Arkadiusz BUKOWIEC
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI,
Licelana 9, 65-417 Zielona Góra

Rozproszony system sterowania o architekturze GALS projektowany z wykorzystaniem sieci Petriego

Dr inż. Arkadiusz BUKOWIEC

Ukończył studia inżynierskie (2001) na Politechnice Zielonogórskiej a następnie studia magisterskie (2004) o specjalności inżynieria komputerowa na Uniwersytecie Zielonogórskim. W 2008 roku obronił z wyróżnieniem rozprawę doktorską na Wydziale Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego w dziedzinie informatyka. Od roku 2004 pracuje w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego, na stanowisku adiunkta.

e-mail: a.bukowiec@iie.uz.zgora.pl

Streszczenie

W artykule omówiona została architektura rozproszonego systemu sterowania zbudowanego z konfigurowalnych struktur FPGA. System specyfikowany jest z wykorzystaniem sieci Petriego. Następnie poddawany jest dekompozycji na składowe automatowe z wykorzystaniem algorytmów kolorowania sieci Petriego. Każda składowa implementowana jest niezależnie w oddzielnym układzie FPGA. Aby umożliwić komunikację pomiędzy poszczególnymi składowymi zastosowano architekturę globalnie asynchroniczną lokalnie synchroniczną (GALS). Każda podsieć synchronizowana jest lokalnym sygnałem zegarowym. Komunikacja pomiędzy poszczególnymi podsieciami zrealizowana jest asynchronicznie z wykorzystaniem dodatkowych sygnałów.

Słowa kluczowe: dekompozycja, FPGA, sieć Petriego, sterownik logiczny, synteza logiczna.

Distributed control system with GALS architecture with use of Petri net

Abstract

The paper presents a new architecture of the distributed specific control system built with FPGA devices. The control algorithm specification is made with use of the control interpreted Petri net. It allows specifying parallel processes in easy way. Next, such a Petri net is decomposed into a set of state-machine type subnets. For this purpose there are applied algorithms of coloring of Petri nets. In this case, each subnet represents one parallel process. Each subnet is independently implemented in different FPGA device. To ensure communication between all subnets, there is used globally asynchronous locally synchronous (GALS) architecture of the whole control system. Each subnet is synchronized by a local clock signal. The global communication between components is buffer-based via additional signals. These signals are generated in particular subnets and they are distributed to other ones. During the synthesis process places of each state-machine subnet are encoded by a minimal-length binary vector. This encoding allows a realization of a microoperation decoder with use of embedded memory blocks of the FPGA device. It leads to balanced usage of all kinds of logic resources of the FPGA device.

Keywords: decomposition, FPGA, Petri net, logic controller, logic synthesis.

1. Wstęp

Systemy cyfrowe realizowane są jako połączenie ścieżki danych (bloków funkcjonalnych) oraz ścieżki sterowania (jednostki sterującej) [1, 7]. Podział taki jest przydatny nie tylko ze względu na wygodę projektowania, ale również ze względu na możliwość wielokrotnego wykorzystania wcześniej zaprojektowanych modułów wykonawczych. Tak widziany system wymaga jednak zaprojektowania dedykowanej jednostki sterującej. Istnieje wiele metod specyfikacji takich jednostek, jednak najbardziej popularne jest stosowanie skończonych automatów stanów [2] albo sieci Petriego [3, 6]. Przewagę jaką dają sieci Petriego jest łatwość modelowania procesów równoległych.

Zastosowanie nowych metod syntezy blokowej w procesie projektowania rozproszonej jednostki sterującej umożliwi wymianę tylko pewnych bloków układu w sytuacji, gdy nastąpi modyfikacja algorytmu sterowania. Nowoczesne układy FPGA są wyposażone w osadzone bloki pamięci, które również można wykorzystać do implementacji logiki układu sterującego. Zastosowanie dekompozycji blokowej daje możliwość podziału układu cyfrowego jednostki sterującej na podzespoły, które w zależności od swojej funkcji mogą zostać łatwo zaimplementowane z wykorzystaniem elementów logicznych różnego typu, pracujących z różnymi szybkościami.

Jako metoda specyfikacji wykorzystywana jest hierarchiczna sieć Petriego typu PT. Umożliwia to specyfikację sterowania hierarchicznymi procesami równoległymi.

W celu rozproszenia systemu sterowania zastosowana jest specyfikacja algorytmu sterowania z wykorzystaniem zbioru połączonych sieci Petriego. Dekompozycji na składowe podsieci dokonuje się z wykorzystaniem algorytmów kolorowania sieci Petriego [5, 9, 10]. W efekcie, każdy kolor reprezentuje jedną podsieć typu automatowego. Aby umożliwić komunikację pomiędzy poszczególnymi podsieciami sterowania zastosowano architekturę globalnie asynchroniczną lokalnie synchroniczną (GALS). Każda podsieć synchronizowana jest lokalnym sygnałem zegarowym. Komunikacja pomiędzy poszczególnymi podsieciami zrealizowana jest asynchronicznie z wykorzystaniem dodatkowych sygnałów.

Każda podsieć typu automatowego może zostać poddana syntezie i implementacji do układu FPGA z wykorzystaniem innych metod syntezy blokowej [4]. W rozwiązaniu takim układ logiczny dekomponowany jest na trzy składowe: układ kombinacyjny, rejestr i dekoder mikrooperacji. Układ kombinacyjny odpowiedzialny jest za realizację równań logicznych warunków odpalenia tranzycji. Rejestr odpowiedzialny jest za zapamiętanie aktualnego globalnego stanu rozpatrywanej podsieci. W opracowanym podejściu stosowane jest kodowanie z wykorzystaniem minimalnej ilości bitów. Dekoder mikrooperacji odpowiedzialny jest za generowanie sygnałów wyjściowych na podstawie aktualnego stanu. Ponieważ kod stanu reprezentowany jest na minimalnej ilości bitów możliwe jest zastosowanie osadzonych bloków pamięci do jego realizacji. Ponieważ typowe osadzone bloki pamięci w układach FPGA są synchroniczne, dekoder ten pełni również funkcję rejestru wyjściowego, co zwiększa stabilność działania jednostki sterującej.

2. Sieć Petriego

Prosta sieć Petriego zdefiniowana jest, jako [8, 6]:

$$PN = (P, T, F), \tag{1}$$

gdzie:

- P - jest skończonym niepustym zbiorem miejsc,
- T - jest skończonym niepustym zbiorem tranzycji,
- F - jest zbiorem łuków z miejsc do tranzycji i z tranzycji do miejsc:

$$F \subseteq (P \times T) \cup (T \times P),$$

$$P \cap T = \emptyset.$$

Zbiory tranzycji wejściowych i wyjściowych dla miejsca $p \in P$ są zdefiniowane następująco:

$$\cdot p = \{t \in T: (t,p) \in F\},$$

$$p \cdot = \{t \in T: (p,t) \in F\}.$$

Zbiory miejsc wejściowych i wyjściowych dla tranzycji $t \in T$ są zdefiniowane następująco:

$$\cdot t = \{p \in P: (p,t) \in F\},$$

$$t \cdot = \{p \in P: (t,p) \in F\}.$$

Znakowanie sieci Petriego określone jest funkcją:

$$M: P \to \mathbb{N}.$$

Opisuje ono liczbę żetonów $M(p)$ umieszczonych w miejscu p. Kiedy miejsce zawiera żeton, mówi się, że jest ono oznakowane. Tranzycja t może zostać odpalona wtedy i tylko wtedy, gdy wszystkie jej miejsca wejściowe są oznakowane. Odpalenie tranzycji usuwa jeden żeton ze wszystkich jej miejsc wejściowych i umieszcza po jednym żetonie w każdym jej miejscu wyjściowym. Sieć Petriego może mieć zdefiniowane znakowanie początkowe M_0 i wtedy definiuje się ją jako:

$$PN = (P, T, F, M_0). \quad (2)$$

Interpretowana sieć Petriego stanowi rozszerzenie sieci Petriego o mechanizm do wymiany informacji [6]. Wymiana ta wykonywana jest przy użyciu sygnałów binarnych. Można wyróżnić interpretowane sieci Petriego typu Moore'a i Mealy'ego.

Zmienne boolowskie występujące w interpretowanej sieci Petriego można podzielić na trzy zbiory:

X - zbiór zmiennych wejściowych,
Y - zbiór zmiennych wyjściowych,
Z - zbiór zmiennych wewnętrznych, w praktyce niewykorzystywany i zazwyczaj $Z = \emptyset$.

Interpretowana sieć Petriego posiada warunek φ przypisany do każdej tranzycji t. Warunek ten zdefiniowany jest, jako funkcja boolowska zmiennych wejściowych i wewnętrznych. W szczególnym przypadku może on być zdefiniowany, jako logiczna wartość 1. W interpretowanej sieci Petriego, tranzycji t może zostać odpalona wtedy i tylko wtedy, gdy wszystkie jej miejsca wejściowe są oznakowane i aktualna wartość przypisanej do niej funkcji boolowskiej φ wynosi 1.

Interpretowane sieci Petriego typu Moore'a i Mealy'ego, podobnie jak skończone automaty stanów, różnią się jedynie w sposobie generowania sygnałów wyjściowych. W omawianej pracy wykorzystywane będą jednak tylko sieci Petriego typu Moore'a. W takiej sieci Petriego elementarna koniunkcja ψ, zbudowana z afirmacji zmiennych wyjściowych, przypisana jest do miejsca p. Jeżeli miejsce p jest znakowane, to zmienne wyjściowe z odpowiedniej koniunkcji ψ mają przypisaną wartość 1.

Kolejnego rozszerzenia sieci Petriego można dokonać poprzez przypisanie kolorów do miejsc i tranzycji [8]. Taka sieć nazywana jest kolorowaną siecią Petriego albo kolorowaną interpretowaną siecią Petriego, jeżeli oba rozszerzenia zostaną zastosowane [11]. Kolory te pozwalają intuicyjnie i formalnie dokonać walidacji wszystkich sekwencyjnych procesów w rozważanej sieci Petriego. Każdy kolor identyfikuje jedną podsieć typu automatowego. Zasady kolorowania sieci Petriego są następujące [3]:

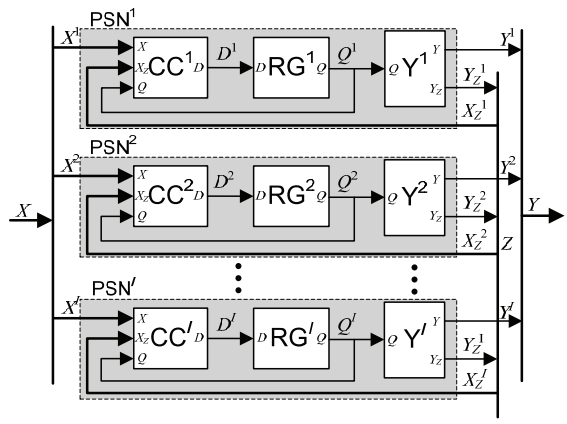
- Każde miejsce i tranzycja muszą posiadać przynajmniej jeden kolor.
- Jeżeli miejsce posiada dany kolor każda jego wejściowa i wyjściowa tranzycja musi również posiadać ten sam kolor.
- Miejsca wejściowe jednej tranzycji muszą posiadać różne kolory.
- Miejsca wyjściowe jednej tranzycji muszą posiadać różne kolory.
- Miejsca wejściowe i wyjściowe jednej tranzycji muszą dzielić ten sam zbiór kolorów.
- Miejsca należące do znakowania początkowego nie mogą dzielić dokładnie tego samego zbioru kolorów.
- Liczba różnych kolorów, które są współdzielone przez miejsca należące do znakowania początkowego jest równa liczbie wszystkich kolorów.

W omawianej pracy wykorzystywane będą jedynie kolory przypisane do miejsc.

3. Idea rozproszonego systemu sterowania

3.1. Architektura systemu

W omawianym systemie sterowania zastosowano architekturę globalnie asynchroniczną lokalnie synchroniczną (GALS). Ponieważ każdy kolor determinuje jeden proces sekwencyjny, przyjęto, że na podstawie tych kolorów dokona się dekompozycji na podsieć typu automatowego (PSN). W wyniku uzyska się I takich podsieci, gdzie I jest również liczbą kolorów występujących w sieci Petriego. Proces dekompozycji zostanie omówiony w dalszej części tego artykułu, przy opisie metody syntezy. Każda z podsieci zostanie zrealizowana z wykorzystaniem układu sekwencyjnego, posiadającym lokalny sygnał zegarowy, o dwupoziomowej architekturze (rys. 1).



Rys. 1. Architektura rozproszonego systemu sterowania
Fig. 1. Distributed control system architecture

Układ kombinacyjny CC^i pierwszego poziomu odpowiedzialny jest za wygenerowanie funkcji wzbudzeń przerzutników i jego działanie można opisać funkcją:

$$D^i = D^i(X, X_Z, Q^i), \quad (3)$$

gdzie X jest zbirem (podzbiorem) zmiennych wejściowych sieci Petriego; X_Z jest podzbiorem zbioru Z dodatkowych zmiennych boolowskich, utworzonych w procesie syntezy, służących do synchronizacji pracy wszystkich podsieć typu automatowego; Q^i jest zbiorem zmiennych boolowskich wykorzystanych do zapamiętania kodu aktualnie oznakowanego miejsca. Ponieważ podsieć typu automatowego reprezentuje jeden proces sekwencyjny, w każdym momencie, może w niej być oznakowane tylko jedno

miejsce. Pamięć układu RG^i, która pamięta kod aktualnie oznakowanego miejsca, zbudowana jest z przerzutników typu D. Liczba wymaganych przerzutników uzależniona jest od liczby miejsc w danej podsieci oraz sposobu kodowania i zostanie zdefiniowana przy omawianiu metody syntezy. Dekoder Y^i drugiego poziomu odpowiedzialny jest za generowanie zmiennych wyjściowych Y oraz dodatkowych zmiennych boolowskich Y_Z służących do synchronizacji pracy wszystkich podsieć typu automatowego. Jego funkcjonowanie można opisać funkcjami:

$$\begin{aligned} Y^i &= Y^i(Q^i), \\ Y_Z^i &= Y_Z^i(Q^i). \end{aligned} \tag{4}$$

Do prawidłowego funkcjonowania całego systemu należy odpowiednio połączyć wszystkie podsieci PSN^i (rys. 1). Magistrala Z służy do przesyłana dodatkowych synchronizujących zmiennych boolowskich. Składa się ona ze zmiennych generowanych przez układy poszczególnych podsieci:

$$Z = \bigcup_{i=1}^{I} Y_Z^i$$

i jej podzbiory stanowią wejścia dla poszczególnych podsieci

$$X_Z^i \subseteq Z.$$

Magistrala Y stanowi wyjście całego systemu i jest konkatenacją sygnałów wyjściowych generowanych przez poszczególne układy:

$$Y = \bigcup_{i=1}^{I} Y^i.$$

W praktyce, nie jest ona realizowana, gdyż poszczególne podzbiory sygnałów wyjściowych, generowanych przez konkretny układ, podłączane są bezpośrednio do sterowanego obiektu.

Magistrala X stanowi wejście całego systemu i jej podzbiory stanowią wejścia do dla poszczególnych podsieci

$$X^i \subseteq X.$$

W praktyce, również nie jest ona realizowana, gdyż poszczególne podzbiory sygnałów wejściowych, podłączane są do układu bezpośrednio ze sterowanego obiektu.

3.2. Metoda syntezy

Metoda syntezy oparta jest na dekompozycji sieci Petriego na podsieci automatowe oraz kodowanie miejsc w każdej podsieci z wykorzystaniem minimalnej liczby bitów. Dekompozycji dokonuje się na podstawie kolorów przypisanych do miejsc w pokolorowanej sieci Petriego. Zmienne wyjściowe, przypisane do miejsc, dekodowane są w układzie drugiego poziomu, który realizowany jest z wykorzystaniem osadzonych bloków pamięci. Podejście takie pozwala na równomierne wykorzystanie różnorodnych zasobów sprzętowych dostępnych w nowoczesnych układach FPGA [4].

Punktem wejścia do metody syntezy jest pokolorowana sieć Petriego. Istnieje wiele algorytmów kolorowania sieci Petriego [5, 9, 10] i nie będą one omawiane w niniejszej pracy. Ponieważ algorytmy te nie uwzględniają w procesie kolorowania sygnałów wyjściowych, należy sprawdzić czy miejsca, których elementarne koniunkcje zawierają ten sam sygnał wyjściowy y, posiadają ten sam kolor. Własność taka wymagana jest do poprawnego funkcjonowania systemu sterowania i zapobiega konfliktowi dostępu do jednego zasobu przez procesy współbieżne. Można ją zdefiniować, jako rozszerzenie kolorowania na elementarne koniunkcje, poprzez przypisaniu im kolorów miejsca, do którego są przypisane. Do zasad kolorowania należałoby wtedy dodać jedne punkt:

- Elementarne koniunkcje, które zawierają wspólny sygnał wyjściowy y, muszą posiadać przynajmniej jeden ten sam kolor.

Cały proces syntezy składa się z następujących kroków:

1. **Dekompozycja na składowe podsieci typu automatowego**. Celem tego kroku jest uzyskanie podsieci z początkowej pokolorowanej sieci Petriego. Załóżmy, że sieć Petriego jest pokolorowana I różnymi kolorami. Rozpocznijmy proces dekompozycji od pierwszego koloru ($i = 1$). Wszystkie miejsca pokolorowane tym kolorem utworzą pierwsza podsieć typu automatowego PSN_1. Następne podsieci tworzone są w podobny sposób, z tym, że wszystkie sekwencje miejsc, które zostały już umieszczone w jednej ze wcześniej uzyskanych podsieci (są pokolorowane również kolorem o niższym numerze niż obecnie analizowany), są zastępowane makromiejscem, a w tworzonej podsieci umieszczany jest dubler tego makromiejsca $dmp_m \in DMP_i$. W tworzonej podsieci możemy uzyskać kilka takich dublerów. Jeżeli występują one sekwencyjnie to można zastąpić je jednym dublerem.

2. **Utworzenie zbioru zmiennych synchronizujących**. Celem tego kroku jest uformowanie zbioru zmiennych Z synchronizujących pracę całego systemu sterowania i przypisania ich do poszczególnych tranzycji i miejsc. Należy zbadać każdą tranzycję. Jeżeli tranzycja t_s należy do więcej niż jednej podsieci PSN_i, to w każdej z tych podsieci elementarna koniunkcja ψ przypisaną do miejsce wejściowe tej tranzycji należy zastąpić nową elementarna koniunkcja $\psi^* = \psi \wedge y_Z^{p_m}$, gdzie $y_Z^{p_m}$ jest nowo utworzonym sygnałem synchronizującym i $y_Z^{p_m} \in Y_Z^i$. Następnie, należy zastąpić warunek φ przypisany do tej tranzycja t_s, w każdej z podsieci PSN_i niezależnie, nowym warunkiem $\varphi^* = \varphi \wedge \left(\bigwedge x_Z^{p_m}\right)$, gdzie $\bigwedge x_Z^{p_m}$ jest sumą logiczną sygnałów $y_Z^{p_m}$ generowanych w pozostałych podsieciach i podłączonych do odpowiadających wejść $x_Z^{p_m} \in X_Z^i$ rozważanej podsieci PSN_i.

3. **Kodowanie miejsc**. Celem tego kroku jest przypisanie binarnego kodu $K(p)$ każdemu miejscu, w każdej z podsieci niezależnie. Kodowanie wykonane zostanie na minimalnej liczbie bitów. Umożliwi to efektywne zastosowanie osadzonych bloków pamięci do realizacji dekodera Y^i drugiego poziomu. Wymagane jest, więc zastosowanie

$$R_i = \lceil \log_2 |P_i| \rceil \tag{5}$$

bitów do kodowania miejsc w poszczególnych podsieciach, gdzie $P_i \subseteq P \cup DMP_i$ jest zbiorem miejsc danej podsieci PSN_i. Do zapamiętania kodu miejsca zostaną wykorzystane zmienne ze zbioru $Q^i = \{q_0, \dots, q_{R_i-1}\}$. Miejsce, które należy do znakowania początkowego M_0, otrzymuje kod równy 0. Jeżeli dana podsieć nie posiada miejsca należącego do znakowania początkowego M_0 to kod równy 0 przypisywany jest dublera makromiejsca, który zastąpił to miejsce. Pozostałe miejsca otrzymują kolejne kody zgodne z kodem Grey'a lub ewentualnie naturalnym kodem binarnym.

4. **Formowanie koniunkcji**. Celem tego kroku jest utworzenie koniunkcji opisujących miejsca, tranzycje i warunki utrzymania miejsca. Są one przydatne do łatwego opisana równań logicznych tworzących system (3). Tworzy je się niezależnie dla każdej podsieci PSN_i. Koniunkcja opisująca miejsce p zawiera tylko afirmację lub negację zmiennych $q_r \in Q^i$, użytych do przechowania kodu miejsca. Jeżeli kod $K(p)$ miejsca p posiada wartość 0 na bicie o indeksie r to w skład koniunkcji wchodzi negacja zmiennej q_r, natomiast, jeżeli posiada on wartość 1 to w skład koniunkcji wchodzi afirmacja zmiennej q_r:

$$p = \bigwedge_{r=0}^{R-1} q_r^l,$$

gdzie $l \in \{0,1\}$ jest wartością bitu o indeksie r w kodzie $K(p)$: $q_r^0 = \overline{q_r}$, $q_r^1 = q_r$. Koniunkcja opisująca tranzycje t składa się z koniunkcji miejsca wejściowego do tej tranzycji i warunku φ (lub φ^*, jeżeli został nim zastąpiony) przypisanego do tej tranzycji:

$$t = p \wedge \varphi.$$

Koniunkcja utrzymania miejsca p składa się z negacji sumy wszystkich tranzycji wyjściowych tego miejsca i koniunkcji tego miejsca:

$$hp = \overline{\bigvee t} \wedge p.$$

5. **Formowanie równań logicznych**. Tworzone równania logiczne opisują system (3), implementowany przez układ kombinacyjny CC^i. Są one tworzone na bazie równania przerzutnika typu D. Są one zbudowane z koniunkcji tranzycji i utrzymania miejsca. Jeżeli zmienna q_r ma wartość 1 w kodzie $K(p)$ miejsca p wtedy suma odpowiadającej zmiennej D_r składa się z sumy koniunkcji tranzycji wejściowych miejsca p i koniunkcja utrzymania miejsca p:

$$D_r = \bigvee \left(\bigvee t \wedge hp \right).$$

6. **Formowanie pamięci dekodera**. Ze względu na regularność systemu (4) efektywne jest zaimplementowanie dekodera Y^i z wykorzystaniem osadzonych bloków pamięci. Umożliwia równomierne wykorzystanie zasobów układu FPGA. Zawartość tej pamięci może być opisana za pomocą tablicy prawdy lub równań logicznych. Zastosowanie tablicy prawdy jest jednak wygodniejszym sposobem opisu zawartości pamięci. W takim przypadku zmienne wejściowe $q_r \in Q^i$ formują adres a zmienne wyjściowe $y \in (Y^i \subseteq Y) \cup Y_Z^i$ stanowią słowo zapisane pod tym adresem. Zbiór Y^i zawiera tylko te zmienne wyjściowe, które są kontrolowane przez podsieć PSN_i. W każdym wierszu tablicy wartość 1 przyjmują tylko te zmienne wyjściowe y, które wchodzą w skład elementarnej koniunkcji ψ (lub ψ^*, jeżeli została nią zastąpiona) przypisanej do miejsca p reprezentowanego przez zmienne wejściowe q_r, tworzące jego kod $K(p)$.

7. **Utworzenie układu logicznego**. Krok ten opisuje zasady tworzenia modelu podsieci w języku opisu sprzętu oraz ich połączenia w system sterowania. Moduł opisujący układ kombinacyjny CC^i opisuje się z wykorzystaniem przypisań ciągłych i operatorów logicznych. Opisuje on wszystkie równania utworzone w kroku 5. Pamięć układu RG^i należy opisać jako R_i-bitowy rejestr typu D z asynchronicznym sygnałem zerującym. Dekoder Y^i można opisać jako proces z wykorzystaniem konstrukcji **case**. Ponieważ osadzone bloki pamięci są synchroniczne, lista czułości tego procesu powinna zawierać sygnał zegarowy. Sygnał zegarowy musi być zrealizowany jako synchroniczny ponieważ osadzone bloki pamięci nie wspierają asynchronicznego zerowania. W celu zapewnienia, że moduł ten zostanie zsyntezowany z wykorzystaniem osadzonych bloków pamięci należy dodatkowo umieści specjalną dyrektywę dla narzędzia syntezy. Składnia tej dyrektywy zależna jest od docelowego układu FPGA. Selektor konstrukcji **case** stanowią zmienne wejściowe tablicy prawdy utworzonej w kroku 6., a przypadki przypisania wartości do zmiennych wyjściowych. Moduł główny podsieci powinien zawierać opis połączeń tych trzech bloków zgodnie z wydzielonym fragmentem dla pojedynczej podsieci na rysunku 1. Dodatkowo lokalny sygnał zegarowy i zerujący należy odpowiednio podłączyć do wejść zegarowych i zerujących rejestru RG^i i dekodera Y^i. Dekoder powinien być wyzwalany na przeciwnym zboczy niż rejestr. Zapienia to, że wyjścia będą generowane w ciągu jednego cyklu zegarowego [4]. W przypadku rozproszonej implementacji nie tworzy się głównego modułu dla całego systemu sterowania. Połączenia poszczególnych układów logicznych podsieci realizuje się fizycznie podczas instalacji systemu sterowania zgodnie z rysunkiem 1.

4. Podsumowanie

W artykule przedstawiono metodę realizacji rozproszonego systemu sterowania. Jako metodę specyfikacji algorytmu przyjęto sieć Petriego, która bardzo dobrze opisuje procesy współbieżne. Zaproponowana architektura systemu bazuje na lokalnej synchronizacji układów logicznych realizujących funkcję poszczególnych podsieci. Ze względu na problemy z przekazywaniem sygnałów zegarowych w systemach rozproszonych zrezygnowano z globalnej synchronizacji. Doprowadziło to do powstania architektury GALS. W celu synchronizacji pracy całego systemu zastosowano natomiast dodatkowe sygnały binarne. Zasady ich generowania i wykorzystania zostały opisane w zaproponowanej metodzie syntezy. Metoda ta ukierunkowana jest na wykorzystanie nowoczesnych układów FPGA. W celu równomiernego wykorzystania różnorodnych zasobów sprzętowych, w nich występujących, zaproponowano realizację dekodera z wykorzystaniem osadzonych bloków pamięci.

5. Literatura

[1] Adamski M., Barkalov A.: Architectural and Sequential Synthesis of Digital Devices. University of Zielona Góra Press, Zielona Góra, 2006

[2] Barkalov A., Titarenko L.: Logic Synthesis for FSM-based Control Units. Springer-Verlag, Berlin, 2009.

[3] Biliński K., Adamski M., Saul J., Dagless E.: Petri-net-based algorithms for parallel-controller synthesis. IEE Proceedings - Computers and Digital Techniques, 1994, Vol. 141, No. 6, 405-412.

[4] Bukowiec A.: Synthesis of Finite State Machines for FPGA Devices Based on Architectural Decomposition, vol. 13 Lecture Notes in Control and Computer Science. University of Zielona Góra Press, Zielona Góra, 2009.

[5] Jensen K., Kristensen L., Wells L.: Coloured Petri Nets and CPN Tools for Modelling and Validation of Concurrent Systems. International Journal on Software Tools for Technology Transfer, 2007, Vol. 9, No. 3/4, 213-254.

[6] Karatkevich A.: Dynamic Analysis of Petri Net-Based Discrete Systems, vol. 356 Lecture Notes in Control and Information Sciences. Springer, Berlin, 2007.

[7] Łuba T.: Synteza układów logicznych. Oficyna Wydawnicza Politechniki Warszawskiej, Warszawa, 2005.

[8] Murata T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE, 1989, Vol. 77, No. 4, 541-580.

[9] Tkacz J.: Kolorowanie automatowe sieci Petriego metodą wnioskowania symbolicznego. Pomiary, Automatyka, Kontrola, 2007, Nr 5, 120-122.

[10] Węgrzyn A.: On decomposition of Petri net by means of coloring. Proceedings of IEEE East-West Design & Test Workshop EWDTW'06, 2006, 407-413. Sochi, Rosja.

[11] Węgrzyn M., Wolański P., Adamski M., Monteiro J.: Coloured Petri net model of application specific logic controller programs. Proceedings of IEEE International Symposium on Industrial Electronics ISIE'97, 1997, vol. 1, 158-163. Guimarães, Portugalia.

otrzymano / received: 15.03.2012
przyjęto do druku / accepted: 01.05.2012 *artykuł recenzowany / revised paper*