Janusz JABŁOŃSKI
UNIWERSYTET ZIELONOGÓRSKI, WYDZIAŁ MATEMATYKI INFORMATYKI I EKONOMETRII

Zdalnie rekonfigurowalny system rozproszony z FPSLIC

Dr inż. Janusz JABŁOŃSKI

W latach 1994 – 2003 asystent, następnie adiunkt Instytutu Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Od września 2004 adiunkt Wydziału Matematyki, Informatyki i Ekonometrii UZ. Obecnie prowadzi zajęcia z zakresu Inżynierii Oprogramowania. Posiada, również doświadczenie dydaktyczne w zakresie architektury komputerów oraz logiki programowalnej. Podstawowe badania związane z wykorzystaniem arytmetyki resztowej oraz układów FPGA w akceleracji obliczeń.

e-mail: J.Jablonski@wmie.uz.zgora.pl

Streszczenie

Ostatnia dekada potwierdziła możliwości wykorzystania logiki programowalnej zarówno w realizacji systemów prototypowych jak również w aplikacjach wymagających znacznych mocy obliczeniowych. Rozwój technologii oraz metod projektowania systemów opartych na FPGA zmierza w kierunku budowy systemów konfigurowalnych dynamicznie – w czasie działania aplikacji. W takich aplikacjach funkcjonalność i całkowita efektywność systemu uzależniona jest od efektywności algorytmów oraz czasu rekonfiguracji jak również przepustowości i zasięgu interfejsu rekonfigurującego zasoby FPGA. W prezentowanym artykule przedstawiona została metoda pozwalająca na znaczącą redukcję czasu rekonfiguracji przez zmniejszenie rozmiaru strumienia danych konfiguracyjnych. Zaproponowano również interfejs pozwalający na efektywne wykorzystanie układów typu FPSLIC w budowie rozproszonego rekonfigurowanego systemu sterowania.

Słowa kluczowe: FPGA, systemy rozproszone, zdalna rekonfiguracja

Remote reconfigurable distributed system with FPSLIC

Abstract

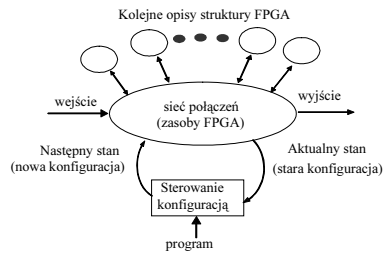
Over the last decade programmable logic proved its power in hardware computing acceleration and systems prototyping in many applications. The goal of evolution reconfigurable architectures is to realize applications or systems which need to be reconfigured frequently during system run-time. In this systems total performances can be degraded by the size of exchanged resources and reconfiguration time. This paper presents applications which generates specific differential bit stream for FPGA integrated in FPSLIC. The main advantage of proposed application is significantly smaller bit stream size for new functions. It is also proposed interface for dynamic FPGA reconfiguration which may be used for building distributed reconfigurable logic controller or reconfigurable elements for grid computing system.

Keywords: FPGA, reconfigurable controllers, distributed system, remote reconfigurations.

1. Wstęp

Najczęściej pod nazwą kontroler programowalny, PLC (ang. *Programmable Logic Controller*) kryje się realizacja – przez mikrokontroler, umieszczonego w pamięci programu – odpowiadającego funkcji sterowania. W rzeczywistości jest to układ mikroprocesorowy – programowany pamięciowo, pozwalający na zmiany w algorytmach sterowania lub modyfikacje funkcjonalności przez zmiany w oprogramowaniu. W takim rozwiązaniu - przy nie zmienionej mocy obliczeniowej, dodanie funkcjonalności powoduje spadek efektywności – dłuższy czas reakcji. Może to doprowadzić do niespełnienia, przez sterownik, rygorów czasowych. Konsekwencją niedotrzymania wymogów – czasu realizacji zadań może, najczęściej jest konieczność wymiany mikrokontrolera na szybszy, wydajniejszy i zazwyczaj droższy.

Układy programowalne znajdują zastosowanie przede wszystkim w realizacji systemów prototypowych, jednakże ze względu na specyficzne właściwości coraz częściej wykorzystywane są również w urządzeniach oferowanych komercyjnie. Szczególnie w cyfrowych systemach przetwarzania informacji – wymagających dużych mocy obliczeniowych, przydatne okazują się technologie rekonfigurowalne [3, 5]. Obiecujące możliwości zastosowań wiązane są z dynamiczną rekonfiguracją zasobów FPGA – głównie w układach typu SoC (ang. *System on Chip*). Jedną z zalet dynamicznej rekonfiguracji (ang. *run-time reconfigurations*) jest częściowe uniezależnienie funkcjonalności systemu od rzeczywistych zasobów sprzętowych. Osiągane jest to przez kolejkowanie zadań i implementację funkcjonalności przewidzianych do realizacji w następnym kroku przetwarzania, w zasobach zwolnionych przez funkcje już zrealizowane lub przez przełączanie kontekstów [4]. Takie podejście, przedstawione na rys.1, przypomina mechanizm programowania z wykorzystaniem pamięci *cache.* W literaturze odnoszącej się do technologii FPGA taka technika realizacji systemu rekonfigurowanego znana jest jako koncepcja CCM (ang. *Custom Computing Machine*) [2, 6].



Rys. 1. Model koncepcji CCM
Fig. 1. Model for CCM concept

Jako zalety takiego podejścia w realizacji systemu cyfrowego można wymienić: dłuższą żywotność, znaczne oszczędności zużycia energii wynikające z mniejszych wymaganych rzeczywistych zasobów oraz szerokie możliwości adaptacji do zmieniających się wymagań i realizowanych algorytmów.

2. Zdalna rekonfiguracja FPGA - ograniczenia

Programy badawcze takie jak Cure czy Seti wykorzystują przetwarzanie rozproszone w celu doraźnego stworzenia superkomputera. Rozwój technologii gridowych oraz sieci peer-to-peer wspierany jest przez takich potentatów branży technologii internetowych jak IBM czy Intel. Równie obiecujące możliwości tkwią w wykorzystaniu wydajnych i energooszczędnych – opartych na technologii FPGA oraz koncepcji CCM, sterowników oraz węzłów przetwarzania informacji. Możliwe zastosowania to systemy pomiarowo kontrolne czasu rzeczywistego na przykład w elektroenergetyce. W systemach informatycznych może to być pełnienie funkcji rozproszonych rekonfigurowanych węzłów superkomputera. Moc obliczeniowa, którą mógłby udostępnić superkomputer oparty na węzłach złożonych z połączonych rekonfigurowanych jednostek obliczeniowych, takich jak telefony komórkowe byłaby ogromna.

Jednakże, realizacja koncepcji CCM w implementacji systemów zdalnie rekonfigurowanych napotyka na szereg problemów. Podstawowe z nich dotyczą ograniczeń związanych z interfejsem rekonfiguracji:

- zasięg interfejsów rekonfiguracji zaledwie kilka metrów,
- rekonfiguracja najwyżej kilku połączonych układów,
- niewielka (100kB) przepustowość interfejsu.

Wymusza to centralizację re-konfigurowalnych węzłów z FPGA oraz ustalenie w początkowej fazie projektu ich liczby. Dodatkowe ograniczenia dotyczą efektywności systemu związanej

z rekonfiguracją. Czas rekonfiguracji dodaje się do czasu przetwarzania, co przy częstych zmianach konfiguracji znacznie zmniejsza szybkość realizacji zadań. Oddzielną grupę ograniczeń, w realizacji koncepcji CCM, dotyczy braku narzędzi wspomagających cykl projektowania systemów rekonfigurowanych opartych na FPGA.

Barierę czasu rekonfiguracji można łagodzić przez wykorzystanie w projekcie rodziny układów FPGA zaliczanych do grupy wielo-kontekstowych [3]. Dla których typowe metody generowania programującego strumienia różnicowego (Diff-BST, ang. *Differential Bit-Stream*), odpowiadającego dwóm różnym funkcjonalnością, korzystają z wyników etapu syntezy a typowy format plików będących danymi wejściowymi etapu implementacji to EDIF lub XNF [8].

Dla kontrolerów FSM (ang. *Finie State Machine*) korzystających z wbudowanej w FPGA pamięci RAM implementowanych w układach firmy Altera czy też Xilinx (rodzina Virtex) efektem może być zmniejszenie rozmiaru strumienia konfiguracyjnego nawet o kilkadziesiąt procent. Jednakże, dla projektów korzystających z logiki programowalnej implementowanej w tablicach LUT (ang. *Look Up Table*), między innymi ze względu na konieczność rekonfiguracji pełnymi *slice*-ami oraz duży nakład na połączenie zasobów, efekt nie jest już tak spektakularny.

Powyżej opisane podejście pozwala na zmniejszenie strumienia danych konfiguracyjnych, jednak wymaga to dostępu do źródeł specyfikacji projektu w języku opisu sprzętu co dla projektów złożonych z bloków typu IP-Core (ang. *Intelectual Property*-Core) może być kosztowne lub wręcz niemożliwe. Ponadto układy wielo-kontekstowe, oferujące niezaprzeczalnie ogromne możliwości, są stosunkowo drogie.

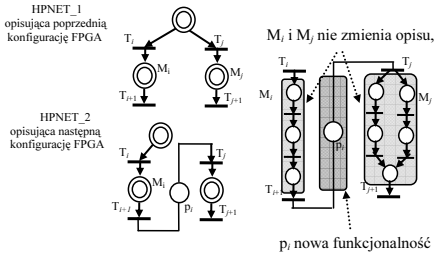
3. Proponowane rozwiązania dla FPSLIC

Uwzględniając możliwości, wady i zalety oraz koszty układów rekonfigurowanych dalsze rozwiązania ukierunkowane są przede wszystkim na systemy wykorzystujące układ AT94K firmy Atmel. Nazwa rynkowa tego, zawierającego w jednej obudowie: mikrokontroler AVR (ATmega 103), rekonfigurowaną matrycę AT60k, 32kB pamięci programu i danych jak również niezbędne do budowy jednoukładowego system cyfrowego peryferia, to FPSLIC (ang. Field Programmable System Integrate Circuits) [8].

3.1. Generowanie strumienia różnicowego

Sieci Petriego stanowią matematyczną reprezentację dyskretnych systemów rozproszonych. Szczególnie chętnie w modelowaniu, specyfikacji oraz analizie kontrolerów współbieżnych wykorzystywane są sieci hierarchiczne [7].

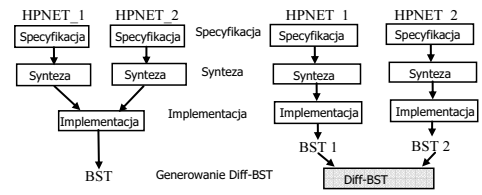
Podstawę mechanizmu minimalizacji rozmiaru strumienia danych dla kolejnych konfiguracji FPGA w FPSLIC przedstawia przykładowy model (rys. 2) dla sieci hierarchicznej HPNET (ang. *Hierarchical Petri Net*).



Rys. 2. Model Hierarchicznej Sieci Petriego
Fig. 2. Hierarchical Petri Net model

Ponieważ kolejne konfiguracje zmieniają tylko część opisu zasobów FPGA zatem pozostała funkcjonalność nie ulega rekonfiguracji. Konfiguracje FPGA dla HPNET_1 i HPNET_2 (rys. 2) niewiele różnią się między sobą. Zatem programując układ FPGA strumieniem bitów wygenerowanym dla HPNET-2 zostanie wykonana „wirtualna" rekonfiguracja większości zasobów. W rzeczywistości opis systemu został rozszerzony o implementację jednego miejsca p_i. Zatem, rzeczywista rekonfiguracja dotyczy tylko niewielkiego fragmentu FPGA związanego z nową funkcjonalnością (rys. 2).

Proponowane rozwiązanie polega na generowaniu danych odpowiadających rzeczywistym różnicom w konfiguracji pomiędzy dwoma strumieniami konfiguracyjnymi BST (ang. *Bit-strem*) opisującymi poprzednią oraz następną konfigurację zasobów FPGA. Model typowej i proponowanej ścieżki projektowej generowania opisu zasobów re-konfigurowalnego dynamicznie systemu został przedstawiony na rys. 3.

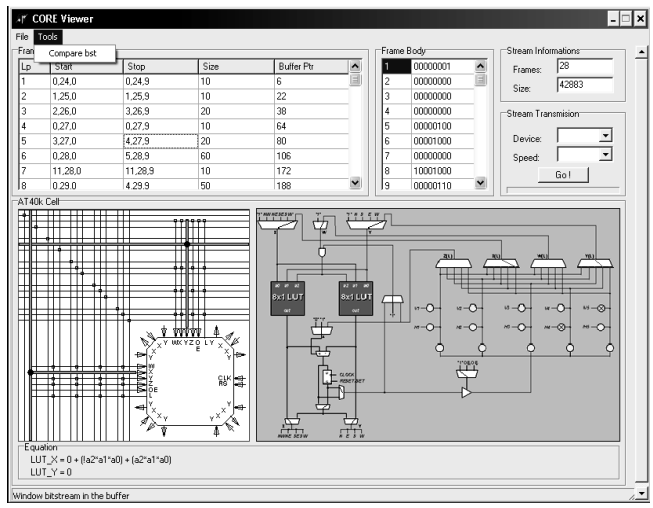


Rys. 3. Schemat projektu z FPGA: konwencjonalnej i proponowanej
Fig. 3. Pattern for FPGA project

Powyższy schemat wskazuje, iż proponowana ścieżka projektowa pozwala na generowanie Diff-BST bez dostępu do specyfikacji systemu w języku opisu sprzętu, jak również bez konieczności posiadania wyników etapu syntezy. Zaletą proponowanego podejścia w realizacji ścieżki projektowej dla FPSLIC jest więc możliwość osiągnięcia mniejszych rozmiarów pliku z opisem zasobów, bez względu na ukierunkowanie systemu, zarówno dla systemów sterowania ukierunkowanego na korzystanie z pamięci RAM, jak również dla układów operacyjnych. Jednakże oferowane przez producentów narzędzia projektowania nie udostępniają możliwości generowania Diff-BST dla proponowanej (rys. 3) ścieżki projektowej.

3.2. Aplikacja generująca Diff-BST

W celu sprawdzenia skuteczności rekonfiguracji oraz możliwości w zakresie generowania minimalnego rozmiaru Diff-BST jako strumienia danych rekonfiguracyjnych FPGA w FPSLIC zaprojektowana oraz przygotowana została dedykowana aplikacja, której funkcjonalność przedstawia rys. 4.



Rys. 4. Aplikacja generująca Diff-BST
Fig. 4. Aplication for Diff-BST generate

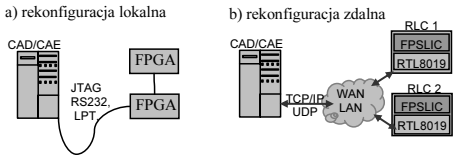
Efektem porównania dwóch strumieni BST (Compare bst, rys. 4) jest struktura danych zawierająca informacje umożliwiające rekonfigurację FPGA w FPSLIC, którą można wykorzystać do

zaprogramowania układu bezpośrednio z aplikacji poprzez interfejs RS232 lub wygenerowany plik przechować w celu późniejszego wykorzystania. Przygotowana aplikacja pozwala na wizualizację konfiguracji poszczególnych elementów wchodzących w skład pojedynczego CLB (ang. *Configurable Logic Block*), jak również wizualizacji konfiguracji połączeń globalnych i lokalnych zasobów FPGA. Unikalną w stosunku do innych dedykowanych do projektowania systemów cyfrowych opartych na matrycy rekonfigurowanej w FPSLIC jest wizualizacja równań logicznych opisujących zawartość tablic LUT bloków przeznaczonych do realizacji funkcjonalności układu.

3.3. Zdalna rekonfiguracja zasobów

Pod nazwą system rozproszony kryją się specyficzne wymagania określone przez takie cechy jak współdzielenie zasobów, otwartość, współbieżność, skalowalność a także odporność na błędy i transparentność.

Konwencjonalne metody oraz interfejsy programowania (rys. 5) wraz z ich ograniczeniami, z których istotniejsze z punktu widzenia realizacji założeń wymieniono w pkt. 2, nie dają możliwości budowy zdalnie rekonfigurowanego systemu rozproszonego spełniającego powyższe wymagania.

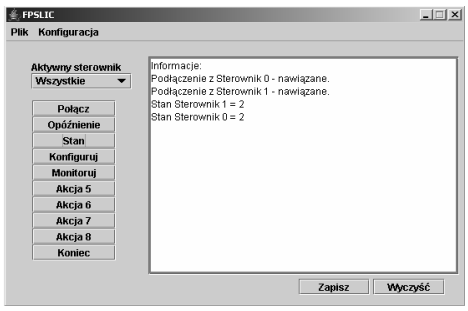


Rys. 5. Schemat rekonfiguracji FPGA a) konwencjonalny, b) zaproponowany
Fig. 5. Schema for FPGA reconfiguraion a) typical, b) proposed

Wychodząc naprzeciw wymaganiom związanym z rozproszeniem oraz zdalnej rekonfiguracji zaproponowane zostało rozwiązanie mechanizmu wykorzystujące interfejs oparty na protokole TCP/IP lub UDP - w zależności od potrzeb można wybrać odpowiednią usługę. Proponując interfejs TCP/IP kierowano się również względami bezpieczeństwa, gdzie pod pojęciem „bezpiecznego" rozumiana jest pewność iż system z FPSLIC przyjął Diff-BST oraz wykonał właściwie operację zdalnej rekonfiguracji.

Prototyp systemu wykorzystuje do obsługi pierwszych warstw modelu OSI oraz protokołu komunikacyjnego układ RTL8019AS Przygotowany, na platformę FPSLIC, driver przystosowany jest do obsługi protokołu TCP/IP oraz UDP (wsparcie dla mechanizmów klient-server oraz peer-to-reer). Obsługą danych oraz pakietów pochodzących z „sieci" jak również konfiguracją zajmuje się program realizowany przez wbudowany w FPSLIC mikrokontroler AVR. Oprócz typowych usług takich jak „echo" aplikacja obsługuje przyporządkowaną do jednego z wybranych portów TCP/IP usługę rekonfiguracji zasobów FPGA.

Interfejs użytkownika ostatniego z przygotowanych narzędzi wspomagających realizację rekonfigurowanego systemu przedstawia rys. 6.



Rys. 6. Okno aplikacji nadzorującej pracę i rekonfigurację sterowników
Fig. 6. Application window to supervising and reconfiguration controllers

W prototypie, systemu o architekturze klient-server, do weryfikacji komunikacji oraz kontroli poprawności rekonfiguracji jak również nadzorowania pracy systemu przygotowano aplikację napisaną w języku JAVA. Aplikacja uruchomiona na komputerze, CAD/CAE w modelu z rys. 5, oprócz zarządzania rozsyłaniem Diff-BST wybranym sterownikom z FPGA pozwala na szereg dodatkowych a pomocnych funkcji. Jako dodatkowa funkcjonalność możliwe jest sprawdzenie opóźnienia pakietów na drodze do sterownika, odczyt stanu automatu FSM realizowanego w FPGA, monitoring stanu urządzenia oraz przesyłanie danych jak również innych nie zdefiniowanych w wersji prototypowej funkcjonalności. Przykładem może być funkcja sprawdzająca opóźnienie w dostarczaniu pakietów, które pozwala na właściwe dobranie czasów związanych z synchronizacją odczytu stanów automatu lub synchronizację przepływu danych konfiguracyjnych.

4. Podsumowanie

Zaproponowane rozwiązania rozszerzają zakres stosowalności koncepcji CCM na możliwość zdalnej rekonfiguracji FPGA i przetwarzanie rozproszone.

Wstępne, badania wykorzystujące przygotowane prototypowe rozwiązania i aplikacje, potwierdzają możliwość znacznej redukcji strumienia danych dla zaproponowanej metody generowania Diff-BST. Dalekosiężny interfejs TCP/IP wykorzystany do rekonfiguracji pozwala na rozproszenie terytorialne rekonfigurowanych węzłów przetwarzania. Duża szybkość lub raczej przepustowość interfejsu w połączeniu ze znacząco mniejszym strumieniem danych opisujących kolejne konfiguracje pozwala na wielokrotną rekonfigurację zasobów FPGA w ciągu jednej sekundy. Wykorzystując narzędzia dedykowane przez producentów oprogramowania do konwencjonalnej ścieżki projektowej praktycznie nie jest możliwe osiągnięcie takiego rezultatu.

Programowa realizacja wszystkich zadań związanych z obsługą interfejsu i rekonfiguracji pozostawia całość zasobów FPGA dla realizacji systemu rekonfigurowanego. W wersji prototypowego sterownika oraz aplikacji realizowanej przez mikrokontroler w FPSLIC wykorzystane zostaje ok. 40% zasobów pamięci programu i danych, pozostałe 60% procent zasobów pozwala na realizację dodatkowych funkcji. Przewidywane jest między innymi rozszerzenie funkcjonalności sterownika o obsługę mechanizmów (DHCP) związanych ze skalowalnością systemu.

Praca naukowa finansowana ze środków Komitetu Badań Naukowych w latach 2004-2006 jako projekt badawczy (grant nr 3 T11C 046 26)

5. Literatura

[1] Bondalapati, K, V.K. Prasanna, (2000), Loop Pipelining and Optimization for Run Time Reconfiguration, Technical report: Computation Models for Reconfigurable Machines, Department of Electrical Enginieric Systems, University of Southern California, USA
[2] Buel D. A. and Pocek K. L., Custom Computing Machines: An introduction. Journal of Supercomputing, 9(3), ss. 219 – 230, 1995
[3] Jasinski K and Zbysinski P., Reconfigurable system coprocessor: Universal Calculation Platform, RUC'2000, Szczecin, Poland, pp.343-349
[4] Maestre R., F. J. Kurdahi, M. Fernández, R. Hermida, N Bagherzadeh, and Hartej Singh, A, Framework for Reconfigurable Computing: Task Sched-uling and Context Management, IEEE Trasn. on VLSI Sys. Vol 9, No. 6, 2001
[5] Meyer-Baese U. Digital Signal Processing with Field Programmable Gate Arrays, Second Edition, Springer Verlag, Berlin 2004
[6] Sima M. at all, Field-Programmable Custom Computing Machines – A taxonomy, International Conference on Field Programmable Logic and Aplication, ss 79 – 88, Montpelier, France, Sept. 2002
[7] Węgrzyn A., Węgrzyn M., Petri net-based specification, analysis and synthesis of logic controllers-Proc. Of Symp.: IEEE International Syposium on Industrial Electronics ISIE'2000, Dec. Mexico, ss. 20 - 26
[8] http://WWW..Xilinx.com
[9] http://WWW.Atmel.com

Artykuł recenzowany