Konferencja naukowa KNWS'04
„Informatyka - sztuka czy rzemiosło"
23-25 czerwca 2004, Czocha

ŚRODOWISKO PROJEKTOWE DLA POTRZEB SYNTEZY SYSTEMÓW OSADZONYCH

Zbigniew Skowroński

Instytut Informatyki i Elektroniki, Uniwersytet Zielonogórski
65-246 Zielona Góra, ul. Podgórna 50

e-mail: Z.Skowronski@iie.uz.zgora.pl

STRESZCZENIE

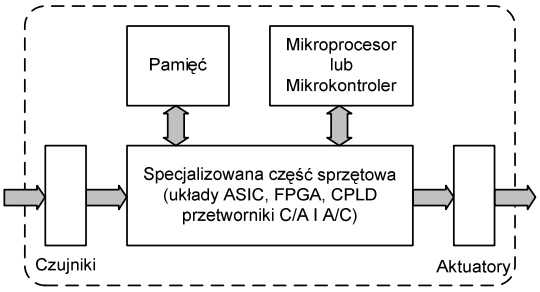
Praca omawia środowisko projektowe dla potrzeb syntezy systemów osadzonych. Dedykowane jest ono głównie dla zintegrowanego projektowania systemów sprzętowo-programowych, których specyfikacja funkcjonalna ma charakter niejednorodny. Z tego względu, przedstawione zostaną metody specyfikowania systemów sprzętowo-programowych oraz ich architektura. Elementami środowiska są translatory specyfikacji funkcjonalnej systemu (opis w językach VHDL i Verilog) do modelu formalnego (interpretowane sieci Petriego) oraz jego symulator czasowo-funkcjonalny.

1. WPROWADZENIE

Coraz częściej projektowanie systemów cyfrowych nie ogranicza się tylko do realizacji sprzętu. Większość złożonych urządzeń cyfrowych zbudowana jest zarówno z części sprzętowej, jak i oprogramowania pracującego na platformie sprzętowej. Przykładami takich urządzeń mogą być systemy osadzone (będące specjalizowanymi układami przetwarzania danych i sterowania), których szczególnym przypadkiem są systemy reaktywne (niejednokrotnie nazywane systemami czasu rzeczywistego). Synteza takich urządzeń stanowi najnowszą dziedzinę syntezy systemowej i nosi nazwę zintegrowanego projektowania lub kosyntezy (ang. Hardware/Software Co-Design). Uproszczoną architekturę zintegrowanych systemów przedstawia rysunek 1.

Ważnymi parametrami mającymi duży wpływ na proces zintegrowanej syntezy są koszt i szybkość działania projektowanych systemów. Parametry te zależą od podziału (dekompozycji) systemu na część sprzętową i część programową. Szybkość działania danego systemu zwiększa się dostosowując w odpowiedni sposób konstrukcję części sprzętowej do oprogramowania lub odwrotnie, ale ma to wpływ na koszt realizacji. Rozwiązania czysto sprzętowe mogą charakteryzować się większą szybkością działania, wynikającą z równoległego wykonywania operacji, ale wymagają zastosowania jednego lub kilku specjalizowanych układów cyfrowych. Z drugiej strony, rozwiązania programowe umożliwiają zastosowanie szybkich procesorów o niskim koszcie, wynikającym z ich

masowej produkcji. Niemniej jednak, sekwencyjne wykonywanie operacji lub brak realizacji niektórych operacji może zmniejszyć w znaczny sposób szybkość działania systemu.



Rys. 1. Uproszczona architektura systemów zintegrowanych

W chwili obecnej narzędzia CAD/CAE są w niewielkim stopniu dostosowane do potrzeb zintegrowanej syntezy. Ich ewentualny rozwój hamowany jest przez brak odpowiedniego modelu formalnego oraz spójnego języka opisu systemów sprzętowo-programowych. Po drugie szacowanie kosztu oraz szybkości działania systemów niejednorodnych odgrywa ważną rolę w podejmowaniu decyzji o podziale, jak również dotyczących syntezy.

W pracy proponuje się nowe środowisko projektowe dedykowane dla potrzeb zintegrowanego projektowania, a szerzej syntezy systemowej. W środowisku tym części sprzętowe systemu specyfikowane są w językach HDL, a część programowa w języku C. Jako pośredni model formalny zostały wybrane interpretowane sieci Petriego (IPN) [16, 20].

2. SPECYFIKACJA SYSTEMÓW SPRZĘTOWO-PROGRAMOWYCH

Specyfikacja złożonego systemu cyfrowego składającego się z części sprzętowej i programowej na poziomie systemowym jest formułowana zgodnie z jednym z dwóch schematów [1, 2]:

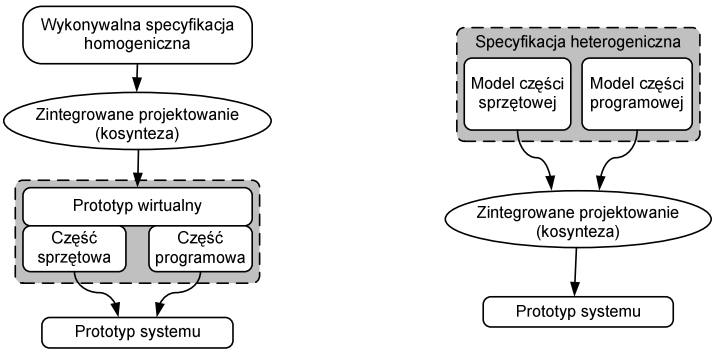
- homogeniczna - specyfikacja jest tworzona w jednym języku, obejmując zarówno część sprzętową, jak i programową (Rys. 2a);
- heterogeniczna - do specyfikacji sprzętu i oprogramowania wykorzystuje się różne języki; typowym przykładem jest połączenie C-VHDL (Rys. 2b).

Obie formy wymagają innej organizacji środowiska projektowego.

2.1. Specyfikacja homogeniczna systemu

Do specyfikacji całego systemu stosuje się w tym przypadku jeden język. Proces projektowania rozpoczyna się tu od specyfikacji globalnej, która może być niezależna od przyszłej implementacji oraz podziału (dekompozycji) na część sprzętową i programową. W konsekwencji proces projektowania musi obejmować etap dekompozycji, służący

rozdzieleniu jednolitej specyfikacji systemowej na części: sprzętową i programową. Efektem dekompozycji jest architektura składająca się z procesów sprzętowych i programowych, nazywana zwykle *prototypem wirtualnym*. Prototyp wirtualny jest przedstawiany w jednym lub wielu językach, np. C dla części programowej oraz VHDL dla części sprzętowej.



a) b)

Rys. 1. Specyfikacja systemu sprzętowo-programowego: homogeniczna (a), heterogeniczna (b)

Podstawową trudność w omawianym środowisku stanowi zachowanie wyraźnego odniesienia między koncepcjami zastosowanymi w specyfikacji początkowej oraz koncepcjami dostępnymi w docelowym modelu, reprezentowanym przez prototyp wirtualny. W praktyce oznacza to konieczność przetworzenia konstrukcji o wyższym, systemowym poziomie abstrakcji, jak np. sterowanie rozproszone czy nieinterpretowana (abstrakcyjna) komunikacja między procesami, na języki takich konstrukcji pozbawione. Zadanie takie jest niejednokrotnie bardzo skomplikowane. W celu zmniejszenia różnic między konstrukcjami i koncepcjami stosowanymi w specyfikacji oraz prototypie wirtualnym, wiele narzędzi projektowania zintegrowanego do specyfikacji systemowej stosuje języki przeznaczone w zasadzie do niższego poziomu, starając się jedynie rozszerzyć je o konstrukcje niezbędne do reprezentacji systemów. Przykładowo, w pakiecie Cosyma stosowane jest rozszerzenie języka C o nazwie C^X [3], pakiet Vulcan korzysta z innego rozszerzenia tego samego języka - Hardware C [4]. Pakiety LYCOS [5] i Castle wykorzystują język C. Inne pakiety z kolei używają języków opisu sprzętu, zwłaszcza języka VHDL. Stosowane są także języki specjalizowane, wyższego poziomu: w pakiecie Polis [6] jest to Esterel [7], w SpecSyn - SpecCharts [8, 9], zaś w systemie Cosmos - SDL [10].

2.2. Specyfikacja heterogeniczna systemu

W specyfikacji heterogenicznej część sprzętowa i programowa opisywane są z wykorzystaniem specyficznych, dostosowanych do tego celu języków. W przypadku takiego podejścia cały proces projektowy rozpoczyna się od prototypu wirtualnego (Rys. 2b), w którym podziału na część sprzętową i programową dokonał już projektant. Projektowanie

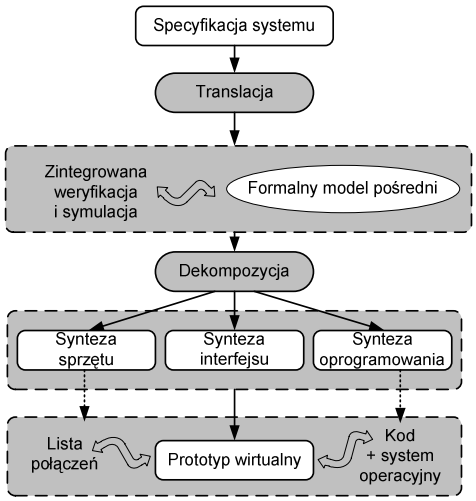
zintegrowane sprowadza się tu zatem do odwzorowania poszczególnych elementów specyfikacji na właściwe procesory [2].

Podstawowymi kwestiami w projektowaniu zintegrowanym na bazie specyfikacji heterogenicznej są walidacja (uwiarygodnienie) i interfejsy. Zastosowanie specyfikacji wielojęzycznej wymaga nowych technik walidacji zdolnych do obsługi takich specyfikacji. Zamiast symulacji potrzebna jest zintegrowana symulacja - współsymulacja. Podobnie w miejsce weryfikacji potrzebna jest zintegrowana weryfikacja - współweryfikacja. Ponadto, niezbędne jest określenie interfejsów między elementami opisanymi w różnych językach. Interfejsy te powinny być dodatkowo sprawdzone pod kątem zgodności między specyfikacją wejściową, a zbudowanym prototypem.

Przykładem środowiska projektowego skonstruowanego na bazie specyfikacji heterogenicznej jest pakiet zaproponowany w pracy.

3. KONCEPCJA PROPONOWANEGO ŚRODOWISKA PROJEKTOWEGO

Opis zachowania części sprzętowej docelowego systemu specyfikowany jest w języku opisu sprzętu HDL (Verilog, VHDL), a części programowej w języku C. W pierwszej fazie procesu projektowego dokonywana jest translacja wejściowej specyfikacji na pośredni model formalny [15] i [17]. W celu uzyskania opisu z maksymalnym stopniem równoległości procesów w komponentach składowych systemu jako model wybrano interpretowane sieci Petriego (dokładnie omówione w [16, 17]). Ważnym elemenmtem środowiska jest symulator czasowo-funkcjonalny modelu formalnego [17], którego zadaniem jest dokonanie oceny projektowanego systemu pod kątem funkcjonalności i czasu realizacji oraz moduł dekompozycji [18, 19].



Rys. 3. Metodologia projektowania

3.1. Metodologia projektowania

W zintegrowanym podejściu do problemu projektowania systemów mikroprocesorowych obie części systemu (programowa i sprzętowa) specyfikowane są jako całość. Jako całość są także traktowane możliwie długo w procesie projektowania i całościowo są również analizowane. Wymusza to opracowanie odpowiedniej metodologii projektowania (rys. 3) takich systemów oraz ich docelowej architektury [11, 12, 13, 17, 20].

Prezentowane rozwiązanie charakteryzuje się łatwością specyfikacji i weryfikacji systemu, elastycznością w rozwiązywaniu przeciwstawnych kryteriów projektowych, szybszymi rozwiązaniami niż czysto programowe oraz niższym kosztem niż w przypadku rozwiązań czysto sprzętowych. Układy programowane dużej złożoności wciąż należą do najdroższych elementów elektronicznych, znacznie droższych od masowo produkowanych mikrokomputerów jednoukładowych. Jednak przy odpowiednim podejściu umożliwiają realizację szybkiego prototypowania projektowanych układów (ang. Rapid Prototyping).

3.2. Prototyp systemu

Do przedstawienia metodologii zintegrowanego projektowania systemów sprzętowo-programowych przyjmuje się architekturę systemu proponowaną w pracach [11, 17, 20]. System taki składa się z konkretnych aplikacji sprzętowych, połączonych magistralą systemową z uniwersalną jednostką centralną lub wbudowanym systemem komputerowym, uruchamianym pod odpowiednim systemem operacyjnym.

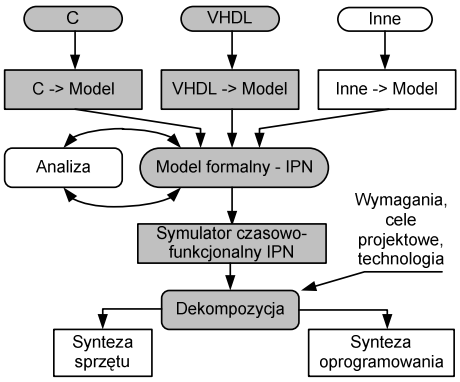
Konkretna aplikacja sprzętu projektowana jest tak, aby mogła współpracować z aplikacjami oprogramowania, uruchamianymi na jednostce centralnej (ang. CPU). Dla konkretnej aplikacji sprzętu, interfejs magistrali pracuje z programem obsługi urządzenia, umożliwiającym odczyt i zapis danych do i z urządzenia, przy transmisji z potwierdzeniem. Przyjmuje się, że jednostka centralna CPU uruchamiana jest w systemie operacyjnym, zdolnym do komunikowania się z urządzeniami współpracującymi z układami We/Wy sterowanymi przerwami. System może zawierać pamięć i inne urządzenia We/Wy.

3.3. Elementy środowiska projektowego

Zintegrowane projektowanie (kosynteza) składa się z następujących elementów [12, 13, 17, 20]:

- wyboru odpowiedniej specyfikacji wejściowej docelowego systemu bądź jego fragmentu, którymi są języki programowania (C, C++) dla komponentów programowych i języki opisu sprzętu (VHDL, Verilog) dla komponentów sprzętowych;
- wyboru odpowiedniego modelu formalnego - interpretowane sieci Petriego;
- analizy specyfikacji, np. pod kątem czasu realizacji i parametrów technicznych;
- dekompozycji systemu - kluczowego składnika zintegrowanego projektowania.

Pierwszym etapem proponowanego procesu projektowania (rys. 4) jest przekształcenie specyfikacji wejściowej w formalny model pośredni, umożliwiający dalszą obróbkę danych. Następnie, model formalny podawany jest analizie pod kątem parametrów technicznych, czasu realizacji i weryfikacji formalnej w symulatorze modelu pośredniego. Po uzyskaniu informacji o czasie realizacji wybranych fragmentów specyfikacji (bądź całości) oraz liczbie bloków funkcjonalnych potrzebnych do jej zrealizowania następuje dekompozycja. Kolejnym etapem jest jest kompilacja uzyskanych specyfikacji i implementacja sprzętu oraz oprogramowania.



Rys. 4. Proponowane środowisko projektowe

Moduł dekompozycji jest najistotniejszą częścią składową zintegrowanego projektowania systemów sprzętowo-programowych. Dekompozycja na podstawie specyfikacji wejściowej oraz zadanych wymagań (czasowych, finansowych, użytkowych i innych) dzieli specyfikację systemu na moduły realizowane w postaci programu dla mikroprocesora oraz mapy programowej dla układów programowanych, współpracujących z procesorem. Podział specyfikacji jest tak realizowany, aby zachowanie docelowego systemu było zgodne z zakładanymi celami projektowymi.

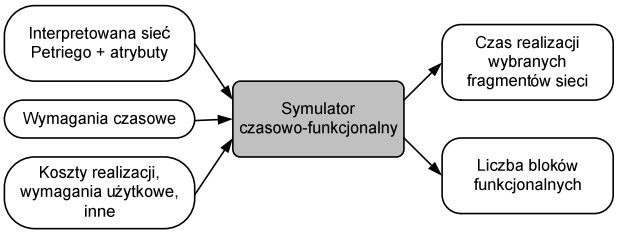
Aktualnie znane są dwa podstawowe style dekompozycji. Pierwszy z nich polega na specyfikacji całego systemu jako sprzętu. Następnie, na podstawie określonych ograniczeń dokonywana jest ocena systemu oraz przenoszenie operacji do programu. Przykładem takiego podejścia jest system Vulcan [4]. W drugim stylu, system specyfikowany jest jako program i po dokonaniu oceny odpowiednie operacje przenoszone są do sprzętu. Przykładem może być system COSYMA [3].

3.4. Symulator czasowo-funkcjonalny interpretowanych sieci Petriego

Systemy specyfikowane w postaci interpretowanych sieci Petriego, gdzie poszczególnym operacjom wykonywanym przez układ odpowiadają miejsca sieci, wzbogacone o dodatkowe atrybuty (typ bloku funkcjonalnego, realizujący daną operację, czas realizacji tej operacji,

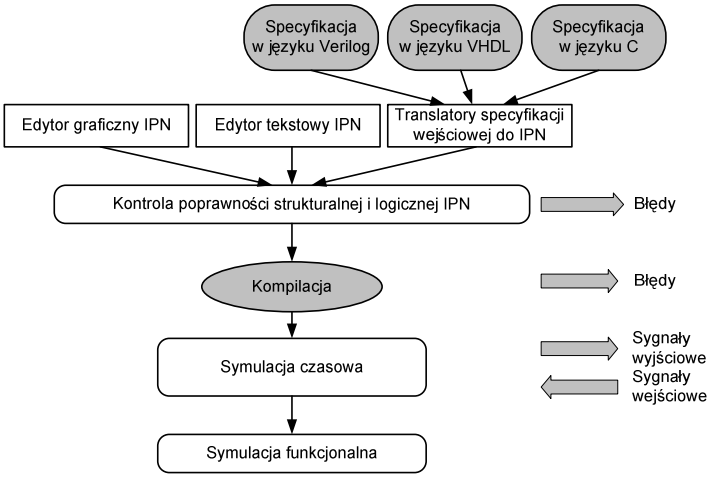
itp.) poddaje się analizie w symulatorze czasowo-funkcjonalnym (rys. 5, 6 i 7) [14, 17]. Jego zadaniem jest dostarczenie informacji, na podstawie których możliwe jest przetworzenie tej specyfikacji tak, aby uzyskana na jej podstawie implementacja była bliższa optimum.

Jedną z głównych zalet symulatora jest jego szeroko pojęta elastyczność i uniwersalność. W związku z faktem, iż istnieje bardzo wiele języków umożliwiających specyfikację systemu sprzętowo-programowego, stworzenie narzędzia umożliwiającego symulację każdego z nich osobno byłoby nie tylko niepraktyczne, ale wręcz niemożliwe. Z tego względu, zaproponowano zastosowanie modelu pośredniego, wykorzystywanego jako wejściowa specyfikacja symulatora. Pozostałe specyfikacje, przekształcane są do modelu pośredniego za pomocą translatorów stanowiących integralną częścią symulatora.



Rys. 5. Symulator czasowo-funkcjonalny - zadania

Symulator zawiera interfejs, umożliwiający dołączenie do niego kolejnych translatorów. Oznacza to, że za pomocą tak zaprojektowanego środowiska istnieje możliwość symulacji systemów, opisanych w dowolnie podanej specyfikacji. Przedstawiana wersja programu umożliwia translację specyfikacji podanych w językach VHDL i Verilog oraz języku C.

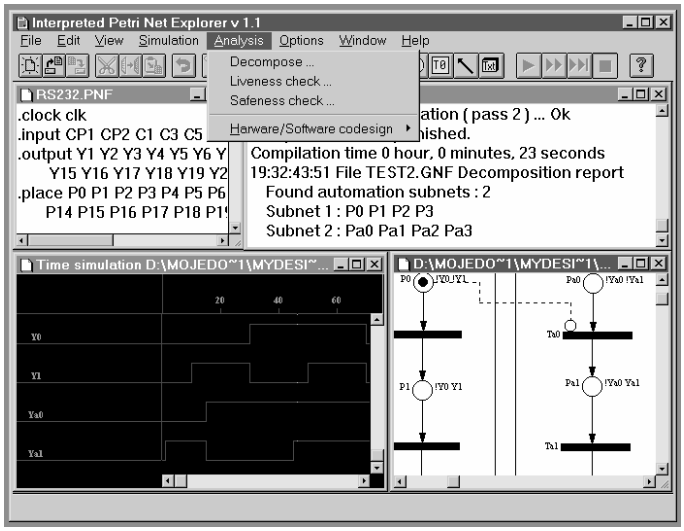


Rys. 6. Schemat przepływu danych w aplikacji

Symulator czasowo-funkcjonalny integruje w sobie siedem odrębnych logicznie i funkcjonalnie modułów [14, 17]:

- graficzny edytor interpretowanych sieci Petriego (IPN);
- tekstowy edytor IPN w formacie PNSF i bloków funkcjonalnych FBF (ang. Functional Block File);
- translator IPN z języka C;
- translator IPN z języka VHDL;
- translator IPN z języka Verilog;
- symulator czasowy IPN;
- symulator funkcjonalny IPN.

Schemat przepływu danych w aplikacji (ang. Application Data Flow Diagram) przedstawiono na rysunku 6, natomiast przykładowe okno z widocznym przebiegiem czasowym na rysunku 7.



Rys. 7. Symulator czasowo-funkcjonalny - widok aplikacji

Badany system jest specyfikowany graficznie lub tekstowo. Specyfikacja może być również wynikiem działania jednego z translatorów. Następnie, sprawdzana jest jej poprawność strukturalna i logiczna. Jeżeli specyfikacja jest prawidłowa, to rozpoczyna się proces kompilacji, którego zadaniem jest transformacja opisu sieci z formatu tekstowego na postać zrozumiałą dla algorytmów symulacyjnych. Tak przetworzona sieć poddawana jest symulacji czasowej, a następnie funkcjonalnej.

4. KIERUNKI ROZSZERZEŃ ŚRODOWISKA PROJEKTOWEGO

Zintegrowane projektowanie systemów mikroprocesorowych (sprzętowo-programowych), a przede wszystkim ich dekompozycja, pociąga za sobą scharakteryzowanie oczekiwań jakie

mają spełniać sprzęt i oprogramowanie, identyfikację części sprzętowej i programowej, przejście z opisu funkcjonalnego w odpowiednią część oraz syntezę sprzętowych i programowych rezultatów. Zakres możliwych struktur systemu i celów projektowych jest szeroki. W związku z tym, zintegrowane projektowanie systemów mikroprocesorowych przyjmuje różnorodne formy. W rozdziale tym zaprezentowano jedynie te elementy środowiska projektowego, które wchodzą w zakres pracy. Niektóre elementy proponowanego środowiska, np. moduł dekompozycji [12, 13, 17, 18, 19] są ciągle w trakcie badań i wymagają dodatkowych ulepszeń.

Praca naukowa finansowana ze środków Komitetu Badań Naukowych w latach 2003-2006 jako projekt badawczy nr 4 T11C 006 24.

LITERATURA

[1] V. M. T. Sakamoto, G. De Micheli: *Run-time scheduler synthesis for hardware-software systems and application to robot control design*, in Proceedings of the CHDL'97, pp. 95-99, March 1997

[2] A. A. Jerraya, M. Romdhani, C. A. Valderrama, P. Le Marrec, F. Hessel, G. F. Marchioro, J. M. Daveau: *Languages for System-Level Specification and Design*, In: [21], pp. 235-262

[3] A. Österling, Th. Benner, R. Ernst, D. Herrmann, Th. Scholz, Wei Ye: *The Cosyma System*, In: [21], pp. 263-281

[4] D. Ku, G. De Micheli: *Hardware C - a language for hardware design*, Tech. Rep. CSL-TR-88-362, Computer Systems Laboratory, Stanford University, August 1988

[5] J. Madsen, J. Grode, P. V. Knudsen: *Hardware/Software Partitioning using the LYCOS System*, In: [21], pp. 283-305

[6] 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

[7] G. Berry: *Hardware implementation of pure Esterel*, Proceedings of the ACM Workshop on Formal Methods in VLSI Design, January 1991

[8] F. Vahid, S. Narayan: *SpecCharts: A language for system-level synthesis*, Proceedings of CHDL, April 1991, pp. 145-154

[9] F. Vahid, S. Narayan, D. Gajski: *SpecCharts: A VHDL front-end for embedded systems*, IEEE Transactions on CAD of Integrated Circuits and Systems, vol. 14, 1995, pp. 694-706

[10] C. A. Valderrama, M. Romdhani, J.M. Daveau, G.F. Marchioro, G. Marchioro, A. Changuel, A. A. Jerraya: *COSMOS: A Transformational Co-Design Tool for Multiprocessor Architectures*; In: [21], pp. 307-357

[11] D. E. Thomas, J. K. Adams, H. Schmit: *A Model and Methodology for Hardware-Software Co-design*, IEEE Design & Test of Computers, vol. 10, No 3, September 1993, pp. 6-15

[12] Z. Skowroński: *Dekompozycja systemów mikroprocesorowych współpracujących ze specjalizowanymi układami cyfrowymi na część sprzętową i programową,* Materiały I Krajowej Konferencji Naukowej - Reprogramowalne Układy Cyfrowe, Szczecin, 12-13 Marzec 1998, str. 75-82, ISBN 83-87362-07-7

[13] Z. Skowroński, J. Mirkowski: *Zintegrowane projektowanie systemów mikroprocesorowych ze specjalizowanymi układami cyfrowymi*, II Konferencja Informatyka na wyższych uczelniach dla gospodarki narodowej, Gdańsk, 22-24 listopada 1996 r, tom I, str. 123-126

[14] T. Dzikuć, Z. Skowroński: *Symulator czasowo - funkcjonalny do specyfikacji systemów cyfrowych zadanych w postaci interpretowanych sieci Petriego*, 20 MSN, Politechnika Zielonogórska, Zielona Góra, 11-12 Maja 1998, tom III - Informatyka, str. 295-301, ISBN 83-85911-60-X

[15] Z. Skowroński: *Problem translacji specyfikacji funkcjonalnej układów cyfrowych w języku VHDL na interpretowaną sieć Petriego w zintegrowanej syntezie systemów sprzętowo-programowych*, Reprogramowalne Układy Cyfrowe - RUC 2001, Materiały IV Krajowej Konferencji Naukowej, Szczecin, Polska, 2001, str. 103-113

[16] Z. Skowroński: *Interpretowane sieci Petriego jako formalny model pośredni w syntezie systemowej*, Reprogramowalne Układy Cyfrowe - RUC 2000, Materiały III Krajowej Konferencji Naukowej, Szczecin, Polska, 2000, str. 155-164

[17] Z. Skowroński: *Translacja specyfikacji funkcjonalnej układów cyfrowych na sieć Petriego dla potrzeb syntezy systemowej*, Rozprawa doktorska, Promotor: dr hab. inż. Marian Adamski, Politechnika Szczecińska, Wydział Informatyki, 2000, 156 str.

[18] A. Stasiak, M. Michalczak, Z. Skowroński: *Algorytm dekompozycji systemowej dla potrzeb projektowania zintegrowanego*, Reprogramowalne Układy Cyfrowe - RUC 2003, Materiały VI Krajowej Konferencji Naukowej, Szczecin, Polska, 2003, str. 113-122, ISBN 83-87362-56-5

[19] Z. Skowroński, A. Stasiak: *Automatyczna dekompozycja systemów osadzonych*, Reprogramowalne Układy Cyfrowe - RUC 2003, Materiały VI Krajowej Konferencji Naukowej, Szczecin, Polska, 2003, str. 101-112, ISBN: 83-87362-56-5

[20] M. Adamski, Z. Skowroński: *Interpretowane sieci Petriego - model formalny w zintegrowanym projektowaniu mikroprocesorowych systemów sprzętowo-programowych*, Pomiary Automatyka Kontrola, 2003, nr 2-3, str. 17-20

[21] Edited by J. Staunstrup, W. Wolf*: Hardware/Software Co-Design Principles and Practice*, Kluwer Academic Publishers, 1997, ISBN 0-7923-8013-4