Zrób sobie procesor: PicoBlaze w FPGA, część 1

Zróbmy sobie procesor! Brzmi banalnie, strasznie, niewiarygodnie? A jednak, dlaczego nie? Ci co dzisiaj tworzą dowolne aplikacje na AVR'ach, 51'ach i tym podobnych mikrokontrolerach zapytają się pewnie jak to? Ano tak po prostu! Czy możesz dzisiaj zaprojektować dowolną prostą aplikację mikroprocesorową? No pewnie!

Na rynku jest dostępnych wiele gotowych, uniwersalnych mikroprocesorów 8–, 16– i 32–bitowych, ale niewielu inżynierów potrafi (a często nawet nie zna takiej możliwości) wykonać własny mikroprocesor, szyty na miarę.

W kolejnych artykułach pokażemy jak jest to proste.

W artykule zajmiemy się 8–bitowym procesorem PicoBlaze, udostępnionym bezpłatnie przez firmę Xilinx. Został on opracowany przez Kena Chapmana, inżyniera tej firmy. Dostępne są jego implementacje na różne rodziny PLD firmy Xilinx, my zajmiemy się jego wersją na układy Spartan 3.

Warto na wstępie zauważyć, że podobne rozwiązania mają inne firmy produkujące układy PLD – na przykład Lattice oferuje swój własny procesor 8–bitowy o nazwie Mico8 oraz wersję 32–bitową o nazwie Mico32. Z kolei firma Altera ma w swojej ofercie procesory Nios (16–bitowy) i Nios II (32–bitowy).

Większość obecnie dostępnych systemów operacyjnych to systemy wielowątkowe, tzn. symulują one na jednym procesorze jednoczesne wykonywanie wielu wątków oprogramowania. Z drugiej strony w układach reprogramowalnych bardzo często wykorzystuje się maszyny stanów. Najkrócej mówiąc jest to sprzęt, który opisano w taki sposób, by wykonywał zadane operacje w sposób sekwencyjny. Taką maszyną stanów jest właśnie procesor, na którym wykonywany jest program. W ten sposób zamknięte zostało koło sprzęt–oprogramowanie. Jak widać są to dwa nierozerwalne, wzajemnie się uzupełniające elementy. PicoBlaze jest właśnie taką maszyną stanów. Mikroprocesor ten, funkcjonuje również pod nazwą KCPSM (*Constant(**K**) **C**oded **P**rogrammable **S**tate **M**achine*, lub *„**K**en **C**hapman's **PSM**"*), co w wolnym tłumaczeniu można przetłumaczyć jako „programowana maszyna stanów zakodowana na stałe" lub „programowana maszyna stanów Kena Chapmana".

Ponieważ PicoBlaze jest „malutkim" mikroprocesorem, właściwszą nazwą wydaje się być programowa maszyna stanów. I w zasadzie taką nazwę w literaturze anglojęzycznej można częściej spotkać. W artykułach pokażemy, że pomimo swojej prostoty PicoBlaze w pełni zasługuje na miano mikroprocesora.

Jak już wcześniej wspomniano podstawową przewagą mikrokontrolerów implementowanych bezpośrednio w strukturach FGPA/CPLD jest możliwość ich dowolnego kształtowania i dokonywania zmian sprzętowych w gotowym produkcie. Jest to kolejny krok w zmianach jakie zapoczątkowało stosowanie popularnej '51 w latach 90–tych ubiegłego wieku. Już nie tylko możemy użyć procesora w dowolnych aplikacjach, ale i sami możemy go zaprojektować i przygotować taki mikrokontroler, jaki w danej aplikacji będzie najbardziej odpowiedni, a gdy warunki pracy aplikacji się zmienią, nie trzeba zmieniać całego rozwiązania, tylko je na nowo dostosować, jak zwykły program.

Porównajmy możliwości standardowych mikrokontrolerów z tym,

Tab. 1. Zestawienie cech wybranych mikrokontrolerów 8–bitowych

Cecha\Mikrokontroler	8051	AVR Atiny13	ST6215	PicoBlaze
Pamięć RAM	256B	64B	64B	64B
Rejestry	32	32	5	16
Szerokość rejestru	8 bitów	8 bitów	8 bitów	8 bitów
Pamięć programu (liczba słów)	4 k	1k	2k	1k
Szerokość instrukcji	8 bitów	8 bitów	8 bitów	18 bitów
Liczba taktów zegara potrzebnych do wykonania 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
Wewnętrzny generator taktujący	jest	jest	pomocniczy	zależnie od użytego układu reprogramowalnego
Interfejs I^2C	–	–	–	*
Interfejs UART	jest	–	–	*
Interfejs SPI	–	jest	–	*

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

co daje FPGA – patrz zestawienie w **tab. 1**.

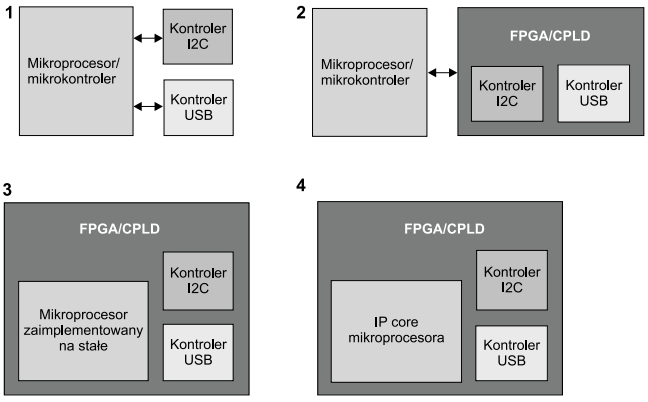
Zestawienie nie jest oczywiście „idealne”, gdyż na rynku dostępnych jest wiele wariantów różnych mikrokontrolerów, z różną ilością dostępnej pamięci RAM, pamięci programu, linii IO i przeróżnymi układami peryferyjnymi. Pokazuje ono jednak dwie najważniejsze rzeczy:

1 Mikrokontroler PicoBlaze jest funkcjonalnie takim samym mikrokontrolorem jak większość 8–bitowych układów dostępnych od wielu lat na rynku i w niczym im nie ustępuje.
2 W układzie reprogramowalnym możemy go skonfigurować z dowolną liczbą peryferii, będąc ograniczonym jedynie tym, co dostarcza producent FPGA/CPLD.

Konfiguracje systemów z mikroprocesorami wbudowanymi

Historycznie patrząc układy mikroprocesorowe (mikrokontrolery) i układy reprogramowalne rozpoczęły swój lawinowy rozwój w ubiegłej dekadzie. Układy FPGA/CPLD były traktowane – ze względu na stosunkowo niewielkie zasoby logiczne – jako rozszerzenia możliwości oferowanych przez dostępne mikrokontrolery i mikroprocesory.

Z punktu widzenia zastosowania układów reprogramowalnych, można umownie wydzielić cztery główne konfiguracje systemów mikroprocesorowych (**rys. 1**).

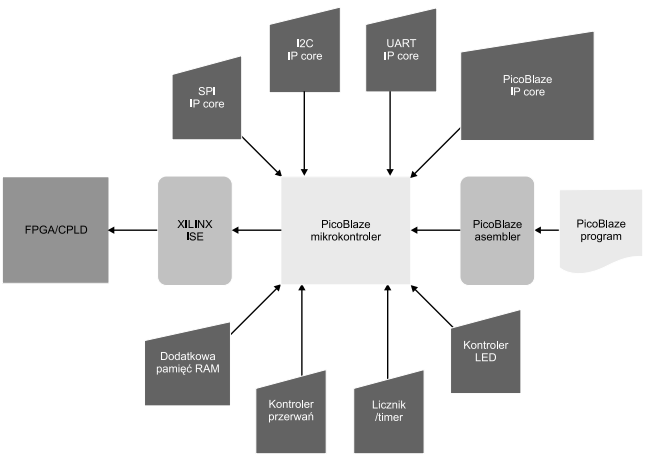


Rys 1. Przykładowe konfiguracje systemów mikroprocesorowych, w których wykorzystano układy PLD

Przepis na „procka”

Aby przygotować własny mikrokontroler potrzebny jest rdzeń i współpracujące z nim peryferia. W ten sposób, po połączeniu rdzenia z peryferiami otrzymujemy mikrokontroler „szyty na miarę”. Różnego rodzaju peryferia współpracujące z PicoBlazem są dostępne w Internecie, ale można je przygotować także samemu (pokażemy jak to zrobić).

Kod źródłowy programu dla PicoBlaze jest „wkompilowywany” w pamięć ROM. Następnie pamięć ta jest integrowana z mikroprocesorem. Całość po zaimplementowaniu do układu FPGA/CPLD funkcjonuje jak zwyczajny mikrokontroler. Na **rys. 2** pokazano sposób intergracji wszystkich składowych do postaci gotowego produktu.



Rys. 2. Sposób łączenia rdzenia PicoBlaze z peryferiami

Ponieważ w odróżnieniu od zwykłego (scalonego) mikrokontrolera będzie można teraz samemu dodawać i usuwać peryferia, niezbędna jest znajomość sposobu realizacji tego zadania. Stosowane peryferia muszą być opisane za pomocą jednego z języków HDL, przy czym nie jest wymagana znajomość takiego języka w stopniu bardzo dobrym. Podstawowa znajomość jest wystarczająca. W kolejnych odcinkach cyklu przedstawimy przykłady różnych peryferii i konfiguracji mikrokontrolera, które posłużą nam do przygotowania przykładów.

W raz zaimplementowanym mikrokontrolerze można do woli zmieniać zarówno oprogramowanie, jak i istniejące peryferia, łącznie z samym rdzeniem. Jeśli PicoBlaze się nam znudzi, albo zaistnieje sytuacja wymagająca użycia silniejszego procesora, można bez żadnego problemu go wymienić, tak samo łatwo, jak obecnie zmieniamy oprogramowanie scalonych mikrokontrolerów.

PicoBlaze

Jak już wcześniej wspomniano, PicoBlaze jest dostępny w postaci IP core. Ze strony firmy Xilinx można

Konfiguracje systemów mikroprocesorowych z układami PLD

Mikroprocesor/mikrokontroler

Połączenie mikroprocesora z układami peryferyjnymi, jako jeden układ scalony lub też jako kilka układów. Jest to jedna z najpopularniejszych obecnie konfiguracji, znana większości elektronikom na całym świecie.

Mikroprocesor/mikrokontroler plus układ PLD

Obecne najpopularniejsze zastosowanie układów FPGA/CPLD. Rozwiązanie sprawdza się znakomicie przy rozszerzaniu dobrych i bardzo dobrych systemów procesorowych o kolejne możliwości, jakie dają układy reprogramowalne.

Układy PLD zintegrowane z mikroprocesorem

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

Układy PLD z zaimplementowanym mikroprocesorem

Ta konfiguracja jest używana od dość dawna. Układy FPGA/CPLD wykorzystują firmy produkujące zwykłe układy scalone do projektowania i weryfikacji swoich nowych projektów. Jednakże takie zastosowanie układów reprogramowalnych do tej pory było bardzo kosztowne i tylko więksi rynkowi „gracze” mogli sobie pozwolić na takie rozwiązanie. Wraz ze wzrostem wydajności i zasobów logicznych układów PLD, ta konfiguracja zaczyna być coraz bardziej konkurencyjna dla obecnie produkowanych mikrokontrolerów. Ta konfiguracja jest rozwiązaniem, jakie pokażemy w kolejnych odcinkach.

ściągnąć źródła mikroprocesora w postaci listy połączeń, opisanej w języku VHDL lub Verilog. Dodatkowo, dostępny jest pełny opis mikroprocesora w języku angielskim w formie noty aplikacyjnej oraz asembler.

Opis mikroprocesora w postaci listy połączeń powoduje, że jest go bardzo trudno zmodyfikować. Z drugiej strony po implementacji w strukturach układów reprogramowalnych firmy Xilinx, zajmuje on bardzo mało zasobów. W Internecie jest dostępna również implementacja tego rdzenia o nazwie *pacoBlaze*. Jest to funkcjonalnie, pełny odpowiednik PicoBlaze, ale opisany tylko w Verilogu, na wyższym poziomie abstrakcji niż lista połączeń. Ułatwia to znacznie wprowadzanie zmian w samym mikrokontrolerze. Rdzeń *pacoBlaze* jest udostępniony na licencji BSD.

Ponieważ będziemy traktować PicoBlaze jako „czarną skrzynkę" bez wprowadzania do rdzenia jakichkolwiek zmian, sposób jego opisania nie stanowi dla nas żadnej wady.

Narzędzia

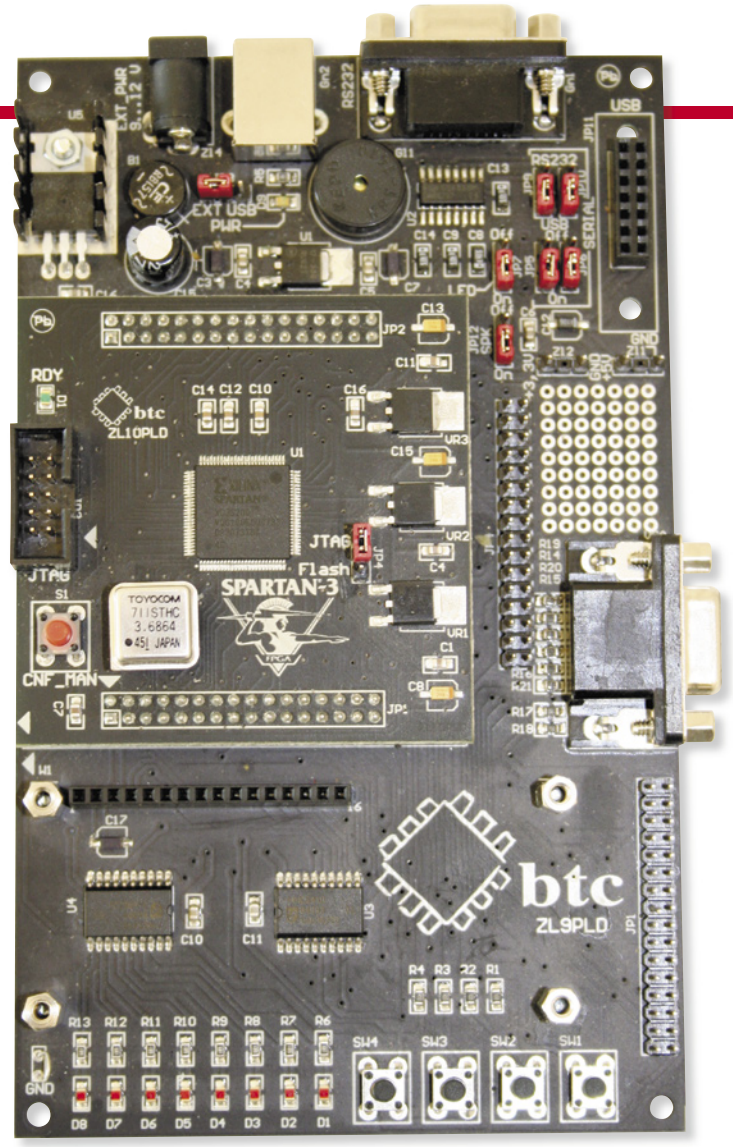
Udostępniony przez firmę Xilinx asembler KCPSM3 do PicoBlaze realizuje dodatkowo jedną bardzo ważną funkcję. Dokonuje on konwersji kodu maszynowego bezpośrednio do postaci opisu pamięci ROM opisanej w języku VHDL lub Verilog. Tak opisane elementy są gotowe do podłączenia do mikrokontrolera.

Do testowania implementacji oprogramowania można używać symulatora *pBlazeIDE* dla Windows lub *kpicosim* dla platformy Linux. Oba programy są dostępne bezpłatnie. Mimo, że korzystają z trochę innej składni asemblera, można je wykorzystywać zamiennie. W kolejnych rozdziałach zostaną ukazane różne możliwości wykorzystania symulatora pBlazIDE.

Do testowania implementacji mikrokontrolera w językach HDL należy użyć jednego z wielu dostępnych progamów do symulacji, m.in. Active–HDL firmy Aldec lub ModelSim firmy Mentor Graphics. Firma Xilinx oferuje również darmową wersję symulatora, która jest dostępna razem z oprogramowaniem do implementacji *ISE Simulator Lite*.

Sprzęt i oprogramowanie

Wszystkie przykłady, jakie przedstawimy w kolejnych odcinkach kursu przetestowano na płytce ewaluacyjnej



Fot. 3. Zestaw uruchomieniowy z układem XC3S200, który jest platformą uruchomieniową dla projektów opisanych w artykule

ZL9PLD + ZL10PLD (**fot. 3**) z firmy *Kamami.pl*. W skład zestawu uruchomieniowego wchodzi układ FPGA Spartan3 firmy Xilinx (XC3S200). Jego zasoby logiczne pozwalają na zaimplementowanie 12 pełnych mikroprocesorów PicoBlaze, zostawiając jeszcze dużo miejsca na dodatkowe układy peryferyjne.

Do implementacji opisanych projektów w strukturach reprogramowalnych zastosowano program WebPack, firmy Xilinx w wersji 9.2i.

Marcin Nowakowski

Bibliografia

[1] http://www.xilinx.com/bvdocs/userguides/ug129.pdf – opis PicoBlaze w języku angielskim.
[2] Kevin Skahill: „Język VHDL", WNT 2006
[3] Mark Zwoliński: „Projektowanie układów cyfrowych z wykorzystaniem języka VHDL", WKŁ, Wydanie 2, Warszawa 2007
[4] Jacek Majewski, Piotr Zbysiński: „Układy FPGA w praktyce", BTC, Warszawa 2007
[5] Praca zbiorowa pod redakcją Józefa Kalisza: „Język VHDL w praktyce", WKŁ
[6] Piotr Zbysiński, Jerzy Pasierbiński: „Układy programowalne – pierwsze kroki", BTC
[7] Jerzy Pasierbiński, Piotr Zbysiński: „Układy programowalne w praktyce", WKŁ
[8] Tadeusz Łuba, Krzysztof Jasiński, Bogdan Zbierzchowski: „Specjalizowane układy cyfrowe w strukturach PLD i FPGA", WKŁ
[9] Tadeusz Łuba, Bogdan Zbierzchowski: „Komputerowe projektowanie układów cyfrowych", WKŁ
[10] Praca zbiorowa pod redakcją Tadeusza Łuby: „Synteza układów cyfrowych", WKŁ
[11] http://bleyer.org/pacoblaze/ – domowa strona pacoBlaze
[12] http://www.mediatronix.com/tools – domowa strona symulatora pBlazeIDE
[13] http://www.xilinx.com/bvdocs/appnotes/xapp213.pdf – nota plikacyjna, omawiająca budowę PicoBlaze (wersje 1 i 2) w języku angielskim