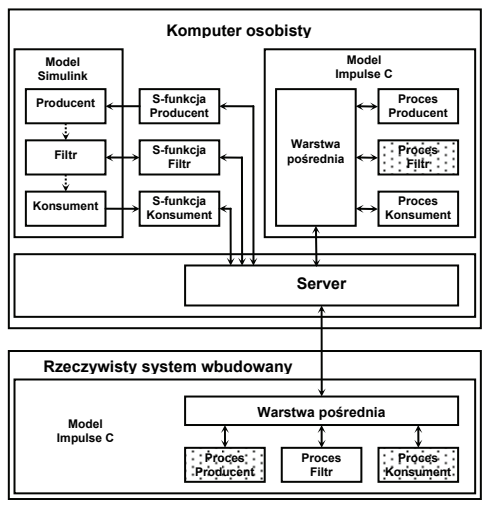
Maksymalne pokrycie modelu programowania Impulse C - sprzęg powinien obsługiwać jak najwięcej elementów modelu programowania Impulse C bez wpływania na ich sposób działania.

Dowolność lokalizacji - powinna istnieć możliwość dowolnego podziału elementów modelu Impulse C pomiędzy symulacje na komputerze PC, a rzeczywisty system wbudowany zbudowany z użyciem FPGA.

Wparcie dla różnych architektur systemu wbudowanego - sprzęg powinien działać na różnych architekturach systemów wbudowanych, a dodawanie kolejnych powinno być maksymalnie ułatwione.

Po przeanalizowaniu charakterystyki środowisk MATLAB/Simulink oraz Impulse C można stwierdzić, że możliwe jest jednoczesne użycie tych dwóch narzędzi oraz, że założenia funkcjonalne mogą być zrealizowane. Do reprezentacji procesów modelu Impulse C mogą zostać użyte bloki S-funkcji Simulink. Natomiast, do reprezentacji strumieni i sygnałów można użyć połączeń, które w Simulink noszą nazwę ”sygnały” i służą do łączenia pomiędzy sobą bloków modelu Simulink. Przezroczystość sprzęgu można zapewnić poprzez zastosowanie takich samych typów danych w modelu Simulink, jak te przepływające w strumieniach i sygnałach modelu Impulse C. Dowolność lokalizacji można zrealizować stosując niezależne, równoległe wykonywanie sie procesów Impulse C oraz wykorzystując architekturę sprzęgu typu klient-serwer. Klientem jest każdy z procesów, a dane przepływają pomiędzy procesami za pośrednictwem modelu Simulink, wiec lokalizacja procesu jest bez znaczenia dla modelu Impulse C, jako całości. Ostatnie założenie dotyczące wsparcia różnych architektur systemów wbudowanych można zrealizować poprzez generowanie warstwy pośredniej komunikacji klientów z serwerem na podstawie szablonu, który może być dostosowywany do nowych architektur.

Rys. 1 przedstawia schemat sprzęgu dla przykładowego modelu Impulse C (Filtr). Model ten wykonuje sie jednocześnie na komputerze osobistym (symulacja), jak i w systemie wbudowanym (sprzęt). Zaznaczone (kropki) procesy są ignorowane przez serwer, który posiada informacje o podziale procesów pomiędzy symulacje a system wbudowany. Każdy model komunikuje się z serwerem poprzez warstwę pośrednią, która jest dostarczana w postaci kodu za pośrednictwem aplikacji sprzęgającej. Zadaniem warstwy pośredniej jest przesyłanie danych pomiędzy serwerem, a poszczególnymi procesami. Bloki w modelu Simulink, które są odpowiednikami procesów w modelu Impulse C, komunikują sie z serwerem poprzez S-funkcje. S-funkcje są generowane przez aplikacje sprzęgającą, kompilowane przez kompilator mex do postaci bibliotek dynamicznych, a następnie w momencie rozpoczęcia symulacji, dołączane przez Simulink do modelu. Przerywane linie w modelu Simulink przedstawiają połączenia pomiędzy blokami w Simulink reprezentującymi procesy. Strzałki pomiędzy tymi procesami i S-funkcjami pokazują rzeczywisty kierunek przepływu danych.



Rys. 1. Schemat struktury powiązań dla aplikacji sprzęgającej środowiska MATLAB/Simulink, CoDeveloper oraz rzeczywistego systemu wbudowanego

Fig. 1. Block diagram of Integrating Application structure with MATLAB/Simulink, CoDeveloper and embedded system hardware

Poniższa lista przedstawia korzyści, jakie daje jednoczesne użycie pakietu MATLAB/Simulink i CoDeveloper:

Podgląd sygnałów. Aplikacja CoDeveloper udostępnia funkcje monitorowania modelu Impulse C (*Application Monitoring*). Jednak nie jest tam możliwy podgląd danych przepływających przez strumienie, a jedynie ilość danych przez nieprzesłanych i aktualnie wykorzystywana pojemność strumienia. Dzięki sprzęgowi dane te dostępne są z poziomu Simulink, tam mogą być w łatwy sposób wizualizowane, np. na wykresach, czy zachowywane w zmiennych przestrzeni roboczej MATLAB. Dane przed wizualizacja mogą być dodatkowo przetwarzane przez odpowiednie bloki Simulink lub funkcje MATLAB.

Modyfikowalność sygnałów. Oprócz podglądu sygnałów w Simulink, sygnały mogą być również modyfikowane w trakcie wykonania modelu Impulse C. Bez wykorzystania sprzęgu jest to jedynie możliwe poprzez modyfikacje wprowadzane bezpośrednio w kodzie Impulse C.

Usuwanie zbędnych procesów. Procesy modelu Impulse C mogą być usuwane z poziomu Simulink. Oznacza to, że próby zapisu lub odczytu strumieni dokonywane z poziomu procesu są ignorowane.

Użycie modelu Impulse C, jako bloku w Simulink. Aplikacja sprzęgająca umożliwia użycie modelu Impulse C, jako części większego systemu w Simulink. Umożliwia to np. testowanie i weryfikacje działania modelu Impulse C za pomocą wirtualnego środowiska testowego (*Test-bench*).

Podział procesów Impulse C pomiędzy symulacje i system wbudowany. Modele Impulse C nie posiadają możliwości wykonywania procesów w środowisku mieszanym, tzn. nie jest możliwe wykonywanie części procesów w symulacji na komputerze PC, a pozostałych w rzeczywistym systemie wbudowanym. Aplikacja sprzęgająca umożliwia taki scenariusz, gdyż każdy z proce-

sów wykonuje sie faktycznie oddzielnie, a wszystkie dane w strumieniach przesyłane są za pośrednictwem serwera.

Dostęp do procesów w systemie wbudowanym z poziomu Simulink. Aplikacja sprzęgająca umożliwia dostęp do procesów wykonywanych w zdalnym systemie wbudowanym. Oznacza to, ze np. dane z przetwornika analogowo-cyfrowego mogą być przekazywane bezpośrednio do Simulink, który może poddawać je dalszej obróbce lub wizualizacji z szybkością ograniczoną do wydajności komunikacji pomiędzy modelem Impulse C, a serwerem wbudowanym w aplikację sprzęgającą.

Powyższe korzyści pokazują, ze jednoczesne użycie MATLAB i modeli Impulse C może w znaczący sposób podnieść efektywność cyklu projektowania. Korzyści takie, jak podział procesów Impulse C pomiędzy symulacje i rzeczywisty system wbudowany, dzięki zastosowaniu aplikacji sprzęgającej staja sie możliwe.

W obecnej wersji aplikacja sprzęgająca obsługuje tylko procesy i strumienie w modelu Impulse C. Ponieważ cała komunikacja pomiędzy procesami realizowana jest za pośrednictwem serwera, aplikacja sprzęgająca nie tylko nie obsługuje elementów modelu programowania innych niż procesy i strumienie, ale również nie pozwala na ich obecność w modelu. Z uwagi na fakt, ze procesy i strumienie są fundamentalnymi elementami modeli Impulse C można stwierdzić, że taki zakres realizacji pozwala na użycie aplikacji do sprzęgania większości modeli Impulse C z programem Simulink. Otwartość zastosowanego protokołu komunikacji opartego na liście komend pozwala na rozszerzanie możliwości sprzęgu o dodatkowe elementy modelu programowania Impulse C. Głównym problemem do rozwiązania jest tutaj odbiór i wysyłanie danych do procesów poprzez porty wejścia-wyjścia takie, jak sygnały, czy rejestry oraz sposób ich prezentacji w modelu Simulink.

Przydatność aplikacja sprzęgającej została przebadana na podstawie przykładowego modelu aplikacji Impulse C. Wyniki tych badań pozwalają stwierdzić, ze sprzęg jest łatwy do zrealizowania, a uzyskane rezultaty udowadniają prawidłowe jego działanie. Aplikacja sprzęgająca została przebadana również pod katem spadku szybkości wykonywania modelu Impulse C. Spadek ten jest naturalna konsekwencja wprowadzenia sprzęgu, a jego wielkość jest maksymalna w przypadku systemu wbudowanego komunikującego sie z komputerem PC. Największy wpływ na szybkość komunikacji po tej stronie ma zastosowany zegar systemowy, który jest wyznacznikiem czasu przełączania wątków, na których oparty jest model Impulse C i warstwa pośrednia realizująca komunikację w sprzęgu. W przypadku systemu wbudowanego opartego na jądrze Xilkernel uzyskane wyniki wskazują na problem z zastosowanym stosem sieciowym lwIP. Najlepszy wynik wydajności (ok. 3000 kroków na sekundę) uzyskano dla jednego procesu wykonywanego na komputerze PC. Najlepszy wynik spośród wariantów angażujących system wbudowany, to 160 kroków/sekundę. Uzyskano go dla jednego procesu z jednym strumieniem. Zwiększanie ilości procesów spowodowało sukcesywny spadek wydajności. Na uwagę zasługuje wydajność modelu bez sprzęgu uzyskana w systemie wbudowanym opartym o system uClinux, w porównaniu z komputerem PC. Wydajności te są zbliżone (dla procesora Microblaze o częstotliwości taktowania 75 MHz), co daje obraz potencjału języka Impulse C.

Aplikacja sprzęgająca została zrealizowana na platformie Microsoft .NET Framework 2.0 i Microsoft Visual Studio 2005, przy użyciu języka C# (interfejs graficzny, parser plików SMD z opisem modelu Impulse C oraz generowanie modelu Simulink i kodów źródłowych dla architektur docelowych) oraz języka C++ (serwer). Wybór platformy i środowiska programowania został podyktowany bogatym zestawem bibliotek udostępnianych przez .NET Framework 2.0.

5. Wnioski

Charakter przetwarzanych danych i mobilność systemów wymusiły łączenie procesorów sygnałowych z mikrokontrolerami. Kolejnym krokiem było połączenie zalet tych dwóch typów procesorów w jedna architekturę, zwaną procesorem konwergentnym. Obecny trend polega na wykorzystaniu układów reprogramowalnych w roli platform implementacji systemów wbudowanych. Wraz z tym trendem pojawiły się narzędzia konwersji opisu języków sekwencyjnych na języki opisu sprzętu, które mają za zadanie najlepiej wykorzystać potencjał tkwiący w układach reprogramowalnych. Dalszym warunkiem rozwoju systemów wbudowanych jest rozbudowa i tworzenie nowych narzędzi, które pozwolą na skracanie cykli projektowych, zwiększając przy tym niezawodność urządzeń oraz pozwalając na wykorzystywanie coraz bardziej skomplikowanych architektur i struktur obliczeniowych. Wyraźne dążenie do jak największej automatyzacji tworzenia systemów wbudowanych wymusza tworzenie narzędzi, w których projektant skupia sie na funkcjonalności, a nie na szczegółach implementacyjnych.

Celem przedstawionej pracy było zbadanie możliwości wykonania, zaimplementowanie oraz przetestowanie aplikacji sprzęgającej, która umożliwia jednoczesne użycie pakietu MATLAB/ Simulink oraz Impulse C. Cel został osiągnięty, a aplikacja sprzęgająca pozwala w wygodny sposób tworzyć modele Simulink będące odzwierciedleniem modeli Impulse C, testować je i wpływać na ich działanie w procesie wytwarzania systemu wbudowanego. Zastosowany model klient-serwer pozwala na umieszczanie procesów Impulse C w dowolnej lokalizacji, dzięki temu możliwa jest bezpośrednia komunikacja z systemem wbudowanym z poziomu modelu Simulink. Możliwy jest również podgląd sygnałów zarówno w trakcie, jak i po zakończeniu wykonywania modelu. Sygnały mogą być usuwane, mogą być zmieniane ich procesy źródłowe i docelowe. Możliwa jest też zmiana kolejności procesów w kontekście przepływających danych oraz ich selektywne usuwanie z modelu.

6. Literatura

[1] Ito S.A., Carro L.: A comparison of microcontrollers targeted to FPGA-based embedded applications, Integrated Circuits and Systems Design Proceedings, 13th Symposium on, pp. 397 – 402, 2000.

[2] Sharbari B., Amitabha S.: Performance Analysis of Different DSP Algorithms on Advanced Microcontroller and FPGA, Advances in Computational Tools for Engineering Applications, ACTEA '09, International Conference on, pp. 609 – 613, 2009.

[3] Jicheng Ch., Qingdong Y., Peng L., Ce C., MD16: DSP with some RISC features for embedded system, Signal Processing Proceedings, ICSP '04, 7th International Conference on, vol.1, pp. 144 – 147, 2004.

[4] Katz D.: Convergent processors solve development challenges, http://www.embedded.com/columns/technicalinsights/188100351, 2006.

[5] Bailey B., Martin G., Piziali A.: ESL Design and Verification, Morgan Kaufmann Publishers, 2007.

[6] Menotti R., Cardoso J.M.P., Fernandes M.M., Marques E., LALP: A Novel Language to Program Custom FPGA-Based Architectures, Computer Architecture and High Performance Computing, SBAC-PAD '09, 21st International Symposium on, pp. 3 – 10, 2009.

[7] Nanda R., Chia-Hsiang Yang, Markovic D.: DSP architecture optimization in Matlab/Simulink environment, VLSI Circuits, IEEE Symposium on, pp. 192 – 193, 18-20 June 2008.

[8] Barlas T., Moallem M.: Next generation of embedded controllers: Developing FPGA-based reconfigurable controllers using Matlab/Simulink, Emerging Technologies and Factory Automation, ETFA 2008, IEEE International Conference on, pp. 1055 – 1058, 2008.

[9] Antola A., Santambrogio M.D., Fracassi M., Gotti P., Sandionigi C.: A Novel Hardware/Software Codesign Methodology Based on Dynamic Reconfiguration with Impulse C and Codeveloper, Programmable Logic, SPL '07. 2007 3rd Southern Conference on, pp. 221 – 224, 2007.

otrzymano / received: 09.05.2010
przyjęto do druku / accepted: 04.06.2010 *artykuł recenzowany*