

# Wprowadzenie

PicoBlaze udostępniany przez firmę Xilinx jest procesorem, którego opis w językach HDL (ang. *Hardware Description Language* – język opisu sprzętu) opracował Ken Chapman, inżynier tej firmy. Jest to bardzo prosty procesor 8-bitowy i właśnie dzięki temu, że opracowano go do układów programowalnych, jest idealnym układem w wielu zastosowaniach o dużej przewodze funkcjonalnej nad innymi mikrokontrolerami. PicoBlaze jest zbudowany tak, aby jak najlepiej korzystał z wewnętrznych bloków układów programowalnych, produkowanych przez Xilinksa. Na analogicznych zasadach są oferowane procesory 32-bitowe microBlaze.

Należy zauważyć, że podobne procesory mają w swoich ofertach inne firmy produkujące układy programowalne. Lattice oferuje własny procesor 8-bitowy Mico8 oraz wersję 32-bitową Mico32. Z kolei Altera ma w ofercie procesory Nios (16-bitowy) i Nios II (32-bitowy).

Dużą zaletą wspomnianych rozwiązań jest to, że są one udostępniane za darmo. Do tej pory większość rozwiązań urządzeń opisywanych językami HDL, znanych jako IP core (ang. *Intellectual Property core* – rdzeń będący własnością intelektualną), jest w przeważającej części oferowana za bardzo wygórowane honoraria.

Dla osób, które mało interesowały się układami programowalnymi, z pewnością nowym pojęciem jest właśnie IP core. W mikrokontrolerach nieodłączną częścią jest oprogramowanie, czyli opis kolejnych instrukcji, które musi wykonać procesor. FPGA/CPLD podobnie potrzebuje opisu, w jaki sposób urządzenie ma działać lub jak ma być zbudowane. IP core jest właśnie takim modułem dla układu programowalnego, jakim dla oprogramowania może być biblioteka lub zestaw podprogramów. Dostępne są IP core'y w postaci źródłowej, do użycia na podobnej zasadzie jak funkcjonuje open source. Ale również IP core można kupić od firmy, zajmującej się ich tworzeniem.

Warto tu zaznaczyć dodatkową różnicę pomiędzy programem dla procesora a fragmentem opisującym działanie sprzętu. Program wykonywany jest z reguły sekwencyjnie, sprzęt funkcjonuje zaś zazwyczaj równolegle – współbieżnie.

W układach programowalnych bardzo często wbudowane są układy sekwencyjne (maszyny stanów). Z takich bloków między innymi jest zbudowany właśnie procesor, który wykonuje program. PicoBlaze jest takim układem – maszyną stanów. Mikroprocesor ten jest znany również pod nazwą KCPSM (*Constant(K) Coded Programmable State Machine* lub *Ken Chapman's PSM*), co w wolnym tłumaczeniu znaczy „programowana maszyna stanów zakodowana na stałe” lub „programowana maszyna stanów Kena Chapmana”. Ponieważ jest to bardzo małutki mikroprocesor, właściwszą nazwą wydaje się być „miękki procesor” (*soft processor*) – procesor łatwy do implementacji w układach programowalnych.

Podstawową przewagą mikrokontrolerów implementowanych bezpośrednio w strukturach FPGA/CPLD nad innymi sprzętowymi realizacjami jest możliwość ich dowolnego kształtowania i dokonywania zmian bloków funkcjonalnych zgodnie z potrzebami.

Porównajmy możliwości standardowych mikrokontrolerów z tym, co oferuje procesor w FPGA – **tabela** poniżej. Porównujemy tylko rozwiązania 8-bitowe, chociaż w przypadku FPGA istnieją projekty mikrokontrolerów 32-bitowych i są one z powodzeniem stosowane.

Zestawienie cech wybranych mikrokontrolerów

Opis,Mikrokontroler	8051	AVR ATTiny13	ST6215	PicoBlaze
Pamięć RAM	256 B	64 B	64 B	64 B
Rejestry	32	32	5	16
Szerokość rejestru	8 bitów	8 bitów	8 bitów	8 bitów
Pamięć programu (liczba słów)	4k	1k	2k	1 k
Szerokość instrukcji	8 bitów	8 bitów	8 bitów	18 bitów
Liczba taktów zegara potrzebnych na wykonanie jednej instrukcji	1	1	1	2
Licznik/timer	2	1	2	*
Układ przerwań	jest	jest	jest	jedno przerwanie *
Porty zewnętrzne	4	6 IO	20 IO	do 256
Oscylator wewnętrzny	jest	jest	pomocniczy	zależnie od użytego układu programowalnego
Kontroler I <sup>2</sup> C	–	–	–	*
Kontroler UART	jest	–	–	*
Kontroler SPI	–	jest	–	*

\* – w zależności od implementacji HDL dana część mikrokontrolera może być dowolnie rozbudowana

Powyższe zestawienie nie jest pełne, gdyż na rynku dostępnych jest wiele mikrokontrolerów z pamięciami RAM i Flash o różnej pojemności, liczbie linii IO i przeróżnymi układami peryferyjnymi. Ilustruje jednak dwie najważniejsze rzeczy:

- 1) Mikrokontroler PicoBlaze jest funkcjonalnie takim samym mikrokontrolerem jak większość 8-bitowych układów.
- 2) W układzie programowalnym możemy go skonfigurować z dowolną liczbą peryferii, ograniczoną zasobami układu FPGA/CPLD.

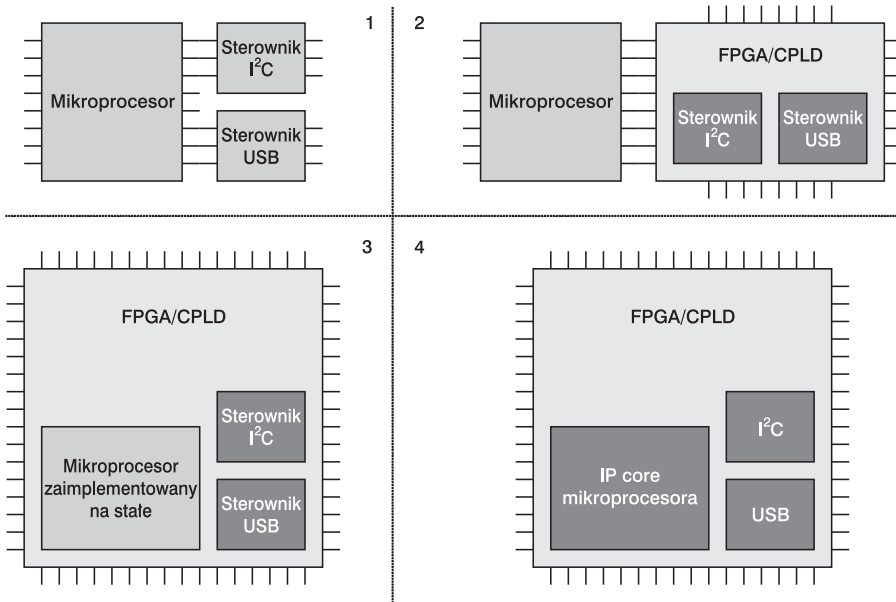
### Konfiguracje mikroprocesorowe

Z punktu widzenia współdziałania mikroprocesorów i układów programowalnych można rozróżnić cztery główne konfiguracje mikroprocesorowe (**rysunek** z następnej strony).

#### 1. Mikroprocesor.

Połączenie mikroprocesora z układami peryferyjnymi w jeden układ scalony lub też jako kilka układów. Jest to jedna z najpopularniejszych obecnie konfiguracji.

#### 2. Mikroprocesor/mikrokontroler plus układ programowalny.



Sposób implementowania kompletnego mikrokontrolera w układzie FPGA

Obecnie najpopularniejsze zastosowanie układów FPGA/CPLD. Rozwiązanie sprawdza się znakomicie przy rozszerzaniu dobrych systemów procesorowych o kolejne możliwości, jakie dają układy programowalne.

3. Układy programowalne z wbudowanym procesorem (lub też odwrotnie: mikrokontroler z wbudowaną częścią FPGA w jednym układzie scalonym).

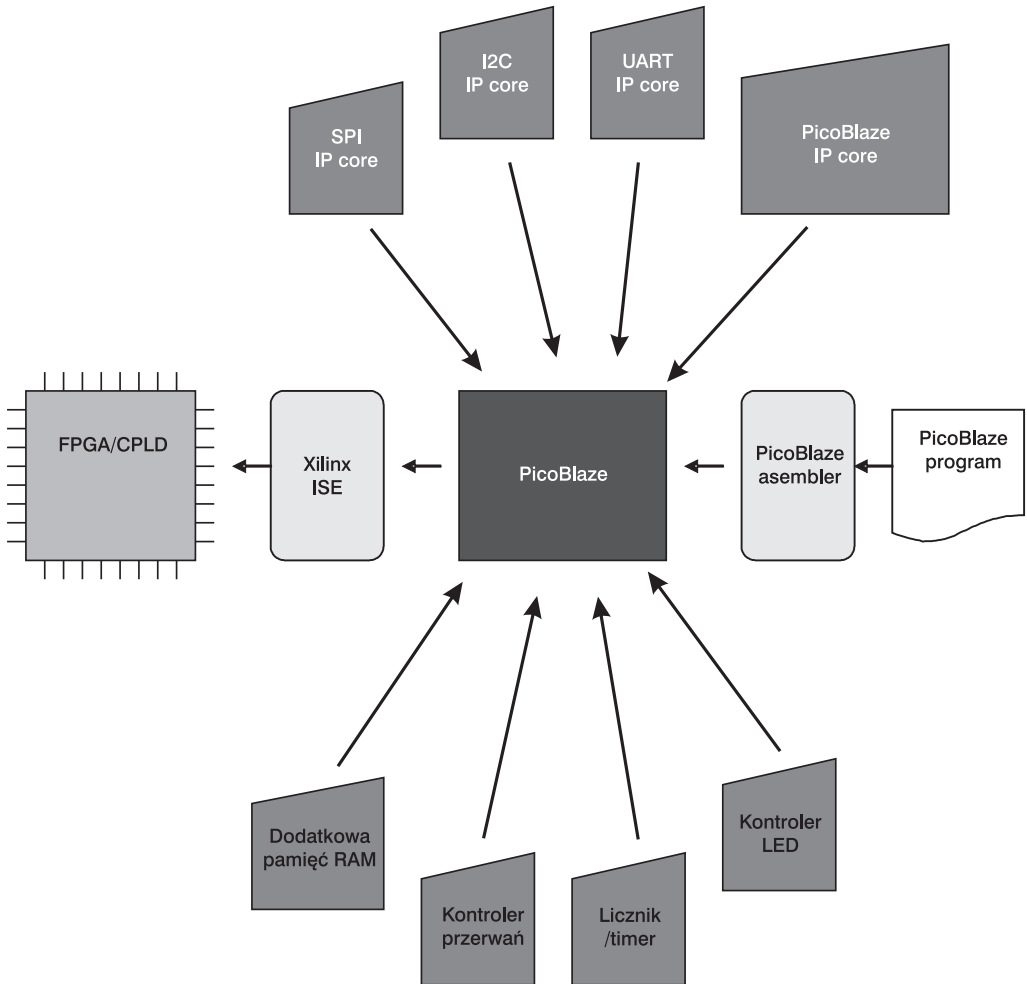
Konfiguracja ta już jakiś czas temu była wprowadzona przez większość producentów układów programowalnych. Na przykład Xilinx wprowadził rodzinę układów FPGA Virtex II z wbudowanym co najmniej jednym procesorem PowerPC, Altera zaś rodzinę układów Excalibur, z wbudowanym procesorem ARM922T. Jednakże (prawdopodobnie ze względu na cenę takich układów) nie przyjęły się one na rynku.

4. Układy programowalne z zaimplementowanym mikroprocesorem.

Ta konfiguracja jest już dosyć długo używana. Układy FPGA/CPLD są wykorzystywane przez firmy produkujące zwykłe układy scalone do projektowania i weryfikacji swoich nowych projektów. Jednakże takie zastosowanie układów programowalnych było do tej pory bardzo kosztowne. Wraz ze wzrostem wydajności i pojemności tańszych układów ta konfiguracja zaczyna być coraz bardziej konkurencyjna dla obecnie produkowanych mikrokontrolerów. I właśnie ta konfiguracja wydaje się być najbardziej optymalną do zastosowań PicoBlaze.

W książce omówiono IP core mikroprocesora PicoBlaze, a ściślej jego trzecią wersję o nazwie KCPSM3. Opis peryferii do tego rdzenia można znaleźć w Internecie bądź przygotować samemu. W ten sposób, po połączeniu w jedną całość, otrzymujemy mikroprocesor szyty na miarę. Oprogramowanie dla niego można przygotować, używając asemblera. Kod źródłowy jest wkompilowywany w pamięć ROM,

zintegrowaną z mikroprocesorem. Po wgraniu kodu wynikowego do układu FPGA/CPLD otrzymuje się zwyczajny procesor – **rysunek** poniżej.



Jak zrobić mikrokontroler

W książce są zaprezentowane przykłady różnych peryferii i konfiguracji mikroprocesora, które pozwolą na zrealizowanie podstawowych projektów z PicoBlaze. Większość przykładów opisano językiem VHDL. Jednakże, w najistotniejszych miejscach, przedstawiono sposoby wykonania implementacji również w języku Verilog.

### PicoBlaze

Jak wcześniej wspomniano, PicoBlaze jest dostępny w postaci IP core. Ze strony internetowej firmy Xilinx można ściągnąć pakiet z plikami źródłowymi mikroproce-

sora, opisanego w języku VHDL lub Verilog oraz notę aplikacyjną, asembler i opisy niektórych bloków funkcjonalnych.

W Internecie jest dostępna również dokumentacja o nazwie *pacoBlaze*. Jest to funkcjonalnie pełny odpowiednik *PicoBlaze*, ale opisany tylko w Verilogu. *PacoBlaze* jest udostępniony na licencji BSD<sup>1</sup>.

Nie będziemy dokonywać żadnych poważniejszych zmian w mikroprocesorze *PicoBlaze* – będzie on traktowany jak czarna skrzynka. Dlatego podczas korzystania z książki można używać zarówno *PicoBlaze*, jak i *pacoBlaze*. Ważną zaletą *pacoBlaze* jest to, że można go stosować w dowolnym układzie programowalnym. Używając *PicoBlaze*, musimy ograniczyć się jedynie do produktów Xilinx – to ograniczenie występuje również w licencji tego mikroprocesora.

### Narzędzia

Udostępniony przez firmę Xilinx asembler *KCPSM3* do *PicoBlaze* realizuje dodatkowo jedną bardzo ważną funkcję. Dokonuje on wpisu kodu maszynowego bezpośrednio do elementu pamięci ROM, opisaney w języku VHDL albo Verilog. Tak opisane elementy są gotowe do dołączenia do mikrokontrolera.

Pewną wadą *PicoBlaze*, która miejmy nadzieję szybko zostanie usunięta, jest brak sprawnego kompilatora języka C lub innego języka wyższego poziomu. Na szczęście jego instrukcje są na tyle proste, że przygotowanie oprogramowania w asemblerze nie stanowi żadnego problemu, nawet dla niedoświadczonych programistów.

Do sprawdzenia oprogramowania można używać symulatora *pBlazeIDE* dla Windowsa lub *kpicosim* dla platformy Linuksa. Oba programy są dostępne bez żadnej licencji i gwarancji. Mimo że korzystają trochę z innej składni asemblera, można je wykorzystywać zamiennie. W kolejnych rozdziałach pokazano różne możliwości wykorzystania symulatora *pBlazeIDE*.

Do symulacji funkcjonalnej mikrokontrolera opisanego w językach HDL należy użyć jednego z programów do symulacji, m.in. *Avctive-HDL* firmy Aldec lub *ModelSim* firmy Mentor Graphics. Xilinx oferuje również darmową wersję symulatora *ISE Simulator Lite*, która jest dostępna razem z oprogramowaniem projektowym: *WebPack* firmy Xilinx.

Wszystkie przykłady prezentowane w książce sprawdzono na płycie ewaluacyjnej *ZL9PLD + ZL10PLD* z firmy BCM. Płytkę zawiera układ FPGA *Spartan3* firmy Xilinx. Zasoby logiczne układu *XC3S200* umożliwiają zaimplementowanie 12 pełnych mikroprocesorów *PicoBlaze*, przy czym pozostanie jeszcze dużo miejsca na dodatkowe układy peryferyjne. Dodatkowymi elementami płytki ewaluacyjnej są m.in.: łącze szeregowe *RS232*, 8 diod *LED*, 4 przyciski i wyświetlacz  $2 \times 16$  znaków.

---

<sup>1</sup> Licencja typu BSD zezwala na modyfikowanie kodu i jego rozpowszechnianie w dowolnej postaci. Produkty tej licencji można włączyć do zamkniętego oprogramowania pod warunkiem załączenia do produktu informacji o autorach oryginalnego kodu i treści licencji.

## Układ książki

### *Pierwszy program*

W pierwszym rozdziale zajmujemy się uruchomieniem prostej aplikacji z wykorzystaniem PicoBlaze. Najpierw przedstawiono sam mikroprocesor jako jednostkę sprzętową w postaci czarnej skrzynki, z objaśnieniem podstawowych bloków funkcjonalnych. Następnie opisano wejścia i wyjścia mikroprocesora. Dowiemy się, jak ściągnąć PicoBlaze ze strony producenta oraz omówimy pliki, które dostarczane są w pakiecie. Zdefiniujemy pierwszy projekt i sposób jego działania na płycie ewaluacyjnej. Przygotujemy pierwszy program w assemblerze, który będzie realizował proste zadanie migotania diodą LED. Jednakże szczegóły dotyczące zastosowanych instrukcji assemblerowych omówiono w rozdziale 2. Opiszemy program `kcpsm3.exe`, który na podstawie przygotowanego kodu w assemblerze wygeneruje moduł pamięci ROM zawierający kod wynikowy programu assemblerowego dla PicoBlaze. Następnie skompilujemy opis projektu w ISE WebPACK i wpiszemy plik wynikowy do układu FPGA na płycie ewaluacyjnej.

### *Instrukcje mikroprocesora*

W tym rozdziale zajmujemy się assemblerem PicoBlaze oraz jego budową arytmetyczno-logiczną. Jak na mikroprocesor całość nie jest duża (16 rejestrów, arytmetyka 8-bitowa i 64 bajty pamięci), ale za to niezwykle prosta w obsłudze. Przybliżamy sposób realizacji podstawowych instrukcji oraz obsługiwanie podprogramów. Następnie pokazujemy, jak realizować operacje arytmetyczne na więcej niż ośmiu bitach oraz jak wykonać mnożenie i dzielenie. Dzięki tym informacjom będziemy mogli przygotować licznik operujący na liczbach wielobitowych i przerobimy trochę projekt z pierwszego rozdziału. Na koniec przedstawimy symulator pBlaze IDE i za jego pomocą prześledzimy działanie zmodyfikowanego programu.

### *Porty wejścia i wyjścia*

Do komunikacji PicoBlaze z peryferiami zewnętrznymi służą porty wejściowy i wyjściowy. W tym rozdziale omówimy instrukcje assemblera PicoBlaze służące do wykonywania operacji na portach. Przedstawimy opisy w językach HDL pozwalające najefektywniej korzystać z wielu portów. Następnie zaprezentujemy przykład projektu obsługującego za pomocą portów diody LED oraz przyciski. Na koniec omówiono testbench ułatwiający sprawdzenie symulacyjne przygotowanych projektów.

### *Podręczna pamięć RAM*

W tym rozdziale omówimy wykorzystanie podręcznej pamięci RAM przez PicoBlaze. Przedstawimy instrukcje assemblera, które ją obsługują. Jako przykład wykorzystania pamięci pokażemy, jak podłączyć do PicoBlaze peryferyjny UART i jak go obsłużyć z poziomu assemblera. Ponadto przedstawimy prosty program do wydawania komend mikroprocesorowi z komputerem PC poprzez UART. Na koniec zasugerujemy, jak można efektywnie testować układ z wykorzystaniem testbenchu.

### *Przerwania*

W tym rozdziale omówimy sposób obsługi przerwania. Najpierw opiszemy instrukcje asemblera PicoBlaze niezbędne do realizacji obsługi przerwania. Następnie podłączymy do PicoBlaze moduł zegarowy (licznik), który będzie mógł zgłaszać zdarzenia w postaci alarmu (przerwania). Na koniec pokażemy, jak zrealizować sprzętowo-programową obsługę wielu przerwań.

### *PicoBlaze i JTAG*

Standard JTAG opracowano do testowania układów cyfrowych. W układach FPGA jest wykorzystywany m.in. do ich konfigurowania. Dla układów z rodziny Spartan i innych jest dostępny parametryzowany moduł biblioteczny BSCAN, który umożliwia realizowanie interfejsu JTAG do własnych zastosowań. Pokażemy także, jak wykorzystać ten interfejs w PicoBlaze oraz w jaki sposób możemy dzięki niemu podmienić kod programu bez rekonfigurowania całego układu programowalnego. Dodatkowo zaprezentujemy przykładową implementację własnego portu JTAG oraz omówimy dwa formaty plików wykorzystywanych przy używaniu tego portu: BSDL i SVF.

### *Urządzenia peryferyjne*

Poza samym mikroprocesorem w systemie mikrokontrolerowym występują dodatkowe układy peryferyjne. We wcześniejszych rozdziałach książki omówiono już kilka z nich, np. interfejsy UART i JTAG oraz moduły wewnętrzne do obsługi diod LED, przycisków czy licznika. W tym rozdziale zajmiemy się kilkoma praktycznymi aspektami współdziałania PicoBlaze z zewnętrznym światem techniki cyfrowej.

### *Magistrala systemowa*

W poprzednich rozdziałach opisywaliśmy przyłączanie układów peryferyjnych do PicoBlaze za pomocą portów wejścia-wyjścia. Wśród otwartych projektów IP core dostępnych w Internecie można znaleźć wiele opisów systemów, w których do połączenia peryferii z procesorem wykorzystuje się magistralę. W projektach (dostępnych m.in. na [www.opensource.org](http://www.opensource.org)) najczęściej korzysta się z dostępnej darmowo specyfikacji magistrali o nazwie Wishbone. Pokażemy więc, jak za pomocą magistrali Wishbone przyłączyć do PicoBlaze gotowy IP core I<sup>2</sup>C master i wykorzystać go do sterowania zewnętrznym układem zegarowym.

### *Rozszerzenia*

W tym rozdziale zajmiemy się możliwościami rozbudowania PicoBlaze oraz optymalizacją jego syntezy. Pierwszy przykład będzie ilustrował zwiększenie możliwości PicoBlaze poprzez zwiększenie rozmiaru dostępnej pamięci podręcznej RAM. W drugim przykładzie zaprezentujemy dodanie układu peryferyjnego obsługującego semafor, dzięki któremu można zrealizować połączenie dwóch PicoBlaze tak, aby mogły obsługiwać bezpiecznie współdzielone zasoby. Następnie pokazane zostaną różnice w optymalizacji syntezy PicoBlaze oraz wyniki symulacji czasowej po jego implementacji.

*Dodatek A*

Opis instrukcji mikroprocesora PicoBlaze.

*Dodatek B*

Zestawienie wyników implementacji projektów omawianych w książce.