Radosław CZARNECKI, Stanisław DENIZIAK
KATEDRA INFORMATYKI TECHNICZNEJ, POLITECHNIKA KRAKOWSKA

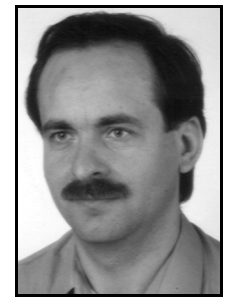
Kosynteza systemów dynamicznie rekonfigurowalnych reprezentowanych przez warunkowe grafy zadań

Mgr inż. Radosław CZARNECKI

Ukończył studia na Wydziale Inżynierii Elektrycznej i Komputerowej Politechniki Krakowskiej, obronił pracę magisterską w 2001 r. Jest asystentem w Katedrze Informatyki Technicznej Politechniki Krakowskiej. Jego zainteresowania naukowe to szybka prototypizacja systemów informatycznych, projektowanie wbudowanych systemów komputerowych w oparciu o układy FPGA.

e-mail: czarneck@pk.edu.pl

Dr hab. inż. Stanisław DENIZIAK



Ukończył studia na Wydziale Elektroniki Politechniki Warszawskiej, obronił pracę doktorską w 1994r, a habilitacyjną w 2006r. Jest profesorem nadzwyczajnym w Katedrze Informatyki Technicznej Politechniki Krakowskiej oraz profesorem wizytującym w Katedrze Informatyki Politechniki Świętokrzyskiej. Jego zainteresowania naukowe to metody szybkiego prototypowania systemów informatycznych, projektowanie systemów wbudowanych, testowanie i diagnostyka systemów cyfrowych.

e-mail: pedenizi@cyf-kr.edu.pl

Streszczenie

W pracy zaprezentowany jest rafinacyjny algorytm kosyntezy wieloprocesorowych, dynamicznie rekonfigurowalnych systemów wbudowanych. Jest to pierwszy algorytm wykorzystujący informacje o zadaniach wzajemnie się wykluczających (ZWW) do optymalizacji systemów dynamicznie rekonfigurowalnych. Specyfikacja takich zadań jest możliwa przy pomocy warunkowych grafów zadań. Wykorzystując dynamiczną rekonfigurację systemu możliwe jest przyporządkowanie zadań ZWW do tych samych zasobów sprzętowych. W ten sposób można zmniejszyć powierzchnię, a w wolnej przestrzeni alokować inne zadania sprzętowe, co również może prowadzić do zwiększenia szybkości systemu.

Słowa kluczowe: kosynteza, dynamiczna rekonfiguracja, FPGA, warunkowy graf zadań.

Co-synthesis of dynamically reconfigurable SOPC systems described by conditional task graphs

Abstract

In this work a co-synthesis method, which allows for optimization of dynamically reconfigurable multiprocessor SOPC system architecture, is presented. To our best knowledge, this is the first algorithm that takes into consideration mutually exclusive tasks in optimization of dynamically reconfigurable systems. Such tasks are presented using conditional task graphs. Partially reconfigurable FPGAs let reuse of the same hardware resources for mutually exclusive tasks. In this way the area occupied by embedded system can be decreased and free space can be used for other hardware tasks. It can also increase SOPC's performance.

Keywords: co-synthesis, dynamic reconfiguration, FPGA, conditional task graph.

1. Wstęp

Rozwój technologii układów FPGA pozwala na projektowanie coraz bardziej złożonych systemów wbudowanych. Istnieje też tendencja w kierunku tworzenia architektur wieloprocesorowych, a takie możliwości dają również współczesne układy FPGA. Tym samym rozwijane są systemy SOPC pozwalające na integrację całego systemu w jednym układzie. Wykorzystanie dynamicznej rekonfiguracji systemów wbudowanych powinno, przy krótkich czasach reprogramowania, umożliwiać uzyskanie systemów szybszych i tańszych, poprzez wielokrotne wykorzystanie tych samych zasobów dla różnych funkcjonalności.

W praktyce, wiele funkcji realizowanych przez systemy wbudowane są to zadania wzajemnie się wykluczające (ZWW). Specyfikacja takich zadań jest możliwa przy pomocy warunkowych grafów zadań. Mając informacje o takich zadaniach i wykorzystując dynamiczną rekonfigurację systemu możliwe jest przyporządkowanie tych zadań do tych samych zasobów sprzętowych.

W ten sposób w znacznym stopniu można zmniejszyć powierzchnię zajmowaną przez system, a w wolnej przestrzeni alokować inne zadania sprzętowe, co również może prowadzić do zwiększenia szybkości systemu.

W pracy zostanie zaprezentowany rafinacyjny algorytm kosyntezy wieloprocesorowych, dynamicznie rekonfigurowalnych systemów wbudowanych (COSEDYRES-CTG), który maksymalizuje szybkość projektowanego systemu DRSOPC przy zadanym ograniczeniu powierzchni układu FPGA. Jest to pierwszy algorytm wykorzystujący informacje o zadaniach ZWW do optymalizacji systemów dynamicznie rekonfigurowalnych.

W kolejnym rozdziale jest przedstawiony przegląd dotychczas stosowanych rozwiązań w zakresie kosyntezy systemów dynamicznie rekonfigurowanych. Rozdział 3 zawiera sformułowanie problemu. W rozdziale 4 został opisany algorytm COSEDYRES-CTG. W rozdziale 5 zaprezentowano przykład optymalizacji systemu wbudowanego z wykorzystaniem informacji o ZWW. Rozdziały 6 i 7 zawierają wyniki eksperymentów oraz wnioski.

2. Metody kosyntezy systemów dynamicznie rekonfigurowalnych

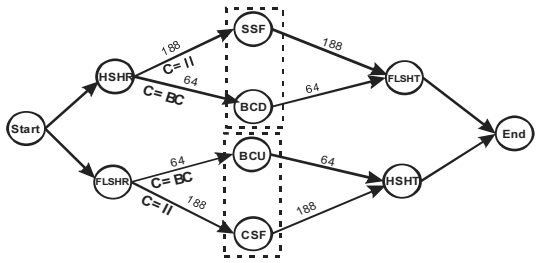
W wielu istniejących metodach kosyntezy systemów dynamicznie rekonfigurowanych [1-3] stosuje się układy reprogramowane w całości i nie wykorzystuje się możliwości częściowej rekonfiguracji. Poza tym stosowane są architektury wieloukładowe, zatem metody te nie nadają się dla systemów SOPC. Algorytm SLOPES [4] wykorzystuje możliwości układów częściowo reprogramowalnych firmy Xilinx. Nie bierze się natomiast pod uwagę rozmieszczenia rekonfigurowalnych modułów w układzie, podobnie jak w [5]. Algorytm [6], minimalizujący pobór mocy, uwzględnia implementację wieloprocesorową w postaci SOPC i wykorzystuje częściowo reprogramowalne układy FPGA, ale stosowany jest bardzo uproszczony model w postaci grafu liniowego. Metodę projektowania dla dynamicznie rekonfigurowalnych systemów przedstawiono również w [7]. Architektura zawiera wbudowany w FPGA procesor, który dynamicznie zmienia architekturę systemu, ale metoda nie jest przeznaczona dla systemów wieloprocesorowych.

Nie ma metody, która uwzględniałaby wszystkie cechy istotne dla współczesnych układów i wymagań. Rozwój technologii układów FPGA powoduje, że niektóre z metod są już nieaktualne. Na przykład, jedynie w algorytmie [8] uwzględniono ograniczenia dotyczące rozmieszczenia rekonfigurowalnych modułów w układach częściowo reprogramowalnych FPGA. W wielu metodach stosuje się architektury wieloukładowe, metody te nie nadają się dla systemów SOPC. Żadne, ze znanych z literatury, metod kosyntezy dynamicznie rekonfigurowalnych systemów wbudowanych nie uwzględnia implementacji systemu w formie wieloprocesorowego systemu DRSOPC.

Metody kosyntezy operują na reprezentacji wewnętrznej (modelu) systemu, tworzonej na podstawie specyfikacji w formie komunikujących się procesów (zadań). Najczęściej przyjmowanym modelem systemu jest graf zadań [1-3, 8, 9], ale w kilku pracach uwzględniono również specyfikację systemu w formie grafu zadań z rozejściami warunkowymi [10, 11]. Nie są to jednak metody przeznaczone dla systemów dynamicznie rekonfigurowalnych. W prezentowanej pracy przyjęto reprezentację systemu w postaci warunkowego grafu zadań. W żadnym z istotnych algorytmów kosyntezy dla systemów DRSOPC nie uwzględniono możliwości warunkowego wykonania zadań.

3. Sformułowanie problemu

System wbudowany reprezentowany jest w postaci warunkowego grafu zadań *CTG (V, E_s, E_c)*. Zakłada się, że od jednego węzła v_i, reprezentującego *i*-te zadanie, może wychodzić dowolna liczba krawędzi warunkowych, ale wszystkie warunki muszą być zakończone w jednym wspólnym węźle. Warunki mogą być zagnieżdżone. Krawędzie c_{ij} reprezentują komunikację pomiędzy zadaniami v_i i v_j. Na rys. 1 przedstawiono specyfikację modułu Translatora Transakcji w koncentratorze USB2.0 w postaci warunkowego grafu zadań.



Rys. 1. Graf zadań z rozejściami warunkowymi dla Translatora Transakcji
Fig. 1. Conditional task graph for Transaction Translator

W bibliotece komponentów systemu SOPC znajdują się trzy rodzaje zasobów: rdzenie procesorów uniwersalnych (*GPP*), komponenty sprzętowe (*VC*) i łącza komunikacyjne (*CL*). W celu zmniejszenia złożoności obliczeniowej metody kosyntezy wprowadza się pojęcie sektora dynamicznie rekonfigurowalnego *RS* - fragmentu układu FPGA o stałej wielkości, w której może być alokowanych kilka *VC*. Parametry zadań i komponentów dostępnych w bibliotece przedstawiono w tabeli 1.

Tab. 1. Parametry zadań dla Translatora Transakcji
Tab. 1. Library of software and hardware components for TT

Zadanie	SW (100 CLB)	HW	
	t [µs]	t [µs]	S [CLB]
HSHR	2400	20	400
HSHT	2400	20	400
SSF	1800	12	100
CSF	2000	12	135
BCD	1600	12	120
BCU	1600	12	120
FLSHR	1400	8	170
FLSHT	1400	8	170

Niech architektura systemu składa się z procesorów GPP_i (i=1, …, p), modułu sterującego rekonfiguracją GPP_r o powierzchni Su_r, sektorów RS_j (i=p+1,…, r) i łączy komunikacyjnych CL_j (j=1,…, c). Powierzchnia całkowita systemu DRSOPC jest zdefiniowana następująco:

$$S = \sum_{i=1}^{p} Su_i + \sum_{i=p+1}^{r} S_{RS\,i} + \sum_{j=1}^{c} Sc_j + Su_r \qquad (1)$$

Celem algorytmu COSEDYRES-CTG jest znalezienie najszybszej, dynamicznie rekonfigurowanej architektury, spełniającej wszystkie wymagania funkcjonalne określone w grafie *CTG* i ograniczenia układu FPGA.

4. Algorytm COSEDYRES-CTG

Algorytm COSEDYRES-CTG prezentowany w tej pracy jest rozwinięciem metody opisanej w [12] i [13] o możliwość reprezentacji systemu DRSOPC w postaci warunkowego grafu zadań. Działanie algorytmu polega na stopniowej rafinacji rozwiązań, startując od pewnego rozwiązania początkowego. W algorytmie na etapie inicjalizacji odbywa się generacja dostępnych rozmiarów sektorów oraz generacja rozwiązania początkowego, a następnie wykonywana jest rafinacja systemu poprzez tworzenie kolejnych rozwiązań na drodze modyfikacji poprzednich.

4.1. Inicjalizacja

Na początku generowane są sektory w taki sposób, aby wielkości tych sektorów były możliwie jak najbardziej zróżnicowane. Spośród wszystkich sum powierzchni *VC*, wybierane są te sumy, które najczęściej się powtarzają lub mają zbliżoną wielkość, a największy sektor jest nie mniejszy od powierzchni największego *VC*.

Znalezienie w warunkowym grafie zadań wszystkich zadań ZWW może być stosunkowo trudne, gdy istnieje wiele różnych warunków i gdy te warunki są dodatkowo zagnieżdżone. W celu identyfikacji wszystkich wykluczających się wzajemnie zadań w grafie stosowany jest algorytm etykietowania grafu zadań. Wykonanie etykietowania, jako pierwszego etapu w algorytmie COSEDYRES-CTG, ułatwia analizę zadań ZWW w kolejnych krokach algorytmu.

W rozwiązaniu początkowym wszystkie zadania grafu *CTG* przydzielone są do jednego procesora uniwersalnego. Rozwiązanie takie pozwala na największą „elastyczność" w poszukiwaniu kolejnych rozwiązań (największa przestrzeń rozwiązań).

4.2. Rafinacja

Stosowane metody rafinacji decydują o złożoności obliczeniowej algorytmu. W każdym kroku algorytmu rafinacyjnego COSEDYRES-CTG rozpatrywane są różne modyfikacje aktualnego rozwiązania i do następnego kroku wybierane jest rozwiązanie dające największy zysk (który zależy nie tylko od przyrostu szybkości systemu, ale także możliwości optymalizacji w dalszych krokach – wolnej przestrzeni w FPGA). Uwzględniane są następujące modyfikacje: dodanie zasobu (*GPP* lub *RS*) oraz usunięcie zasobu i przeniesienie wcześniej przyporządkowanych mu zadań do innych zasobów. Obie modyfikacje mogą być wykonane w jednym kroku algorytmu. Więcej szczegółów na temat miary jakości rozwiązań i stosowanych metod rafinacji podano w [13]. Jeśli zadania ZWW są przydzielone do tego samego sektora, to powierzchnia zajmowana przez nie wynosi $max(S_{vc}(v_i),\ldots, S_{vc}(v_{i+n}))$, gdzie $S_{vc}(v_i)$ jest powierzchnią zajmowaną przez komponent sprzętowy realizujący zadanie v_i. Tym samym możliwe jest zmniejszenie powierzchni zajmowanej przez system. Opłacalne może być przydzielenie kilku zadań wykluczających się wzajemnie do jednego sektora, gdyż, aby wykonać jedno z takich zadań nie jest konieczna rekonfiguracja, dopiero po zmianie warunku, inne zadania mogą być alokowane w tych samych sektorach układu w wyniku reprogramowania. W przypadku przydzielenia zadań ZWW do jednego *GPP* zadania takie mogą być uszeregowane równolegle (wykonywane w zależności od warunku). Tym samym przydzielając zadania ZWW do jednego sektora zmniejsza się powierzchnię systemu, a poprzez przydział takich zadań do *GPP* również czas obliczeń przez ten procesor może być mniejszy, niż w przypadku nie uwzględnienia informacji o warunkowym wykonaniu zadań. Aby to wykorzystać konieczne jest uwzględnienie tych informacji w algorytmie szeregowania. W metodach rafinacji uwzględniono kilka zmian w stosunku do algorytmu [13].

W czasie przenoszenia zadania v_i do/z innych zasobów, algorytm sprawdza, czy nie występują jeszcze inne zadania, które wzajemnie wykluczają się z v_i. Jeśli jest kilka takich zadań, to następuje próba przeniesienia równocześnie wszystkich takich zadań do dodawanego zasobu, natomiast przy usuwaniu zasobu – również przesunięcia zadań ZWW do jednego zasobu. Przenosząc taką grupę zadań jest bardzo duże prawdopodobieństwo zmniejszenia kosztu systemu, a co za tym idzie zwiększenie szybkości w dalszych krokach.

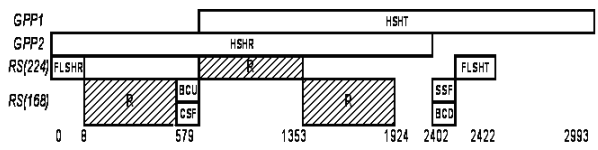
5. Przykład

Przykładem systemu wbudowanego jest koncentrator USB2.0 (ang. *Hub*). Jednym z bardziej złożonych modułów koncentratora jest translator transakcji (TT). Do minimalizacji całkowitego czasu wykonania wszystkich procesów modułu TT można wykorzystać algorytm COSEDYRES-CTG. Specyfikacja TT (Rys.1) składa się z następujących procesów: HSH –implementujący komunikację pomiędzy TT i Hostem, który można podzielić na dwa: **HSHR** (odbieranie danych z Hosta) i **HSHT** (przesyłanie danych do Hosta); **SSF** - buforujący transmisje okresowe od Hosta do urządzeń; **CSF** - buforujący transmisje okresowe z urządzeń do Hosta; BC - buforujący transmisje blokowe i sterujące (podzielony na: **BCU** i **BCD**). Na podstawie specyfikacji [14] można uzyskać 2 rozłączne grafy zadań, jeden opisujący transmisję z Hosta do urządzeń, drugi transmisję w przeciwnym kierunku. Procesy SSF i BCD oraz BCU i CSF wzajemnie się wykluczają. Informacja o zadaniach ZWW może pozwolić na lepszą optymalizację systemu.

Koncentrator USB zaimplementowano w układzie FPGA Xilinx Virtex 2 XC2V2000 o powierzchni **2688 CLB**. Zakłada się, że dostępny jest jeden typ procesora 10MHz, który może być zaimplementowany w postaci modułu o powierzchni 100CLB. Wielkości transmitowanych danych w TT zaznaczono w grafie z rys. 1 jako etykiety odpowiadających im krawędzi. Czasy transmisji wynoszą odpowiednio:

- przy przesyłaniu 188 bajtów: 188*8/64 cykle z częstotliwością 10 MHz, czyli **2350ns**,
- przy przesyłaniu 64 bajtów: 64*8/64 cykle z częstotliwością 10 MHz, czyli **800ns**.

Przyjęto ograniczenie powierzchni dla modułu TT: **800 CLB**. Dla tak określonych parametrów bez uwzględnienia rozejść warunkowych całkowity czas wykonania zadań wynosi 4408μs przy powierzchni 741CLB (3 zadania wykonywane są sprzętowo bez dynamicznej rekonfiguracji). W wyniku kosyntezy dla warunkowego grafu zadań otrzymano architekturę składającą się z 2 procesorów oraz dwóch sektorów dynamicznie rekonfigurowalnych. Zadania wzajemnie się wykluczające zostały umieszczone w tym samym zasobie sprzętowym (sektorze). Dzięki temu nie jest konieczne jednoczesne alokowanie takich zadań, można je alokować w systemie w zależności od potrzeb (warunków). Tym samym powierzchnia zajmowana przez nie zmniejsza się dwukrotnie i zmniejsza się powierzchnia całego systemu. Uszeregowanie zadań dla takiej implementacji pokazano na rys. 2.



Rys. 2. Wykres Gantta dla TT w postaci DRSOPC reprezentowanego przez CTG
Fig. 2. Gantt chart for TT implemented as DRSOPC and described by CTG

Całkowity czas wykonania wszystkich zadań wynosi: **2993μs**, a zajmowana powierzchnia **760 CLB**. Dla takiego ograniczenia powierzchni (800 CLB) niemożliwe byłoby uzyskanie szybszej architektury, nawet z wykorzystaniem dynamicznej rekonfiguracji, ale bez uwzględnienia rozejść warunkowych. Jak się okazuje można zaimplementować architekturę tańszą o **33%** i szybszą o **27%** w stosunku do implementacji SOPC.

6. Wyniki eksperymentów

W celu pokazania możliwości optymalizacji systemów DRSOPC, w przypadku reprezentowania systemu przez warunkowy graf zadań, wykonano eksperymenty dla kilku losowych grafów. Porównano wyniki uzyskane dla systemu reprezentowanego przez grafy CTG z wynikami dla systemów reprezentowanych przez grafy TG. W obu przypadkach wszystkie parametry w bibliotece komponentów były takie same, również taka sama była struktura grafu zadań. W bibliotece komponentów dostępne były: jeden typ procesora *GPP*, po jednym typie *VC* dla każdego zadania, jeden typ łącza komunikacyjnego. Dostępna powierzchnia była zwiększana proporcjonalnie wraz ze wzrostem wielkości grafu. Wraz ze wzrostem liczby węzłów w grafie zwiększano również proporcjonalnie liczbę wzajemnie się wykluczających zadań. Kolejne kolumny tabeli 2 oznaczają: liczbę zadań grafu, ograniczenie powierzchni dla systemu wbudowanego, dla każdego z algorytmów kolejno: czas wykonania zadań i powierzchnię zajmowaną przez system DRSOPC, liczbę zadań wykonywanych sprzętowo (HW). Dwie ostatnie kolumny to: liczba zadań ZWW w grafie CTG i procentowy wzrost szybkości systemów DRSOPC reprezentowanych przez grafy CTG w stosunku do szybkości systemów reprezentowanych przez grafy TG.

Tab. 2. Wyniki eksperymentów
Tab. 2. Experimental results

Graf	ograniczenie powierzchni	COSEDYRES			COSEDYRES-CTG			liczba par zadań ZWW	przyrost szybkości [%]
		czas	powierzchnia	HW	czas	powierzchnia	HW		
10	1500	**941**	1498	4	**804**	1490	5	1	**15**
30	2000	**4642**	1863	13	**3448**	1986	15	2	**26**
50	2500	**7054**	2303	6	**6531**	2373	13	3	**8**
70	3000	**9174**	2935	21	**8396**	2964	35	4	**9**

Wyniki eksperymentów pokazują, że uwzględnienie informacji o zadaniach ZWW w grafie zadań prowadzi do zwiększenia szybkości systemu DRSOPC otrzymanego w wyniku zastosowania algorytmu COSEDYRES-CTG. Dzięki dynamicznej rekonfiguracji w wyniku przydzielenia zadań ZWW do jednego sektora zwiększa się powierzchnia dla zadań wykonywanych sprzętowo, gdyż zadania ZWW zajmują wtedy tą samą powierzchnię układu FPGA. Wzrost szybkości zależy jednak od liczby zadań ZWW w grafie CTG. Jeśli liczba zadań ZWW, w stosunku do liczby wszystkich zadań w grafie, jest niewielka, to również wzrost szybkości nie będzie duży, bo korzyść wynikająca z równoległego uszeregowania zadań ZWW w tym samym zasobie zostanie zniwelowana przez zrównoleglenie pozostałych zadań. Również istotne jest to, gdzie zadania ZWW są usytuowane w warunkowym grafie zadań. Jeżeli znajdują się na ścieżce krytycznej, to korzyść z umieszczenia ich w jednym zasobie będzie większa, niż gdyby były wykonywane na ścieżkach w mniejszym stopniu decydujących o szybkości całego systemu (w eksperymentach zadania ZWW zostały rozmieszczone losowo).

7. Wnioski

W pracy przedstawiono algorytm kosyntezy, który umożliwia syntezę systemów DRSOPC reprezentowanych przez CTG. Opis systemu w postaci grafu CTG daje większe możliwości optymalizacji systemów wbudowanych przy wykorzystaniu dynamicznej rekonfiguracji, w celu uzyskania jeszcze szybszego systemu. Mając informacje o zadaniach ZWW, zadania te, jeśli są alokowane w tym samym sektorze układu FPGA, mogą być przydzielone do tego samego fragmentu układu, zajmując tym samym mniejszą powierzchnię i zostawiając więcej miejsca dla innych zadań realizowanych sprzętowo. Zgodnie z naszą wiedzą jest to pierwszy

algorytm kosyntezy wieloprocesorowych systemów DRSOPC specyfikowanych w formie warunkowego grafu zadań.

Prace nad algorytmami kosyntezy wbudowanych systemów rozproszonych są kontynuowane. Badane będą inne możliwości rekonfiguracji systemu, inne sposoby reprezentacji systemu. Prace będą też skierowane na udoskonalenie metod rafinacji systemów DRSOPC.

8. Literatura

[1] R. P. Dick , N. K. Jha, "CORDS: Hardware-Software Co-synthesis of Reconfigurable Real-time Distributed Embedded Systems", Proc. of the 1998 International Conference on Computer-Aided Design, pp. 62-67, 1998.

[2] K. S. Chatha , R. Vemuri, "Hardware-Software Codesign for Dynamically Reconfigurable Architectures", Proc. FPL'99, pp. 175-184, 1999.

[3] K. B. Chehida, M. Auguin, "HW/SW Partitioning Approach for Reconfigurable System Design", Proc. CASES 2002, pp. 247-251, 2002.

[4] L. Shang, R. P. Dick, N. K. Jha, " SLOPES: Hardware–Software Cosynthesis of Low-Power Real-Time Distributed Embedded Systems With Dynamically Reconfigurable FPGAs", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, pp. 508-526, 2007.

[5] R. Huang, R. Vemuri, "On-Line Synthesis for Partially Reconfigurable FPGAs", Proc. of the 18th International Conference on VLSI Design held jointly with 4th International Conference on Embedded Systems Design (VLSID'05), pp. 663-668, 2005.

[6] J. Ou, S. B. Choi, V. K. Prasanna, „Energy-Efficient Hardware/Software Co-synthesis for a Class of Applications on Reconfigurable SoCs", International Journal of Embedded Systems 2005 - Vol. 1, No.1/2, pp. 91 - 102, 2005.

[7] F. Ferrandi, M.D. Santabrogio, D. Sciuto, "A Design Methodology for Dynamic Reconfiguration: The Coronte Architecture", 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05) – Workshop 3, pp. 163-166, 2005.

[8] S. Banerjee, E. Bozorgzadeh, N. Dutt, "Physically-aware HW-SW Partitioning for Reconfigurable Architectures with Partial Dynamic Reconfiguration", Proc. DAC'05, pp. 335-340, 2005.

[9] Y. Qu, J.-P. Soininen, J. Nurmi, "A Parallel Configuration Model for Reducing the Run-time Reconfiguration Overhead", Proc. DATE'06, pp. 965-969, 2006.

[10] A. Daboli, P. Eles, "Scheduling Under Data and Control Dependencies for Heterogeneous Architectures", Proc. of the ICCD 1998, pp. 602-608, 1998.

[11] Y. Xie, W.Wolf, "Allocation and Scheduling of Conditional Task Graph in Hardware/Software Co-synthesis", Proc. DATE'01, pp. 620-625, 2001.

[12] R. Czarnecki, S. Deniziak, "Resource Constrained Co-synthesis of Self-reconfigurable SOPCs", Proc. DDECS'07, pp. 49-54, 2007.

[13] R. Czarnecki, S. Deniziak, "Kosynteza samorekonfigurowalnych systemów SOPC", Czasopismo Techniczne Politechniki Krakowskiej, Informatyka, z.1-I/2007, Zeszyt 7, str. 3-16, 2007.

[14] S. Deniziak, "Metodologia szybkiego prototypowania systemów cyfrowych", Wydawnictwo Politechniki Krakowskiej, 2005.

Artykuł recenzowany