Andrzej Stasiak
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI

Algorytm dekompozycji specyfikacji funkcjonalnej sprzętowo-programowej mikrostruktury cyfrowej

dr inż. Andrzej STASIAK

Absolwent Uniwersytetu Zielonogórskiego. Asystent w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Zakres tematyczny prowadzonych badań obejmuje zagadnienia zintegrowanego projektowania sprzętu i oprogramowania mikrosystemów cyfrowych. Zainteresowania skupiają się w szczególności na projektowaniu i implementacji systemów osadzonych w układach reprogramowalnych klasy SOPC, z wykorzystaniem języków opisu sprzętu (VHDL, Verilog).

e-mail: A.Stasiak@iie.uz.zgora.pl

Streszczenie

W procesie projektowania zintegrowanych sprzętowo-programowych systemów cyfrowych, wymagane są właściwe metody i algorytmy pozwalające osiągnąć spodziewany a pożądany rezultat. Jednym z etapów projektowania jest dekompozycja funkcjonalna, którą determinuje algorytm podziału specyfikacji systemu na wyodrębnione bloki zadaniowe. Artykuł przedstawia, opracowany na rzecz pro0jektowania zintegrowanego, algorytm dekompozycji funkcjonalnej sprzętowo-programowej mikrostruktury cyfrowej.

Abstract

There are required appropriate methods and algorithms to achieve expected and requested results designing hybrid hardware-software digital microsystem . The decomposition process is one of the most important stage of each known codesign method. It is determinate by an algorithm that do partitioning of a given behavior specification and assign particular system functions to hardware or software sets. This paper presents a new decomposition algorithm that is dedicated for new elaborated hardware-software digital microstructure but may by use for other embedded hybrid microsystems.

Słowa kluczowe: projektowanie zintegrowane, sieci Petriego, model formalny, systemy osadzone, systemy cyfrowe, mikro systemy cyfrowe, FPGA

Keywords: hardware-software co-design, Petri nets, formal model, embedded systems, digital systems, digital microsystems, FPGA, PLD

1. Sprzętowo-programowa mikrostruktura cyfrowa.

Do realizacji systemów, mikrosystemów i mikrostruktur cyfrowych, wykorzystuje się układy reprogramowalne na bazie rozwiązań FPGA [2,21]. W większości rozpatrywanych projektów, po procesie projektowym SOPC, w układzie FPGA pozostaje pewien obszar niewykorzystanych zasobów sprzętowych. Średni procent wykorzystania układu FPGA w projektach realizowanych na świecie waha się w przedziale 70-90% [2,21]. Jednocześnie, dla każdego realizowanego projektu, obszar wolnych zasobów logiki reprogramowalnej jest różny. W pracy [20] proponuje się wykorzystać wolną logikę programowalną systemu SOPC jako akcelerator sprzętowy, tj. moduł wspomagający pracę mikroprocesora; którego zadaniem jest zwiększenia wydajności pracy procesora ze względu na czas wykonywania przydzielonych zadań sterowania i przetwarzania danych. Sprzętowo-programową mikrostrukturą cyfrową SPMC określa się ściśle zintegrowany **s**przętowo-**p**rogramowy **s**ystem **c**yfrowy, będący składową większego mikrosystemu cyfrowego rezydującego w układzie FPGA, składający się z rdzenia procesora ogólnego przeznaczenia oraz wolnej logiki reprogramowalenj układu FPGA.

2. Dekompozycja funkcjonalna SPMC

Problematyka algorytmu dekompozycji funkcjonalnej (podziału) sprzętowo-programowej mikrostruktury cyfrowej, ze względu na zdefiniowaną architekturę SPMC oraz zdecydowanie mniejszy zakres obszaru poszukiwań rozwiązań, pozbawiona jest problemów występujących w znanych metodach kosyntezy [3,10] systemów cyfrowych, tj. poszukiwania optymalnych architektur procesorów, wyznaczania technologii części sprzętowej (ASIC,FPGA,CPLD), przyporządkowania zadań, poboru mocy, i inne. Do opracowania i realizacji algorytmu dekompozycji funkcjonalnej SPMC wskazane jest zastosowanie algorytmów wyszukiwania wyczerpującego [5] lub metod opierających się na programowaniu liniowym całkowitoliczbowym [14]. Z drugiej strony, korzystnym jest zaprzęgnięcie metod heurystycznych, wykorzystywanych w syntezie systemowej, takich jak: algorytmy konstrukcyjne [4,6], rafinacyjne [9,12] lub algorytmy probabilistyczne [7]. W procesie dekompozycji funkcjonalnej SPMC, algorytm podziału dokonuje szacowania, kwalifikując poszczególne zadania do realizacji programowej [17] lub sprzętowej. Za model formalny sprzętowo-programowej mikrostruktury cyfrowej oznaczono PNHSDM [18], natomiast zapis elektroniczny specyfikacji wejściowej podawany jest w formacie SPNF [15].

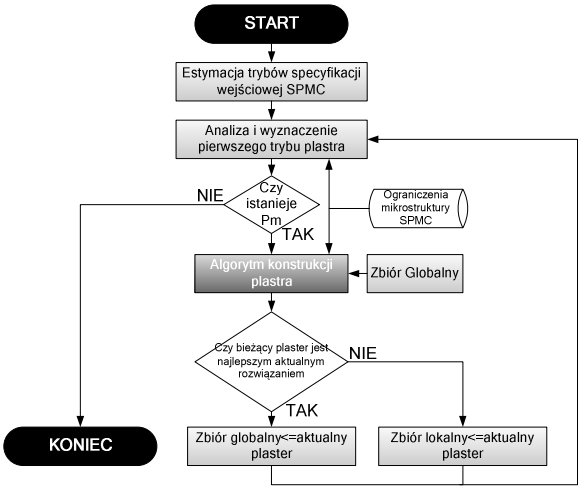
3. Algorytm dekompozycji SPMC

Proponowany algorytm dekompozycji funkcjonalnej SPMC jest wynikiem studiów literaturowych oraz badań nad eksperymentalnym systemem dekompozycji funkcjonalnej ADSO [19]. Algorytm SPMC zorientowany jest na grupowanie zadań części programowej i sprzętowej oraz eliminację ziarnistości i rozproszenia realizacji funkcjonalnej SPMC. Ze względu na definiowanie rozwiązań dekompozycji w postaci plastrów zadaniowych oraz technikę wychodzenia z lokalnych minimów (wyznaczanie kolejnych punktów startowych z poza bieżącego obszaru analizy), algorytm SPMC można zaliczyć do algorytmów heurystycznych, z grupy konstrukcyjnych. Wybrane rozwiązania algorytmu SPMC wzorowane były na opracowaniach algorytmu COSYM [6]. Pod względem koncepcyjnym, są one do siebie zbliżone. Natomiast znacząco różnią się algorytmem budowy plastra zadaniowego, co wynika bezpośrednio z rozważanej architektury mikrostruktury SPMC. Za *plaster zadaniowy SPMC* oznacza się integralny/nierozłączny zbiór sąsiadujących ze sobą miejsc i tranzycji modelu PNHSDM [18] zakwalifikowany do realizacji programowej lub sprzętowej. Konieczność budowy tzw. plastra zadaniowego wynika z własności pracy systemów heterogenicznych. Jednym z głównych źródeł spadku wydajności pracy systemów heterogenicznych, o ścisłej integracji części programowej i sprzętowej, jest proces wewnętrznej komunikacji sprzęt-program, dotyczący synchronizacji danych i procesów sterowania. Przeprowadzone badania [16] oraz rozważania literaturowe [11] wskazują na konieczność optymalizacji i minimalizacji punktów komunikacyjnych w sprzętowo-programowej mikrostrukturze cyfrowej. Algorytm budowy plastra zadaniowego dąży do minimalizacji punktów komunikacyjnych w architekturze programowo-sprzętowej mikrostruktury cyfrowej. Diagram pracy algorytmu SPMC przedstawia rysunek 3.1.

Za realizację początkową oznacza się realizację programową. Pierwszym etapem jest wyznaczenie początkowego miejsca P_m sieci dla pierwszego sprzętowego plastra zadaniowego. Miejsce P_m sieci musi spełniać następujące warunki wyboru:

1. Miejsce znajduje się w podzbiorze sieci o największym współczynniku współbieżności.
2. Miejsce ma najgorsze parametry realizacji programowej.
3. Miejsce ma najkorzystniejsze parametry realizacji sprzętowej.
4. Miejsce ma największy współczynnik aktywności funkcjonalnej.
5. Miejsce nie należy do żadnego znalezionego rozwiązania zbioru lokalnego i globalnego.

6. Miejsce ma największą liczbę miejsc przyległych do końcowej (współbieżnej) tranzycji zbiorczej.



Rys. 3.1. Algorytm dekompozycji funkcjonalnej SPMC

Algorytm dekompozycji funkcjonalnej SPMC składa się z dwóch części: AWPN, AKP. W pierwszej, pętli nadrzędnej, pracuje algorytm AWPN, który odpowiedzialny jest za: a) wyznaczanie punktu P_m będącego punktem startowym kolejnego plastra zadaniowego, b) zarządzanie znalezionymi rozwiązaniami (lokalnym i globalnym), rysunek 3.1. Druga część to algorytm konstrukcji sprzętowego plastra zadaniowego, który realizuje proces konstrukcji pojedynczego plastra sieci Petriego na podstawie podanego punktu startowego P_m. W procesie pracy algorytmu dekompozycji SPMC, wyznaczanych jest maksymalnie PN plastrów zadaniowych, gdzie PN jest liczbą miejsc sieci Petriego specyfikującej zachowanie mikrostruktury cyfrowej pomniejszoną o zbiór miejsc już odwiedzonych. Wyodrębniono dwa zbiory miejsc sieci Petriego, gdzie składowane są wyniki przeprowadzonych analiz i rozwiązań. Zbiór globalny przechowuje aktualnie najkorzystniejsze znalezione rozwiązanie, natomiast zbiór lokalny zapamiętuje wszystkie znalezione rozwiązania. W procesie wyznaczania kolejnego punktu początkowego plastra P_m, mogą brać udział miejsca sieci nie oznaczone wcześniej jako punkt P_m. Proces dekompozycji SPMC kończy pracę w chwili, gdy nie można wyznaczyć kolejnego punktu początkowego P_m plastra (znaleziono wszystkie możliwe rozwiązania) lub aktualny zbiór globalny spełnia minima dekompozycji. Szkic algorytmu przedstawia rysunek 3.2.

```
Alg.3.1
estymacja kosztów realizacji i czasu ∀m∈M
Zbiór.Globalny=0;
outher_loop: do
   Pm:=wyznacz_miejsce_początkowe_klastra(Zbiór.Miejsc);
   Zbiór.Aktualny=uruchom_algorytm_plastra(Pm, Zbiór.Globalny);
   if Zbiór.Aktualny(zysk)>Zbiór.Globalny(zysk)
      Zbiór.Globalny:=Zbiór.Aktualny;
   Zbiór.Lokalny:= Zbiór.Lokalny +ZbiórAktualny;
   Zbiór.Miejsc:=Zbiór.Miejsc-Zbiór.Aktualny;
while((Pm ≠∅) || (Zbiór.Globalny(zysk) >= minimum))
rozwiazanie:=Zbiór.Globalny;
```

Rys. 3.2. Algorytm zarządzania plastrami zadaniowymi oraz wyznaczaniem punktów startowych

Zadanie znalezienia najbardziej korzystnego rozwiązania spoczywa na algorytmie konstrukcji plastra AKP. Od niego zależy, który element składowy sieci specyfikującej zachowanie SPMC, zostanie przypisany do realizacji sprzętowej, a który do programowej. W tym celu, niezbędne jest wyznaczenie funkcji zysku podziału, na podstawie której algorytm konstrukcji plastra będzie podejmował decyzje alokacji wybranego zadania do części sprzętowej lub programowej. Dla rozwiązań SPMC funkcja kosztów przeniesienia wybranego zadania do części sprzętowej wyrażona jest równaniem nr 1:

$$\Delta t = \sum_{\forall M \in S} tp(M) - \sum_{\forall M \in P} tp(M) + \sum_{\forall M \in P} ts(M) + Kc_{ps} + Kc_{sp} \quad (1)$$

gdzie:

S–zbiór miejsc specyfikacji SPMC,
P–zbiór miejsc plastra zadaniowego,
Kc_{sp}–koszt czasu transmisji sprzęt→program,
Kc_{ps}–koszt czasu transmisji program→sprzęt,
tp(M)–czas pracy miejsca specyfikacji SPMC jako realizacja programowa,
tp(M)–czas pracy miejsca plastra zadaniowego jako realizacja programowa,
ts(M)–czas pracy miejsca plastra zadaniowego jako realizacja sprzętowa,
Δt–różnica czasu po przeniesieniu miejsca do części sprzętowej – **zysk podziału**.

Zysk podziału rozumiany jest jako skrócenie (przyrost ujemny) czasu przetwarzania zadanej specyfikacji przez mikrostrukturę SPMC. Zysk Δt jest różnicą sumy czasów tp(M) wykonania zadań miejsc M specyfikacji wejściowej S przez mikroprocesor i sumy czasów wykonania miejsc M sprzętowego plastra zadaniowego P, zwiększoną o koszty komunikacji Kc oraz sumy czasów wykonania miejsca M specyfikacji plastra P jako realizacji sprzętowej.

Algorytm dekompozycji funkcjonalnej SPMC dokonuje przeniesienia wybranego miejsca pracy sieci z części programowej do sprzętowej przy spełnionej nierówności:

$$\Delta t_{actual} < \Delta t_{previous} \quad (2)$$

gdzie:

t_{aktual}– czas wykonania specyfikacji SPMC po przeniesieniu miejsca *M* do części sprzętowej

$t_{previous}$– czas wykonania specyfikacji SPMC przed przeniesieniem miejsca M do części sprzętowej.

Algorytm AKP dokonuje wstępnej kwalifikacji miejsca M specyfikacji SPMC jako realizację sprzętową według następujących kryteriów:

- miejsce nie jest wykluczone z bieżącego rozwiązania (możliwe jest podanie zbioru miejsc wykluczonych z poszukiwań rozwiązania dekompozycji funkcjonalnej),
- miejsce posiada największy współczynnik aktywności funkcjonalnej,
- miejsce w relacji zależności przepływu danych ,
- miejsce położone jest najbliżej punktu startowego P_m plastra P,
- miejsce posiada najgorsze parametry realizacji programowej,
- miejsce posiada najkorzystniejsze parametry realizacji sprzętowej,
- nie przekroczono zasobów sprzętowych,
- osiągnięte zyski spełniają nierówność.

Algorytm konstrukcji plastra zadaniowego SPMC operuje na miejscach sieci przydzielając do wspólnej grupy wybrane zadania ze względu na zyski czasu pracy SPMC oraz koszt realizacji aktualnego plastra. Nadrzędny proces dekompozycji funkcjonalnej SPMC, przekazuje do algorytmu plastra AKP wybrane miejsce początkowe P_m, będące punktem startowym poszukiwanego rozwiązania. Warunkiem pracy algorytmu jest wyznaczenie miejsca sąsiadującego P_{next} plastra, spełniającego kryteria kwalifikacji przedstawione wyżej. Następnie, badany jest koszt realizacji sprzętowej plastra wzbogaconego o nowe miejsce programowe oraz zysk czasu pracy SPMC przy wstępnej kwalifikacji miejsca P_{next} do części sprzętowej. Brak spełnionego warunku kosztu lub zysku skutkuje wykluczeniem miejsca P_{next} z aktualnie prowadzonych analiz. Szkic algorytmu przedstawia rysunek 3.3.

```
Alg.3.2
Zbior.Wykluczen=0;
Pactual=Pm;
while((Pnext:=wyznacz.miejsce.sasiadujace(S,Pactual,Zbior.Wykluczen))!=0){
   if (Ograniczenia.Systemu(SPMC)>Koszt(Pactual + Pnext))
      if Δt(S,Pactual+Pnext)<Δt(S,Pactual))
         Pactual+=Pnext;
      else
         Zbior.Wykluczen+=Pnext;
   else
      Zbior.Wykluczen+=Pnext;
return(Pactual);
```

Rys. 3.3. Algorytm pracy konstrukcji plastra zadaniowego AKP

Proces jest kontynuowany dopóty, dopóki można wyznaczyć kolejne miejsce Pnext sprzętowego plastra zadaniowego. Algo-

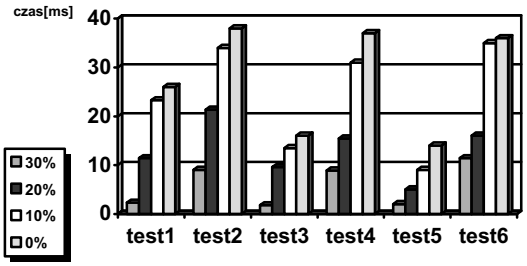
rytm zwraca aktualne znalezione rozwiązanie do pętli nadrzędnej. W szczególności, postępowanie pracy algorytmu wyznaczania sprzętowego plastra zadaniowego, a w szczególności punktu Pnext, przedstawiono w trzech następujących punktach:

1. Krok pierwszy.
   a) Dla plastra P znajdź sąsiada będącego w relacji sekwencji przepływu sterowania tylko z P_m, oznacz miejsce jeśli spełnione są warunki:
      i. miejsce nie jest wykluczone z bieżącego rozwiązania,
      ii. miejsce o najgorszych parametrach realizacji programowej,
      iii. miejsce o najkorzystniejszych parametrach realizacji sprzętowej,
      iv. nie przekroczono zasobów sprzętowych,
      v. osiągnięte zyski spełniają nierówność 2.
   b) Jeśli brak spełnienia warunków 1.a) v-vi dla znalezionego miejsca, odrzuć bieżące miejsce (oznakowanie wykluczające miejsce dla bieżącego rozwiązania). Przejdź do kroku nr 1a).
   c) Jeśli brak możliwych do wyznaczenia miejsc sekwencyjnych, przejdź do kroku nr 2.
   d) Jeśli znaleziono dwa lub więcej miejsc równorzędnych ze względu na punkty a)i-iv, przejdź do punktu 3.
   e) Przejdź do punktu 1a).
2. Krok drugi.
   a) Dla plastra P znajdź miejsce sąsiadujące P_m, będący w relacji współbieżności przepływu sterowania, według klucza:
      i. miejsce nie jest wykluczone z bieżącego rozwiązania,
      ii. miejsce o największym współczynniku aktywności funkcjonalnej,
      iii. miejsce w relacji zależności przepływu danych,
      iv. miejsce położone najbliżej punktu startowego P_m plastra K,
      v. miejsce o najgorszych parametrach realizacji programowej,
      vi. miejsce o najkorzystniejszych parametrach realizacji sprzętowej,
      vii. nie przekroczono zasobów sprzętowych,
      viii. osiągnięte zyski spełniają nierówność 2.
   b) Jeśli brak spełnienia warunków 2a) v-vi dla znalezionego miejsca, odrzuć bieżące miejsce (oznakowanie wykluczające miejsce dla bieżącego rozwiązania). Przejdź do kroku nr 1.
   c) Jeśli brak możliwych do wyznaczenia miejsc równoległych, koniec budowy plastra zadaniowego. Zwróć rozwiązanie do pętli głównej algorytmu nadrzędnego.
   d) Przejdź do kroku nr 1.
3. Realizuj przeszukiwanie wszystkie ścieżek równych miejsc do chwili znalezienie najlepszego rozwiązania spełniającego punkty1a)i-iv. W momencie niepowodzenia, wybierz rozwiązanie w sposób losowy. Przejdź do punktu 1a).

Sterowanie dalszymi krokami postępowania algorytmu dekompozycji SPMC, przejmuje algorytm nadrzędny. Rozwiązanie aktualne zostaje zapamiętane w zbiorze lokalnym. Jeżeli koszt realizacji i pracy mikrostruktury SPMC dla aktualnego plastra zadań jest mniejszy od kosztów najlepszego znalezionego rozwiązania mikrosystemu SPMC, wówczas aktualny plaster zostaje zaakceptowany i staje się plastrem globalnym.

4. Podsumowanie

Przeprowadzone badania [18] wykazały wzrost wydajności pracy SPMC, który zależny jest od liczby wolnych zasobów sprzętowych układu FPGA systemu SOPC. W zbiorze modeli testowych, dla których przeprowadzono badania algorytmu SPMC, znajdują się specyfikacje sterowania i przetwarzania stosowane w przemyśle (Kronopol [13], Zielonogórska Elektrociepłownia [22]) oraz przykłady literaturowe [1,8] wkomponowane w rozbudowane syntetyczne modele testowe. Wyniki pośrednie pracy algorytmu dekompozycji funkcjonalnej specyfikacji SPMC, przedstawia rysunek 4.1.



Rys. 4.1. Zależność wzrostu wydajności SPMC względem zdefiniowanej wolnej logiki reprogramowalnej układu FPGA

Oś X rysunku 4.1 reprezentuje numer testu, natomiast oś Y czas przetwarzania zadanej specyfikacji wejściowej przez mikrostrukturę SPMC. Dla każdego z testów przeprowadzono badania dotyczące wpływu liczby (obszaru) wolnej logiki reprogramowalnej, na czas wykonania specyfikacji programu przez jednostkę SPMC. Na podstawie doniesień [21], zdefiniowano przedział od 30% do 1% zasobów sprzętowych pozostających do zagospodarowania po procesie implementacji mikrosystemu cyfrowego w układzie FPGA. Testy przeprowadzono dla układu Xilinx Spartan2E o łącznej liczbie bloków konfiguracyjnych 1728 [21]. Analiza rysunku 4.1 pozwala zaobserwować znaczące skrócenie czasu wykonania określonego zadania przy wzroście obszaru wolnej logiki rekonfigurowalnej FPGA. Opracowany algorytm efektywnie, ze względu na czas i koszt realizacji systemu cyfrowego, przydziela zadania do realizacji programowej lub sprzętowej. Prowadzone są dalsze prace nad przedstawionym algorytmem, zmierzające do zbadania m.in. charakterystyki algorytmu dla zróżnicowanych specyfikacji funkcjonalnych systemów cyfrowych, w tym: a) ze względu na charakter specyfikowanych zadań (zadania współbieżne, zadania sekwencyjne i mieszane), b) typ realizowanych zadań (przetwarzanie danych, sterowanie, przesyłanie/przetwarzanie potokowe).

Literatura

[1] M.Adamski, Parallel Controller Implementation using standard PLD Software, in W. R. Moore, W. Luk (Eds.), FPGA, Abingdon EE&CS Books, Abingdon, England, 1991, pp. 296-304
[2] www.altera.com
[3] F.Balarin, M.Chiodo, P.Giusto, H.Hsieh, L.Lavagno, C.Passerone, A.Sangiovanni-Vincentelli, E.Sentovich, K.Suzuki, B.Tabbara, Hardware/Software Co-Design of Embedded Systems - The Polis Approach, Kluwer Academic Publishers, 1997, ISBN 0-7923-9936-6
[4] L.Bianco, M.Auguin, A.Pegatoquet, A path analysis based partitioning for time constrained embedded systems, Proc. of the Int. Workshop on Hardware/Software Codesign, 1998
[5] H.D'Ambrosio, X.Hu, Configuration-level hardware/software partitioning for real-time systems, Proc. of the Int. Workshop on Hardware/Software Codesign, 1994
[6] B.P.Dave, G.Lakshminarayana, N.K.Jha, COSYN: Hardware-Software Co-synthesis of Embedded Systems, Proc. Of the Design automation Conference, 1997, pp.703-708
[7] R.P.Dick, N.K.Jha, MOGAC: A multiobjective genetic algorithm for the cosynthesis of hardware-software embedded systems, Proc. of the Int. Conference on Computer Aided Design, 1997
[8] P.Eles, K.Kuchciński, Z.Pend, System Synthesis with VHDL, Kluwer Akademic Publishers, Londyn 1998, ISBN 0-7923-8082-7
[9] P.Eles, Z.Peng, K.Kuchciński, A.Doboli, System level hardware/software partitioning based on simulated annealing and tabu serach, Design Automation for Embedded Systems, vol.2, no.1, 1995
[10] R.Ernst, J.Henkel, Th.Benner, The Cosyma environment for Hardware/Software cosynthesis of small embedded systems, Technische Universitat Braunschweig
[11] D.D.Gajski, F.Vahid, S.Narayan, J.Gong, Specification and Design of embedded Systems, Prentice-Hall, New Jersey 1994, U.S.A., ISBN 0-13-150731-1
[12] J.Hou, W.Wolf, Process partitioning for distributed embedded systems, Proc. of the Int. Workshop on Hardware/Software Codesign, 1996
[13] www.kronopol.com.pl
[14] S.Prakash, A.Parker, SOS: Synthesis of application-specific heterogeneous multiprocessor systems, Journal of Parallel and Distributed Comp. vol.16, 1992

[15] A.Stasiak, M.Adamski, System Petri net specification , Programmable Devices and Embedded Systems - PDeS 2006 : proceedings of IFAC workshop. Brno, Czechy, 2006 .- Brno, 2006
[16] A.Stasiak, Metody synchronicznej wymiany danych w systemach osadzonych dla potrzeb dekompozycji systemowej, W: Materiały XVIII Sympozjum Koła Zainteresowań Cybernetycznych. Warszawa, Polska, 2002 .- Warszawa : BEL Studio Sp. z o.o., 2002, s. 98--107 .- ISBN: 83-88442-39-2
[17] A.Stasiak, System zrównoleglenia pracy mikrokontrolera sekwencyjnego dla sprzętowo-programowej architektury systemu osadzonego, Materiały KKE'2003, wyd. Politechnika Koszalińska, Kołobrzeg 2003, Polska, str.375-380, ISBN 83-7365-030-X
[18] A.Stasiak, Automatyczne dekompozycja specyfikacji behawioralnej sprzętowo-programowej mikrostruktury cyfrowej, Rozprawa Doktorska, Wydział Elektortechniki Informatyki i Telekomunikacji Uniwersytet Zielonogórski, Zielona Góra 2007, Polska
[19] A.Stasiak, M.Michalczak, Z.Skowroński, Algorytm dekompozycji systemowej dla potrzeb zintegrowanego projektowania, RUC'03, Pracowania Poligraficzna Politechniki Szczecińskiej, Szczecin 2003, ISBN 83-87362-56-5
[20] A.Stasiak, Z.Skowronski, An Universla Coprocessors for SOC Systems, 49 IWK, Technical University of Ilmenau, Ilmenau September 2004, Germany
[21] www.xilinx.com,www.xilinx.com/company/success/appsig.htm
[22] www.ec.zgora.pl

Title: A decomposition algorithm of functional specification of the hardware-software digital microsystem.