Zrób sobie mikrokontroler, część 1

RISC w VHDL: PicoBlaze firmy Xilinx

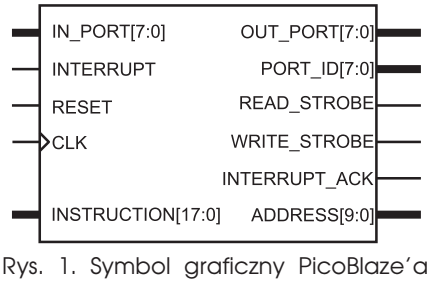
Koniec nudów! Koniec z (aplikacyjnym) panowaniem AVR–ów, mikrokontrolerów '51, czy też PIC–ów! Jeżeli chcesz być projektantem nowoczesnej oryginalnej elektroniki i sprawia Ci przyjemność bycie w awangardzie, to masz niepowtarzalną szansę, którą stworzył i udostępnił jeden z inżynierów firmy Xilinx!

Rekomendacje:

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

Realizacja projektów z wykorzystaniem układów PLD ma moim zdaniem jedną poważną niedogodność: sterowanie wieloma urządzeniami współpracującymi z układami PLD (jak choćby wyświetlaczami LCD ze sterownikiem HD44870) wymaga tworzenia specyficznych automatów sterujących, spełniających rolę interfejsów. Co prawda, na początku zabawy z PLD większość projektantów traktuje tworzenie tych automatów jako interesujące wyzwanie, ale z czasem (zwłaszcza, gdy projekty zaczynają „rosnąć") żmudne opisywanie sekwencji stanów zaczyna być poważnym utrudnieniem, często zniechęcającym do zastosowania układu programowalnego.

Co z tym można zrobić? Pierwszym, wdrożonym przeze mnie, pomysłem pozwalającym „obejść" konieczność budowania takich automatów było opisanie w VHDL uniwersalnego, mikroprogramowalnego automatu, który wykonywał programy składające się z 8 poleceń. Możliwości tego automatu wyczerpały się dość szybko i konieczne okazało się znaczne zwiększenie liczby dekodowanych poleceń, co „trąciło" koniecznością opracowania własnego mikrokontrolera i – oczywiście – odpowiednich narzędzi programowych (jak kompilator asemblera czy programowy symulator). Poddałem się. Z odsieczą przyszedł pomysł wdrożony przez Kena Chapmana, który jest jednym z brytyjskich inżynierów firmy Xilinx. Kilka lat temu napotkał na podobne problemy i – jak się okazało – w podobny sposób zaczął je rozwiązywać. Dość szybko stwierdził, że budowanie mikroprogramowalnego automatu nie ma sensu, w związku z czym z pierwotnie prymitywnego programowalnego automatu *Constant (k) Coded Programmable State Machine* – KCPSM powstał prawdziwy 8–bitowy mikrokontroler RISC. Autor nazwał go PicoBlaze, co jest nawiązaniem do komercyjnego, 32–bitowego procesora MicroBlaze (oferowanego przez firmę Xilinx). To właśnie PicoBlaze (którego symbol schematowi pokazano na **rys. 1**) jest bohaterem naszego artykułu. Autor (z przyzwyczajenia?) swoje dzieło nadal nazywa KCPSM, ale jego najnowsza wersja nosi numer 3 (KCPSM3).



Rys. 1. Symbol graficzny PicoBlaze'a

Był sobie inżynier…

…który nadmiar wolnego entuzjazmu ulokował w opisanie w języku VHDL rdzenia niewielkiego (co do wymagań sprzętowych) lecz bardzo uniwersalnego, 8–bitowego mikrokontrolera RISC. Człowiek ten nazywa się Ken Chapman, a jego dzieło – po wielu ewolucjach – PicoBlaze. Jak przekonają się uważni Czytelnicy, PicoBlaze jest „pico” tylko w odniesieniu do wymagań stawianych użytkownikowi, jego komputerowi, docelowemu układowi PLD i nakładom, jakie należy ponieść, żeby móc go stosować. Możliwości tego mikrokontrolera są co najmniej „Mega”.

RISC–owy mikrokontroler w VHDL?

PicoBlaze jest „miękkim” mikrokontrolerem, co oznacza, że jest dostępny wyłącznie w postaci opisu w jednym z najpopularniejszych języków opisu sprzętu – VHDL. Nie ma więc możliwości kupienia gotowego układu z PicoBlaze „w środku” w jakimkolwiek sklepie, ale dzięki dostępności plików źródłowych z jego opisem każdy projektant (przy założeniu, że trochę się zna na układach PLD i podstawowych narzędziach projektowych) ma możliwość zastosowania go we własnym urządzeniu (podobnie jak mikrokontroler 8051, którego implementację VHDL–ową przedstawiliśmy w EP2 i 3/2005). Warto zwrócić uwagę, że obecnie dostępna wersja PicoBlaze'a jest pełnowartościowym (bez żadnych ograniczeń funkcjonalnych lub czasowych!), 8–bitowym mikrokontrolerem RISC, do którego są dostępne narzędzia w postaci środowiska IDE (dla Windows), prostego symulatora funkcjonalnego oraz kompilator asemblera.

Jaki jest praktyczny sens stosowania PicoBlaze'a? Ze względu na niewielkie zajmowane przez niego zasoby, możliwość „wbudowania” pamięci programu w układ PLD, a przy tym dużą wydajność, mikrokontroler ten doskonale nadaje się do realizowania wszelkich zadań typowych dla „dyskretnych” mikrokontrolerów oraz wielostanowych automatów. W przypadku wykorzystania jako platformy sprzętowej układu FPGA, można w nim zaimplementować kilka niezależnych procesorów PicoBlaze.

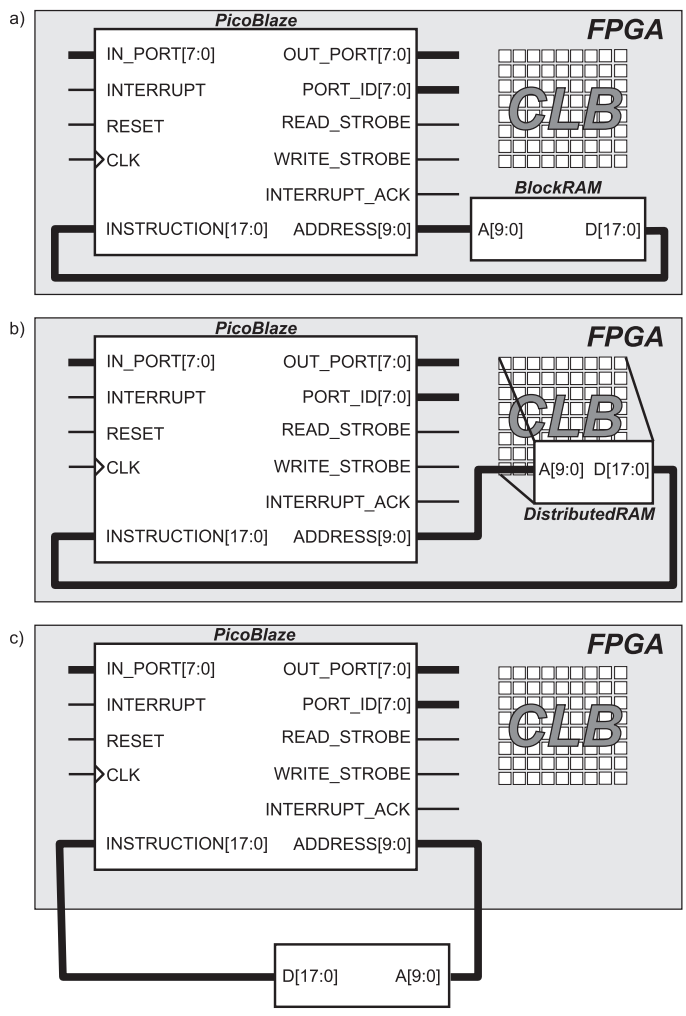
W zależności od wymagań użytkownika i możliwości platformy sprzętowej, pamięć programu może być zaimplementowana w:

- wyspecjalizowanych blokach BlockRAM (konfigurowalne bloki pamięci, w których można emulować pamięć ROM i implementować inne rodzaje pamięci RAM: jedno– i dwuportowe FIFO, klasyczne SRAM itp.),
- przerzutnikach wchodzących w skład konfigurowalnych bloków logicznych (mikrokomórki, *slice*, itp.) – jest to tzw. implementacja rozproszona (*distributed RAM*),
- zewnętrznej, równoległej pamięci Flash, EPROM, EEPROM., dołączonej do mikrokontrolera za pomocą magistral wyprowadzonych na nóżki FPGA.

Warianty te przedstawiono na **rys. 2**. Najprostszym koncepcyjnie (bo przeniesiony wprost z typowych sys-

Dla tych to co piszą w Verilogu

Ken Chapman przygotował alternatywną, w stosunku do prezentowanej w artykule, wersję PicoBlaze'a napisaną w Verilogu. Jest ona dostępna w tym samym archiwum co wersja VHDL, wraz z dokumentacją i kompilatorem asemblera.

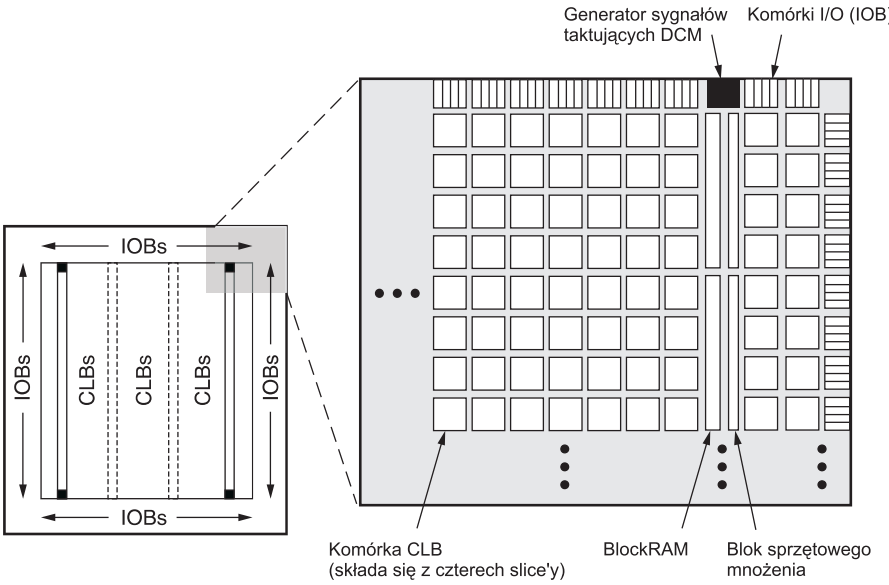


Rys. 2. Możliwe sposoby dołączenia pamięci programu do mikrokontrolera PicoBlaze

Tab. 1. Zestawienie najważniejszych parametrów wybranych układów FPGA z rodziny Spartan 3

Typ układu	Liczba zintegrowanych w układzie bramek	Liczba komórek logicznych	Liczba CLB	Liczba slice'y	Pojemność pamięci rozproszonej	Pojemność pamięci BlockRAM	Liczba BlockRAM–ów	Maksymalna liczba linii I/O[1]	Cena układu[2]	Wykorzystane zasoby FPGA[3]	Koszt implementacji jednego procesora PicoBlaze
XC3S50	50000	1728	192	768	12288	73728	4	124	48 zł	12,5%	6 zł
XC3S200	200000	4320	480	1920	30720	221184	12	173	59 zł	5%	2,95 zł
XC3S400	400000	8064	896	3584	57344	294912	16	264	105 zł	2,67%	2,8 zł
XC3S1000	1000000	17280	1920	7680	122880	442368	24	391	–	1,25%	–

Uwagi:
1. Liczba linii I/O zależy także od typu obudowy układu FPGA.
2. Podano ceny netto układów FPGA dla sprzedaży detalicznej (za *www.kamami.pl*).
3. W rozliczeniu nie uwzględniono zajętości pamięci BlockRAM (dla jednego PicoBlaze konieczny jest jeden tak blok).



Rys. 3. Schemat blokowy ilustrujący architekturę układów Spartan 3

temów mikroprocesorowych) jest bez wątpienia ostatni z wymieniowych (rys. 2c), ale ma on poważne wady praktyczne:

– Ze względu na zastosowanie dla rozkazów 18–bitowego (w niektórych implementacjach 16–bitowego) słowa, konieczne by było zastosowanie dwóch lub trzech, połączonych równolegle, klasycznych pamięci Flash z 8–bitowymi magistralami danych.

Chcesz być nowoczesny? Masz szansę!

„Miękki" mikrokontroler PicoBlaze umożliwia samodzielne budowanie zintegrowanych systemów cyfrowych (*System–on–Chip*) w układach FPGA i CPLD. Jego implementacja nie wymaga żadnych inwestycji poza kupieniem docelowego układu PLD lub platformy sprzętowej, którą opiszemy w następnym numerze EP. Narzędzia uruchomieniowe (w tym kompilator i symulator VHDL) są dostępne bezpłatnie, tak jak i sam rdzeń.

– Dołączenie pamięci o tak szerokiej magistrali wymaga zarezerwowania dużej liczby wyprowadzeń (co najmniej 29) co powoduje, że nie zawsze będzie możliwe stosowanie układów FPGA w obudowach o mniejszej liczbie wyprowadzeń (czytaj: tańszych).
– Ze względu na synchronizację odczytu danych z pamięci programu za pomocą sygnału zegarowego konieczne może się okazać zastosowanie dodatkowych rejestrów *latch* „przytrzymujących" odczytywane dane.
– Zastosowanie zewnętrznej pamięci programu powoduje, że otoczenie FPGA niepotrzebnie się rozrasta, redukując zalety stosowania tego układu.

Niezbyt korzystne jest także implementowanie pamięci w sposób pokazany na rys. 2b, a to z tego względu, że komórki pamięci zostaną rozproszone po strukturze układu FPGA, co może dość znacznie ograniczyć (ze względu na czasy propagacji sygnałów na liniach adresowych i danych) maksymalną częstotliwość taktowania PicoBlaze'a.

Z tych powodów optymalnym wyjściem jest „wbudowanie" PicoBlaze'a (lub kilku PicoBlaze'ów) w układ FPGA wyposażony w bloki konfigurowalnej pamięci (*BlockRAM*) – przykładem takiej rodziny jest Spartan 3 (do tego są jedne z najtańszych FPGA na rynku, dostępne także w sprzedaży detalicznej). Taką właśnie implementację przedstawimy w artykule. W **tab. 1** przedstawiono najważniejsze parametry wybranych układów z rodziny Spartan 3 oraz koszty implementacji PicoBlaze'a w układach dostępnych w sprzedaży detalicznej (za *www.kamami.pl*). Na **rys. 3** pokazano uproszczony schemat blokowy ilustrujący budowę układów Spartan 3.

Warianty PicoBlaze

PicoBlaze jest dostępny w czterech wariantach, których opisy HDL zostały zoptymalizowane pod kątem architektur docelowych układów PLD. Zręczność Kena Chapmana w projektowaniu w VHDL umożliwiła zaimplementowanie PicoBlaze'a nawet w układzie CPLD (XC2C256 z rodziny CoolRunner II – **tab. 2**), co – biorąc pod uwagę złożoność projektu – wydaje się zadaniem karkołomnym. Najważniejsze różnice pomiędzy wersjami kodów źródłowych wynikają z optymalizowania budowy rdzenia mikrokontrolera do możliwości architektury układu docelowego. W dalszej prezentacji skupimy się na najbardziej rozbudowanej funkcjonalnie wersji PicoBlaze'a, przeznaczonej dla układów Spartan 3, Virtex II i Virtex II PRO.

Tab. 2. Zestawienie najważniejszych parametrów dostępnych wersji PicoBlaze'a

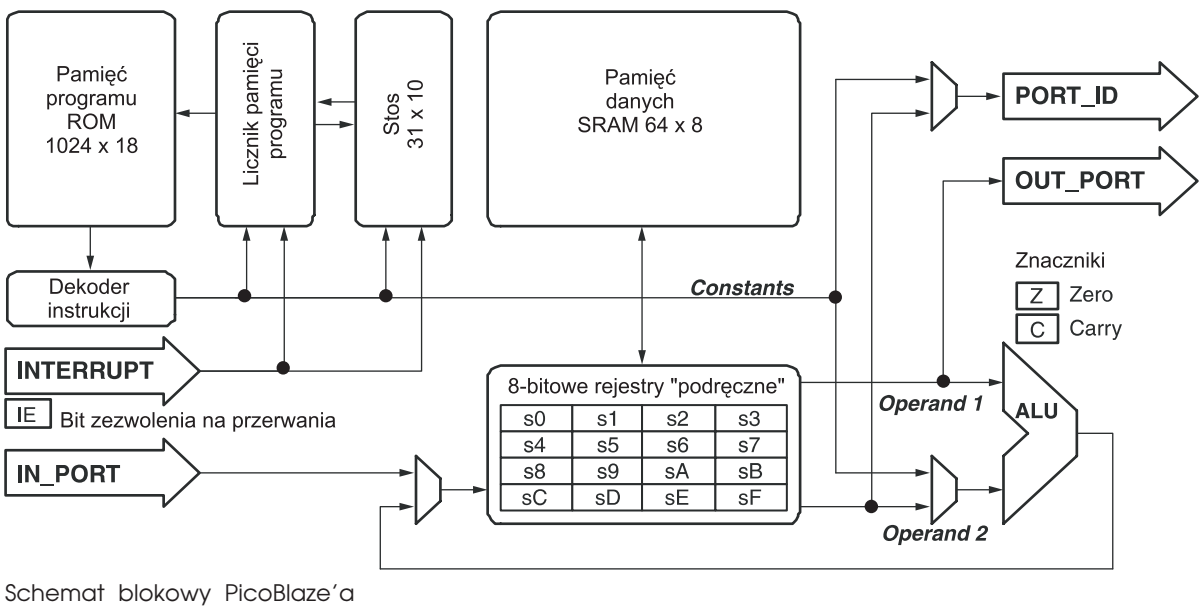
Parametr	Wersja dla układów: Spartan 3, Virtex II, Virtex II PRO	Wersja dla układów: Virtex, Virtex E, Spartan II, Spartan IIE	Wersja dla układów Virtex II, Virtex IIE	Wersja dla układów CoolRunner II
Obszar adresowania pamięci programu	1024 słowa	1024 słów	256 słów	256 słów
Szerokość słowa instrukcji	18 bitów	16 bitów	18 bitów	16 bitów
Głębokość stosu	31 słów	15 słów	31 słowa	4 słowa
Liczba komórek niezbędnych do implementacji	96 slice (Spartan 3)	76 slice (Spartan IIE)	84 slice (Virtex II)	212 makrokomórek
Prędkość wykonywania instrukcji (najszybsze wersje układów)	45 MIPS (Spartan 3)	37 MIPS (Spartan IIE)	64 MIPS (Virtex II)	21 MIPS
Liczba 8–bitowych rejestrów	16	16	32	8
Liczba linii I/O (standardowo/maksymalnie)	8/256	8/256	8/256	8/256
Podręczna pamięć SRAM	64 B	–	–	–
Liczba instrukcji	59	57	57	57
Przerwania	+	+	+	+
Pamięć programu	wbudowana	wbudowana	wbudowana	zewnętrzna

Architektura PicoBlaze'a

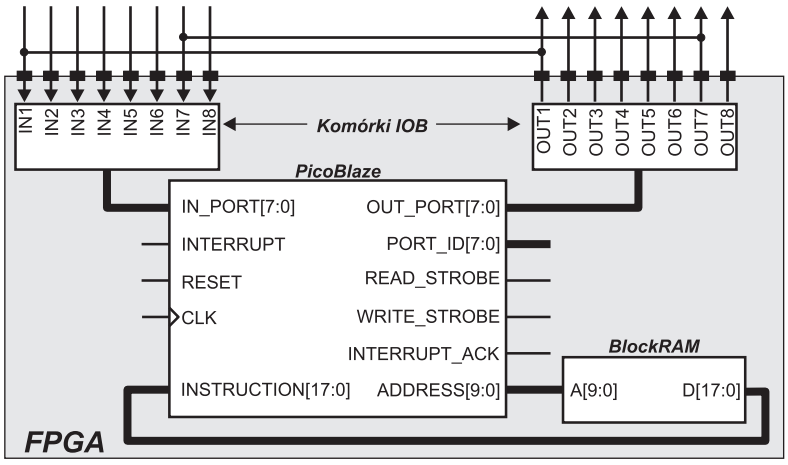
Schemat blokowy ilustrujący wewnętrzną budowę PicoBlaze'a pokazano na **rys. 4**. Jednostka arytmetyczno–logiczna ALU służy do wykonywania podstawowych operacji logicznych (AND, OR, ExOR) i arytmetycznych (dodawanie, odejmowanie, przesuwanie (mnożenie) i rotacja argumentów), przy czym argumenty jak i wynik operacji mogą być lokowane w dowolnym rejestrze z grupy 16 rejestrów „podręcznych" (s0…sF). PicoBlaze nie ma więc wyróżnionego rejestru (akumulatora), w którym są lokowane wyniki operacji. ALU może, na drodze sprzętowej, wykonywać operacje przesunięcia i rotacji wskazanego argumentu, odpowiada także za ustawianie flag (znaczników) *Zero* i *Carry*, które informują o szczególnych cechach wyniku wykonanej operacji. Znacznik *Carry* może być wykorzystany także jako wskaźnik parzystej liczby jedynek podczas wykonywania instrukcji TEST.

W prezentowanej wersji PicoBlaze jest wyposażony w 64 B podręcznej pamięci SRAM, którą można wykorzystać do przechowywania zmiennych na których operuje program. Dostęp do tej pamięci umożliwiają dwa rozkazy: STORE oraz FETCH. Wydzielonym obszarem pamięci jest 31–poziomowy stos, który służy do magazynowania adresów powrotu

Komunikacja mikrokontrolera z otoczeniem jest możliwa dzięki liniom I/O. PicoBlaze jest wyposażony w 8 linii wejściowych i 8 wyjściowych, ale dzięki zintegrowanemu dekoderowi portów I/O (z wyjściami PORT_ID[7:0] i sygnałami sterującymi READ_STROBE i WRITE_STROBE) mikrokontroler można wyposażyć łącznie w 256 8–bitowych portów wejściowych i wyjściowych. Rozdzielenie linii wejściowych od wyjściowych ułatwia implementację mikrokontrolera w układach PLD – konfigurowalne komórki I/O mają bowiem w różnych układach różną budowę i nie zawsze zaproponowany przez autora opis HDL będzie możliwy do zaimplementowania w fizycznej strukturze układu. Jeżeli konieczne jest łączenie linii wejściowych i wyjściowych, można to najprościej wykonać w sposób pokazany na **rys. 5** (dla linii IN/OUT1 oraz IN/OUT7). W zależności od możliwości układu FPGA, w którym jest implementowany mikrokontroler, linia wyjściowa może być skonfigurowana jako trójstanowa, z wyjściem typu otwarty dren lub pracować w konfiguracji *push–pull*. Także konfiguracja linii wejściowych jest zależna od możliwości układu FPGA. Wejścia mogą „pły-



Rys. 4. Schemat blokowy PicoBlaze'a



Rys. 5. Sposób utworzenia dwukierunkowych linii I/O

wać" bez wstępnej polaryzacji, mogą być podciągnięte do plusa (*pull–up*) lub masy (*pull–down*) zasilania, mogą być także zabezpieczone przed negatywnymi skutkami niepodłączenia za pomocą aktywnego bloku polaryzującego *bus–hold* (który podtrzymuje na linii wejściowej ostatni ustalony poziom logiczny).

Jednostka centralna PicoBlaze'a obsługuje jedno maskowane przerwanie. Można je zgłosić zmieniając stan wejścia INTERRUPT z „0" na „1", natomiast przyjęcie obsługi przerwania jest sygnalizowane za pomocą „1" na wyjściu INTERRUPT_ACK. Zgłoszenie przerwania wymusza wykonanie przez CPU skoku do ostatniej komórki pamięci (CALL 3FF), gdzie programista musi umieścić wektor jego obsługi. Podczas aktywnej obsługi przerwania znaczniki *Carry* i *Zero* są zastępowane drugą parą znaczników (wykorzystywaną tylko podczas obsługi przerwania), automatycznie blokowana jest także możliwość przyjęcia kolejnego przerwania. Obsługa przerwań następuje niemalże w czasie rzeczywistym – po maksymalnie 5 taktach zegarowych.

Podobnie jak inne mikroprocesory, także PicoBlaze wymaga wstępnego wyzerowania. Inicjalizację CPU umożliwia wejście synchronicznego (z sygnałem zegarowym CLK) zerowania RESET.

Jedną z bardzo istotnych cech PicoBlaze'a jest wykonywanie wszystkich rozkazów w dwóch taktach zegarowych, niezależnie od tego jaki rozkaz jest wykonywany. Dzięki temu oszacowanie szybkości wykonywania programu jest łatwe, co pozwala w wygodny sposób projektować pętle do programowego odliczania czasu.

Tab. 3. Wyprowadzenia PicoBlaze'a i ich funkcje

Nazwa wyprowadzenia	Kierunek	Opis
CLK	I	Wejście sygnału zegarowego, synchronizującego wszystkie procesy
RESET	I	Wejście synchronicznego (z CLK) sygnału zerującego
ADDRESS	O	10–bitowa magistrala adresowa (adresuje pamięć programu)
INSTRUCTION	I	18–bitowa magistrala danych pobieranych z pamięci programu
IN_PORT	I	8–bitowy port wejściowy
OUT_PORT	O	8–bitowy port wyjściowy
READ_STROBE	O	Sygnał strobujący odczyt z dodatkowego portu wejściowego
WRITE_STROBE	O	Sygnał strobujący zapis do dodatkowego portu wyjściowego
PORT_ID	O	8–bitowy wskaźnik adresujący aktywny port wejściowy lub wyjściowy
INTERRUPT	I	Wejście sygnału przerwania
INTERRUPT_ACK	O	Wyjście sygnału przyjęcia przerwania

Funkcje wyprowadzeń

Wyprowadzenia komponentu KCPSM3 (czyli PicoBlaze) zostały zadeklarowane w sposób pokazany na **list. 1**. Jest to kompletny opis interfejsu mikrokontrolera, który jest widziany przez otoczenie, zgodnie z symbolem graficznym pokazanym na rys. 1. Opis funkcji poszczególnych wyprowadzeń pokazano w **tab. 3**. Nie podano oczywiście numerów przypisanych im wyprowadzeń, ponieważ użytkownik może je rozmieścić w praktycznie dowolny sposób, a w niektórych przypadkach (gdy całe urządzenie „mieści się" w FPGA) mogą w ogóle nie być wyprowadzane na zewnątrz układu. Warto zwrócić uwagę na fakt, że w przypadku implementacji PicoBlaze'a w układzie z rodziny Spartan 3 nie ma konieczności wyprowadzenia na zewnątrz układu magistral: danych i adresowej, dołączana jest do nich bowiem wewnętrzna pamięć ROM (emulowana w module BlockRAM), której deklarację interfejsu przedstawiono na **list. 2**.

Lista rokazów

Lista rozkazów obsługiwanych przez PicoBlaze (w wersji KCPSM3) składa się z 59 poleceń. Można je podzielić na 7 grup przedstawionych poniżej. Mnemoniki są wystarczająco czytelne, żeby zrozumieć funkcje poleceń, a argumenty poleceń są następujące:

aaa – oznacza adres z zakresu 0x000…0x3FF

kk – oznacza stałą o wartości z zakresu 0x00…0xFF

ss – adres pamięci SRAM z zakresu 0x00…0x3F

pp – adres portu I/O z zakresu 0x00…0xFF

sX, sY – adresy rejestrów „podręcz-

List. 1. Deklaracja interfejsu komponentu KCPSM3 (PicoBlaze) w języku VHDL

```
component kcpsm3
Port (
address : out std_logic_vector(9 downto 0);
instruction : in std_logic_vector(17 downto
0);
port_id : out std_logic_vector(7 downto 0);
write_strobe : out std_logic;
out_port : out std_logic_vector(7 downto
0);
read_strobe : out std_logic;
in_port : in std_logic_vector(7 downto 0);
interrupt : in std_logic;
interrupt_ack : out std_logic;
reset : in std_logic;
clk : in std_logic);
end component;
```

Dwutaktowiec

Rdzeń PicoBlaze'a wykonuje wszystkie instrukcje w dwóch taktach zegara.

List. 2. Deklaracja interfejsu komponentu PROG_ROM, czyli pamięci programu procesora PicoBlaze

```
component prog_rom
Port (
address : in std_logic_vector(9 downto 0);
instruction : out std_logic_vector(17
downto 0);
clk : in std_logic);
end component;
```

nych" z zakresu 0x0…0xF

Programista stosujący PicoBlaze'a ma do dyspozycji następujące rozkazy:

1. Polecenia sterujące przebiegiem programu.

Rozkazy skoków warunkowych i bezwarunkowych

JUMP aaa
JUMP Z,aaa
JUMP NZ,aaa
JUMP C,aaa
JUMP NC,aaa

Rozkazy warunkowego i bezwarunkowego wywołania podprogramów

CALL aaa
CALL Z,aaa
CALL NZ,aaa
CALL C,aaa
CALL NC,aaa

Rozkazy warunkowego i bezwarunkowego powrotu z podprogramów

RETURN
RETURN Z
RETURN NZ
RETURN C
RETURN NC

2. Operacje arytmetyczne

ADD sX,kk
ADDCY sX,kk
SUB sX,kk
SUBCY sX,kk
COMPARE sX,kk
ADD sX,sY
ADDCY sX,sY
SUB sX,sY
SUBCY sX,sY
COMPARE sX,sY

3. Operacje logiczne

LOAD sX,kk
AND sX,kk
OR sX,kk
XOR sX,kk
TEST sX,kk
LOAD sX,sY
AND sX,sY
OR sX,sY
XOR sX,sY
TEST sX,sY

4. Operacje przesunięć i rotacji

SR0 sX
SR1 sX
SRX sX
SRA sX
RR sX
SL0 sX
SL1 sX
SLX sX
SLA sX
RL sX

5. Operacje na portach wejścia i wyjścia

INPUT sX,pp
INPUT sX,(sY)
OUTPUT sX,pp
OUTPUT sX,(sY)

6. Operacje dostępu do pamięci SRAM

STORE sX,ss
STORE sX,(sY)
FETCH sX,ss
FETCH sX,(sY)

7. Polecenia związane z obsługą przerwań

RETURNI ENABLE
RETURNI DISABLE
ENABLE INTERRUPT
DISABLE INTERRUPT

Szczegółowy opis wymienionych poleceń opublikujemy w kolejnych numerach EP.

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