Konferencja naukowa KNWS'04
„Informatyka - sztuka czy rzemiosło"
23-25 czerwca 2004, Czocha

STEROWNIKI LOGICZNE PLC I RLC

Marian Adamski

Instytut Informatyki i Elektroniki, Uniwersytet Zielonogórski
65-246 Zielona Góra, ul. Podgórna 50

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

STRESZCZENIE

Referat dotyczy metodologii systematycznego projektowania algorytmów i programów dla sterowników logicznych LC (Logic Controller). Programy sterowania binarnego wykonywane są zazwyczaj w sposób konwencjonalny, z wykorzystaniem przemysłowego systemu komputerowego PLC (Programmable Logic Controllers). Alternatywnym, proponowanym w pracy sposobem ich realizacji jest implementacja mikroukładowa w formie sterowników RLC (Reconfigurable Logic Controllers), budowanych na przykład z nowoczesnych, rekonfigurowanych scalonych elementów cyfrowych FPGA. W pracy omówiono koncepcję hierarchicznego modularnego sterownika logicznego, opisanego za pomocą programowej sieci Petriego, będącej bardziej abstrakcyjną formą diagramów SFC, Grafcet, Grafchart oraz map stanów Statechart.

1. WPROWADZENIE

Referat dotyczy podstaw systematycznego projektowania programów dla sterowników logicznych, w tym zwłaszcza dla sterowników RLC (*Reconfigurable Logic Controllers*). Mogą być one realizowane z wykorzystaniem już ugruntowanych na rynku elektronicznym programowanych struktur logicznych FPL, takich jak rekonfigurowane scalone elementy cyfrowe FPGA [2] Najbardziej perspektywiczną platformą implementacyjną są obecnie mikrosystemy cyfrowe lub mikrosystemy cyfrowo-analogowe. W przypadku projektowania sterowników z rekonfigurowaną strukturą logiczną FPGA, profesjonaliści stosują języki opisu sprzętu, VHDL lub Verilog [17], przeznaczone do komputerowej symulacji i syntezy układów cyfrowych. Bezpośrednie przekształcanie nieformalnej specyfikacji, na ogół niepełnej, a nawet czasami sprzecznej wewnętrznie, na program w języku modelowania nie przynosi jednak spodziewanych efektów praktycznych [11].

Konwencjonalne sterowniki logiczne PLC (*Programmable Logic Controllers*) są specjalizowanymi mikrokomputerami, z góry przeznaczonymi do sterowania dyskretnymi, binarnymi procesami przemysłowymi. Sterownik RLC może być uważany za sprzętowy, reprogramowalny strukturalnie odpowiednik standardowych programowalnych sterowników logicznych PLC [3, 5, 14].

Podstawową normą dotyczącą rekomendowanych języków programowania PLC jest standard *IEC 1131-3* [3]. Priorytetowe znaczenie i akceptację w rozwiniętych krajach przemysłowych zyskuje język graficzny SFC (*Sequential Function Chart*), wywodzący się ze znacznie wcześniej opracowanego języka *Grafcet*, opartego na teorii sieci Petriego [8]. Diagram SFC, jest powiązany z podprogramami, napisanymi w języku tekstowym ST (*Structured Text*). W sposób behawioralny opisuje on funkcjonowanie modułu sterowania o ściśle określonym przeznaczeniu, Moduł reprezentowany jest graficznie za pomocą blokowego diagramu funkcyjnego FBD (*Functional Block Diagram*).

Przeznaczone początkowo tylko dla sterowników PLC języki programowania SFC i ST, wsparte strukturami modułowymi FBD mogą być również wykorzystane jako narzędzia do specyfikacji behawioralnej funkcjonowania sterownika cyfrowego, realizowanego w postaci mikrosystemu o zmiennej strukturze, układowo odzwierciedlającej zadany algorytm sterowania [4]. Współczesna inżynieria programowania przyniosła jednak bardziej efektywne środki, przeznaczone do wstępnej specyfikacji lub dokładnej analizy matematycznej, jakimi są odpowiednio zunifikowany język modelowania UML (*Unified Modelling Language*) [15] i różnorodne formy interpretowanych sieci Petriego [5, 6, 9, 12, 13, 16]. Z drugiej strony wciąż powstają różnorodne warianty języków grafowych (sieciowych), takie jak diagramy *Grafchart*, będące ulepszeniem sieci *Grafcet* i *SFC* [7]oraz diagramy *SpecCharts* i *SpecC* [10, 11], uważane za użyteczną mutację map stanów *statechart* Harela.

W referacie wykorzystuje się nowy model teoretyczny, integrujący różnorodne języki specyfikacji, który jest rozwijany na bazie teorii automatów i teorii sieci Petriego. Rozpatrywany automat (maszyna stanów) jest przydatny w komputerowym procesie układowego odwzorowywania formalnie zweryfikowanych i zoptymalizowanych algorytmów współbieżnych w programowanych strukturach logicznych FPL.

2. STEROWNIKI MODULARNE O REKONFIGUROWALNEJ STRUKTURZE

W sterowniku typu RLC algorytm sterowania binarnego realizowany jest nie jako ciąg rozkazów odzwierciedlający program, napisany np. zgodnie ze standardem IEC 1131-3, ale jako regularna struktura odpowiednio połączonych makrokomórek cyfrowych zawartych w reprogramowalnej (rekonfigurowalnej) matrycy logicznej FPGA lub CPLD, realizującej tę samą specyfikację behawioralną, co przemysłowy sterownik PLC. Rekonfigurowany sterownik kompaktowy [14] może być programowany za pośrednictwem profesjonalnych systemów CAD, wykorzystujących języki opisu sprzętu, takie jak VHDL. Potrzeba układowej implementacji programów sterowania pojawia się wyraźnie w zintegrowanym projektowaniu sprzętu i oprogramowania (Hardware-Software Codesign) [4, 9, 11, 16].

Realizacja sterowników RLC na bazie elementów FPL przynosi wymierne korzyści techniczne. Są to, między innymi: zwiększona szybkość i niezawodność urządzenia, zwartość

konstrukcji, pełniejsza ochrona przed nieautoryzowanymi zmianami w programie i jego kopiowaniem. Reprogramowalność (rekonfigurowalność) elementów FPL (CPLD i FPGA) w połączeniu z nowoczesnymi środkami CAD symulacji i syntezy w środowisku VHDL - Verilog sprawiają, że elastyczność realizowanego sterownika oraz komfort procesu projektowania są bardzo duże.

Bogaty, wciąż rozwijający się światowy przemysł elektroniczny wymusza bardzo kosztowny rozwój oprogramowania, przeznaczonego do symulacji, weryfikacji i automatycznej syntezy układów cyfrowych. Rozpowszechnienie profesjonalnego oprogramowania wśród użytkowników programowanych struktur logicznych FPL powoduje, że koszt stanowiska projektowego nie jest już zbyt wygórowany, w stosunku do jego użyteczności i uniwersalności. Nie bez znaczenia jest również fakt, że dla nowej generacji specjalistów z inżynierii komputerowej posługiwanie się językami opisu sprzętu jest już sprawą rutynową.

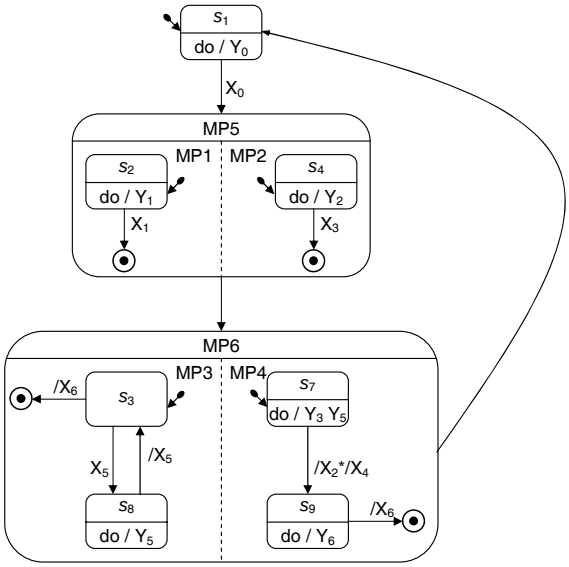
3. ROLA INŻYNIERII PROGRAMOWANIA W SPECYFIKACJI ALGORYTMÓW STEROWANIA LOGICZNEGO

Istnieje szereg sposobów opisu automatu współbieżnego (współbieżnej maszyny stanów CFSM), które po odpowiedniej adaptacji mogą być przydatne do modelowania sterowników logicznych, w tym opisanych normą IEC. Należy tutaj wymienić w szczególności opracowania zespołu prof. Gajskiego (USA), związane z modelowaniem i implementacją systemów reaktywnych [10, 11]:

- Finite State Machine with Data Path (FSMD),
- Hierarchical Concurrent Finite State Machine (HCFSM),
- Program-State Machine (PSM).

W literaturze przedmiotu bardzo często nie rozróżnia się odrębnymi nazwami diagramów przejść (grafów stanów) od opisywanych przez nie struktur dyskretnych (automatów o skończonej liczbie stanów, nazywanych również maszynami stanów). Skrót FSM w zależności od kontekstu oznacza zarówno automat skończony (maszynę stanów), jak i konkretny diagram, opisujący algorytm realizowany przez maszynę stanów, na przykład sieć działań ASM (Algorithmic State Machine) [11, 17].

Zaproponowany przez Daniela Gajskiego model automatu FSMD (automat cyfrowy z częścią operacyjną na poziomie RTL) bardzo mocno przypomina znacznie wcześniej opisane i przedstawione w Polsce i Europie Środkowej rozwiązania, znane jako automaty mikroprogramowe. W hierarchicznym współbieżnym automacie skończonym HCFSM dodatkowo dopuszczono występowanie złożonych stanów wewnętrznych o strukturze hierarchicznej, z możliwością zagnieżdżania się w nich zarówno stanów wzajemnie względem siebie sekwencyjnych, jak i wzajemnie względem siebie współbieżnych (rys. 1).



Rys. 1. Przykład hierarchicznej mapy stanów Statechart

Diagramy *SpecChart*, wprowadzone przez Gajskiego w celu opisu automatów HCFSM, są wyraźnie wzorowane na mapach stanów *statechart*, zaproponowanych przez Harela. Gajski znacznie wzbogacił tradycyjną formę graficzną diagramów *statechart* przez wprowadzenie do niej fragmentów specyfikacji, bezpośrednio zapisanych w postaci podprogramów, podanych w języku VHDL lub w języku C. Odpowiednie fragmenty programu są wykonywane, gdy rozpatrywany podstan lokalny programowej maszyny stanów PSM jest aktywny i wszystkie jej nadrzędne stany hierarchiczne (nadstany) są aktywne. Hierarchiczny, współbieżny diagram stanów programowej maszyny stanów Gajskiego HCPSM zawiera wszystkie wcześniej omówione elementy.

W literaturze przedmiotu pojawił się również wątek, polegający na skonstruowaniu grafowego opisu hierarchicznego algorytmu sterowania z wykorzystaniem różnych wariantów hierarchicznych, programowych sieci Petriego, naśladujących diagramy Gajskiego [1, 6, 13, 19]. Istotną cechą omawianych sieci jest występowanie tranzycji wywłaszczających (*exception transitions*), wymuszających przejścia między złożonymi stanami (superstanami). Elementem nowości jest wprowadzenie różnorodnych pod względem graficznym i koncepcyjnym form złożonych makromiejsc, reprezentujących makrostany (superstany). Warto znaczyć, że tranzycje wywłaszczające zostały wprowadzone przez Arzena do sieci Grafchart [8], którą można uważać za pierwszą przemysłową formę sieci sterowania, zbliżoną koncepcyjnie do hierarchicznej programowej sieci Petriego.

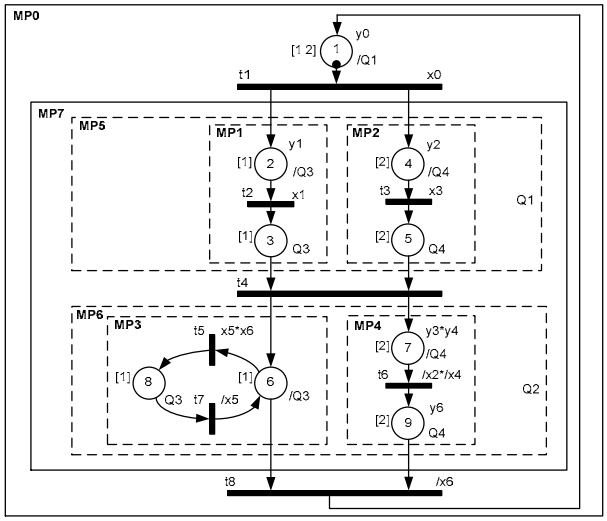
Ciekawa i użyteczna propozycja programowej sieci Petriego przeznaczonej głównie do projektowania wieloprocesorowych sterowników mikrokomputerowych została przedstawiona w monografii [6].

4. ALGORYTMICZNY WSPÓŁBIEŻNY HIERARCHICZNY AUTOMAT PROGRAMOWY

W pracy przedstawiono koncepcję algorytmicznego, współbieżnego, hierarchicznego automatu programowego ACHPSM (Algorithmic Concurrent Hierarchical Program State Machine). W swojej pierwotnej wersji podobna maszyna stanów (automat skończony), wprowadzona przez Gajskiego, odwzorowywała hierarchiczne mapy stanów Harela (diagramy *statechart*) [10]. Automat współbieżny opisywany jest algorytmiczną interpretowaną siecią Petriego sterowania. Proponuje się potraktowanie automatu (maszyny stanów) ACHPSM również jako abstrakcyjnego modelu sieci SFC oraz różnorodnych form map stanów.

Istotną cechą hierarchicznego i współbieżnego automatu skończonego jest oczywiste, naturalne odwzorowanie zadanej specyfikacji funkcjonalnej (behawioralnej) w jego elementach składowych: stanach lokalnych, lokalnych zmianach stanu, stanach globalnych i makrostanach. W programowej sieci Petriego elementom automatu odpowiadają miejsca, tranzycje, zbiory równocześnie oznakowanych miejsc (konfiguracje) oraz hierarchicznie zbudowane makromiejsca.

Najistotniejszymi cechami rozpatrywanego przez Harela automatu są współbieżność, hierarchia i rozgłaszanie. W proponowanej wersji automatu istotną rolę odgrywają również zakładane już na poziomie specyfikacji więzi przyczynowo-skutkowe, odzwierciedlające zamierzenia projektanta. Zapisywane są one poprzez cechowanie (kolorowanie) odpowiednich miejsc sieci Petriego. Wykorzystanie mechanizmu rozgłaszania jest celowo ograniczone, a jego rolę przejmują tranzycje zabraniające i zezwalające.

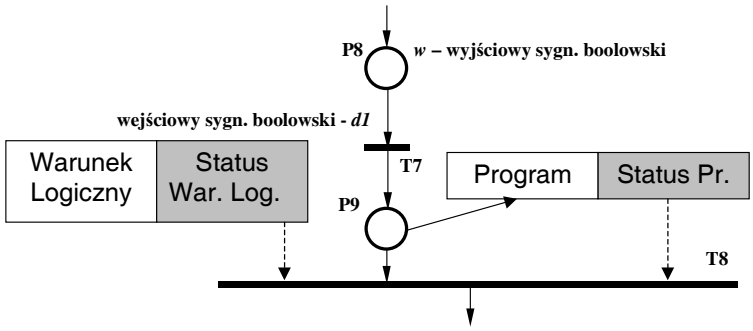


Rys. 2. Przykład modularnej kolorowanej hierarchicznej sieci Petriego

Diagramy Gajskiego przedstawiają silnie hierarchiczny współbieżny automat cyfrowy, w którym przejrzysta hierarchia wysuwa się na plan pierwszy przed nieco ukrytą współbieżnością. Sieci Petriego w znacznie wyraźniejszy sposób wskazują zależności przyczynowo-skutkowe, zwłaszcza w przypadku celowego wyodrębnienia współdziałających ze sobą współbieżnych procesów sekwencyjnych i nacechowania ich kolorami, przy czym hierarchia i modularność niestety schodzi na drugi plan. Dodatkowym środkiem pokazującym w jawny sposób relację współbieżności i sekwencyjności pomiędzy poszczególnymi makrostanami automatu (makromiejscami sieci Petriego) może być diagram hierarchii. Z tego względu postuluje się równoczesne posługiwanie się modularną, kolorowaną, hierarchiczną siecią Petriego [1], uzupełnioną kolorowanym diagramem hierarchii [12] (rys. 2).

5. TRANSFORMACJA SIECI SFC NA PROGRAMOWĄ SIEĆ PETRIEGO

Sieć SFC może być uważana za szczególny rodzaj interpretowanej sieci Petriego. Jeżeli wszystkie akcje kwalifikowane są jako akcje boolowskie typu N (sygnały logiczne) oraz tranzycje są etykietowane wyrażeniami boolowskimi zbudowanymi z symboli sygnałów binarnych, to podobieństwo z interpretowaną, sygnałową siecią Petriego sterowania jest niemal oczywiste. W przypadku ogólnym sieć SFC jest równoważna bardziej rozbudowanej, interpretowanej programowo sieci Petriego, w której warunki realizacji poszczególnych tranzycji oraz wykonywane akcje są opisane w języku ST (rys. 3). Dopiero po rozdzieleniu części sterującej od części operacyjnej, taka sieć Petriego przybiera postać klasycznej, interpretowanej sieci Petriego sterowania, abstrahowanej następnie jako wydzielony, centralny blok sterujący FBD. Pozostałe elementy sieci programowej realizowane są w postaci oddzielnych operacyjnych bloków FBD, często o charakterze bibliotecznym, w przejrzysty sposób skojarzonych z wydzielonym blokiem sterującym.



Rys. 3. Fragment programowej sieci Petriego równoważnej fragmentowi sieci SFC

Założono, że graficzna postać sieć Petriego i graf (sieć) SFC, po ewentualnych niewielkich modyfikacjach, powinny być grafami izomorficznymi, aby ułatwić wzajemną transformację opisywanych przez nie modeli. Specyfikacja jest uwiarygodniana poprzez animację

(obserwację zbioru aktywnych kroków w sieci SFC - czyli konfiguracji) oraz równoczesną obserwację przebiegów czasowych na portach wejściowych i wyjściowych wirtualnego sterownika (symulator). Dokładniejsze sprawdzenie specyfikacji behawioralnej, weryfikacja pod względem czasowym oraz ewentualna synteza układowa na poziomie RTL może się odbywać w środowisku projektowym VHDL. Podstawowe informacje i przegląd wybranych prac z dziedziny układowej implementacji sieci Petriego zawiera artykuł [5] i książka [16].

6. WNIOSKI

Funkcjonowanie sterownika PLC lub RLC modeluje się w postaci diagramu SFC, traktując go jako znormalizowaną formę programowej, interpretowanej sieci Petriego. Obok wyraźnej współbieżności i wystarczających w praktyce elementów hierarchii zawiera ona adnotacje w formie podprogramów napisanych w języku ST lub bezpośrednio w języku VHDL, które są przypisane do poszczególnych etapów dyskretnego algorytmu sterowania. Dokładna symulacja oraz eksperymentalne uwiarygodnienie są realizowane w środowisku VHDL. Po wstępnym etapie weryfikacji założona specyfikacja może być implementowana zarówno jako program w uniwersalnym, przemysłowym sterowniku logicznym PLC, jak i w postaci struktury połączeń w matrycy logicznej, znajdującej się w odpowiednio dobranym mikrosystemie cyfrowym, pełniącym funkcję sterownika RLC.

Praca naukowa finansowana ze środków Komitetu Badań Naukowych w latach 2003-2006 jako projekt badawczy nr 4 T11C 006 24.

LITERATURA

[1] M. Adamski: *Programowa sieć Petriego jako model formalny systemu cyfrowego*, IV Krajowa Konferencja Naukowa Reprogramowalne Układy Cyfrowe, RUC'2001, Politechnika Szczecińska, Szczecin 2001, ss. 139-146

[2] M. Adamski, M. Węgrzyn: *Reprogrammable controllers for reactive embedded systems*, in: Real-time programming 2003 (WRTP 2003): a Proceedings Volume from the 26th IFAC/IFIP/IEEE Workshop / M. Colnaric, M. Adamski, M. Węgrzyn, Oxford: Elsevier, 2003, ss. 39-44

[3] M. Adamski, M. Chodań: *Modelowanie Układów Sterowania Dyskretnego z Wykorzystaniem Sieci SFC*, Monografia, Wydawnictwo Politechniki Zielonogórskiej, Zielona Góra, 2000

[4] M. Adamski, Z. Skowroński: *Interpretowane sieci Petriego - model formalny w zintegrowanym projektowaniu mikroprocesorowych systemów sprzętowo-programowych*, Pomiary Automatyka Kontrola, 2003, nr 2-3, s. 17-20

[5] M. Adamski, M. Węgrzyn: *Implementacja współbieżnych algorytmów sterowania w reprogramowalnych sterownikach logicznych*, Pomiary Automatyka Kontrola, 2003, nr 2-3, s. 21-25

[6] G. Andrzejewski: *Programowy model interpretowanej sieci Petriego dla potrzeb projektowania mikrosystemów cyfrowych*, Oficyna Wydawnicza Uniwersytetu Zielonogórskiego, Zielona Góra, 2003

[7] K. E.Arzen: *Grafcet for Intelligent Supervisory Control Applications*, Automatica (Elsevier Science Ltd.), Vol.30, No.10, ss. 1513-1525, 1994

[8] R. David, H. Alla: *Petri Nets & Grafcet. Tools for modeling discrete event systems*, Prentice Hall, New York, 1992

[9] P. Eles, K. Kuchciński, Z. Peng: *System Synthesis with VHDL*, Kluwer, Boston, 1998

[10] D. Gajski, F. Vahid, S. Narayan, J. Gong: *Specification and Design of Embedded Systems*, Prentice Hall, Englewood Cliffs, New Jersey, 1994

[11] A.A. Jerraya, J. Mermet [Ed.]: *System-Level Synthesis*, Kluwer Academic Publishers, Dordrecht, 1999

[12] G. Łabiak, M. Adamski: *The method of concurrency matrix genration for statechart-based digital controllers*, Mixdes'2004, Szczecin, Poland, ISBN 83-919289-7-7

[13] N. Marranghello, de Oliveira W.L.A., F. Damianini: *Modeling a Procesor with a Petri Net Extension for Digital Systems*, Conference on Design, Analysis and Simulation of Distributed Systems - DASD 2004, Part of the ASTC. Washington DC, USA, 2004

[14] A. Milik, E. Hrynkiewicz, T. Wilk: *Rekonfigurowalny sterownik kompaktowy oparty o układy FPGA firmy XILINX*, II Krajowa Konferencja Naukowa - Reprogramowalne Układy Cyfrowe, RUC'99, Szczecin, 14-16.03.1999, ss. 293-300

[15] J. Rumbaugh, I. Jacobson, G. Booch: *The Unified Modelling Language UML. Reference Manual*, Addison Wesley, Longman, Reading, Massachusetts, 1999

[16] A. Yakovlev, L. Gomes, L. Lavagno (Ed.): *Hardware Design and Petri Nets*, Kluwer Academic Publishers, Boston, 2000

[17] M. Zwolinski: *Projektowanie układów cyfrowych z wykorzystaniem języka VHDL*, Warszawa Wydawnictwa Komunikacji i Łączności, 2002