II Konferencja Naukowa KNWS'05
"Informatyka- sztuka czy rzemiosło"
15-18 czerwca 2005, Złotniki Lubańskie

REALIZACJA SYSTEMU OPERACYJNEGO CZASU RZECZYWISTEGO W FPSLIC

Janusz Jabłoński [1], Marek Węgrzyn [2]

[1] Zakład Matematyki Dyskretnej, Algebry i Informatyki; [2] Instytut Informatyki i Elektroniki
Uniwersytet Zielonogórski, 65-246 Zielona Góra, ul. Podgórna 50

e-mail: J.Jabłoński@wmie.uz.zgora.pl, M.Wegrzyn@iie.uz.zgora.pl

STRESZCZENIE

Nowoczesne układy FPGA zawierają, oprócz programowanej struktury również wbudowany rdzeń układu mikroprocesorowego, na przykład typu AVR. Daje to możliwość pełnej realizacji systemu w jednym tylko układzie scalonym (ang. *System-on-chip, SoC*). Przygotowując program, który będzie współpracował z odpowiednio skonfigurowaną częścią programowalnych zasobów FPGA otrzymuje się system, który do tej pory dostępny był przy zastosowaniu złożonych technologii wytwarzania układów ASIC. Aby jednak przygotować odpowiedni projekt układu sterowania o podwyższonym stopniu bezpieczeństwa muszą być spełnione określone rygory czasowe. Dlatego słusznym podejściem jest implementacja dopasowanych systemów operacyjnych spełniających wymagania czasu rzeczywistego (ang. *real-time operating system, RTOS*). Jednakże nawet w przypadku korzystania z RTOS i zarządzania zasobami przez system operacyjny (w oparciu o wywłaszczanie) jest trudne w implementacji. Podejściem umożliwiającym pełniejszą kontrolę nad czasem realizacji procesów, w tym również sterowania, jest implementacja krytycznych funkcji systemu w sprzęcie. W referacie przedstawiono możliwości przeniesienia określonych funkcji systemu do sprzętu (FPGA) w układach serii FPSLIC firmy ATMEL.

1. WPROWADZENIE

Rozwój systemów informatycznych zmierza w kierunku technologii komponentowych oraz przenośnych urządzeń posiadających zdolność przyłączenia się do Internetu [2]. Coraz częściej oferowane są urządzenia nie będące komputerem w tradycyjnym znaczeniu acz posiadające niezbędne dla podłączenia do Internetu zasoby sprzętowo programowe [6]. Urządzenia mobilne posiadają zdumiewająco praktyczne możliwości w zakresie wymiany informacji oraz wspomagania procesów pozyskiwania wiedzy. Najlepszym przykładem tego typu urządzeń, określanych jako mobilne, są telefony komórkowe czy też palmtopy. Urządzenia sterowane za pośrednictwem Internetu i szerokopasmowej transmisji bezprzewodowej nie są już nowością natomiast nowe technologie oraz prototypowe rozwiązania potrafią zaspokoić najbardziej ekstrawaganckie wymagania. Trudno oprzeć się

wrażeniu, że rozwój technologii internetowych jest samonapędzającym się mechanizmem, u podstaw którego stoi potrzeba komunikacji i powszechna dostępność medium. Kolejny krok rozwoju to konstruowanie niekonwencjonalnych urządzeń wykorzystujących Internet dla celów nadzorowania i synchronizacji procesów technologicznych, zdalne sterowanie ruchem ulicznym, zdalnie sterowany transport publiczny, medycyna a w niedalekiej przyszłości mobilne roboty militarne. Wskazuje to, iż systemy mobilne przyszłości powinny zapewniać wysoki stopień autonomii, przewidywalne czasy opóźnień, wielokrotne użycie, możliwość adaptacji pozwalający na realizację zmiennych wymagań oraz wysoki poziom bezpieczeństwa danych i realizowanych procesów.

2. SYSTEMY OPERACYJNE

Opanowanie złożoności oraz właściwe zarządzanie zasobami systemu cyfrowego wspomagane jest przez systemy operacyjne. Pośród wielu zadań SO ideą przewodnią przy ich projektowaniu jest udostępnienie użytkownikowi abstrakcji systemu mikroprocesorowego (programowalnego) łatwiejszej do programowania i uruchamiania programów użytkownika, nazywanej maszyną wirtualną. W zależności od wymagań konstruowane są maszyny wirtualne o zróżnicowanych cechach wynikających z określonych trybów przetwarzania [1]. Przykładowo tryb czasu rzeczywistego wykorzystywany jest w obsłudze procesów technologicznych, w których kolejność i czas operacji jest ściśle określony (rygory czasowe). Teoretycznie wymagania nakładane na czas reakcji w systemach czasu rzeczywistego RTOS (ang. *Real Time Operating System*) mogą być spełniane przez okresowe sprawdzanie stanu realizowanych zadań lub zagwarantowaniu, że reakcja systemu na określone zdarzenie nie przekroczy z góry ustalonego limitu czasu. W celu zagwarantowania interakcji SO z użytkownikiem udostępniany jest tryb użytkownika (konteksty użytkowników) oraz tryb jądra systemu zarządzającego: uprawnieniami użytkowników, priorytetami przerwań oraz konfiguracją i wykorzystaniem innych zasobów systemu. Takie podejście pozwala na implementację „spoolingu" oraz przetwarzania wielu programów należących do różnych grup użytkowników. Gotowe do wykonania zadanie staje się procesem w momencie obsługi przez procesor, przy czym realizowany mechanizm może polegać na wywłaszczeniu wykonującego się procesu lub bez wywłaszczania, czyli pozwolić procesowi na dokończenie wszystkich jego operacji przed przystąpieniem do obsługi nowego zdarzenia. Strategia obsługi procesów bez wywłaszczania jest prosta w implementacji i naturalna w realizacji jedno programowych lub wsadowych SO. Program użytkownika kończy się wywołaniem przerwania konfigurującego kontekst jądra systemu operacyjnego. Zatem konstruowanie interaktywnych systemów czasu rzeczywistego w oparciu o strategię bez wywłaszczania jest trudne w realizacji i nieefektywne: trudne, ponieważ nie można przewidzieć (z wyprzedzeniem) wszystkich konstrukcji programu użytkownika, więc również głębokości zagnieżdżenia wywoływanych procesów obsługujących przerwania, a nieefektywne ponieważ nie można

przewidzieć szybkości reakcji użytkownika oraz czasu reakcji dołączonych do standardowego wejścia-wyjścia (np. RS232, I^2C). Zatem projektowanie interaktywnych systemów mikroprocesorowych przeznaczonych do budowy urządzeń gwarantujących bezpieczeństwo zarówno użytkownikowi jak również realizowanym procesom sterowania powinno opierać się na wykorzystaniu systemów czasu rzeczywistego ze wspomaganiem funkcji krytycznych ze względu na czas wykonania przez zasoby sprzętowe. W klasycznym ujęciu prowadzi to do projektowania systemów sprzętowo programowych. Ograniczeniem klasycznego podejścia w projektowania systemów sprzętowo programowych jest dedykowany konkretnemu rozwiązaniu i sztywny podział pomiędzy procesy realizowane programowo i sprzętowo, przyjęty w fazie projektowania.

3. KONCEPCJA REKONFIGURACJI DLA RTOS

Wymagania użytkowników dotyczące szybkości reakcji, zabezpieczeń, odporności na uszkodzenia oraz innych parametrów użytkowych dla systemów mobilnych wystarczające „wczoraj" dziś już nie spełniają wymagań użytkowników. Poprawa funkcjonalności oraz efektywności przyspiesza „efekt starzenia się urządzeń" a próby ograniczenia tego efektu mogą polegać na wykorzystaniu w projektowaniu urządzeń mobilnych układów typu SoC (ang. *System on Chip*) integrujących w jednej obudowie system mikroprocesorowy i elementy logiki programowanej [6]. W niektórych dziedzinach takich jak np.: badanie kosmosu dostosowanie do nowych zadań przez okresową wymianę lub rozbudowę urządzeń jest zbyt kosztowne lub niemożliwe, natomiast zastosowanie układów programowalnych umożliwia łączenie korzyści wynikających z akceleracji sprzętowej oraz okresowej zmiany pełnionej funkcji [5]. Zastosowanie układów FPGA w realizacji systemu cyfrowego umożliwia projektowanie z wykorzystaniem wirtualnych zasobów przy nieznacznym wzroście kosztów projektu [8]. Powyżej opisany mechanizm wykorzystania zasobów FPGA polega na implementacji modułów w zależności od zapotrzebowania na realizowaną funkcję (lub długości życia algorytmu) kojarzony jest przetwarzaniem opartym na koncepcji CCM (ang. *Custom Computing Machine*). Nowe możliwości jej wykorzystania oparte są na programowaniu połączeń lub wykorzystaniu magistral programowanych również z wykorzystaniem światła [3,4]. Podobne zasady korzystania z zasobów realizują wieloprogramowe systemy operacyjne, w których mechanizm przetwarzania opiera się na podziale czasu i implementacji strategii wymiatania procesów z pamięci operacyjnej. Jednakże w CCM zakłada się że procedury przetwarzania realizowane są sprzętowo a nie programowo. Takie podejście, pozwala na realizację w jednym elemencie scalonym projektów sprzętowych lub sprzętowo programowych o dużej złożoności lub zmienności realizowanych procesów. Pozwala to na realizację RTOS cechujących się dużą szybkością reakcji, w szczególności dedykowanych systemom nadzoru procesów technologicznych. W takim, sprzętowo programowym systemie podstawowe i krytyczne czasowo zadania

związane z obsługą urządzeń, przełączaniem zadań, zarządzania procesami można realizować sprzętowo i mogą one ulegać częstym modyfikacjom. Jednakże podstawową barierą w realizacji takiego systemu jest brak wsparcia tej koncepcji przez dostępne oprogramowanie CAD, jak również niewielkie możliwości układów programowalnych w zakresie rekonfiguracji dynamicznej oraz w czasie rzeczywistym [7]. Większość układów programowalnych nie umożliwia zachowania danych przechowywanych w FPGA podczas aktualizowania lub zmiany konfiguracji, co znacznie ogranicza możliwość pamiętania danych z poprzedniego etapu przetwarzania oraz realizacji koncepcji RTOS opartego na wirtualnych zasobach układu reprogramowalnego. W grupie układów FPGA wspierających mechanizm rekonfiguracji dynamicznej (ang. *Run Time Reconfiguration*) na szczególną uwagę zasługują SoC typu FPSLIC firmy Atmel. Jednakże niewielki rozmiar pamięci wewnętrznej (32kB) znacznie ogranicza możliwość przechowywania kilku konfiguracji, natomiast duży rozmiar (ponad 400kB) kompletnego generowanego strumienia konfiguracyjnego (ang. *Bit stream*) czyni nieefektywnymi metody rekonfiguracji oparte na typowym wykorzystaniu interfejsu szeregowego. W przypadku układów firm Xilinx lub Altery rozmiar strumienia jest kilkakrotnie większy.

Proponowana koncepcja RTOS opartego na wirtualnych zasobach opiera się na fakcie, iż z reguły modyfikacje w realizowanym systemie dotyczą niewielkiego zakresu funkcji. Zatem ze względu na liczbę aktualizowanych funkcji możliwe jest przyjęcie przyrostowego modelu realizacji systemu, w którym mniejszy zakres zmian funkcji implementowanych w układach wielo-kontekstowych typu FPSLIC skutkuje mniejszym rozmiarem rekonfigurowanych zasobów oraz krótszym czasem rekonfiguracji. Na podstawie przeprowadzonych doświadczeń zauważono również, że dla zmian realizowanych wewnątrz funkcji generowane są niewielkie różnice w danych konfiguracyjnych. Wynika to z faktu, iż modyfikowane są realizowane funkcje i połączenia. Przykładowo, zmiana funkcji jednej tablicy odniesienia LUT (ang. *Look Up Table*) generuje różnicę w strumieniu o rozmiarze jednego bajta, natomiast kompletna informacja dotycząca identyfikacji oraz realizowanej funkcji wymaga czterech bajtów. Tak więc dla układów typu FPSLIC, w których do rekonfiguracji można wykorzystać zintegrowany mikroprocesor, możliwe jest znaczne ograniczenie rozmiaru przesyłanych danych konfiguracyjnych i co z tym jest związane czasu rekonfiguracji. Warto zauważyć również, iż proces rekonfiguracji przebiega bez utraty informacji i danych z poprzedniego etapu przetwarzania. Jednakże dostępne narzędzia CAD nie umożliwiają generowania tego typu różnicowego strumienia konfigurującego. Zatem w celu weryfikacji powyższej tezy należało zaprojektować prototyp systemu mogącego realizować koncepcję CCM i rekonfiguracji. Strumienie różnicowe generuje specjalnie do tego celu przygotowana aplikacja uruchamiana na dedykowanym serwerze, która oprócz wizualizacji podstawowych elementów konfiguracji FPGA informuje również o funkcji logicznej realizowanej przez wybrane LUT. Wejściowe strumienie konfigurujące mogą zostać przygotowane w dowolnym przeznaczonym do tego celu systemie CAD (np. w *System*

Designer firmy Atmel). Aplikacja serwera uruchamiana na wbudowanym w FPSLIC mikrokontrolerze AVR obsługuje zasoby w szczególności obsługuje rekonfigurację zintegrowanego układu FPGA. Cały system powinien udostępniać trzy tryby pracy: tryb administratora, tryb programisty systemowego oraz tryb użytkownika. Wymaga to osadzenia na FPSLIC systemu sterującego udostępniającego, co najmniej cztery procesy: kontrola stanu systemu sterowanego, kontrola wystąpienia błędów, obsługa zdarzeń klienta i rekonfiguracja oraz przekazywanie informacji do systemu monitorującego w trybie konwersacyjnym. Przy czym, ze względu na zasoby oraz czas reakcji niektóre z zadań będą realizowane sprzętowo przez FPGA. Efektem powyższych wymagań wyróżniane są trzy tryby pracy systemu jako przypadki użycia systemu:

- tryb *administratora*, w którym realizowane są następujące zadania: ustalenie praw użytkowników, nadanie priorytetów realizowanym zadaniom, jak również sterowanie procesami rekonfiguracji FPGA;
- tryb *programisty systemowego* dotyczący podejmowania decyzji, które z procesów realizowanych przez urządzenia terminalne lub które z funkcji RTOS będą zaimplementowane w sprzęcie czyli przez zintegrowany FPGA; ponadto przypadek użycia przez programistę systemowego dotyczy przygotowania modelu oraz wygenerowania strumienia danych konfiguracyjnych dla procesów, które będą realizowane sprzętowo;
- tryb *użytkownika*, czyli bieżące nadzorowanie i raportowanie stanu realizowanych procesów sterowania oraz monitoring.

Aby umożliwić zdalną pracę urządzenia, jako interfejs wymiany danych pomiędzy bazą danych z konfiguracjami poszczególnych strumieni dla elementów opartych na FPSLIC wykorzystano protokół TCP/IP. Przygotowany w tym celu driver umożliwia obsługę przez FPSLIC układu RTL8019, jednakże ze względu na zasoby FPSLIC rozmiar pakietu ograniczony jest do 128 bajtów danych. Pomimo ograniczonego rozmiaru pakietu przeprowadzone doświadczenia, zdalnej rekonfiguracji zasobów, potwierdziły efektywność proponowanej metody.

Dalsze prace będą zmierzać w kierunku dynamicznej rekonfiguracji dla procesów jądra systemu operacyjnego w szczególności uzupełnienia wymagań czasowych oraz dostosowania funkcjonalności systemu do wymagań użytkownika lub procesu sterowania. W szczególności kontynuowane będą prace polegające na wyeliminowaniu czasochłonnych programowych procedur obsługujących komunikację między procesową i zastąpieniu ich przez realizację sprzętową. Ponadto na podstawie testów określony zostanie czasu potrzebny na realizację strategii wywłaszczania i synchronizacji procesów zapewniający możliwość rekonfiguracji FPGA bez utraty kontroli nad poprawnością realizowanych procesów sterowania a spełniających podwyższone rygory czasowe.

4. PODSUMOWANIE

W referacie przedstawiono propozycję nowej strategii projektowania systemów umożliwiającą efektywną dynamiczną rekonfigurację przystosowaną do realizacji w układach typu FPSLIC. Dla zaproponowanej metody przygotowane zostały odpowiednie aplikacje oraz system prototypowy, pozwalający na weryfikację postawionej tezy dotyczącej efektywności rekonfiguracji. Wstępne doświadczenia z wykorzystaniem proponowanych rozwiązań potwierdzają możliwość znaczącej redukcji (nawet 10-krotnie) rozmiaru strumienia konfiguracyjnego oraz zwiększenie efektywności systemów wykorzystujących koncepcję CCM. Ponadto opracowany system pozwala na rekonfigurację zdalną i łatwo może zostać zaadoptowany do sterowania w oparciu o technologię bezprzewodową. Oprócz przedstawionych zalet zauważono, iż zaproponowane rozwiązanie rekonfiguracji strumieniem różnicowym uniemożliwia rozpoznanie konfiguracji systemu poprzez „podsłuchanie". Ustalenie realizowanego przez FPGA oraz system algorytmu możliwe jest tylko wówczas, jeżeli znana byłaby konfiguracja początkowa oraz kolejno wszystkie modyfikacje. Ponieważ konfiguracje te przechowywane są w bazie danych na serwerze, zatem można uznać, że są bezpieczne.

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

LITERATURA

[1] W. Stallings: *Organizacja I architektura systemu komputerowego*, WNT, Warszawa 2001
[2] J. Cheesman, J. Daniels: *Komponenty w UML*, Wydawnictwo Naukowo Techniczne, Warszawa, 2004
[3] M. Vasilko, D. Ait-Boudaoud: *Optical Reconfigurable FPGS*, Proc. of 6th International Workshop on Field-Programmable Logic and Aplications, FPL'96, Germany, 1996, w: *Lectures Notes in Computer Science*, Vol. 1142, Springer-Verlag, 1996, ss.270-279
[4] D. Stroobandt: *Recent Advances in System-level Interconnect Prediction*, IEEE Circuits and Systems Society Newsletter, Vol. 11 No. 4, USA 200, ss.4-20
[5] K. Bondalapati, V. K. Prasma: *Dynamic Precision Management for Loop Computations on Reconfigurable Architectures*, Field Programmable Custom Computing Machines, FCCM'99
[6] T. Cain, et.al.: *SoC Designs Skills: Collaboration Builds a Stronger SoC Design Team*, Proc. of the 2001 IEEE International Conference on Microelectronic Systems Education, Las Vegas, 2001, ss.42-43
[7] A. Brinkmann, et.al.: *A Rapid Prototyping Enviroment for Microprocessor based System-on-Chip and its Aplication to the Development of a Network Processor*, 10th International Conference on Field Programmable Logic and Aplictions, FPL'2000, Austria, 2000, ss.838-841
[8] S. M. Scalera, J. R. Vazquez: *The Design and Implementations of a context-switching FPGA*, 6th IEEE Workshop on FPGAs for Custom Computing Machines, FCCM'98, USA, ss.78-85