Andrzej STASIAK, Zbigniew SKOWROŃSKI
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI

Model formalny sprzętowo-programowych systemów cyfrowych

Mgr 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

Dr inż. Zbigniew SKOWROŃSKI

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

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

Streszczenie

Projektowanie sprzętowo-programowych zintegrowanych systemów cyfrowych jest jedną z najnowszych i wciąż rozwijanych technologii projektowania systemów osadzonych. Innowacja polega na zmianie punktu decyzyjnego w procesie projektowym, tj. punktu podziału systemu na dwie części: sprzęt i program. Podczas procesu projektowego, który operuje na modelu pośrednim systemu, decyzja podziału jest opóźniana tak długo jak to możliwe. W chwili, gdy znane są wszelkie aspekty dotyczące wydajności i ograniczeń analizowanego systemu (takich jak: czas, koszty, interfejs wewnętrzny, i inne), wówczas możliwy jest właściwy podział systemu na część programową i sprzętową. Projektowanie heterogenicznych systemów z wykorzystaniem metodologii projektowania zintegrowanego, wymaga posługiwania się sformalizowanym, matematycznym modele formalnym, który jest pryzmatem budowy modelu pośredniego systemu. Artykuł prezentuje nowy model formalny bazujący na sieciach Petriego, dedykowany dla heterogenicznych systemów zintegrowanych.

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

The formal model for hardware-software digital systems

Abstract

The hardware/software co-design is the one of a few newest and still under development design technologies dedicated for embedded systems. The innovation depends on change of the point decision in the design flow, which the design flow concerns system partitioning process. The partitioning decision is delayed as long as it is possible while processing decomposition operations on permanently integrated design. The decision is taken, when there are known all detailed data about possibilities and limitations of analyzed system, e.g.: time, costs, interconnections, etc. To design heterogonous system using hardware-software co-design methodology, there is required (good) formulated, mathematic model that describes complete system functionality, its properties and configuration. This paper presents a new formal model for hardware-software digital systems based on Petri nets.

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

1. Wstęp

Współczesne projektowanie systemów cyfrowych wymaga przeprowadzenia rozważań, badań oraz analiz szerokiej gamy cech, właściwości i parametrów pracy projektowanego urządzenia [2]. Kluczowym krokiem podejmowanym już na wstępie procesu projektowego, jest opracowanie lub wybór właściwego modelu formalnego w pełni specyfikującego zachowanie dowolnego systemu cyfrowego. Ze względu na różnorodność projektowanych systemów cyfrowych, dobry model formalny powinien wspierać systemy pracujące współbieżnie, synchroniczne, asynchroniczne oraz mieszane, hierarchiczne, heterogeniczne; równocześnie zapewniając opis homogeniczny całego systemu. Istniejące modele posiadają wady, w większości natury formalnej, dyskwalifikujące ich zastosowanie lub nawet przystosowanie do specyfikacji zachowania modelu systemu cyfrowego [1, 5, 7].

2. Cechy dobrego modelu dla potrzeb opisu systemów osadzonych

W przypadku zintegrowanego projektowania sprzętowo-programowych systemów osadzonych (ang. *Hardware/Software Co-design of Embedded Systems*) dobry model powinien reprezentować [5, 7]:

- system w postaci stanów i przejść między nimi,
- hierarchię zachowania,
- równoległość,
- obsługę wyjątków (ang. *exception handling*),
- konstrukcje programistyczne,
- zakończenie zachowania (ang. *behavior completion*),
- asynchroniczność,
- czas,
- sekwencyjność,
- oraz być podatnym na analizę oraz przekształcenia dla potrzeb syntezy;

oraz spełniać następujące wymagania:

- jednoznaczność,
- łatwość translacji do i z specyfikacji funkcjonalnej i syntezowalnej,
- aparat matematyczny,
- dostępność efektywnych algorytmów,
- semantyczną spójność z językami specyfikacji i implementacji,
- niezależność od specyfikacji i implementacji (syntezy).

A zatem *dobry model musi reprezentować problem projektowy najdokładniej, jak to tylko możliwe, jednoznacznie odwzorowywać operacje projektowanego urządzenia oraz charakteryzować się przejrzystością, dostępnością narzędzi i standardem*.

3. Sieci Petriego a wymagania stawiane modelowi systemu osadzonego

Przejścia między stanami. Sieci Petriego [6] zbudowane są ze stanów (miejsc) i przejść (tranzycji) między nimi.

Konstrukcje programistyczne. Cecha ta jest odpowiedzią na pytanie: *Jakie elementy sieci (traznycje czy miejsca), powinny reprezentować operacje w modelu?* Najczęściej operacje (działania) są przypisywane (przyporządkowywane) do przejść, a oznakowanie ich miejsc wejściowych oznacza stan, w którym wszystkie sygnały wejściowe (wejścia) gotowe są dla przygotowanej do realizacji operacji. Można jednak zastosować inne podejście, w którym operacje przyporządkowane są do miejsc. Za przyjęciem takiego rozwiązania przemawiają następujące fakty:

- Takie podejście wynika wprost z definicji systemu osadzonego podanego w [3]: „*System osadzony jest takim systemem, którego zachowanie jest definiowane przez jego interakcje z otoczeniem, zazwyczaj przez sekwencję zbioru trybów, w której każdy tryb może reprezentować pozostawanie w stanie lub stan, w którym wykonywane są obliczenia*”. W tym sensie miejsce w sieci Petriego reprezentuje tryb.
- Jeśli operacje są przyporządkowane do przejść, to nie mogą być przerywane, ponieważ jako tranzycje są akcjami atomowymi.

- Specyfikacja systemu osadzonego może być podzielona na część sprzętową i programową. Taki problem może być reprezentowany przez kolorowanie sieci. Na przykład przyporządkowanie koloru do miejsca (nie tranzycji) reprezentuje wyróżnioną część systemu.
- Tranzycje w sieciach Petriego mają charakter atomowy (bezpośrednio wynikający z definicji) i próba zastosowania w nich hierarchii wymusza rozważenie problemu atomowości na różnych poziomach. Z drugiej strony miejsca nie muszą być atomowe, dlatego przypisanie i definiowanie hierarchii dla miejsc nie wymaga rozstrzygnięcia takich problemów, jakie mogłyby wystąpić w przypadku tranzycji. Oznakowanie miejsca oznacza realizację (wykonanie) danej operacji.

Hierarchia zachowania. Istnieje kilka podejść reprezentujących hierarchię w sieciach Petriego. Zasadniczo miejsce lub tranzycja może reprezentować podsieć niższego poziomu hierarchii w taki sposób, że dana podsieć (bez dodatkowych przekształceń sieci głównej) może zostać zastąpiona przez miejsce lub tranzycję. W konsekwencji do miejsca może być przyporządkowana nie tylko pojedyncza, atomowa operacja, ale również fragment kodu (nie koniecznie sekwencyjnego), który może być zastąpiony przez szczegółową reprezentację na niższym poziomie hierarchii.

Reprezentacja czasu. Opisując urządzenie na poziomie systemu zazwyczaj nie dysponuje się wystarczającymi detalami mówiącymi o dokładnych wymaganiach czasowych, ale czasami mogą być one wymagane. Parametry czasowe, podobnie jak operacje, mogą być również przyporządkowane do miejsc. Wówczas parametr czasowy reprezentuje czas wykonania danej operacji. W takim przypadku tranzycje wyjściowe miejsca z określonym czasem zostaną zrealizowane tylko wtedy, gdy upłynie zdefiniowany czas (liczony od momentu pojawienia się znacznika w danym miejscu).

Równoległość. Sieci Petriego z definicji są równoległe (opisują zdarzenia współbieżne) i nie jest konieczne definiowanie dodatkowych rozszerzeń.

Obsługa wyjątków. Pomimo, że wyjątki zmieniają stan systemu i powinny być reprezentowane przez tranzycje, to posiadają one różną semantykę ze względu na przygotowanie do realizacji i samą realizację. Z tego powodu niezbędne jest zdefiniowanie nowego typu tranzycji. W sieciach Petriego dla potrzeb modelowania systemów osadzonych będą one nazywane *tranzycjami wyjątku* (ang. *exceptions transitions*) lub krótko wyjątkami.

Zakończenie zachowania. W związku z przedstawioną reprezentacją operacji, spełnienie wymagania zakończenia zachowania (ang. *behavior completion*) jest automatyczne. Zachowanie (lub jego porcja, część) uważa się za zakończone jeśli tranzycja występująca po odpowiadającym mu miejscu została zrealizowana.

Komunikacja asynchroniczna. Sieci Petriego w podstawowej definicji są asynchroniczne. Tranzycje i wyjątki zostają zrealizowane, jeśli są przygotowane do realizacji oraz przypisane do nich warunki są prawdziwe, bez odwoływania się do synchronizujących sygnałów zegarowych. Wprowadzenie sygnałów synchronizujących nie stwarza dodatkowych problemów. W ten sposób można łatwo specyfikować synchroniczne, asynchroniczne lub mieszane systemy cyfrowe.

4. Definicja modelu formalnego heterogenicznych systemów cyfrowych

Proponowana definicja wynika z przeprowadzonej powyżej analizy poszczególnych wymagań, jakie stawia się przed modelem formalnym bazującym na sieciach Petriego dla potrzeb opisu i modelowania systemów osadzonych. W pracy [8] została zaprezentowana architektura sprzętowo-programowego mikrosystemu cyfrowego *SPMC*, którego model pośredni w procesie projektowym bazuje na przedstawionym w pracy modelu formalnym systemu cyfrowego.

Definicja 1. Sprzętowo-programowa sieć Petriego *HSPN* (ang. *Hardware/Software Petri Net*) jest uporządkowaną siódemką:

$$HSPN = (M, T, A, L, R, C, \mu_0) \tag{1}$$

gdzie: M jest zbiorem *trybów*, T jest zbiorem *tranzycji*, A jest zbiorem *łuków*, L jest zbiorem *instrukcji języka programowania*, R jest zbiorem *produktów*, C jest zbiorem *warunków logicznych* i μ_0 jest *oznakowaniem początkowym,* taką że:

- Tryb M_i jest uporządkowaną czwórką:

$$M_i = (P_i, \lambda_i, \mu_i, \tau_i) \tag{2}$$

gdzie: P_i jest *miejscem* sieci Petriego; $\lambda: P_i \rightarrow L \cup \{\varnothing\}$ jest funkcją przypisującą do miejsca zbiór instrukcji programistycznych specyfikujących zachowanie (miejsce *reprezentuje* tryb i instrukcje przypisane do niego); $\mu_i \in \{0,1\}$ jest *oznakowaniem* miejsca (miejsce może zawierać albo zero albo jeden znacznik, obecność znacznika w miejscu oznacza uruchomienie instrukcji reprezentowanych przez to miejsce); $\tau_i \in N$ jest skończonym czasem przypisanym do miejsca (reprezentuje czas wykonania instrukcji reprezentowanych przez odpowiednie miejsce); pojedyncza instrukcja l_i ze zbioru L jest przyporządkowana do produktu r_i ze zbioru produktów R; każdy produkt r_i posiada dwa parametry $\{h,s\}$ określające jego funkcjonalność w trakcie zmiany trybu pracy systemu; operacja przypisania nowej wartości dla produktu zależna jest od stanu logicznego parametru s:

$$\forall r_i \in P_i \; \exists s_{ij} = \begin{cases} 1, clk' \; active \Rightarrow r_i = \lambda(l_i) \\ 0, r_i = \lambda(l_i) \end{cases} \tag{3}$$

po wykonaniu instrukcji l_i przypisanej do trybu M_i, wartość produktu r_i może być podtrzymana lub zerowana (przypisanie wartości inicjalizującej) w zależności od parametru h:

$$\forall r_i \in P_i \; \exists h_{ij} = \begin{cases} 1, r_i = r_i \\ 0, r_i = r_{init} \end{cases} \tag{4}$$

- Tryb może być hierarchiczny. W takim przypadku miejsce P_i może być zastąpione przez *HSPN*. Taki tryb określany jest nazwą: *makromiejsce*.
- *Tranzycja* może mieć przypisany warunek logiczny, nazywany *strażnikiem* (ang. *guard*) lub *predykatem* danej tranzycji, tzn. istnieje funkcja:

$$\chi_{Ti} : T_i \rightarrow C \cup \{\varnothing\} \tag{5}$$

- *Łuki* łączą tryby z tranzycjami:

$$A \subseteq M \times T \cup T \times M \tag{6}$$

- *Oznakowanie* sieci jest zbiorem oznakowań wszystkich trybów danej sieci, tj.:

$$\mu = \{ \mu_1, \mu_2, ..., \mu_n \} \quad dla \quad M_1, M_2, ..., M_n \in M \tag{7}$$

Zasady semantyczne dotyczące *HSPN* są następujące:

- Oznakowanie danego trybu M_i jest *dostępne* dla wyjściowych tranzycji jeśli tryb jest oznakowany co najmniej przez czas przypisany do niego, tj.

$$\theta(\mu_i) \geq \tau_i \tag{8}$$

gdzie: $\theta(x)$ jest czasem, w którym x pozostaje niezmienione.

- *Tranzycja jest przygotowana* do realizacji wtedy, gdy wszystkie jej tryby wejściowe są oznakowane i ich indywidualne oznakowania są dostępne.
- *Tranzycja zostaje zrealizowana* wtedy, gdy jest przygotowana do realizacji i jej warunek jest prawdziwy (*strażnik*, *predykat*). Realizacja tranzycji usuwa znaczniki z wszystkich jej trybów wejściowych i umieszcza znaczniki we wszystkich jej trybach wyjściowych.

Gdy instrukcja programistyczna jest przyporządkowana do miejsca, to występowanie znacznika w takim trybie reprezentuje wykonanie przypisanej operacji. Z powodu niezerowego czasu wykonania danej operacji miejsce może być uwarunkowane czasem. W takim przypadku znacznik staje się dostępny dla wyjściowej tranzycji tylko wtedy, gdy upłynie czas przebywania w miej-

scu oznakowania. Jest to możliwe również wtedy, gdy rezultaty wykonania instrukcji stają się dostępne dla otoczenia.

Tryb M_j danej sieci *HSPN* jest nazywany *hierarchicznym*, jeśli reprezentuje inny *HSPN*. Taki tryb jest nazywany *makromiejscem* a sieć *podsiecią* i jest oznaczona przez:

$$\propto (M_j) \tag{9}$$

Podsieć ta spełnia następujące warunki:

- Posiada ona jeden *tryb wejściowy* (ang. *entry mode*) oznaczony jako $m_{in}(M_j)$, który pobiera znacznik, gdy opowiadające makromiejsce odbiera znakowanie. Tryb wejściowy ma ten sam zbiór tranzycji wejściowych co makromiejsce, tj.:

$$^{\bullet}(m_{in}(M_j)) = {}^{\bullet}M_j \tag{10}$$

- Oznakowanie makromiejsca reprezentuje każde obowiązujące (ważne) znakowanie podsieci.
- Posiada ona jeden tryb *wyjściowy – końcowy* (ang. *termination mode*) oznaczony jako $m_{out}(M_j)$, który jest jedynym oznakowanym trybem podsieci, w przypadku ostatniego ważnego oznakowania podsieci. Tryb końcowy ma ten sam zbiór tranzycji wyjściowych co makro miejsce, tj.:

$$(m_{out}(M_j))^{\bullet} = M_j{}^{\bullet} \tag{11}$$

5. Model formalny sprzętowo-programowych mikrosystemów cyfrowych

Definicja 2. Sieć Petriego dla potrzeb opisu sprzętowo-programowego mikrosystemu cyfrowego *PNHSDM* (ang. *Petri Net for Hardware/Software Digital Microsystems*) jest uporządkowaną trójką:

$$PNHSDM = (HSPN, E, EA) \tag{12}$$

gdzie: *HSPN* jest sprzętowo-programową siecią Petriego określoną w definicji 1, *E* jest zbiorem *wyjątków*, *EA* jest zbiorem *rozszerzonych łuków* (ang. *extended arcs*), takich że:

- *Wyjątek* jest formą tranzycji z unikalnym strażnikiem (ang. *guard condition*), tj. dla każdego wyjątku E_i istnieje funkcja:

$$\chi_{Ei} : E_i \rightarrow C \tag{13}$$

- *Rozszerzone łuki* są rozszerzeniem łuków sieci *HSPN* w taki sposób, że łuki łączą tryby z tranzycjami i wyjątkami:

$$EA \subseteq A \cup M \times E \cup E \times M \tag{14}$$

- Pojęcia zbiorów wejściowych i wyjściowych węzłów (wierzchołków) są rozszerzeniem wyjątków.

Jeśli makromiejsce *Mj* jest wejściem wyjątku E_k, to wówczas wszystkie tryby w jego podsieci są jawnie podłączone do tego samego wyjątku, tj.:

$$\forall m_i \in \propto (M_j) \quad M_j \in {}^{\bullet}E_k \Rightarrow m_i \in {}^{\bullet}E_k \tag{15}$$

Zasady realizacji wyjątków różnią się od zasad realizacji tranzycji. W szczególności w celu przygotowania do realizacji wyjątku, wystarczy aby tylko jeden z jego trybów wejściowych był oznakowany. Dodatkowo, jeśli dany tryb ma przyporządkowany czas, to jego oznakowanie jest zawsze dostępne dla wyjątku.

Realizacja wyjątku może mieć szczególny wpływ na stan podsieci makromiejsca wywołującego obsługę wyjątku. Możliwe jest wstrzymanie (pauza) wykonywania lub wprowadzenie w stan początkowy podsieci w chwili utraty znakowania przez makromiejsce. Stan logiczny ρ determinuje właściwe zachowanie podsieci:

$$\forall M \; \exists \rho = \begin{cases} 1, {}^{\bullet}M_j = M_{exception}{}^{\bullet} \\ 0, {}^{\bullet}M_j = {}^{\bullet}M_{init} \end{cases} \tag{16}$$

Zasady semantyczne dotyczące *PNHSDM* są zdefiniowane następująco:

- *Wyjątek jest przygotowany* do realizacji, jeśli co najmniej jeden z jego trybów wejściowych jest oznakowany.
- *Wyjątek zostaje zrealizowany*, gdy jest przygotowany do realizacji i jego warunek logiczny (strażnik, predykat) jest prawdziwy. Realizacja wyjątku usuwa znaczniki ze wszystkich jego oznakowanych trybów wejściowych (także z tych, dla których nie jest dostępne oznakowanie) i umieszcza znaczniki we wszystkich jego trybach wyjściowych, gdy $\rho=0$. Natomiast, wyjątek pozostawia (nie usuwa) znaczniki we wszystkich oznakowanych trybach wejściowych i oznacza wszystkie swoje tryby wyjściowe, gdy $\rho=1$.
- Wyjątki są przygotowanego do realizacji przez każde oznakowanie swoich trybów wejściowych, niezależnie od czasu w jakim są oznakowane.

6. Podsumowanie

Dla potrzeb opisu zachowania i dokumentacji systemu cyfrowego specyfikowanego modelem formalnym *PNHSDM* opracowany został format *SPNF* (ang. *System Petri Net Format*) [9] w standardzie XML [11]. Specyfikacja *SPNF* zawiera mechanizmy umożliwiające opis zachowania dowolnego systemu cyfrowego, w tym systemów heterogenicznych. Badania oraz prace implementacyjne prowadzone są dla architektury *SPMC* opracowanej w Instytucie Informatyki i Elektroniki, Uniwersytetu Zielonogórskiego, opublikowanej w pracy [8]. Ponadto, dostępny jest zestaw narzędzi CAD obsługujących model *PNHSMD* zapisany w formacie *SPNF*, wspomagających proces projektowy zintegrowanych sprzętowo-programowych mikrosystemów cyfrowych *SPMC*.

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

7. Literatura

[1] Adamski M., Skowroński Z.: Interpretowane sieci Petriego - model for-malny w zintegrowanym projektowaniu mikroprocesorowych systemów sprzętowo-programowych, Pomiary Automatyka Kontrola, 2003, nr 2-3, wyd. spec., s. 17-20

[2] De Micheli, G.: Computer-Aided Hardware/ Software Codesign, In: IEEE Micro, vol. 14, No. 4, pp. 10-16

[3] Gajski D. D., Vahid F., Narayan S., Gong J.: Specification and Design of Embedded Systems; Prentice Hall, Englewood Cliffs, NJ, 1994

[4] Kozlowski, T., Dagless E. L., Saul J. M., Adamski M., Szajna J.: Parallel controller synthesis using Petri nets, In: IEE Proc. - Comput. Digit. Tech., vol. 142, No. 4

[5] Mirkowski J., Yakovlev A.: 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

[6] Murata T.: Petri Nets: Properties, Analysis and Applications, Proceedings of the IEEE, Vol.77, nr 4, April 1989

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

[8] Stasiak A.: Zintegrowany mikrosystem sprzętowo-programowy jako głów-na jednostka przetwarzania w systemach SOPC, KKE04, Kołobrzeg, Polska, 2004

[9] Stasiak A., Adamski M.: System Petri Net Format, PDS'2006, Brno, Czechy, 2006

[10] Wolf W.: Hardware/Software Co-Design of Embedded Systems, In: Proceedings of the IEEE, vol. 82, No. 7, pp. 967-989

[11] Organization for the Advancement of Structured Information Standards (OASIS), http://www.oasis-open.org/home/index.php

Artykuł recenzowany