Adam ZIĘBIŃSKI, Rafał CUPEK, Wojciech SROKA
POLITECHNIKA ŚLĄSKA, WYDZIAŁ AEI, INSTYTUT INFORMATYKI
ul. Akademicka 16, 44-100 Gliwice

Aplikacja w języku Java realizująca funkcję parsera pseudokodu opisującego strukturę specjalizowanego koprocesora sterownika PLC do VHDL

Dr inż. Adam ZIĘBIŃSKI

Ukończył studia w 1996 r. w Instytucie Informatyki na Wydziale Automatyki Elektroniki i Informatyki Politechniki Śląskiej. Pracę doktorską obronił w 2002 r. Obecnie pracuje na stanowisku adiunkta w Instytucie Informatyki na wydziale AEI PolŚl. Jego zainteresowania to sprzętowo-programowe projektowanie koprocesorów problemowo zorientowanych z wykorzystaniem układów reprogramowalnych VLSI.

e-mail: adam.ziebinski@polsl.pl

Dr inż. Rafał CUPEK

Ukończył studia w 1992 r. w Instytucie Informatyki na Wydziale Automatyki Elektroniki i Informatyki Politechniki Śląskiej. Pracę doktorską obronił w 1999r. Obecnie pracuje na stanowisku prodziekana w Instytucie Informatyki na Wydziale Automatyki Elektroniki i Informatyki Politechniki Śląskiej. Jego zainteresowania to komunikacja w przemysłowych systemach komputerowych oraz model komponentowy w systemach informatyki przemysłowej.

e-mail: rafal.cupek@polsl.pl

Mgr inż. Wojciech SROKA

Ukończył studia w 2009r. w Instytucie Informatyki na Wydziale Automatyki Elektroniki i Informatyki Politechniki Śląskiej. Obecnie pracuje na stanowisku konsultanta ds. systemów informatycznych w firmie Vattenfall. Jego zainteresowania to wykorzystanie języka JAVA w procesie projektowania układów w VHDL.

e-mail: sroka.wojciech@gmail.com

Streszczenie

Artykuł przedstawia koncepcję projektowania w VHDL systemu pełniącego funkcję specjalizowanego koprocesora sterownika PLC, realizującego tylko wyszczególniony zestaw zadań. W pracy pokrótce omówiono poszczególne moduły realizujące funkcję koprocesora sterownika PLC. Ponadto przedstawiono funkcjonalność parsera pseudokodu w języku Java, opisującego strukturę specjalizowanego sterownika PLC do VHDL. Na końcu zaprezentowano wyniki porównania działania przykładowej aplikacji w sterowniku PLC GE Fanuc CPUE05 i układzie FPGA XC3S500e.

Słowa kluczowe: FPGA, Java, PLC, systemy wbudowane, VHDL.

Application in Java language realizing the function parser of pseudocode describing structure of a specialized coprocessor of PLC in VHDL

Abstract

The paper presents a project of embedded system realization on a FPGA array, fulfilling the function of a specialized coprocessor PLC. There are described individual modules realizing the function of the coprocessor of PLC in VHDL: the memory map of the controller (Fig.1) including the controller registers and the controller of the memory (Fig.2) for read/write the data in the registers. Moreover, functionality of the parser of pseudocode in the Java language, describing the structure of specialized PLC to VHDL, is presented. The components in VHDL [4] used by the parser are described in the pseudocode and presented in Table 1. The instructions in the pseudocode are equivalent to those in the GE-Fanuc Versa Max controller family. The comparison results of working of an exemplary application in PLC GE Fanuc Versa Max CPUE05 [3] and FPGA XC3S500e are given. The exemplary application for the controller is shown in Fig. 3. As a result of parsering by the PLC2VHDL program there was received the code in VHDL realizing the described task (Fig.4). The code VHDL was subjected to testing, synthesis and implementation with utilization of tools ISE ™ Foundation ™. As a result of implementation, there was obtained the configurational file for the FPGA. The project takes about 1 % resources in the XC3S500e and can work with the maximum 79MHz. The controller work cycle (Fig.5) in FPGA takes 3 tacts and lasts 37.863ns.

Keywords: embedded systems, FPGA, Java, PLC, VHDL.

1. Wstęp

Narastające zapotrzebowanie systemów przemysłowych na wyższe szybkości przesyłu i przetwarzania danych, skłania ku wykorzystywaniu zaawansowanych technologicznie rozwiązań [1, 2] specjalizowanych układów VLSI (Very-large-scale integration) czy też matryc programowalnych FPGA (Field Programmable Gate Arrays). Biorąc pod uwagę, że układy FPGA doskonale nadają się do realizacji systemów prototypowych, realizowane były prace nad koncepcją projektowania w VHDL systemu pełniącego funkcję specjalizowanego koprocesora sterownika PLC [3, 4, 5], wykonującego tylko wyszczególniony zestaw zadań. W efekcie prowadzonych prac powstała biblioteka elementów realizujących podstawowe i złoże funkcje sterownika PLC [4, 5]. W celu usprawnienia wykonywania testów i projektowania aplikacji dla koprocesora sterownika PLC, rozpoczęto prace nad narzędziem w języku Java, wspomagającym generowanie struktury sterownika w VHDL. Aplikacja ta pozwala na napisanie w pseudokodzie projektu sterowania, na bazie elementów zawartych w bibliotece. Powstały w ten sposób projekt następnie podlega parsowaniu, w wyniku którego generowany jest kod sterownika PLC w VHDL. Każda zmiana w projekcie sterownika będzie wymagała ponownej syntezy i implementacji z wykorzystaniem narzędzi CAD dla wybranego układu reprogramowalnego. W efekcie każde zadanie sterownia będzie zajmowało różną liczbę zasobów i połączeń w układzie FPGA.

Główna różnica pomiędzy tradycyjnym sterownikiem PLC, a proponowanym rozwiązaniem polega na tym, że sterownik PLC realizuje dowolny zestaw funkcji sterowania zależny od napisanego programu bez zmian struktury wewnętrznej sterownika, natomiast sterownik opisany w VHDL realizuje tylko jeden zestaw funkcji dla którego generowany jest odpowiadający mu zestaw połączeń w układzie FPGA. Poprzez wykorzystywanie jedynie zasobów niezbędnych do realizacji danego zestawu zadań, możliwe będzie zaoszczędzenie zasobów sprzętowych. Ponadto PLC wykonuje instrukcje sekwencyjnie, natomiast układ programowalny pozwala zrównoleglić wykonywanie niezależnych operacji, co dodatkowo wpłynie na zwiększenie szybkości pracy systemu.

2. Specjalizowany sterownik PLC w VHDL

Głównym celem prowadzonych przez nas badań jest wykonanie koprocesora realizującego wybrane funkcje sterownika przemysłowego PLC w układzie programowalnym FPGA. Koprocesor ten będzie działał na podobnych zasadach jak współczesne sterowniki PLC. Na obecnym etapie prac zaimplementowano podstawową funkcjonalność sterownika przemysłowego obejmującą pierwsze cztery kroki cyklu rozkazowego, w tym:

- inicjalizację cyklu,
- obsługę wejść,

- wykonanie programu sterującego,
- obsługę wyjść.

W pierwszym etapie następuje ustawienie odpowiednich rejestrów wartościami domyślnymi oraz odczyt wartości z wejść sterownika. Krok ten zrealizowano jako proces w języku VHDL, który jest uruchamiany jednorazowo tuż po uruchomieniu sterownika, następnie sterownik przechodzi do normalnego cyklu pracy.

W kroku drugim sterownik cyklicznie pobiera wartości z wejścia układu. Analogicznie zaimplementowano krok czwarty, podczas którego uaktualniane są wyjścia sterownika. Oba procesy dokonują operacji na wejściach i wyjściach po upływie określonej liczby cykli zegara. Liczba ta jest zależna od długości wykonywania się aplikacji sterującej.

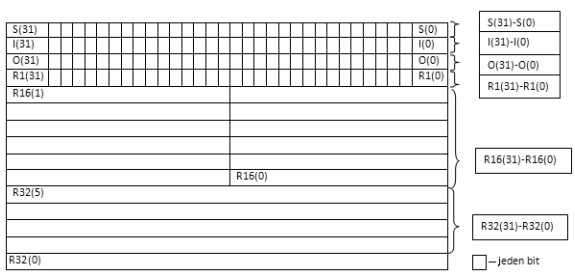
Krok trzeci realizujący funkcję procesu obsługującego program sterujący, zawiera wymaganą aplikację wykonującą funkcję sterowania. Aplikacja ta jest budowana z wykorzystaniem wcześniej przygotowanych elementów bibliotecznych realizujących podstawowe funkcje sterownika PLC. Przy czym w odróżnieniu od standartowego sterownika PLC, umożliwia ona wykonywanie wielu procesów równolegle, na co pozwalają właściwości układów reprogramowalnych.

By zapewnić powyższą funkcjonalność rozpoczęte zostały prace nad implementacją następujących elementów w VHDL:

- mapa pamięci zawierająca rejestry sterownika,
- kontroler pamięci sterownika, odczytujący i zapisujący dane w rejestrach.

Projekt powyższych elementów był realizowany z wykorzystaniem narzędzi ISE™ Foundation™ with ISE™ Simulator [6] firmy Xilinx i ModelSim [7] firmy Mentor Graphics.

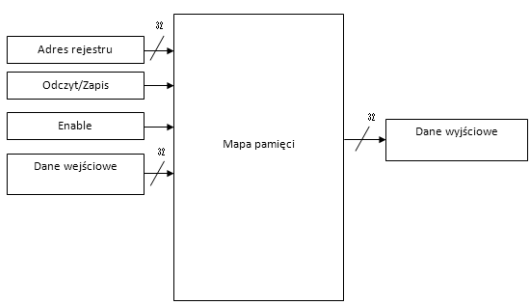
Mapą pamięci sterownika jest zespół rejestrów różnej długości, przechowujących dane, z których korzysta aplikacja sterownika. Mapa pamięci składa sie z piętnastu rejestrów 32 bitowych (rys. 1).



Rys. 1. Mapa pamięci sterownika
Fig. 1. The memory map of the controller

Użytkownik ma do dyspozycji 32 rejestry systemowe jednobitowe, 32 rejestry wejściowe jednobitowe, tyle samo rejestrów wyjściowych, 32 rejestry jednobitowe ogólnego przeznaczenia, dwanaście rejestrów szesnastobitowych i osiem 32 bitowych. Dostęp do żądanego rejestru będzie możliwy po podaniu numeru rejestru.

Kontroler pamięci (rys.2) pełni funkcję modułu udostępniającego na zewnątrz dane zawarte w mapie pamięci. Dostęp do danego rejestru będzie odbywał się poprzez podanie adresu rejestru, podanie sygnału odczytu/zapisu oraz odczytanie wyniku z wyjścia lub podanie na wejście wartość do zapisu.



Rys. 2. Kontroler pamięci
Fig. 2. The memory controller

3. Parser pseudokodu opisującego strukturę specjalizowanego sterownika PLC do VHDL

Na potrzeby przeprowadzenia badań zaprojektowano aplikację w języku Java PLC2VHDL tłumaczącą pseudokod opisujący projekt sterowania na język VHDL. Umożliwia ona znacznie szybszą realizację konkretnego zadania poprzez wykorzystanie wcześniej przygotowanych komponentów. Komponenty w VHDL [4], które są wykorzystywane przez parser, zostały opisane w postaci rozkazów pseudokodu i przedstawione w tabeli 1. Rozkazy te są odpowiednikami rozkazów stosowanych w rodzinie sterowników GE-Fanuc Versa Max.

Tab. 1. Skrócona lista podstawowych rozkazów pseudokodu
Tab. 1. The shortened list of the basic instructions of the pseudocode

Typ funkcji	Rozkaz
Liczniki i przekaźniki	DNCTR enable ,reset ,PV, Q; UPCTR enable ,reset ,PV, Q; OFDT CLOCK ,enable, Q ,PV ,CV; ONDTR CLOCK ,reset ,PV,Q,enable, CV; TMR CLOCK,enable ,Q ,PV ,CV;
Funkcje matematyczne	ADD_INT enable ,IN1 , IN2,ok ,Q; SUB_INT enable , IN1, IN2,ok ,Q; DIV_INT enable , IN1, IN2,ok ,Q; MUL_INT enable , IN1, IN2,ok ,Q;
Relacje matematyczne	LT_INT enable , IN1, IN2,Q; GE_INT enable , IN1, IN2,Q; EQ_INT enable , IN1, IN2,Q; NE_INT enable , IN1, IN2,Q; GT_INT enable , IN1, IN2,Q LE_INT enable , IN1, IN2,Q;

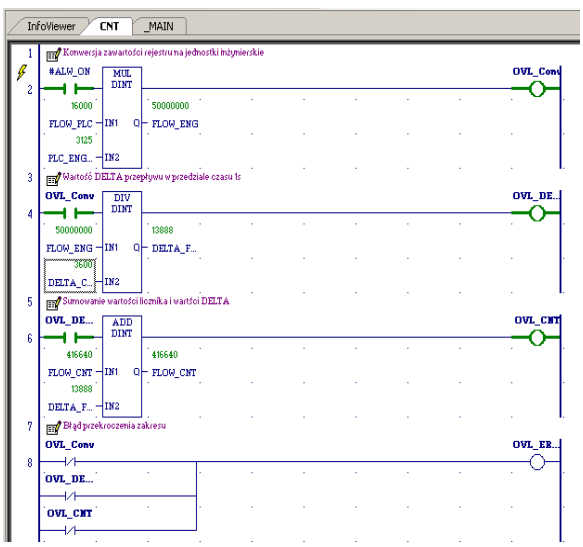
Zaprojektowanie sterownika na podstawie pseudokodu nie będzie wymagało od projektanta znajomości VHDL. Wystarczy zaznajomienie się z gramatyką rozkazów pseudokodu.

Aplikacja wykorzystuje interfejs graficzny z wykorzystaniem biblioteki Swing. Użytkownik ma do dyspozycji okno z dwoma obszarami tekstowymi: w jednym można wprowadzać pseudokod, a w drugim będzie wyświetlany kod przetłumaczony na język VHDL. Podczas parsowania aplikacja weryfikuje poprawność wprowadzonego pseudokodu. Sprawdzana jest poprawność wprowadzonych rozkazów, liczba argumentów rozkazu oraz czy zmienne symbolizujące rejestry mają odpowiednią długość. W trakcie parsowania, wszystkie wartości stałe są automatycznie konwertowane na wewnętrzną reprezentację przechowywaną w rejestrach sterownika. Parser pobiera z wejścia kod sterownika i najpierw wydziela z niego wszystkie użyte rozkazy. Następnie wydzielone zostają nazwy rejestrów którym nadawana jest odpowiednia szerokość (1 bit, 16 bitów, 32 bity) w oparciu o specyfikację rozkazu. Następnie obliczany jest rozmiar mapy pamięci i w końcowym etapie zostają wygenerowane połączenia pomiędzy poszczególnymi komponentami. Pliki projektu VHDL wygenerowane za pomocą parsera mają ściśle określoną budowę i nie mogą zawierać błędów. Jest to konieczne aby wygenerowany kod mógł być przetwarzany przez narzędzia syntezy HDL. Następnie powinny zostać przyporządkowane wejścia i wyjścia dyskretne odpowiednim wyprowadzeniom układu programowalnego. Projekt może być rozbudowany o dodatkowe komponenty w dowolnym edytorze VHDL. Po wykonaniu syntezy i implementacji projektu wykorzystującego wygenerowane przez parser pliki VHDL, otrzymamy strukturę sterownika realizującą zaprojektowane zadanie opisane w pseudokodzie.

4. Wyniki testów

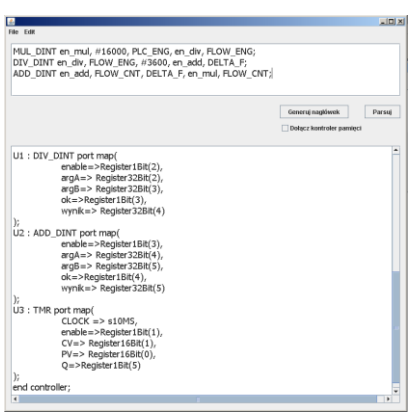
Dla potrzeb testów wykonano aplikację dla sterownika GE Fanuc Versa Max z jednostką centralną CPUE05 [3]. Aplikacja zawiera procedurę CNT realizującą funkcjonalność licznika, na które składają się następujące operacje: przeskalowanie zakresu

pomiaru przepływu z 0-32000 jednostki pomiarowe PLC odpowiadające zakresowi pomiaru prądu 4-20 mA na jednostki fizyczne pomiaru przepływu z zakresu 0 – 100 m3/h, obliczenie przepływu za okres 1 cyklu pomiarowego (testowano cykl 1s.), zliczanie sumarycznej wartości przepływu. Operacje arytmetyczne wykonywane są na 32 bitowych zmiennych typu DINT (integer) przy zliczaniu z krokiem 0,000001m3. Fragment kodu sterownika PLC odpowiadający za realizację powyższych funkcji oraz kontrolę przepełnienia przedstawiono na rys. 3 Procedura CNT wywoływana była z cyklem 1 s uzyskiwanym na podstawie zmiennych systemowych sterownika. Przeprowadzono eksperyment polegający na zliczaniu przepływu dla symulowanej stałej wartości pomiaru odpowiadającej połowie zakresu (wartość pomiaru w PLC równa 16000). Zmierzona w czasie testu wartość 12,499200 odpowiadała wartości planowanej z niedokładnością wynikającą z realizacji konwersji zakresów przy zastosowaniu operacji na liczbach całkowitych. Obserwowany czas cyklu sterownika PLC wahał się pomiędzy 2,7 ms a 3,1 ms, przy czym czas trwania samych operacji arytmetycznych wynosi 151 µs (na podstawie danych katalogowych: MUL(DINT) – 50 µs, DIV(DINT) – 51 µs, ADD(DINT) - 50 µs).



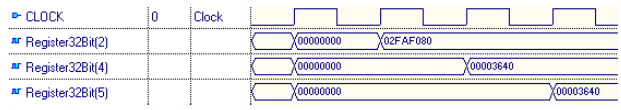
Rys. 3. Przykładowa aplikacja dla sterownika GE Fanuc Versa Max CPUE05
Fig. 3. Exemplary application for GE Fanuc Versa Max CPUE05 controller

Program dla sterownika PLC napisano również w aplikacji PLC2VHDL napisanej w języku Java (rys. 4).



Rys. 4. Przykładowa aplikacja dla sterownika PLC w edytorze PLC2VHDL
Fig. 4. Exemplary application for PLC in PLC2VHDL editor

W wyniku parsowania otrzymaliśmy kod w VHDL realizujący wcześniej opisane zadanie. Kod VHDL został poddany testowaniu syntezie i implementacji z wykorzystaniem narzędzi ISE™ Foundation™ z ISE™ Simulator. W wyniku implementacji otrzymaliśmy zbiór wynikowy dla układu FPGA. Powstały projekt zajmuje około 1 % zasobów typu Slices (w tym 1% typu Slice Flip Flops oraz 1% typu 4 input LUTs) i może pracować z maksymalną częstotliwością 79.236MHz, przy czym minimalny pojedynczy takt trwa 12.621ns. Dane te zostały wyznaczone dla układu XC3S500e [8]. Cykl pracy (Rys.5) sterownika w FPGA zajmuje trzy takty i trwa 37.863ns, (inicjalizacja początkowa rejestrów zajmuje jeden dodatkowy takt).



Rys. 5. Cykl pracy sterownika PLC w FPGA
Fig. 5. The cycle of the work of the controller in FPGA

5. Podsumowanie

Przeprowadzone testy wykazują, że czas wykonywania operacji arytmetycznych w sterowniku PLC GE-Fanuc jest zdecydowanie dłuższy (około 3973 razy) niż w koprocesorze sterownika w FPGA. O ile w większości przypadków pomiaru przepływu medium różnica ta nie jest aż tak istotna, ze względu na opóźnienie toru pomiarowego. To jednak w przypadku sygnałów szybkozmiennych i przetworników analogowo cyfrowych gwarantujących czas przetwarzania poniżej 1 ms, różnica cyklu pracy sterownika może być istotna. Dodatkową zaletą rozwiązania zaimplementowanego w VHDL jest stałość cyklu pomiarowego co w przypadku operacji całkowania wartości chwilowych może mieć znaczący wpływ na dokładność pomiaru. Dużą zaletą naszej aplikacji jest umożliwienie szybkiego projektowania struktury koprocesora PLC na bazie wcześniej przygotowanej biblioteki elementów podstawowych. Wykonany przez nas parser w języku Java pozwala na łatwe wprowadzanie programu i wygenerowanie kodu koprocesora sterownika w VHDL. W efekcie projektant nie musi znać VHDL by wygenerować strukturę sterownika. Niestety by wygenerować plik konfiguracyjny dla danego układu FPGA użytkownik musi skorzystać z narzędzi CAD do tego celu przeznaczonych. Już na obecnym etapie możliwe jest przetwarzanie równoległe, na co pozwalają właściwości układów FPGA. Wykonywanie w ten sposób niezależnych operacji realizowane jest praktycznie automatycznie. Natomiast poprawne wykonywanie operacji wzajemnie zależnych realizowanych po części równolegle, wymaga przeprowadzenia dalszych badań.

6. Literatura

[1] Daoshan Du, Yadong Liu, Xingui Guo Kazuo Yamazaki, Mako-to Fujishima: Study on LD-VHDL conversion for FPGA-based PLC implementation. Int J Adv Manuf Technol (2009) 40:1181–1190 Springer-Verlag London Limited 2008.
[2] Daoshan Du, Xiaodong Xu, Kazuo Yamazaki: A study on the generation of silicon-based hardware Plc by means of the direct conversion of the ladder diagram to circuit design language. nt J Adv Manuf Technol (2010) 49:615–626 Springerlink.com
[3] Sterowniki programowalne Seria 90-30/VersaMax/Micro GEFanuc Automation, Kraków, wrzesień 1999.
[4] Ziębiński A., Znamirowski L., Sroka W.: Implementacja wybranych funkcji sterownika przemysłowego w układzie programowalnym, Systemy czasu rzeczywistego, Metody i zastosowania, WKŁ, Warszawa 2007, pp. 209-220.
[5] Ziębiński A., Sroka W.: Realizacja funkcji statystycznych w sterowniku przemysłowym z wykorzystaniem układu FPGA. praca zbiorowa pod redakcją Mazura Z. i Huzara Z.: Modele i zastosowania systemów czasu rzeczywistego, WKiŁ, Wwa 2008 s. 115 -126.
[6] ISE™ Foundation™ with ISE™ Simulator , http://www.xilinx.com
[7] ModelSIM, http://model.com/
[8] http://www.xilinx.com/support/documentation/spartan-3e.html

otrzymano / received: 13.05.2011
przyjęto do druku / accepted: 04.07.2011 *artykuł recenzowany*