Andrzej STASIAK
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI

Optymalizacja realizacji układowej hierarchicznych sieci Petriego

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

Streszczenie

Projektowanie wysokiego poziomu systemów cyfrowych dotyczy szeregu aspektów związanych bezpośrednio z wybraną metodologią projektowania, modelem formalnym, algorytmami syntezy sprzętowej i programowej opisu behawioralnego. Rozważając sieci Petriego jako model formalny projektowanego systemu, szczególnie interpretowane hierarchiczne sieci Petriego, projektant ma do dyspozycji bogaty zbiór algorytmów analizy formalnej, metod syntezy programowej i sprzętowej. Jednak znane rozwiązania dotyczą przede wszystkim opisu zachowania funkcjonalnego sterowników logicznych, gdzie miejsce sieci reprezentuje logiczną operację przypisania poziomu '0' lub '1' do wyjścia układu (jeden bit lub wektor). Zauważalny jest jednak brak propozycji naukowych wspierających projektowanie systemu opisanego sieciami Petriego, gdzie w miejscu sieci realizowane są złożone instrukcje arytmetyczne lub logiczne. Przykładem sieci Petriego wspierającej w pełni projektowanie systemu sprzętowo-programowego jest siec PNHSDM (ang. Petri Net for Hardware So-ftware Digital Microsystem). Artykuł w sposób ogólny przedstawia model formalny sprzętowo-programowych sieci Petriego PNSHDM, skupiając się szczególnie na metodzie optymalizacji algorytmu syntezy sprzętowej sieci PNHSDM do reprogramowalnych układów FPGA. W pracy podjęto tematykę szeregowania zadań (ASAP, ALAP). Rozwiązano problem systemu przełączania sprzętowych modułów wykonawczych w kontekście planowanych do wykonania zadań. Wyznaczono wzór pozwalający na oszacowanie kosztów realizacji sprzętowej systemu przełączania, który zależny jest od liczby instancjacji i złożoności harmonogramowanej instrukcji.

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

The implementation optimization of the hierarchical Petri nets

Abstract

The high level designing concerns several aspects that are directly related to the designing methodology as well as formal model definition, hardware/software synthesis algorithms, etc. When a Petri nets are considered as a formal model of developing system, especially hierarchical interpreted Petri nets, then a designer has a lot of ready to use and very well documented algorithms dedicated for formal verification, hardware and software synthesis, transformations, etc. However, most of the know solutions for Petri nets concerns designing logic controllers, where a one net place implements simple assigning operation of logical state to the output. There is lack of solutions that use a Petri nets to formalize and describe really system, where a place implements complex algebraic or logic functions. A formal model based on Petri nets for hardware-software digital microsystems (PNHSDM) has been elaborated to eliminate this gap. This paper shortly describes PNHSDM model, but author have concentrate on optimization of hardware Petri nets synthesis algorithm that is responsible to translate functional description into VHDL-RTL language. This approach takes into consideration task scheduling (ASAP, ALAP), presents elaborated solution of switching system for PNHSDM Petri nets as well as provides static hardware cost estimations of the switching system.

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

1. Model formalny PNHSDM

Projektowanie sprzętowo-programowej mikrosystemu lub systemu cyfrowego wymaga przeprowadzenia rozważań, badań oraz analiz szerokiej gamy cech, właściwości i parametrów pracy projektowanego systemu [5]. Kluczowym krokiem podejmowanym już na wstępie procesu projektowego, jest opracowanie lub wybór właściwego modelu formalnego, w pełni specyfikującego dowolne zadanie. Ze względu na różnorodność realizowanych operacji, dobry model formalny powinien wspierać systemy pracujące współbieżnie, synchroniczne, asynchroniczne oraz mieszane, hierarchiczne, heterogeniczne; równocześnie zapewniając opis homogeniczny specyfikacji SPMC. Istniejące modele posiadają pewne wady, w większości natury formalnej, dyskwalifikujące ich zastosowanie lub nawet dostosowanie do specyfikacji zachowania modelu sprzętowo-programowego systemu cyfrowego [2, 3]. Dobry model musi reprezentować problem projektowy najdokładniej, jak to tylko możliwe, jednoznacznie odwzorowywać operacje projektowanego urządzenia oraz charakteryzować się przejrzystością oraz dostępnością narzędzi. W pracy [4] za model formalny przyjęto interpretowane, czasowe, hierarchiczne sieci Petriego. Opracowano nowy model formalny PNHSDM [4]. Nowatorstwo polega na wprowadzeniu szeregu nowych definicji i rozwiązań technicznych umożliwiających specyfikowanie heterogenicznych systemów cyfrowych. Cechą wyróżniającą model PNHSDM jest możliwość opisu dowolnego systemu, w tym instrukcji programistycznych oraz zadań a wręcz konstrukcji sprzętowych.

Metoda syntezy sprzętowej modelu układu cyfrowego opisanego sieciami PNHSMC, wykorzystuje opracowania pracy doktorskiej P.Wolańskiego [6], w szczególności metodę syntezy zorientowaną na tranzycje. Nowatorskim rozwiązaniem, proponowanym w artykule, a dotyczącym syntezy sieci Petriego do języków opisu sprzętu, jest zastosowanie opracowanej metody optymalizacji zasobów sprzętowych w realizacji układowej hierarchicznych sieci Petriego.

2. Optymalizacja obszaru implementacji części sprzętowej systemu opisanego sieciami PNHSDM

Opracowana metoda syntezy sprzętowej sieci Petriego optymalizuje obszar implementacyjny części sprzętowej systemu poprzez wielokrotne wykorzystanie jednego, kilkukrotnie instancjonowanego, sprzętowego bloku funkcyjnego. Podobne techniki wykorzystywane są w syntezie wysokiego poziomu podczas procesu szeregowania ASAP, ALAP [1]. Nie są znane natomiast autorowi jakiekolwiek implementacje algorytmów szeregowania w procesie syntezy hierarchicznych sieci Petriego.

Ze względu na charakter prowadzonych rozważań, a w szczególności ze względu na znaczące ograniczenia dotyczące obszaru logiki programowalnej, podjęto prace naukowo-badawcze, których celem było opracowanie techniki optymalizacji implementacyjnej hierarchicznych sieci Petrigo. Postawiono jedno główne założenie dla metody optymalizacji:

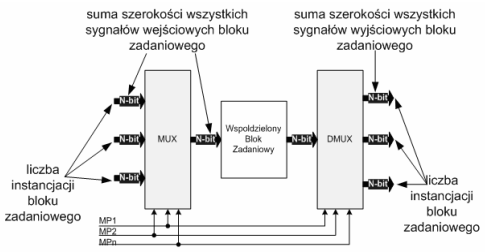
- Wyznaczenie funkcji kosztów optymalizacji. Niezbędne jest określenie kosztu implementacji systemu kontroli i przełączania optymalizowanych miejsc. Istnieje niebezpieczeństwo zwiększenia kosztów implementacyjnych w sytuacji, gdy suma kosztów logiki przełączania (obszar) jest większy od sumy kosztów implementacji zasobów zadaniowych poddawanych optymalizacji.

Procesowi optymalizacji mogą zostać poddane jedynie makromiejsca typu współdzielonego [5], instancjonujące ten sam model nie posiadający deklaracji sygnałów z podtrzymaniem oraz makromiejsca nie będące w relacji współbieżności względem siebie.

System przełączania/sterowania współdzielonego bloku zadaniowego zbudowany jest z wejściowego bloku wyboru – multiplekser, oraz wyjściowego bloku wyboru – demultiplekser. Oba bloki sterowane są wspólnym zbiorem sygnałów wyboru, rysunek 1. Cały

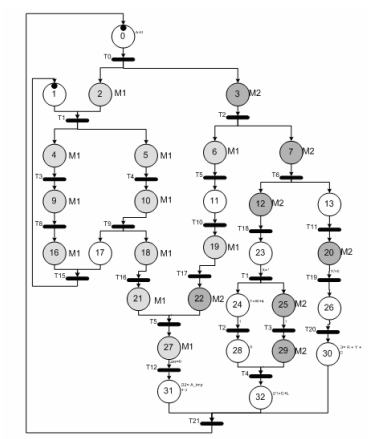
system przełączania specyfikowany jest w języku opisu sprzętu VHDL. Zadanie polega na obliczeniu liczby bramek logicznych (liczba bloków HMAP i FMAP komórki programowalnej CLB układu Xilinx FPGA [7]) niezbędnych do implementacji multipleksera i demultipleksera.

Proces syntezy logicznej kodu VHDL realizowany jest przez narzędzia komercyjne. W pracy wykorzystano oprogramowanie Xilinx ISE 7.3 oraz MentorGraphic LeonardoSpectrum 2002, gdzie w procesie syntezy logicznej układy kombinacyjne realizowane są z wykorzystaniem bramek 2-wejściowych.



Rys. 1. System przełączania współdzielonego bloku zadaniowego
Fig. 1. An elaborated switching system

Nowatorskim opracowaniem jest statyczne szacowanie kosztów implementacji cyfrowego układu przełączania optymalizowanych komponentów zadaniowych (makromiejsc). Sieć Petriego specyfikująca zachowanie części sprzętowej poddawana jest szczegółowej analizie formalnej. Wyznaczone zostają wszystkie możliwe sekwencje deklaracji makromiejsc spełniających postawione wcześniej wymagania. Na rysunku 2 przedstawiono sieć ze zdefiniowanymi makromiejscami instancjonującymi modele: M1 i M2.



Rys. 2. Sieć hierarchiczna (deklaracje wielu makromiejsc instancjonujących wspólny model)
Fig. 2. Model specified with use of hierarchical Petri net (declaration of several macro-places that instantiate the same resource/module)

Rezultatem analizy jest sześć zbiorów miejsc sekwencji instancjonujących wspólny zasób zadaniowy:

zbiór 1→M1:2,4,9,16
zbiór 2→M1:2,5,10,18,21,27
zbiór 3→M1:6,19,27
zbiór 4→M2:3,22
zbiór 5→M2:3,7,12,25,29
zbiór 6→M2:3,7,20.

Algorytm optymalizuje zbiory zaczynając od grupy o największej liczbie makromiejsc instancjonujących wspólny model. W przypadku równości zbiorów pod względem liczby składników, kosztów realizacji sprzętowej i programowej, czasów pracy – wybór dokonywany jest losowo. Rozważając przykład z rysunku 2, w pierwszym kroku algorytmu, dla modelu M1 optymalizacji poddany zostanie zbiór nr 2 (miejsca 2,5,10,18,21,27), natomiast dla modelu M2 zbiór nr 5 (miejsca 3,7,12,25,29). Miejsca zakwalifikowane do optymalizacji zostają usunięte ze wszystkich zbiorów. Dla omawianego przykładu pozostaną cztery zbiory o zredukowanej liczbie miejsc sieci:

zbiór 1→M1:4,9,16
zbiór 3→M1:6,19
zbiór 4→M2:22
zbiór 6→M2:20.

Zbiory zawierające jedno miejsce nie są poddawane optymalizacji. Dla rozważanego zadania z rysunku 2, procesowi optymalizacji poddane zostaną zbiory: 2, 5, 1, 3. Optymalizacja obszaru implementacji w układzie FPGA realizowana jest za pomocą systemu przełączania współdzielonego komponentu zadaniowego, pomiędzy zdefiniowany zbiór miejsc specyfikacji funkcjonalnej.

Jednym z kluczowych problemów przedstawionej metody optymalizacji zasobów sprzętowych w implementacji sprzętowych sieci Petriego (w szczególności sieci hierarchicznych), jest znalezienie odpowiedzi na pytanie: *Czy wybrana sekwencja przyniesie wymierne korzyści implementacyjne?*

W procesie analizy i estymacji kosztów części sprzętowej, dla każdego makromiejsca określany jest koszt realizacji układowej wyrażony jako liczba konfigurowalnych bloków logicznych CLB układu reprogramowalnego FPGA. W celu zobrazowania problemu rozważono następujący przykład.

Przykład. Suma zbioru nr 2 wszystkich makromiejsc rysunku 2 wynosi 346 [CLB]. Ile zasobów logiki reprogramowalnej zostanie zajętych przez system przełączania poszczególnych instancji? Nie jest znany koszt implementacji logiki przełączania (sterowania) współdzielonego modelu M1 w zależności od oznakowania sieci, który zależny jest od interfejsu wejścia (liczba i szerokość sygnałów wejściowych) oraz wyjść (liczba i szerokość sygnałów wyjściowych) instancjonowanego modelu. Jeśli koszt układu przełączania będzie mniejszy od kosztów implementacji kompletnej listy rozważanych makromiejsc, wówczas optymalizacja przyniesie zysk. W przeciwnym przypadku algorytm optymalizacji **zwiększy** koszt implementacji części sprzętowej ←sytuacja NIEDOPUSZCZALNA.

2.1. Wyznaczenie kosztu implementacji multipleksera

Blok multipleksera składa się z N liczby M-wejściowych bramek AND oraz W-wejściowej bramki OR. Na wejście bramki AND podawany jest wektor sumy wszystkich sygnałów wejściowych bloku zadaniowego oraz zbiór sygnałów sterujących, więc:

$$A1 = IN_{count} + \log(IMPL_{count}) \tag{1}$$

gdzie: $A1$ – liczba bitów wejścia bramki AND, IN_{count} – suma szerokości wszystkich sygnałów wejściowych bloku zadaniowego (makromiejsca), $IMPL_{count}$ – liczba implementacji bloku zadaniowego.

Znajdujemy liczbę dwu-wejściowych bramek AND do realizacji multipleksera. Ponieważ bramkę o N wejściach można rozłożyć na (N-1) bramek 2-wejściowych, dlatego liczba bramek jednej instancji bloku zadaniowego jest równa A1=A1-1. Aby obliczyć całkowitą liczbę 2-wejściowych bramek AND multipleksera, należy pomnożyć liczbę bramek AND dla realizacji jednej instancji przez liczbę implementacji, więc

$$sumaAND = A1 * IMPL_{count} \tag{2}$$

Na wejście bramki OR podawany jest wektor sumy wszystkich sygnałów wejściowych bloku zadaniowego pomnożony przez liczbę implementacji, więc

$$sumaOR = (IN_{count} * IMPL_{count}) - 1 \tag{3}$$

Koszt implementacji układowej multipleksera: $K_{mux} = sumaAND + sumaOR$, więc:

$$K_{mux} = IMPL_{count}\left[IN_{count} + \log(IMPL_{count}) - 1\right] + (IN_{count} * IMPL_{count}) - 1 \tag{4}$$

2.2. Wyznaczenie kosztu implementacji demultipleksera

Blok demultipleksera składa się z N liczby M-bitowych bramek wyjściowych AND. Pierwszym zadaniem jest wyznaczenie liczby bramek jednej instancji,

$$A2 = OUT_{count} + \log(IMPL_{count}) \tag{5}$$

gdzie: $A2$ – liczba bitów wejścia bramki AND, OUT_{Mount} – suma szerokości wszystkich sygnałów wyjściowych bloku zadaniowego, $IMPL_{count}$ – liczba implementacji bloku zadaniowego.

Znajdujemy liczbę dwu-wejściowych bramek AND w realizacji demultipleksera A2=A2-1. Koszt implementacji układowej demultipleksera jest równy liczbie bramek jednej instancji pomnożonej przez liczbę implementacji:

$$Kdmux = (OUT_{count} + \log(IMPL_{count}) - 1) * IMPL_{count} \tag{6}$$

2.3. Wyznaczenie funkcji kosztu całkowitego

Całkowity koszt implementacji systemu przełączania i sterowania jest równy sumie realizacji układowej multipleksera i demultipleksera, czyli:

KS=Kmux+Kdmux;
KS=IMPLcount[INcount +log(IMPLcount)-1]+(INcount* IMPLcount) - 1 + +(OUTcount+ log(IMPLcount) -1) *IMPLcount

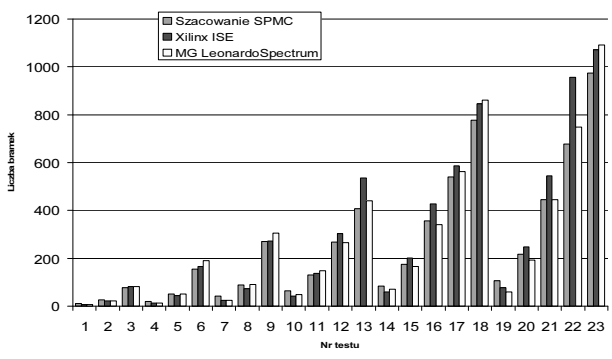
to:

$$KS = IMPL_{count}[IN_{count} + OUT_{count} + 2\log(IMPL_{count}) - 2] + (IN_{count} * IMPL_{count}) - 1 \tag{7}$$

gdzie: IN_{count} – suma szerokości wszystkich sygnałów wejściowych bloku zadaniowego, OUT_{count} – suma szerokości wszystkich sygnałów wyjściowych bloku zadaniowego, $IMPL_{count}$ – liczba implementacji bloku zadaniowego.

Poprawność wyznaczonego równania zweryfikowano doświadczalnie poprzez porównanie wyników syntezy logicznej z kosztami obliczonymi na podstawie wzoru 7.

Badaniu poddano 23 projekty o zróżnicowanej architekturze wejścia/wyjścia oraz zmiennej liczbie instancjacji bloku zadaniowego. Konfrontacji poddano narzędzia komercyjne (Xilinx ISE 7.1sp3 oraz MentorGraphics LeonardoSpectrum 2002b) realizujące proces syntezy logicznej badanego układu przełączania oraz wyniki estymacji przeprowadzonej według wzoru 7. Wyniki prezentuje rysunek 3. Wyniki wyznaczonego wzoru są akceptowalne, szczególnie zważając na fakt rozbieżności syntezy logicznej produktów komercyjnych firmy Xilinx i MentorGraphics.



Rys. 3. Wyniki konfrontacji estymacji statycznej SPMC zasobów logiki FPGA systemu przełączania z wynikami implementacji
Fig. 3. The FPGA resource SPMC static estimations vs commercials tools confrontation results

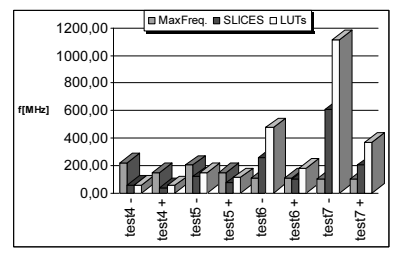
Nowatorski algorytm syntezy sprzętowej sieci Petriego, realizujący optymalizację implementacyjną części sprzętowej, posługuje się zależnością warunkującą redukcję kosztów logiki reprogramowalnej FPGA, wyrażoną nierównością:

$$KS < (IMPL_{count} - 1) * KMP \tag{8}$$

gdzie: KS – koszt całkowity systemu przełączania, KMP – koszt implementacji jednego bloku zadaniowego (makromiejsca), $IMPL_{count}$ – liczba makromiejsc poddawanych optymalizacji.

3. Podsumowanie

Wyniki syntezy złożonych sieci hierarchicznych przedstawia rysunek 4. Rezultaty procesu syntezy sprzętowej sieci Petriego PNHSDM z uwzględnieniem przedstawionej w artykule metody potwierdzają pozytywny wpływ optymalizacji na realizację układową modelu. Wzrost liczby instancjacji jednego komponentu w syntezie sprzętowej nie powoduje lawinowego wzrostu zajętości obszaru logiki układu reprogramowalnego w porównaniu z opracowaniami [6]: a) „test6+" oraz „test6-", b) „test7+" oraz „test7-". Natomiast, konsekwencją realizacji systemu przełączania jest obniżenie maksymalnej częstotliwości pracy części sprzętowej. Testy 4 i 5 są przykładami opisu modelu instancjonującego odpowiednio 11 i 18 współdzielonych komponentów sieci o małej złożoności, tj. 3 miejsca, 4 tranzycje sieci Petriego.



Rys. 4. Wpływ optymalizacji SPMC na koszt implementacji oraz maksymalną częstotliwość pracy części sprzętowej dla sieci hierarchicznych, gdzie „test-" – optymalizacja wyłączona, „test+" – optymalizacja włączona
Fig. 4. The influence of the SPMC optimization to implementation costs and frequency for model specified with use of hierarchical Petri nets: "test-" - optimization OFF, "test+" - optimization ON

Opracowana metoda optymalizacji obszaru realizacji sprzętowej sieci Petriego w układzie FPGA, przy spełnionym założeniu 8, przynosi oszczędności w liczbie alokowanych bloków CLB. Ponadto, statyczne oszacowanie kosztów implementacji systemu przełączania, wzór 7, pozawala na podęcie kluczowych decyzji w procesie syntezy sprzętowej sieci Petriego.

4. Literatura

[1] P.Eles, K.Kuchciński, Z.Pend, System Synthesis with VHDL, Kluwer Akademic Publishers, Londyn 1998, ISBN 0-7923-8082-7
[2] D.D.Gajski, F.Vahid, Specification and Design of Embedded Hardware-Software Systems , IEEE Design & Test of Computers, vol. 12, no 1, Spring 1995, pp. 53-67
[3] Z.Skowroński, Translacja specyfikacji funkcjonalnej układów cyfrowych na sieć Petriego dla potrzeb syntezy systemowej, PhD, Politechnika Szczecińska, Szczecin 2000
[4] A.Stasiak, Automatyczna dekompozycja specyfikacji behawioralnej sprzętowo-programowego mikrosystemu cyfrowego, Rozprawa Doktorska, Uniwersytet Zielonogórski, Zielona Góra, 2007
[5] A.Stasiak, Z.Skowroński, The intermediate model for hardware/ software Microsystems based on Petri nets, DESDes'04, Wydawnictwo Uniwersytetu Zielonogórskiego, Dychów 2004, ISBN 83-89712-16-4
[6] P.Wolański, Modelowanie układów cyfrowych na poziomie RTL z wykorzystaniem sieci Petriego i podzbioru języka VHDL, Rozprawa doktorska, Warszawa 1998
[7] http://www.xilinx.com

Artykuł recenzowany