Adam MILIK
POLITECHNIKA ŚLĄSKA, INSTYTUT ELEKTRONIKI

Sprzętowo wspomagana, selektywna realizacja programu w sterowniku logicznym

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

Artykuł przedstawia metodę selektywnej realizacji programu sterowania. W programie sterowania podzielonym na bloki realizowane są te, których argumenty uległy zmianie od ostatniego cyklu obliczeniowego. Elementem niezbędnym do selektywnej realizacji programu jest pamięć procesu z mechanizmem wykrywania różnic w jej zawartości. Powyższe zadanie może zostać zrealizowany w sposób efektywny, przy wykorzystaniu unikalnych cech układów FPGA.

Słowa kluczowe: FPGA, sterownik logiczny, pamięć obrazu procesu.

Hardware Supported Selective Control Program Execution In A PLC

Abstract

The paper presents method of selective control program execution by a PLC. From entire program are executed only these blocks that variables have changed since last calculation. In order to determine program blocks that require recalculation in current program loop specific hardware support is used. The memory content difference detection unit allow to determine changes in memory content since last comparison. General idea of the change detector is presented in Fig. 2. Variables that are used by program block usually are stored in various location of process image memory. In order to precisely determine execution condition change detector should watch desired regions of memory while other part should not be considered. Following approach require to equip change detector with map of watched memory regions (Fig. 3). Finally change detector units together with process memory has been implemented in an FPGA device. Two different constructions that utilize distributed RAMs and block RAMs were considered. Requirements for general purpose resources for both solution is comparable. Using dedicated resources of an FPGA (Block-RAMs) roughly 12 times more regions can be defined than using standard approach. Presented results show that not only logic synthesis but also system level design determines efficiency of implementation. Works over efficient PLC design are on going and presented detector is a part of the larger design of hardware accelerated event driven PLC CPU.

Keywords: FPGA, PLC, Process Image Memory.

1. Wprowadzenie

Zdolność obliczeniowa sterownika logicznego jest określona przez czas konieczny do wykonania tysiąca instrukcji [5, 6]. Im ten parametr jest mniejszy, tym szersze są możliwości zastosowania sterownika do różnych zadań, w szczególności o znacznych wymaganiach czasowych. Zaprojektowanie i zbudowanie jednostki centralnej, która będzie wykonywała program sterowania w możliwie najkrótszym czasie staje się istotnym problemem i zadaniem do rozwiązania [1…4].

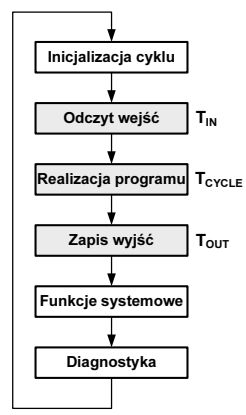
Sterownik programowalny wykonuje program w sposób szeregowo cykliczny. Instrukcje są pobierane i wykonywane w nieskończonej pętli co zapewnia uzyskanie ciągłego sterowania. W najogólniejszym przypadku, czas reakcji sterownika na zdarzenie zewnętrzne nie przekracza czasu dwukrotnego obiegu pętli programowej.

Przedstawione podejście do realizacji programu sterowania jest niezależne od zmian sygnałów dla których wyznaczane jest sterowanie. W przypadku zmiennych, które nie uległy zmianie od poprzedniego cyklu obliczeniowego, jednostka centralna realizuje obliczenia których wynik jest już znany. Obserwacja ta stanowi podstawę umożliwiającą przedstawienie zmodyfikowanego podejścia do realizacji programu sterowania.

Nowa koncepcja bazuje na zdarzeniowej realizacji programu sterowania. Oparta jest na realizacji wybranych fragmentów programu sterowania, pod warunkiem wystąpienia zmian w argumentach (sygnałach wejściowych, wyjściowych lub znacznikach) danego bloku programowego. Komercyjnie dostępne sterowniki programowalne mogą implementować zdarzeniową realizację programu sterowania na drodze programowej. Implementacja programowa wymusza rozbudowanie programu sterowania o dodatkowe fragmenty analizujące warunkową realizację poszczególnych bloków, co potencjalnie może wydłużyć czas obiegu pętli programowej. Sterownik oparty o zdarzeniową realizację programu wymaga odpowiedniego wsparcia dedykowanych układów sprzętowych, umożliwiających wykrycie zmian i natychmiastowe wybranie odpowiednich bloków programowych do realizacji. Pozwoli to uzyskać maksymalne skrócenie czasu odpowiedzi poprzez realizację tylko niezbędnych fragmentów programu.

2. Realizacja programu sterowania

Sterownik programowalny podczas realizacji programu sterowania wykonuje następujące operacje: odczyt wejść, realizacja programu sterowania, aktualizacja wyjść [5]. Obok przedstawionych operacji, które są bezpośrednio widoczne przez użytkownika, sterownik realizuje zadania wymagane przez system operacyjny (rys. 1) [6].



Rys. 1. Cykl pracy sterownika programowalnego
Fig. 1. Operation cycle of a PLC

W przedstawionym podejściu, program jest realizowany niezależnie od zmian sygnałów i zmiennych wewnętrznych. Znaczna część obliczeń realizowana przez jednostkę centralną może zostać pominięta bez wpływu na wynik obliczeń. Powyższe obserwacje skłaniają do fragmentarycznej realizacji programu sterowania, która pozwala na wybór tylko tych fragmentów programu sterowania które wymagają przeliczenia.

Z punktu widzenia języków programowania sterowników logicznych język SFC definiuje bloki podlegające realizacji warunkowej. Użytkownik jest zobowiązany do odpowiedniego zapisania warunków wyzwalających przeliczenie bloku. Przedstawiona w niniejszym artykule koncepcja, zdejmuje z użytkownika trud przygotowania warunków wyzwalających blok, przekazując je do kompilatora programu a następnie dedykowanych układów sprzętowych.

2.1. Realizacja zdarzeniowa programu sterowania

Wyznaczanie sterowania odbywa się na podstawie stanu trzech zbiorów zmiennych tj. wejść, wyjść oraz znaczników. Zmiana wartości w dowolnym zbiorze wymaga wyznaczenia nowej wartości sterowania w sterowniku.

$$Y_n = \lambda \left(X, Q, Y_{n-1}\right) \tag{1}$$

Gdzie:

Y_n – wynik obliczeń,
λ – funkcja przetwarzania,
X – zmienne wejściowe,
Q – zmienne wewnętrzne (markery),
Y_{n-1} – zmienne wyjściowe z chwili poprzedniej.

Podzielenie programu na bloki, jest zasadniczym elementem realizacji zdarzeniowej programu. Na podstawie analizy można określić zbiór argumentów funkcji λ dla każdego bloku. Niezbędnym do konstrukcji zdarzeniowego systemu sterowania, jest możliwość rejestracji zmian zachodzących w zmiennych sterownika i gromadzenia informacji o ich wystąpieniu dla każdego bloku programowego z osobna. Przed rozpoczęciem realizacji bloku, sprawdzany jest znacznik wykonania warunkowego. Jeżeli w zbiorze argumentów funkcji sterowania nie wystąpiły zmiany od ostatnich obliczeń, znacznik pozostaje nieustawiony a blok nie podlega przeliczeniu.

3. Pamięć procesu z detektorem zmiany zawartości

Istotnym problemem do rozwiązania pozostaje sposób wyznaczania zmian w argumentach funkcji sterowania. Wyznaczanie zmian na drodze programowej jest nieefektywne ze względu na pojawienie się dodatkowych instrukcji oraz wydłużenie programu sterowania. W skrajnym przypadku gdy wszystkie bloki programowe wymagają przeliczenia, rozwiązanie klasyczne będzie obliczone szybciej aniżeli realizacja warunkowa (wyznaczanie warunku wykonania oraz wykonanie bloku).

W sterowniku programowalnym cykl obliczeniowy rozpoczyna się od kopiowania stanu sygnałów wejściowych do pamięci obrazu procesu. Moment kopiowania zmiennych należy wykorzystać do porównania stanu zmiennych chwili t_n (bieżącego cyklu obliczeniowego) oraz t_{n-1} (poprzedniego cyklu obliczeniowego). W tym czasie realizacja programu sterowania zostaje zawieszona. Wykorzystując operację kopiowania stanu wejść do aktualizacji znaczników warunkowego wykonania bloków programowych, uzyskuje się rozwiązanie całkowicie transparentne dla dowolnego sposobu realizacji programu, które nie wymaga dodatkowych nakładów czasowo-programowych na realizację.

Jak wykazano realizacja selektywna programu sterowania pozwala wykonywać tylko niezbędne obliczenia. Łącząc właściwości jednostki centralnej sterownika ze specyficzną konstrukcją pamięci procesu, można uzyskać sterownik o bardzo dużej szybkości realizacji programu.

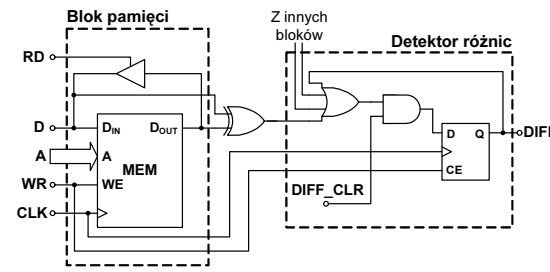
Głównym ograniczeniem przy wyznaczaniu zmian zawartości pamięci jest konieczność wykonania dodatkowego cyklu odczytu pamięci. Bieżąca zawartość słowa pamięci musi zostać porównana z słowem wpisywanym w jego miejsce.

Czy istnieje możliwość skonstruowania układu wykrywającego zmiany zawartości pamięci tak aby jego działanie było niewidoczne przy dostępie do pamięci obrazu procesu? W dalszej części rozdziału przedstawiono sposób realizacji takiej pamięci z wykorzystaniem specyficznych własności układów programowalnych FPGA.

3.1. Detektor zmiany zawartości pamięci

Pamięć w sterowniku programowalnym wykorzystywana jest do przechowania argumentów operacji. W celu przyśpieszenia procesu obliczeniowego, argumenty wejściowe są kopiowane z modułów wejściowych do pamięci obrazu procesu sterownika, przed rozpoczęciem właściwego procesu obliczeniowego. Po zakończeniu obliczeń w bieżącym obiegu pętli, wyniki obliczeń zostają przepisane z pamięci obrazu procesu do modułów wyjściowych. W pamięci obrazu procesu jest zarezerwowany obszar przeznaczony do przechowywania zmiennych wewnętrznych zwanych znacznikami.

W systemie sterowanym zdarzeniami, obliczenia są wykonywane tylko wtedy gdy została wykryta zmiana stanu w argumentach funkcji λ tj. (X, Q, Y_{n-1}) . Takie podejście wymaga obserwacji zachodzących zmian w pamięci procesu w trzech niezależnych obszarach: wejścia, wyjścia oraz znaczników. Niezależnie od obszaru przeznaczenia, zawartość pamięci może ulec modyfikacji tylko podczas operacji zapisu. W celu zbudowania detektora wykrywającego zmianę zawartości pamięci podczas zapisu, konieczne jest porównanie bieżącej i nowej zawartości komórki pamięci. Podczas gdy pamięć jest dołączona do jednostki centralnej sterownika, dodatkowe cykle dostępu do pamięci nie powinny być wykonywane. Problem można rozwiązać przez wykorzystanie pamięci o rozdzielonych wejściach i wyjściach danych, tak aby umożliwić jednoczesny odczyt i zapis danych. Schemat blokowy takiej pamięci przedstawiono na rysunku (rys. 2).



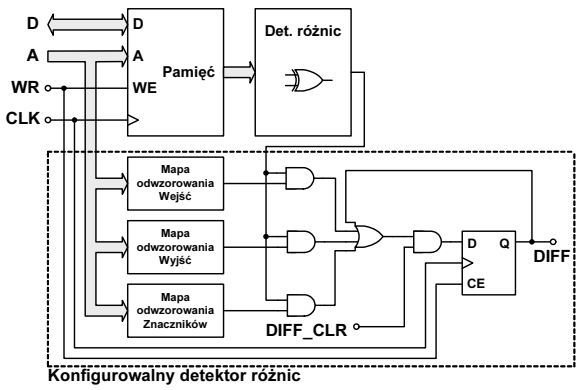
Rys. 2. Blok pamięci z detektorem zmiany zawartości
Fig. 2. Memory block with content change detector

Z pamięci o rozdzielonych wejściach i wyjściach danych, został zbudowany blok pamięci dołączony do magistrali danych procesora (D). Blok pamięci jest typowym elementem układu FPGA (BlockRAM). Niezwykle istotnym jest fakt synchronicznego wpisu do pamięci. Pozwala on uzyskać na wyjściu pamięci, bieżącą zawartość komórki przed wpisaniem do niej nowej wartości. Stan następny komórki pamięci jest porównywany ze stanem bieżącym za pomocą bramki XOR. Detektor zmiany zawartości zapamiętuje wystąpienie dowolnej zmiany w obszarze pamięci, która informuje o konieczności przeliczenia programu sterowania. W celu skasowania znacznika DIFF należy uaktywnić linie DIFF_CLR, która wyzeruje znacznik. Aby było możliwe sterowanie linią DIFF_CLR została ona dołączona do obszaru znaczników i może zostać uaktywniona przez jednostkę centralną. Na schemacie (rys. 2) przedstawiono jednobitową strukturę detektora różnic. Układ można rozszerzyć na wielobitową strukturę gdzie wyjścia detektorów różnicowych (bramek XOR) są sumowane logicznie i wprowadzane do rejestru DIFF.

3.2. Selektywny detektor różnic

Możliwość programowalnego przypisywania wybranych fragmentów pamięci do poszczególnych detektorów różnic, pozwala znacząco uprości mechanizm warunkowej realizacji. Rozszerzając koncepcje sterownika zdarzeniowego, można zastąpić programowe kwalifikowanie zadań do realizacji sprzętowym układem koordynującym przeliczanie poszczególnych bloków programowych.

Podejście takie wprowadza nowe wymagania stawiane detektorowi różnic. Detektor różnic dla każdego z bloków programowych musi mieć możliwość zdefiniowania nadzorowanego obszaru. W konsekwencji każdy z detektorów musi zostać wyposażony w mapę pamięci przechowującą informację o obszarach dozorowanych. Szczegółowa konstrukcja układu została pokazana na rysunku (rys. 3). Z blokiem pamięci procesu związany jest układ wykrywania różnicy zawartości komórek pamięci. Przesyła on informację o wykrytej różnicy do bloku rejestrów. Każdy z rejestrów posiada własną mapę obszaru dozorowanego, która odpowiada rozmieszczeniu zmiennych używanych w danym bloku programowym w przestrzeni pamięci obrazu procesu. Jeżeli informacja o zmianie odnosi się do obszaru dozorowanego, zostaje zapamiętana w rejestrze.



Rys. 3. Konfigurowalny detektor różnic
Fig. 3. Configurable difference detector

W przypadku zdarzeniowej realizacji programu sterowania obok właściwego programu sterowania podzielonego na bloki programowe, należy wygenerować dodatkową informację umożliwiająca konfigurację detektorów różnic zawartości pamięci. Każdy z bloków programowych musi posiadać mapę odwzorowania zmiennych prze niego wykorzystywanych, która pozwoli na właściwe działanie detektora różnic. W procesie implementacji należy dążyć do uzyskania możliwie największej liczby detektorów różnicowych a także zbudowania mapy odwzorowania pamięci z możliwie najmniejszych fragmentów. Niestety, spełnienie obu wymagań nie jest możliwe gdyż prowadzi do nadmiernej alokacji zasobów logicznych lub rozmiaru pamięci odwzorowania.

3.3. Efekty implementacji

Bloki detektorów różnic zostały zaimplementowane z użyciem języka opisu sprzętu. Do celów implementacji wybrano układy FPGA firmy Xilinx rodziny Spartan 3 [7]. Należą one do grupy układów symetrycznych ze zintegrowanymi pamięciami blokowymi. Każda z pamięci blokowych może pomieścić 18kb informacji w organizacji 16kb + 2kb. Bardzo istotną cecha tych pamięci jest pełna dwubramowa architektura z możliwością konfiguracji szerokości magistrali danych. W pierwszym podejściu zaimplementowano detektor różnic przy wykorzystaniu pamięci RAM z tablic LUT. Liczba grup możliwych do określenia w trzech przestrzeniach pamięci zawierała się od 16 do 128 (wiersze 1-4 w tab. 1).

Tab. 1. Zasoby logiczne konieczne do implementacji
Tab. 1. Logic requirements of change detektor

Liczba grup	LUT	LUT-RAM	BRAM
3 x 16	20	3	1
3 x 32	33	6	1
3 x 64	66	12	1
3 x 128	110	24	1
36 x 512	147	0	2

Odwzorowanie przynależności pamięci procesu do poszczególnych zadań może wykorzystywać pamięć blokową. Maksymalna szerokość magistrali danych tej pamięci wynosi 32 + 4 bity. Pozwala to na zaimplementowanie 36 detektorów różnic. Przestrzeń pamięci podzielona jest na 512 segmentów, których przynależność można zdefiniować dla każdego z detektorów z osobna. Poddano również implementacji drugie z przedstawionych rozwiązań. (wiersz 5 w tab. 1). Oba rozwiązania wymagają zbliżonej liczby zasobów logicznych ogólnego przeznaczenia. Wykorzystując pamięć blokową, uzyskano 12 krotnie więcej grup realizowanych warunkowo a także 4 krotny wzrost rozdzielczości przy definiowaniu obszarów dozorowanych.

4. Podsumowanie

Przeprowadzone rozważania nad zdarzeniową realizacją programu sterowania wykazały możliwość znacznego przyśpieszenia realizacji programu. Zaproponowane rozwiązanie pozwala na zredukowanie czasu odpowiedzi sterownika poprzez sprzętowo wspomagany wybór fragmentów programu wymagających przeliczenia. Pozwala to wyeliminować nadmiarowe wyliczanie fragmentów programu sterowania nie wypracowujących nowych rezultatów. Prace nad architekturą sterownika są dalej prowadzone. W następnym etapie przewiduje się opracowanie sprzętowego zarządzania i kolejkowania zadań do obliczeń na podstawie zmian argumentów lub interwału czasowego realizacji.

5. Literatura

[1] Aramaki N., Shimokawa Y., Kuno S., Saitoh T., Hashimoto H. 1997, „A new Architecture for High-performance Programmable Logic Controller", Proceedings of the IECON'97 23rd International Conference on Industrial Electronics, Control and Instrumentation, IEEE part vol.1, pp.187-190, New York, USA

[2] Chmiel M. E. Hrynkiewicz, A. Milik 2005, „Concurrent operation of the processors in Bit-Byte CPU of a PLC", Preprints of the IFAC World Congress, Prague, Czech Republic, July 3-8, 2005

[3] Chmiel M. E. Hrynkiewicz 2005, „Remarks on Parallel Bit-Byte CPU structures of Programmable Logic Controllers" in Design of Embedded Control Systems, Section V, 231-242, edited by: Adamski M. A., A. Karatkevich, M. Węgrzyn, , Springer Science + Business Media, Inc, 2005

[4] Donandt J. 1989, "Improving response time of Programmable Logic Controllers by use of a Boolean coprocessor", IEEE Comput. Soc. Press., Washington, DC, USA, 4:167-169.

[5] Michel G. 1990, Programmable Logic Controllers, Architecture and Applications, John Wiley & Sons, West Sussex, England.

[6] Modicon 1990, Modicon 984 Programmable Controller – System Manual, AEG Modicon.

[7] Xilinx 2006, Spartan 3 FPGA Familly Complete Data Sheet (DS099), 2006.

Artykuł recenzowany