PicoBlaze: przygotowywanie programów w środowisku pBlazIDE, część 1

Opublikowany w EP5 i 6/2005 opis „miękkiego" mikrokontrolera PicoBlaze wywołał wśród Czytelników spore zainteresowanie, co objawiło się dużą liczbą pytań związanych ze szczegółami stosowania tego rdzenia w praktycznych aplikacjach. Odpowiedzi na część nadesłanych pytań przedstawiamy w artykule.

Mikrokontroler PicoBlaze jest idealnym rozwiązaniem dla tych konstruktorów, którzy nie boją się nowoczesnych rozwiązań i chcą korzystać z zalet rozwiązań sprzętowo – programowych. Podobne cechy użytkowe, choć na inną skalę, oferują (wymieniam tylko najbardziej popularne) mikrokontrolery PSoC firmy Cypress, układy FPSLIC firmy Atmel oraz przejęte przez chińską firmę Zylogic układy CSoC firmy Triscend (informacje o wszystkich wymienionych układach można znaleźć w archiwalnych wydaniach EP). Co ciekawe, specyfika projektowania programowo – sprzętowego została uwzględniona m.in. w Protelu DXP i DXP2004, ale do naszych celów nie ma konieczności korzystania z tak potężnego narzędzia. Korzystanie z PicoBlaze'a jest po prostu bardzo łatwe!

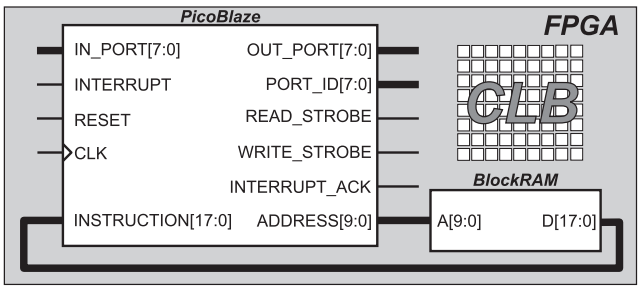
Jak przygotować program dla PicoBlaze'a

Wiele pytań dotyczyło tego właśnie zagadnienia. Opis przedstawiony w EP6/2005 z konieczności był skrótowy, więcej szczegółów przedstawiam poniżej.

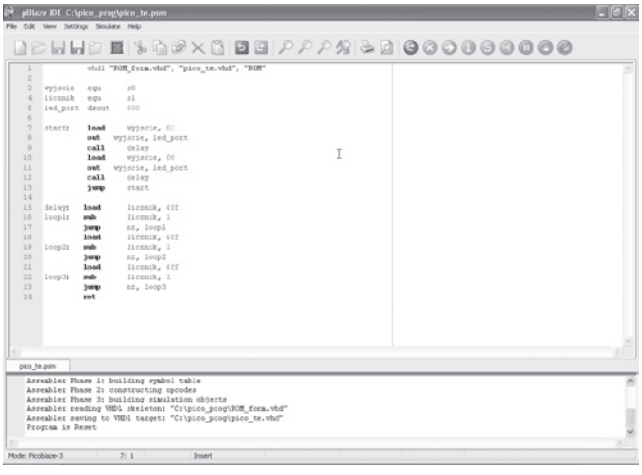
W przypadku implementacji mikrokontrolera PicoBlaze w układach Spartan 3 najrozsądniejszym sposobem przechowywania programu jest wykorzystanie pamięci *BlockRAM* (jednego jej bloku) skonfigurowanej jako pamięć ROM (**rys. 1**).

Opis HDL PicoBlaze'a udostępniony przez firmę Xilinx w wersji dla rodziny układów Spartan 3, został zoptymalizowany do takiej właśnie konfiguracji pamięci programu, stąd „sprzętowy" obszar adresowy jest 10–bitowy, a szerokość słowa odczytywanego z pamięci programu wynosi 18 bitów. Nic oczywiście nie stoi na przeszkodzie w innym zorganizowaniu pamięci programu, ale biorąc pod uwagę proponowaną sprzętową platformę docelową wykorzystującą układ Spartan 3, wszelkiego typu „kombinacje" nie są w rzeczywistości opłacalne i to pod wieloma względami. Skupimy się zatem na prezentacji sposobu przygotowania programu dla mikrokontrolera wbudowanego w FPGA w sposób standardowy, wykorzystującego jako pamięć programu blok konfigurowalnej pamięci SRAM – *BlockRAM*.

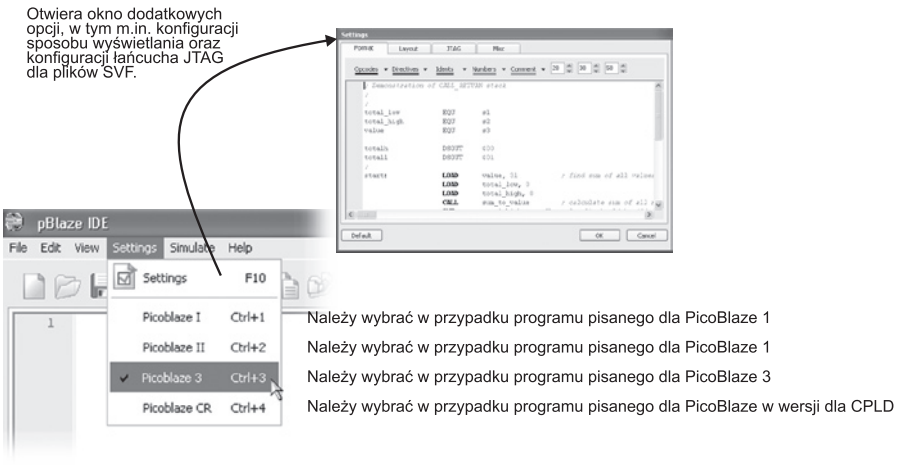
Środowiskiem zalecanym do przygotowywania programów dla PicoBlaze'a jest bezpłatny pBlazIDE, którego wersje 3.6 i 3.74β publikujemy na CD EP8/2005B. Jest to kompilator – symulator asemblera, którego mnemoniki różnią się nieco od tego, co zaproponował autor PicoBlaze'a (patrz EP6/2005),



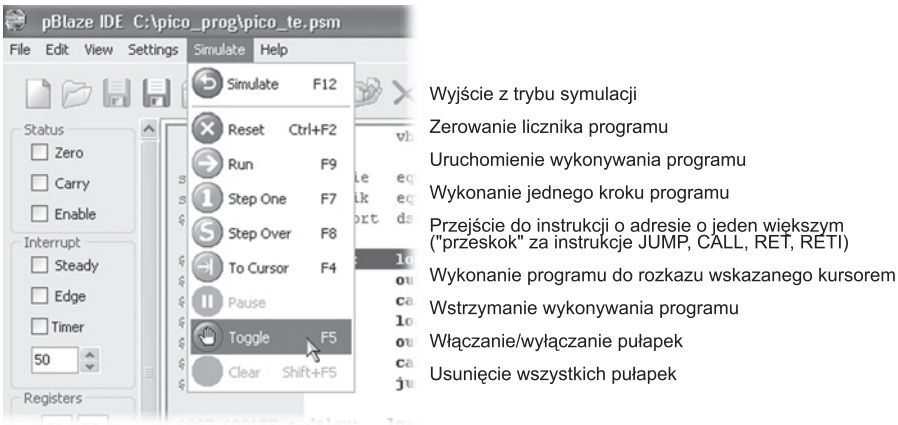
Rys. 1. Zalecany sposób połączenia PicoBlaze'a z pamięcią programu



Rys. 2. Widok głównego okna programu pBlazIDE



Rys. 3. Opcje dostępne w menu *Settings*



Rys. 4. Opcje dostępne w menu *Simulate*

ale zmiany mają charakter wyłącznie kosmetyczny i nie wpływają na działanie mikrokontrolera.

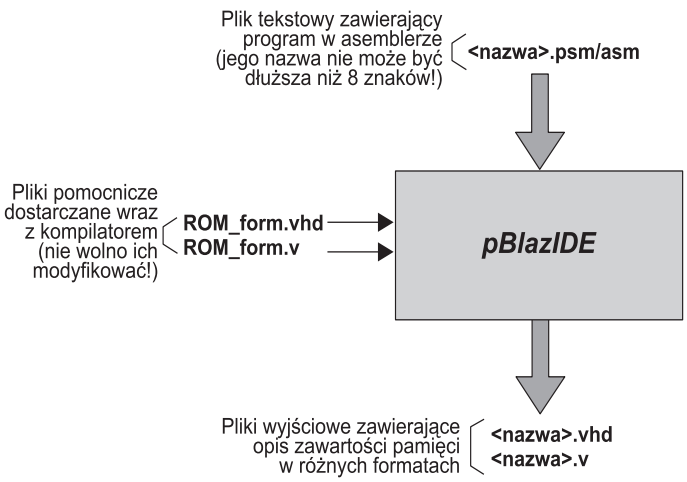
Widok głównego okna programu pBlazIDE pokazano na **rys. 2**. Operacje na plikach przebiegają w sposób typowy dla aplikacji Windows (menu *File* i odpowiednie ikony), specjalnego omówienia nie wymagają także opcje w menu *Edit* oraz *View*. Nieco więcej uwagi trzeba poświęcić opcjom w menu *Settings* i *Simulate*, których wygląd i funkcje przedstawiono na **rys. 3** i **4**. W naszym przypadku w menu *Settings* należy zaznaczyć opcję *Picoblaze 3*, co spowoduje, że plik wynikowy będzie zawierał deklaracje binarnych kodów w formacie akceptowanym przez PicoBlaze 3. Ustawienia pozostałych opcji zależy od indywidualnych upodobań użytkownika programu, za wyjątkiem zakładki *JTAG* okna opcji z rys. 3. W przypadku konfiguracji układu FPGA za pomocą pliku w formacie innym niż SVF ustawienia tych opcji są bez znaczenia. Dostęp do poszczególnych opcji jest możliwy także za pomocą skrótów klawiszowych lub za pośrednictwem ikon rozmieszczonych w dolnej części paska narzędziowego.

Na **rys. 5** pokazano obieg plików podczas pracy z pBlazIDE. Kompilator generuje plik wynikowy (HDL!) w postaci opisu budowy i zawartości pamięci ROM w języku VHDL lub Verilog. Symbol graficzny kompletnej pamięci w oknie edytora schematów z pakietu WebPack ISE pokazano na **rys. 6**. Uzyskanie takiego opisu wymaga dopilnowania, żeby w domyślnym katalogu projektu znajdował się plik wzorcowy zawierający „szkielet" opisu pamięci ROM (*ROM_form.v* lub *ROM_form.vhd* – dostarczane wraz z kompilatorem), należy go ponadto zadeklarować w pliku asemblerowym za pomocą dyrektywy (w przypadku VHDL–a):

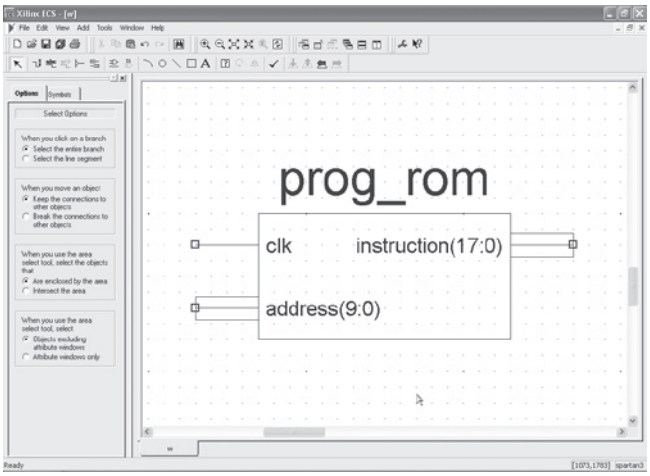
vhdl „ROM_form.vhd", „pico_te.vhd", „ROM"

Generacja pliku wynikowego (jego nazwa jest podawana jako drugi parametr dyrektywy przedstawionej powyżej) odbywa się w chwili zainicjowania symulacji programu (za pomocą klawisza F12, opcji w menu lub ikony w pasku narzędziowym). Ostatni parametr dyrektywy określa nazwę generowanej jednostki *entity* w języku VHDL.

Piotr Zbysiński, EP
piotr.zbysinski@ep.com.pl



Rys. 5. Obieg plików w środowisku pBlazIDE



Rys. 6. Symbol graficzny pamięci ROM w oknie edytora schematów WebPack ISE, uzyskany z pliku HDL