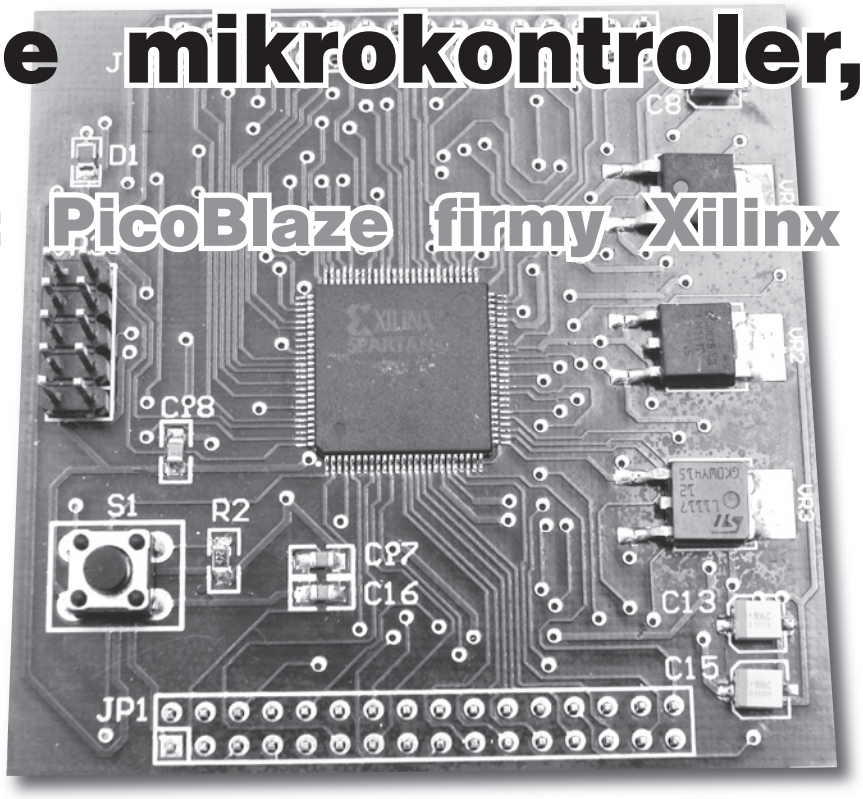
Zrób sobie mikrokontroler, część 2

RISC w VHDL: PicoBlaze firmy Xilinx

AVT-457



Zgodnie z obietnicami, w tej części artykułu przedstawiamy uniwersalną platformę sprzętową wykonaną na układzie FPGA z rodziny Spartan 3, która może posłużyć między innymi do implementacji mikrokontrolera PicoBlaze. Pokażemy w paru krokach jak to zrobić, przy okazji prezentując także dostępne bezpłatnie narzędzia projektowe.

Rekomendacje:

opracowanie przeznaczone zarówno dla entuzjastów PLD, jak i wszelkiej maści techniki cyfrowej. Mikrokontrolery potrafią przecież wszystko!

Platforma sprzętowa

Jak wspomniałem w poprzedniej części artykułu, PicoBlaze jest „miękkim" mikrokontrolerem, opisanym w języku VHDL. „Miękkim" oznacza, że użytkownik może samodzielnie wybrać (oczywiście w określonych granicach) typ układu PLD, w jakim mikroprocesor zostanie zaimplementowany. W wersji prezentowanej w artykule, PicoBlaze zoptymalizowano pod kątem aplikowania w układach z rodzin: Spartan 3 oraz Virtex II i Virtex II PRO, lecz drobne modyfikacje plików źródłowych (nie jest to zgodne z licencją!) pozwalają zaimplementować go także w układach PLD produkowanych przez firmy inne niż Xilinx. Dodatkową, choć łatwą do pokonania, trudnością jest zoptymalizowanie kompilatora asemblera pod kątem układów firmy Xilinx. Piszę o tym wyłącznie po to, żeby pokazać uniwersalność i elastyczność rozwiązań IP–core'owych.

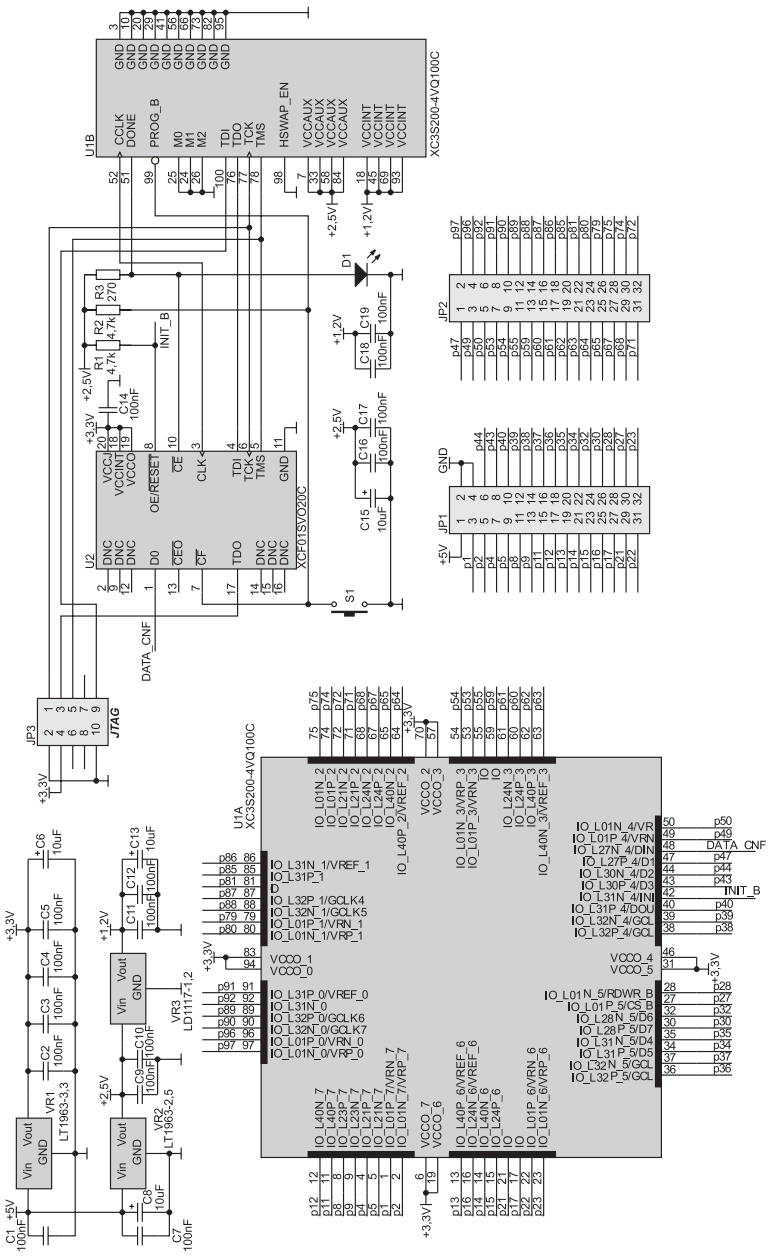
Poznanie zarówno samego PicoBlaze'a, jak i możliwości współczesnych układów, ułatwi prosta platforma sprzętowa, zbudowana na bazie układu XC3S200VQ100 firmy Xilinx. Jest to drugi „od dołu" układ z rodziny Spartan 3, charakteryzujący się doskonałym wyposażeniem wewnętrznym i niską ceną (patrz tab. 1, EP5/2005). Zaimple-

PODSTAWOWE PARAMETRY

Płytka o wymiarach 56 x 56 mm
Zasilanie +5 V/200 mA
Możliwość zastosowania układu XC3S50 lub XC3S200
Wbudowany konfigurator XCF01S
Liczba dostępnych linii I/O: 61

W ofercie AVT są dostępne:
- [AVT-457C] moduł zmontowany i uruchomiony

Listingi do tego artykułu są umieszczone na płycie CD EP6/2005 oraz na stronie http://download.ep.com.pl



Rys. 6. Schemat elektryczny zestawu

mentowanie w tym układzie mikroprocesora PicoBlaze zabiera zaledwie 5% zasobów logicznych i jeden blok pamięci BlockRAM.

Schemat elektryczny urządzenia pokazano na **rys. 6**. Na płytce drukowanej, oprócz układu XC3S200, zamontowano także pamięć konfigurującą XCF01S (typu Flash), która służy do przechowywania danych o konfiguracji układu FPGA i umożliwia automatyczne odtworzenie pamięci konfigurującej SRAM wbudowanej w układ FPGA po każdorazowym włączeniu zasilania. Zawartość pamięci Flash można modyfikować za pomocą prostego programatora ISP, dołączonego do złącza JP3. Pamięć Flash i pamięć konfigurująca SRAM wbudowana w FPGA są połączone szeregowo w łańcuchu JTAG, którego budowę pokazano na **rys. 7**. Pierwszym układem (od strony wejścia TDI) jest XC3S200, natomiast drugim – konfigurator XCF01S. Dodatkowo – jak widać na rys. 7 – konfigurator jest dołączony do FPGA za pomocą specjalnego interfejsu szeregowego, który służy do automatycznego konfigurowania układu FPGA po włączeniu zasilania. Taki zestaw połączeń umożliwia użytkownikowi przeprowadzenie następujących operacji:

- programowania (via JTAG) pamięci Flash, której zawartość będzie automatycznie kopiowana do pamięci konfigurującej FPGA po każdym włączeniu zasilania,
- bezpośredniego modyfikowania zawartości (via JTAG) pamięci konfigurującej SRAM układu FPGA, dzięki czemu można prowadzić niedestrukcyjne (jak w przypadku pamięci Flash, która po każdym kasowaniu się degraduje), wielokrotne modyfikacje jej zawartości,
- automatycznego konfigurowania FPGA poprzez skopiowanie zawartości pamięci Flash do pamięci konfigurującej (proces ten można zainicjować ręcznie poprzez wciśnięcie S1).

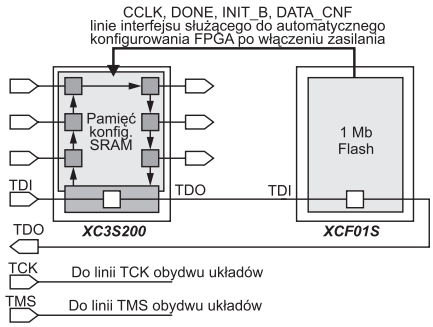
Pierwsze dwie z wymienionych operacje można wykonać za pomocą taniego i łatwego w wykonaniu programatora, którego schemat elektryczny przedstawiamy na **rys. 8**

Da się, lecz nie wolno

PicoBlaze da się zaimplementować także w układach innych producentów niż Xilinx, lecz wymaga to pewnych modyfikacji kodu źródłowego i nie jest zgodne z licencją.

Bezpłatny rdzeń

Pliki źródłowe, dokumentację w postaci elektronicznej oraz kompilator asemblera dla mikrokontrolerów PicoBlaze można pobrać ze strony internetowej firmy Xilinx: *http://www.xilinx.com/ipcenter/processor_central/picoblaze/index.htm*.



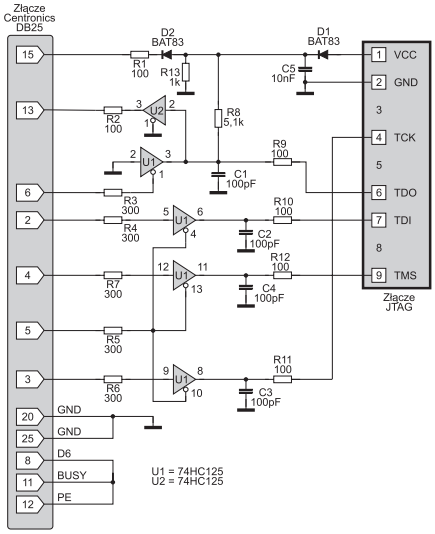
Rys. 7. Konfiguracja łańcucha JTAG w zestawie

List. 3. Listing przykładowego programu napisanego w asemblerze PicoBlaze

```
vhdl „ROM_form.vhd", „pico_te.vhd",
„PROG_ROM"
wyjscie equ   s0
licznik equ   s1
led_port   dsout $00

start:  load  wyjscie, 01
  out   wyjscie, led_port
  call  delay
  load  wyjscie, 00
  out   wyjscie, led_port
  call  delay
  jump  start

delay:  load  licznik, $ff
loop1:  sub   licznik, 1
  jump  nz, loop1
  load  licznik, $ff
  ret
```



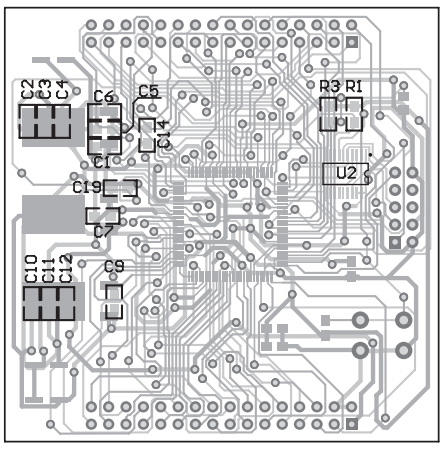
Rys. 8. Schemat elektryczny programatora ISP dla układów firmy Xilinx

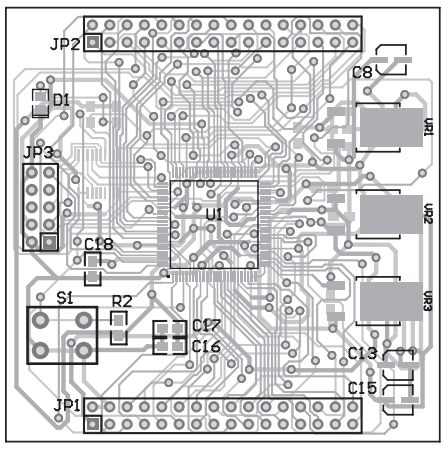
(szczegółowo opisany w Miniprojektach w EP4/2001, można zastosować także nowocześniejszy UnISProg, który został opisany w EP1/2004).

Dioda świecąca D1 sygnalizuje poprawne skonfigurowanie układu FPGA.

Montaż i uruchomienie

Urządzenie zmontowano na dwustronnej płytce drukowanej, której schematy montażowe (elementy są rozlokowane na obydwu stronach PCB) pokazano na **rys. 9**. Ze względu na zastosowanie układów o bardzo małych odstępach pomiędzy wyprowadzeniami (0,5 mm), ręczny montaż płytek jest mocno utrudniony – osoby mające niewielką wprawę w lutowaniu powinny pomyśleć o zakupie gotowych (zmontowanych) modułów. Sposobem lutowania ręcznego dającym niezłe wyniki (mam tu na myśli wyłącznie skuteczność) jest – po dokładnym wypozycjonowaniu układu na punktach lutowniczych – przylutowanie go z użyciem dużej ilości stopu lutowniczego, którego nadmiar jest później usuwany za pomocą miedzianej plecionki (najlepiej ze sproszkowaną kalafonią). Po lutowaniu płytkę trzeba dokładnie umyć, na przykład za pomocą rozpuszczalnika Kontakt PCC (dostarczanego w butelce ciśnieniowej ze szczotką).





Rys. 9. Schemat montażowy zestawu

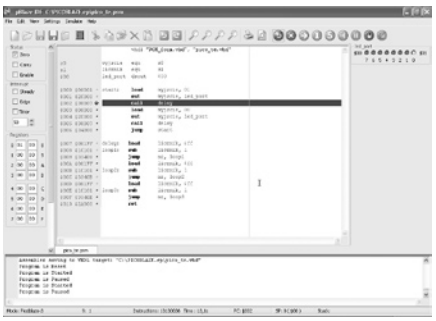
Narzędzia projektowe

Korzystanie z PicoBlaze'a wymaga zastosowania zestawu dwóch programów narzędziowych, spośród trzech które są dostępne bezpłatnie:

- WebPacka ISE w dowolnej wersji powyżej 3.3WP8.0 – udostępniony przez firmę Xilinx (*www.xilinx.com*),
- kompilatora asemblera dla PicoBlaze'a (*kcpsm3.exe*) – udostępniony przez firmę Xilinx (*www.xilinx.com*),
- opcjonalnie można skorzystać z IDE z symulatorem PicoBlaze'a – program pBlazIDE opracowany i udostępniony bezpłatnie przez holenderską firmę Mediatronix (*www.mediatronix.com/tools*).

Niezależnie od tego czy będziemy stosować liniowy kompilator asemblera, czy też zostanie wykorzystane środowisko zintegrowane (widok okna pBlazIDE pokazano na **rys. 10**), narzędziem niezbędnym do zaimplementowania projektu w układzie PLD jest WebPack – kompletne środowisko projektowe dla układów firmy Xilinx. To właśnie WebPack odpowiada za prawidłową implementację samego procesora i wykorzystywanej przez niego pamięci programu w strukturę docelowego układu PLD.

Na **rys. 11** zilustrowano pracę asemblera *kcpsm3.exe*, który na „wejściu", oprócz pliku źródłowego potrzebuje także trzech plików zawierających „formatki" opisu pamięci w językach Verilog (*ROM_form.v*)



Rys. 10. Widok okna programu pBlazIDE

i VHDL (*ROM_form.vhd*) oraz w formacie *COEfficient* (*ROM_form.coe*), stosowanym (rzadko) przez niektóre narzędzia syntezy logicznej. Pliki zawierające „formatki" udostępnia firma Xilinx wraz z asemblerem, ich zawartości zalecam nie modyfikować.

Projektantów przyzwyczajonych do Windows (czyli – jak szacuję – 99,9 % czytających) „surowy" interfejs (w zasadzie jego brak) asemblera *kcpsm3.exe* z pewnością będzie deprymował. Wszystkich zdeprymowanych zachęcam do sięgnięcia po program pBlazIDE, którego efekty działania są w zasadzie identyczne jak *kcpsm3*.

Różnice w menmonikach

Mnemoniki niektórych rozkazów rozpoznawanych przez pBlazIDE są inne niż ustalone pierwotnie przez autora PicoBlaze'a. Różnice zestawiono w tab. 4.

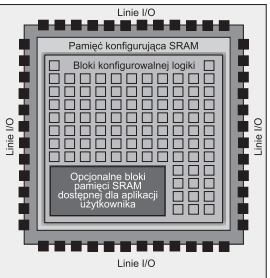
Kurs obsługi WebPacka (i nie tylko!) dla każdego

Czytelnikom zamierzającym poznać „z bliska" system projektowy WebPack firmy Xilinx polecamy książkę „Układy programowalne, pierwsze kroki" (*http://www.btc.pl/index.php?id=uppk*), dostępną w ofercie handlowej AVT. Książka zawiera CD–ROM z WebPackiem i narzędziami projektowymi firmy Altera.

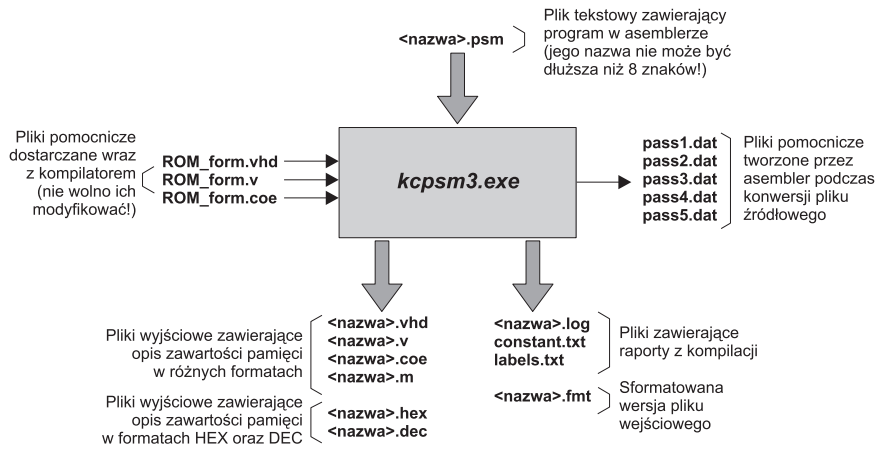
FPGA i wyłączanie zasilania

Z małymi wyjątkami, układy FPGA mają przedziwną konstrukcję: za ich konfigurację (czyli spełnianą funkcję) odpowiada pamięć SRAM, która jest pamięcią ulotną – nie potrafi przechowywać danych bez zasilania. Tak więc, bez dodatkowych zabiegów, FPGA po włączeniu zasilania... nie może poprawnie działać!

Jak rozwiązano ten problem? Otóż przez krótki czas (do kilkuset milisekund od chwili włączenia zasilania) FPGA jest konfigurowany, co oznacza po prostu odtworzenie poprawnej zawartości konfigurującej pamięci SRAM. Dostępnych jest wiele sposobów konfiguracji, z których największą popularnością w praktyce cieszą się wyspecjalizowane, zewnętrzne pamięci Flash z wyjściem szeregowym, które są przystosowane do automatycznego konfigurowania układów FPGA.



Należy pamiętać, że niektóre rodziny układów FPGA są wyposażone w drugi rodzaj pamięci SRAM, która – w odróżnieniu od pamięci konfigurującej – jest dostępna dla aplikacji użytkownika. W układach Spartan 3 pamięć ta jest podzielona na bloki o nazwie BlockRAM.



Rys. 11. Obieg plików projektowych przy uzyciu asemblera kcpsm3.exe

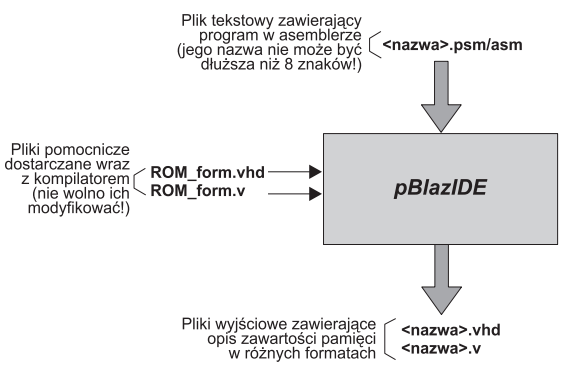
exe (**rys. 12**), a daje on dodatkowo możliwość symulowania pracy PicoBlaze'a! pBlazeIDE obsługuje dwa formaty plików wejściowych i wyjściowych – zapisanych w VHDL lub w Verilogu. Zapewnia to wystarczającą uniwersalność narzędzia, ponieważ wszystkie znane mi współczesne syntezery języków HDL radzą sobie z implementacją pamięci ROM. pBlazeIDE generuje pliki wyjściowe zawierające opis zawartości pamięci na podstawie szablonów (*ROM_form.vhd* i *ROM_form.v*). Konieczne jest użycie w pliku **.psm* dyrektywy:

```
vhdl „ROM_form.vhd”, „pico_te.vhd”
```

lub (wbrew sugestii zawartej w nazwie dyrektywy)

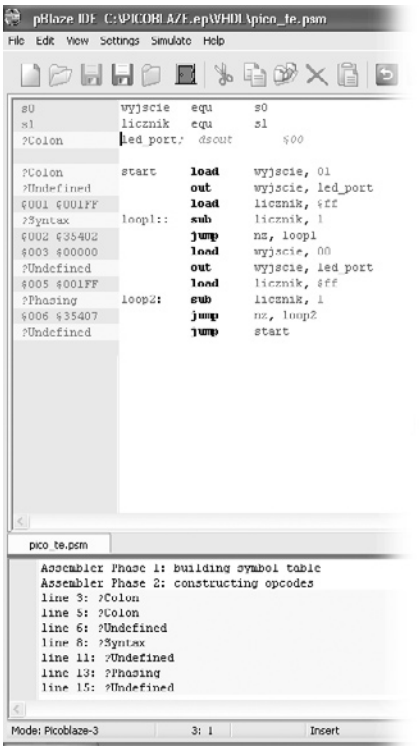
```
vhdl „ROM_form.v”, „pico_te.v”,
```

gdzie *pico_te.vhd* i *pico_te.v* to nazwy plików wynikowych. Pliki wynikowe są generowane w chwili zainicjowania pracy symulatora, a ewentualne błędy są automatycznie wskazywane przez program, jak to widać na **rys. 13** (opis po znaku „?” w lewej części okna).



Rys. 12. Obieg plików projektowych przy użyciu pBlazIDE

Warto zwrócić uwagę, że mnemoniki niektórych poleceń rozpoznawane przez pBlazIDE różnią się od przyjętych jako standardowe w liniowej wersji asemblera. Nie są to duże



Rys. 13. pBlaz wyraźnie wskazuje błędy wykryte w programie

różnice, do tego częściowo zniwelowane przez wbudowany w pBlazIDE, ale warto o nich wiedzieć, dlatego w **tab. 4.** znajduje się ich zestawienie.

Mamy program – implementacji krok następny

Po napisaniu i przesymulowaniu programu nadchodzi czas implementacji rdzenia procesora i pamięci programu wraz z jej zawartością. Prosty program przykładowy, służący do zilustrowania implementacji pokazano na **list. 3**. Spełnia on niezwykle „fantazyjne” zadanie migania diodą LED dołączoną do portu wyjściowego o adresie 00. Kody poleceń dla procesora są zapisane (w prezentowanym przykładzie) w pliku *pico_te.vhd* (jego najważniejsze frag-

Tab. 4. Różnice pomiędzy mnemonikami rozpoznawanymi przez pBlazIDE i kompilator KCPSM

pBlazIDE	KCPSM
RET	RETURN
RETI	RETURNI
ADDC	ADDCY
SUBC	SUBCY
IN	INPUT
OUT	OUTPUT
EINT	ENABLE INTERRUPT
DINT	DISABLE INTERRUPT

menty pokazano na **list. 4**, całość udostępniamy na CD–EP6/2005B).

Opis HDL PicoBlaze'a domyślnie „domaga" się pliku zawierającego jednostkę (*entity*) PROG_ROM (**rys. 14**). W przygotowanym przez nas pliku *pico_te.vhd* tak właśnie została ona nazwa, dzięki czemu po dołączeniu tego pliku do projektu (**rys. 15**), można go od razu poddać kompilacji. Jeżeli zakończy się ona sukcesem, należy jeszcze przypisać nazwy sygnałów do odpowiednich wyprowadzeń układu FPGA i projekt ponownie skompilować. W projekcie udostępnionym na płycie przypisania sygnałów nie zostały wymuszone, co wynika z uniwersalnej budowy płytki uruchomieniowej.

Podczas prób należy zwrócić uwagę na to, że maksymalne częstotliwości taktowania PicoBlaze'a są dość wysokie, a procedura opóźnienia określająca czas trwania świecenia i wygaszenia LED trwa zaledwie 1030 taktów zegara. W związku z tym podczas pierwszych prób zalecam stosowanie źródła sygnału zegarowego o częstotliwości 5...10 kHz lub zmodyfikowanie procedury opóźnienia (choćby w sposób pokazany na **list. 5**), żeby wydłużyć czas jej trwania (opóźnienie zabiera 261632 taktów zegara), ewentualnie weryfikować pracę procesora za pomocą oscyloskopu dołączonego do linii portu, na której jest generowany sygnał prostokątny.

List. 4. Najważniejsze fragmenty pliku zawierającego opis programu

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;

library unisim;
use unisim.vcomponents.all;

entity PROG_ROM is
    Port (      address : in std_lo-
gic_
vector(9 downto 0);
           instruction : out std_lo-
gic
_vector(17 downto 0);
                  clk : in std_lo-
gic);
    end PROG_ROM;

architecture low_level_definition of
PRO
G_ROM is
attribute INIT_00 : string;
attribute INIT_01 : string;
attribute INIT_02 : string;
……….
attribute INITP_05 : string;
attribute INITP_06 : string;
attribute INITP_07 : string;

attribute INIT_00 of ram_1024
x_18  : label is „540EC10101F-
F540BC10101FF5408C10101F-
F40000007C00000000007C0000001";
attribute INIT_01 of ram_1024_x_18  :
label is „0000000000000000000000000
0000000000000000000000000000000000A-
000";
attribute INIT_02 of ram_1024_x_18  :
label is „0000000000000000000000000
0000000000000000000000000000000000";
……….
attribute INITP_07 of ram_1024_x_18 :
label is „0000000000000000000000000
0000000000000000000000000000000000";

begin

 generic map ( INIT_00 => X"540E-
C10101FF540BC10101FF5408C10101F-
F40000007C00000000007C0000001",
               INIT_01 => X"000000000
0000000000000000000000000000000000
0000000000000A000",
               INIT_02 => X"000000000
0000000000000000000000000000000000
0000000000000000",
……….
……….
……….
X"000000000000000000000000000000000000
0000000000000000000000000000",
               INIT_3E => X"000000000
0000000000000000000000000000000000000
0000000000000000",
               INIT_3F => X"000000000
0000000000000000000000000000000000000
0000000000000000",
              INITP_00 => X"000000000
0000000000000000000000000000000000000
000000002D34D3E38",
              INITP_01 => X"000000000
0000000000000000000000000000000000000
0000000000000000",
……….
……….
              INITP_06 => X"000000000
0000000000000000000000000000000000000
0000000000000000",
              INITP_07 => X"000000000
0000000000000000000000000000000000000
0000000000000000")

port map(    DI => „0000000000000000",
             DIP => „00",
              EN => ‚1',
              WE => ‚0',
             SSR => ‚0',
             CLK => clk,
            ADDR => address,
              DO => instruction(15
downto 0),
             DOP => instruction(17
downto 16));
end low_level_definition;
```

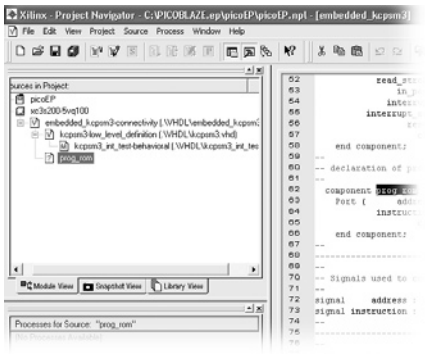
List. 5. Zmodyfikowana wersja programu testowego

```
Wyjscie equ   s0
licznik1   equ   s1
licznik2   equ   s2
led_port   dsout $00

start:  load  wyjscie, 01
  out   wyjscie, led_port
  call  delay
  load  wyjscie, 00
  out   wyjscie, led_port
  call  delay
  jump  start

delay:  load  licznik2, $ff
loop1:  load  licznik1, $ff
loop2:  sub   licznik1, 1
  jump  nz, loop2
  sub   licznik2, 1
  jump  nz, loop1
  ret
```

WYKAZ ELEMENTÓW
Rezystory
(obudowy 0805)
R1, R2: 4,7 kΩ
R3: 270 Ω
Kondensatory
(obudowy 0805/A)
C1, C2, C3, C4, C5, C7, C9, C10, C11, C12, C14, C16, C17, C18, C19: 100 nF
C6, C8, C13, C15: 10 µF/10 V
Półprzewodniki
U1: XC3S200–5VQ100
U2: XCF01SVO20C
VR1: LT1963–3,3
VR2: LT1963–2,5
VR3: LD1117–1,2
D1: LED (0805)
Inne
JP1, JP2: gold–piny 2x16
JP3: ZWS10

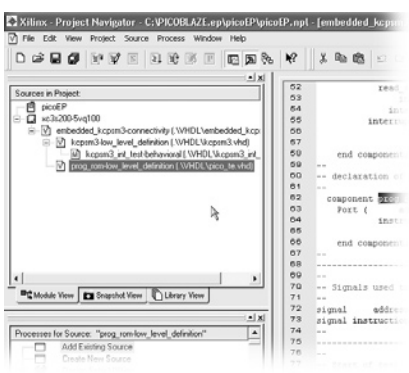


Rys. 14. PicoBlaze ma przedefiniowaną pamięć w jednostce PROG-ROM

Podsumowanie

Po kilkudniowych, bardzo intensywnych, testach PicoBlaze'a doszedłem do wniosku, że jest to obecnie najlepszy, pośród dostępnych bezpłatnie, rdzeń mikrokontrolera na znanym mi „świecie". Ze względu na dużą wydajność, stabilność działania, kompletną listę instrukcji (zwłaszcza w wersji na rodzinę Spartan 3), dobrej jakości narzędzia projektowe (także dostępne bezpłatnie), niewielkie zasoby wymagane do pełnej implementacji projektu, do tego dobrze przygotowany opis HDL (gwarantujący przenośność pomiędzy różnymi platformami) PicoBlaze jest wart polecenia wszystkim użytkownikom układów PLD, którzy preferują wygodne, nowoczesne rozwiązania sprzętowo – programowe (*hardware – software codesign*).

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



Rys. 15. Widok okna WebPacka po zadeklarowaniu pliku z opisem pamięci programu