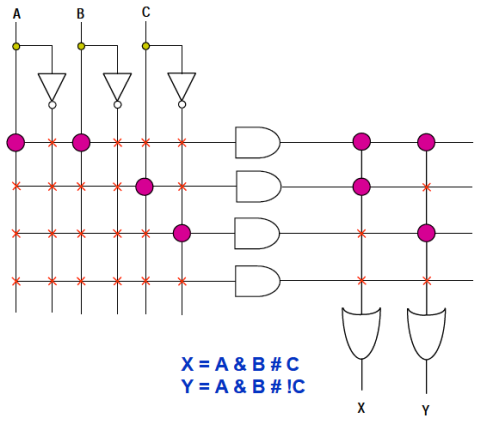
mlazoryszczak@wi.zut.edu.pl

LABORATORIUM nr 5

Temat: Realizacja mikroprocesora w układach CPLD

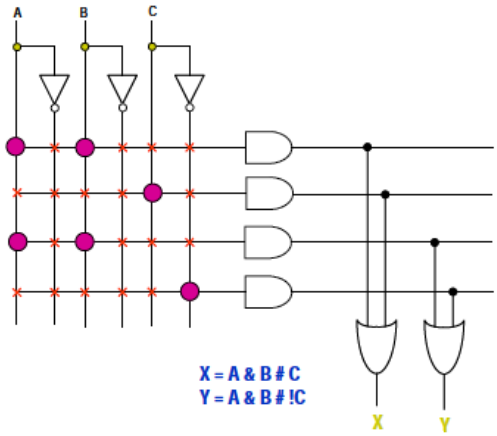
1. UKŁADY REKONFIGUROWALNE

Układy programowalne (ang. *Programmable Logic Devices*, PLD) stanowiły atrakcyjną alternatywę dla standardowych układów logicznych już od chwili pojawienia się na rynku w latach siedemdziesiątych XX w. Początkowo ich możliwości ograniczały zastosowanie do zastępowania standardowych układów logicznych małej skali integracji np. serii 74xx. W najprostszy sposób budowę pierwszych układów reprogramowalnych można sprowadzić do matrycy podstawowych bramek (AND, OR, NOT), w ramach której możliwe było konfigurowanie połączeń pomiędzy poszczególnymi bramkami. Jedną z takich struktur – PLA (ang. *Programmable Logic Array*) przedstawiono na rys. 1. Funkcje realizowane z wykorzystaniem takich struktury ograniczone były tylko zasobami wewnętrznymi danego układu.



Rys. 1. Struktura typu PLA [5].

Alternatywną strukturą jest PAL (ang. *Programmable Array Logic*). W przeciwieństwie do poprzedniej struktury PAL charakteryzuje się stałą kombinacją połączeń na wejściach bramek OR. Ogranicza to z jednej strony liczbę możliwych do uzyskania funkcji logicznych, uniemożliwiając wielokrotne wykorzystanie tych samych termów iloczynowych, ale z drugiej strony zapewnia potencjalnie większą szybkość działania.

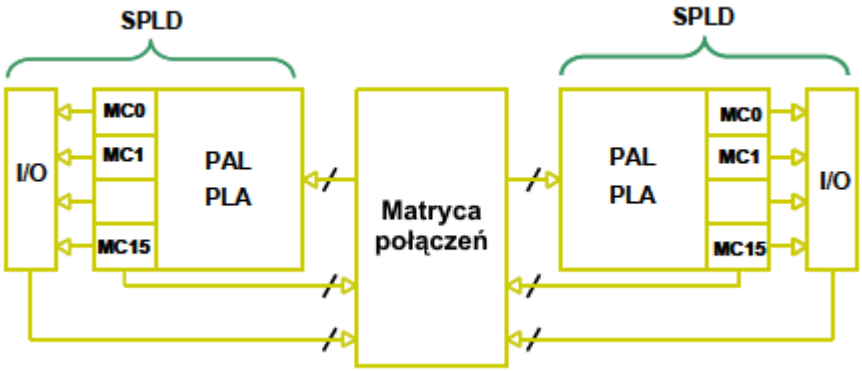


Rys. 2. Struktura typu PAL [5].

Wraz z rozwojem technologii i wzrostem skali integracji układów scalonych pojawiły się nowe rozwiązania układowe i zwiększyły się możliwości układów reprogramowalnych. Obecnie układy te są stosowane szeroko w różnych dziedzinach takich jak cyfrowe przetwarzanie sygnałów, systemy wbudowane, komputery i superkomputery itp. W ramach systemów wbudowanych na uwagę zasługuje możliwość implementacji dowolnego mikroprocesora (mikrokontrolera) w postaci tzw. modułu IP core (ang. *Intellectual Property core*) i wykorzystania go w specyficznym projekcie użytkownika.
Celem niniejszego ćwiczenia jest zapoznanie się z technologią reprogramowalną w zakresie obejmującym implementację tzw. *soft* procesora (procesora „programowego"), istniejącego pierwotnie w postaci modelu, opisanego za pomocą języków typu HDL (ang. Hardware Description Language) takich jak VHDL czy Verilog. Przewaga implementacji procesora w układach reprogramowalnych nad tradycyjnymi procesorami polega między innymi na elastyczności tego pierwszego sposobu. W przypadku procesora programowego możliwa jest choćby zmiana struktury procesora i jej dopasowanie do aktualnych potrzeb. Dotyczy to zarówno zestawu instrukcji jak i modułów peryferyjnych (np. UART, timer itp.). Możliwa jest także integracja dodatkowych modułów mających na celu wspomaganie działania samego procesora pod kątem wydajności obliczeniowej. Ponadto procesor istniejący w postaci modelu może zostać zaimplementowany w dowolnej technologii i w dowolnym rodzaju układu, o ile pozwalają na to zasoby sprzętowe danej rodziny. Należy także zwrócić uwagę na fakt, iż projektant może stworzyć swój własny model procesora lub mikrokontrolera przeznaczonego do realizacji specyficznych zadań bądź może też skorzystać z niezwykle szerokiej gamy modeli mikroprocesorów istniejących na rynku w postaci układów specjalizowanych (ASIC) i zintegrować je w ramach pojedynczego układu scalonego razem z własnymi rozszerzeniami.
W ćwiczeniach, z racji posiadanej bazy sprzętowej, wykorzystywane są układy oraz oprogramowanie do projektowania firmy Xilinx. Dla porządku należy także wymienić innych producentów struktur reprogramowalnych takich jak Altera, Lattice, Actel, QuickLogic i inni. W dalszym ciągu opisane zostaną układy, które na zasadzie wyboru mogą być używane w trakcie ćwiczeń laboratoryjnych.

1.1. CPLD

Ewolucja podstawowych struktur programowalnych określanych wspólną nazwą SPLD (ang. *Simple Programmable Logic Devices*) doprowadziła do powstania układów typu CPLD (ang. *Complex Programmable Logic Devices*). Główna idea polegała na umieszczeniu kilku mniej złożonych struktur typu PAL/PLA w jednym układzie i umożliwieniu wzajemnych połączeń między nimi za pomocą tzw. matrycy połączeń (rys. 3).



Rys. 3. Architektura CPLD [5].

Reprezentantem układów CPLD firmy Xilinx jest rodzina CoolRunner II. Wg zapewnień producenta wśród zastosowań tej rodziny znajdują się komputery przenośne, zestawy telefoniczne, urządzenia typu PDA, tablety itp. Do głównych zalet, które mają zasadniczy wpływ na wymienione wyżej możliwości aplikacyjne należą:

- globalna matryca połączeń,
- bardzo niski pobór mocy,
- duża liczba portów We/Wy,
- deterministyczne przebiegi czasowe,
- zachowanie konfiguracji bez obecności zasilania.

Poszczególne wyprowadzenia pełniące funkcje portów We/Wy mogą być konfigurowane w szerokim zakresie do pracy z różnego rodzaju sygnałami oraz z różnymi napięciami zarówno w trybie wejścia jak i wyjścia. Wejścia mogą być skonfigurowane do pracy z napięciami 1,5 V; 1,8 V; 2,5 V oraz 3,3 V. Ponadto w celu ochrony przed szumem mogą być dołączone do przerzutnika Schmitta. Z kolei wyjścia mogą pracować w trybie bezpośrednim, jako wyjścia trójstanowe lub wyjścia z otwartym drenem (ang. *open drain*). Dodatkowo możliwy jest wybór prędkości narastania zboczy (SLOW/FAST).
Układy rodziny CoolRunner II wykonywane są kilku wielkościach, przy czym każdy układ może być umieszczony w różnych obudowach. Przegląd podstawowych parametrów układów rodziny CoolRunner II przedstawiono w tabeli 1. Wyróżniony w niej został układ XC2C256 w obudowie TQ144, który jest wykorzystywany w zestawie uruchomieniowym.

Tabela. 1: Przegląd rodziny CoolRunner II.

Cechy	**Symbol układu**					
	XC2C32A	**XC2C64A**	**XC2C128**	**XC2C256**	**XC2C384**	**XC2C512**
F_{SYSTEM} [MHz]	323	263	244	256	217	179
Maks. liczba We/Wy	33	64	100	184	240	270
Liczba banków We/Wy	2	2	2	2	4	4
Pobór mocy w trybie czuwania [µW]	28,8	30,6	34,2	37,8	41,4	45,0
Obudowa (wymiary [mm])	**Maks. liczba We/Wy dla użytkownika**					
QFG32 (5 x 5)	21					
VQ44 (12 x 12)	33	33				
PC44 (17,5 x 17,5)	33	33				
QF48 (7 x 7)		37				
CP56 (6 x 6)	33	45				
VQ100 (16 x 16)		64	80	80		
CP132 (8 x 8)			100	106		
TQ144 (22 x 22)			100	118	118	
PQ208 (30,6 x 30,6)				173	173	173
FT256 (17 x 17)				184	212	212
FG324 (23 x 23)					240	270

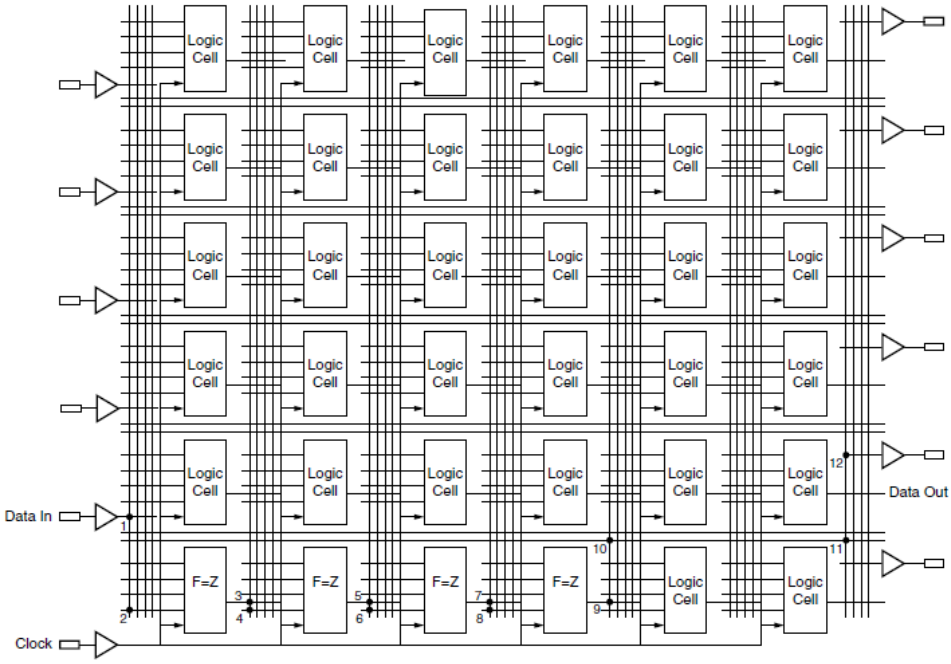
Grupowanie wyprowadzeń We/Wy w tzw. banki umożliwia dopasowanie poziomu napięcia do jednej z obsługiwanych wartości. W celu redukcji poboru mocy zaimplementowano cały szereg rozwiązań. Jednym z nich jest DataGATE czyli wczesne blokowanie sygnału za pomocą dedykowanej szyny. Zmniejszeniu mocy służy także rozwiązanie CoolCLOCK, polegające na wysterowaniu linii połączeniowych zegarem o mniejszej częstotliwości i podwojeniu częstotliwości zegara bezpośrednio w docelowym module.
Architetura CoolRunner II zapewnia także wysoki poziom bezpieczeństwa własności intelektualnej np. w celu ochrony konfiguracji przed kradzieżą. Ochrona realizowana jest na czterech poziomach za pomocą specjalnych bitów. Bity te mogą zostać usunięte jedynie podczas operacji kasowania całej konfiguracji wewnętrznej.

1.2. FPGA

Technologia FPGA (ang. *Field Programmable Gate Array*) w komercyjnej postaci powstała w 1985 roku za sprawą założycieli firmy Xilinx. Ogólna budowa układów FPGA opiera się na regularnej strukturze komórek logicznych oraz połączeń między nimi (rys. 4). Wszystkie zasoby są kontrolowane przez projektanta w znacznie większym stopniu niż układy CPLD. Podstawowe cechy układów FPGA są następujące:

- kanałowe połączenia,
- przebiegi czasowe wyznaczane w fazie rozlokowania,
- drobnoziarnista struktura,
- konieczność użycia skomplikowanych programowych narzędzi do projektowania.

Układy FPGA produkowane są w dwóch zasadniczych wariantach. Dominującym typem są układy oparte na pamięci typu SRAM. Oznacza to, że elementy pamięci są wykorzystywane zarówno do realizacji generatora funkcji logicznych w postaci tzw. tablic LUT (ang. *Look-Up Table*), jak i wszystkich programowalnych połączeń w układzie. Układy te wymagają dołączenia zewnętrznej pamięci, zawierającej konfigurację układu, ładowaną każdorazowo po załączeniu zasilania. Drugi typ – OTP (ang. *One Time Programming*) w sensie funkcjonalnym bardziej zbliżony jest do układów CPLD. W przypadku konieczności zmiany konfiguracji układu OTP FPGA niezbędna staje się wymiana całego układu.



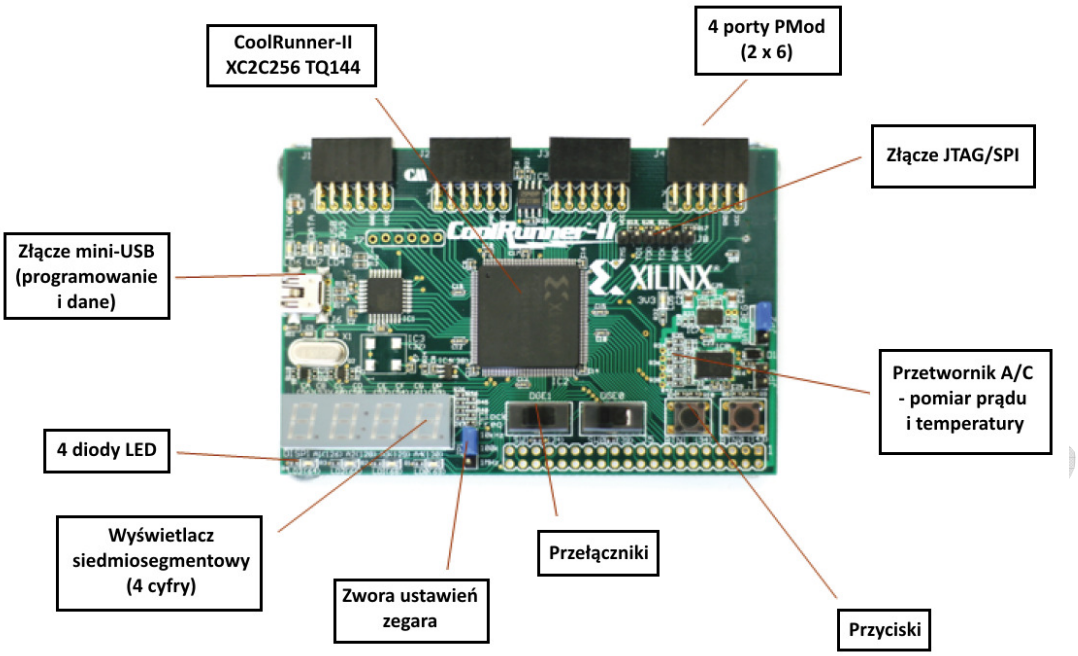
Rys. 4. Ogólna architektura FPGA [5].

W ramach technologii FPGA powstało znacznie więcej elementów oraz generacji niż w przypadku układów CPLD. Jedną z najbardziej popularnych rodzin jest seria Spartan 3. Obecna generacja rodziny Spartan oznaczona jest numerem 6. Obok układów Spartan rozwijana jest rodzina Virtex począwszy od generacji drugiej do wdrożonej pod koniec 2010 roku generacji Virtex-7. Układy FPGA oprócz struktury reprogramowalnej zawierają także często elementy o stałych funkcjach, nierzadko posiadają zaimplementowane kompletne rozwiązania procesorowe jak np. PowerPC czy też Dual ARM Cortex A9 MPCore w przypadku najnowszej rodziny Zynq.
W obecnej wersji ćwiczenia wykorzystane zostaną układy CPLD, dlatego też w celu bardziej szczegółowego zapoznania się z układami FPGA konieczne będzie skorzystanie z dostępnej literatury bądź dokumentacji technicznej (np. [5]).

2. ZESTAW URUCHOMIENIOWY COOLRUNNER II STARTER KIT

Zestaw CoolRunner II Starter Kit stanowi kompleksowe rozwiązanie uruchomieniowe (rys. 5). Płyta zasilana jest za pomocą złącza USB, które służy także do programowania układu CPLD oraz do ew. przesyłania danych między układem, a komputerem. Na płycie uruchomieniowej znajdują się także:

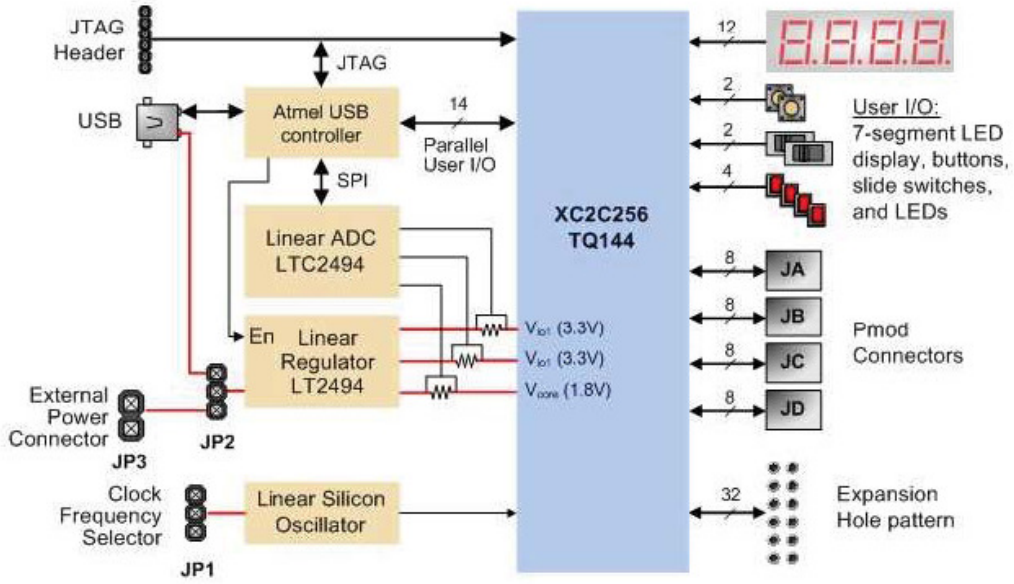
- przetwornik analogowo-cyfrowy służący do pomiaru prądu pobieranego ze źródła V_{CCINT} oraz dwóch źródeł V_{CCIO} zasilających sterowniki linii wyjściowych podczas normalnej pracy,
- konfigurowalny oscylator umożliwiający pracę układu z częstotliwościami 10, 100 oraz 1000 kHz,
- 64 linie We/Wy dostępne za pomocą portów rozszerzeń PMod oraz za pomocą wyprowadzeń na płycie drukowanej,
- złącze dla zewnętrznej pamięci SPI PROM.



Rys. 5. Płyta uruchomieniowa CoolRunner II Starter Kit [4].

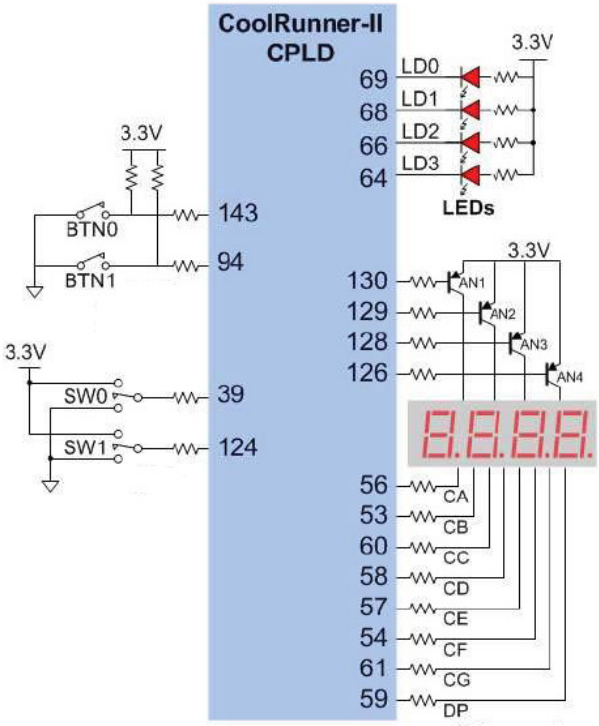
Na płycie uruchomieniowej znajduje się także mikrokontroler AT90USB162, którego zadaniem jest przede wszystkim obsługa złącza USB. W chwili obecnej brak wsparcia projektowego dotyczącego mikrokontrolera zawartego w układzie AT90USB162 ze strony producenta zestawu uruchomieniowego. Zapowiedź takiego wsparcia znajduje się jednak w dokumentacji do zestawu dołączonej na płycie CD. Do dyspozycji użytkownika pozostają natomiast następujące urządzenia We/Wy:

- dwa przełączniki,
- dwa przyciski,
- 4 diody LED,
- wyświetlacz siedmiosegmentowy zawierający 4 cyfry z kropkami dziesiętnymi.



Rys. 6. Schemat blokowy zestawu uruchomieniowego CoolRunner II Starter Kit. [4].

Na rys. 6 przedstawiającym schemat blokowy zestawu uruchomieniowego przedstawiono szczegółowo urządzenia peryferyjne oraz złącza i elementy konfiguracyjne. Na uwagę zasługuje złącze JP3, dzięki któremu układ może zostać zasilony z popularnej baterii 9V.



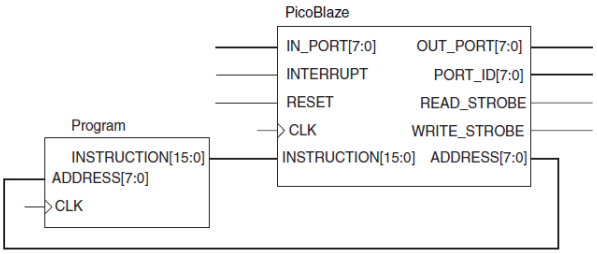
Rys. 7. Szczegółowy schemat podłączenia urządzeń peryferyjnych do układu CoolRunner II [4].

Na rys. 7 przedstawiony został schemat urządzeń We/Wy podłączonych do układu CPLD. Wynika z niego sposób działania przełączników (SW0, SW1), przycisków (BTN0, BTN1), diod (LED0 ÷ LED3) oraz wyświetlacza siedmiosegmentowego. Na schemacie zaznaczono numery portów układu CoolRunner II, do których zostały podłączone poszczególne urządzenia. Numery portów można także odczytać bezpośrednio z płytki uruchomieniowej. Umieszczone są one w nawiasach przy nazwach urządzeń i przy samych urządzeniach. Rozkład wyprowadzeń przypisanych do portów rozszerzeń dla modułów PMod (załącza J1, J2, J3, J4) oraz złącza J5 dostępny jest w dokumentacji [4].
Ze schematu na rys. 7 wynika, że do portów, do których dołączone są przyciski w stanie wyciśniętym, doprowadzone jest napięcie w stanie wysokim. Dzieje się tak za sprawą rezystorów podciągających do napięcia 3,3 V. Z kolei diody są zapalane napięciem niskim. Wyświetlacz siedmiosegmentowy w odróżnieniu od poprzednich ćwiczeń stanowi zestaw segmentów (diod) ze wspólną anodą. Stąd zapalenie danego segmentu wymaga ustalenia stanu niskiego na katodach diod (wyprowadzenia CA ÷ CG oraz DP) i stanu wysokiego na wyprowadzeniach AN1 ÷ AN4. Sam proces wyświetlania odbywa się analogicznie jak w przypadku poprzednich ćwiczeń w sposób sekwencyjny. Jest to wynikiem równoległego połączenia odpowiednich wyprowadzeń CA ÷ CG oraz DP w ramach zestawy wyświetlaczy. Zatem uzyskanie różnych znaków na poszczególnych wyświetlaczach wymaga przełączania zasilania poszczególnych sekcji za pomocą sygnałów AN1 ÷ AN4.
Płyta prototypowa jest dostarczana z programem demonstracyjnym, w ramach którego umieszczono obsługę wyświetlacza siedmiosegmentowego, diod, przycisków oraz przełączników. Program demonstracyjny realizuje funkcję stopera sekundowego. Jednak jest on wykonany w sposób całkowicie sprzętowy (projekt w języku VHDL), natomiast celem niniejszego ćwiczenia jest implementacja procesora programowego i obsługa urządzeń zewnętrznych w sposób programowy.

3. MIKROKONTROLER PICOBLAZE

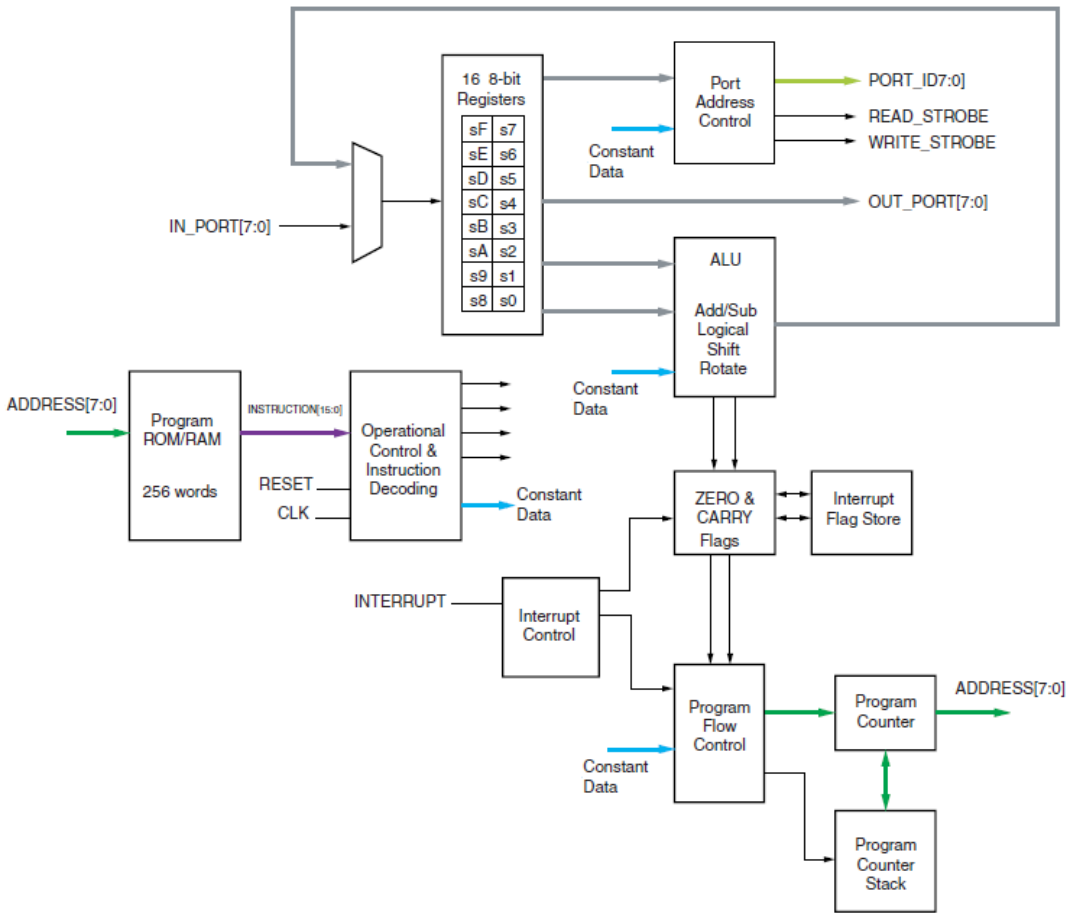
Mikrokontroler PicoBlaze został opracowany przez jednego z inżynierów firmy Xilinx – Kena Chapmana. Jest to stosunkowo prosty mikrokontroler 8-bitowy, opisany w językach opisu sprzętu. Jego niezaprzeczalną zaletą jest zakres funkcjonalny oraz dopasowanie do struktur reprogramowalnych. PicoBlaze występuje w dokumentacji także pod nazwą KCPSM (ang. *Constant (**K**) Coded Programmable State Machine*). PicoBlaze jest udostępniany bezpłatnie po uprzedniej rejestracji na stronach firmy Xilinx. Mikrokontroler ten istnieje w różnych wersjach, przeznaczonych na różne platformy sprzętowe m.in. CPLD, Spartan, Virtex itp. Architektura PicoBlaze jest nieco

ograniczona w stosunku do np. Spartan-3 czy innych. Związane jest to przede wszystkim z innym sposobem realizacji pamięci w technologii CPLD w stosunku do układów FPGA.
W ramach projektu mikrokontrolera PicoBlaze dostępne są źródła w języku VHDL, co umożliwia szczegółowe zapoznanie się z jego budową i zasadą działania, ale także stwarza rzadką okazję do eksperymentowania z modyfikacjami architektury i dostosowywaniem zasobów wewnętrznych do wymagań projektanta systemu. Ponieważ udostępniony jest także kod źródłowy asemblera dla PicoBlaze, możliwe jest zatem modyfikowanie zbioru instrukcji zarówno w modelu mikrokontrolera jaki i ich obsługi na poziomie asemblera.



Rys. 8. Schemat blokowy mikrokontrolera PicoBlaze wraz z pamięcią programu i wyprowadzeniami zewnętrznymi [2].

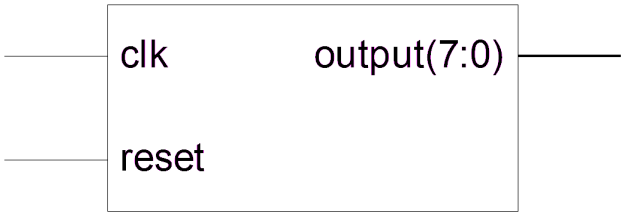
Na rys. 8 przedstawiono ogólny schemat blokowy podstawowej wersji mikrokontrolera PicoBlaze w podstawowej konfiguracji zawierającej pamięć programu. Widoczne są ponadto wyprowadzenia związane z obsługą portów zewnętrznych – zarówno wejść jak i wyjść wraz z sygnałami sterującymi, linia przerwań, sygnał RESET oraz wejście sygnału zegarowego. Od strony procesora widoczny jest jeden port wyjściowy (OUT_PORT) i wejściowy (IN_PORT). Oba mają rozmiar ośmiu bitów. Widoczny jest także wektor PORT_ID, który służy do wyboru (adresowania) portu zewnętrznego. Po zaimplementowaniu dodatkowego układu zewnętrznego, np. zatrzasku, możliwe jest zwiększenie liczby wyprowadzeń zewnętrznych do 256 (8 bitów portu x 8 linii adresowych)



Rys. 9. Ogólna architektura mikrokontrolera PicoBlaze [2].

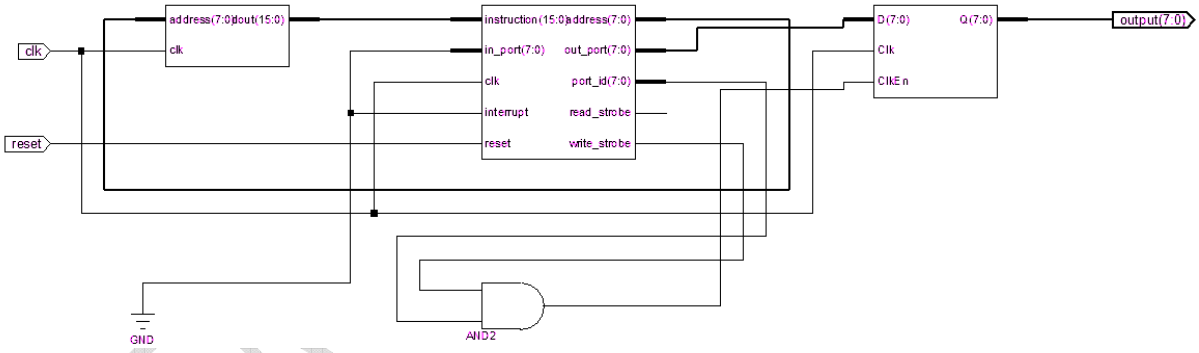
Szczegółową architekturę PicoBlaze przedstawia rys. 9. Widoczny jest tam zestaw rejestrów wewnętrznych. Standardowo występuje 16 rejestrów ośmiobitowych o nazwach s0 ÷ sF. W szczególnych przypadkach liczba ta może zostać zmieniona poprzez modyfikację kodu źródłowego VHDL. Jednostka arytmetyczno-logiczna wykonuje podstawowe operacje, przy czym w PicoBlaze brak zaimplementowanej np instrukcji mnożenia. Odpowiednik rejestru statusu lub słowa stanu znanego z innych procesorów zredukowany jest praktycznie do dwóch bitów: znacznika przeniesienia (CARRY) oraz znacznika zera (ZERO). W przypadku implementacji PicoBlaze w technologii FPGA pamięć programu składa się z 1024 szesnastobitowych słów, ale układach CPLD jest ona zredukowana do 256 słów o rozmiarze ośmiu bitów.

Wersja mikrokontrolera PicoBlaze wykorzystywana w trakcie ćwiczeń laboratoryjnych jest przedstawiona na rys. 10. Układ posiada tylko jeden ośmiobitowy port wyjściowy, wejście zegarowe oraz wejście sygnału RESET.



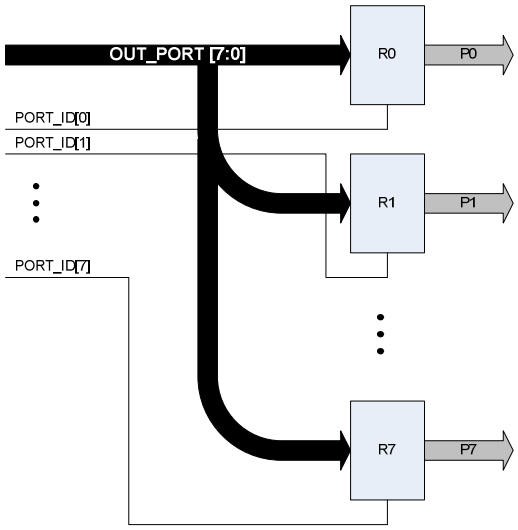
Rys. 10. Wyprowadzenia mikrokontrolera PicoBlaze w najprostszej wersji.

Strukturę wewnętrzną wykorzystywanego mikrokontrolera ilustruje rys. 11. Widoczne są tam połączenia pamięci programu z blokiem mikrokontrolera oraz układ logiczny połączony z jednym, ośmiobitowym, rzeczywistym portem wyjściowym. Liczba rejestrów wewnętrznych mikrokontrolera PicoBlaze dla CPLD jest ograniczona do ośmiu (o nazwach s0 ÷ s7).



Rys. 11. Mikrokontroler PicoBlaze w połączeniu z pamięcią programu i portem wyjściowym

Pewnego komentarza wymaga implementacja portów We/Wy oraz sposób ich obsługi w mikrokontrolerze PicoBlaze. Jak wspomniano wyżej, PicoBlaze posiada możliwość obsługi do 256 portów zewnętrznych. Jednak model właściwego procesora posiada tylko jedno wyjście OUT_PORT o rozmiarze ośmiu bitów. Istnieje ponadto wyjście PORT_ID, również o rozmiarze ośmiu bitów. W ostatecznej zatem implementacji systemu jako całości konieczne jest zaprojektowanie układu logiki, za pomocą której bieżąca wartość portu Wy (na liniach OUT_PORT) zostanie przekazana na zewnątrz układu. Wymaga to w praktyce zastosowania dodatkowych rejestrów zewnętrznych uaktywnianych wybranymi liniami PORT_ID. Dzięki temu możliwy jest wybór jednego, kilku bądź nawet wszystkich fizycznych rejestrów zewnętrznych (rys. 12).
W załączonym do ćwiczenia projekcie mikrokontrolera PicoBlaze przeznaczonego do implementacji w strukturach CPLD zaprojektowany został mechanizm przedstawiony powyżej. Liczba właściwych portów zewnętrznych został ograniczona w praktyce do jednego. Nie zwalnia to jednak użytkownika z konieczności adresowania portu zewnętrznego nawet w tym przypadku.
Realizacja zadań wymagających większej liczby portów implikuje konieczność ingerencji w kod źródłowy projektu demonstracyjnego. Szczegóły niezbędnych zmian zostaną przedstawione dalej, przy okazji omówienia zadań do samodzielnej realizacji.



Rys. 12. Obsługa portów zewnętrznych z mechanizmem adresowania

Lista instrukcji obejmuje następujące grupy instrukcji: sterujące wykonaniem programu, logiczne, arytmetyczne, przesuwania oraz rotacji bitowej, wejścia/wyjścia oraz obsługi przerwań. Listę instrukcji zawiera tabela 2. Szczegółowy opis instrukcji znajduje się w dokumentacji [2].

Tabela. 2: Lista rozkazów mikrokontrolera PicoBlaze.

Sterowanie wykonaniem programu
JUMP Z,aa
JUMP C,aa
CALL aa
CALL NZ,aa
CALL NC,aa
RETURN Z
RETURN C
Instrukcje logiczne
AND sX,kk
XOR sX,kk
AND sX,sY
XOR sX,sY
ADD sX,kk
SUB sX,kk
ADD sX,sY
SUB sX,sY
Instrukcje przesuwania i rotacji bitów
SR1 sX
SRA sX
SL0 sX
SLX sX
RL sX
INPUT sX,pp
OUTPUT sX,pp
Instrukcje obsługi przerwań
RETURNI DISABLE
DISABLE INTERRUPT

JUMP aa
JUMP NZ,aa
JUMP NC,aa
CALL Z,aa
CALL C,aa
RETURN
RETURN NZ
RETURN NC
LOAD sX,kk
OR sX,kk
LOAD sX,sY
OR sX,sY
Instrukcje arytmetyczne
ADDCY sX,kk
SUBCY sX,kk
ADDCY sX,sY
SUBCY sX,sY
SR0 sX
SRX sX
RR sX
SL1 sX
SLA sX
Instrukcje obsługi portów We/Wy
INPUT sX,(sY)
OUTPUT sX,(sY)
RETURNI ENABLE
ENABLE INTERRUPT

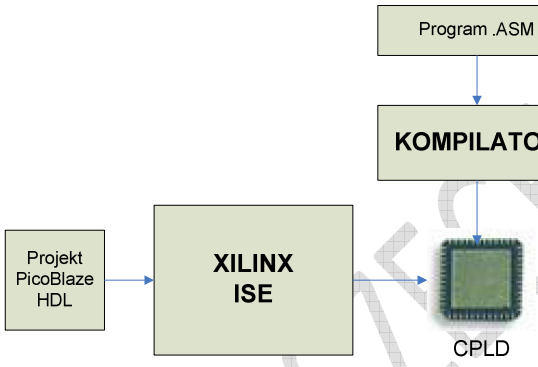
Legenda:

X, Y – odniesienie do definicji rejestru „s" w zakresie od 0 do F
kk – stała wartość z zakresu 00 ÷ FF
aa – adres z zakresu 00 ÷ FF
pp – adres portu z zakresu 00 ÷ FF

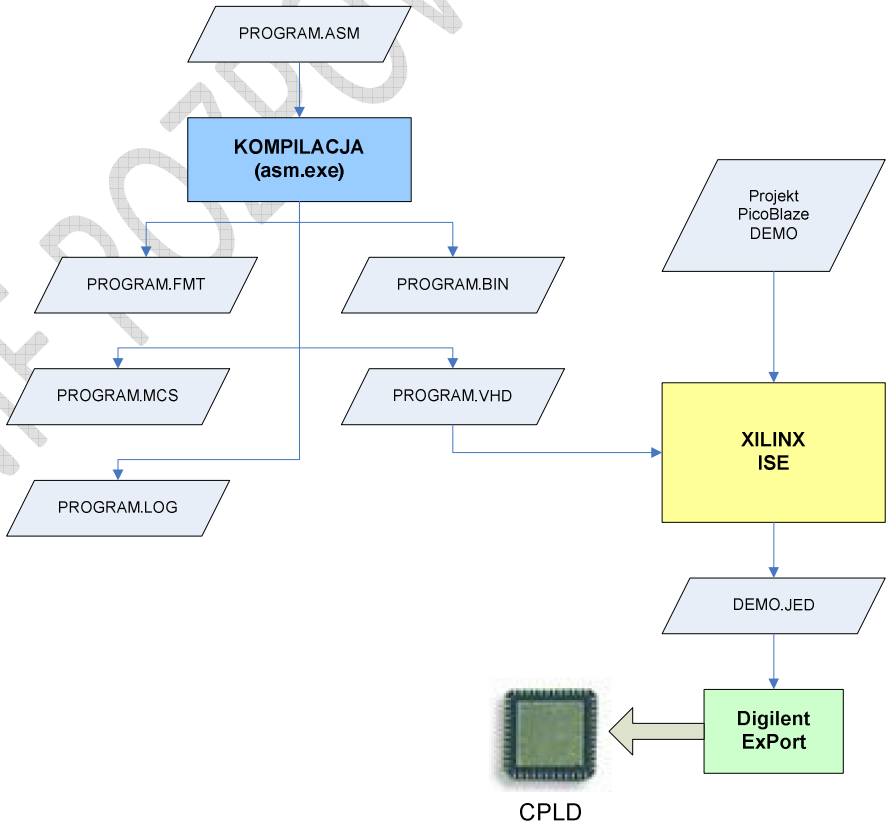
W ramach udostępnionych na ćwiczeniach materiałów znajduje się kompletny projekt zawierający zweryfikowany model mikrokontrolera PicoBlaze opisanego w języku VHDL oraz przykładowy program w asemblerze, a także sam asembler. Zasady posługiwania się narzędziami niezbędnymi do wykonania ćwiczenia opisane zostaną w kolejnym rozdziale.

4. NARZĘDZIA PROGRAMOWE

Intuicyjny proces implementacji projektu sprzętowo-programowego w strukturze reprogramowalnej mógłby wyglądać tak, jak przedstawiono na rys. 13. Oznacza to, że projekt struktury sprzętowej może zostać całkowicie rozdzielony od projektu sprzętowego. Zatem projektant w pierwszej kolejności projektuje sprzęt w postaci mikrokontrolera i implementuje go w strukturze, a następnie projektuje oprogramowanie i rezultat kompilacji umieszcza w pamięci programu. W ten sposób ew. zmiana programu nie wymaga ingerencji, ani nawet ponownego wykorzystania narzędzi obsługujących projekt sprzętowy.



Rys. 13. Schemat przepływu danych w przypadku rozdzielenia procesu projektowania sprzętu i oprogramowania



Rys. 14. Rzeczywista procedura kompleksowej implementacji projektu mikrokontrolera PicoBlaze

Niestety, z powodu ograniczeń w postaci wyboru struktury reprogramowalnej oraz określonej platformy uruchomieniowej rzeczywisty proces projektowania wymaga każdorazowego przeprowadzenia etapu implementacji sprzętowej (rys. 14). W zestawie uruchomieniowym brak bowiem zewnętrznej pamięci programu, a implementacja pamięci nieulotnej z możliwością zapisu poza procesem implementacji projektu struktury mikrokontrolera jest w praktycznie niewykonalna. Determinuje to więc konieczność zapisania programu mikrokontrolera w pamięci stałej oraz zintegrowania jej z całym projektem mikrokontrolera PicoBlaze. Z tego też powodu program po każdorazowej kompilacji musi zostać dołączony do projektu sprzętowego i poddany procesowi implementacji.
Wszystkie pliki potrzebne do wykonania ćwiczenia powinny zostać przekazane przez prowadzącego zajęcia. Przy czym struktura folderów oraz nazewnictwo poszczególnych plików może odbiegać od przyjętych w niniejszym opisie.

4.1. KOMPILACJA PROGRAMU

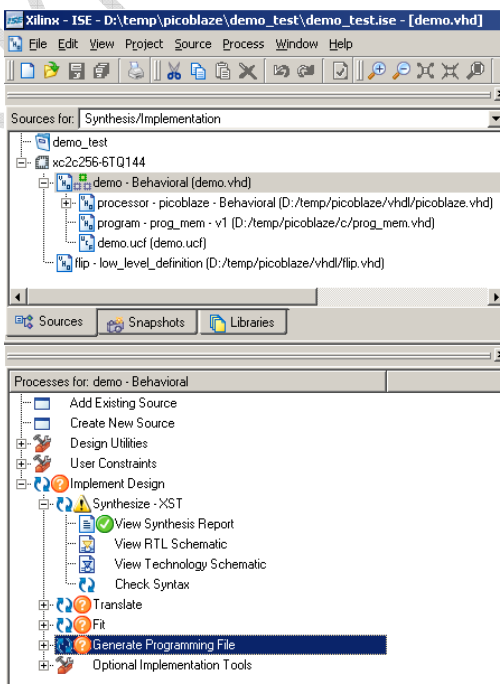
Asembler PicoBlaze jest programem konsolowym. W pierwszej kolejności należy zatem uruchomić linię komend i przejść do folderu, w którym znajduje się asembler oraz plik programu napisanego w asemblerze na mikrokontroler PicoBlaze. Ewentualną edycję pliku z kodem programu należy przeprowadzić z wykorzystaniem prostego edytora tekstowego (np. notepad.exe).Następnie w linii komend należy wydać polecenie:

```
X:\folder_docelowy\asm.exe prog_mem.asm,
```

gdzie prog_mem.asm jest nazwą pliku zawierającego program. Nazwa pliku jest istotna, ponieważ taka sama nazwa jest używana w projekcie sprzętowym mikrokontrolera PicoBlaze. Może ona zostać zmieniona, ale wymaga to ingerencji w kodzie źródłowym projektu sprzętowego w celu zapewnienia spójności powiązań plików.
W wyniku bezbłędnej kompilacji powinien powstać zbiór plików, przy czym z punktu widzenia całości najbardziej istotny jest plik `prog_mem.vhd`. Jest to wynik kompilacji umieszczony w postaci pamięci ROM w strukturze pliku VHDL, który następnie może zostać bezpośrednio dołączony do projektu sprzętowego.

4.2. ŚRODOWISKO XILINX ISE

Po utworzeniu i skompilowaniu programu asemblerowego należy uruchomić środowisko projektowe Xilinx ISE. Jeśli po uruchomieniu programu automatycznie zostanie wczytany poprzedni projekt należy go zamknąć korzystając z opcji File | Close Project z głównego menu programu.

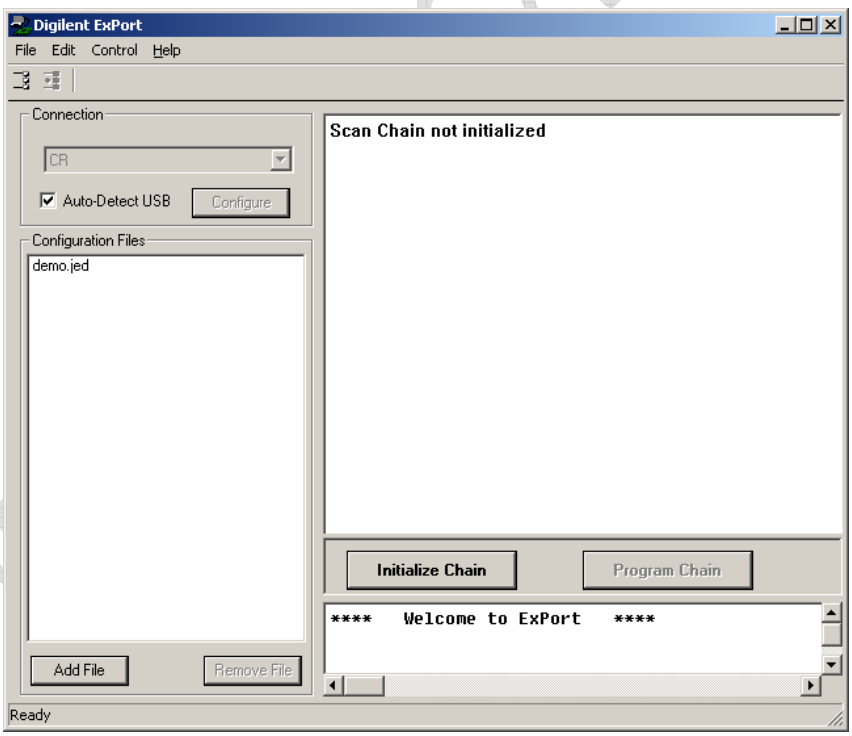


Rys. 15. Okna Sources i Processes programu Xilinx ISE

Następnie należy otworzyć projekt o nazwie `demo_test.ise`. Fragment głównego okna programu ISE zawierający okno dokowane ze źródłami projektu oraz dostępnymi procesami przedstawia rys. 15. Zawartość okna **Processes** zmienia się w zależności od wyboru (podświetlenia) pliku w oknie **Sources**.
Po podświetleniu programu nadrzędnego (w przypadku omawianego projektu jest nim plik `demo.vhd`), w oknie procesów należy rozwinąć klucz **Implement Design** i kliknąć podwójnie na opcji **Generate Programming File**. Po zakończonym sukcesem procesie implementacji ikona obok opcji Generate Programming File powinna zmienić kolor na zielony. W wyniku implementacji w oknie głównym automatycznie otwarty zostanie raport dotyczący projektu. Można w nim znaleźć np. poziom wykorzystania dostępnych zasobów sprzętowych przez projektowany system.
Po implementacji można zapoznać się z architekturą otrzymanego projektu w formie schematu hierarchicznego[1]. Należy w tym celu kliknąć podwójnie opcję **Implement Design | Synthesize – XST | View RTL Schematic**.
Wynikiem implementacji jest plik `demo.jed`, który może posłużyć bezpośrednio do zaprogramowania układu docelowego na platformie uruchomieniowej. W tym celu należy podłączyć płytkę uruchomieniową do komputera za pomocą kabla USB. Następnie należy uruchomić program **ExPort** korzystając z menu **Start | Digilent | Adept** systemu Windows.

UWAGA:
Z powodu ograniczeń konta użytkownika po podłączeniu płyty uruchomieniowej do komputera za pomocą kabla USB może pojawić się konieczność zainstalowania sterowników sygnalizowana obecnością okna logowania. W takim przypadku należy niezwłocznie powiadomić prowadzącego zajęcia i nie zamykać ww. okienka.

Jeśli zestaw uruchomieniowy został prawidłowo podłączony do komputera i urządzenie zostało prawidłowo rozpoznane przez system operacyjny, to należy wcisnąć przycisk **Initialize Chain** (rys. 16).

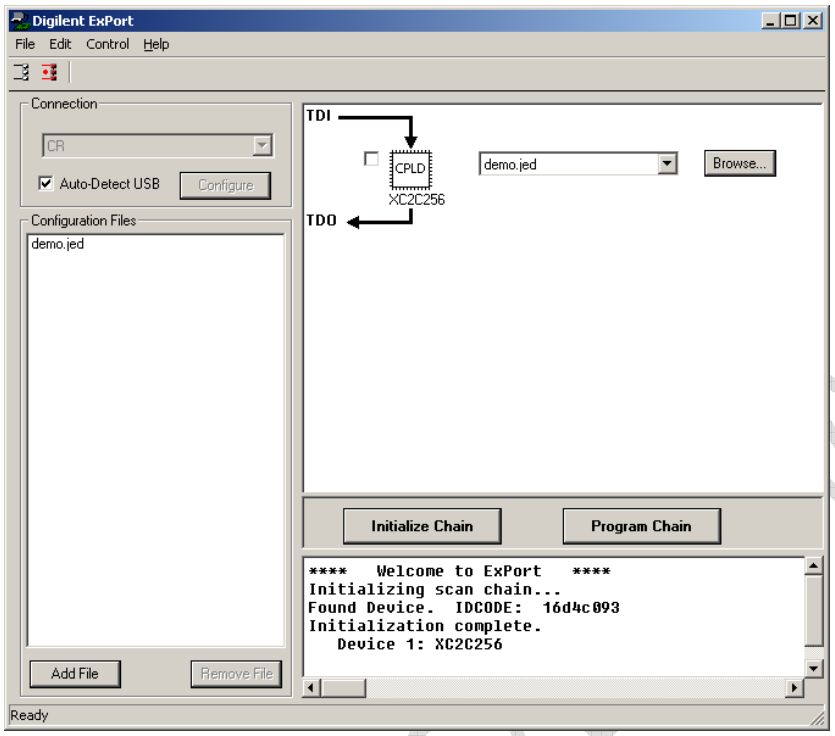


Rys. 16. Okno główne programu ExPort.

Po prawidłowym rozpoznaniu urządzenia przez program ExPort w oknie głównym powinien pojawić się obrazek przedstawiający układ CPLD (rys. 17). Następnie należy wskazać lokalizację pliku z rozszerzeniem `*.JED` (w przypadku ćwiczenia jest to plik `demo.jed`) za pomocą przycisku **Browse** obok obrazka z układem CPLD. Po wykonaniu tych czynności należy wcisnąć przycisk **Program Chain**. Po prawidłowym zaprogramowaniu

[1] W ten sposób otrzymano rys. 10 i rys. 11

układu należy na płytce uruchomieniowej wcisnąć przycisk BTN0, który w opisywanym projekcie pełni funkcję sygnału RESET.



Rys. 17. Okno programu ExPort po wykonaniu opcji Initialize Chain

Po wykonaniu modyfikacji w programie asemblerowym należy wykonać wszystkie opisane wyżej czynności ponownie, tzn.:

- kompilacja programu asemblerowego,
- wygenerowanie pliku do zaprogramowania zestawu uruchomieniowego w środowisku Xilinx ISE,
- programowanie układu programowalnego za pomocą programu ExPort.

WSKAZÓWKA:
Po skompilowaniu programu i przełączeniu aktywnego programu do środowiska ISE powinien ponownie zmienić się status projektu, sygnalizowany kolorem pomarańczowym ikon z lewej strony poszczególnych pozycji w okienku Processes.

5. ZADANIA

Wykonanie niniejszego ćwiczenia ma na celu zapoznanie się z procesem projektowania i implementacji systemów sprzętowo-programowych w postaci mikrokontrolera i programu przeznaczonego na tenże mikrokontroler. Nowością w ćwiczeniu jest sama filozofia systemu sprzętowo-programowego oraz architektura i zestaw instrukcji mikrokontrolera, pozostałe natomiast elementy systemu elementy powinny być znane z ćwiczeń poprzednich.

5.1. ZADANIE NR 1

W zadania nr 1 należy zaimplementować program przedstawiony na wydruku 1. Program (`prog_mem.asm`) został także dostarczony w materiałach źródłowych wraz z pełnym projektem PicoBlaze.
W liniach nr 01 – 03 znajdują się definicje stałych, używanych w programie. Są to kolejno: adres portu wyjściowego, liczniki pętli wykorzystywane w procedurze DELAY. Pętla główna znajduje się w liniach 07 – 13. Procedura opóźnienia realizowana jest w zasadzie przez dwie pętle. Wartości stałych TH i TL pełnią funkcję początkowych wartości liczników pętli. Realizacja procedury opóźnienia wstrzymuje oczywiście wykonywanie

pętli głównej. W tym kontekście jest nieefektywna i niezgodna z praktyką programistyczną, choć pozwala na stosunkowo szybkie osiągnięcie zamierzonego efektu.

1. Zaimplementuj przykładowy program z wydruku 1 w zestawie uruchomieniowym.
2. Dokonaj szczegółowej analizy programu z pomocą listy instrukcji PicoBlaze.
3. Zapoznaj się z rezultatem implementacji w postaci schematu (opcja View RTL Schematic).

```
01            CONSTANT LED_port, 01
02            CONSTANT TH, 5F
03            CONSTANT TL, FF
04
05            LOAD s7, EF
06
07     main:  OUTPUT s7, LED_port
08            CALL delay
09            LOAD s7, FF
10            OUTPUT s7, LED_port
11            CALL delay
12            LOAD s7, EF
13            JUMP main
14
15     delay: LOAD s5, TH
16     lp1:   LOAD s6, TL
17     lp2:   SUB s6, 01
18            JUMP NZ, lp2
19            LOAD s6, TL
20            SUB s5, 01
21            JUMP NZ, lp1
22            RETURN
```

Wydruk 1: Procedura obsługi przerwania układu transmisji szeregowej

5.2. ZADANIE NR 2

1. Napisz program na tzw. „wędrującą" diodę. Zapalona dioda powinna stwarzać wrażenie przemieszczania się na zmianę od lewej do prawej strony i z powrotem. Wykorzystaj algorytm przedstawiony na rys. 18.
2. Sprawdź, do których wyprowadzeń portu wyjściowego dołączone są diody i które bity w rejestrze wyjściowym odpowiadają za sterowanie pracą diod.

UWAGA:
Ścisły związek programu z architekturą mikrokontrolera, będący wynikiem wspólnego projektu sprzętowego ma wpływ na końcowy efekt w postaci błędów procesu implementacji (przekroczenie dozwolonych zasobów struktury CPLD) nawet wtedy, gdy program zostanie bezbłędnie skompilowany.
W miarę wierna implementacja algorytmu z wydruku 1 zapewnia poprawność procesu implementacji.

5.3. ZADANIE NR 3

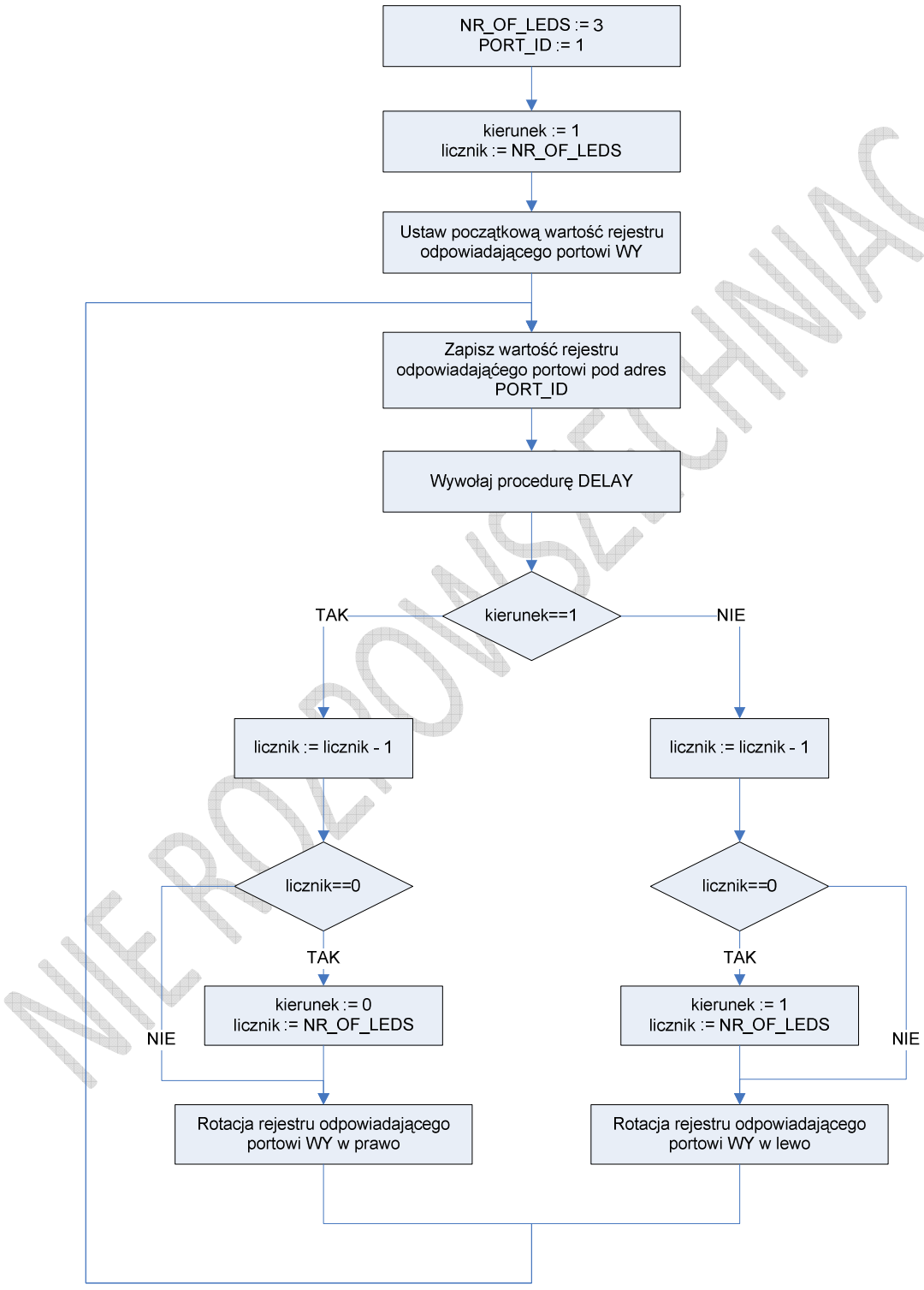
Napisz program, obsługujący umieszczony na płycie uruchomieniowej wyświetlacz siedmiosegmentowy. W wersji podstawowej program może wyświetlać na stałe różne cyfry na wszystkich wyświetlaczach. W wersji rozbudowanej program może realizować funkcję stopera sekundowego odmierzającego minuty i sekundy w formacie mm.ss.

W celu realizacji zadania nr 3 niezbędne będzie wprowadzenie zmian w kodzie źródłowym VHDL, które zwiększą liczbę dostępnych portów wyjściowych (w wersji podstawowej projektu mikrokontroler posiada tylko 8 linii wyjściowych).
Ponadto niezbędna będzie ingerencja w kod programu prog_mem.vhd, która będzie miała na celu ograniczenie wielkości programu poprzez zmianę wartości stałej oznaczającej wielkość programu:

```
constant ROM_LENGTH: INTEGER:= 128;
```

zamiast wartości domyślnej ustawianej przez kompilator oraz redukcję linii programu z 256 do 128.

Po zwiększeniu dostępnych linii wyjściowych konieczne będzie także przypisanie nazwom zmiennych (tablic) wykorzystanych w kodzie VHDL do obsługi portów konkretnych numerów wyprowadzeń układu CoolRunner II połączonych dalej z wejściami wyświetlacza siedmiosegmentowego zgodnie z dokumentacją [4].
Operacje taką można przeprowadzić dzięki opcji **User Constraints | Assign Package Pins** w oknie Processes.



Rys. 18. Algorytm ilustrujący zasadę działania programu z „wędrującą” diodą.

6. LITERATURA

[1] Nowakowski M.: PicoBlaze. Mikroprocesor w FPGA, BTC, Legionowo 2010.
[2] Xilinx (Chapman K.): PicoBlaze 8-Bit Microcontroller for Virtex-E and Spartan-II/IIE Devices, Application Note, 2003, (http://www.xilinx.com/support/documentation/application_notes/xapp213.pdf, dostęp: marzec 2011).
[3] Xilinx: PicoBlaze 8-Bit Microcontroller for CPLD Devices, Application Note, 2003, (http://www.xilinx.com/support/documentation/application_notes/xapp387.pdf, dostęp: marzec 2011).
[4] Xilinx: CoolRunner-II Evaluation Board Reference Manual, 2008, (http://www.xilinx.com/support/documentation/boards_and_kits/ug501.pdf, dostęp: marzec 2011).
[5] Xilinx: Programmable Logic Design. Quick Start Guide, 2008, (http://www.xilinx.com/support/documentation/boards_and_kits/ug500.pdf, dostęp: marzec 2011).
[6] Xilinx: Dokumentacja i oprogramowanie dołączone do zestawu uruchomieniowego Xilinx CoolRunner II CPLD Starter Kit.