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

Metody walidacji i weryfikacji specyfikacji funkcjonalnej mikrosystemu cyfrowego

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

Proces projektowy systemów cyfrowych, a w szczególności zintegrowanych sprzętowo-programowych mikrosystemów cyfrowych realizowanych z wykorzystaniem platformy SOPC, nie może być zakończony pomyślnie bez przeprowadzenia sprawdzenia i korekcji ewentualnych błędów projektowych. Do tego celu stosuje się metody walidacji zapisu specyfikacji funkcjonalnej zachowania systemu oraz metody weryfikacji funkcjonalnej projektowanego modelu. W artykule scharakteryzowano oraz omówiono opracowane na Uniwersytecie Zielonogórskim metody i oprogramowanie do walidacji i weryfikacji funkcjonalnej specyfikacji funkcjonalnej modelu opisanej sieciami Petriego.

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

Validation and verification methods of the digital microsystem functional specification

Abstract

The design process of the today digital systems, especially integrated hardware-software digital microsystems for SOPC platform, can not be finalized successfully without the verification and debug process. To meet the product requirements, like: system functionality, system stability, time-to-market, project costs, etc; there has to be performed several validation and/or verification tasks to confirm the finale model behaviour with initial/input functional specification. This paper describes validation and verification methods as well as software/tools elaborated and developed at University of Zielona Góra, by the Computing and Engineering Department.

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

1. Wstęp

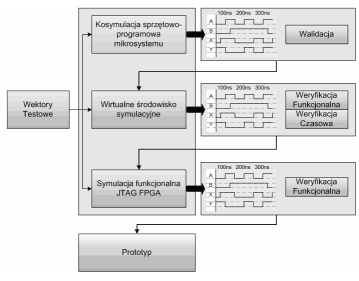
Współczesne projektowanie mikrosystemów cyfrowych [5] wymaga przeprowadzenia rozważań, badań oraz analiz szerokiej gamy cech, właściwości i parametrów pracy projektowanego urządzenia lub systemu [1, 3]. Podążanie w kierunku gotowego produktu musi odbywać się pod pełną kontrolą. Do tego celu służą specjalizowane metody i techniki weryfikacji funkcjonalnej realizowanego zadania. Proponowanych jest szereg narzędzi typu CAD wspierających etap wykrywania i eliminacji błędów, tj. symulacja, symulacyjna analiza weryfikowanego kodu (ang. code coverage), generatory ruchu (np. TCP/IP,), graficzna generacja wektorów testowych, i inne. Temat analizy i wykrywania błędów funkcjonalnych w oprogramowaniu lub opisie zachowania sprzętu jest bardzo szerokim zagadnieniem. W artykule autorzy skupili się jedynie na usystematyzowaniu oraz propozycji opracowanych metod walidacji i weryfikacji ze szczególnym uwzględnieniem aspektów projektowania zintegrowanego sprzętowo-programowych (mikro)systemów cyfrowych.

2. Metody walidacji i weryfikacji opisu behawioralnego

Istotą procesu walidacji jest zapewnienie zgodności funkcjonalnej opisu behawioralnego (mikro)systemu cyfrowego (czyli opisu wzrocowego/wejściowego) względem modelu pośredniego uzyskanego w procesie translacji [3] oraz po procesie dekompozycji funkcjonalnej w projektowaniu zintegrowanym. Natomiast, weryfikacja funkcjonalna dotyczy kontroli wyników uzyskanych w procesie symulacji funkcjonalnej systemu (na poziomie RTL) lub symulacji czasowej – ze specyfikacją funkcjonalną systemu.

Artykuł przedstawia narzędzia oraz metody walidacji i weryfikacji specyfikacji funkcjonalnej opracowane na Uniwersytecie Zielonogórskim, zorientowane na zaproponowany w pracy [5, 7] model formalny sprzętowo-programowych sieci Petriego [2] PNHSDM (ang. Petri Nets for Hardware Software Digital Microsystems) [5, 7]. Proponowane rozwiązania dotyczą szczególnie metodologii zintegrowanego projektowania sprzętu i oprogramowania, dlatego w artykule autorzy głównie odnoszą się do metodologii projektowania homogenicznego. Pomimo to, omawiane metody i narzędzia znajdują również szerokie zastosowanie i mogą zostać wykorzystywane w procesie projektowania heterogenicznego/klasycznego sprzętu i oprogramowania. Rysunek 1 przedstawia kolejność/harmonogram i zależność metod walidacji i weryfikacji w połączeniu ze wspólnym źródłem wektorów testowych.

W punkcie 2.1. omówiono walidację opisu zachowania systemu po procesie translacji oraz dekompozycji funkcjonalnej. Na rzecz prowadzonych prac naukowych opracowano i wykonano kosymulator CoSPeN sprzętowo-programowych sieci Petriego, realizujący proces kosymulacji z zachowaniem semantyki sieci Petriego oraz modelu formalnego mikrosystemu cyfrowego PNHSDM. Koniecznością wykonania nowego symulatora był brak rozwiązań akademickich i komercyjnych w zakresie realizacji kosymulacji sprzętowo-programowej sieci Petriego.



Rys. 1. Walidacja/weryfikacja specyfikacji funkcjonalnej [4]
Fig. 1. Validation/verification of the functional specification [4]

Punkt 2.2 przedstawia weryfikację funkcjonalną pracy projektowanego systemu poprzez symulację zintegrowanego (mikro)systemu sprzętowo-programowego w opracowanym wirtualnym środowisku projektowym. Punkt 2.3 omawia metodę weryfikacji funkcjonalnej (mikro)systemu cyfrowego z wykorzystaniem

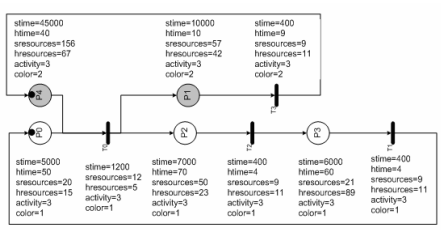
opracowanego na rzecz prowadzonych prac naukowych symulatora JTAG-SIM, który pozwala na przeprowadzenie symulacji po procesie implementacji, bezpośrednio w układzie FPGA.

2.1. Walidacja specyfikacji funkcjonalnej w projektowaniu zintegrowanym

Na podstawie specyfikacji funkcjonalnej systemu, opracowywane są zbiory wektorów testowych (X), pozwalające na szczegółową analizę behawioralną na poziomie systemu (A). Po zakończeniu procesu translacji specyfikacji wejściowej do modelu pośredniego, niezbędne jest przeprowadzenie walidacji wyników translacji. Do tego celu wykorzystuje się dedykowane symulatory, które muszą wspierać model formalny projektowanego systemu. Dla modelu PNHSDM opracowano symulator CoSPeN. Po procesie translacji, model pośredni poddawany jest walidacji z wykorzystaniem przygotowanych wcześniej wektorów testowych (X) oraz dokonuje się porównania uzyskanych wyników walidacji (B) z danymi zbioru (A). Proces walidacji przeprowadza się również po zakończeniu etapu dekompozycji funkcjonalnej systemu w procesie projektowania zintegrowanego. Tutaj niezbędna jest walidacja poprawności podziału, a w konsekwencji pracy opracowanego rozwiązania sprzętowo-programowego systemu cyfrowego – w szczególności zależności czasowych. Walidacja dokonywana jest poprzez kosymulację programowo-sprzętowej podzielonej sieci Petriego. W procesie kosymulacji wymagane jest podanie tego samego wektora testowego, na podstawie, którego przeprowadzono walidację modelu pośredniego.

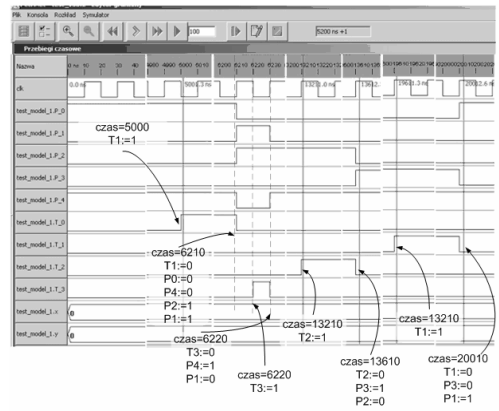
Wynikiem procesu dekompozycji specyfikacji funkcjonalnej systemu cyfrowego są dwa zbiory sieci Petriego: zbiór programowy i zbiór sprzętowy. Model pośredni zapisany w postaci elektronicznej, w formacie SPNF [6], przechowuje wszystkie informacje niezbędne do przeprowadzenia procesu kosymulacji w opracowanym symulatorze sprzętowo-programowej sieci Petriego CoSPeN. Symulator dokonuje analizy zapisu sieci uwzględniając znaczniki alokacji IMPLEMENT oraz punkty podziału sieci na część programową i sprzętową [5]. Tranzycjom podziału przydzielany jest czas, wynikający z kosztów realizacji komunikacji program-sprzęt lub sprzęt-program. Sprzętowo-programowa sieć Petriego charakteryzuje się parametrami opóźnień czasowych i kosztów implementacji, adekwatnie do przydziału implementacyjnego miejsca.

Dla każdego miejsca sieci z rysunku 2, przydzielono w sposób syntetyczny koszt czasu realizacji zadania dla części programowej i sprzętowej.



Rys. 2. Specyfikacja modelu poddawanego walidacji
Fig. 2. An example of the model specification subjected to the validation process

Dla przykładu z rysunku 2, przyjmuje się następujący podział zadań przydzielonych do implementacji jako rozwiązania programowe: P0,T0,P2,T2,P3,T4; i sprzętowe: P4,P1. Każdy obiekt modelu pośredniego otrzymuje znacznik implementacyjny (przydział do realizacji programowej lub sprzętowej), na podstawie którego oprogramowanie CoSPeN realizuje proces współsymulacji, czyli pobiera właściwe wartości czasowe obiektu modelu do symulacji. Cykl pracy badanego modelu sprzętowo-programowego mikrosystemu wynosi 20010 [ns] (suma wartości parametru *stime* dla realizacji programowych i suma wartości parametrów *htime* obiektów realizacji sprzętowej). Natomiast minimalny czas cyklu pracy tej samej specyfikacji wykonywanej tylko przez mikroprocesor wynosi 75400[ns] – suma wszystkich czasów parametru *stmie*. Na rysunku 3 przedstawiono wyniki pracy symulatora CoSPeN.



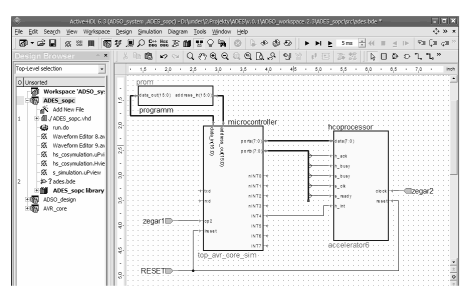
Rys. 3. Przykład procesu kosymulacji sprzętowo-programowej sieci Petriego
Fig. 3. Hardware-software cosimulation of the Petri nets

Wyniki walidacji sprzętowo-programowej sieci Petriego specyfikującej system cyfrowy, dostarczają wzorców pracy systemu niezbędnych do weryfikacji wyników symulacji funkcjonalnej uzyskanych w środowisku wirtualnym.

2.2. Weryfikacja funkcjonalna z wykorzystaniem wirtualnego środowiska prototypowego

Opracowane na rzecz prowadzonych prac naukowo-badawczych, wirtualne środowisko prototypowania i weryfikacji systemu i mikrosystemu sprzętowo-programowego, zbudowane zostało na bazie modelu IP CORE mikroprocesora AVR Atmega103 oraz bloku sprzętowego HDL w postaci komponentu. Symulacja czasowa badanego (mikro)systemu cyfrowego odzwierciedla rzeczywistą pracę modelu wirtualnego w układzie FPGA, zapewniając zachowanie czasowych parametrów implementacyjnych z dokładnością 98% [8]. W pracy wykorzystano model syntezowanego mikroprocesora RISC AVR Atmega103 ze względu na licencję użytkowania (GPL, ang.General Public License), parametrów pracy oraz stan zaawansowania prac nad rdzeniem procesora. Testy przeprowadzono za pomocą dwu symulatorów języków opisu sprzętu: MentorGraphics ModelSim [9] oraz Aldec Active-HDL [10]. Wykonane środowisko wirtualnego prototypowania zostało opisane w języku VHDL gwarantując niezależność od dostawcy symulatora HDL.

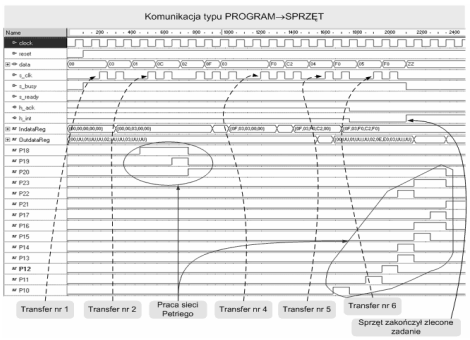
Przygotowanie nowego (mikro)systemu do współ-symulacji programowo-sprzętowej, czyli weryfikacji funkcjonalnej lub czasowej, sprowadza się do wprowadzenia plików wynikowych procesu projektowego (syntezy sprzętowej i programowej) [5] do właściwej struktury katalogów projektu środowiska wirtualnego. Środowisko jest gotowe do natychmiastowego uruchomienia procesu symulacji wirtualnej. Budowę blokową środowiska wirtualnej symulacji SPMC przedstawia rysunek 4.



Rys. 4. Wirtualne środowisko prototypownia
Fig. 4. The prototyping virtual environment

Wyniki przykładowej symulacji transmisji program↔sprzęt (czyli pomiędzy procesorem a jednostką sprzętową) oraz zachowania części programowej i sprzętowej, przedstawiono na rysunku 5. Przykład obrazuje transmisję typu program→sprzęt badanego przykładowego mikrosystemu sprzętowo-programowego z wykorzystaniem opracowanego środowiska prototypowania. Na przebiegu

cyfrowym (rysunek 5) realizowany jest proces komunikacji program-sprzęt. Obserwacji poddawany jest nie tylko proces komunikacji (weryfikacja protokołu komunikacyjnego badanego systemu), ale również możliwy jest podgląd stanu wewnętrznych rejestrów mikroprocesora oraz stan pracy zaprojektowanego urządzenia cyfrowego (rysunek 5, etykieta „Praca sieci Petriego").



Rys. 5. Przykład procesu symulacji
Fig. 5. An example of the simulation process

Metoda weryfikacji funkcjonalnej znajduje szerokie zastosowanie w projektowaniu układów i (mikro)systemów cyfrowych. Przedstawione środowisko jest gotowym i uniwersalnym rozwiązaniem. Ze względu na długi czas symulacji czasowej (ang. gate-level), możliwe jest przeniesienie opracowanego środowiska wirtualnego do akceleratora procesu symulacji HDL (np. HES [10]). Ze względu na wzrastającą złożoność realizowanych projektów (mikro)systemów cyfrowych, często pomija się weryfikację typu „gate-level", zakańczając weryfikację badanego modelu na symulacji funkcjonalnej.

2.3. Weryfikacja w rzeczywistym układzie FPGA

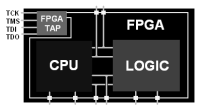
Ostatnią metodą weryfikacji funkcjonalnej (mikro)systemu cyfrowego jest symulacja w docelowym układzie FPGA. Opracowano oprogramowanie pozwalające na przeprowadzenie rzeczywistej weryfikacji i testy zaprojektowanego (mikro)systemu już po procesie implementacji, w układzie FPGA. Wykonane oprogramowanie JTAG-SIM może współpracować z komercyjnym oprogramowaniem Xilinx ISE lub Altera Quartus2 [11]. Testy funkcjonalne przykładowego systemu cyfrowego przeprowadzono z wykorzystaniem układów Spartan II oraz Spartan IIE w obudowie PQ208. Wyniki w programie JTAG-SIM prezentowane są na dwa sposoby: a) za pomocą wykresów przebiegów czasowych (ang. waveform), rysunek 6, b) poglądowy obiekt obudowy układu FPGA z wyprowadzeniami (ang. chip view) [5].

Rys. 6. Przykładowe wyniki symulacji w oprogramowaniu JTAG-SIM
Fig. 6. JTAG-SIM simulation results

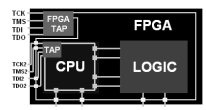
Wykresy przebiegów czasowych służą głównie do wizualizacji przeprowadzonych testów, które to wyniki będą w późniejszym czasie analizowane bądź porównywane z innymi danymi symulacji funkcjonalnej. Doskonałym przykładem takich testów może być przeprowadzanie symulacji funkcjonalnej w układzie FPGA za pomocą instrukcji INTEST z wykorzystaniem skryptu testującego (ang. testbench). Taki test można przeprowadzać dla kolejnych wersji rozwiązań systemu, a następnie porównać zmiany dla różnych wersji implementacyjnych lub dokonać porównania z wynikami walidacji lub symulacji funkcjonalnej HDL.

Jedną z newralgicznych części homogenicznego systemu cyfrowego jest magistrala komunikacyjna pomiędzy procesorem a współpracującą logiką (część sprzętowa). Proces symulacji przeprowadzany jest wewnątrz struktury FPGA, zatem dostęp do tych informacji jest niezbędny.

Opracowano trzy metody dostępu do wewnętrznej struktury (mikro)systemu. Pierwsza polega na wyprowadzeniu linii magistral komunikacyjnych znajdujących się pomiędzy procesorem a współpracującą logiką, na wejścia/wyjścia układu – rysunek 7. Druga metoda, przedstawiona na rysunku 8 polega na umieszczeniu wewnątrz struktury FPGA dodatkowego, dedykowanego kontrolera TAP, służącego do kontroli pracy procesora. Dzięki takiemu rozwiązaniu możliwa jest pełna kontrola nad wejściami oraz wyjściami procesora znajdującymi się wewnątrz struktury FPGA, co pozwala na testowanie procesora niezależnie od otaczającej go logiki, oraz logiki niezależnie od działania procesora.

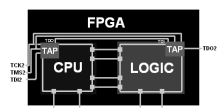


Rys. 7. Wykorzystanie głównego kontrolera TAP
Fig. 7. The use of main FPGA TAP controller



Rys. 8. Kontrolera TAP dla procesora
Fig. 8. The use of embedded processor TAP controller

Ostatnia proponowana metoda, zobrazowana na rysunku 9, całkowicie rezygnuje z kontrolera TAP układu FPGA na rzecz połączonych ze sobą wewnętrznych kontrolerów TAP sterujących pracą poszczególnych modułów (na przykład procesora oraz dedykowanej logiki).



Rys. 9. Wykorzystanie dedykowanych kontrolerów TAP dla procesora i modułu sprzętowego
Fig. 9. Implementation of a dedicated TAP controller for the FPGA module and embedded processor

3. Podsumowanie

Aktualność praw Moore'a potwierdzają najwięksi producenci układów cyfrowych [12]. Wzrost złożoności projektowanych dziś układów i systemów cyfrowych wymaga podjęcia dodatkowych kroków zmierzających do podniesienia wiarygodności ich poprawnej funkcjonalności. Przedstawione w artykule rozwiązania systematyzują znane opracowania, proponując schemat procesu walidacji i weryfikacji projektowanego systemu cyfrowego jednocześnie dostarczając opracowane przez autorów gotowe rozwiązania.

4. Literatura

[1] Adamski M., Skowroński Z.: Interpretowane sieci Petriego - model formalny w zintegrowanym projektowaniu mikroprocesorowych systemów sprzętowo-programowych, Pomiary Automatyka Kontrola, 2003, nr 2-3, wyd. spec., s. 17-20
[2] Murata T.: Petri Nets: Properties, Analysis and Applications, Proceedings of the IEEE, Vol.77, nr 4, April 1989
[3] Z.Skowroński, Translacja specyfikacji funkcjonalnej układów cyfrowych na sieć Petriego dla potrzeb syntezy systemowej, PhD, Politechnika Szczecińska, Szczecin 2000
[4] Stasiak A.: Zintegrowany mikrosystem sprzętowo-programowy jako główna jednostka przetwarzania w systemach SOPC, KKE04, Kołobrzeg, Polska, 2004
[5] A.Stasiak, Automatyczna dekompozycja specyfikacji behawioralnej sprzętowo-programowego mikrosystemu cyfrowego, Rozprawa Doktorska, Uniwersytet Zielonogórski, Zielona Góra, 2007
[6] Stasiak A., Adamski M.: System Petri Net Format, PDS'2006, Brno, Czechy, 2006
[7] Z.Skowroński, A.Stasiak, The intermediate model for hardware/software Microsystems based on Petri nets, DESDes'04, Wydawnictwo Uniwersytetu Zielonogórskiego, Dychów 2004, ISBN 83-89712-16-4
[8] http://www.xilinx.com
[9] http://www.modelsim.com
[10] http://www.aldec.com, http://www.alatek.com
[11] http://www.altera.com
[12] http://www.intel.com

Artykuł recenzowany