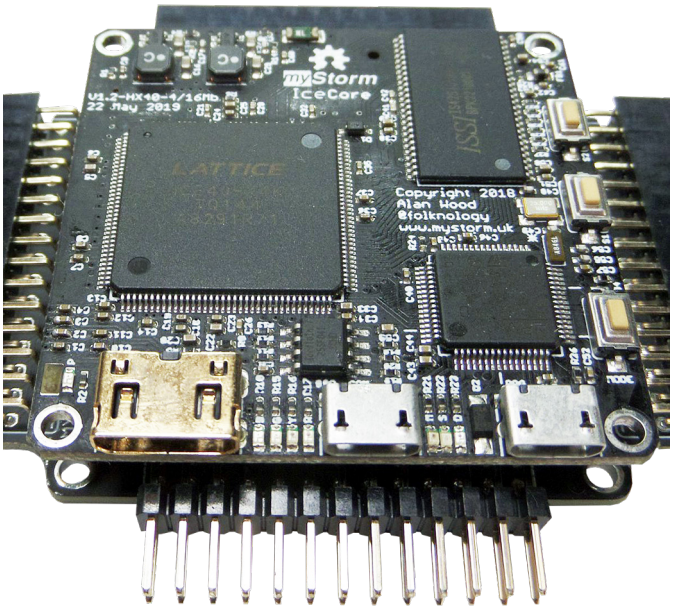
FPGA a Open Source

Płytka prototypowa IceCore z układem Lattice ICE40

W poprzednim artykule, opublikowanym w EP 9/20 (http://bit.ly/35XvZJT), poznaliśmy otwartoźródłowe narzędzia przydatne podczas tworzenia urządzeń opartych na układach FPGA. Aby zastosować wspomniane narzędzia w praktyce, podczas prototypowania będziemy potrzebowali płytki ewaluacyjnej, która pozwoli na prowadzenie własnych eksperymentów. Z uwagi na ograniczenia oraz pełne wsparcie dla otwartych narzędzi, zaczynając od syntezy, na generowaniu pliku bitstream kończąc, najlepszym wyborem będzie płytka na bazie układu ICE40 firmy Lattice.



Na rynku istnieje wiele rozwiązań dla rodziny układów Lattice, jednak jeśli chcemy pozostać wierni idei otwartego oprogramowania, najlepszym wyborem będzie zakup urządzenia, którego dokumentacja jest dostępna na licencji Open Hardware. Jednym z takich rozwiązań jest płytka ICECore ICE40 HX, do której jest dostępna pełna dokumentacja projektowa. Można ją nabyć w cenie około 50 USD (*tindie.com*), co wydaje się stosunkowo atrakcyjną ceną, gdy nie planujemy zajmować się własnoręcznym wykonaniem płytki na podstawie dostępnej dokumentacji.

Za wyborem tego rozwiązania przemawia również fakt, że w skład zestawu wchodzi nie tylko sam układ ICE40, ale również dodatkowy mikrokontroler STM32F730 taktowany zegarem 200 MHz, który jest połączony z FPGA za pomocą dwóch magistral SPI. Dodatkowo do dyspozycji mamy pamięć SDRAM oraz FLASH-SPI, co umożliwia tworzenie zaawansowanych rozwiązań, jak na przykład implementacja własnych układów SOC.

Budowa płytki IceCore ICE40 HX

Zestaw uruchomieniowy ICE core jest jednym z tańszych zestawów ewaluacyjnych, którego dokumentacja dostępna jest na otwartej licencji Open Hardware. Charakteryzuje się następującymi parametrami:

- bazuje na układzie FPGA Lattice HX4K zawierającym 4000 bloków LUT oraz 80 kb pamięci Block RAM,
- zawiera mikrokontroler STM32F730 mający 256 kB pamięci RAM oraz 64 kB pamięci FLASH,
- pamięć SDRAM 16 Mbit (2 MB) o maksymalnej częstotliwości 143 MHz,
- pamięć FLASH 16 MBit (2 MB) o maksymalnej częstotliwości pracy 100 MHz,
- złącze karty SD,
- 4 diody LED oraz 2 przyciski,
- dwa złącza micro USB, w tym jedno przeznaczone do programowania oraz konfiguracji,
- dodatkowe cztery złącza zewnętrzne dołączone zarówno do mikrokontrolera, jak i układu FPGA.

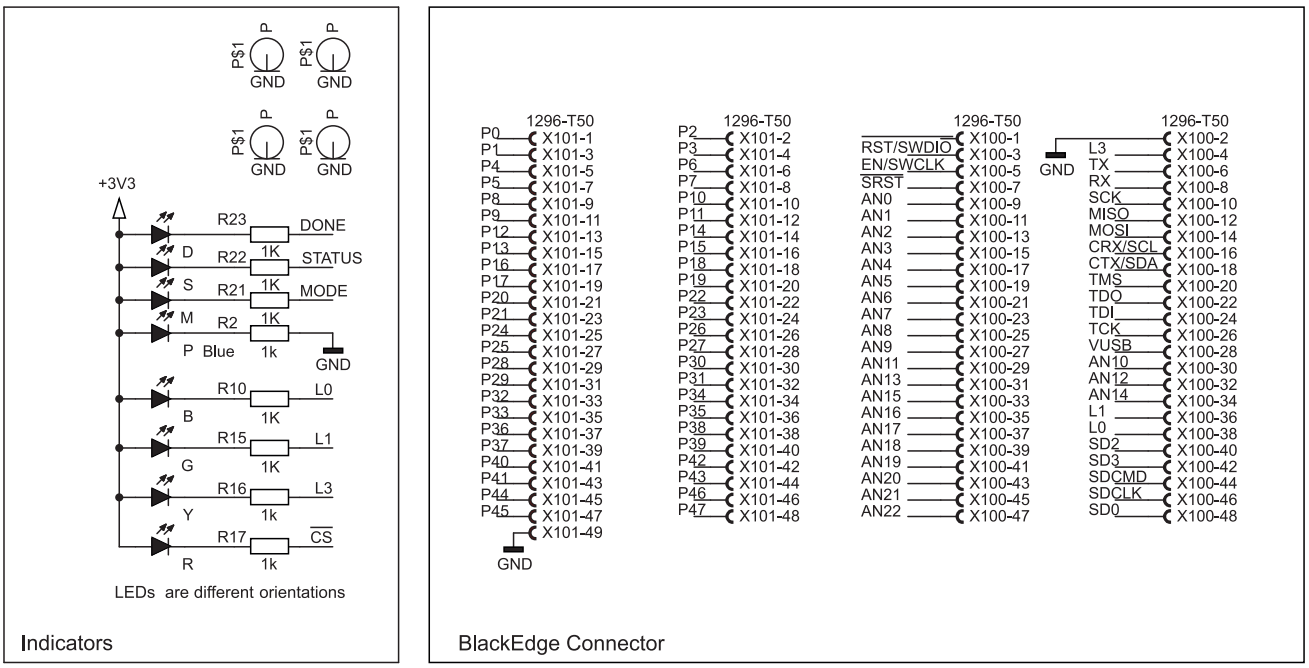
Schemat ideowy zestawu został pokazany na **rysunku 1**. Zasilanie doprowadzane jest bezpośrednio z portu USB za pomocą złączy USB-COM lub USB-PRG. Napięcie z portu USB po odfiltrowaniu podawane jest na dwa stabilizatory impulsowe dostarczające napięć 1,2 V do zasilania rdzenia FPGA (Vcc) oraz 3,3 V do zasilania części IO FPGA i mikrokontrolera. Napięcie 3,3 V po podaniu przez dodatkowy filtr RC jest również podawane do części analogowej układu IC2.

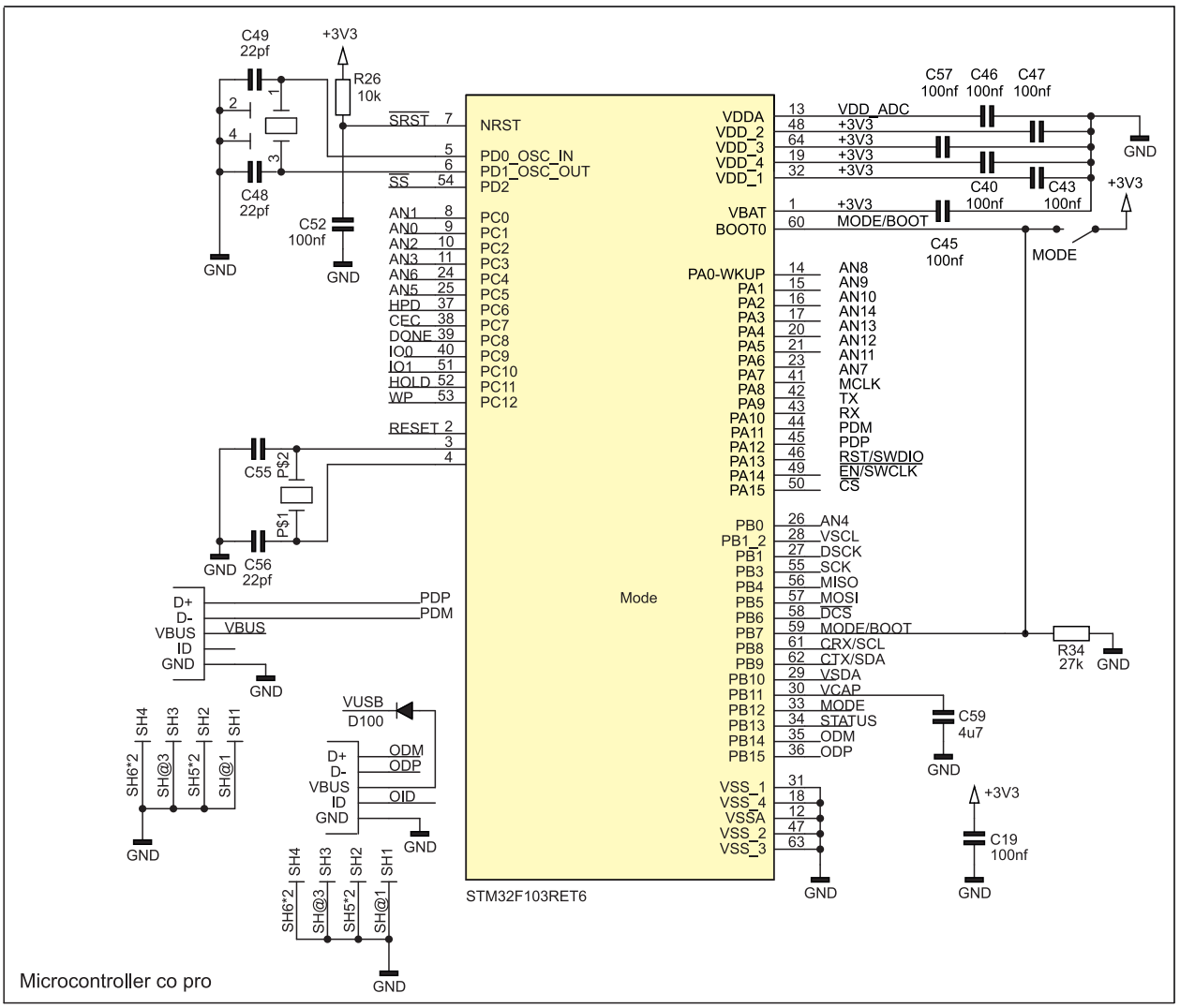
Istotną funkcję w zestawie pełni mikrokontroler STM32F730 (IC2), połączony jest z układem programowalnym za pomocą dwóch magistral SPI. Magistrala dołączona do linii 73...75 układu FPGA przeznaczona jest do ogólnej komunikacji pomiędzy układem FPGA i mikrokontrolerem. Natomiast magistrala dołączona do linii 63, 64, 67, 68, 70, 71, które stanowią wejście sprzętowego kontrolera SPI, pełni potrójną funkcję. Jej głównym zadaniem jest konfigurowanie FPGA przez mikrokontroler (linia wyboru CS (56)), po przesłaniu pliku konfiguracyjnego z portu USB-PROG przez komputer. Drugą funkcją magistrali jest dostęp do pamięci SPI-FLASH, która aktywowana jest za pomocą linii wyboru CS1 (55). Trzecią funkcją jest dostęp do pamięci SERIAL-FLASH, FPGA oraz mikrokontrolera za pomocą złącza zewnętrznego (COMS/DBG). Ponieważ pamięć dostępna jest zarówno z mikrokontrolera, jak i FPGA, może być używana przez oba układy.

Część linii mikrokontrolera oznaczonych jako AN0...AN23 dostępna jest na złączach zewnętrznych i może być używana np. do pomiaru wielkości analogowych za pośrednictwem mikrokontrolera, a następnie przekazania zmierzonych wartości przez magistralę SPI do układu FPGA. Mikrokontroler ma również dwa kontrolery USB (HS i FS), z których jeden z domyślnym firmware jest potrzebny do konfigurowania układu FPGA (złącze USB-PRG), natomiast drugi (USB-COM) jest przeznaczony do ogólnego użycia w aplikacjach.

Linie interfejsu SWD zostały wyprowadzone na złącze COMS/DBG, dzięki czemu oprogramowanie wewnętrzne może być dowolnie zmieniane oraz debugowane. Linie portu szeregowego UART1 STM32 TX/RX, które z domyślnym oprogramowaniem pełnią funkcję wirtualnego portu szeregowego COM-USB, dołączono do linii 61 i 62 układu FPGA. Zatem jeżeli zaimplementujemy sprzętowy port szeregowy po stronie FPGA i dołączymy go do powyższych wyprowadzeń, nie

będzie konieczności używania dodatkowej przejściówki USART ↔ USB. Na złącza zewnętrzne wyprowadzono również dwie sprzętowe magistrale I²C mikrokontrolera, które nie są połączone z układem programowalnym. Jest to rozsądna decyzja, ponieważ magistrala I²C nie jest zbyt szybka a implementacja w FPGA zajmuje dużo cennych bloków. Płytka dysponuje pamięcią SDRAM o niezbyt dużej pojemności 16 Mb, dołączonej bezpośrednio do linii IO układu ICE40. Przy tak niewielkiej pojemności lepiej było zastosować pamięć statyczną lub układ o pojemności 128 Mb, co znacząco zwiększałoby możliwości zestawu.





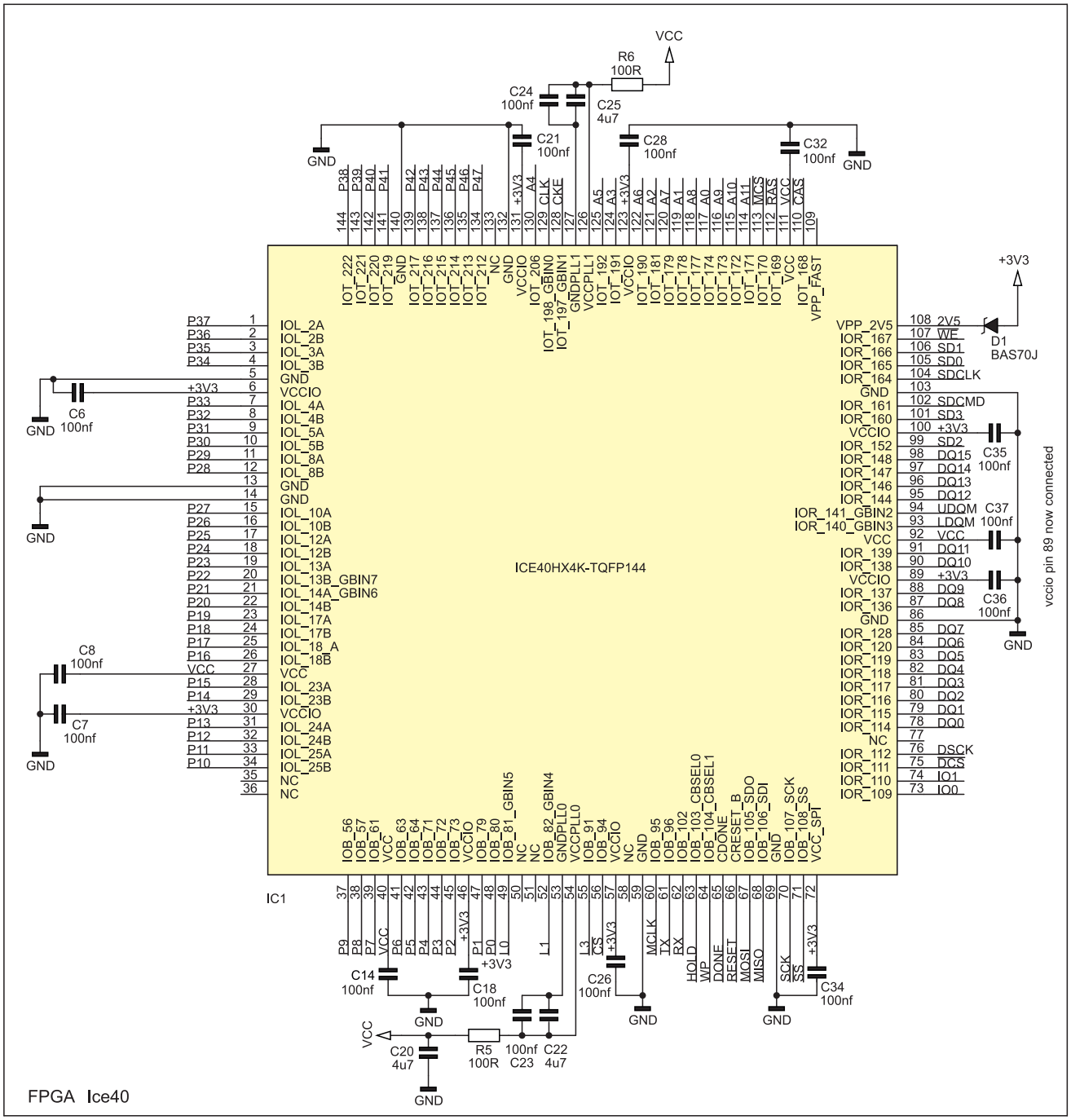
Rysunek 1. Schemat ideowy zestawu IceCore ICE40 HX

Sygnał zegarowy wytwarzany jest przez generator HSE mikrokontrolera za pomocą rezonatora kwarcowego o częstotliwości 25 MHz. Służy on do taktowania mikrokontrolera oraz jako źródło sygnału zegarowego dla układu FPGA. W tym celu wyjście MCLK mikrokontrolera połączone jest z wejściem GPIOB_95 układu FPGA. Jeśli będziemy zmieniali domyślny program zawarty w mikrokontrolerze, musimy pamiętać, aby pozostawić aktywną funkcję wyjścia sygnału zegarowego na linii MCLK. W przeciwnym przypadku pozbawimy się źródła taktowania.

Złącze karty SD zostało połączone wyłącznie z układem programowalnym, a zatem obsługa karty SD wymagać będzie zaimplementowania kontrolera SPI lub SDIO bezpośrednio w tym układzie.

Od strony użytkownika do dyspozycji mamy 7 diod LED, z których 3 z nich dołączone są bezpośrednio do portów GPIO mikrokontrolera. Pozostałe 4 diody LED dołączone są do FPGA. Diody dołączone do mikrokontrolera, z domyślnym firmware pełnią funkcję diagnostyczną, informując o stanie, w jakim obecnie znajduje się układ programowalny. Dioda oznaczona jako CDDONE (czerwona) informuje nas o tym, że konfiguracja układu przebiegła pomyślnie. Dioda oznaczona jako STATUS (zielona) świeci się w trakcie procesu konfiguracji. Dioda oznaczona jako MODE (żółta) dołączona jest do linii CS pamięci FLASH, a zatem informuje nas o aktywności tej pamięci.

Płytka ewaluacyjna wyposażona jest w złącza z rastrem 2,54 mm wyprowadzające część linii FPGA oraz mikrokontrolera na zewnątrz (**rysunek 2**). Dzięki nim możemy rozszerzyć możliwości układu za pomocą innych dołączonych urządzeń. Do dyspozycji mamy złącza MX1... MX3 wyprowadzające linie IO układu FPGA oznaczone za pomocą symboli P oraz linie GPIO mikrokontrolera oznaczone jako symbole zaczynające się od prefiksu AN. Szczególny nacisk położono na to, aby na złączu wyprowadzone było jak najwięcej linii pełniących funkcję wejścia przetwornika analogowo-cyfrowego. Po odpowiednim zmodyfikowaniu firmware będziemy mieli możliwość przesłania wielkości analogowych do układu FPGA za pomocą wewnętrznej magistrali SPI łączącej oba układy. Na płytce zostało również wyprowadzone złącze



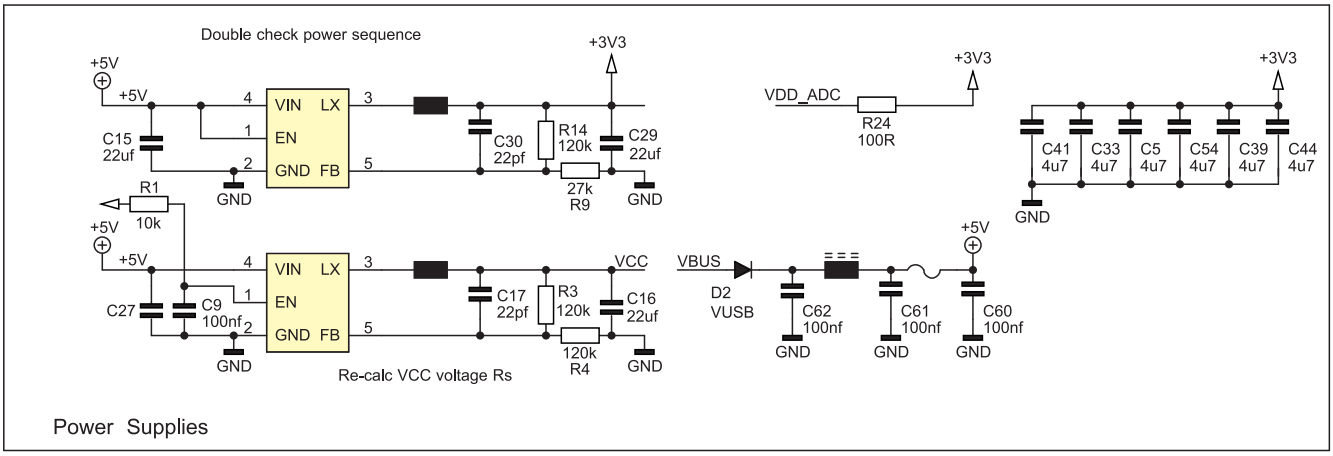
Rysunek 1. cd.

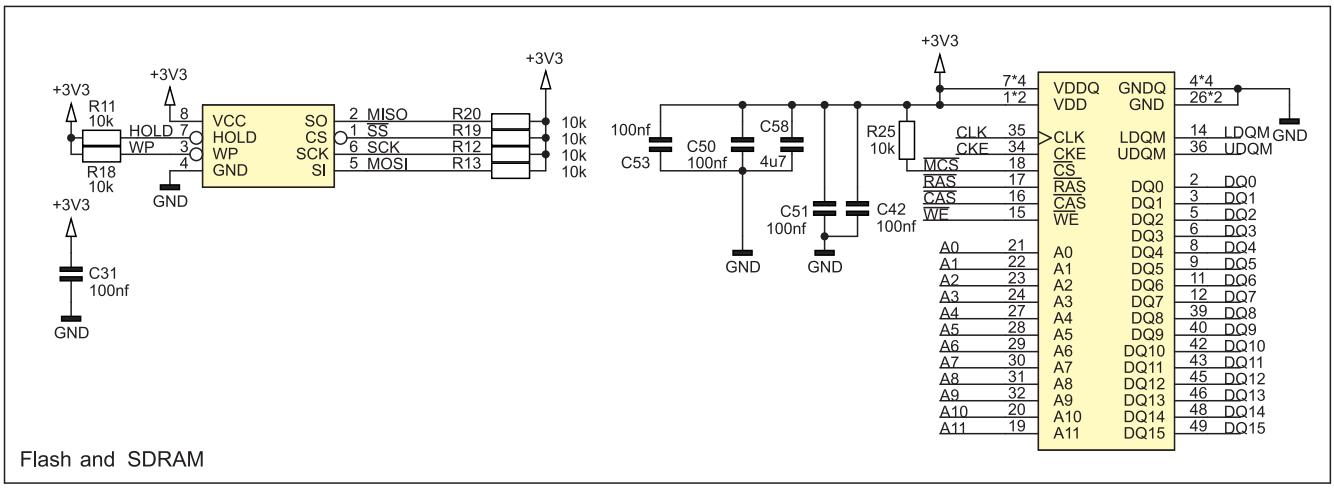
MINI-HDMI, jednak zgodnie z dokumentacją zestawu nie działa ono prawidłowo, z tego powodu jego opis został pominięty.

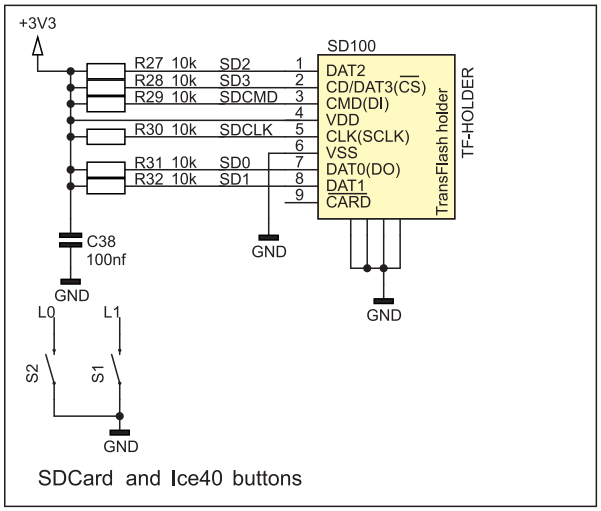
Lattice ICE40HX-4K jest stosunkowo tanim układem programowalnym wyprodukowanym w litografii 40 nm. Występuje zarówno w obudowach BGA, jak i w obudowach TQFP, które bez większych problemów mogą być stosowane w projektach amatorskich. Układ zawiera około 4000 makrokomórek oraz pamięć BlockRAM o wielkości 80 kb. Jego budowę wewnętrzną pokazuje **rysunek 3**. Topologia układu zorganizowana jest w postaci dwuwymiarowej tablicy programowalnych bloków logicznych składających się z 8 makrokomórek. Porty wejścia-wyjścia zostały umieszczone na obrzeżach bloków logicznych i są pogrupowane w 4 banki. Pomiędzy poszczególnymi blokami rozmieszczona jest pamięć BlockRAM, która może pracować jako pamięć ROM, RAM lub w trybie kolejki FIFO.

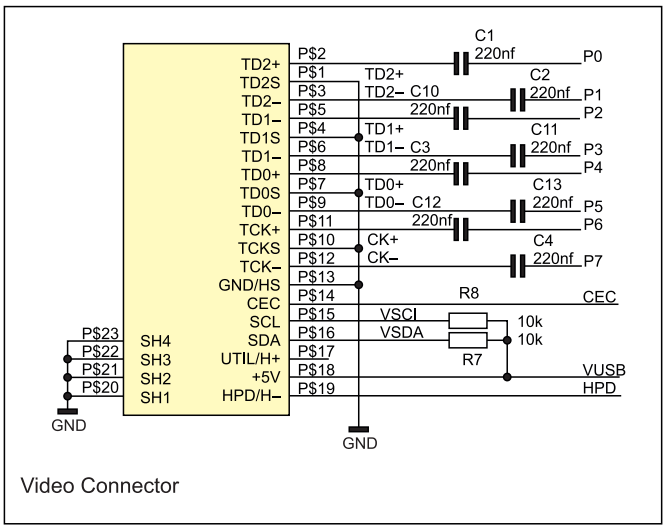
W układzie dodatkowo mamy pętlę PLL, dzięki czemu istnieje możliwość powielenia sygnału zegarowego pochodzącego z dowolnego wejścia IO. Sprzętowy kontroler SPI pełni podwójną funkcję, z jednej strony służy do konfiguracji układu, natomiast po zakończeniu konfiguracji jest dostępny jako blok wewnętrzny i może służyć np. do obsługi pamięci szeregowej.

Każda makrokomórka zbudowana jest w oparciu na bloku LUT4, który pozwala zrealizować dowolną funkcję logiki kombinacyjnej z maksymalnie 4 wejściami. Wyjście bloku LUT połączone jest z wejściem przerzutnika typu D z opcjonalnym wejściem zezwolenia

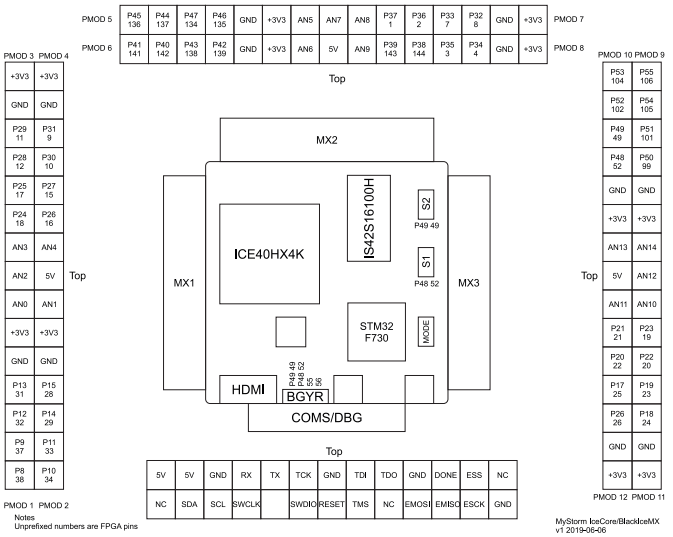




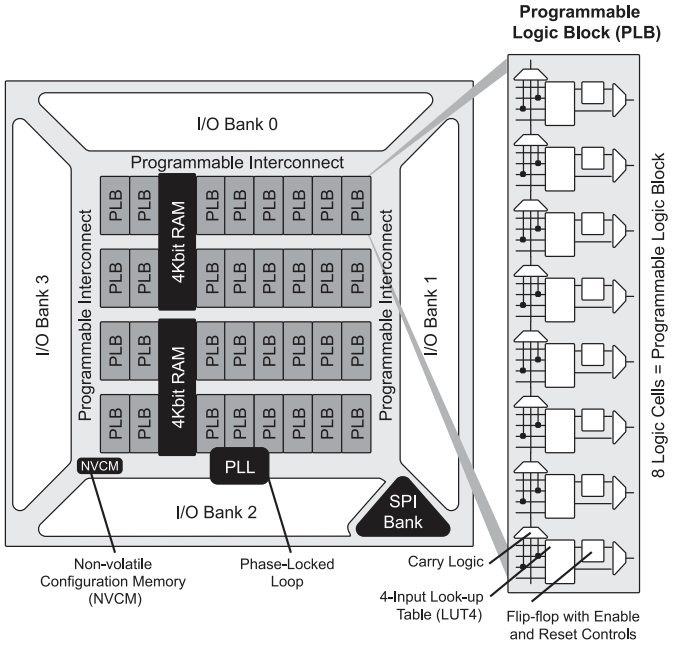




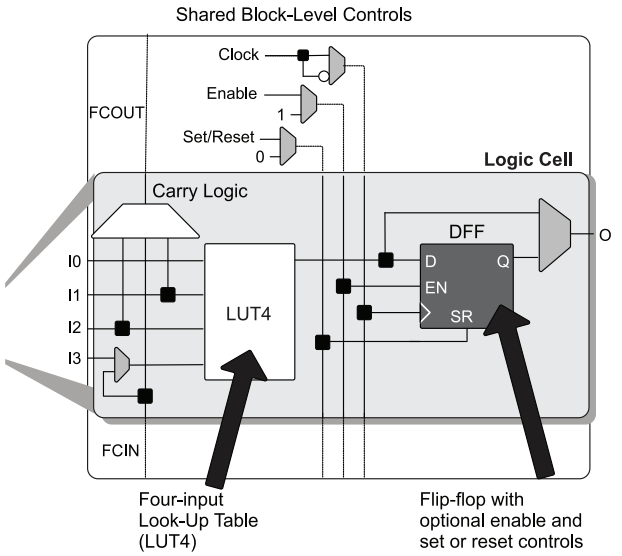
Rysunek 1. cd.



Rysunek 2. Część linii FPGA oraz mikrokontrolera wyprowadzona jest na złącza rozszerzeń



Rysunek 3. Uproszczona budowa wewnętrzna układu Lattice ICE-40HX-4K



Rysunek 4. Budowa makrokomórek logicznych zastosowanego układu FPGA

na sygnał zegarowy oraz wejściem SR, które jest dołączone do globalnej linii reset (**rysunek 4**). Dodatkowy blok logiczny sumatora zwiększa efektywność implementacji funkcji arytmetycznych.

Układ FPGA zawiera 8 globalnych sieci, które mogą być używane do dystrybucji sygnałów zegarowych lub innych szybkozmiennych sygnałów. Globalny sygnał RESET dołączony jest do wszystkich wewnętrznych bloków i jest automatycznie aktywowany podczas konfiguracji, przez co wszystkie bloki logiczne po uruchomieniu ustawiane są w stan domyślny.

Obsługa zestawu

Zestaw ewaluacyjny dostarczany jest z zaprogramowanym układem STM32F730, którego oprogramowanie wewnętrzne pełni następujące funkcje:

- generuje sygnał zegarowy na wyjściu MCLK, który jest głównym sygnałem zegarowym dla FPGA;
- jest odpowiedzialny za konfigurację układu FPGA, z komputera PC poprzez port USB;
- steruje diodami LED sygnalizującymi stan pracy układu programowalnego;
- udostępnia funkcję wirtualnego portu szeregowego, którego linie dołączone są do układu FPGA;
- udostępnia wirtualny port szeregowy, do którego należy przesłać plik bitstream, aby skonfigurować układ programowalny.

Jeśli planujemy skupić się jedynie na układzie programowalnym, to oprogramowanie mikrokontrolera możemy pozostawić w stanie fabrycznym. W tym trybie mikrokontroler pełni jedynie funkcję konfiguratora dla układu FPGA oraz udostępnia dodatkowy wirtualny port szeregowy po stronie komputera PC, za pomocą którego można przesłać konfiguracje. Po połączeniu płytki ewaluacyjnej z komputerem za pomocą złącza USB-PROG urządzenie zgłasza się jako wirtualny port szeregowy. W systemie Linux jest on widoczny jako:
`/dev/ttyAMA0`,
natomiast w systemie OSX jako:
`/dev/cu.usbmodem00000000001A1`.

Jeśli chcemy skonfigurować układ programowalny, wystarczy, że prześlemy bezpośrednio do tego urządzenia plik bitstream, którym chcemy zaprogramować układ, np. za pomocą polecenia cat:

```
cat bistream.bin > /dev/cu.usbmodem0000000001A1,
```

a mikrokontroler automatycznie rozpocznie konfigurację układu FPGA, przekazując dane konfiguracyjne za pomocą interfejsu SPI. Po zakończeniu transmisji przesłana konfiguracja jest automatycznie aktywowana. Jeśli konfiguracja układu zakończyła się powodzeniem, zostanie zapalona dioda STATUS w kolorze zielonym. Natomiast na etapie programowania świecić się będzie dioda czerwona oznaczona jako BUSY. Należy zwrócić uwagę na to, że konfiguracja jest zapisywana tylko w pamięci konfiguracyjnej FPGA, zatem po wyłączeniu napięcia zasilającego układ straci swoją zawartość. Taki tryb pracy jest wygodny na etapie uruchamiania i prototypowania, jednak jest nieprzydatny w praktycznych zastosowaniach. Gdybyśmy chcieli używać zestawu do praktycznych celów, wtedy oprogramowanie STM32 należy tak zmodyfikować, aby konfiguracja była odczytywana z nieulotnej pamięci FLASH po włączeniu zestawu. Naturalnie oprogramowanie domyślne również trzeba będzie wymienić, gdy będziemy chcieli realizować projekty, w których układ programowalny będzie współpracował z mikrokontrolerem.

Dokumentacja oraz oprogramowanie

Największą zaletą opisanej płytki ewaluacyjnej jest pełna dokumentacja projektowa oraz wszystkie kody źródłowe, które możemy pobrać z repozytorium projektu pod następującym adresem URL: http://bit.ly/39Nf9Pg. Repozytorium zostało podzielone na 4 podkatalogi:

- *examples* – zawiera dwa przykładowe projekty błyskające diodą LED, opracowane z użyciem otwartych narzędzi opisanych w poprzednim artykule;

- *cad* – zawiera projekt PCB oraz schemat, zaprojektowany z użyciem oprogramowania Eagle, oraz pliki PDF i GERBER, stanowiące dokumentacje wynikową;
- *firmware* – zawiera kod źródłowy domyślnego firmware umieszczonego w pamięci mikrokontrolera STM32F730.

Oprogramowanie sterujące zostało napisane w języku C z użyciem bibliotek standardowych dostarczanych przez firmę ST w środowisku STM32Cube. Zawiera ono wygenerowany przez konfigurator kod źródłowy z klasą USB CDC-ACM uzupełniony o funkcje odpowiedzialne za komunikację z układem FPGA. Z uwagi na to, że magistrala SPI układu programowalnego została dołączona do portów GPIO w taki sposób, że nie jest możliwe użycie sprzętowego układu SPI, obsługa magistrali SPI i FPGA została zrealizowana w sposób programowy z użyciem portów GPIO. Najistotniejsze fragmenty programu związane z odbieraniem danych oraz konfiguracją układu FPGA znajdują się w pliku *iceboot/iceboot.c*.

Listing 1. Fragment funkcji loop(), która stanowi główny kod programu

```
void loop(void){
    uint8_t b = 0;
    static int err = 0;

    if (err) {
        status_led_toggle();
        HAL_Delay(100);
        if(gpio_ishigh(MODE_BOOT)) {
            err = 0;
            status_led_low();
        }
        return;
    }
    cdc_puts("Waiting for USB serial\n");
    do {
        if(gpio_ishigh(MODE_BOOT)) {
            mode_led_toggle();
            mode = ~ mode;
            HAL_Delay(1000);
        }
        if (cdc_stopped && USBD_CDC_ReceivePacket(&hUsbDeviceFS) == OK)
            cdc_stopped = 0;
        fifo_get(&in_fifo, &b);
    } while (b != 0x7E);
    status_led_high();
    protect_flash();
    hold_flash();
    spi_reattach();
    err = ice40_reset();
    if (err)
        return;
    crc_reset();
    ice_write(&b, 1);
    crc_update(b);
    if ((err = rbits(rbyte_uart_send, b)) != OK) {
        cdc_puts("rbits failed\n");
        return;
    }
    err = ice40_configdone();
    spi_detach();
    release_flash();
    free_flash();
    status_led_low();
}
```

Główny kod programu realizowany jest przez funkcję *loop()*, której fragment został pokazany na **listingu 1**. Funkcja oczekuje na otrzymanie nowej porcji danych ze stosu USB realizującego funkcję urządzenia klasy CDC-ACM. Po odebraniu prawidłowego pakietu startowego diody LED są ustawiane w stan początkowy oraz zerowany jest układ FPGA. W kolejnym kroku poszczególne znaki odebrane z biblioteki USB są analizowane przez wewnętrzną maszynę stanów, której zadaniem jest sprawdzanie poprawności przesyłanych danych. Równocześnie dane te trafiają również do pamięci konfiguracyjnej FPGA za pomocą magistrali SPI. W przypadku stwierdzenia błędu transmisji konfiguracja jest przerywana oraz włączana jest dioda LED sygnalizująca błąd. Następnie oprogramowanie ignoruje odebrane dane i przechodzi do oczekiwania na rozpoczęcia nowego procesu transmisji. W przypadku, gdy wszystkie dane konfiguracyjne zostały poprawnie odebrane, następuje proces aktywacji przesłanej konfiguracji, za co odpowiada funkcja *ice40_configdone()*. Od tego momentu układ FPGA wykonuje powierzoną funkcję, natomiast mikrokontroler przechodzi w stan oczekiwania, oczekując na rozkaz ponownej konfiguracji.

Bazując na kodzie firmware dostarczonym przez twórcę zestawu, możemy dowolnie modyfikować zachowanie płytki, np. zapisując dane konfiguracyjne równocześnie w pamięci FLASH, oraz zmienić oprogramowanie tak, aby po włączeniu napięcia zasilającego konfiguracja była odtwarzana z pamięci nieulotnej. Naturalnym zastosowaniem, jakie się nasuwa, jest współpraca układu FPGA z STM32, który będzie pełnił funkcję procesora wykonującego obliczenia, a w układzie programowalnym będą zaimplementowane nietypowe urządzenia peryferyjne. Można również przygotować program, w którym dane z przetworników analogowo-cyfrowych będą przesyłane po wewnętrznej magistrali SPI, przez co ICE40 zyska możliwość przetwarzania sygnałów analogowych.

Innym aspektem otwartości projektu jest bogate wsparcie społeczności, która skupia się wokół forum: http://bit.ly/3914npp. Możemy tam znaleźć szereg ciekawych porad oraz wiele przykładowych projektów od najprostszych do zaawansowanych, takich jak przykładowy mikrokontroler RISC-V nazwany Pico-SOC.

Mając dostęp do pełnego kodu źródłowego oraz dokumentacji, ogranicza nas tylko wyobraźnia. Możemy znaleźć tysiące różnych zastosowań dla mikrokontrolera STM32 połączonego z układem FPGA, co czyni wspomniany zestaw rozwiązaniem bardzo uniwersalnym a dodatkowo dostępnym w bardzo rozsądnej cenie.

Lucjan Bryndza, EP
lucjan.bryndza@ep.com.pl