Grzegorz ANDRZEJEWSKI, Zbigniew SKOWROŃSKI
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI

Zrównoleglanie algorytmów sterowania w systemach klasy PLC

Dr inż. Grzegorz ANDRZEJEWSKI

Dr inż. Grzegorz Andrzejewski urodził się w roku 1970. Studia ukończył w roku 1995 na Politechnice Poznańskiej. Stopień doktora z zakresu informatyki uzyskał na Politechnice Szczecińskiej w roku 2002. Jego zainteresowania naukowe ukierunkowane są na zagadnienia modelowania i syntezy systemów sterowania cyfrowego.

e-mail: g.andrzejewski@ iie.uz.zgora.pl

Dr inż. Zbigniew SKOWROŃSKI

Adiunkt w Instytucie Informatyki i Elektroniki Uniwersytetu Zielonogórskiego. Zainteresowania badawcze obejmują zagadnienia zintegrowanego projektowania cyfrowych systemów sprzętowo-programowych, ze szczególnym uwzględnieniem języków opisu sprzętu oraz systemów osadzonych, zawierających reprogramowalne układy logiczne.

e-mail: z.skowronski@iie.uz.zgora.pl

Streszczenie

W artykule przedstawiono metodykę projektowania algorytmów sterowania, w których można wydzielić procesy współbieżne. Wskazano możliwość wykorzystania profesjonalnych narzędzi wspomagających projektowanie systemów cyfrowych, a w szczególności automatycznego translatora opisu HDL do interpretowanej sieci Petriego. Przedstawiono zarys algorytmu zrównoleglania procesów sekwencyjnych, a także metodę realizacji automatów skończonych i interpretowanych sieci Petriego, z wykorzystaniem języka LD (IEC 61131-3).

Słowa kluczowe: procesy współbieżne, interpretowane sieci Petriego, systemy PLC.

Parallelising process of control algorithms in PLC systems

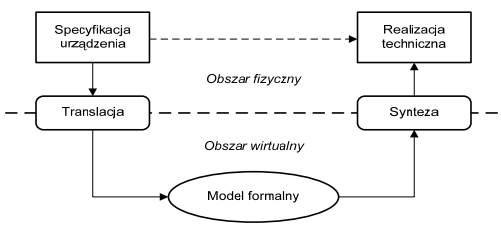
Abstract

A method for designing controller algorithms (with some concurrent processes) has been discussed. Particularly, this paper presents a usage of computer-aided design tools to solve synthesis problems of digital controllers and describes a method for transforming their specifications that are frequently given in sequential form - for example, processes in HDL into interpreted Petri nets. A paper also shows a parallelising process of control algorithms and a method of implementation of FSM and Petri nets with LD language (IEC 61131-3).

Keywords: parallel processes, interpreted Petri nets, PLC systems.

1. Modelowanie algorytmów sterowania

Większość narzędzi wspomagających prace inżynierskie i proces projektowania - CAD/CAE (ang. *Computer Aided Design*/*Computer Aided Engineering*) nie jest przystosowana do rozwiązywania problemów w obszarze fizycznym. Sytuacja taka wymusza konieczność translacji specyfikacji urządzenia do modelu formalnego (reprezentującego urządzenie najdokładniej jak to tylko możliwe) i przesunięcie działań związanych z projektowaniem z obszaru fizycznego do wirtualnego (rys. 1) [1].



Rys. 1. Uproszczony proces projektowy wykorzystujący narzędzia CAD/CAE
Fig. 1. A simplified design flow with CAD/CAE tools

Model formalny urządzenia może być wówczas analizowany, symulowany, emulowany lub transformowany, tak by w końcowej fazie procesu projektowania zostać poddanym syntezie.

Efektywność narzędzi CAD/CAE w oczywisty sposób zależy zarówno od zastosowanych algorytmów jak i modeli formalnych, na jakich te algorytmy operują. W związku z tym, zastosowany model formalny powinien charakteryzować się [1]: jednoznacznością, reprezentacją równoległości, semantyczną spójnością z językami specyfikacji i implementacji, łatwością translacji, aparatem matematycznym, dostępnością efektywnych algorytmów oraz niezależnością od specyfikacji i implementacji. Jest to szczególnie istotne w przypadku opisu równoległości, komunikacji, synchronizacji, specyfikacji przepływu danych i sterowania oraz modelowania czasu. Czytelność specyfikacji odgrywa istotną rolę w efektywności jej wykorzystania. Specyfikacja graficzna może być uważana za bardziej przejrzystą i czytelną od tekstowej, choć wielu projektantów woli wersję tekstową. Należy przy tym pamiętać, że obie te wersje są wzajemnie komplementarne.

Najpopularniejszym modelem jest automat skończony FSM (ang. *Finite State Machine*). Znajduje on szerokie zastosowanie do opisu układów sterowania, bowiem zachowanie takich układów w czasie najprościej opisuje się poprzez stany i przejścia między nimi. W automacie skończonym oprócz stanów i przejść między stanami można wyróżnić także akcje związane ze stanami (automat *Moore'a*) lub przejściami (automat *Mealy'ego*).

Def. 1. Automat skończony z wyjściami typu Moore'a:

$$FSM = \{S, F, X, Y, \delta, \lambda\} \tag{1}$$

gdzie: $S = \{s_1,\ldots,s_j\}$ jest niepustym, skończonym zbiorem stanów; F jest niepustym, skończonym zbiorem łuków skierowanych, takich że: $F \subset (S \times S)$; $X = \{x_1,\ldots,x_i\}$ jest skończonym niepustym zbiorem wejść; $Y = \{y_1,\ldots,y_k\}$ jest skończonym niepustym zbiorem wyjść; δ jest funkcją przyporządkowującą każdemu łukowi pewien podzbiór z przestrzeni wejścia δ: $F \rightarrow X$; λ jest funkcją przyporządkowującą każdemu stanowi pewien podzbiór z przestrzeni wyjścia λ: $S \rightarrow Y$.

Innym modelem formalnym, spełniającym wiele wymienionych cech, mogą być sieci Petriego. Wprowadzają one bezpośrednio i graficznie najważniejsze paradygmaty przetwarzania równoległego: sekwencyjność/zależność, konflikt (wybór niedeterministyczny) i równoległość.

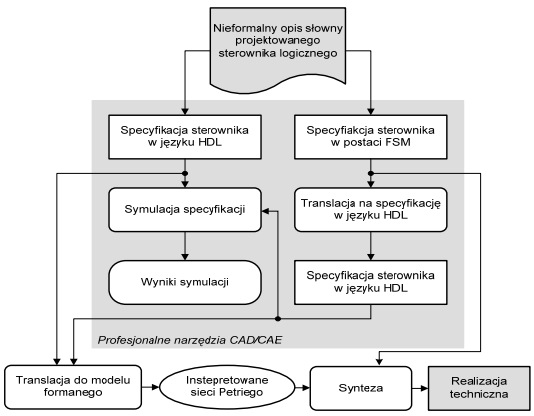
Def. 2. Interpretowana płaska sieć Petriego:

$$IPN = \{P, T, F, X, Y, m_0, \delta, \lambda\} \tag{2}$$

gdzie: P jest niepustym, skończonym zbiorem miejsc; T jest niepustym, skończonym zbiorem tranzycji ($P \cap T = \varnothing$); F jest niepustym, skończonym zbiorem łuków skierowanych, takich że: $F \subset (P \times T) \cup (T \times P)$; X i Y są alfabetami wejściowym i wyjściowym; m_0 jest funkcją znakowania początkowego m_0: $P \rightarrow N \cup \{0\}$; δ jest funkcją przyporządkowującą każdej tranzycji pewien podzbiór z przestrzeni wejścia δ: $T \rightarrow X$; λ jest funkcją przyporządkowującą każdemu miejscu pewien podzbiór z przestrzeni wyjścia λ: $P \rightarrow Y$.

2. Metodyka projektowania

Pierwszym krokiem w proponowanej w pracy syntezie sterowników logicznych (rys. 2) jest przekształcenie wejściowej specyfikacji w interpretowaną sieć Petriego, stanowiącą formalny model tej specyfikacji. Wejściowa specyfikacja najczęściej zadana jest w językach HDL (ang. *Hardware Description Languages*) lub w postaci FSM i jest niejednokrotnie tworzona przez projektantów bezpośrednio na podstawie opisu słownego.



Rys. 2. Proponowany schemat procesu projektowego sterowników logicznych
Fig. 2. Proposed design flow of logic controllers

I choć języki HDL (VHDL [6], Verilog HDL [7]) posiadają mechanizmy tworzenia opisu z równoległością, to specyfikacja wejściowa może posiadać pewną wadę wynikającą z faktu, że umysł ludzki pracuje w sposób sekwencyjny, czego naturalną konsekwencją jest trudność w opracowywaniu równoważnego opisu wielu współbieżnie przebiegających zdarzeń. W efekcie prowadzi to otrzymania opisu, który zdecydowanie odbiega od kryterium minimalnego czasu realizacji i charakteryzuje się słabym stopniem zrównoleglenia procesów w układzie [5].

Z tego względu, bardzo ważnym zagadnieniem, na które należy zwrócić uwagę w trakcie realizacji procesu projektowego jest sposób translacji wejściowej specyfikacji na model formalny. Metoda translacji powinna zapewniać duży stopień zrównoleglenia procesów w układzie i zachowywać semantykę specyfikacji behawioralnej. W pracy zastosowano interpretowane sieci Petriego [1], dzięki którym możliwe jest wprowadzenie w sposób jawny współbieżności operacji [5] i uwzględnienie konsekwencji cyklu symulacji [4], wynikającego z dyskretnego taktowania zdarzeń.

3. Zrównoleglanie procesów sterujących

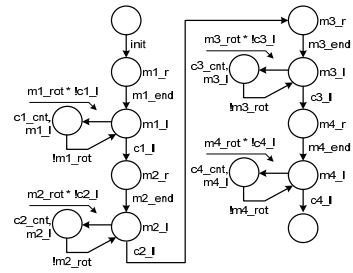
Podstawowymi założeniami, jakie przyjęto w proponowanej metodzie translacji [5] są:

- zamiana podczas translacji specyficznych instrukcji przypisania (charakterystycznych dla specyfikacji w językach HDL) na współbieżnie procesy, zawierające instrukcje sekwencyjne;
- przypisanie każdej operacji wewnątrz procesu do miejsca sieci w taki sposób, że znakowanie rozpatrywanego miejsca odpowiada wykonywaniu danej operacji. Jeśli wykonanie operacji jest warunkowe, wówczas warunek ten przypisany jest do tranzycji wejściowej miejsca reprezentującego daną operację.

Algorytm metody translacji, upraszczający się do przekształcenia operacji sekwencyjnych w interpretowaną sieć Petriego (w przypadku nie uwzględniania konsekwencji cyklu symulacji) przedstawiono poniżej. Dokładny opis poszczególnych etapów algorytmu tranaslacji dokładnie opisują prace [4, 5].

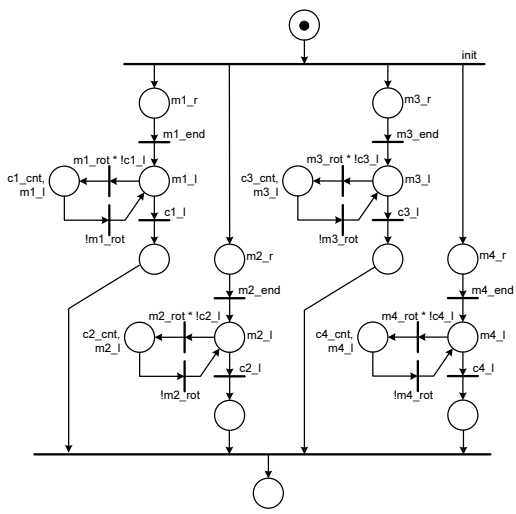
Krok 1. Specyfikacja funkcjonalna w języka HDL.
Krok 2. Przekształcenie instrukcji przypisania języka HDL na współbieżne procesy.
Krok 3. Czy są procesy do przekształcenia? Tak – idź do kroku 4; Nie – Zakończ.
Krok 4. Określenie predykatów lokalnych i globalnych.
Krok 5. Analiza zależności danych i sterowania.
Krok 6. Usunięcie nadmiarowych zależności danych lub sterowania.
Krok 7. Zdefiniowanie typu modułu sieci Petriego.
Krok 8. Połączenie modułów w sieć odzwierciedlającą specyfikację procesu.
Krok 9. Redukcja sieci Petriego. Idź do kroku 3.

Rys. 3 przedstawia fragment algorytmu kalibracji robota 3D opisanego z wykorzystaniem modelu FSM.



Rys. 3. Fragment algorytmu kalibracji robota 3D opisanego w FSM
Fig. 3. A part of calibration algorithm of 3D robot with FSM using

W wyniku zastosowania prezentowanej metody zrównoleglenia, otrzymano sieć Petriego, w której wyodrębnić można cztery procesy równoległe (rys. 4).



Rys. 4. Procesy współbieżne w algorytmie kalibracji robota 3D
Fig. 4. Parallel processes in calibration algorithm of 3D robot

4. Synteza w sterownikach klasy PLC

Dla pełniejszego zobrazowania przyjętej metody syntezy, zaproponowano szereg pojęć pomocniczych [2, 3].

Def. 3. Funkcja aktywności stanu (dla automatu FSM) lub miejsca (dla sieci Petriego) ac(s) $\rightarrow$ {true, false} jest to funkcja, która każdemu stanowi (lub miejscu) przypisuje wartość true, jeśli stan (lub miejsce) jest aktywny, oraz false w sytuacji przeciwnej.

Działanie automatu FSM można opisać poprzez określenie warunków przejścia automatu z pewnego stanu do stanu kolejnego oraz akcji związanych z tym przejściem.

Niech łuk skierowany f_i będzie parą uporządkowaną (s_m,s_n) taką, że s_m jest stanem przyporządkowanym do początku łuku, a s_n stanem przyporządkowanym do końca łuku.

Warunkami przejścia automatu ze stanu s_m do stanu s_n są:

war. 1) stan początkowy łuku musi być aktywny:
$$ac(s_m) = true$$

war. 2) warunek związany z łukiem musi być spełniony:
$$\delta(f_i) = true$$

Akcjami związanymi z przejściem automatu ze stanu s_m do stanu s_n są:

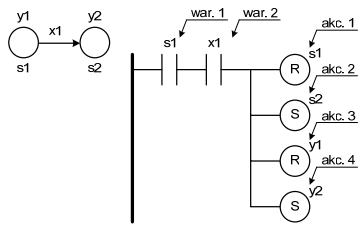
akc. 1) stan początkowy jest deaktywowany: $ac(s_m) := false$

akc. 2) stan końcowy jest aktywowany: $ac(s_n) := true$

akc. 3) wyjścia związane ze stanem s_m są deaktywowane: $\forall(y \in \lambda(s_m)) : y := false$

akc. 4) wyjścia związane ze stanem s_n są aktywowane: $\forall(y \in \lambda(s_n)) : y := true$

Tak sformułowany opis działania automatu można w prosty sposób zrealizować w dowolnym języku programowania sterowników klasy PLC. Na rys. 5 zamieszczono przykład ilustrujący zasadę realizacji z wykorzystaniem diagramów drabinkowych LD (ang. *Ladder Diagram*) [8].



Rys. 5. Realizacja automatu FSM w języku LD
Fig. 5. A LD implementation of FSM

Dla interpretowanych płaskich sieci Petriego dodatkowo można zdefiniować zbiory miejsc wejściowych i wyjściowych tranzycji.

Def. 4. Zbiorem miejsc wejściowych tranzycji t nazywamy zbiór P_t^{in}, taki że:

$$P_t^{in} = \{p \in P : (p,t) \in F\} \tag{3}$$

Def. 5. Zbiorem miejsc wyjściowych tranzycji t nazywamy zbiór P_t^{out}, taki że:

$$P_t^{out} = \{p \in P : (t,p) \in F\} \tag{4}$$

Działanie sieci wyznaczone jest poprzez warunki przygotowania tranzycji oraz akcje związane z jej wykonaniem.

Warunki przygotowania tranzycji t:

war. 1) każde miejsce wejściowe tranzycji t musi posiadać znacznik: $\forall(p \in P_t^{in}) : ac(p) = true$

war. 2) warunek logiczny związany z tranzycją musi być spełniony: $\delta(t) = true$

Akcje związane z wykonaniem tranzycji t:

akc. 1) wszystkie miejsca wejściowe tranzycji są deaktywowane (pozbawiane znaczników): $\forall(p \in P_t^{in}) : ac(p) := false$

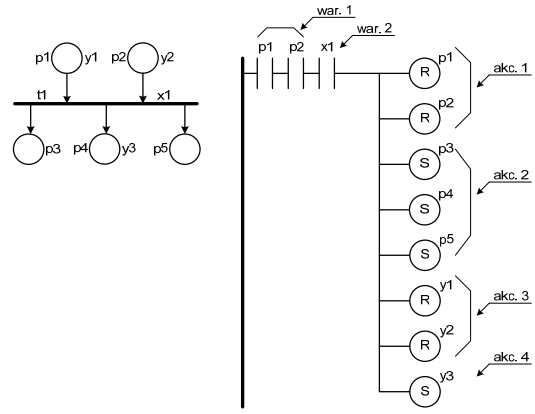
akc. 2) do każdego miejsca wyjściowego tranzycji są wprowadzane znaczniki (są aktywowane): $\forall(p \in P_t^{out}) : ac(p) := true$

akc. 3) wszystkie sygnały wyjściowe związane ze wszystkimi miejscami wejściowymi tranzycji są deaktywowane: $\forall(p \in P_t^{in}) : \forall(y \in \lambda(p)) : y := false$

akc. 4) wszystkie sygnały wyjściowe związane ze wszystkimi miejscami wyjściowymi tranzycji są aktywowane: $\forall(p \in P_t^{out}) : \forall(y \in \lambda(p)) : y := true$

Tak sformułowany opis działania sieci Petriego można zrealizować w sposób analogiczny, do realizacji automatu FSM (rys. 6).

Przykłady z rys. 3 i 4 poddano syntezie z wykorzystaniem prezentowanej metody realizacji. Jako platformę realizacyjną przyjęto sterownik PLC klasy VersaMax Micro IC200UAL005 firmy GEFanuc. Jako narzędzie CAD/CAE wykorzystano Cimplicity Machine Edition v.5.0.



Rys. 6. Realizacja sieci Petriego z wykorzystaniem języka LD
Fig. 6. A LD implementation of interpreted Petri net

Wyniki syntezy obydwu realizacji przedstawia tabela 1, w której jako kryterium porównania przyjęto zużycie pamięci programu użytkownika, oraz pamięci zmiennych wewnętrznych.

Tab. 1. Zestawienie wyników syntezy
Tab. 1. Results of synthesis

	Pamięć programu			Zmienne wewn.		
	całk.	użyta	%	Całk.	użyta	%
FSM	18kB	412B	2,3	64	14	22
sieć Petriego	18kB	448B	2,5	64	18	28

5. Podsumowanie

W artykule przedstawiono metodę realizacji algorytmów sterowania dla potrzeb projektowania systemów klasy PLC. Wskazano potrzebę zrównoleglania procesów sterujących oraz sposób jej wykonania. Poddano także syntezie przykładowy algorytm kalibracji robota 3D.

Uzyskane wyniki mogą sugerować, iż zrównoleglenie procesu sekwencyjnego prowadzi do wzrostu zużycia zasobów wewnętrznych sterownika. Sytuacja ta zmieni się w przypadku opisu systemów skomplikowanych o naturze współbieżnej. Warto jednak zauważyć, że nawet dla prostych przykładów strata jest niewielka (poniżej 1% dla pamięci), a korzyść może objawić się w skróceniu czasu wykonywania całego procesu sterowania.

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

6. Literatura

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

[2] G. Andrzejewski: Program model of Petri net, Proc. of CAD DD 2001, Minsk, Belarus, 2001, Vol. 1, s. 87-92

[3] G. Andrzejewski: Programowy model interpretowanej sieci Petriego dla potrzeb projektowania mikrosystemów cyfrowych, Zielona Góra, Oficyna Wydawnicza UZ, 2003

[4] Z. Skowroński: Problem translacji specyfikacji funkcjonalnej układów cyfrowych w języku VHDL na interpretowaną sieć Petriego w zintegrowanej syntezie systemów sprzętowo-programowych, Mat. konf. RUC 2001, Szczecin, 2001, s. 103-113

[5] Z. Skowroński: Metoda transformacji specyfikacji behawioralnej układów cyfrowych na sieci Petriego w syntezie systemowej, Mat. konf. KNWS' 05, Złotniki Lubańskie, 2005, s. 55-65

[6] IEEE Standard VHDL Language Reference Manual, IEEE, New York, 1988

[7] IEEE Standard HDL Based on the Verilog® HDL, IEEE, New York, 1996

[8] IEC 61131 Programmable Controllers, New York, 1993

Artykuł recenzowany