Marian ADAMSKI, Zbigniew SKOWROŃSKI
UNIWERSYTET ZIELONOGÓRSKI
INSTYTUT INFORMATYKI I ELEKTRONIKI

Interpretowane sieci Petriego - model formalny w zintegrowanym projektowaniu mikroprocesorowych systemów sprzętowo-programowych

Prof. dr hab. inż. Marian ADAMSKI

Dyrektor Instytutu Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Zainteresowania badawcze obejmują projektowanie systemów cyfrowych realizowanych w postaci mikrosystemów cyfrowych oraz formalnych metod programowania sterowników logicznych PLC. Członek Polskiego Towarzystwa Elektrotechniki Teoretycznej i Stosowanej oraz Polskiego Towarzystwa Informatycznego.

e-mail: *M.Adamski@iie.uz.zgora.pl*

Dr inż. Zbigniew SKOWROŃSKI

Adiunkt w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Pełni funkcję zastępcy dyrektora Instytutu. Zainteresowania badawcze obejmują zagadnienia zintegrowanego projektowania cyfrowych systemów sprzętowo-programowych, ze szczególnym uwzględnieniem języków opisu sprzętu (VHDL i Verilog) oraz systemów osadzonych, zawierających reprogramowalne układy logiczne.

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

Streszczenie

Podejście systemowe do projektowania urządzeń o niejednorodnym charakterze wymaga stosowania formalnych metod specyfikacji, syntezy i analizy. Metody i narzędzia projektowe z kolei bazują na formalnym modelu obliczeniowym. Z tego względu dobór właściwego modelu ma fundamentalne znaczenie dla efektywności całego procesu projektowania. W pracy zaproponowano środowisko projektowe dla potrzeb zintegrowanego projektowania, w którym części sprzętowe systemu specyfikowane są w języku VHDL, część programowa w języku C, a modelem formalnym są interpretowane sieci Petriego. Prezentowane wyniki prac znajdują również zastosowanie w projektowaniu mikrosystemów cyfrowych, zawierających układy FPGA.

Abstract

In order to model heterogeneous systems some common representation vehicle is needed. The model should have several features, the most important of which are: to be well suited both for software and hardware representation, allow for different manipulations (including partitioning) and be able to cope explicitly with parallelism. Interpreted Petri nets can meet all three requirements. The paper analyses the suitability of the Petri nets for a representation of heterogeneous systems and outlines some practical aspects of the application of Petri nets in modelling such systems.

1. Wprowadzenie

W klasycznym podejściu do projektowania mikroprocesorowych systemów sprzętowo-programowych (systemów osadzonych - ang. Embedded Systems) podział zadań między część programową (realizowaną przez mikroprocesor) a sprzętową (realizowaną przez elementy specjalizowane) następuje bardzo wcześnie. W zasadzie jedynie w fazie definiowania opisu systemu jest on traktowany jako jedna całość. Wkrótce potem specyfikacja dzielona jest między sprzęt a oprogramowanie, które odtąd projektowane są w zasadzie niezależnie od siebie. Dopiero w fazie wykonywania i testowania prototypu obie te części ponownie łączone są w jedną całość.

Jakkolwiek taki sposób projektowania stosowany jest skutecznie w praktyce, to stosunkowo łatwo można wymienić jego słabe punkty:

- Dobry (tzn. korzystny z punktu widzenia oczekiwanych parametrów - ceny, wielkości, szybkości działania itd.) podział wymaga bardzo dużego doświadczenia inżynierskiego, którego na tak wczesnym etapie procesu projektowania nie można wesprzeć żadnymi wynikami analizy czy szacowania parametrów układu.
- Nawet bardzo duże doświadczenie eksperckie nie gwarantuje optymalności rozwiązania. W efekcie po zrealizowaniu poszczególnych części systemu może się okazać, że działa on zbyt wolno, bądź kosztuje zbyt wiele wskutek zastosowania nadmiernie dużych układów specjalizowanych w stosunku do niezbędnego minimum. Zaprojektowany w ten sposób układ może zatem odbiegać od optimum spełnienia oczekiwań zleceniodawcy projektu.

Obydwa problemy w znacznej mierze rozwiązuje podejście zintegrowane do procesu projektowania mikroprocesorowych systemów sprzętowo-programowych [2, 5, 12].

2. Zintegrowane projektowanie mikroprocesorowych systemów sprzętowo-programowych

Podstawową ideą jest tu traktowanie projektowanego systemu - możliwie jak najdłużej - jako jednej całości [12]. W ten sposób w procesie projektowania pojawia się nowy etap: *specyfikacja formalna* systemu. Dysponując taką specyfikacją projektant może przeprowadzić analizę całego układu pod kątem możliwości i konsekwencji implementacji poszczególnych zadań w części sprzętowej lub programowej.

Przypisanie zadań do części je realizujących (dekompozycja, ang. Partitioning) odbywa się o wiele później niż w metodzie klasycznej. Dzięki temu podczas dekomponowania układu projektant dysponuje większą ilością informacji ułatwiających optymalną realizację pod kątem przyjętych kryteriów.

Stąd zintegrowane projektowanie systemów sprzętowo-programowych (ang. Hardware/Software Co-Design of Embedded Systems) wymaga odpowiednio zdefiniowanego modelu formalnego reprezentującego projektowane urządzenie. Model taki musi odznaczać się m.in. zdolnością do reprezentowania: czasu, współbieżności zdarzeń, asynchronicznej komunikacji między procesami, zarówno dla części sterującej i operacyjnej [14, 15].

Spośród istniejących modeli abstrakcyjnych, jakie mogą być brane pod uwagę jako modele formalne dla potrzeb syntezy systemowej, żaden nie spełnia wszystkich tych wymagań w stopniu zadowalającym i nadal trwają poszukiwania nowych, lepszych rozwiązań. Jednym z nich mogą być sieci Petriego, które w naturalny sposób spełniają kilka wymagań stawianych przed takim modelem.

Do najczęściej obecnie stosowanych modeli formalnych należą hierarchiczne automaty równoległe HCFSM [3] (ang. Hierarchical Concurrent Finite State Machine) oraz grafy przepływu (DFG - ang. Data Flow Graph, CDFG - ang. Control-Data Flow Graph) [5]. W pierwszym przypadku system musi być przedstawiony w postaci współpracujących automatów sekwencyjnych, co ogranicza uniwersalność zastosowania takiego modelu. W przypadku grafów przepływu zasadniczą wadą jest naturalne ukierunkowanie na przetwarzanie danych, związane z niedostatkiem efektywnych mechanizmów reprezentacji sterowania.

Obydwu wad nie mają sieci Petriego. W praktyce przeprowadzano już wiele prób ich wykorzystania w kontekście zintegrowanego projektowania systemów sprzętowo-programowych. Przykładami takich rozwiązań mogą być systemy opracowane przez Stoya [4], Kleinjohanna [2] czy Machado [9] oraz prace [7, 8, 14, 15]. Dokładny przegląd dotychczas opublikowanych rozwiązań przeprowadzony został w pracy [6].

4. Cechy dobrego modelu formalnego

Efektywność narzędzi CAD/CAE (ang. Computer Aided Design / Computer Aided Engineering) w oczywisty sposób zależy od zastosowanych algorytmów oraz modeli, na jakich te algorytmy operują. Dobry model powinien charakteryzować się kilkoma cechami [14]: jednoznacznością, reprezentacją współbieżności, semantyczną spójnością z językami specyfikacji i implementacji, łatwością translacji, aparatem matematycznym, dostępnością efektywnych algorytmów oraz niezależnością od specyfikacji i implementacji.

W związku z tym *dobry model* musi reprezentować problem projektowy najdokładniej, jak to tylko możliwe. Jest to szczególnie istotne w przypadku opisu współbieżności, komunikacji, synchronizacji, specyfikacji przepływu danych oraz modelowania czasu. Powinien być także odpowiedni dla zastosowanych algorytmów w procesie projektowym. Wymaganie to powinno obejmować zarówno złożoność obliczeniową, jak i formalne podstawy teoretyczne, umożliwiające jednoznaczne odwzorowanie operacji projektowanego urządzenia w model.

Model formalny powinien również charakteryzować się: przejrzystością, dostępnością narzędzi i standardem. Czytelność specyfikacji odgrywa istotną rolę w efektywności jej wykorzystania.

3. Interpretowane sieci Petriego

Modelem formalnym spełniającym wiele cech dobrego modelu, mogą być sieci Petriego. Wprowadzają one bezpośrednio i graficznie najważniejsze paradygmaty przetwarzania równoległego [14]: sekwencyjność/zależność, konflikt i równoległość.

Sieć Petriego [10] jest dwudzielnym grafem skierowanym o dwóch rodzajach wierzchołków: *miejscach* oraz *tranzycjach*. Miejsca i tranzycje połączone są skierowanymi łukami. Znakowaniem nazywa się przypisanie *znaczników* do miejsc. Rozmieszczenie i liczba znaczników w sieci zmienia się w czasie podczas jej realizacji, odbywającej się na podstawie prostych zasad przygotowywania i odpalania tranzycji [10].

W pracy jako model formalny wykorzystywano sieci Petriego wysokiego poziomu, zwane interpretowanymi sieciami Petriego. Są one rozszerzeniem podstawowej definicji, polegającym na tym, że do każdej tranzycji można przypisać predykat zmieniający zasady odpalania tranzycji [7, 14]. Tranzycja z predykatem (funkcją logiczną o wartości wyznaczanej podczas realizacji sieci) może zostać odpalona, kiedy jest przygotowana i przypisany do niej predykat ma wartość logiczną *1*. Predykat może także odpowiadać wybranemu znakowaniu sieci. Jest to przydatne zwłaszcza w przypadku stosowania tak zwanych łuków zezwalających (ang. Enabling Arcs), które w przeciwieństwie do zwykłych łuków nie powodują usunięcia znakowania z miejsca po odpaleniu jego tranzycji wyjściowej.

Analogicznie do łuków zezwalających można wprowadzić także łuki zabraniające (ang. Inhibitor Arcs). Jeżeli miejsce wejściowe tranzycji jest z nią połączone łukiem zabraniającym, to wówczas tranzycja będzie gotowa do odpalenia, gdy miejsce to *nie* jest oznakowane i oznakowane są wszystkie miejsca wejściowe tej tranzycji związane z nią łukami zwykłymi i zezwalającymi.

Dodatkowo, zastosowana sieć Petriego musi być [14]: żywa (ang. Live), bezpieczna (ang. Safe), deterministyczna i wolna od konfliktów (ang. Conflict Free).

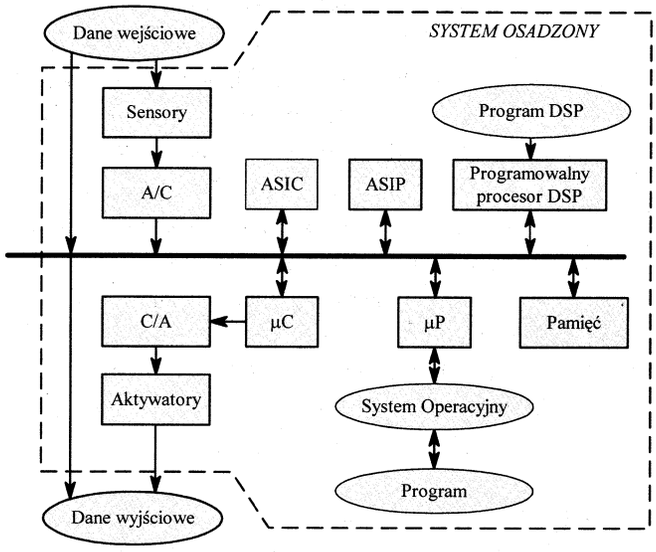
5. Środowisko projektowe dla potrzeb zintegrowanego projektowania

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. Architekturę zintegrowanych systemów mikroprocesorowych przedstawia rysunek 1.

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.

Stąd też w pracy proponuje się nowe środowisko projektowe dla potrzeb zintegrowanego projektowania, a szerzej syntezy systemowej. W środowisku tym części sprzętowe systemu specyfikowane są w języku VHDL, a część progra-

mowa w języku C. Jako model formalny zostały wybrane interpretowane sieci Petriego.



Rys. 1. Architektura systemów zintegrowanych

5.1. Wejściowa specyfikacja systemów sprzętowo-programowych

Specyfikacja złożonego systemu cyfrowego na poziomie systemowym może być formułowana jako homogeniczna lub heterogeniczna [1, 11].

Do specyfikacji homogenicznej systemu stosuje się jeden język. Proces projektowania rozpoczyna się tu od specyfikacji globalnej, która może być niezależna od przyszłej implementacji i dekompozycji na część sprzętową i programową. W konsekwencji proces projektowania musi obejmować etap dekompozycji, którego efektem jest architektura składająca się z procesów sprzętowych i programowych, nazywana zwykle *prototypem wirtualnym*. Prototyp wirtualny może być podany w jednym lub wielu językach, np. C dla części programowej oraz VHDL dla części sprzętowej.

W specyfikacji heterogenicznej część sprzętowa i programowa opisywane są przy użyciu specyficznych, dostosowanych do ich potrzeb języków. W przypadku takiego podejścia cały proces projektowy rozpoczyna się od prototypu wirtualnego, w którym podziału na część sprzętową i programową dokonał już projektant. Projektowanie zintegrowane sprowadza się w tym przypadku do odwzorowania poszczególnych elementów specyfikacji na właściwe procesy [1]. Podstawowymi kwestiami w projektowaniu zintegrowanym na bazie specyfikacji heterogenicznej są walidacja 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.

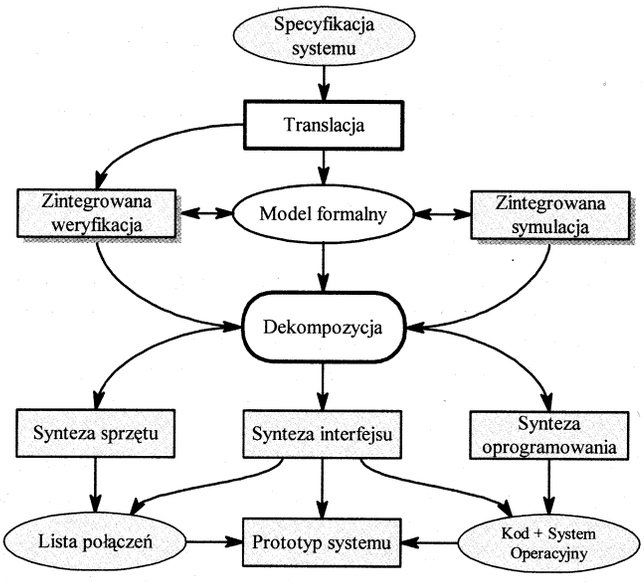
Przykładem środowiska projektowego skonstruowanego na bazie specyfikacji heterogenicznej jest pakiet zaproponowany w pracy.

5.2. Koncepcja proponowanego środowiska projektowego

Zintegrowane podejście do problemu projektowania systemów sprzętowo-programowych wymusza opracowanie odpowiedniej metodologii projektowania (rysunek 2) takich systemów oraz ich docelowej architektury [4, 13, 14, 15].

Architektura systemu proponowana w pracy zakłada, że system sprzętowo-programowy 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 po-twierdzeniem. Dodatkowo 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.



Rys. 2. Metodologia projektowania

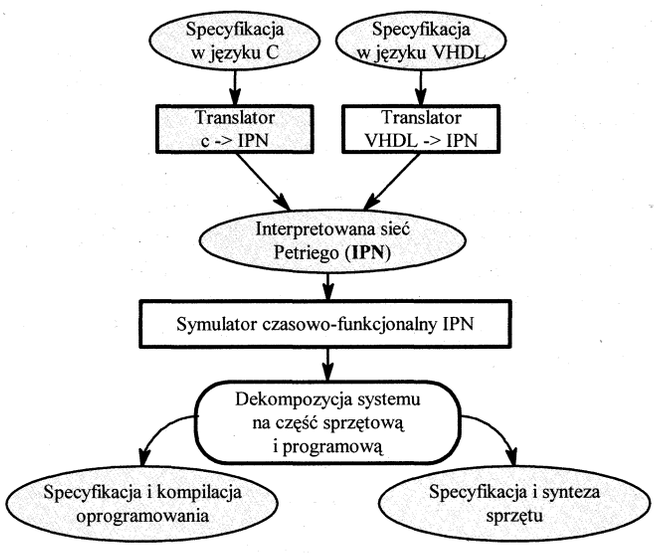
5.3. Elementy środowiska projektowego

Zintegrowane projektowanie składa się z następujących elementów [12, 13, 14]:

- translacji odpowiedniej specyfikacji wejściowej docelowego systemu bądź jego fragmentu do określonego modelu formalnego;
- analizy specyfikacji;
- dekompozycji systemu.

Pierwszym etapem proponowanego procesu projektowania (rysunek 3) jest przekształcenie specyfikacji wejściowej w model formalny, umożliwiający dalszą obróbkę danych. W kolejnym kroku model formalny podawany jest analizie pod kątem czasu realizacji, parametrów technicznych i weryfikacji formalnej w symulatorze modelu formalnego. Po uzyskaniu informacji o czasie realizacji wybranych fragmentów specyfikacji (bądź całości) oraz liczbie bloków funkcjonalnych potrzebnych do jej zrealizowania następuje dekompozycja. Następnie dokonywana jest kompilacja uzyskanych specyfikacji i zaimplementowanie sprzętu oraz oprogramowania.

Moduł dekompozycji stanowiący najistotniejszą częścią składową zintegrowanego projektowania systemów sprzętowo-programowych oraz symulator czasowo-funkcjonalny interpretowanych sieci Petriego wraz translatorem specyfikacji zadanej w języku VHDL na interpretowaną sieć Petriego zostały szeroko przedstawione w pracy [14].



Rys. 3. Proponowane środowisko projektowe

6. Podsumowanie

Proponowane w pracy środowisko projektowe mikroprocesorowych systemów sprzętowo-programowych jest ciągle w fazie rozwoju i opracowań. Wykonane dotychczas prace badawcze świadczą o tym, że charakteryzuje się łatwością specyfikacji (języki opisu sprzętu HDL i języki programowania) i weryfikacji systemu (aparat matematyczny sieci Petriego). Dodatkowo umożliwia elastyczność w rozwiązywaniu przeciwstawnych kryteriów projektowych oraz dostarcza rozwiązania szybsze niż czysto programowe i tańsze niż w czysto sprzętowe. Istnieje również możliwość realizacji szybkiego prototypowania układów (ang. Rapid Prototyping).

Literatura

[1] 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: [5], pp. 235-262

[2] B. Kleinjohann, J. Tacken, C. Tahedl: *Towards a Complete Design Method for Embedded Systems Using Predicate/Transition-Nets*; Proc. of the International Conference on Computer Hardware Design Languages CHDL'97, Toledo, Spain, March 1997

[3] D. Drusinsky, D. Harel: *Using Statecharts for hardware description and synthesis*, IEEE Transactions on Computer Aided Design, 1989

[4] E. Stoy: *A Petri Net Based Unified Representation for Hardware/Software Co-Design*, Licentiate Thesis No. LiU-Tek-Lic 1995: 21, Linköping University, Linköping, Sweden, 1995

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

[6] J. Mirkowski, A. Yakovlev: *A Petri Net Model for Embedded Systems*; Proc. of the 2nd International Conf. Design & Diagnostics of Electronic Circuits and Systems DDECS'98, Szczyrk, Poland, 2-4 Sept. 1998, pp. 313-321, ISBN 83-908409-6-0

[7] J. Mirkowski, Z. Skowroński: *Interpreted Petri Nets as a Formal Representation of Hardware/Software Systems*, Proceedings of 4th International Workshop *Mixed Design of Integrated Circuit and Systems*, Poznań, 12-14 June 1997, pp. 173-178, ISBN 83-87202-40-1

[8] J. Mirkowski, Z. Skowroński: *Translation of C and VHDL specifications into interpreted Petri Nets for Hardware/Software Codesign.* Mixed Design of Integrated Circuits and Systems, pp. 163-168, Kluwer Academic Publishers, 1998, ISBN 0-7923-8116-7

[9] R. J. Machado, J. M. Fernandes, A. J. Proença: *Specification of Industrial Digital Controllers with Object-Oriented Petri Nets*; Proc. IEEE International Symposium on Industrial Electronics ISIE'97, Guimarães, Portugal, July 1997

[10] T. Murata: *Petri Nets: Properties, Analysis and Applications*, Proceedings of the IEEE, vol. 77, No 4, April 1989, pp. 548-580

[11] 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

[12] 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

[13] 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

[14] Z. Skowroński: *Translacja specyfikacji funkcjonalnej układów cyfrowych na sieć Petriego dla potrzeb syntezy systemowej*, Praca doktorska, Politechnika Szczecińska, Wydział Informatyki, 2000

[15] G. Andrzejewski: *Programowy model interpretowanej sieci Petriego dla potrzeb projektowania mikrosystemów cyfrowych*, Praca doktorska, Politechnika Szczecińska, Wydział Informatyki, 2002

Artykuł recenzowany.