Marek WĘGRZYN
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI

Częściowa rekonfiguracja sterowników binarnych opisanych sieciami Petriego

Dr inż. Marek WĘGRZYN

Adiunkt w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Pełni funkcję kierownika Zakładu Inżynierii Komputerowej. Zainteresowania badawcze obejmują zagadnienia projektowania systemów cyfrowych, ze szczególnym uwzględnieniem logiki programowalnej i języków opisu sprzętu (VHDL i Verilog).

e-mail: M.Wegrzyn@iie.uz.zgora.pl

Streszczenie

W prezentowanym artykule przedstawiono metodę projektowania reprogramowalnych sterowników binarnych pod kątem ich częściowej rekonfiguracji. Do specyfikacji algorytmów sterowania stosowana jest hierarchiczna sieć Petriego. Projektowane układy implementowane są w strukturach programowalnych FPGA. Opracowana metoda uwzględnia możliwość zastąpienia złożonych fragmentów sieci (podsieci lub mikromodułów, tzn. makromiejsc i makrotranzycji) innymi w celu dopasowania algorytmu sterowania do realizacji nowych zadań. W przeprowadzonych testach wykorzystano układu FPGA firmy Xilinx. Testy wykazały skuteczność metody przy zachowaniu łatwości modyfikacji algorytmu.

Słowa kluczowe: sterownik binarny, hierarchiczna sieć Petriego, częściowa rekonfiguracja, FPGA.

A partial reconfiguration of Petri net-described Logic Controllers

Abstract

In the paper a new method of Logic Controller design for a partial reconfiguration is presented. Programs of Logic Controllers are specified by means of hierarchical Petri nets. As a final technology programmable logic is applied. The proposed method allows replacing complex parts of the net (i.e. subnet, macroplaces, or macrotransitions) by other ones for realization of new tasks by the controller. Xilinx FPGAs have been used in tests. The tests showed a method effectiveness, while modifications of the algorithms were easy carried out.

Keywords: Logic Controller, Hierarchical Petri Net, Partial reconfiguration, FPGA.

1. Wstęp

W ostatnich latach obserwuje się duże zmiany w metodach projektowania układów i systemów cyfrowych. Nastąpiło przejście od strukturalnej reprezentacji układu do opisu układu na poziomie behawioralnym przy zastosowaniu języków opisu sprzętu (ang. *Hardware Description Language*, *HDL*). Również wprowadzane są nowe technologie wykonywania urządzeń cyfrowych. Popularność zyskują specjalizowane układy nowej generacji - układy programowalne przez użytkownika, do najważniejszych zaliczane są FPGA (ang. *Field Programmable Gate Array*) oraz CPLD (ang. *Complex Programmable Logic Device*). Układy te często charakteryzują się możliwością reprogramowania, a nawet dynamicznej rekonfiguracji [6].

Coraz więcej projektantów wykorzystujących układy FPGA zwraca się ku częściowej rekonfiguracji. Występuje to w przypadkach, gdy w projekcie można wyróżnić część statyczną (nie zmienia się w całym czasie pracy) oraz dynamiczną (dopasowywaną w zależności od realizowanych zadań). Daje ona możliwości przeprogramowania dynamicznej części układu, podczas gdy statyczna część kontynuuje pracę. Takie rozwiązanie redukuje zużycie energii oraz koszty i rozmiary projektu. Wielu producentów układów programowalnych wykonuje układy FPGA z możliwością częściowej rekonfiguracji, np. Atmel [1] oraz Xilinx [9]. Xilinx przedstawia częściową rekonfigurację jako analogię do przełączania przez mikroprocesor procesów programowych, jednak w układach FPGA przełączany jest sprzęt, a nie oprogramowanie.

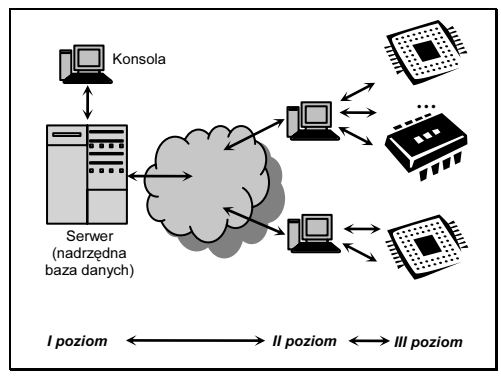
Częściowa rekonfiguracja znajduje duże zastosowanie również w układach sterowania. Takie podejście jest szczególnie przydatne w systemach, gdzie dane konfigurujące przesyłane są poprzez kanały transmisyjne o niewielkiej przepustowości. Przykładem może tu być rozproszony system sterowania przedstawiony w [7].

W rozdziale drugim zostanie opisany rozproszony system sterowania. W rozdziale trzecim przedstawione zostanie rozszerzenie metody projektowania rekonfigurowanych sterowników binarnych pod kątem częściowej rekonfiguracji.

2. Ogólny schemat rozproszonego systemu sterowania

Rozwinięcie metod sprzętowo-programowych stwarza nowe rozwiązania również w projektowaniu systemów sterowania. Zamiast wielu niezależnie działających mniejszych układów sterujących, istnieje możliwość zintegrowania ich w jeden większy system, dla którego globalne algorytmy sterowania reprezentowane są przez system rozproszony.

Zaproponowana struktura rozproszonego systemu sterowania składa się z trzech poziomów [8]. Poziomem najwyższym jest aplikacja z bazą danych, za pomocą której wybierane są, na podstawie podejmowanych decyzji, odpowiednie podprogramy sterowania do realizacji na niższych poziomach. Kolejny poziom nadzoruje wykonywanie zadań przez układy sterowania na najniższym poziomie. Połączenie pomiędzy poziomami nadrzędnym i pośrednim realizowane jest za pośrednictwem komputerowych sieci rozległych, wraz z rezerwowym kanałem komunikacyjnym (łączność radiowa lub telefonia komórkowa). Poziom pośredni reprezentowany jest przez komputer nadzorujący. Komunikacja bazy danych ze sterownikami odbywa się z wykorzystaniem mechanizmów dostępnych w systemie zarządzania bazami danych. Poprawność wykonania poszczególnych zadań potwierdzana jest odpowiednim wpisem do nadrzędnej bazy danych. Schemat proponowanej struktury systemu przedstawia rys. 1.



Rys. 1. Ogólny schemat systemu rozproszonego
Fig. 1. A general schema of the distributed system

Do specyfikacji zarówno całego systemu, jaki i lokalnych sterowników wykorzystywane są hierarchiczne sieci Petriego [5, 7]. Pomimo koncentracji badań na z góry określonej strukturze, istnieje możliwość wprowadzania dodatkowych stopni pośrednich, zarówno dodatkowych baz danych, jak i dedykowanych, w tym

dynamicznie rekonfigurowanych, sprzętowo-programowych elementów sterujących i wykonawczych.

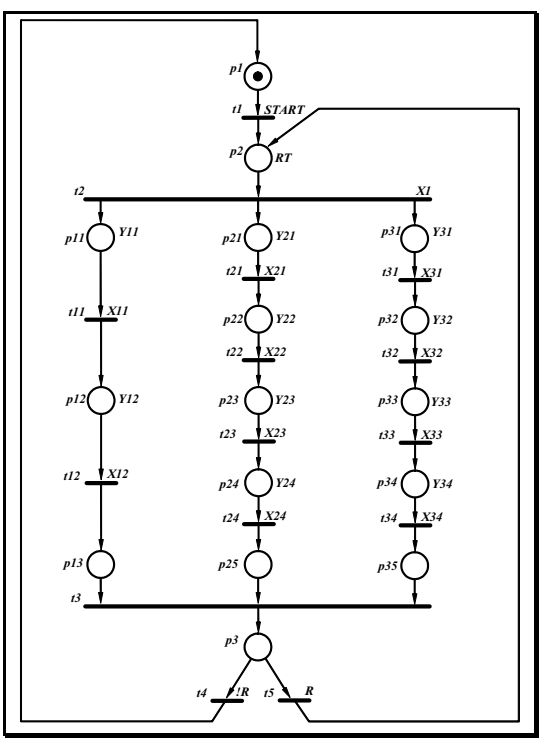
3. Projektowanie sterowników binarnych zorientowane na częściową rekonfigurację

Metoda projektowania rekonfigurowalnych sterowników binarnych z wykorzystaniem układów FPGA została przedstawiona w [3, 7].

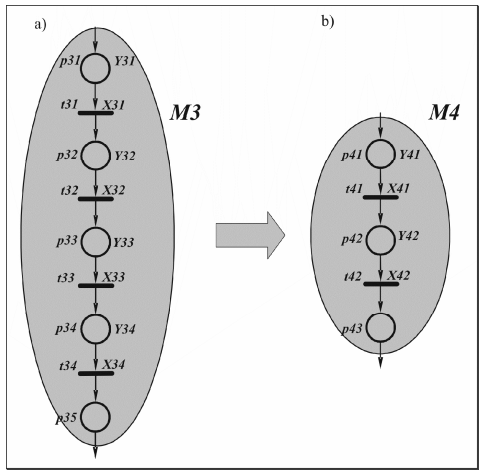
W prezentowanej metodzie do specyfikacji algorytmów sterowania wykorzystywane są sieci Petriego, zarówno płaskie, jak i hierarchiczne. Tak przygotowana sieć może zostać zamodelowana w wybranym języku opisu sprzętu, a następnie poddana syntezie i implementacji w układach FPGA [2]. Alternatywnym sposobem realizacji jest bezpośrednia synteza sieci Petriego opisanej w formacie PNSF2 (Petri Net Specification Format v.2) [7]. W wyniku tej operacji otrzymywana jest lista połączeń w formacie XNF lub EDIF. Realizowane jest to poprzez narzędzia opracowane w Uniwersytecie Zielonogórskim.

Rozszerzenie metody projektowania sterowników pod kątem częściowej rekonfiguracji polega na pokazaniu możliwości wymiany fragmentów sieci, reprezentujących fragmenty procesów sterowania. Niezmienne fragmenty sieci stanowią statyczną część układu reprogramowalnego, natomiast zastępowane moduły definiują dynamiczną część układu.

Proponowana metoda zostanie przedstawiona na przykładzie zapożyczonym z [4]. Jednak ze względów na ułatwienie analizy wprowadzono drobne modyfikacje, polegające między innymi na zmianie nazw poszczególnych elementów sieci i wprowadzenie dodatkowych wejść oraz wyjść. Sieć Petriego dla omawianego układu sterowania pokazano na rys. 2. Tak przygotowany model sieci jest dobrze znany i można przeprowadzić implementację w wybranej technologii. Jednakże modyfikacja algorytmu sterowania polegająca na zastąpieniu podsieci reprezentującej fragment tego procesu sterowania jest utrudniona. Dlatego wprowadzenie makromodułów upraszcza strukturę i interpretację algorytmu. Na rys. 3(a) pokazano podsieć składającą się z miejsc *p31÷p35* oraz tranzycji *t31÷t13*, która tworzy makromiejsce *M3*. W podobny sposób reprezentowane są podsieci *p11÷p13* oraz *p21÷p25* przez makromiejsca *M1* i *M2*. Tak przygotowaną makrosieć pokazano na rys. 4.

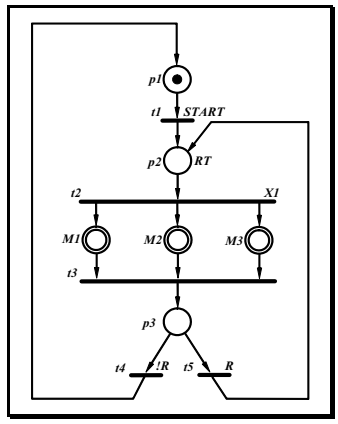


Rys. 2. Sieć Petriego opisująca algorytm sterowania
Fig. 2. A Petri-net-based model of the control system



Rys. 3. Podsieć przed (a) i po (b) modyfikacji
Fig. 3. Subnet before (a) and after (b) modification

Rys. 3 pokazuje dwa alternatywne fragmenty sieci, (a) podstawową wersję algorytmu sterowania, tzn. przed modyfikacją, oraz (b) nowy fragment, po modyfikacji. Prezentowane zmiany są niezbędne w celu wykazania możliwości przygotowania różnych danych konfigurujących.



Rys. 4. Hierarchiczny model sieci Petriego
Fig. 4. A hierarchical Petri net model

W tabeli 1 zamieszczono ogólny opis poszczególnych sygnałów wejściowych i wyjściowych, dla sieci przed modyfikacją oraz po zastąpieniu procesu 3 (makromiejsca *M3*) procesem 4 (*M4*).

Tab. 1. Opis sygnałów wejściowych i wyjściowych
Tab. 1. Description of inputs and outputs

Sygnały wejściowe	***Opis***
START	rozpoczęcie procesu
R	powtarzanie operacji
X1	zakończenie obracania
X11, X12	wejście w procesie 1
X21, X22, X23, X24	wejście w procesie 2
X31, X32, X33, X34	wejście w procesie 3
X41, X42	wejście w procesie 4
Sygnały wyjściowe	***Opis***
RT	obrót elementów
Y11, Y12	wyjście w procesie 1
Y21, Y22, Y23, Y24	wyjście w procesie 2
Y31, Y32, Y33, Y34	wyjście w procesie 3
Y41, Y42	wyjście w procesie 4

Sieć Petriego zapisywana jest w tekstowym zapisie regułowym, tzn. w formacie PNSF2, w którym jawnie określane są makromiejsca sieci (rys. 5). Modyfikacja algorytmu sterowania polegająca na zastąpieniu w programie makromiejsca *M3* makromiejscem *M4* jest wyjątkowo proste i polega zdefiniowaniu nowego makromiejsca (w sekcji .macroplace) oraz modyfikacji makrosieci w sekcji .net części .part Sterowanie.

Następnym krokiem jest konwersja opisu sieci do postaci listy połączeń, zgodnie ze strukturą przedstawioną w pracy [7]. Do kodowania miejsc sieci Petriego zastosowano zmodyfikowaną metodę *one-hot*, tzn. każdy lokalny stan (miejsce sieci) jest reprezentowany przez oddzielny przerzutnik.

```
.clock CLK
.inputs START X1 R X11 X12 X21 X22 X23 X24 X31 X32 X33 X34
.outputs RT Y11 Y12 Y21 Y22 Y23 Y24 Y31 Y32 Y33 Y34

.macroplace MP1(X11 X12, Y11 Y12)
.interface  P11, P13
.places P12
.transitions t11 t12
.net
 t11: P11 * X11 |- P12;
 t12: P12 * X12 |- P13;
.MooreOutputs
 P11  |- Y11;
 P12  |- Y12;
//...

.macroplace MP3(X31 X32 X33 X34, Y31 Y32 Y33 Y34)
.interface  P31, P35
.places P32 P33 P34
.transitions t31 t32 t33 t34
.net
 t31: P31 * X31 |- P32;
 t32: P32 * X32 |- P33;
 t33: P33 * X33 |- P34;
 t34: P34 * X34 |- P35;
.MooreOutputs
 P31 |- Y31;
 P32 |- Y32;
 P33 |- Y33;
 P34 |- Y34;

.part Sterowanie
.places P1 P2 P3 MP1=MP1(X11 X12, Y11 Y12)
.places MP2=MP2(X21 X22 X23 X24, Y21 Y22 Y23 Y24)
.places MP3=MP3(X31 X32 X33 X34, Y31 Y32 Y33 Y34)
.transitions t1 t2 t3 t4 t5
.predicates pred_notR
.net
 t1: P1 * START |- P2;
 t2: P2 * X1 |- MP1 * MP2 * MP3;
 t3: MP1 * MP2 * MP3 |- P3;
 t4: P3 * pred_notR |- P1;
 t5: P3 * R |- P2;
.predicateDescriptions
 pred_notR = !R;
 .MooreOutputs
 P2  |- RT;
.marking P1
```

Rys. 5. Fragment specyfikacji w PNSF2
Fig. 5. A part of PNSF2 model

Implementację omawianego przykładu przeprowadzono z wykorzystaniem układów FPGA serii Spartan2E firmy Xilinx. Przygotowano dane konfigurujące dla obu algorytmów, tzn. zgodnie z siecią podaną na rys. 2, a następnie wprowadzając modyfikację polegającą na zastąpieniu makromiejsca M3 makromiejscem M4. Ponadto wykorzystano podejście płaskie (rys. 2) oraz hierarchiczne (rys. 3 i 4). Analiza wyników w postaci porównania rozmiarów plików konfiguracyjnych podano w tabeli 2. Testy wykazały, że podejście hierarchiczne wykazuje większą różnicę pomiędzy pełną a częściową konfiguracją. Wynika to między innymi z faktu, że przy wersji hierarchicznej układ charakteryzuje się większą modułowością, a co za tym idzie poszczególne moduły są implementowane w blisko siebie położonych blokach logicznych (ang. *configurable logic block, CLB*).

Tab. 2. Porównanie rozmiarów plików z danymi konfiguracyjnymi FPGA
Tab. 2. Comparison of FPGA configuration file sizes

Przykład	*Rozmiar pełnego pliku*	*Rozmiar pliku różnicowego*	*Różnica [%]*
Wersja płaska	177 kB	28 kB	84%
Wersja hierarchiczna	177 kB	25 kB	86%

W większości niezłożonych układów sterowania można zastosować proste rozwiązanie bazujące na odpowiednio wywołanym oprogramowaniu generującym pliki konfigurujące. Jednak w przypadku bardziej rozbudowanych układów należy poszczególne makromoduły (a dokładniej zasoby układu programowalnego) ręcznie umiejscowić za pomocą plików z ograniczeniami projektowymi (pliki typu *UCF*, ang. *user constraints file*) [9]. Odpowiednie metody są uzależnione od stosowanych układów FPGA oraz dostępnych narzędzi CAD wspomagających proces projektowania.

4. Podsumowanie

Zastosowanie sieci Petriego w procesie projektowania sterowników binarnych jest znane od wielu lat. Tą tematyką zajmowało się już wielu autorów. Również wykorzystanie układów logiki programowalnej do realizacji reprogramowalnych układów sterowania nie jest nowym rozwiązaniem, z tego też względu w wielu rzeczywistych rozwiązaniach z powodzeniem są stosowane. Jednak stale zainteresowanie również nowymi obszarami zastosowań odpowiednich układów reprogramowalnych zaowocowało implementacjami w środowiskach, gdzie rozmiar danych rekonfigurujących odgrywa rolę krytyczną. Dlatego coraz szerzej rozwijane są metody projektowania uwzględniające możliwość zmiany konfiguracji układu przy zastosowaniu mniejszych danych zawierających konfigurację tylko zmienianego fragmentu układu.

W prezentowanym artykule zaprezentowano metodę projektowania sterowników binarnych bazującej na specyfikacji algorytmów sterowania z zastosowaniem hierarchicznych sieci Petriego pod kątem częściowej rekonfiguracji. Opracowana metoda uwzględnia możliwość zastąpienia złożonych fragmentów sieci (mikromodułów lub podsieci) innymi w celu dopasowania algorytmu sterowania do realizacji nowych zadań. W przeprowadzonych testach wykorzystano układu FPGA firmy Xilinx. Testy wykazały skuteczność metody przy zachowaniu łatwości modyfikacji algorytmu.

Praca naukowa finansowana ze środków Komitetu Badań Naukowych w latach 2004-2006 jako projekt badawczy (grant nr 3 T11C 046 26).

5. Literatura

[1] Atmel, “Field Programmable Gate Array”, www.atmel.com/products/FPGA, 2006.
[2] M.Adamski, M.Węgrzyn, P.Wolański, “A VHDL based Approach to Logic Controllers Design”, Proceedings of the International Conference - Programmable Devices and Systems, PDS’98, 24-25.02.1998, ss.9-16.
[3] M.Adamski, M.Węgrzyn, “Implementacja współbieżnych algorytmów sterowania w reprogramowalnych sterownikach logicznych”, Pomiary Automatyka Kontrola, 2003, nr 2-3, ss.21-25.
[4] L.Ferrarini, “An Incremental Approach to Logic Controller Design with Petri Nets”, IEEE Transactions on System, Man, and Cybernetics, May/June 1992, Vol.22, No.3, ss.461-474.
[5] L.Gomes, A.Steiger-Garção, “Petri net based hierarchical state machine model for reactive real-time systems”, Proceedings of the 2nd Conference on Automatic Control CONTROL’96, Porto (Portugalia), 11-13.09.1996, ss.709-714.
[6] R.Hartenstein, “Reconfigurable Computing: a New Business Model - and its Impact on SoC Design”, Proceedings of the EUROMICRO Symposium on Digital Systems Design, DSD 2001, ss.103-110.
[7] M.Węgrzyn, M.Adamski, “Hierarchical Approach for Design of Application Specific Logic Controller”, IEEE International Symposium on Industrial Electronics ISIE’99, Bled, Slovenia, 12-16.07.1999, Vol.3, ss.1389-1394.
[8] M.Węgrzyn, „Rozproszony system sterowania bezpiecznego z rekonfigurowalnymi układami lokalnymi”, Materiały II Konferencji Naukowej - Informatyka - sztuka czy rzemiosło, KNWS 2005, Złotniki Lubańskie, Polska, 2005, Oficyna Wydaw. Uniwersytetu Zielonogórskiego, Zielona Góra, 2005, ss.127-132.
[9] Xilinx, “XAPP290, Two Flows for Partial Reconfiguration: Module-Based or Difference-Based”, www.xilinx.com, 2004.

Artykuł recenzowany