Agnieszka WĘGRZYN, Marek WĘGRZYN
UNIWERSYTET ZIELONOGÓRSKI

Implementacja sieci Petriego sterowania w częściowo rekonfigurowanych układach FPGA

Dr inż. Agnieszka WĘGRZYN

Adiunkt w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Zainteresowania naukowe obejmują: modelowanie i weryfikację układów sterowania opisanych sieciami Petriego, formalne metody specyfikacji algorytmów sterowania, bazy danych, projektowanie systemów informacyjnych oraz aplikacje internetowe. Członek PTI, IEEE oraz IFAC.

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

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). Członek PTI, POLSPAR, IEEE oraz IFAC (przewodniczący komitetu TC 3.1).

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

Streszczenie

Artykuł przedstawia zastosowanie dekompozycji równoległej sieci Petriego do celów projektowania częściowo rekonfigurowanych sterowników logicznych. Do dekompozycji sieci Petriego zastosowano metody symboliczne bazujące na analizie wybranych właściwości sieci i wyznaczaniu P-niezmienników. Otrzymane w ten sposób połączone maszyny stanów są modelowane w wybranym języku opisu sprzętu. W artykule zaprezentowano modele w języku Verilog. Do implementacji układowej wykorzystywane są układy FPGA (firmy Xilinx). Wymiana wybranej składowej maszyny stanów, podczas powtórnej implementacji i porównaniu danych konfiguracyjnych, umożliwia zmianę konfiguracji tylko wybranego fragmentu projektu plikiem różnicowym.

Słowa kluczowe: sterownik logiczny, sieci Petriego, dekompozycja, częściowa rekonfiguracja, FPGA.

Control Petri Net Implementation into Partial Reconfigurable FPGA

Abstract

In the paper Petri-net decomposition based design of Logic Controller for partial reconfiguration is presented. Symbolic methods of analysis of some Petri net properties and P-invariants calculation is applied for decomposition. Obtained linked state machines are modeled using Hardware Description Languages. Verilog models of such decomposed Petri net are presented. Xilinx FPGA devices are used for final implementation. Replacement of selected state machine, after next implementation and bit-streams' comparison, provides exchange of configuration only selected part of project using differential bit-stream.

Keywords: Logic Controller, Petri net, decomposition, partial reconfiguration, FPGA.

1. Wstęp

Układy programowalne, szczególnie układy FPGA, wykorzystywane są nie tylko do implementacji układów realizujących złożone algorytmy DSP (ang. *Digital Signal Processing*) [12,14], ale również jako rekonfigurowalne układy sterowania [1,7,11,17,20]. Układy FPGA charakteryzują się możliwością dynamicznej, częściowej rekonfiguracji [5,6,8,10].

Większość prac dotyczących metod projektowania zorientowanych na częściową rekonfigurację ukierunkowana jest na projektowanie rekonfigurowalnych systemów obliczeniowych [9,12,13,14]. Projektowanie sterowników logicznych i części sterującej systemu cyfrowego stanowi nieliczną grupę badań [17], które nie zaspakajają potrzeb projektantów na tego typu metody. Prezentowany artykuł przedstawia zastosowania dekompozycji sieci Petriego do celów projektowania tej grupy układów. Wcześniejsze prace [21,22,23] przedstawiały jedynie ogólną zasadę, aktualnie opracowana metoda stanowi integralną cześć rozwijanego akademickiego systemu CAD – PeNLogic (wcześniejsza wersja PeNCAD) [19].

2. Częściowa rekonfiguracja układów FPGA

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. Xilinx [10]. 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 [22].

3. Specyfikacja układów sterowania

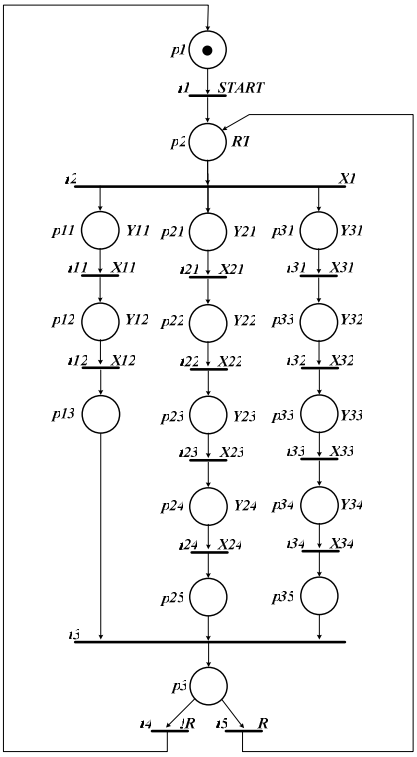
Układy współbieżne stanowią ważną grupę układów cyfrowych, które są coraz częściej wykorzystywane ze względu na skutek postępu technologicznego w dziedzinie elementów scalonych typu ASIC. Dużego znaczenia nabiera również konstruowanie równolegle działających części układu operacyjnego i związanego z nim sterownika współbieżnego. Do opisu układów współbieżnych można zastosować różne sposoby modelowania, np. języki opisu sprzętu (ang. *Hardware Description Language, HDL*), grafy stanów, sieci działań, diagramy przejść, itp. [1,3,4].

Jednakże, ze względu na możliwość łatwej reprezentacji współbieżności oraz ze względu na dobrze zdefiniowane pojęcia, sieci Petriego najlepiej nadają się do modelowania układów sterowania dyskretnego. Opisanie za ich pomocą działania układu sterującego sekwencją czynności, wykonywanych współbieżnie jest znacznie prostsze, niż opis funkcjonowania tego samego układu za pomocą innych metod. Ponadto, układy opisane sieciami Petriego, dzięki rozbudowanemu aparatowi matematycznemu i dużemu zestawowi metod analizy, mogą być weryfikowane w sposób formalny [16].

Sieci Petriego mogą być reprezentowane w sposób graficzny (graf dwudzielny) lub tekstowy. Miejsca sieci reprezentują stany lokalne sterownika, tranzycje ich zmiany. Wszystkie miejsca oznakowane w tym samym czasie, definiują stan globalny sterownika. Znaczniki wyznaczają, które stany sterownika są aktywne w danym czasie. Oznakowanie początkowe reprezentuje stan początkowy układu. Każda tranzycja typu *rozwidlenie*, tzn. tranzycja posiadająca więcej niż jedno miejsce wyjściowe, jest początkowym punktem dla równoległych procesów, natomiast każda tranzycja typu *złączenie*, tzn. tranzycja posiadająca więcej niż

jedno miejsce wejściowe, synchronizuje procesy, które łączą się w tym punkcie. Ponadto w przypadku dużych projektowanych systemów sterowania należy wykorzystywać hierarchiczne sieci Petriego, które w znaczny sposób ułatwiają proces weryfikacji sieci oraz pozwalają na wykorzystanie tych samych elementów kilkukrotnie [18,20].

Rys. 1 przedstawia przykładową sieć Petriego opisującą układ sterowania stanowiskiem do wiercenia, którego początkowa wersja została przestawiona w [7]. Sterownik na podstawie stanu wejść (np. *START*, *X1*, *R*), występujących jako warunki przy tranzycjach, zmienia swój stan (np. *p1*, *p2*, *p3*) oraz generuje właściwą wartość na wyjściach (np. *RT*, *Y11*, *Y12*).



Rys. 1. Przykład sieci Petriego
Fig. 1. Example of Petri net

Graficzna postać sieci Petriego jest bardzo wygodna i wyraźnie prezentuje procesy współbieżne. Jednak do dalszej analizy wykorzystywana jest równoważna postać tekstowa. Przykładem takiego opisu jest format PNSF2 [20] (Rys. 2).

```
.clock CLK
.inputs START X1 R X11 X12 X21 ...
.outputs RT Y11 Y12 Y21 Y22 ...

.part Sterownik
.places p1 p2 p3 p11 p12 p13 p21 ...
.transitions t1 t2 t3 t4 t5 t11 ...

.net
 t1: p1 * START  |- p2;
 t2: p2 * X1 |- p11 * p21 * p31;
 t3: p13 * p25 * p35 |- p3;
 t4: p3 * !R |- p1;
 t5: p3 *  R |- p2;
 ...

.MooreOutputs
 p2  |- RT;
 p11 |- Y11;
 p12 |- Y12;
 ...

.marking p1
.end
```

Rys. 2. Specyfikacja sieci Petriego w formacie PNSF2 (fragment)
Fig. 2. PNSF2 specification of Petri net (a part)

Weryfikację układu sterowania opisanego siecią Petriego można sprowadzić do badania pewnych własności sieci Petriego, takich jak żywotność i ograniczoność. Cechy te zapewniają odpowiednio, że w układzie nie dojdzie do zapętlenia procesów lub ich zatrzymania.

4. Dekompozycja sieci Petriego

Sieć Petriego pozwala w precyzyjny i jasny sposób przedstawić procesy współbieżne i koordynację między nimi, natomiast nie dokumentuje w sposób wystarczający intencji projektanta [20]. Procesy, rozpatrywane przez projektanta układu, nie znajdują właściwego odzwierciedlenia w modelującej jej sieci P/T [16], gdyż więzi pomiędzy poszczególnymi stanami lokalnymi, należącymi do tego samego wyróżnionego procesu, ulegają zatarciu. Z tego względu często konieczne jest ich odtworzenie w sposób formalny, jako tak zwanych P-niezmienników [16,18], czyli fragmentów sieci o charakterze sekwencyjnym, zbliżonym do klasycznego automatu cyfrowego. Często wtórnie odtworzony i wykorzystywany proces nie odzwierciedla intencji projektanta, choć formalnie jest poprawny. Tego rodzaju sytuacja służy jako argument dla zwolenników powiązanych grafów automatowych (np. [3]), którzy uważają, że tylko one w sposób jasny i rzeczywisty odzwierciedlają intencje projektanta.

Dekompozycja automatu współbieżnego na sekwencyjne automaty składowe wykorzystywana jest często w celu ułatwienia procesu syntezy układu cyfrowego. W przypadku zamodelowania takiego układu siecią Petriego, problem dekompozycji sprowadza się do wyodrębnienia podsieci typu automatowego, czyli podsieci zawierających tylko jeden znacznik [3,23].

Wykorzystując algorytm weryfikacji sieci Petriego bazujący na wyznaczaniu wszystkich blokad i pułapek w sieci, można sprawdzić żywotność oraz ograniczoność sieci. Metoda analizy został szerzej opisany w [18] i bazuje na symbolicznych metodach [15] badania równań opisujących sieć. Zbiory miejsc, które są jednocześnie blokadami i pułapkami związane są z P-niezmiennikami. Jeżeli wyznaczone P-niezmienniki pokrywają całą sieć, wówczas można określić, że sieć jest ograniczona. Wyznaczone P-niezmienniki dekomponują sieci Petriego na składowe automatowe (SM-komponenty, ang. *state machine*). Każda składowa automatowa reprezentowana jest przez jeden P-niezmiennik.

W celu dekompozycji należy pokolorować każdy z automatów oddzielnym kolorem [20]. Liczba kolorów, jakimi należy pokolorować sieć zależy od liczby wyznaczonych P-niezmienników sieci. Pokolorowana sieć Petriego, oprócz informacji o współbieżności i sekwencyjności zdarzeń, niesie informację o przynależności miejsc do poszczególnych procesów sekwencyjnych. Przedstawione podejście pozwala również na kodowanie miejsc sieci, a tym samym stanów lokalnych automatu współbieżnego [1,11]. Odpowiednie kodowanie jest zasadniczym środkiem umożliwiającym bezpośrednią implementację sieci Petriego w programowalnych strukturach logicznych.

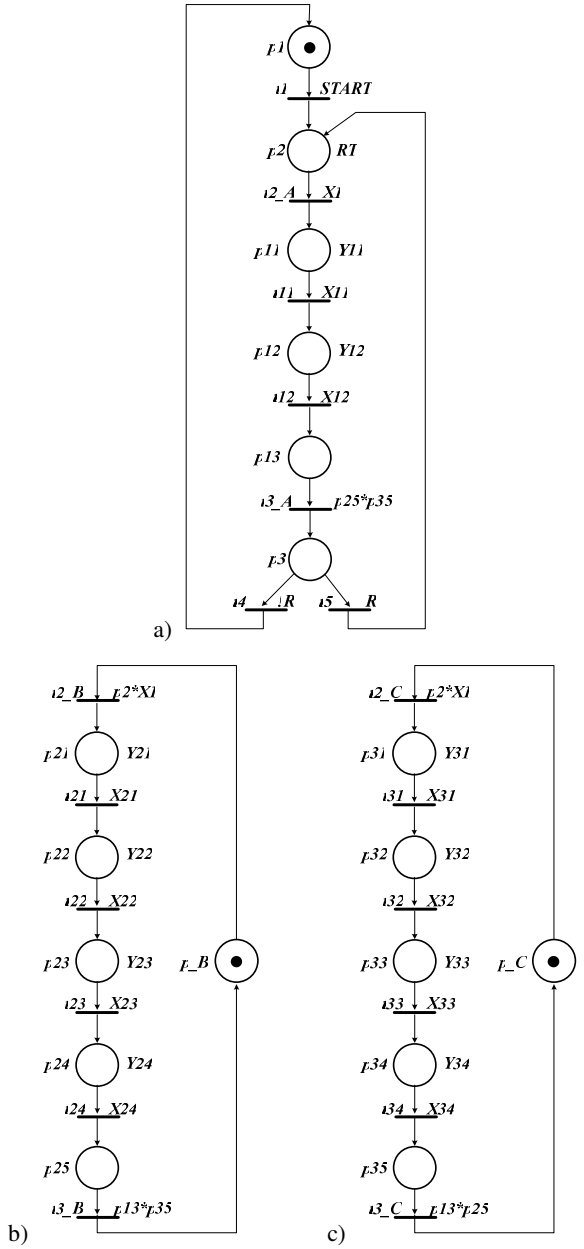
Miejsca stanowiące wspólną część w kilku P-niezmiennikach pozostają tylko w jednej wybranej SM-sieci, natomiast do pozostałych wstawiane są *miejsca spoczynkowe*. Analizując rozpatrywany przykład (Rys. 1), tranzycje *t2* i *t3* stanowią węzły graniczne dla wyodrębnianych SM-komponentów. Miejsca *p1*, *p2* i *p3* zostają połączone z miejscami *p11*, *p12* i *p13* tworząc automat *SM_A* (Rys. 3a). Do automatów *SM_B* (Rys. 3b) oraz *SM_C* (Rys. 3c) dodano miejsca spoczynkowe, odpowiednio, *p_B* i *p_C*.

Rozłożenie sieci na składowe automatowe umożliwia łatwą, częściową rekonfigurację układu sterownika. Modularna struktura pozwala w sposób systematyczny modyfikować układ, zmieniając tylko jego wyróżnioną część równoległa, bez naruszania reguł kolorowania. Dzięki temu, zmieniany jest tylko jeden wybrany proces – automat składowy, a układ nie wymaga dodatkowej analizy.

5. Implementacja sterownika w strukturach FPGA

Układ sterowania opisany grupą wszystkich (w rozpatrywanym przykładzie trzech) połączonych SM-sieci jest implementowany wykorzystując odpowiednio przygotowane modele tych komponentów w wybranym języku opisu sprzętu. W prezentowanym podejściu wybrano język Verilog. Na kolejnych rysunkach poka-

zano modele poszczególnych modułów. Rys. 4 przedstawia model modułu *SM_A*, natomiast rys. 5 – modułu *SM_B*. Model komponentu *SM_C* jest analogiczny, jak dla *SM_B*, dlatego nie zamieszczono jego model.



Rys. 3. Zdekomponowana sieć Petriego z rys. 1
Fig. 3. Decomposed Petri net from Fig. 1

Modele SM-komponentów przygotowano wykorzystując ogólną metodę modelowania sieci Petriego [20]. Do opisu każdego stanu lokalnego zastosowano oddzielny proces 'always', konsekwencją czego jest zastosowanie metody *gorąca jedynka* (ang. *one hot*) kodowania miejsc sieci (stanów lokalnych). Ponieważ w zdekomponowanych sieciach występują tylko procesy sekwencyjne, dlatego możliwe jest zastosowanie również metod modelowania opracowanych dla klasycznych automatów cyfrowych, których przegląd został przedstawiony w pracy [4].

Do symulacji w języku Verilog wszystkie trzy modele reprezentujące SM-komponenty mogą być traktowane jako moduły nadrzędne (ang. *top-level*). W prezentowanym przykładzie zdecydowano się na przygotowanie jednego modułu nadrzędnego, w którym umieszczono instancje modeli poszczególnych sieci składowych (Rys. 6). Taki model reprezentuje cały sterownik i może być również wykorzystany do celów syntezy.

```
module wiercenie_SM_A (Clk, Reset, START, X1, X11, X12, p25,
                       p35, R, RT, Y11, Y12, p2, p13);
input Clk, Reset;
input START, X1, X11, X12, p25, p35, R;
output RT, Y11, Y12, p2, p13;

reg p1, p2, p3, p11, p12, p13;

assign RT  = p2 ;
assign Y11 = p11 ;
assign Y12 = p12 ;

wire t1, t2_A, t3_A, t4, t5, t11, t12;
assign t1 = p1 & START;
assign t2_A = p2 & X1;
assign t3_A = p13 & p25 & p35;
assign t4 = p3 & ~R;
assign t5 = p3 & R;
assign t11 = p11 & X11;
assign t12 = p12 & X12;

always @(posedge Clk)
        if (Reset) p1 <= 1'b1; else p1 <= t4 | (p1 & ~t1);
always @(posedge Clk)
        if (Reset) p2 <= 1'b0; else p2 <= (t1 | t5) | (p2 & ~t2_A);
always @(posedge Clk)
        if (Reset) p11 <= 1'b0; else p11 <= t2_A | (p11 & ~t11);
always @(posedge Clk)
        if (Reset) p12 <= 1'b0; else p12 <= t11 | (p12 & ~t12);
always @(posedge Clk)
        if (Reset) p13 <= 1'b0; else p13 <= t12 | (p13 & ~t3_A);
always @(posedge Clk)
        if (Reset) p3 <= 1'b0; else p3 <= t3_A | (p3 & ~(t4 | t5));

endmodule
```

Rys. 4. Model SM-sieci A
Fig. 4. Verilog model of SM-net A

```
module wiercenie_SM_B (Clk, Reset, X1, X21, X22, X23, X24,
                       p2, p13, p35, Y21, Y22, Y23, Y24, p25);
input Clk, Reset;
input X1, X21, X22, X23, X24, p2, p13, p35;
output Y21, Y22, Y23, Y24, p25;

reg p21, p22, p23, p24, p25, p_B;

assign Y21 = p21 ;
assign Y22 = p22 ;
assign Y23 = p23 ;
assign Y24 = p24 ;

wire t2_B, t3_B, t21, t22, t23, t24;
assign t2_B = p2 & X1;
assign t3_B = p25 & p13 & p35;
assign t21  = p21 & X21;
assign t22  = p22 & X22;
assign t23  = p23 & X23;
assign t24  = p24 & X24;

always @(posedge Clk)
        if (Reset) p_B <= 1'b1; else p_B <= t3_B | (p_B & ~t2_B);
always @(posedge Clk)
        if (Reset) p21 <= 1'b0; else p21 <= t2_B | (p21 & ~t21);
always @(posedge Clk)
        if (Reset) p22 <= 1'b0; else p22 <= t21 | (p22 & ~t22);
always @(posedge Clk)
        if (Reset) p23 <= 1'b0; else p23 <= t22 | (p23 & ~t23);
always @(posedge Clk)
        if (Reset) p24 <= 1'b0; else p24 <= t23 | (p24 & ~t24);
always @(posedge Clk)
        if (Reset) p25 <= 1'b0; else p25 <= t24 | (p25 & ~t3_B);

endmodule
```

Rys. 5. Model SM-sieci B
Fig. 5. Verilog model of SM-net B

```
module PN_SM(Clk, Reset, START, R, X1, X11, X12, X21, X22, X23, X24,
             X31, X32, X33, X34, RT, Y11, Y12, Y21, Y22, Y23, Y24,
             Y31, Y32, Y33, Y34);

input Clk, Reset, START, R, X1;
input X11, X12, X21, X22, X23, X24, X31, X32, X33, X34;
output RT, Y11, Y12, Y21, Y22, Y23, Y24, Y31, Y32, Y33, Y34;

wire p2, p13, p25, p35;

wiercenie_SM_A SMA (Clk, Reset, START, X1, X11, X12,
                    p25, p35, R, RT, Y11, Y12, p2, p13);
wiercenie_SM_B SMB (Clk, Reset, X1, X21, X22, X23, X24,
                    p2, p13, p35, Y21, Y22, Y23, Y24, p25);
wiercenie_SM_C SMC (Clk, Reset, X1, X31, X32, X33, X34,
                    p2, p13, p25, Y31, Y32, Y33, Y34, p35);

endmodule
```

Rys. 6. Model nadrzędny z instancjami trzech składowych SM-sieci
Fig. 6. Verilog top-level model with 3 SM-nets

Syntezę przeprowadzono zarówno dla wszystkich SM-komponentów, jak również dla kompletnego sterownika. Wyniki syntezy zamieszczono w tabeli 1. Dla porównania przeprowadzono również implementację sterownika na podstawie pierwotnej (współbieżnej) sieci, tzn. sieci przed dekompozycją (Rys. 1). Różnice dla rozpatrywanego przykładu są do pominięcia.

Tabela 1. Wyniki syntezy (Xilinx FPGA – XC3S250epq208-5)
Table 1. Synthesis summary (Xilinx FPGA – XC3S250epq208-5)

Model	*Liczba bloków (slice)*	*Liczba tablic LUT4*	*Liczba przerzutników (FF)*
Moduł SM_A	4	8	6
Moduł SM_B	3	6	5
Moduł SM_C	3	6	5
Sterownik (realizacja z SM-sieci)	10	18	16
Sterownik (realizacja współbieżna)	10	17	16

Należy podkreślić, że liczba wykorzystanych przerzutników przy realizacji modułów *SM_B* i *SM_C* są o jeden mniejsza (tzn. 5) niż łączna liczba występujących stanów, pomimo wyboru metody kodowania *one hot*. Wynika to z faktu, że z wprowadzonym stanem spoczynkowym nie jest skojarzone żadne wyjście oraz, że do ustawienia pozostałych stanów wymagane są tylko dane zewnętrzne (odpowiednio dla stanów *p21* i *p31*) lub pozostałe stany występujące w podstawowej sieci (*p22*, …, *p25* oraz *p32*, …, *p35*). Stan spoczynkowy występuje, gdy wszystkie przerzutniki reprezentujące pozostałe stany są równe '0'.

W celu wymiany wybranego modułu SM-sieci, np. *SM_C*, przeprowadzana jest ponowna implementacja tego sterownika z tym zamienionym modułem. Otrzymany nowy plik konfiguracyjny zostaje porównany z plikiem konfiguracyjnym podstawowego sterownika wykorzystując standardowe oprogramowanie dostarczane przez producenta układów FPGA. W wyniku porównania zostaje wygenerowany plik różnicowy, którego rozmiar jest znacznie mniejszy niż pełny plik konfiguracyjny [5,21].

6. Wnioski

Nowoczesne układy programowalne dostarczają nowych możliwości przy realizacji złożonych systemów cyfrowych. Wymaga to jednak ciągłego rozwijania metod projektowania, aby sprostać nowym wymaganiom. Częściowa rekonfiguracja układów FPGA umożliwia szybką zmianę funkcjonalności rozpatrywanego systemu cyfrowego poprzez wymianę danych konfiguracyjnych jedynie fragmentu układu. W artykule zaproponowano zastosowanie znanej metody dekompozycji sieci Petriego do przygotowania projektu uwzględniającego potrzeby częściowej rekonfiguracji.

Układ sterowania opisany jest siecią Petriego. Dekompozycja specyfikacji współbieżnej na grupę połączonych ze sobą automatów stanów wprowadza rozbicie na częściowo niezależne moduły, wymagające jedynie okresowej synchronizacji, które mogą zostać umieszczone w określonych fragmentach struktury FPGA. Wprowadzenie zmian tylko do wybranego modułu nie pociąga za sobą potrzeby wymiany całej konfiguracji, a jedynie tego zmienianego fragmentu. Wszystkie otrzymane po dekompozycji SM-komponenty są modelowane w językach opisu sprzętu, np. Verilog, a następnie implementowane w wybranych układzie FPGA. Porównując, przy wykorzystaniu standardowego oprogramowania, dane konfigurujące przed i po zmianie otrzymywane są dane różnicowe, za pomocą których modyfikuje się początkową strukturę układu.

Zaprezentowana metoda stanowi integralną cześć rozwijanego akademickiego systemu CAD – PeNLogic, przeznaczonego do projektowania rekonfigurowalnych układów sterowania oraz części sterującej układów cyfrowych.

7. Literatura

[1] M.Adamski: Parallel Controller Implementation using Standard PLD Software. w: W.R.Moore, W.Luk (Eds), FPGAs, The Oxford 1991 International Workshop on Field Programmable Logic and Applications, Abingdon EE&CS, Abingdon (UK), 1991, ss.296-304.

[2] M.Auguin, F.Boeri, C.Andre: New Design Using PLAs and Petri Nets. Proceedings of the International Symposium on Measurement and Control, Athens, ss.55-68, 1978.

[3] H.Belhadj, L.Gerbaux, M.-C.Bertrand, G.Saucier: Specification and Synthesis of Communicating Finite State machines. Synthesis for Control Dominated Circuits, (A-22), Elsevier Science Publishers B.V. (North Holland), 1993, ss.91-101.

[4] S.Chmielewski, M.Węgrzyn: Modelling and synthesis of automata in HDLs. Proceedings of SPIE, Vol. 6347, Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2006; Ryszard S. Romaniuk (Red.), 2006, ss. 63470J1-13.

[5] M.Doligalski, M.Węgrzyn: Metody częściowej rekonfiguracji układów FPGA. Przegląd Telekomunikacyjny i Wiadomości Telekomunikacyjne, 2008, nr 6, ss.773-775 [CD-ROM].

[6] E.Eto: XAPP290: Difference-Based Partial Reconfiguration. v.2.0, www.xilinx.com, 2007.

[7] L.Ferrarini, An Incremental Approach to Logic Controller Design with Petri Nets. IEEE Transactions on System, Man, and Cybernetics, Vol.22, No.3, ss.461-474, 1992.

[8] 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.

[9] S.Jung, T.G.Kim: Configuration Sharing to Reduce Reconfiguration Overhead Using Static Partial Reconfiguration. IEICE Transaction on Information & Systems, 2008; Vol.E91-D, No.11, ss.2675-2684, DOI: 10.1093/ietisy/e91-d.11.2675.

[10] C.Kao: Benefits of Partial Reconfiguration. XCell Journal, Vol.55, No.4, 2005, ss. 65-67.

[11] T.Kozłowski, E.L.Dagless, J.M.Saul, M.Adamski, J.Szajna: Parallel controller synthesis using Petri nets. IEE Proceedings-E, Computers and Digital Techniques, Vol.142, No.4, July 1995, ss.263-271.

[12] Ph.Manet, et.al.: An Evaluation of Dynamic Partial Reconfiguration for Signal and Image Processing in Professional Electronics Applications. EURASIP Journal on Embedded Systems, Hindawi Publishing, Vol.2008, Article ID 367860, DOI: 10.1155/2008/367860.

[13] D.Mesquita, F.Moraes, J.Palma, L.Moller, N.Calazans: Remote and partial reconfiguration of FPGAs: tools and trends. Proceedings of the Parallel and Distributed Processing Symposium, 22-26.04.2003. DOI: 10.1109/IPDPS.2003.1213326.

[14] Y.-J.Oh, Ch.-S.Choi, H.Lee, C.-H.Lee: A Reconfigurable CSD FIR Filter Design using Dynamic Partial Reconfiguration. International SoC Design Conference, ISOCC, Seoul, Oct. 2005, ss.381-384.

[15] M.Minoux, K.Barkaoui: Deadlocks and traps in Petri nets as a Horn-satisfability solutions and some related polynomially solvable problems. Discrete Applied Mathematics, Elsevier Science Publishers (North Holland), Vol.29, 1990, ss.195-210.

[16] T.Murata: Petri Nets: Properties, Analysis and Applications. Proceedings of the IEEE, Vol.77, No.4, April 1989, ss.541-580.

[17] P.S.B.Nascimento, P.R.M.Maciel, M.E.Lima, R.E.Santana, A.G.S.Filho: A Partial Reconfigurable Architecture for Controllers based on Petri Nets. Proc. of the SBCCI 2004, pp.16-21.

[18] A. Wegrzyn: Symboliczna analiza układów sterowania z wykorzystaniem wybranych metod analizy sieci Petriego. Oficyna Wydawnicza Uniwersytetu Zielonogórskiego, Zielona Góra, 2003.

[19] A.Węgrzyn, M.Węgrzyn: PeNCAD System – From Modeling to Synthesis of Concurrent Controllers. The 5th IEEE East-West Design & Test International Symposium, Erywań (Armenia), 2007, ss. 384-389.

[20] M.Węgrzyn: Hierarchiczna implementacja współbieżnych kontrolerów cyfrowych z wykorzystaniem FPGA. Rozprawa doktorska, Politechnika Warszawska, Wydział Elektroniki i Technik Informacyjnych, Warszawa, 1998.

[21] M.Wegrzyn: Petri Net Decomposition Approach for Partial Reconfiguration of Logic Controllers. The 3rd IFAC Workshop on Discrete-Event System Design, DESDes'06, Rydzyna (Polska), 26-28.09.2006, ss. 323-328.

[22] M.Węgrzyn: Częściowa rekonfiguracja sterowników binarnych opisanych sieciami Petriego. Pomiary-Automatyka-Kontrola, Nr 6 (bis), 2006, ss.26-28.

[23] M.Węgrzyn, A.Węgrzyn: Implementation of concurrent logic controllers based on decomposition into state machine components. Radioelektronika & Informatika, 2006, No.3, ss.44-47.

Artykuł recenzowany