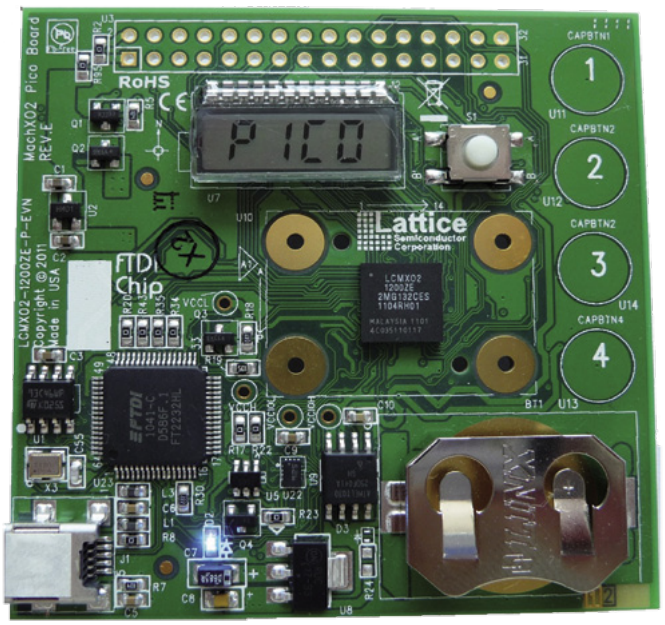
Projektowanie PLD/FPGA z zestawem Lattice MachXO2 Pico Development Kit

Pod koniec pierwszego kwartału bieżącego roku firma Lattice wprowadziła do produkcji seryjnej nową serię układów CPLD o nazwie MachXO2. Produkowane w procesie 65 nm, bazujące na technologii Flash układy MachXO2, w porównaniu ze swoim poprzednikiem – rodziną MachXO, charakteryzują się 3-krotnie większymi zasobami logicznymi, 10-krotnie większą pojemnością wbudowanej pamięci RAM, więcej niż 100-krotnym zmniejszeniem poboru mocy (statyczny pobór mocy wynosi zaledwie 19 mikrowatów) oraz 30% spadkiem ceny. Oznacza to, że układy MachXO2, o zasobach logicznych sięgających 7000 tablic LUT, mogą swobodnie konkurować z „mniejszymi" układami FPGA rodzin Actel IGLOO, Altera Cyclone czy Xilinx Spartan.



Wraz z nowymi układami Lattice udostępnił również zestaw ewaluacyjny MachXO2 Pico, umożliwiający praktyczne poznanie najważniejszych właściwości układów rodziny MachXO2. Celem tego kursu jest zilustrowanie sposobu projektowania stosunkowo nieskomplikowanych systemów cyfrowych, opartych na nowoczesnych układach programowalnych, z wykorzystaniem zestawu MachXO2 Pico.

Kurs rozpoczniemy od zaprezentowania właściwości układów Lattice MachXO2, pokażemy architekturę zestawu *MachXO2 Pico Developmnet Kit*, opiszemy nowe środowisko projektowe *Lattice Diamond Design Software*, a następnie przedstawimy szereg przykładowych projektów zrealizowanych z wykorzystaniem wspomnianego zestawu ewaluacyjnego.

Architektura układów PLD Lattice MachXO2

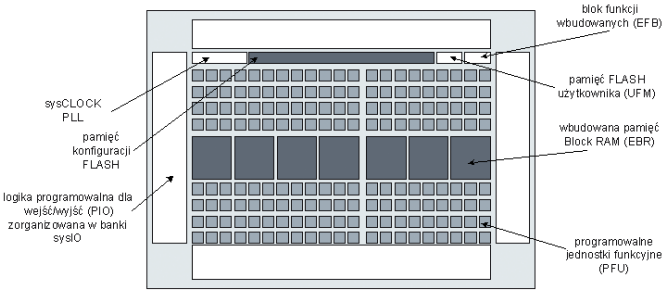
Chociaż producent na kartach dokumentacji technicznej dla tej rodziny konsekwentnie stosuje określenie PLD (*Programmable Logic Device*), jednak układy te mają wiele cech charakterystycznych dla typowych struktur FPGA (np.: komórki logiczne oparte na tablicach LUT, wbudowane bloki pamięci RAM, układy pętli synchronizacji fazowej PLL dla sygnałów zegarowych). W połączeniu z takimi właściwościami, jak bardzo mały pobór mocy, wbudowana nieulotna pamięć konfiguracji Flash z wydzieloną przestrzenią dla danych użytkownika, wewnętrzny oscylator CMOS, wbudowane sprzętowe bloki kontrolera magistral SPI i I²C, a także licznika/czasomierza, wsparcie dla wstępnego przetwarzania sygnałów synchronicznych w komórkach wejścia/wyjścia (DDR, DDR2, LPDDR) oraz szeroki zakres standardów obsługiwanych przez bufory wejścia/wyjścia (LVCMOS, LVTTL, PCI, LVDS, SSTL, HSTL) – z inżynierskiego punktu widzenia rodzina układów MachXO2 wydaje się bardzo atrakcyjna.

Na **rysunku 1** przedstawiono ogólny schemat architektury wybranego układu z rodziny MachXO2. Programowalne jednostki funkcyjne PFU (*Programmable Function Unit*) oraz bloki pamięci EBR (*Embedded Block RAM*) są rozmieszczone w wierszach i kolumnach dwuwymiarowej siatki. Siatka ta jest z zewnątrz otoczona przez programowalną logikę wejścia/wyjścia PIO (*Programmable Input/Output*). Jednostki PFU oraz pamięci EBR są ze sobą połączone za pośrednictwem programowanej matrycy połączeń (*routing channel resources*). Architektura innych układów z tej rodziny jest identyczna z wyjątkiem liczby dostępnych jednostek PFU, bloków pamięci EBR oraz rozmiaru pamięci FLASH użytkownika UFM (*User Flash Memory*). Jedynie „najmniejszy" z oferowanych układów rodziny MachXO2 nie ma wbudowanej pamięci RAM, pamięci UFM oraz bloku PLL.

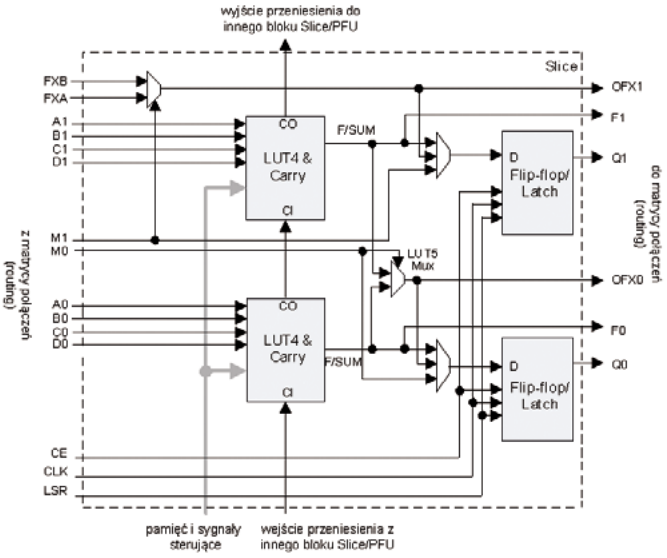
Programowalne jednostki funkcyjne PFU

Zasadniczym elementem architektury układów rodziny MachXO2 są programowalne jednostki funkcyjne PFU, realizujące operacje logiczne i arytmetyczne, a także mogące pełnić funkcję rozproszonych pamięci RAM i ROM (*distributed RAM, ROM*). Każda jednostka PFU składa się z 4 połączonych ze sobą bloków *slice*. Strukturę pojedynczego bloku *slice* przedstawia **rysunek 2**. Każdy blok *slice* zawiera, z kolei, dwie 4-wejściowe tablice LUT (generatory funkcji, *Look-Up Table*) oraz dwa przerzutniki.

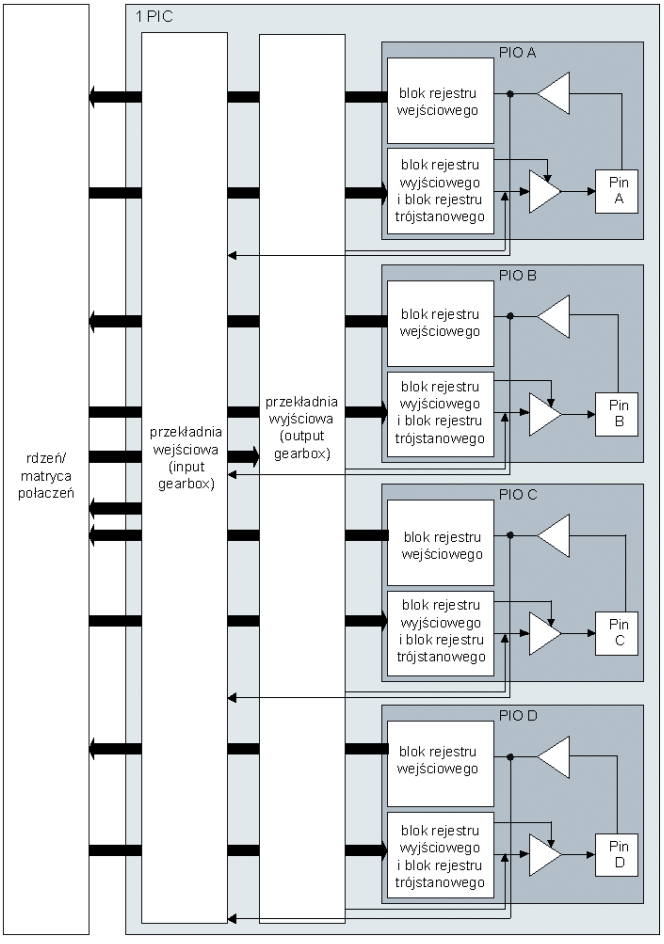
Warto tutaj wspomnieć, że ogólna architektura układów PLD rodziny MachXO2 jest bardzo zbliżona do rodziny XP2 – typowych, „dużych" układów FPGA również oferowanych przez Lattice. Struktura jednostek PFU oraz bloków *slice* jest dla tych dwóch rodzin identyczna. Ponadto budowa samego bloku *slice* wykazuje również pewne podobieństwa do analogicznych bloków *silce* układów FPGA Xilinx rodziny Spartan-2



Rysunek 1. Ogólny schemat architektury układu MachXO2-1200



Rysunek 2. Struktura bloku *slice*



Rysunek 3. Grupa czterech bloków PIO zorganizowana jako komórka PIC

i Spartan-3 (w przypadku układów Spartan-3 sposób połączenia dwóch tablic LUT z dwoma przerzutnikami jest nieco bardziej rozbudowany niż dla rodziny MachXO2, czy XP2).

Programowalne komórki wejścia-wyjścia

Układy z rodziny MachXO2 zapewniają wsparcie dla implementacji szybkich interfejsów przesyłania danych dla aplikacji wykorzystujących techniki SDR (*Single Data Rate*) oraz DDR (*Double Data Rate* – transmisja danych odbywa się z wykorzystaniem obydwu zboczy sygnału zegarowego). Dzięki programowalnym blokom PIO możliwa jest np. implementacja interfejsów do pamięci DDR, DDR2, LPDDR SDRAM, bez wykorzystywania logiki dostępnej w jednostkach PFU.

Logika programowalna związana z wejściami i wyjściami określana jest jako komórka PIO (*Programmable Input/Output*). Pojedyncza komórka PIO jest połączona przez bufory wejścia/wyjścia z zewnętrznymi końcówkami układu programowalnego. Układy MachXO2 zawierają grupę 4 komórek PIO, która nazywana jest programowalną komórką wejścia/wyjścia PIC (*Programmable Input/Output Cell*). Na rysunku 3. pokazano ogólny schemat komórki PIC.

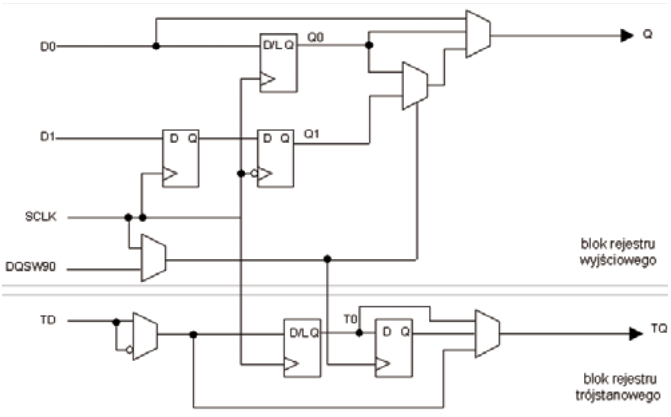
Implementacja szybkich interfejsów DDR realizowana jest poprzez wbudowane tzw. przekładnie (*gearing logic*), znajdujące się w komórkach PIO. Zastosowanie przekładni znajdujących się w blokach wejścia/wyjścia zmniejsza wymagania odnośnie do wydajności układu PLD (FPGA), przy pomocy którego realizowany jest szybki interfejs danych.

W układach MachXO2 dostępne są cztery przełożenia (*gearing ratio*), zależnie od lokalizacji banku wejścia/wyjścia oraz ilości dostępnych zasobów logicznych układu: x1, x2, x4 oraz 7:1 (wykorzystywany w aplikacjach wyświetlaczy wideo lub jako serializer – deserializer danych). Zapewnienie istnienia takich przełożeń realizowane jest za pomocą trzech typów komórek PIO: podstawowej, komórki pamięci PIO oraz komórki wideo PIO. Ogólnie, pierwsze dwa typy komórek zawierają trzy bloki: blok rejestru wejściowego, blok rejestru wyjściowego i blok rejestru trójstanowego (**rysunek 3**). Dla przykładu, na **rysunku 4** i **rysunku 5** pokazano strukturę komórki wyjściowej pamięci PIO oraz strukturę przekładni wejściowej.

Sprzętowy blok funkcji wbudowanych

Wszystkie układy rodziny MachXO2 mają sprzętowy blok funkcji wbudowanych EFB. W ramach tego bloku dostępne są dwa kontrolery magistrali I²C, jeden kontroler SPI oraz układ czasowo-licznikowy (**rysunek 6**). Dodatkowo poprzez blok EFB można zmieniać ustawienia pętli PLL w czasie pracy systemu, uzyskać dostęp do pamięci konfiguracji FLASH oraz pamięci użytkownika UFM, a także sterować trybami oszczędzania energii. Komunikacja z blokiem EFB odbywa się przez standardowy interfejs WISHBONE, który zostanie krótko omówiony w dalszej części kursu.

Każdy z dwóch dostępnych, w ramach bloku EFB, kontrolerów I²C charakteryzuje się następującymi właściwościami: praca zarówno jako układ master, jak i slave, dostępne adresowanie 7- i 10-bitowe, wsparcie



Rysunek 4. Wyjściowa komórka PIO dla aplikacji pamięci DDR

Tabela 1. Rodzina układów MachXO2

Zasoby \ Typ układu	XO2-256	XO2-640	XO2-640U	XO2-1200	XO2-1200U	XO2-2000	XO2-2000U	XO2-4000	XO2-7000
LUT	256	640	640	1280	1280	2112	2112	4320	6864
RAM (Kbit)	2	5	5	10	10	16	16	34	54
EBR SRAM (Kbit)	0	18	64	64	74	74	92	92	240
Liczba bloków EBR w SRAM (9 Kbit/blok)	0	2	7	7	8	8	10	10	26
UFM (Kbit)	0	24	64	64	80	80	96	96	256
Liczba PLL	0	0	1	1	1	1	2	2	2

dla arbitrażu w sytuacji, gdy na magistrali pracuje więcej układów typu master, wydłużanie okresu sygnału zegarowego w przypadku współpracy z wolniejszymi układami I²C, maksymalna szybkość przesyłu danych wynosząca 400 kHz. Pierwszy kontroler I²C współpracuje z końcówkami zewnętrznymi układu PLD, podczas gdy linie magistrali drugiego kontrolera mogą być dowolnie konfigurowane przez użytkownika. Kontroler SPI również może pracować w trybach master oraz slave, zapewnia w pełni dupleksowy transfer danych, umożliwia pracę w trybie z sygnalizacją błędu (możliwość zgłoszenia przerwania do CPU), ma podwójnie buforowany rejestr danych, dysponuje możliwością programowania polaryzacji oraz fazy sygnału zegarowego oraz umożliwia transfer danych z bitem LSB lub MSB jako pierwszym. Z kolei układ czasowo-licznikowy może m.in. spełniać funkcje: układu dozorującego typu watchdog, szybkiego generatora PWM, źródła sygnału zegarowego o programowanej częstotliwości oraz źródła sygnału przerwania (przeładowanie licznika, osiągnięcie określonej wartości zliczeń).

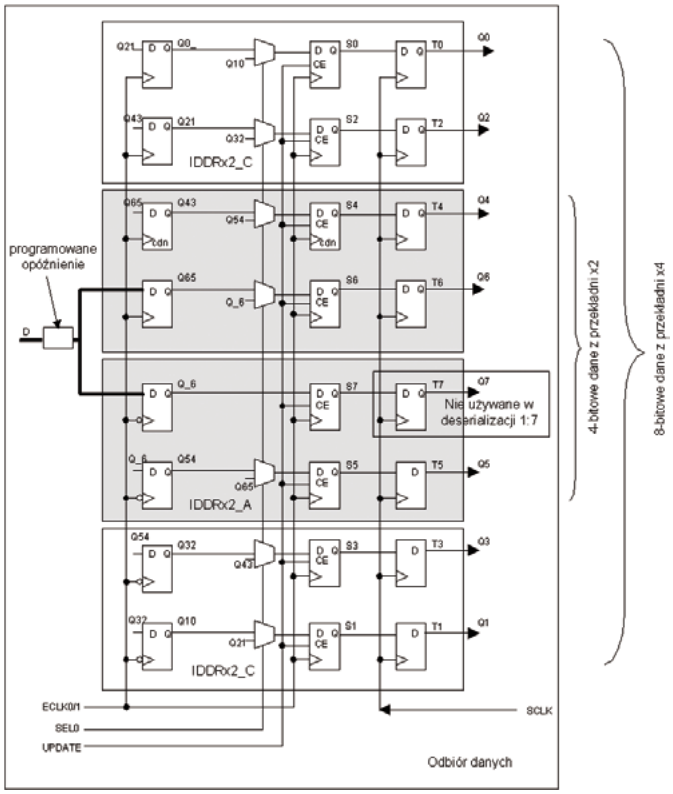
Inne właściwości układów MachXO2

Układy MachXO2 charakteryzują się również możliwością dołączenia do systemu bez wyłączania zasilania (*hot socketing, hot-swapping*). Wynika to ze szczególnego sposobu projektowania zapewniającego przewidywalne zachowanie układu w czasie włączenie i zaniku zasilania. Inną cechą jest wbudowany oscylator CMOS, który może być źródłem częstotliwości taktującej. Ma on dokładność na poziomie 5% i częstotliwość wybieraną spośród kilkudziesięciu predefiniowanych wartości (inną częstotliwość można uzyskać, wykorzystując pętlę PLL). Dodatkowo każdy egzemplarz układów MachXO2 został wyposażony w unikalny 64-bitowy numer seryjny (*TraceID*), przy czym 8 bitów tego kodu może być zaprogramowanych przez użytkownika. Odczyt kodu możliwy jest poprzez blok EFB i interfejs Wishbone.

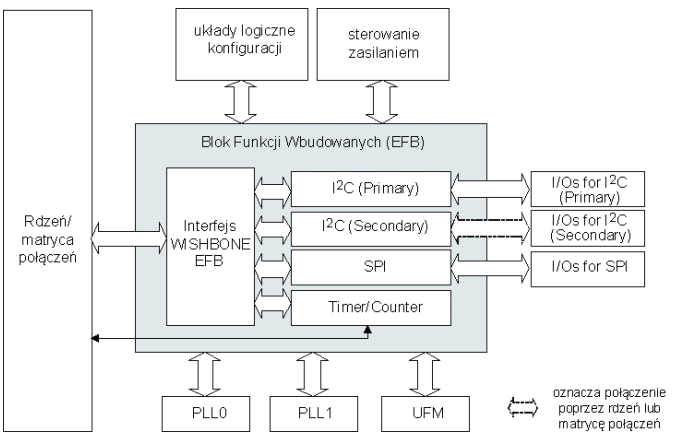
Konfiguracji układów PLD MachXO2 można dokonać na kilka sposobów: przez wewnętrzne (przez blok EFB) zaprogramowanie pamięci Flash, przez interfejs JTAG oraz przez interfejsy SPI i I²C bloku EFB. Podczas procesu rozruchu systemu (włączenia zasilania) lub w odpowiedzi na polecenie przesłane interfejsem JTAG zawartość pamięci konfiguracji Flash jest kopiowana do komórek pamięci SRAM odpowiadającej bezpośrednio za konfigurację logiki programowalnej układu PLD. Poprzez interfejs JTAG możliwy jest dostęp zarówno do pamięci konfiguracji Flash, jak i do pamięci SRAM. Zapewniona jest również ochrona przed nieuprawnionym odczytem obydwu pamięci (*security and one-time programmable mode*).

Lattice oferuje także opracowaną przez siebie technologię TransFR (*Transfer Field Reconfiguration*) umożliwiającą wymianę konfiguracji logiki bez przerywania pracy systemu (w rzeczywistości technologia minimalizuje czas i skutki przerwania pracy systemu). Wykorzystuje ona fakt, że pamięć konfiguracji Flash może być programowana w dowolnym momencie przez zewnętrzny interfejs JTAG. Przed skopiowaniem zawartości Flash do komórek pamięci SRAM, wejścia/wyjścia układu PLD przełączane są w bezpieczny stan zdefiniowany przez użytkownika, następnie kopiowana jest zawartość pamięci (odbywa się właściwa konfiguracja) i system wznawia pracę.

Rodzina MachXo2 obejmuje w sumie 6 układów o pojemnościach od 256 do 6864 tablic LUT (**tabela 1**), produkowanych w oparciu o proces technologiczny 65 nm. Każdy z układów jest dostępny w dwóch wersjach: o bardzo niskim poborze mocy (ZE) oraz o wysokiej wydajności (HC, HE). Układy ZE i HC wymagają zasilania rdzenia napięciem 1,2 V, podczas gdy układy HE mają wewnętrzny regulator napięcia i pracują z napięciem rdzenia 2,5 V lub 3,3 V. Statyczny pobór mocy przez układy MachXO2 serii ZE może sięgać tylko 19 μW (mikrowatów)! Oprócz całej gamy obudów BGA, wszystkie układy MachXO2 dostępne są również w przyjaznych dla montażu obudowach TQFP.



Rysunek 5. Wejściowa przekładnia x2/x4 i 7:1



Rysunek 6. Blok funkcji wbudowanych EFB

Zestaw ewaluacyjny MachXO2 Pico

Wraz z wprowadzeniem na rynek układów MachXO2, firma Lattice zaoferowała również dwa zestawy ewaluacyjne zbudowane na bazie tych układów. Jednym z takich zestawów jest prezentowany tutaj – *MachXO2 Pico Developmnent Kit*. Jak sugeruje nazwa, jest to stosunkowo nieskomplikowany zestaw, niemniej jednak w pełni umożliwiający praktyczne zapoznanie się z najważniejszymi cechami układów MachXO2. Zasadniczymi komponentami zestawu są:

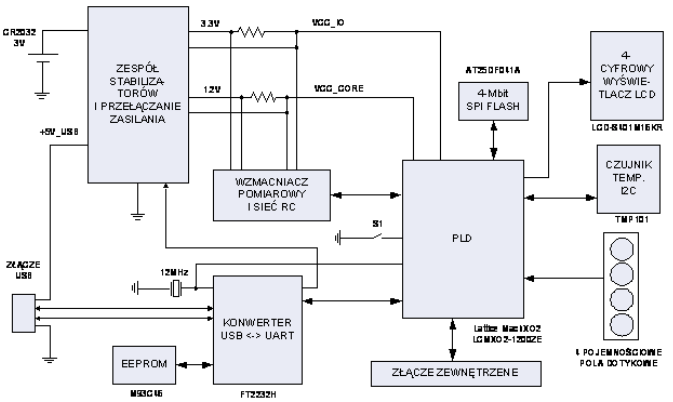
– układ MachXO2 LCMXO2-1200ZE w obudowie csBGA o 132 wyprowadzeniach,
– 4 M-bitowa pamięć Flash SPI,
– układ pomiaru poboru mocy z przetwornikiem delta-sigma,
– czujnik temperatury I²C,
– 4-rocyfrowy wyświetlacz LCD,
– 4 pojemnościowe pola dotykowe,
– złącze zewnętrzne z 32 wyprowadzonymi liniami we/wy układu PLD,
– konwerter RS-232/USB.

Uproszczony schemat blokowy zestawu pokazano na **rysunku 7**. Zestaw zasilany jest poprzez złącze USB lub, w przypadku braku zasilania głównego – za pomocą dodatkowej baterii 3 V. W obwód zasilania rdzenia oraz buforów wejścia/wyjścia układu PLD są włączone rezystory pomiarowe, za pomocą których jest dokonywany pomiar pobieranego prądu. Spadek napięcia na tych rezystorach, wzmacniany przez wzmacniacze pomiarowe, trafia do przetwornika delta-sigma zbudowanego z użyciem logiki programowalnej układu PLD oraz zewnętrznej sieci elementów RC. Zasadniczym elementem pozwalającym na wizualizację działania zestawu jest 4-cyfrowy analogowy wyświetlacz LCD. Zestaw nie zawiera diod LED ogólnego przeznaczenia, ale można je dołączyć za pośrednictwem zewnętrznego złącza o 32 wyprowadzeniach. Zamiast typowych przycisków w zestawie zastosowano pojemnościowe pola dotykowe, które na płytce drukowanej stanowią pola miedzi o kształcie okręgów, pokryte maską przeciwlutowniczą. Rozmieszczenie najważniejszych elementów na płytce drukowanej zestawu przedstawia **rysunek 8**.

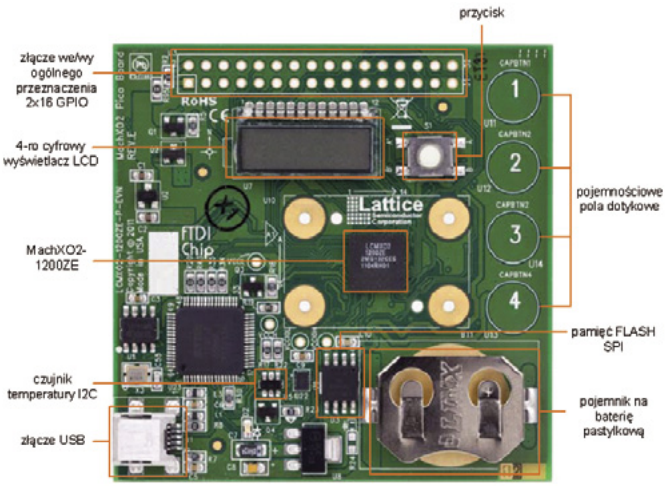
Programowanie (konfiguracja) układu PLD odbywa się przez złącze USB za pośrednictwem wbudowanego konwertera RS232/USB oraz oprogramowania (ispVM System) dostarczanego przez Lattice. Przez konwerter RS232/USB możliwa jest również komunikacja zestawu z komputerem PC za pośrednictwem zwykłego portu UART.

Przygotowanie zestawu do pracy w środowisku Windows

Współpraca zestawu MachXO2 Pico z komputerem PC i narzędziami projektowymi wymaga jedynie zainstalowania odpowiednich sterowników. Są one dostępne do pobrania na stronie producenta oraz znajdują się w materiałach dodatkowych do niniejszego kursu. Kolejność czynności instalacyjnych jest następująca: najpierw należy podłączyć zestaw do komputera PC za pomocą przewodu USB, a następnie w oknie instalacji nowego sprzętu wybrać opcję instalacji sterownika z określonej lokalizacji, wskazując folder, w którym znajdują się pobrane pliki sterownika. W momencie nawiązania połączenia poprzez port USB na płytce zestawu zapala się niebieska dioda LED, sygnalizująca załączenie zasilania układu PLD i peryferii.



Rysunek 7. Uproszczony schemat blokowy zestawu MachXO2 Pico



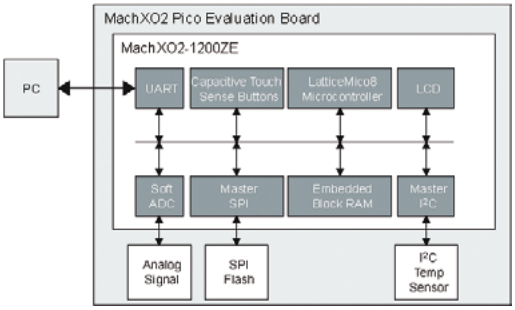
Rysunek 8. Widok płytki zestawu z zaznaczonym rozmieszczeniem najważniejszych elementów

Zestaw MachXO2 Pico dostarczany jest z zaprogramowanym (skonfigurowanym) układem PLD, zawierającym demonstracyjną aplikację, ilustrującą sposób działania ważniejszych komponentów zestawu (odpowiedni plik konfiguracyjny w formacie JEDEC, a także pełne kody źródłowe aplikacji, można pobrać ze strony producenta). Aplikacja ta jest w istocie tzw. zintegrowanym systemem cyfrowym SoC (*System On-Chip*), którego ogólny schemat blokowy pokazano na **rysunku 9**. Sercem systemu jest prosty, 8-bitowy „miękki" mikrokontroler LatticeMico8, którego opis można znaleźć np. w „Elektronice Praktycznej" 12/2005. Mikrokontroler współpracuje z wbudowaną pamięcią RAM, blokami sterownika wyświetlacza LCD, portu UART, odczytu stanu pojemnościowych pól dotykowych, przetwornika analogowo-cyfrowego oraz poprzez blok EFB z kontrolerami SPI oraz I²C. Jako elementy zewnętrzne, do systemu SoC dołączona jest pamięć Flash SPI, czujnik temperatury I²C, pojemnościowe pola dotykowe, wyświetlacz LCD, a także doprowadzone są analogowe sygnały ze wzmacniaczy pomiarowych i drabinki RC.

Aplikacja demonstracyjna ilustruje zastosowanie mikrokontrolera LatticeMico8 wraz z dołączonymi peryferiami jako prostego systemu o niskim poborze mocy, pozwalającego na pomiar napięci/prądu, pomiar temperatury otoczenia, zapis i odczyt danych do/z pamięci nieulotnej, komunikację poprzez port UART, użycie pojemnościowych pól dotykowych i wyświetlacza LCD. Interakcja użytkownika z systemem możliwa jest za pomocą pojemnościowych pól dotykowych lub typowego programu terminalowego (np. HyperTerminal) na komputerze PC.

Niezbędny do skonfigurowania HyperTerminala numeru portu szeregowego można odczytać, posługując się Menedżerem Urządzeń, w zakładce „Porty (COM i LPT)", pod pozycją „USB Serial Port". Parametry transmisji są następujące: szybkość 115200 bit/s, 8 bitów danych, brak bitu parzystości, brak sprzętowej kontroli przepływu danych, 1 bit stopu.

Po włączeniu zasilania (podłączeniu zestawu do złącza USB komputera PC) na wyświetlaczu LCD pojawiają się symbole tworzące napis „PICO". Jeżeli



Rysunek 9. Zintegrowany system cyfrowy aplikacji demonstracyjnej

na komputerze PC uruchomiony jest program HyperTerminal, wówczas naciśnięcie na klawiaturze PC klawisza „m” powoduje wyświetlenie w oknie HyperTerminalu, menu pokazanego na **rysunku 10**. Wciskając teraz odpowiednie klawisze, wyszczególnione w menu, można wykonać przypisane do nich operacje, np.: odczyt poboru prądu, odczyt temperatury itp. Podobny efekt można uzyskać, przykładając palec do jednego z czterech pól dotykowych zestawu, przy czym wartości zmierzonego prądu lub temperatury pojawiają się tylko na wyświetlaczu LCD i nie są wysyłane łączem szeregowym. Jeżeli w ciągu kilkunastu sekund system zanotuje brak jakiejkolwiek aktywności użytkownika, to automatycznie wprowadzany jest tryb o bardzo niskim poborze mocy (gaśnie wyświetlacz LCD). Wyjście z tego trybu i powrót do normalnej pracy możliwy jest po wciśnięciu przycisku znajdującego się na płytce zestawu. Naciśnięcie tego przycisku w czasie normalnej pracy wprowadza system w stan o bardzo niskim poborze mocy.

Środowisko projektowe Lattice Diamond

Synteza i implementacja projektów wykorzystujących układy programowalne wymaga odpowiedniego wsparcia ze strony narzędzi programowych. W przypadku układów FPGA (w tym również rodziny Mach XO2), Lattice oferuje nowe, flagowe środowisko projektowe o nazwie *Diamond*. Jest ono dostępne zarówno dla systemów Windows, jak i Linux. Oprogramowanie można pobrać ze strony producenta (*http://www.latticesemi.com*) wraz z dwoma rodzajami licencji na użytkowanie: bezpłatną i płatną. W tym pierwszym przypadku korzystanie z oprogramowania Diamond jest całkowicie darmowe, jednak wspierane są tylko niektóre rodziny układów Lattice, takie jak LatticeXP, LatticeECP &EC, LatticeECP2, LatticeXP2, a także MachXO i MachXO2.

Lattice Diamond jest kompletnym środowiskiem projektowym, za pomocą którego można przeprowadzić wszystkie fazy projektowania, począwszy od specyfikacji projektu aż do programowania (konfiguracji) docelowego układu programowalnego. W szczególności wspierane są takie procesy jak: specyfikacja projektu (możliwa jest mieszana specyfikacja z wykorzystaniem języków HDL Verilog oraz VHDL, listy połączeń EDIF i edytora schematów), synteza logiczna, implementacja, symulacja, programowanie a także analiza działania systemu uruchomionego w docelowym układzie programowalnym (*on-chip debug hardware analysis*). Jako narzędzie syntezy w ramach środowiska Lattice Diamond oferowany jest wiodący na rynku produkt firmy Synopsys: *Synopsys Synplify Pro* w wersji dla układów Lattice. Dla rodziny MachXO oraz MachXO2 Lattice opracował również własny syntezer (*Lattice Synthesis Engine*), który stanowi alternatywną opcję wyboru narzędzia syntezy dla tej rodziny układów. Z koli jako narzędzie symulacji wykorzystywany jest produkt firmy Aldec o nazwie *Active-HDL Lattice Edition II*.

Pierwszy projekt

Pokażemy teraz szczegółowo, w jaki sposób rozpocząć projektowanie z zestawem MachXO2 Pico, wykorzystując środowisko Lattice Diamond. W naszych przykładowych projektach będziemy stosować język opisu sprzętu Verilog. Chociaż specyfikację projektowanego systemu można również, w znacznej części, wykonać stosując edytor schematów, jednak w przypadku np. użycia bloku funkcji wbudowanych EFB jest to praktycznie nieopłacalne. Tym bardziej że aplikacja generatora wirtualnych komponentów *IPexpress* dla bloku EFB wytwarza kod, stosując jeden z języków HDL: Verilog lub VHDL. Implementacja interfejsu WISHBONE, za pomocą którego odbywa się komunikacja z blokiem EFB, jest co prawda możliwa przy użyciu edytora schematów, jednak znacznie prościej i szybciej można to zrobić, wykorzystując język opisu sprzętu. Oczywiście w przypadku bardzo prostych układów cyfrowych można nadal posługiwać się edytorem schematu lub zastosować go do specyfikacji części większego projektu.

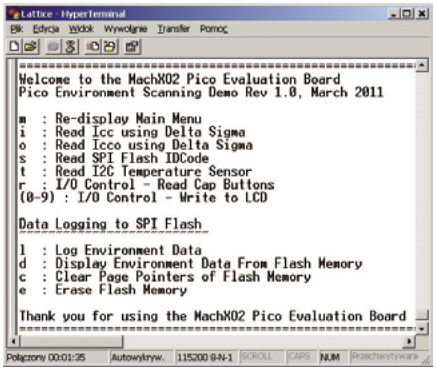
lNa początku pokażemy bardzo prosty projekt, którego zadaniem będzie wytworzenie, na jednym z wyjść układu PLD, fali prostokątnej o częstotliwości około 1 Hz. Przebieg ten można wykorzystać do wysterowania diody LED, która po zaprogramowaniu układu PLD powinna cyklicznie zapalać się i gasnąć. Ponieważ w zestawie MachXO2 nie przewidziano diody LED ogólnego przeznaczenia, aby sprawdzić efekt działania naszego projektu, potrzeba będzie taką diodę dołączyć do zewnętrznego złącza we/wy zestawu lub zamiast LED zastosować np. sondę oscyloskopu.

Na **rysunku 11** pokazano wygląd głównego okna aplikacji Lattice Diamond tuż po uruchomieniu. Warto zauważyć, że z prawej strony okna zakładki *Start Page* dostępne są, w formacie PDF, instrukcje użytkownika dotyczące sposobu wykorzystania środowiska Lattice Diamod, opisy poszczególnych opcji itp.

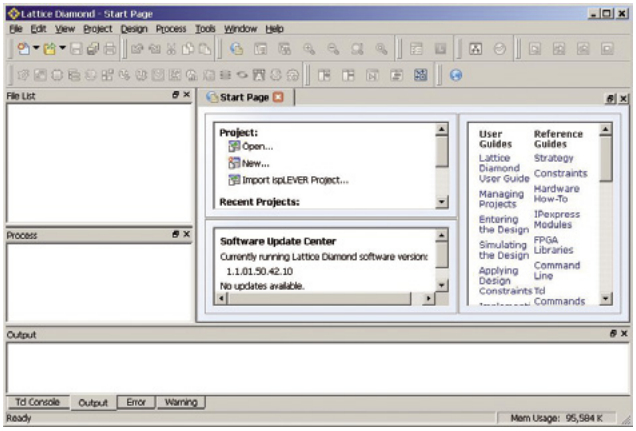
Aby utworzyć nowy projekt, należy z menu *File* wybrać opcję *New* a następnie *Project* lub w zakładce *Start Page*, poniżej etykiety *Project*, kliknąć opcję *New*. Otworzone zostanie okno kreatora nowego projektu, w którym, jako jeden z pierwszych kroków, będziemy musieli wybrać lokalizację dla plików naszego projektu oraz nazwę projektu. Po wciśnięciu przycisku *Next* otrzymamy możliwość dodania istniejących plików źródłowych do naszego projektu. Jeżeli nie mamy takich plików i chcemy dopiero je utworzyć, wybieramy opcję *Next*. Kolejnym – ostatnim krokiem jest wybór docelowego układu programowalnego. W przypadku zestawu MachXO2 Pico należy wybrać parametry pokazane na **rysunku 12**. Wygląd okna aplikacji Lattice Diamond, po utworzeniu projektu o nazwie „flashing_led”, pokazano na **rysunku 13**.

Teraz do projektu można dodawać pliki źródłowe. Można to zrobić, klikając prawym klawiszem, zaznaczony na rysunku 13, folder *Input Files* i wybierając opcję *Add -> New File*. Analogiczny efekt uzyskamy, wybierając z głównego menu aplikacji następujące opcje: *File -> New -> File*. W odpowiedzi na wybraną opcję otwarte zostanie okno z **rysunku 14**. W rozważanym przykładzie nasz plik źródłowy z kodem w języku Verilog nazwaliśmy „top”, sugerując, że jest to nadrzędny moduł w hierarchii projektu. Po naciśnięciu przycisku *New* w głównym oknie aplikacji Diamond, w zakładce o takiej samej nazwie jak właśnie utworzony plik źródłowy (w tym przypadku „top.v”), zostanie otwarte okno edycyjne, gdzie można wprowadzać kod źródłowy.

Na **listingu 1** pokazano kompletny kod źródłowy modułu, który realizuje zadanie sterowania diody LED z częstotliwością około 1 Hz. Moduł ma tylko dwa porty: wejściowy o nazwie RST (zerowanie) i wyjściowy o nazwie LED (stero-

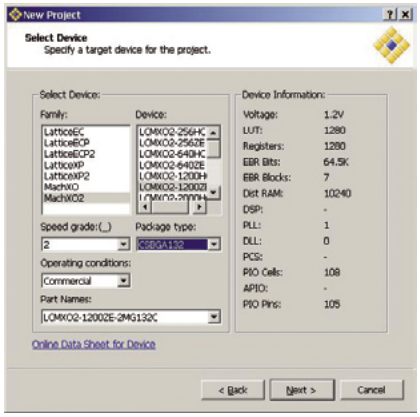


Rysunek 10. Widok okna programu HyperTerminal

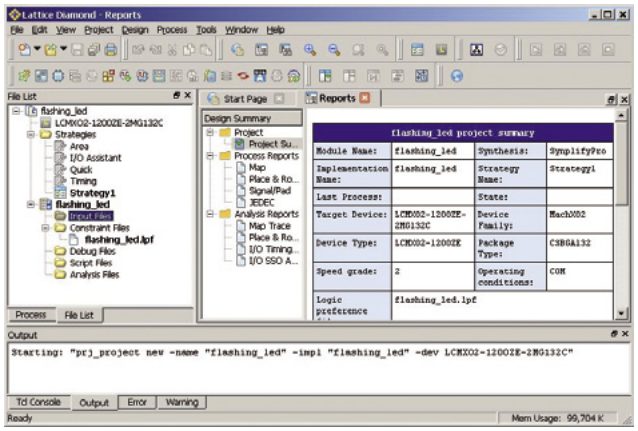


Rysunek 11. Widok okna aplikacji Lattice Diamond po uruchomieniu

wanie diodą). Jako źródło sygnału zegarowego wykorzystaliśmy wbudowany w układzie MachXO2, oscylator COMS. Szczegółowo sposób wykorzystania tego oscylatora opisuje nota aplikacja Lattice TN1199 (*MachXO2 sysCLOCK PLL Design and Usage Guide*), dostarczając kompletny szablon utworzenia instancji oscylatora w językach Verilog i VHDL. Częstotliwość oscylatora może być jedną z kilkudziesięciu wybranych wartości (szczegóły – w wymienionej nocie aplikacyjnej), z których najmniejsza to 2,08 MHz, zastosowana w naszym wypadku.

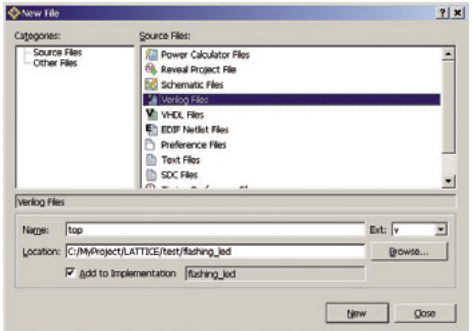


Rysunek 12. Wybór docelowego układu programowalnego



Rysunek 13. Widok okna aplikacji Diamond po utworzeniu projektu

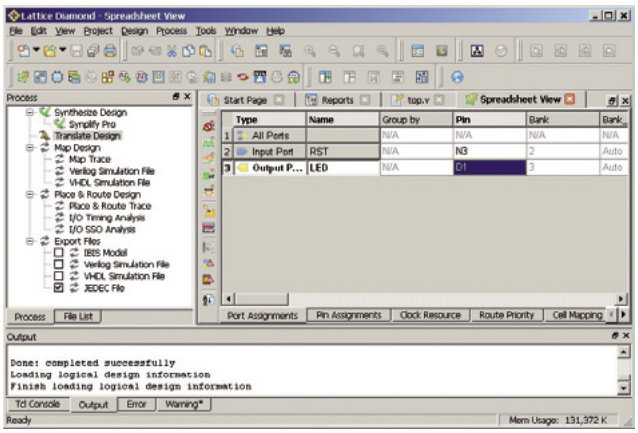
Mając gotowy kod opisujący działanie projektowanego systemu, możemy przystąpić do wykonania operacji syntezy logicznej i implementacji projektu. Wcześniej można jeszcze z menu *Project -> Synthesis Tool* wybrać jedno z trzech dostępnych narzędzi syntezy (Synplify Pro, Precision, Lattice LSE) oraz określić parametry procesu poprzez edycję tzw. strategii (w lewej części okna aplikacji Diamond zakładka *File list*, folder *Strategies*, wytłuszczona opcja *Stategy1* – **rysunek 14**). Teraz w zakładce *Process* wystarczy dwukrotnie kliknąć opcję *Synthesize Design*, aby wykonać operację syntezy projektu lub od razu opcję translacji projektu *Translate Design*. Na tym etapie można przypisać fizyczne numery końcówek układu programowalnego do portów wejścia/wyjścia naszego pliku głównego (nadrzędnego w hierarchii) projektu. Najwygodniej jest to zrobić, wykorzystując wbudowane narzędzie arkusza kalkulacyjnego – menu *Tools -> Spreadsheet View*, zakładka *Port Assignments* – jak pokazano to na **rysunku 15**. W naszym przypadku port RST przypisano do wyprowadzenia N3 (przycisk znajdujący się na płytce zestawu), a port LED do wyprowadzenia D1 (końcówka nr 29 zewnętrznego złącza) wybranego układu programowalnego.



Rysunek 14. Okno wyboru pliku źródłowego

Listing 1. Kod modułu w języku Verilog opisującego działanie przykładowej aplikacji

```
module top (input RST,
            output LED);
defparam OSCH_inst.NOM_FREQ = "2.08"; // częstotliwość
w MHz
// instancja oscylatora
OSCH OSCH_inst(.STDBY(1'b0), // 0 - oscylator włączony
               .OSC(CLK),
               .SEDSTDBY()); // wyjście - niewykorzystane
reg [20:0] cntr;
// prosty licznik binarny
always @(posedge CLK, negedge RST)
if(~RST) cntr<=0;
else cntr<=cntr+1;
// najmłodszy bit licznika steruje diodą LED
assign LED=cntr[20];
endmodule
```

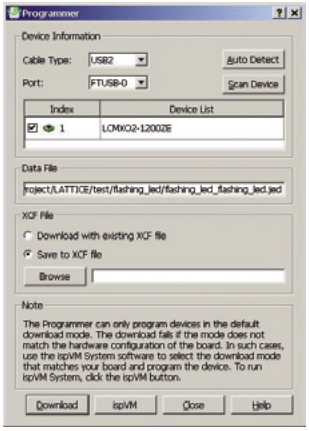


Rysunek 15. Przypisanie numerów końcówek do portów wejścia/wyjścia

Przesuwając w prawo belkę poziomą, można uzyskać dostęp do opcji buforów wejścia wyjścia, takich jak np. standard, w którym ma pracować port, opcje podciągania (*pull mode*), szybkość narastania napięcia wyjściowego (*slew rate*) itp. Analogiczną operację przypisania numerów wyprowadzeń do portów projektu można przeprowadzić, edytując bezpośrednio plik wymuszeń projektanta (zakładka *File List*, folder *Constraint Files*, plik *flashing_led.lpf*).

Dokończenie procesu implementacji można wykonać, klikając bezpośrednio polecenie *Export Files* w zakładce *Process*, zaznaczając jednak wcześniej opcję eksportu pliku do formatu JEDEC. W przypadku gdy mamy już zdefiniowany plik wymuszeń projektanta z przypisaniami wyprowadzeń do portów, dokonaliśmy zmiany w kodzie źródłowym projektu i chcemy ponownie przeprowadzić proces implementacji, wówczas wystarczy wybrać tylko polecenie *Export Files*.

Ostatnim etapem jest programowanie docelowego układu PLD. Z menu *Tools* należy w tym celu wybrać opcję *Programmer* lub kliknąć odpowiednią ikonę na pasku narzędzi. Otwarte zostanie okno pokazane na **rysunku 16**. W przypadku pierwszego uruchomienia programatora konieczny będzie wybór typu przewodu programującego (*Cable type*) i portu (*Port*). Parametry te można wybrać z rozwijanej listy lub wykonać autodetekcję i skanowanie w poszukiwaniu podłączonych układów PLD. Istotne jest również zaznaczenie opcji *Save to XCF file* w sekcji *XCF File*. Załadowanie konfiguracji do pamięci Flash układu PLD następuje po wykonaniu polecenia *Download*.



Rysunek 16. Okno programatora

Zbigniew Hajduk
zhajduk@kia.prz.edu.pl