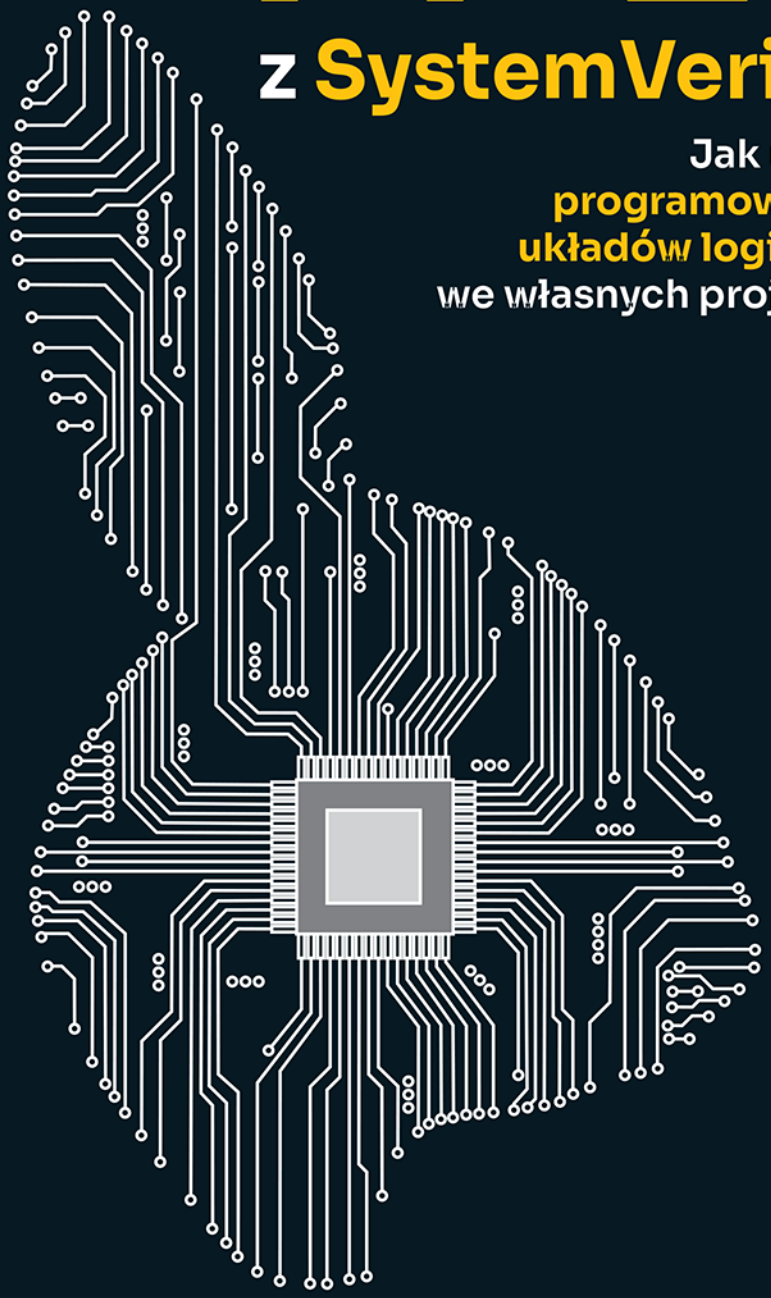
Marta Kozik

Oswoić FPGA z SystemVerilog

Jak używać programowalnych układów logicznych we własnych projektach



Helion

Redaktor prowadzący: Tomasz Gojowy

Skład komputerowy w systemie LaTeX wykonał autor.

Projekt okładki: Studio Gravite/Olsztyn
Obarek, Pokoński, Pazdrijowski, Zaprucki

Materiały graficzne na okładce zostały wykorzystane za zgodą Adobe Stock.

Helion S.A.
ul. Kościuszki 1c, 44-100 Gliwice
tel. 32 230 98 63
e-mail: *helion@helion.pl*
WWW: *https://helion.pl* (księgarnia internetowa, katalog książek)

Drogi Czytelniku!
Jeżeli chcesz ocenić tę książkę, zajrzyj pod adres
https://helion.pl/user/opinie/oswfpg
Możesz tam wpisać swoje uwagi, spostrzeżenia, recenzję.

ISBN: 978-83-289-1444-5

Spis treści

2. A komu to potrzebne?

Układy FPGA są owiane nimbem tajemnicy. Krążą opinie, że są skomplikowane i trudne w użyciu. Postaram się przekonać Cię, Czytelniku, że są to całkiem miłe i przydatne układy, jeżeli tylko wiemy, jak do nich podejść oraz co tak naprawdę mogą nam zaoferować.

FPGA to skrót od angielskiego *field-programmable gate array*. Na polski możemy to spróbować przetłumaczyć jako „bezpośrednio programowalna macierz bramek". Przypuszczam, że to Ci, Czytelniku, nadal nic nie rozjaśniło. A nawet mogło trochę zaciemnić obraz, gdyż obecnie nie znajdziemy już w środku bezpośrednich odpowiedników znanych nam bramek logicznych.

Na początku możemy sobie wyobrazić taki układ jako dużą liczbę elementów realizujących różne funkcje logiczne, przerzutników pozwalających zapisać wyniki obliczeń oraz konfigurowalnych zasobów połączeniowych pozwalających łączyć je ze sobą. W pewnym sensie mamy wielką płytkę stykową, do której wkładamy różne układy scalone, takie jak bramki logiczne, liczniki albo sterowniki wyświetlacza 7-segmentowego, a następnie za pomocą konfigurowalnych kabelków łączymy je między sobą. Tylko tutaj nie musimy robić tego bezpośrednio, a jedynie przygotować opis działania.

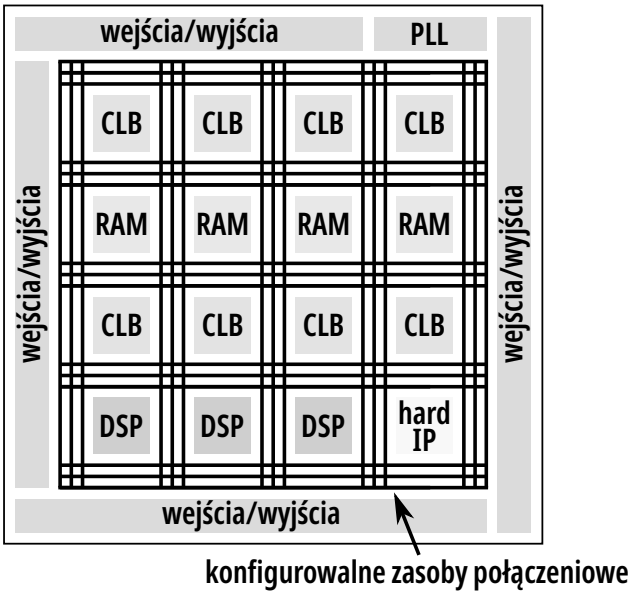
Od razu możemy zauważyć różnice pomiędzy FPGA a mikrokontrolerem. Kod napisany w języku programowania jest zmieniany na ciąg instrukcji. W mikrokontrolerze mamy jednostkę centralną, która wykonuje je jedną po drugiej, czyli **sekwencyjnie**. Nawet gdy jest kilka rdzeni, każdy wykonuje krok po kroku swoje rozkazy.

Sytuacja w układzie FPGA jest diametralnie inna. Nie przygotowujemy programu, ale **konfigurację**. Przy starcie każdy z elementów jest informowany, co ma robić: coś jakbyśmy wybierali z naszych zapasów odpowiednie klocki i wkładali do płytki stykowej. Następnie ustawiamy zasoby połączeniowe, co odpowiada włożeniu łączących je drucików. Od tego momentu wszystkie klocki działają **w tym samym czasie**, czyli **równolegle**. Ciągle pracują na danych, które przepływają przez nasz układ od wejść do wyjść.

Konfigurując układ FPGA, tworzymy swój własny układ scalony. Gdybyśmy zamiast programowalnego układu chcieli wytrawić nasz projekt w krzemie, to stworzylibyśmy tak zwany ASIC (*application-specific integrated circuit* — wyspecjalizowany układ scalony). Jednak jest to proces, którego przygotowanie jest drogie, opłacalne dopiero, gdy potrzebujemy kilkunastu tysięcy sztuk układu (choć koszty te ciągle spadają). Uzyskany tak sprzęt może pracować z większą częstotliwością (szybciej) i przeważnie będzie pobierał mniej prądu. Jednak są też wady. Po uruchomieniu produkcji nie możemy dokonywać już żadnych zmian ani poprawek.

2.1 Co FPGA ma w środku?

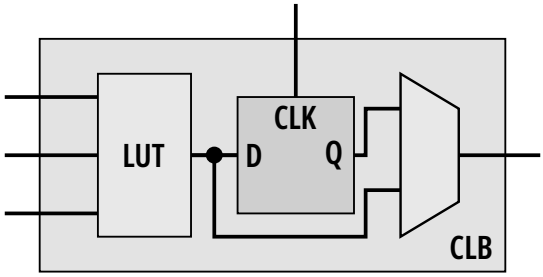
Rysunek 2.1 pokazuje wnętrze typowego układu FPGA. W zależności od producenta poszczególne bloki mogą mieć różne nazwy, ale idea pozostaje niezmienna.



Rysunek 2.1: Budowa wewnętrzna układu FPGA

- **CLB** — to podstawowa cegiełka, z której składamy nasze funkcje logiczne. Jej budowa jest podobna do tej z rysunku 2.2. Składa się z dwóch części. Pierwsza z nich to LUT (*look up table* — tablica wartości, tablicowanie). Jest to konfigurowalna „bramka logiczna". Jak widzimy

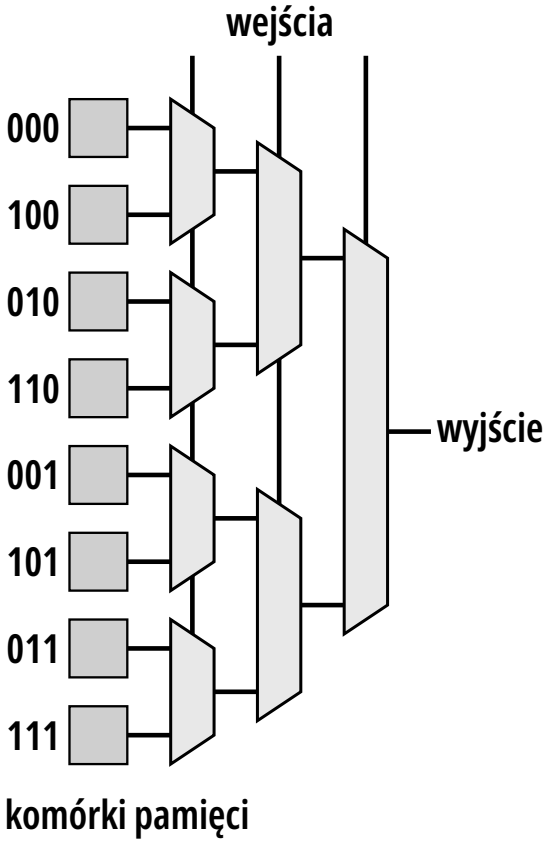
na rysunku 2.3, dla każdej kombinacji wejść mamy jednobitową pamięć, do której na etapie konfiguracji zostanie wpisana oczekiwana wartość. W czasie pracy wejścia sterują multiplekserami, dzięki czemu na wyjście trafia zawartość odpowiedniej komórki pamięci. Wyjście tablicy LUT jest połączone z przerzutnikiem D, w którym możemy zapisać wynik obliczeń. W zależności od producenta i wersji układu jeden blok CLB ma w sobie jedną albo kilka par LUT + przerzutnik. Sama LUT także może mieć różny rozmiar. Przeważnie pozwalają one na zakodowanie funkcji logicznej mającej od trzech do nawet sześciu wejść. Liczba dostępnych bloków także jest różna — od tysiąca w najmniejszych układach do dziesiątek milionów w największych.



Rysunek 2.2: Uproszczona budowa CLB

- **BRAM** — blokowa pamięć RAM. Niewielkie bloki pamięci, zwykle kilka kilobitów. Tak, bitów — uważaj, Czytelniku, gdyż w świecie FPGA często używa się właśnie tej jednostki, a nie częściej spotykanej w informatyce bajt.
- **DSP** — bloki realizujące mnożenie (czasami także inne funkcje matematyczne). Nazwa pochodzi od *digital signal processing* — czyli cyfrowego przetwarzania sygnałów. Inna spotykana nazwa to mnożarka. Nie występują one we wszystkich układach. Ich dokładny opis oraz zalecany sposób pisania kodu, który pozwoli z nich skorzystać, znajdziemy w dokumentacji używanego układu.
- **PLL** — pętla synchronizacji fazy (skrót od *phase-locked loop*). Niech ta skomplikowana nazwa Cię nie przestraszy, Czytelniku. To po prostu blok pozwalający zarządzać zegarem taktującym, spotykany także często w mikrokontrolerach.
- **Blok wejścia/wyjścia** — bloki obsługujące piny naszego układu; pozwalają nam na kontakt ze światem zewnętrznym.
- **Hard IP** — pod tą nazwą kryją się dowolne inne cukiereczki, które przy-

gotował producent po to, aby wyróżnić swój produkt na tle innych. Są to wyspecjalizowane moduły realizujące konkretne zadania. Może to być na przykład kontroler zewnętrznej pamięci czy warstwa fizyczna sieci Ethernet. Wyobraźnia dostawców układów nie zna granic. Szczegółów musimy szukać w dokumentacji.



Rysunek 2.3: Budowa LUT

2.2 Jak powstają projekty?

Wiemy już, z czego składa się układ FPGA. Pierwszą intuicją jest bezpośrednia konfiguracja każdego elementu oraz ręczne ustawianie połączeń pomiędzy nimi. Zasadniczo takie podejście jest możliwe, ale byłoby szalenie niepraktyczne. Przede wszystkim liczba elementów, które trzeba by skonfigurować,

nawet dla prostych projektów byłaby bardzo duża. Po drugie, producenci nie podają na tyle szczegółowych informacji na temat budowy wewnętrznej swoich układów. Niektórzy pasjonaci zajmują się inżynierią wsteczną i odkrywają, za co odpowiadają poszczególne bity w plikach konfiguracyjnych. Ale ich celem nie jest ręczne konfigurowanie układu, tylko tworzenie własnych narzędzi.

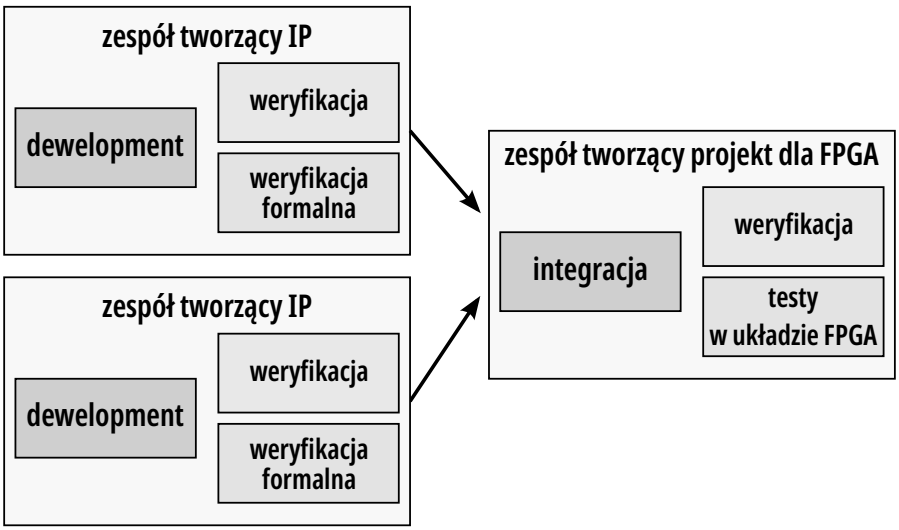
Najczęściej projekty tworzy się w **językach opisu sprzętu**. W przeciwieństwie do języków programowania nie nadają się one do pisania kodu dla procesorów. Pozwalają za to modelować strukturę układów cyfrowych. Jej działanie możemy przetestować za pomocą komputera, ale będzie nam potrzebny dodatkowy program do przeprowadzenia symulacji. Inna nazwa to kod RTL (od angielskiego *register-transfer level*), co odnosi się do przyjętego sposobu modelowania. Opisujemy sygnały, na których wykonujemy operacje logiczne, a wyniki zapisujemy w rejestrach. Dwoma popularnymi obecnie językami są **VHDL** i **SystemVerilog**.

Inną metodą jest graficzne składanie projektu z dostępnych bloków. Metoda ta jest spotykana nawet w dużych projektach. Służy wtedy do integracji mniejszych bloków stworzonych za pomocą innych metod.

Istnieje też wiele języków, gdzie nacisk kładzie się nie na modelowanie układu logicznego, ale opisanie oczekiwanego efektu. Są one zbiorczo nazywane *high level synthesis* (HLS) — synteza wysokopoziomowa. Często wynikiem pracy takiego narzędzia jest kod w którymś z języków RTL, który następnie dołączamy do naszego projektu.

Proces projektowania dla układu FPGA pokazuje rysunek 2.4. Duże zadanie jest zwykle dzielone na małe bloki nazywane **blokami IP** *(Intellectual Property* — własność intelektualna). Zwykle realizują one jakąś konkretną funkcję. Na przykład mogą odpowiadać za obsługę portu szeregowego. Taki moduł może być specyficzny dla konkretnego projektu, ale zwykle jest bardziej uniwersalny. Są firmy, które specjalizują się w dostarczaniu takich modułów. Najbardziej znanym przedstawicielem jest tutaj ARM, który zasadniczo sam nie produkuje swoich procesorów, ale sprzedaje licencje na używanie zaprojektowanego przez siebie rdzenia w produktach innych firm.

Zespół tworzący pojedynczy blok IP dzieli się na kilka grup. Syntezowalny kod RTL tworzą designerzy. Za tworzenie symulacji oraz testów odpowiada weryfikacja. Natomiast grupa od weryfikacji formalnej zajmuje się tworzeniem reguł i udowadnianiem (za pomocą odpowiednich programów), że tworzony kod je spełnia. Celem jest znalezienie i naprawienie jak największej liczby potencjalnych błędów na jak najwcześniejszym etapie projektu.



Rysunek 2.4: Proces tworzenia projektu dla układu FPGA

Następnie kolejny zespół zajmuje się integracją różnych bloków IP w jeden projekt. Na tym etapie również odbywa się symulacja, ale także testy w sprzęcie (weryfikacja sprzętowa). Jak widzimy, duży nacisk kładzie się na sprawdzenie działania projektu w symulacji, a dopiero na końcu w docelowym sprzęcie. Jest to bardzo ważne, gdyż jak już za niedługo się przekonamy, **debugowanie problemów bezpośrednio w układzie FPGA jest praktycznie niemożliwe**, gdyż możemy uzyskać dostęp jedynie do niewielkiej liczby sygnałów. **Symulując działanie naszego projektu na komputerze, możemy uzyskać dużo więcej informacji na temat projektu i szybciej znaleźć popełnione przez nas błędy**.

2.3 Gdzie są używane układy FPGA?

Wiemy już, czym są układy FPGA. Pozostaje jeszcze pytanie, gdzie mogą się przydać. Zwykle w tych miejscach, gdzie mikrokontroler będzie zbyt wolny, a wyprodukowanie odpowiedniego układu scalonego zbyt kosztowne. Najczęściej oznacza to, że:

- Musimy przetworzyć dużą ilość danych w krótkim czasie.
- W przyszłości prawdopodobnie będziemy musieli dodać nowe funkcje.
- Planujemy wyprodukować jedynie niewielką liczbę urządzeń.

Pierwszym przykładem są koprocesory dostępne w chmurach obliczeniowych.

Jednym z bardziej popularnych jest Amazon EC2 F1. Częstotliwościowe prawo Moore'a od pewnego czasu wyhamowało, ale ciągle kolejne generacje układów mają coraz większą liczbę podzespołów. Oznacza to, że aby uzyskać większą wydajność obliczeń, powinniśmy wykonywać jak najwięcej operacji równolegle. Jak już wiemy, tu idealnie sprawują się układy FPGA, dzięki którym użytkownicy mogą stworzyć akcelerator obliczeń dostosowany do rozwiązywanego przez nich problemu.

Innym przykładem jest cyfrowe przetwarzanie sygnałów, które musi odbywać się w czasie rzeczywistym. W tej kategorii mieszczą się takie dziedziny jak telefonia komórkowa, radary czy przetwarzanie obrazu o dużej rozdzielczości. Jeszcze inną dziedziną są zastosowania kosmiczne, gdy chcielibyśmy mieć możliwość naprawy błędów albo dodania nowych funkcji do satelity, który przebywa już na orbicie.

Ważnym zastosowaniem jest też testowanie projektowanych układów scalonych. Ten sam kod w języku RTL może zostać zsyntezowany zarówno w celu wykonania ASIC-a, jak i dla układu FPGA. Pozwala to na wykonanie bardziej złożonych testów, których przeprowadzenie w symulacji trwałoby zbyt długo.

2.4 Kto produkuje sprzęt i narzędzia?

Wiodącymi producentami, którzy mają w swojej ofercie największe układy FPGA, są firmy Intel (kiedyś Altera) oraz AMD (kiedyś Xilinx). Jak widzimy, firmy specjalizujące się w układach FPGA zostały wykupione przez wytwórców procesorów.

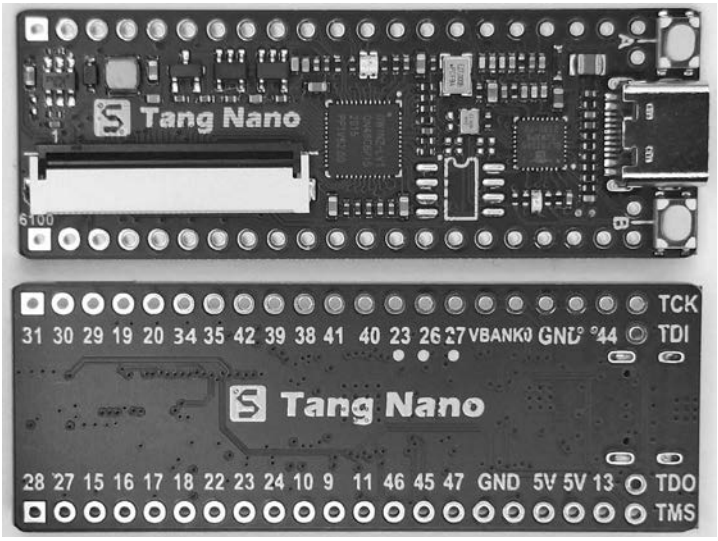
Wśród producentów mających w swoim portfolio mniejsze układy, ale będących stosunkowo długo na rynku, znajdziemy Lattice oraz Microchip. W ostatnim czasie pojawili się też producenci z Chin, tacy jak GOWIN i Analogic.

W kolejnych rozdziałach skupimy się na układach dostarczanych przez firmę GOWIN. Oferują one stosunkowo duże możliwości, przy umiarkowanej cenie zestawów deweloperskich. Jednak naszym celem nie będzie nauczenie się obsługi tego konkretnego układu, ale zdobycie intuicji, czym są układy FPGA oraz jak przygotowywać przeznaczone dla nich projekty. Wprawdzie z konieczności będziemy używać oprogramowania dostarczonego przez producenta układu, **jednak po zapoznaniu się ze środowiskiem innego producenta będzie można wykorzystać zdobytą wiedzę także w pracy z innymi układami.**

Co ciekawe, tworzeniem symulatorów zajmują się całkiem inne firmy. Do najbardziej znanych narzędzi należą VCS firmy Synopsys, Xcelium od Cadence oraz Questa firmy Mentor. W naszych eksperymentach będziemy wykorzystywać program ModelSim, będący uproszczoną wersją tego ostatniego. Istnieją także rozwiązania o otwartym kodzie źródłowym, takie jak Icarus oraz Verdi.

2.5 Co dalej?

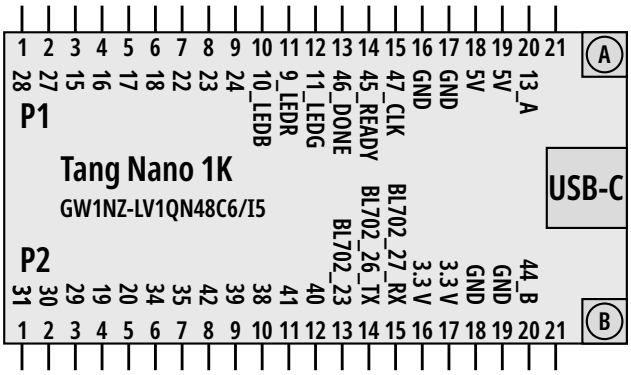
Otóż, drogi Czytelniku, w kolejnych rozdziałach poznasz serię eksperymentów, które będziesz mógł samodzielnie uruchomić w układzie FPGA. Wykorzystamy platformę Tang Nano 1K firmy Sipeed (ważne, aby nie pomylić jej z poprzednią wersją, nazwaną Tang Nano — bez 1K). Jest ona łatwo dostępna w zagranicznych serwisach aukcyjnych. Jej wygląd pokazuje rysunek 2.5. Jest wyposażona w programator oraz układ FPGA GW1NZ-LV1. Dzięki temu poza samą płytką będziemy potrzebować jedynie kabla ze złączem USB-C.



Rysunek 2.5: Płytka Tang Nano 1K

Sam układ jest niewielki. Do dyspozycji będziemy mieli 1152 elementy logiczne oraz 72 Kbity pamięci RAM rozłożonej w 4 blokach. Jednak to wystarczy, aby wykonać wiele ciekawych projektów oraz poznać podstawy języka SystemVerilog.

Większość wyjść układu FPGA została wyprowadzona na złącza. Ich opis prezentuje rysunek 2.6. Na płytce znajdziemy także trójkolorową diodę LED i dwa przyciski. Ważnym elementem jest także programator. W tej roli znajdziemy układ BL702. Jest to mikrokontroler z rdzeniem opartym na technologii RISC-V. Jednak w naszej płytce jest on wyposażony w oprogramowanie, które udaje układ FT2232D. Do układu FPGA dostarcza on interfejs JTAG, natomiast na wyjściach GPIO mamy dostępny port szeregowy. Więcej informacji o tym programatorze znajdziemy pod adresem `github.com/sipeed/RV-Debugger-BL702`.



Rysunek 2.6: Opis wyjść płytki Tang Nano 1K

W następnym rozdziale zainstalujemy symulator ModelSim oraz pakiet GOWIN EDA, a następnie uruchomimy nasz pierwszy projekt.

Programuj i steruj — odkryj tajniki FPGA!

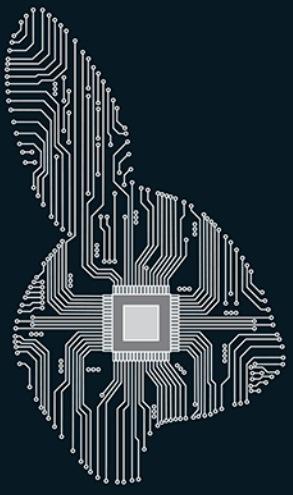
FPGA pochodzi od angielskiego *field-programmable gate array*. Polski odpowiednik to: bezpośrednio programowalna macierz bramek. FPGA jest rodzajem programowalnego układu logicznego. Ma tę samą funkcjonalność co układ scalony, tyle że może być wielokrotnie programowany bez demontażu. Z tego powodu znajduje zastosowanie tam, gdzie wymagana jest możliwość zmiany działania, na przykład w satelitach kosmicznych. Budujesz, instalujesz w urządzeniu docelowym, a potem modyfikujesz układ w zależności od potrzeb. Brzmi praktycznie, prawda?

Tyle niezbędnej teorii, przejdźmy zatem do wspomnianej praktyki, czyli odpowiedzi na pytanie, jak zbudować taki programowalny układ logiczny. Znajdziesz ją właśnie w tej książce. Dowiesz się z niej nie tylko, jakie zastosowanie mają układy FPGA, ale także:

- **Co będzie potrzebne do wykonania własnych eksperymentów**
- **Jak przygotować środowisko pracy**
- **Jakiego rodzaju elementów (układ FPGA, przyciski, diody) należy użyć i jak je połączyć**
- **W jaki sposób zbudować praktyczne projekty, takie jak zegar czy sterownik silnika krokowego**
- **Jak skutecznie obsługiwać port szeregowy**

Marta Kozik

Jest absolwentką automatyki i robotyki na Akademii Górniczo-Hutniczej w Krakowie. Pracowała między innymi z systemem operacyjnym FreeBSD i frameworkiem DPDK. Zajmowała się także implementacją sieci 5G w układach FPGA. Jest autorką blisko pięćdziesięciu artykułów popularnonaukowych z zakresu elektroniki i informatyki.



Helion

helion.pl

HELION S.A.
ul. Kościuszki 1c
44-100 Gliwice
tel.: 32 230 98 63
helion@helion.pl

KOD KORZYŚCI
Sięgnij po więcej!

ISBN 978-83-289-1444-5

9 788328 914445

Cena: 49,90 zł