Marian ADAMSKI, Marek WĘGRZYN
UNIWERSYTET ZIELONOGÓRSKI
INSTYTUT INFORMATYKI I ELEKTRONIKI

Implementacja współbieżnych algorytmów sterowania w reprogramowalnych sterownikach logicznych

Prof. dr hab. inż. Marian ADAMSKI

Dyrektor Instytutu Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Zainteresowania badawcze obejmują projektowanie systemów cyfrowych realizowanych w postaci mikrosystemów cyfrowych oraz formalne metody programowania sterowników logicznych PLC. Członek Polskiego Towarzystwa Elektrotechniki Teoretycznej i Stosowanej oraz Polskiego Towarzystwa Informatycznego.
e-mail: M.Adamski@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 Polskiego Towarzystwa Informatycznego.
e-mail: M.Wegrzyn@iie.uz.zgora.pl

Streszczenie

W artykule omówiono metodologię bezpośredniego odwzorowania sieci Petriego opisującej algorytm sterowania, równoważnej grafowi SFC (Sequential Function Chart), w strukturze reprogramowalnego sterownika logicznego, realizowanego z wykorzystaniem układów typu FPGA i CPLD. Wskazano na rolę języków HDL (VHDL i Verilog) w modelowaniu i syntezie rozpatrywanej klasy układów.

Abstract

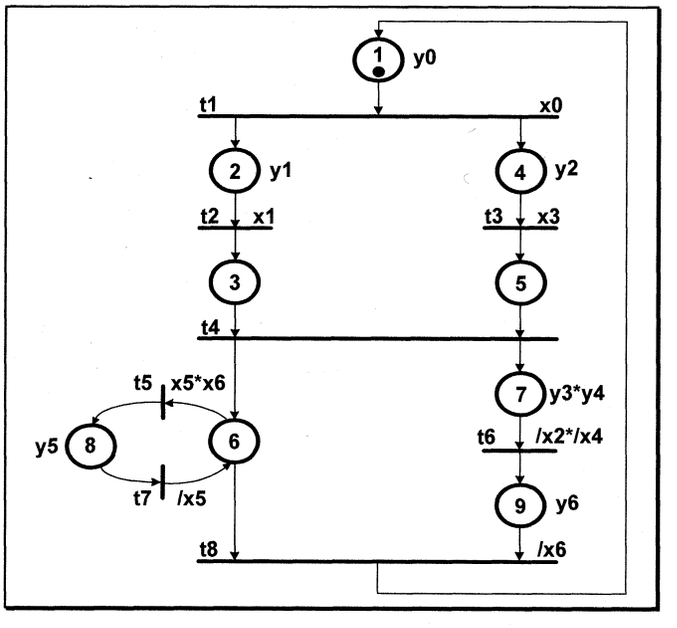
In the paper, a method of direct mapping of Petri net describing logic controllers, or equivalent Sequential Function Chart (SFC), into a structure of reprogrammable logic controller realized by means of programmable logic (FPGA and CPLD), is presented. In addition, using of HDLs in modeling and synthesis of considered circuits is discussed.

1. Wprowadzenie

W artykule przedstawiono przegląd metod układowej implementacji algorytmów sterowania binarnego, ze szczególnym uwzględnieniem dorobku Uniwersytetu Zielonogórskiego, współpracującego w tej dziedzinie z University of Bristol (Anglia), Universidade do Minho (Portugalia), FernUniversitaet Hagen, Technische Universitaet Ilmenau (Niemcy) oraz Akademią Nauk Białorusi w Mińsku.

Prace nad układową, strukturalną implementacją sieci Petriego i sieci Grafcet z wykorzystaniem programowalnych struktur logicznych rozpoczęto w Zielonej Górze już w 1981 roku [1]. Motywacją do podjęcia się tego zadania było stopniowe wdrażanie do praktyki inżynierskiej w USA i Europie strukturalnych metod projektowania układów cyfrowych na poziomie RTL (*Register Transfer Level*), bazujących na specyfikacji automatowej w postaci interpretowanych grafów zorientowanych - grafu stanów lub sieci działań [13].

Podstawowym niedostatkiem czysto automatowej, sekwencyjnej formy specyfikacji, opierającej się wyłącznie na globalnych stanach wewnętrznych, modelowanego i syntetyzowanego systemu, jest zerwanie więzi między implementacją, a naturalnie przebiegającymi, często niezależnymi zdarzeniami współbieżnymi. Pomysł wykorzystania sieci Petriego w celu modelowania układów cyfrowych, a zwłaszcza sterowników logicznych, pojawił się pod koniec lat 70-tych, głównie w Europie, a zwłaszcza we Francji [10].



Rys. 1. Sieć Petriego opisująca algorytm sterowania

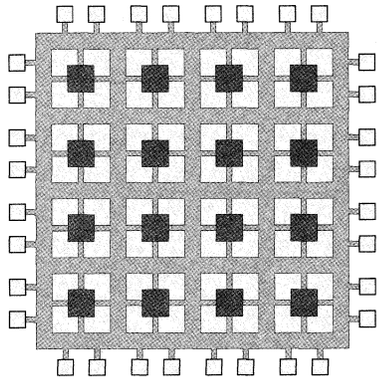
Oryginalnym wynikiem, uzyskanym w ośrodku zielonogórskim było formalne odwzorowanie sieci Petriego w symbolicznej postaci regułowej (logika sekwentów Gentzena) i takie zakodowanie miejsc, aby pełną strukturę sieci można

było jednoznacznie odzwierciedlić w strukturze PLA [1,2]. Uniknięto w ten sposób procedury polegającej na dekompozycji sieci Petriego na poszczególne składowe automatowe i odrębnej implementacji układowej powiązanych ze sobą części sieci. Uniknięto również nieekonomicznej implementacji sieci za pośrednictwem monolitycznego, równoważnego jej pod względem behawioralnym automatu cyfrowego, uzyskiwanego poprzez interpretacje grafu znakowań. Nie bez znaczenia jest możliwość wykorzystania bogatej teorii sieci Petriego oraz licznych narzędzi umożliwiających jej formalną analizę i poglądową animację.

W latach 1986-1990 zrealizowano eksperymentalny system „LOGICIAN" [1,2] umożliwiający układowe odwzorowanie interpretowanej sieci Petriego (Rys.1) w strukturach reprogramowalnych. Następcami systemu „LOGICIAN" był system Paris [9,14], rozwijany w University of Bristol w latach 1991-96, oraz system „CONPAR", zbudowany w Universidade do Minho [11].

Pojawienie się efektywnych przemysłowych systemów CAD do projektowania układów z programowalnymi strukturami logicznymi spowodowało zatrzymanie prac nad własnym systemem akademickim. Nowym kierunkiem badań (1991-1994) było wykorzystanie możliwości istniejącego już na rynku profesjonalnego oprogramowania w syntezie cyfrowych układów współbieżnych [6,16]. W początkowym okresie wykorzystywano do tego celu popularne systemy: CUPL, Palasm, Abel i AMAZE, a dopiero później zwrócono uwagę na bardziej technologicznie zaawansowane, ale znacznie droższe systemy, wykorzystujące VHDL.

Obok nowego sposobu syntezy na poziomie RTL z wykorzystaniem standardowych języków HDL, opracowano i zrealizowano oryginalną metodę bezpośredniego odwzorowania sieci Petriego w strukturze FPGA (Rys.2). Metoda ta została zweryfikowana praktycznie dla formatu XNF i elementów reprogramowalnych firmy XILINX [19]. Na uwagę zasługują również prace dotyczące projektowania reprogramowalnych sterowników logicznych dla potrzeb przemysłowych, w tym sterowników o podwyższonej niezawodności [12,18].




```
entity reaktor is
  port(Clk: in STD_LOGIC;
       Reset: in STD_LOGIC;
       X: in STD_LOGIC_VECTOR(0 to 6);
       Y: out STD_LOGIC_VECTOR(0 to 6));
end reaktor;
```

Rys. 2. FPGA jako reprogramowalnych sterownik logiczny

Począwszy od 1996 r. zintensyfikowano prace nad nowym systemem CAD, nazwanym PeNCAD. System ten ma pełnić rolę nakładki dla komercjalnego oprogramowania, związanego z językami VHDL i Verilog [7,17].

W dalszej części artykułu w syntetyczny sposób przedstawiono wybrane, aktualne zagadnienia badawcze, realizowane w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Przegląd ważniejszych, wcześniejszych prac dotyczących rozpatrywanej dziedziny realizowanych w Instytucie Informatyki i Elektroniki UZ oraz innych ośrodkach naukowych zawiera referat [15]. Podobne badania, zmierzające w stronę sprzętowej implementacji układów sterowania binarnego opisywanych sieciami Petriego były prowadzone w Mińsku (Białoruś) przez zespół kierowany przez prof. A. Zakrevskiego [8].

2. Układowa implementacja systemów reaktywnych w reprogramowalnych strukturach logicznych

Ze względu na intensywny rozwój technologii reprogramowalnych układów cyfrowych (RUC) i związanych z nimi narzędzi do komputerowego projektowania (CAD) wzrasta zainteresowanie metodami bezpośredniej, układowej implementacji algorytmów sterowania binarnego [5]. Perspektywiczną dziedziną jest synteza reprogramowalnych sterowników logicznych RLC (Reprogrammable Logic Controllers) należących do szerszej klasy sterowników ASLC (Application Specific Logic Controllers) [13]. Coraz bardziej zaczyna się ugruntowywać pogląd, że synteza małych systemów reaktywnych z wykorzystaniem elementów FPGA i CPLD jest ekonomicznie i technologicznie uzasadniona. Z drugiej strony w liczących się na świecie laboratoriach naukowych w Europie, USA i Japonii pojawia się tendencja wykorzystywania do tego celu nowych metod specyfikacji behawioralnej lub strukturalnej, opartych na solidnych podstawach matematycznych, wziętych z logiki formalnej, teorii grafów i tradycyjnie wykorzystywanej już od dawna teorii automatów.

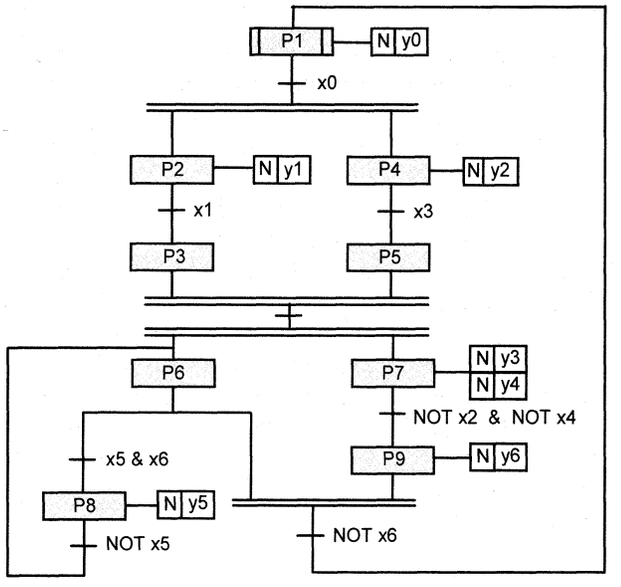
Zdaniem wielu autorów wyrafinowane algorytmy funkcjonowania urządzenia cyfrowego nie powinny być konstruowane metodą intuicyjną i od razu zapisywane (kodowane) w językach VHDL lub Verilog. Opinię taką uzasadniają również doświadczenia zaczerpnięte z inżynierii oprogramowania, gdyż szybki i sprawny algorytm, zapisany w kilkunastu wierszach kodu, usprawnia wielokrotnie funkcjonowanie złożonego programu sterowania.

Postępowanie, polegające na opisie funkcjonowania układu w postaci grafu SFC lub sieci Petriego i transformacji na program w języku VHDL może być dogodniejsze, niż nie zawsze udana próba formułowania problemu, zapisanego w języku naturalnym, od razu w postaci programu [3,4,7,11]. W przypadku systemów silnie reaktywnych, w sposób niemal natychmiastowy odpowiadających na sygnały z otoczenia, bardzo trudno jest w sposób wystarczająco klarowny przedstawić funkcjonowanie projektowanego układu sterującego bez uwzględnienia w programie, pośrednio lub bezpośrednio, pojęć takich, jak: lokalny stan wewnętrzny oraz globalny stan wewnętrzny. W większości przypadków sposób reakcji mikrosystemu dla tego samego stanu wejść zależy od poprzedniej sekwencji wejść, co w projektowaniu intuicyjnym

uwzględnia się w postaci tak zwanych zmiennych statusu, czyli „flag". Są one doraźnie wprowadzonymi stanami lokalnymi i dlatego ich wzajemna, intuicyjna koordynacja jest utrudniona.

Dzięki wykorzystaniu nowoczesnych struktur logicznych FPGA i CPLD realizacja rekonfigurowalnego, szybkiego mikrosystemu cyfrowego, pełniącego rolę małego, specjalizowanego sterownika wraz ze wszystkimi elementami peryferyjnymi, spełniającego wymagania normy IEC 1131-3 i realizującego algorytm sterowania w sposób układowy, staje się technicznie możliwa i uzasadniona.

Opracowana metoda syntezy polega na odzwierciedleniu struktury grafu sterowania SFC (Rys. 3) w elementach FPGA i CPLD (Rys.2) za pośrednictwem interpretowanej sieci Petriego formalnie opisującej proces sterowania (Rys.1) .



Rys. 3. Graf sterowania SFC

3. Bezpośrednia synteza współbieżnego automatu sterującego

Opisywany sposób syntezy metodą bezpośrednią nie wiąże się z bardzo nieekonomiczną i niepraktyczną transformacją sieci Petriego na interpretowany graf znakowań. Przekształcając sieć Petriego na graf stanów klasycznego automatu cyfrowego traci się bowiem więź między specyfikacją algorytmu i jego układową implementacją oraz na ogół uzyskuje się układ o wyjątkowo nadmiarowej złożoności (eksplozja kombinatoryczna). Unika się również kłopotliwej i nienaturalnej dekompozycji równoległej na komunikujące się automaty sekwencyjne. Zaproponowany sposób syntezy polega na pośrednim kodowaniu globalnego stanu wewnętrznego automatu współbieżnego, poprzez superpozycję kodów jego poszczególnych stanów lokalnych. W rozpatrywanym sposobie kodowania wewnętrznych stanów lokalnych:

1) kody miejsc sieci Petriego względem siebie współbieżnych, występujących w tym samym wierzchołku grafu znakowań są nieortogonalne;
2) kody miejsc sieci Petriego względem siebie niewspółbieżnych (nigdy nie występujących w tym samym wierzchołku grafu znakowań) są ortogonalne.

Biorąc pod uwagę coraz szerszą dostępność systemów do komputerowej syntezy układów cyfrowych, zalecanym sposobem postępowania jest bezpośrednie odwzorowanie symbolicznego opisu regułowego w języku VHDL.

Ze względu na współbieżność między procesami sekwencyjnymi, jednemu stanowi lokalnemu (miejscu sieci Petriego, krokowi w sieci SFC) może odpowiadać wiele kodów binarnych, o długości równej długości rejestru stanu. Ich wspólna część zapisywana jest symbolicznie za pomocą termu (koniunkcji), albo odpowiadającego mu wektora ternarnego. W rozpatrywanym przykładzie kodowania miejsc sieci Petriego, a tym samym kroków grafu SFC, metodą heurystyczną, miejsca P1, ..., P9 koduje się wektorem [Q1, Q2, Q3, Q4]:

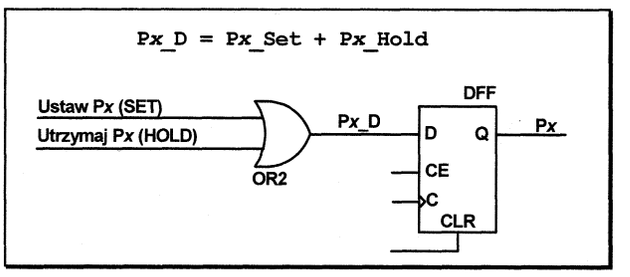
```
P1= [0 - - -]          P1= /Q1
P2= [1 0 0 -]          P2= Q1*/Q2*/Q3
P3= [1 0 1 -]          P3= Q1*/Q2*Q3
P4= [1 0 - 0]          P4= Q1*/Q2*/Q4
   ...                    ... itp.
```

Wypadkowy kod globalnego stanu wewnętrznego musi umożliwiać niezależną realizację wszystkich współbieżnych zmian stanów lokalnych, których obserwowalnym skutkiem jest synchroniczne osiągnięcie nowego, zamierzonego stanu globalnego. Istotnym warunkiem jest rozróżnialność stanów globalnych między sobą, gwarantowana przez odpowiedni dobór kodów stanów lokalnych. Przykładowo, oznakowanie sieci {P2, P4} kodowane jest wektorem [1000], któremu odpowiada term P2*P4, czyli Q1*/Q2*/Q3*/Q4.

W szczególnym przypadku bezpośredniego kodowania typu pierścieniowego („one-hot" - „jeden z N"), każdemu miejscu sieci przypisany jest odrębny bit. Tym samym każdemu miejscu sieci przyporządkowany jest określony przerzutnik rejestru stanu wewnętrznego: [Q1, ..., Q9] (Rys.4). Inny specyficzny rodzaj kodowania polega na pokryciu sieci podsieciami automatowymi (P-sieciami) i zakodowaniu poszczególnych ich miejsc w klasyczny sposób znany z teorii automatów cyfrowych wektorem [Q1,..., Q4]:

```
P1= /Q1 * /Q2 * /Q3 * /Q4;
P2= /Q1 * Q2;
P3=  Q1 * /Q2;
P4= /Q3 * Q4
   ... itp.
```

Podobnie jak w poprzednio rozważanych przypadkach miejsce należące do kilku podsieci otrzymuje kod będący superpozycją jego kodów w poszczególnych podsieciach. Z porównania realizacji sprzętowych wynika, że nowe algorytmy opracowane w Instytucie, opierające się na hierarchicznej dekompozycji sieci Petriego, wykorzystujące metodę heurystyczną łączą w sobie zalety pozostałych metod.



Rys. 4. Realizacja w FPGA miejsca sieci Petriego przy kodowaniu „one-hot"

4. Specyfikacja i synteza sterownika logicznego na poziomie RTL

Graf SFC lub równoważna mu sieć Petriego są szczególnie przydatne do specyfikacji i syntezy sterowników logicznych, głównie ze względu na jawnie uwidocznioną niezależność między równocześnie występującymi stanami lokalnymi oraz poglądowe przedstawianie powiązań miedzy wyodrębnionymi współbieżnymi procesami o charakterze sekwencyjnym. Przejrzyste odwzorowanie hierarchii może odbywać się za pomocą zagnieżdżających się podsieci, abstrahowanych jako makromiejsca.

Interpretowana hierarchiczna sieć Petriego jest dogodnym modelem formalnym, wykorzystywanym w projektowaniu współbieżnych układów cyfrowych, zwłaszcza na poziomie RTL. Dogodną platformą do modelowania i automatycznej syntezy jest komfortowe środowisko języka VHDL, na przykład system Active-HDL firmy ALDEC.

Wykorzystanie modelu części sterującej układu cyfrowego w postaci sieci Petriego i jego bezpośrednie odwzorowanie jej struktury w stosunkowo prostych konstrukcjach języka HDL daje wymierne efekty praktyczne w postaci efektywnej, komputerowej implementacji (Rys. 5). Ponadto VHDL i związane z tym językiem standardowe, profesjonalne i uniwersalne środowisko projektowania zapewnia również możliwość symulacji sieci Petriego, poprzez śledzenie przepływu poszczególnych znaczników (markerów), obserwowanych za pomocą specjalnie w tym celu generowanych sygnałów pomocniczych. W trakcie symulacji, stosując formalne asercje można wykryć podwójne oznakowanie miejsca (gdy sieć nie jest bezpieczna) oraz zabronione konfiguracje miejsc, wejść lub wyjść (gdy w algorytmie występują określone defekty).

```
architecture MIEJSCA of reaktor is
   Signal P : std_logic_vector(1 to 9);
begin
  MI:process (CLK, RESET) -- Miejsca
   begin
      if RESET='1' then P<="100000000";
      elsif CLK'event and CLK='1'
        then P<="000000000";
             if P(1)='1' then
                if X(0)='1' then P(2)<='1';
                                 P(4)<='1';
                           else P(1)<='1';
                end if;
             end if;
             if P(2)='1' then
                if X(1)='1' then P(3)<='1';
                           else P(2)<='1';
                end if;
             end if;
      ...
      end if;
   end process;
  WY:process (P) -- Wyjscia
   begin
    Y(0) <= P(1);
    Y(1) <= P(2);
    ...
   end process;
end MIEJSCA;
```

Rys. 5. Fragment przykładowego modelu w języku VHDL

Problem równoczesnego wykorzystania oprogramowania przeznaczonego do formalnej analizy sieci Petriego i nowoczesnego środowiska HDL do pełnej symulacji i syntezy współbieżnego układu cyfrowego wymagał określenia takich form specyfikacji, które mają oczywiste odbicie zarówno w strukturze sieci Petriego, jak i w odpowiednich konstrukcjach języka HDL.

Istotną zaletą modelowania sieci Petriego w środowisku VHDL, w stosunku do dedykowanych narzędzi przeznaczonych wyłącznie do animacji sieci Petriego, jest bezpośrednie wykorzystanie wyników modelowania w celu komputerowej syntezy całego współbieżnego układu cyfrowego, w tym również części operacyjnej, a nie tylko jego części sterującej.

5. Różnice między metodami implementacji w układach typu FPGA i CPLD

Na poziomie syntezy systemowej nie zwraca się uwagi na rodzaj wykorzystanych elementów reprogramowalnych. Właściwe ukierunkowanie w stronę wykorzystywanej bazy elementowej następuje dopiero na etapie kodowania miejsc. Przykładowo, z punktu widzenia syntezy układowej, z wykorzystaniem elementów CPLD o architekturze MegaPAL najlepiej jest kodować stany lokalne z wykorzystaniem minimalnej liczby przerzutników w taki sposób, aby superpozycja kodów równocześnie występujących stanów lokalnych dawała w rezultacie prawidłowy kod stanu globalnego oraz aby lokalne współbieżne zmiany stanów nie kolidowały ze sobą. Dodatkowe możliwości stwarza wykorzystanie do tego celu rejestrowych sygnałów wyjściowych. Postępowanie takie prowadzi niekiedy do nieznacznego wydłużenia kodu stanu globalnego, ale przy bardzo znacznym uproszczeniu złożoności układu sterownika.

6. Podsumowanie i wnioski

Algorytm sterowania binarnego przedstawia się w postaci grafu SFC, zgodnie z normą IEC 1131-3 i przekształca na równoważną, interpretowaną sieć Petriego sterowania. Behawioralnym modelem formalnym projektowanego układu sterownika cyfrowego na poziomie RTL jest automat współbieżny z częścią operacyjną (*Concurrent State Machine with Data Path, CSMD*) [13]. Pomostem między syntetyzowalnym podzbiorem języka VHDL i siecią Petriego jest symboliczny zapis regułowy. Mikrosystem cyfrowy może być modelowany i syntetyzowany z wykorzystaniem standardowego oprogramowania. Struktura sieci Petriego jest wówczas w sposób wzajemnie jednoznaczny odwzorowana w strukturze FPGA lub CPLD.

Behawioralny opis układu cyfrowego w postaci interpretowanej sieci Petriego i późniejsza jego transformacja na program w języku VHDL lub Verilog jest znacznie dogodniejsza, niż nie zawsze udana próba natychmiastowego formułowania problemu w postaci programu w języku VHDL. Nie bez znaczenia jest możliwość wykorzystania w projektowaniu bogatej teorii sieci Petriego oraz licznych komplementarnych do środowiska HDL narzędzi komputerowych, umożliwiających jej formalną analizę lub poglądową animację.

Literatura

1. Adamski M.: *Projektowanie cyfrowych systematyczną metodą strukturalną*, Seria: Monografie, Nr 49, Wydawnictwo WSInż., Zielona Góra, 1990.
2. Adamski M.: Parallel Controller Implementation using Standard PLD Software, w W.R.Moore, W.Luk (Ed.), *FPGAs*, Abingdon EE&CS Books, Abingdon, England, 1991, str.296-304.
3. Adamski M.: SFC, Petri Nets and Application Specific Logic Controllers. *Proceedings of the IEEE Int. Conf. on Systems, Man, and Cybernetics*, IEEE, San Diego, USA, 11-14.10.1998, str.728-733.
4. Adamski M.: Application Specific Logic Controllers for Safety Critical Systems, *Proceedings of the 1999 IFAC Triennial World Congress*, Beijing, China, Vol.Q, str.519-524.
5. Adamski M.: *Bezpośrednia implementacja sieci Petriego w reprogramowalnych układach cyfrowych.* III Krajowa Konferencja Naukowa Reprogramowalne Układy Cyfrowe, RUC'2000, Szczecin 10-11.05.2000, str.131-138.
6. Adamski M., Monteiro J.L.: Declarative Specification of System Independent Logic Controller Programs, *Proceedings of IEEE International Symposium on Industrial Electronics ISIE'96*, Warsaw, Poland, 1996, str.305-310.
7. Adamski M., Węgrzyn M., Wolański P.: Simulating and Synthesising of Reconfigurable Logic Controllers using VHDL, *42nd International Scientific Colloquium, IWK'97*, Ilmenau 22-25.09.1997, Germany, vol. I, str.522-527.
8. Adamski M., Zakrevskij A.D.: Rule - Based Specification of Reactive Logical Control Devices, *Proceedings of the Polish-German Symposium on Science Research Education, SRE'2000*, Zielona Gora, 28-29.Sept.2000, Vol.1, str.199-204.
9. Biliński K., Adamski M., Saul J.M., Dagless E.L.: Petri net based algorithms for parallel controller synthesis, *IEE Proceedings - E, Computers and Digital Techniques*, Vol. 141, No 6, Nov. 1994, str.405-412.
10. David R., Alla H.: *Petri Nets & Grafcet. Tools for modelling discrete event systems.* Prentice Hall, New York, 1992.
11. Fernandes J.M., Adamski M., Proença A.J.: VHDL Generation from Hierarchical Petri Net Specifications of Parallel Controllers, *IEE Proceedings-E, Computers and Digital Techniques*, Vol. 144, No2, March 1997, str.127-137.
12. Halang W.A., Adamski, M.: A Programmable Electronic System for Safety Related Control Applications, *Proceedings of the International Conference on Safety and Reliability, ESREL'97*, Lisbon, Portugal, European Safety and Reliability Association (ESRA), 17-20.07.1997, str.349-356.
13. Jerraya A.A., Mermet J. (Ed.): *System-Level Synthesis*, Kluwer Academic Publishers, Dordrecht 1999.
14. Kozłowski T., Dagless E.L., Saul J.M., Adamski M., Szajna J.: Parallel controller synthesis using Petri nets, *IEE Proceedings - E, Computers and Digital Techniques*, July 1995, vol.142, no.4, str.263-271.
15. Marranghello N., Mirkowski J., Biliński K.: Synthesis of Synchronous Digital Systems Specified by Petri Nets, *International Conference on Application and theory of Petri Nets ICATPN'98 – Workshop Hardware Design and Petri Nets (HWPN'98)*, Lisbon, Portugal, June 23, 1998, str.111-128.
16. Pardey J., Amroun A., Bolton M., Adamski M., Parallel Controller Synthesis for Programmable Logic, Devices. *Microprocessors and Microsystems*, Vol. 18, No 8, Oct. 1994, str.451-458.
17. Węgrzyn M.: *Hierarchiczna implementacja współbieżnych kontrolerów cyfrowych z wykorzystaniem FPGA*, Rozprawa doktorska, Politechnika Warszawska, Warszawa, 1998.
18. Węgrzyn M.: FPGA-Based Logic Controllers For Safety Critical Systems, *Proceedings of the IFAC Conference on New Technologies for Computer Control, NTCC 2001*, Hong Kong, 19-22.11.2001, str.584-589.
19. Węgrzyn M., Adamski M., Monteiro J.L.: The Application of Reconfigurable Logic to Controller Design, *Control Engineering Practice, Special Section on Custom Processes, IFAC*, Vol.6, 1998, str.879-887.

Artykuł recenzowany.