Mirosław ŁAZORYSZCZAK
ZACHODNIOPOMORSKI UNIWERSYTET TECHNOLOGICZNY, WYDZIAŁ INFORMATYKI,
ul. Żołnierska 52, 71-210 Szczecin

Mikroprocesor PicoBlaze na platformie CPLD w dydaktyce systemów wbudowanych

Dr inż. Mirosław ŁAZORYSZCZAK

Ukończył studia na Wydziale Elektrycznym Politechniki Szczecińskiej, obronił pracę doktorską w 2005 r. Pracuje jako adiunkt w Katedrze Architektury Komputerów i Telekomunikacji na Wydziale Informatyki Zachodniopomorskiego Uniwersytetu Technologicznego w Szczecinie Jest członkiem IEEE. Zainteresowania naukowe dotyczą układów rekonfigurowalnych, systemów wbudowanych oraz przetwarzania dźwięku.

e-mail: mlazoryszczak@wi.zut.edu.pl

Streszczenie

W artykule przedstawiono wybrane aspekty implementacji mikroprocesora PicoBlaze na platformie uruchomieniowej CoolRunner-II CPLD Starter Kit. Szczególną uwagę poświęcono obsłudze portów wejścia/wyjścia, a także wykorzystaniu elementów wbudowanych w platformę, uwzględniając także zewnętrzne moduły rozszerzające. Ograniczenia zasobów układu CPLD wymagają praktycznego zastosowania dekompozycji funkcjonalnej systemu. Jako przykłady aplikacji przedstawiono sterowanie diodami oraz wbudowanym wyświetlaczem siedmiosegmentowym.

Słowa kluczowe: procesor programowy, CPLD.

PicoBlaze microprocessor CPLD implementation for teaching embedded systems

Abstract

In this paper selected aspects of soft processor implementation in CPLD platform are presented. The processor considered here is PicoBlaze. The code of this model is available from Xilinx after registration. The hardware platform is CoolRunner-II CPLD Starter Kit. It is possible to extend simply the base configuration of the board with number of additional modules called Pmods (Fig. 1). The paper presents the main features of PicoBlaze from the teaching of embedded systems point of view. A few paragraphs show the organization of I/O ports and possibilities of their modifications (Fig. 2). Next the main flow of project files is shown (Fig. 3) including compilation and implementation processes. There are three applications used for compare purposes. The first one is the empty loop, the second one is "moving" LED and the third one is seven segment display control. The sample way of modifying selected project files in order to change available I/O ports is presented. Fig. 5 shows the RTL level schematic of the system running LED display control application with particular emphasis on I/O handling. The limitations of implementations as well as advantages of the proposed approach are shown. The main advantage for teaching embedded systems is necessity of common hardware and software design in case of adapting to the platform constraints.

Keywords: soft processor, CPLD.

1. Wstęp

Układy rekonfigurowalne stanowią stały element wielu aplikacji w różnych dziedzinach techniki. Także domena systemów wbudowanych staje się coraz większym obszarem stosowania technologii CPLD/FPGA. Istnieje wiele platform uruchomieniowych, które z powodzeniem mogą być wykorzystywane w zadaniach szybkiego prototypowania systemów sprzętowo-programowych jak również w dydaktyce. Jednym z przykładów może być zestaw Altium Nanoboard NB2. Zawiera on niezwykle rozbudowane układy peryferyjne, zaś jego sercem może być jeden z wybranych układów FPGA różnych producentów np. Cyclone II lub III (Altera), Spartan 3 lub Virtex 4 (Xilinx) oraz Lattice.

Nieodzownym elementem systemu wbudowanego jest mikroprocesor, który w przypadku układów rekonfigurowalnych realizowany jest w postaci procesora programowego (ang. *soft processor*). Istnieje szereg modeli procesorów programowych dostępnych komercyjnie jak i na zasadach *open source* o różnej architekturze oraz możliwościach. Są to modele autorskie, ale także realizujące architektury istniejących mikroprocesorów bądź mikrokontrolerów obecnych na rynku w formie sprzętowej.

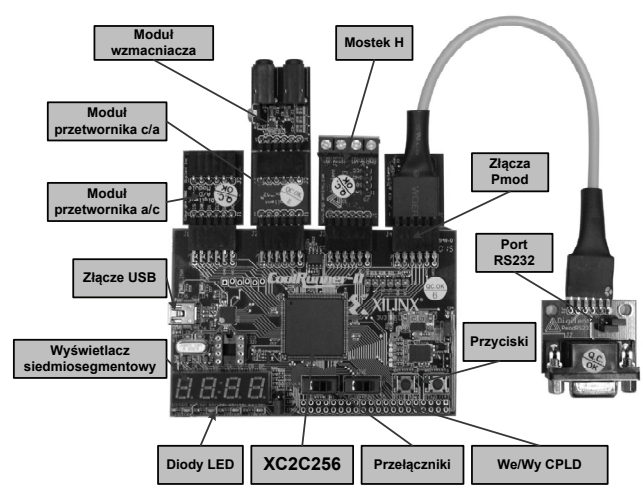
W niniejszym artykule przedstawione zostaną wybrane szczegóły implementacji procesora programowego PicoBlaze na platformie CPLD w nauczaniu systemów wbudowanych.

2. Płyta uruchomieniowa

Do implementacji procesora programowego PicoBlaze posłużono się komercyjną platformą uruchomieniową CoolRunner-II CPLD Starter Kit. W skład płyty wchodzą następujące komponenty:

- układ XC2C256,
- kontroler USB 90USB162,
- przetwornik analogowo-cyfrowy LTC2494,
- dwa przyciski,
- dwa przełączniki,
- cztery diody LED,
- czterosekcyjny wyświetlacz siedmiosegmentowy,
- zwora do wyboru częstotliwości taktowania,
- złącze zewnętrznego zasilania, np. bateryjnego,
- złącze zewnętrznej pamięci SPI PROM,
- cztery złącza modułów rozszerzeń (tzw. Pmod).

Podstawowym elementem płyty jest układ CoolRunner-II XC2C256. Pod względem liczby makrokomórek logicznych, która wynosi 256, układ ten znajduje się pośrodku oferty asortymentowej wśród układów CoolRunner-II. Z rodzaju zastosowanej obudowy wynika maksymalna liczba 118 wejść/wyjść dostępnych dla użytkownika. W omawianym zestawie do dyspozycji pozostaje 69 wejść/wyjść, z których 32 doprowadzone są do złącz rozszerzeń Pmod, natomiast 37 połączonych jest z punktami lutowniczymi. Na rys. 1 przedstawiono płytę uruchomieniową wraz z przykładowym zestawem dostępnych modułów rozszerzeń Pmod.



Rys. 1. Płyta uruchomieniowa z układem CPLD wraz z przykładowym zestawem modułów rozszerzających
Fig. 1. Evaluation CPLD board with extension modules

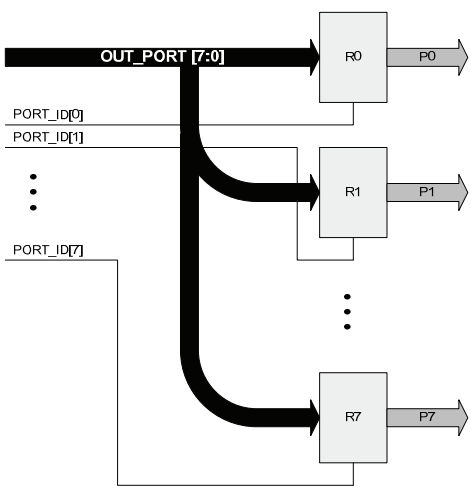
Umieszczenie procesora programowego w strukturze CPLD jest bardziej kłopotliwe od takiej samej operacji w strukturze FPGA,

przede wszystkim z powodu dostępnych zasobów logicznych, ale też sprzyja rozwiązywaniu rzeczywistych problemów projektowych w nieskomplikowanym środowisku.

3. Procesor PicoBlaze

PicoBlaze to stosunkowo prosty mikrokontroler 8-bitowy, opisany w językach opisu sprzętu. Do jego zalet należy zakres funkcjonalny oraz dopasowanie do struktur rekonfigurowalnych. PicoBlaze występuje w dokumentacji także pod nazwą KCPSM (ang. *Constant (K) Coded Programmable State Machine*). Mikrokontroler ten istnieje w różnych wersjach, przeznaczonych na różne platformy sprzętowe m.in. CPLD, Spartan, Virtex itp. Architektura PicoBlaze dostosowana do technologii CPLD jest nieco ograniczona w stosunku do np. Spartan-3 czy też innych układów FPGA. Związane jest to przede wszystkim z różnicami w sposobie realizacji pamięci w technologii CPLD w stosunku do układów FPGA, ale także z liczby dostępnych zasobów logicznych w obu technologiach.

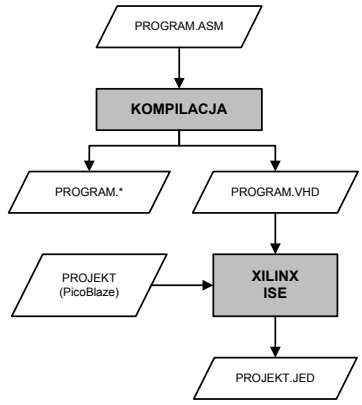
Standardowo w modelu PicoBlaze występuje 16 rejestrów ośmiobitowych o nazwach s0 ÷ sF. W szczególnych przypadkach liczba ta może zostać zmieniona poprzez modyfikację kodu źródłowego VHDL. W wersji CPLD występuje osiem rejestrów (s0 ÷ s7). Jednostka arytmetyczno-logiczna wykonuje podstawowe operacje, przy czym w PicoBlaze brak zaimplementowanej np instrukcji mnożenia. Odpowiednik rejestru statusu lub słowa stanu znanego z innych procesorów zredukowany jest do dwóch bitów: znacznika przeniesienia (*CARRY*) oraz znacznika zera (*ZERO*). W przypadku implementacji PicoBlaze w układach CPLD pamięć programu jest zredukowana do 256 słów o rozmiarze 16 bitów.



Rys. 2. Organizacja portów we/wy w mikroprocesora PicoBlaze
Fig. 2. Organization of PicoBlaze I/O ports

PicoBlaze oferuje możliwość obsługi do 256 portów zewnętrznych. Jednak podstawowy model procesora posiada tylko jedno wyjście OUT_PORT o rozmiarze ośmiu bitów. Istnieje ponadto wyjście PORT_ID, również o rozmiarze ośmiu bitów. Zatem, aby wykorzystać wszystkie możliwe porty konieczne jest dodanie układu logiki, za pomocą której bieżąca wartość portu wyjściowego (na liniach OUT_PORT) zostanie przekazana na zewnątrz układu. Wymaga to w praktyce zastosowania dodatkowych rejestrów zewnętrznych uaktywnianych wybranymi liniami PORT_ID. Na rys. 2 przedstawiono interfejs We/Wy w postaci maksymalnej, z ośmioma rejestrami wyjściowymi. Dzięki takiemu rozwiązaniu możliwy jest wybór jednego bądź kilku jednocześnie fizycznych rejestrów zewnętrznych.

W ramach projektu mikrokontrolera PicoBlaze dostępne są pliki źródłowe w języku VHDL, co umożliwia szczegółowe zapoznanie się z jego budową i zasadą działania, ale także stwarza wyjątkową okazję do eksperymentowania z modyfikacjami architektury i dostosowywaniem zasobów wewnętrznych do wymagań projektanta systemu. Ponieważ udostępniony jest także kod źródłowy asemblera dla PicoBlaze, możliwe są także zmiany zbioru instrukcji zarówno w modelu mikrokontrolera, jak i ich obsługi na poziomie asemblera.



Rys. 3. Przebieg procesu kompilacji programu i implementacji projektu
Fig. 3. The process of program compilation and project implementation

Na rys. 3 przedstawiono zależności pomiędzy plikami w ramach projektu. W wyniku kompilacji powstaje plik zawierający gotowy do implementacji moduł pamięci z wynikowym kodem programu w postaci jednostki projektowej oraz architektury opisanej w języku VHDL.

4. Przykładowe aplikacje

Na potrzeby niniejszego artykułu zostały przygotowane trzy przykłady typowych aplikacji, z którymi zwykle rozpoczyna się etap nauki obsługi zestawów uruchomieniowych:

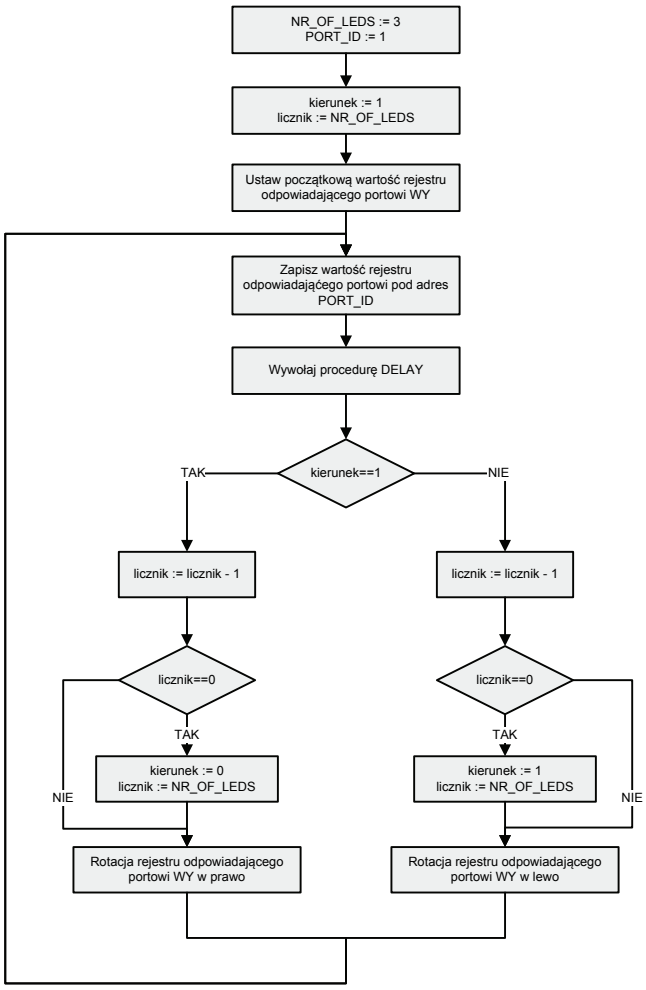
- pusta pętla główna (*empty*),
- „wędrująca" dioda (*led*),
- wyświetlacz siedmiosegmentowy (*7seg*).

W systemie z programem *empty* z pustą pętlą główną zaimplementowano pojedynczy rejestr wyjściowy oraz wykorzystano jednorazowo instrukcję zapisu do portu wyjściowego. W przypadku programu zawierającego tylko i wyłącznie pustą pętlę główną narzędzie do implementacji pomijało praktycznie całą implementację mikroprocesora przy alokacji jedynie pojedynczych makrokomórek logicznych.

Drugi przykład, polegający na implementacji „wędrującej" diody, którego algorytm przedstawiono na rys. 4, został dostosowany do listy rozkazów mikroprocesora PicoBlaze. Podobnie jak w poprzednim przykładzie, do obsługi czterech diod LED znajdujących się na płycie uruchomieniowej, wykorzystano jeden, ośmiobitowy port wyjściowy. W celu odmierzania okresu włączenia i wyłączenia diody wykorzystano procedurę DELAY, składającą się z dwóch zagnieżdżonych, pustych pętli programowych.

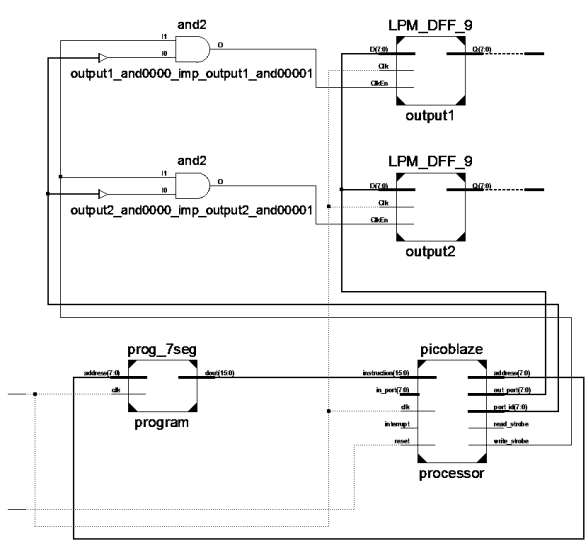
Sterowanie wyświetlaczem siedmiosegmentowym wymaga co najmniej siedmiu wyjść do włączania/wyłączania pojedynczych segmentów oraz czterech wyjść do sterowania wspólną anodą poszczególnych wyświetlaczy. Obsługa wyświetlacza odbywa się w sposób sekwencyjny. Niezbędne jest zatem dołączenie dodatkowego rejestru wyjściowego, umożliwiającego sterowanie szesnastoma wejściami/wyjściami. Przykładowy kod VHDL realizujący to wymaganie został przedstawiony poniżej.

```
IO_registers: process(clk)
begin
  if clk'event and clk='1' then
    if port_id(0)='1' and write_strobe='1' then
      output1 <= out_port;
    end if;
    if port_id(1)='1' and write_strobe='1' then
      output2 <= out_port;
    end if;
 end if;
end process IO_registers;
```



Rys. 4. Algorytm programu sterującego „wędrującą" diodą dostosowany do listy rozkazów PicoBlaze
Fig. 4. Algorithm for „moving" diode adapted to the instruction list of PicoBlaze

Na rys. 5 przedstawiono schemat z generatora schematów RTL odpowiadający implementacji przykładu obsługi wyświetlacza siedmiosegmentowego. Na schemacie, poza blokiem procesora oraz pamięci programu, widoczne są rejestry wyjściowe *output1* oraz *output2*, których wyjścia połączone są w wyświetlaczami siedmiosegmentowymi.



Rys. 5. Schemat RTL układu z programem *7seg*
Fig. 5. RTL schematic of system with *7seg* program

W tabeli 1 przedstawiono porównanie wielkości programu, wykorzystanie zasobów po implementacji na przykładzie makrokomórek oraz wejść bloków funkcyjnych dla trzech programów omawianych w artykule. Jak wspomniano wyżej, program empty zawierał pojedynczy zapis do portu wyjściowego poza główną pętlą programu. Opuszczenie tego zapisu powodowało, iż narzędzie syntezy praktycznie pomijało implementację procesora oraz pamięci programu, pozostawiając wykorzystanie zasobów na poziomie bliskim 0%.

Tab. 1. Rozmiar programu i wykorzystanie zasobów logicznych
Tab. 1. Program size and logic resource summary

Program	Wielkość programu	Makrokomórki wykorzystane/maksimum	Wejścia bloków funkcyjnych wykorzystane/maksimum
empty	6 bajtów	71/256 (28%)	168/640 (27%)
led	50 bajtów	210/256 (83%)	408/640 (64%)
7seg	70 bajtów	223/256 (88%)	441/640 (69%)

Należy także zaznaczyć, iż implementacja przykładów *led*, a zwłaszcza *7seg* wymuszała ograniczenia dostępnej wielkości kodu programu. Domyślnym rozmiarem programu wynikowego dla kompilatora asemblera PicoBlaze jest wartość 256 słów szesnastobitowych. Rozmiar tablicy programu implementowanej w postaci pamięci stałej został każdorazowo zmniejszony do 64 słów. Pomimo przyjętego rozmiaru pamięci zaobserwowano trudności w implementacji niektórych wariantów kodu. Ograniczało to np. możliwość wykorzystania pełnej puli dostępnych rejestrów procesora.

5. Podsumowanie

Implementacja procesora PicoBlaze na platformie CPLD charakteryzuje się szeregiem ograniczeń, co raczej nie przemawia za szerszym wykorzystaniem tego rozwiązania w projektach większych, aniżeli przedstawione w artykule. Jednak w zastosowaniach dydaktycznych, zwłaszcza na początkowym etapie zapoznawania się z procesorami programowymi, platforma CPLD może być wprowadzana z powodzeniem. W przedstawionych przykładach posłużono się niemal „standardową" wersją Pico Blaze'a, ale możliwości systemu zwiększają się dzięki dopasowaniu sprzętowych funkcji procesora programowego do obsługiwanych modułów rozszerzeń typu Pmod. Konieczność ingerencji w kod opisu procesora w celu zaprojektowania dodatkowych modułów czy też usunięcia bądź optymalizacji istniejących pod kątem konkretnego zadania zmusza uczestnika procesu dydaktycznego do spojrzenia z innej perspektywy na proces projektowy, który traktuje niewielkich co prawda rozmiarów system komputerowy jako spójną całość.

6. Literatura

[1] Nowakowski M.: PicoBlaze. Mikroprocesor w FPGA. BTC, Legionowo 2010.
[2] Xilinx (Chapman K.): PicoBlaze 8-Bit Microcontroller for Virtex-E and Spartan-II/IIE Devices, Application Note. 2003, (http://www.xilinx.com/support/documentation/application_notes/xapp213.pdf, dostęp: marzec 2011).
[3] Xilinx: PicoBlaze 8-Bit Microcontroller for CPLD Devices, Application Note. 2003, (http://www.xilinx.com/support/documentation/application_notes/xapp387.pdf, dostęp: marzec 2011).
[4] Xilinx: CoolRunner-II Evaluation Board Reference Manual. 2008, (http://www.xilinx.com/support/documentation/boards_and_kits/ug501.pdf, dostęp: marzec 2011).
[5] Xilinx: Programmable Logic Design. Quick Start Guide. 2008, (http://www.xilinx.com/support/documentation/boards_and_kits/ug500.pdf, dostęp: marzec 2011).

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