Adam MILIK
POLITECHNIKA ŚLĄSKA, INSTYTUT ELEKTRONIKI,
ul. Akademicka 2 A, 44-100 Gliwice

Synteza diagramu LD z operacjami arytmetycznymi przeznaczona dla układów FPGA

Dr inż. Adam MILIK

Ukończył studia na Wydziale Automatyki, Elektroniki i Informatyki Politechniki Śląskiej. Pracę doktorską obronił w 2003 r. Jest adiunktem w Instytucie Elektroniki Politechniki Śląskiej. Jego zainteresowania naukowe to układy logiki programowalnej, sterowniki programowalne, modelowanie i synteza złożonych układów sprzętowo-programowych.

e-mail: adam.milik@polsl.pl

Streszczenie

W artykule przedstawiono automatyczną metodę syntezy układu sterowania danego w postaci diagramu stykowego LD lub listy instrukcji IL do sprzętowego układu sterowania implementowanego w układzie FPGA. Zaproponowana metoda pozwala uzyskać sprzętowy układ sterowania zachowujący sekwencyjne własności przetwarzania wynikające z zapisu LD i IL. Przedstawiony algorytm syntezy pozwala na dokonanie syntezy operacji logicznych i arytmetycznych. Istotnymi celami opracowanego algorytmu jest masowe przetwarzanie, redukcja cykli obliczeniowych oraz odwzorowanie w ograniczonej liczbie zasobów operacji arytmetycznych.

Słowa kluczowe: sterownik programowalny, diagram stykowy, LD, FPGA, synteza logiczna wysokiego poziomu, arytmetyka, układy rekonfigurowane.

LD synthesis with arithmetic operations for FPGA

Abstract

The paper presents the synthesis algorithm of a ladder diagram (LD) or instruction list (IL) into a reconfigurable logic controller implemented in FPGA [5, 8, 9]. The algorithm incorporates synthesis of Boolean and fixed point arithmetic operations. It utilizes the intermediate form of the data flow graph (DFG) [4, 6]. PLCs introduce variable dependencies caused by serial processing of LD (Fig. 1). It has been proved that appropriate distribution of feedback signals allows implementing LD logic dependencies during a single calculation cycle (Fig. 2). The LD diagram is compiled into DFG that records variable dependencies. The presented optimization allows reducing the controller complexity and its response time in comparison to solutions presented in [2, 3] (Fig. 3). Arithmetic operations introduce larger implementation complexity and require more time to calculate than logic operation. The DFG generated from LD or IL is used for scheduling and mapping (Fig. 4). The scheduling and mapping procedure assumes the limited number of arithmetic resources while logic operations are allocated without constraints. The scheduling procedure takes into account operation execution timing (Fig. 4C). The obtained circuit after scheduling with arithmetic operations may require more than one cycle to complete all operations in comparison to the model limited only to logic operations. The presented synthesis procedure enables obtainment of fully functional hardware implementation of the controller given by LD or IL with massively parallel processing and a very short response time (1 to several clock cycles).

Keywords: PLC, ladder diagram, LD, FPGA, high level logic synthesis, arithmetic, reconfigurable hardware.

1. Wstęp

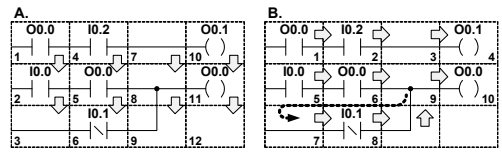
Sterowniki programowalne są konstruowane w oparciu o koncepcję układu mikroprogramowalnego. W koncepcji tej można wyróżnić dwa zasadnicze elementy, platformę sprzętową oraz program. Platforma sprzętowa implementuje elementarne operacje logiczne i arytmetyczne oraz układ pobierania i wykonywania instrukcji. Program natomiast jest uporządkowanym ciągiem instrukcji, realizuje zadania sterowania. Przetwarzanie algorytmu sterowania odbywa się w sposób szeregowy-cykliczny. Przedstawione podejście konstrukcji sprzętu i oprogramowania jest bardzo proste i wygodne w użyciu. Pozwala na bardzo łatwe wprowadzanie zmian w programie sterowania, poprzez wymianę instrukcji w pamięci mikroprogramu. Nowoczesne sterowniki należą do zaawansowanych elektronicznych układów obliczeniowych o wieloprocesorowej strukturze [1]. Czy istnieje możliwość zwiększenia ich wydajności i poprawy parametrów obliczeniowych?

Istotną jakościowo zmianę można zaproponować przez wykorzystanie rekonfigurowanej platformy sprzętowej, jaką oferują ciągle rozwijające się układy programowalne FPGA [5, 8]. Ich pojemność logiczna pozwala na implementację systemów mikroprocesorowych. Takie podejście pozwala badać modyfikacje konstrukcji, jednak przetwarzanie w dalszym ciągu odbywa się w sposób szeregowy. Układ programowalny wykorzystany do implementacji dedykowanego układu o równoległej strukturze realizowanych działań, pozwoli na zapewnienie ogromnej wydajności obliczeniowej, oferując ściśle określone parametry czasowe działania.

Istotną zaletą systemów mikroprocesorowych jest łatwość ich programowania. Projektowanie układów sprzętowych mimo ciągłego rozwoju narzędzi syntezy logicznej, jest procesem bardzo złożonym wymagającym znacznej wiedzy i doświadczenia. Rozwój technik i narzędzi upraszczających ten proces, ma istotne znaczenie [2, 3, 5, 8, 9]. Artykuł prezentuje opracowany algorytm syntezy rekonfigurowalnego sterownika logicznego, zdolnego do przetwarzania zmiennych dwustanowych oraz stałoprzecinkowych operacji arytmetycznych. Nowością jest również włączenie procesu syntezy operacji arytmetycznych, pozwalającego na efektywne wykorzystanie zasobów obliczeniowych.

2. Programowanie sterowników PLC

Metody programowania sterowników logicznych zostały zapożyczone ze schematów drabinkowych, wykorzystywanych do projektowania układów przekaźnikowych. Metody te zostały ujednolicone przez normy IEC1131 oraz EN61131. W części 3 normy określają metody programowania układów sterowania. W niniejszych rozważaniach, jako podstawowy sposób programowania układów sterowania przyjęto listę instrukcji (IL) oraz diagram drabinkowy (LD). Języki wysokiego poziomu takie jak tekst strukturalny (ST) oraz sekwencyjny diagram funkcyjny (SFC), mogą zostać skompilowane do listy instrukcji stanowiącej podstawę działania sterownika programowalnego. Niezwykle istotnym jest, aby rekonfigurowalny sterownik zapewniał identyczne zachowanie jak sterowniki programowalne wraz z odpowiednimi metodami programowania.



Rys. 1. Schematy analizy diagramów stykowych: A. kolumnowy, B. wierszowy
Fig. 1. The LD diagram analysis methods: A. column based, B. row based

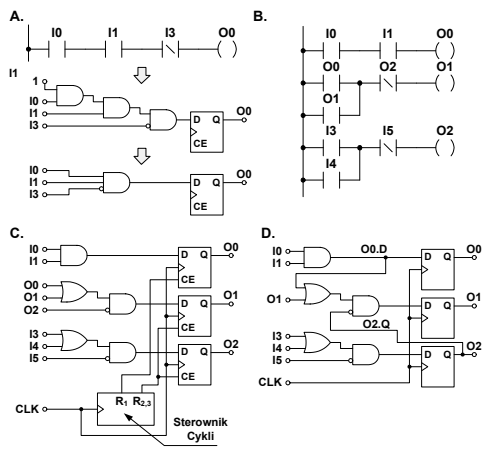
W układach sterownikowych wykorzystuje się przetwarzanie szeregowo-cykliczne z pamięcią obrazu procesu. Cykl rozpoczyna się od pobrania stanu wejść do pamięci procesu, następnie wykonywane są obliczenia, po czym dokonuje się dystrybucji wyników. Diagramu LD mimo iż reprezentuje schemat elektryczny, jednak szeregowa natura przetwarzania nie pozwala uzyskać całkowitej równoległości realizacji operacji, co ujawnia się w zachowaniu sterowanego układu. Program sterowania podzielony jest na sieci

(network), które wykonywane są sposób szeregowy zgodnie z kolejnością deklarowania. Poszczególne diagramy można analizować w sposób kolumnowy (Modicon rys. 1A) lub wierszowy (Siemens, rys. 1B).

3. Synteza operacji logicznych

Rekonfigurowalny sterownik programowalny przetwarza zadania sterowania w sposób równoległy. Istotnym jest stworzenie modelu odwzorowania diagramu LD, zachowującego własności przetwarzania sekwencyjnego zgodne ze sposobem działania układów sterownikowych. Metody przedstawione w [2, 3] proponują budowę i wykorzystanie grafów jednoczesności i zależności. Graf jednoczesności, reprezentuje wszystkie węzły (cewki), które mogą zostać wyznaczone niezależnie. Skierowany graf zależności, przedstawia kolejność wykonania poszczególnych szczebli diagramu. Na podstawie wyznaczonych grafów, jest generowana struktura sprzętowa sterownika. W przypadku zagnieżdżonych wyrażeń logicznych, wymagane jest wykonanie wielu cykli obliczeniowych w celu zachowania zależności wyznaczenia wyniku.

Czy można uprościć proces syntezy binarnego układu sterowania na podstawie diagramu LD, pozwalającego osiągnąć porównywalne lub lepsze rezultaty?

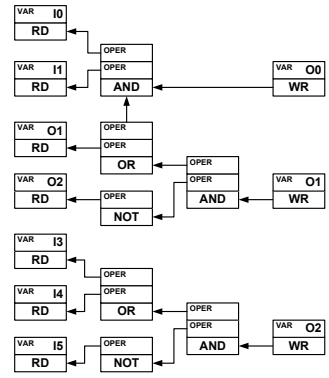


Rys. 2. Algorytm syntezy operacji logicznych z diagramu LD. A. Konwersja i optymalizacja operacji, B. przykładowy układ sterowania, C. Układ otrzymany na podstawie [2, 3]. D. Zaproponowana metoda syntezy

Fig. 2. The LD based logic operations synthesis algorithm A. Conversion and optimization. B. Example of complex network, C. Circuit obtained based on [2, 3], D. Circuit obtained when applying the proposed synthesis method

Synteza LD oraz IL opisana przy użyciu reguł syntezy automatów sekwencyjnych znacząco upraszcza proces syntezy. Opis IL i LD ma charakter sekwencyjnego przetwarzania zmiennych, z zachowaniem ich kolejności deklaracji w wyrażeniach (styki) i przypisaniach (cewki). Na rysunku (rys. 2) została ujęta całościowo zaproponowana koncepcja syntezy operacji logicznych. Pierwszym etapem jest konwersja diagramu stykowego do postaci wyrażeń logicznych (rys. 2A). Styki zamieniane są na bramki iloczynu, węzły z kolei na bramki sum. Następnie dokonuje się optymalizacji logicznej. Na rysunku pokazano propagację stałych i łączenie identycznych bramek w struktury wielowejściowe. Na rysunku (rys. 2B) przedstawiono schemat stykowy, w którym występują zależności sekwencyjne zmiennych wyjściowych. W celu zapewnienia wykonania zadań zgodnego ze sterownikiem logicznym, należy zapewnić odpowiednią kolejność wyznaczania wyników. W celu zachowania kolejności obliczeń [2, 3], należy zastosować układ sterowania cyklem obliczeniowym (rys. 2C). Dokona on zapisu zmiennych w odpowiedniej kolejności. Można zauważyć, możliwość zastąpienia układu sterującego sekwencją zapisu wyników poprzez odpowiednią dystrybucję zmiennych. Wykorzystując pojęcia stanu następnego i bieżącego dla zmiennych rejestrowych, cykl obliczeniowy zostanie skrócony do pojedynczego cyklu zapisu wyników, pozwalając na eliminacje układu sterowania sekwencją obliczeniową (rys. 2D).

W celu efektywnego zautomatyzowanego przetwarzania informacji wykorzystano acykliczny graf skierowany, opisujący operacje [4, 6]. Stanowi on efektywną formę reprezentacji przydatną przy optymalizacji jak i odwzorowaniu układowym. Węzły odwzorowują elementarne operacje. Zbiór operacji elementarnych obok funkcji logicznych, został uzupełniony o operacje odczytu i przypisania wartości zmiennym. Wskaźniki umieszczone w tablicy argumentów każdego węzła, wskazują na węzły będące argumentami operacji. Graf powstaje w sposób sekwencyjny sukcesywnie podczas przetwarzania LD lub IL, zachowując szeregowo cykliczną własność przetwarzania. Warto w tym miejscu zwrócić uwagę na sposób budowania listy zmiennych. Odwołanie się do zmiennej poprzez styk, wymaga odczytania jej wartości. Jeżeli zmienna nie istnieje zostaje utworzona a pole wartości wskazuje odczyt jej samej (bieżący stan rejestru). Licznik referencji odczytu zmiennej zostaje zwiększony każdorazowo po odwołaniu do zmiennej. Przypisanie wartości do wyjścia (symbol cewki), wymaga zadeklarowania zapisu wartości zmiennej. Powstający wynik w toku obliczeń zostaje przypisany do zmiennej. Pole zapisu wartości uzyskuje nowy wskaźnik do węzła operacji, która będzie nadawała wartość zmiennej. Licznik referencji węzła aktualnie sterującego zmienną, zostaje pomniejszony. Jeżeli licznik referencji osiągnie 0 oznacza to, że węzeł nie jest wykorzystywany i może zostać usunięty. Proces usuwania ma charakter iteracyjny. Istotną cechą zaproponowanej metody jest wbudowana sekwencyjność, pozwalająca na automatyczne uszeregowanie operacji w czasie przetwarzania diagramu a także usunięcie pustych przypisań. Graf pozwala również dokonać optymalizacji logicznych takich jak, grupowanie identycznych funkcji wielowejściowych, propagacja stałych. Przykładowy graf dla opisu z (rys. 2B) został pokazany na rysunku (rys. 3).



Rys. 3. Skierowany diagram operacji otrzymany w wyniku kompilacji przykładu z rys. 2B

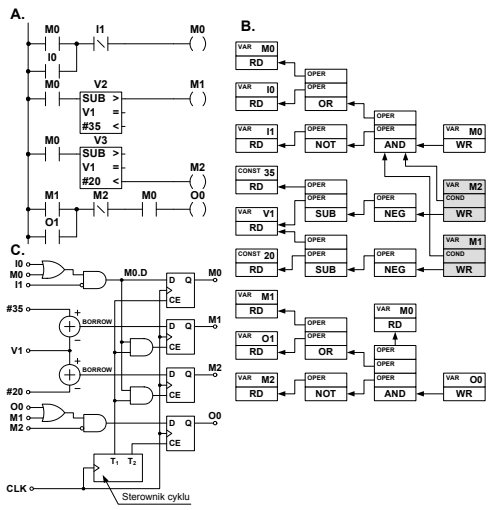
Fig. 3. The data flow graph obtained after compilation of the circuit from Fig. 2B

4. Operacje arytmetyczne

Obok funkcji logicznych, sterownik implementuje zestaw operacji arytmetycznych i arytmetyczno-logicznych. Układy FPGA zostały wyposażone w wiele rozszerzeń, ułatwiających implementację sumatorów oraz dedykowane moduły arytmetyczne, takie jak multiplikatory czy jednostki MAC. W odróżnieniu od operacji logicznych, operacje arytmetyczne wymagają podczas implementacji więcej zasobów logicznych a także liczba dedykowanych elementów arytmetycznych jest ograniczona. Implementacja operacji arytmetycznych wymaga rozłożenia działań w czasie oraz rozdziału zasobów pomiędzy poszczególne operacje. Do dalszej implementacji przyjęto architekturę typu Spartan 3 [7] a także długość słowa liczby całkowitej nie przekraczającego 18 bitów (długość słowa argumentu multiplikatora). Algorytm syntezy musi brać pod uwagę ograniczoną liczbę multiplikatorów. Koszt zbudowania sumatora jest porównywalny do kosztu zbudowania multipleksera o 2 wejściach informacyjnych w układach FPGA o 4 wejściowych generatorach LUT. Multipleksowanie argumentów sumatora nie zawsze pozwoli zmniejszyć liczbę zasobów koniecznych do implementacji układu. Synteza układu sterowania

zawierającego elementy arytmetyczne, wymaga rozbudowania istniejących reguł konstrukcji formy pośredniej, do której będzie kompilowany program oraz na jej podstawie, reguł generacji odpowiedniej struktury układowej. W układach sterownikowych operacje arytmetyczne są obliczane warunkowo. Wynik operacji będzie przypisywany warunkowo do zespołu rejestrowego. W poprzednim rozdziale wykazano możliwość realizowania operacji logicznych w sposób jednocyklowy. Czas propagacji sygnału przez elementy arytmetyczne jest porównywalny do propagacji sygnału przez wielowarstwowy układ logiczny. Układy FPGA posiadają znaczną liczbę rejestrów. Biorąc pod uwagę powyższe przesłanki, zdecydowano się na realizację pojedynczej operacji arytmetycznej lub arytmetyczno-logicznej w ciągu jednego cyklu sygnału zegarowego. Wyjątkiem jest operacja dzielenia, realizowana sekwencyjnie.

Niech będzie dany układ sterowania przedstawiony na rysunku (rys. 4). Układ realizuje funkcję histerezy o dwóch progach przełączenia zadeklarowanych przez zmienne VL i VH. Operacje arytmetyczne są wykonywane warunkowo i zależne od stanu wyjścia znacznika M0. Do realizacji funkcji porównania zastosowano układy odejmujące.



Rys. 4. Algorytm syntezy operacji artmetycznych i logicznych z diagramu LD. A. Diagram LD, B. Graf operacji, C. Implementacja układowa
Fig. 4. The synthesis algorithm of arithmetic and logic operations from LD: A. Input LD, B. Data flow graph, C. Hardware implementation

Zbiór węzłów grafu operacji uzupełniono o węzły operacji arytmetycznych. Złożone bloki (np. blok różnicy z wyjściami relacji) są wprowadzane w postaci poddiagramu zbudowanego z operacji elementarnych. W celu uzyskania warunkowej realizacji obliczeń arytmetycznych, wprowadzono węzeł o warunkowym zapisie wyniku operacji arytmetycznej uzależniony od warunku wyzwalającego blok. Węzeł ten jest przeznaczony do zapisu wyniku operacji arytmetycznych. W czasie budowania diagramu zmienna reprezentowana przez węzeł zapisu warunkowego może zostać wykorzystana tylko w postaci rejestrowej. Nałożone ograniczenie jest konieczne w celu uniknięcie tworzenia długich łańcuchów operacji arytmetycznych o znacznym czasie propagacji. Podobnie jak dla operacji logicznych sprawdza się liczbę referencji zmiennej. Węzły o zerowej liczbie referencji są usuwane.

Odwzorowanie układowe zapisanego algorytmu jest prowadzone na podstawie utworzonego diagramu operacji. W czasie odwzorowania ograniczono liczbę operacji mnożenia i dzielenia jakie mogą zostać wykorzystane jednocześnie. Proces odwzorowania oparto o połączone własności algorytmów listowego i ASAP (możliwie najwcześniejsze wykonanie operacji w ciągu). Dla odwzorowania sprzętowego kluczowym jest moment zapisu zmiennej. Węzły reprezentujące odczyt zmiennych zostają oznaczone indeksem cyklu początkowego (0). Pozostałe węzły posiadają nieprzypisany indeks cyklu. Następnie w sposób iteracyjny przeszukiwane jest drzewo operacji. Dla węzłów operacji, indeks kolejności wykonania jest sumą czasu wykonania oraz największego indeksu przypisanego argumentom operacji. Czas wykonania operacji jest wyrażony w cyklach. Węzeł zapisu zmiennej dziedziczy indeks wskazywanej operacji. Jeżeli z daną zmienną jest skojarzony jej odczyt, otrzymuje on indeks wykonania powiększony o 1 wskazując na moment czasu, od którego zmienna uzyskała poprawną wartość. Dla operacji arytmetycznych, której moduły występują w ograniczonej liczbie zasobów, tworzona jest lista przypisania. Jeżeli w danym cyklu wyczerpane zostały zasoby, operacja zostaje przypisana do najbliższego cyklu dysponującego wolnym pożądanym zasobem. W przypadku elementów wymagających wielu cykli zegarowych, przypisanie zasobu zostaje rozmieszczone w kolejnych cyklach niezbędnych do wykonania operacji. Po przypisaniu indeksów operacjom, rozpoczyna się cykl właściwej generacji struktury układowej. Najwyższy numer nadanego cyklu oznacza całkowitą liczbę cykli koniecznych na wykonanie obliczeń. Na podstawie liczby cykli generowany jest układ sterownika cykli obliczeniowych, synchronizującego przepływ zmiennych w układzie. Przeglądnięcie list elementów o ograniczonej liczbie instancji, pozwala określić maksymalną liczbę wykorzystanych instancji w projekcie. Na podstawie diagramu generowana jest struktura realizująca odpowiednie funkcje logiczne i arytmetyczne. Dla funkcji kombinacyjnych stosuje się przypisania ciągłe. Dla układów występujących w ograniczonej liczbie zasobów, w opisie zadeklarowane zostają tylko wykorzystywane instancje. Następnie generowany jest multiplekser wyboru argumentów, wybierający odpowiednie argumenty w odpowiednich momentach cyklu obliczeniowego.

5. Podsumowanie

W artykule przedstawiono kompleksowe rozwiązanie narzędzia syntezy układu sterowania, na podstawie diagramu stykowego lub listy instrukcji. Umożliwia generację sprzętowego układu sterowania charakteryzującego się równoległością realizacji obliczeń, przy zachowaniu własności zapisu LD i IL. Algorytm uwzględnia ograniczoną liczbę zasobów sprzętowych występujących w systemie. Planuje się dalszy rozwój algorytmu poprzez wprowadzania dodatkowych optymalizacji takich jak, łączenie operacji wspólnych, optymalizacje algebraiczne i logiczne.

Praca naukowa finansowana ze środków na naukę w latach 2010-2012 jako projekt badawczy 5391/B/T02/2010/38.

6. Literatura

[1] Chmiel M., Hrynkiewicz E.: Remarks on parallel bit-byte cpu structures of programmable logic controllers, Design of Embedded Control Systems, M. W. Adamski M. A., A. Karatkevich, Ed. Springer Science + Business Media Inc., 2005, str. 231–242.
[2] Du D., Liu Y., Guo X., Yamazaki K. and Fujishima M.: Study on LD-VHDL conversion for FPGA-based PLC implementation, The International Journal of Advanced Manufacturing Technology, vol. 40, str. 1181–1190, 2009.
[3] Du D., Xu X. and Yamazaki K.: A study on the generation of siliconbased hardware PLC by means of the direct conversion of the ladderdiagram to circuit design language. Springer London, 2010, vol. 49.
[4] Gajski D., Dutt N., We A., Lin S.: High-Level Synthesis Introduction to Chip and System Design, Kluwer Academic Publishers, 1994.
[5] Milik A.: High Level Synthesis – Reconfigurable Hardware Implementation of Programmable Logic Controller, PDeS 2006 Programable Devices and Embedded Systems, Brno, 2006.
[6] Wirth N.: Algorytmy + Struktury Danych = Programy, WNT, Warszawa 1989.
[7] Xilinx, DS-099 Spartan-3 FPGA Family, ver.2.1, Xilinx, 2006.
[8] Yadong L., Kazuo Y., Makoto F., Masahiko M.: Model-driven programmable logic controller design and FPGA-based hardware implementation, ASME International Design Engineering Technical Conferences and Computers and Information in Engineering Conference - DETC2005, 2005, str. 81–88.
[9] Welch J.: Translating unrestricted relay ladder logic into Boolean form, Computers in Industry, vol. 20, str. 45–61, 1992.

otrzymano / received: 17.04.2012
przyjęto do druku / accepted: 01.06.2012 *artykuł recenzowany / revised paper*