Uniwersytet Mikołaja Kopernika
Wydział Fizyki, Astronomii i Informatyki Stosowanej

Mariusz Wiśniewski
254019

Praca inżynierska
na kierunku Automatyka i Robotyka

Interfejs szybkiej wymiany danych pomiędzy układem FPGA a kontrolerem CYUSB3014 z wykorzystaniem programowalnej maszyny stanów GPIF II

Opiekun pracy dyplomowej
dr Robert Frankowski
Zakład Fizyki Technicznej i Zastosowań Fizyki

Toruń 2015

Pracę przyjmuję i akceptuję

.....
data i podpis opiekuna pracy

Potwierdzam złożenie pracy dyplomowej

.....
data i podpis dziekanatu

Serdeczne dziękuję
Dd

Spis treści

1. Wstęp

Uniwersalną magistralę szeregową (USB) opracowano w 1996 roku, a rok później zaczęto ją wdrażać do komputerów stacjonarnych na masową skalę. W chwili obecnej każdy komputer obsługuję USB niezależnie od systemu operacyjnego. Ten standard odniósł spektakularny sukces, ponieważ w wielu przypadkach nie wymaga od użytkownika specjalistycznej wiedzy czy instalacji dedykowanych sterowników - urządzenia działają natychmiastowo po podłączeniu do komputera osobistego. Skorzystali na tym producenci urządzeń peryferyjnych produkujący pendrive'y, drukarki, myszki, klawiatury, aparaty fotograficzne i wiele innego sprzętu Plug and Play.

Również rynek mikroprocesorów otworzył się na tę magistralę, czego dowodem są produkty firmy Atmel z serii Xmega, systemy wbudowane wykorzystujące układy oparte na rdzeniu ARM oraz przede wszystkim firmy produkujące inteligentne telefony komórkowe. Nie trzeba już stosować dodatkowych programatorów lub konwerterów USB w celu zaprogramowania procesora. Zmniejsza to poniesione koszty dla użytkowników końcowych. USB wypiera zarówno w profesjonalnym jak i w amatorskim zastosowaniu magistrale o słabej przepustowości, na przykład popularny RS-232, I2C czy SPI.

1.1. Cel pracy

Niniejsza praca ma na celu dodanie możliwości dokonywania pomiarów oraz ich transfer w celu gromadzenia lub wizualizacji wykorzystując magistralę USB 3.0. Wykorzystano do tego celu komputer, kontroler EZ-USB FX3 firmy Cypress i zestaw ewaluacyjny oparty na układzie programowalnym FPGA Spartan 3E firmy Xilinx. Płytkę łączącą oba zestawy wykonano samodzielnie.

Rozdział drugi ma na celu przybliżyć standard USB, począwszy od USB 1.0 aż do najnowszej wersji, czyli 3.0. Omówiono w nim topologię oraz najistotniejsze elementy składające się na magistralę. Następnie wyjaśniono poszczególne rodzaje transmisji danych, takie jak kontrolna, przerwaniowa, masowa lub izochroniczna.

Rozdział trzeci skupia się na programowalnej maszynie stanów GPIF, która pozwala na implementację interfejsu Slave FIFO łączącego oba procesory. Wyjaśniane są założenia działania tego interfejsu, użyte sygnały, koncept flag sygnalizujących status buforów DMA oraz omówione narzędzia GPIF II Designer. W drugiej części przedstawiono płytkę łączącą FPGA z kontrolerem USB, zamieszczono schemat połączeń oraz zdjęcia wykonane podczas montażu elementów powierzchniowych (SMD). Dalej skupiono się na kontrolerze USB, czyli zestawie CYUSB3014 firmy Cypress. Przybliżono rozwiązania używane przez interfejs Slave FIFO, czyli gniazda, deskryptory i bufory DMA. Dalej skupiono się na programowalnej maszynie stanów GPIF, która pozwala na implementację interfejsu łączącego oba procesory. Wyjaśnione zostały założenia działania 16-bitowej magistrali, użyte sygnały, koncept flag sygnalizujących status buforów DMA oraz omówiono narzędzie GPIF II Designer. W drugiej części rozdziału przedstawiono płytkę łączącą FPGA z kontrolerem USB, zamieszczono schemat połączeń oraz zdjęcia wykonane podczas montażu elementów powierzchniowych (SMD).

Rozdział czwarty opisuje program napisany dla FPGA Spartan 3E. Kod oparty został na maszynach o skończonej liczbie stanów, dzięki którym przebiega transmisja danych. Dla każdej transmisji przesyłania danych zostały przedstawione diagramy i opisy zmiany stanów sygnałów zarządzających magistralą. Został również omówiony użyty ekran LCD, który sygnalizuje obecny stan oraz wysyłane/odbierane bajty danych. Ponadto skupiono się na zaimplementowanych modułach, takich jak Cyfrowe Zarządzanie Zegarem oraz pamięć FIFO.

W ostatnim rozdziale zamieszczono efekty transmisji danych, czyli zrzuty ekranów programów Cypress Control Center oraz Cypress Streamer.

2. Wprowadzenie do USB

USB jest rozwiązaniem idealnym jeśli istnieje potrzeba, aby komputer komunikował się z zewnętrznymi urządzeniami. Interfejs sprawdza się zarówno w projektach produkowanych na skalę przemysłową jak i w amatorskich konstrukcjach. Poniżej przedstawiono korzyści jakie płyną z zastosowania tej magistrali.

Korzyści dla użytkowników końcowych:

- łatwość obsługi - istnieje jeden standard dla wielu urządzeń, wytrzymałe konektory, automatyczna konfiguracja urządzenia w systemie, możliwość działania bez względu na system operacyjny, możliwość rozszerzenia portów USB przy użyciu HUB-ów, prostota instalacji sterowników jeśli zajdzie taka potrzeba
- możliwość zasilania urządzeń napięciem 5V prosto z portu, brak zewnętrznych zasilaczy
- stabilność transferu - USB nie jest wrażliwe jak magistrala RS-232 na środowisko zewnętrzne, w przypadku błędów możliwa jest retransmisja danych

Korzyści dla projektantów urządzeń:

- uniwersalność - posiadając różne typy oraz prędkości przesyłu danych można optymalnie dostosować projekt do wymagań i wykonywanych zadań
- USB jest wspierane na systemach opartych o jądro GNU/Linux, Windowsie oraz komputerach firmy Apple
- liczna literatura w postaci opracowań, mnogość przykładów, wsparcie producentów (Cypress)

2.1. Standard USB 2.0

System USB jest asynchroniczną magistralą komunikacyjną posiadającą głównego hosta oraz do 127 urządzeń peryferyjnych. Host zarządza magistralą, jest on odpowiedzialny za wykrywanie dołączanych urządzeń, inicjalizuje i zarządza transferem danych. Można rozszerzyć tę liczbę dodając HUB-a w topologii gwiazdy. Standard USB 2.0 wykorzystuje dwa przewody sygnałowe w trybie half-duplex (odbieranie i przesyłanie danych odbywa się na przemian co powoduje niższy transfer).

Wspierane są trzy prędkości przesyłu:

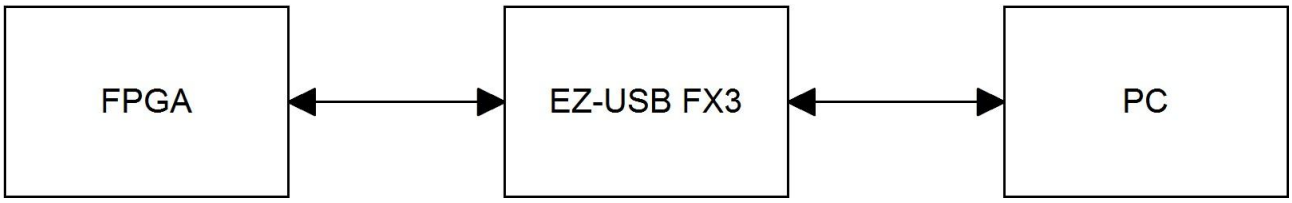
- low speed - transfer z maks. prędkością 1,5 Mbit/s, opracowany w standardzie USB 1.0
- full speed - transfer z maks. prędkością 12 Mbit/s, opracowany w standardzie USB 1.1
- high speed - transfer z maks. prędkością 480 Mbit/s, opracowany w 2001 roku (USB 2.0)

Poniżej zostaną przedstawione najważniejsze

3. Interfejs Slave Fifo oraz programowalna maszyna stanów GPIF II

Jednym z najpopularniejszych interfejsów, który można wykorzystać w urządzeniu EZ-USB FX3 firmy Cypress jest implementacja synchronicznego Slave Fifo. Ten rodzaj transmisji posiada bardzo bogatą dokumentację dołączoną do zestawu CYUSBKIT-001 na płycie CD oraz na oficjalnej stronie producenta. Bezpośredni dostęp do rejestrów FX3 nie jest możliwy w tym zastosowaniu, zatem urządzenie zewnętrzne (w tym przypadku Spartan 3E) zapisuje lub odczytuje dane z buforów FIFO.

Poniższy schemat przedstawia schemat działania interfejsu Slave FIFO:



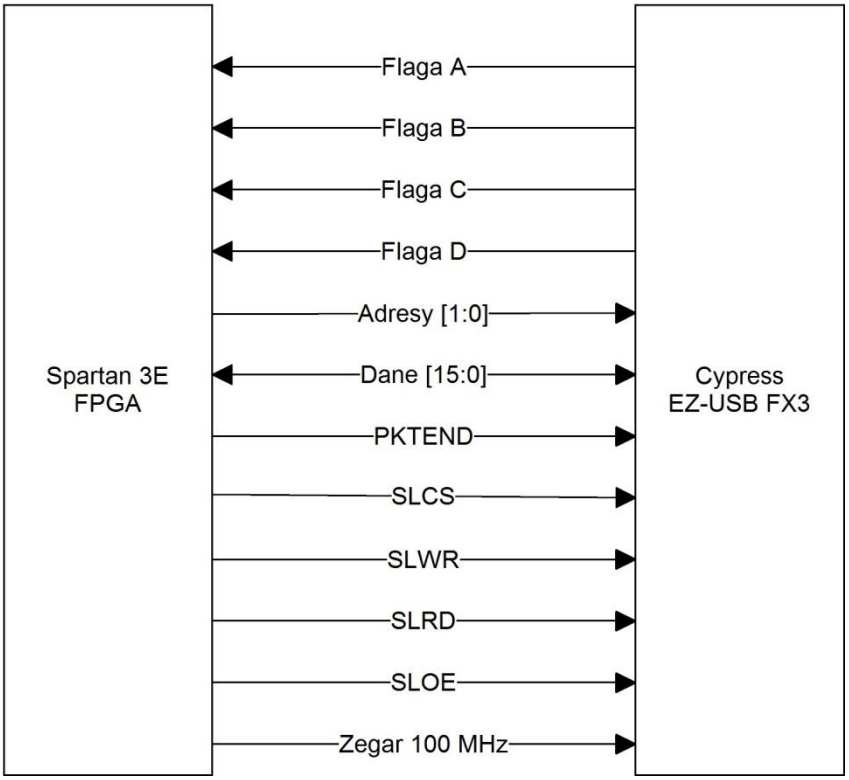
Rys. 1. Schemat działania aplikacji

3.1. Synchroniczny interfejs Slave FIFO

Wybór tego interfejsu jest idealnym rozwiązaniem, gdy zewnętrze urządzenie (dowolny procesor lub układ programowalny) standardowo nie posiada magistrali USB, a zachodzi potrzeba komunikacji z komputerem PC. Można w tym przypadku zastosować komunikację RS-232, ale użytkownik jest ograniczony słabą przepustowością magistrali - Slave Fifo oraz EZ-USB FX3 rozwiązują ten problem.

Diagram prezentuje schematycznie używane sygnały, a w tabeli znajduje się ich szczegółowy opis.

Nazwa sygnału	Opis
Flaga A, B, C, D	Jedyne sygnały (poza danymi), które pochodzą z FX3. Wskazują możliwość dostępu do gniazd FX3 (puste/pełne). Flaga A i B są używane przy transmisji wysyłania danych, a C i D przy odbiorze.
Adresy [1:0]	2-bitowa szyna adresowa, wskazuje na aktualny wątek
Dane [15:0]	W tej pracy szyna danych jest 16-bitowa (może być 8, 16 lub 32-bitowa)
Pktend	Sygnał używany przy transmisji krótkiego pakietu (short packiet) lub o zerowej długości (zero-length packet)
SLCS	Wybór urządzenia, sygnał zawsze aktywny podczas dowolnej transmisji
SLWR	Sygnał zapisu do FX3
SLRD	Sygnał odczytu z FX3
SLOE	Sygnał zezwolenia na wysyłkę danych do FX3
Zegar 100 MHz	Zegar pochodzący z urządzenia nadrzędnego (FPGA) do podrzędnego (FX3), stąd nazwa asynchroniczny interfejs Slave Fifo

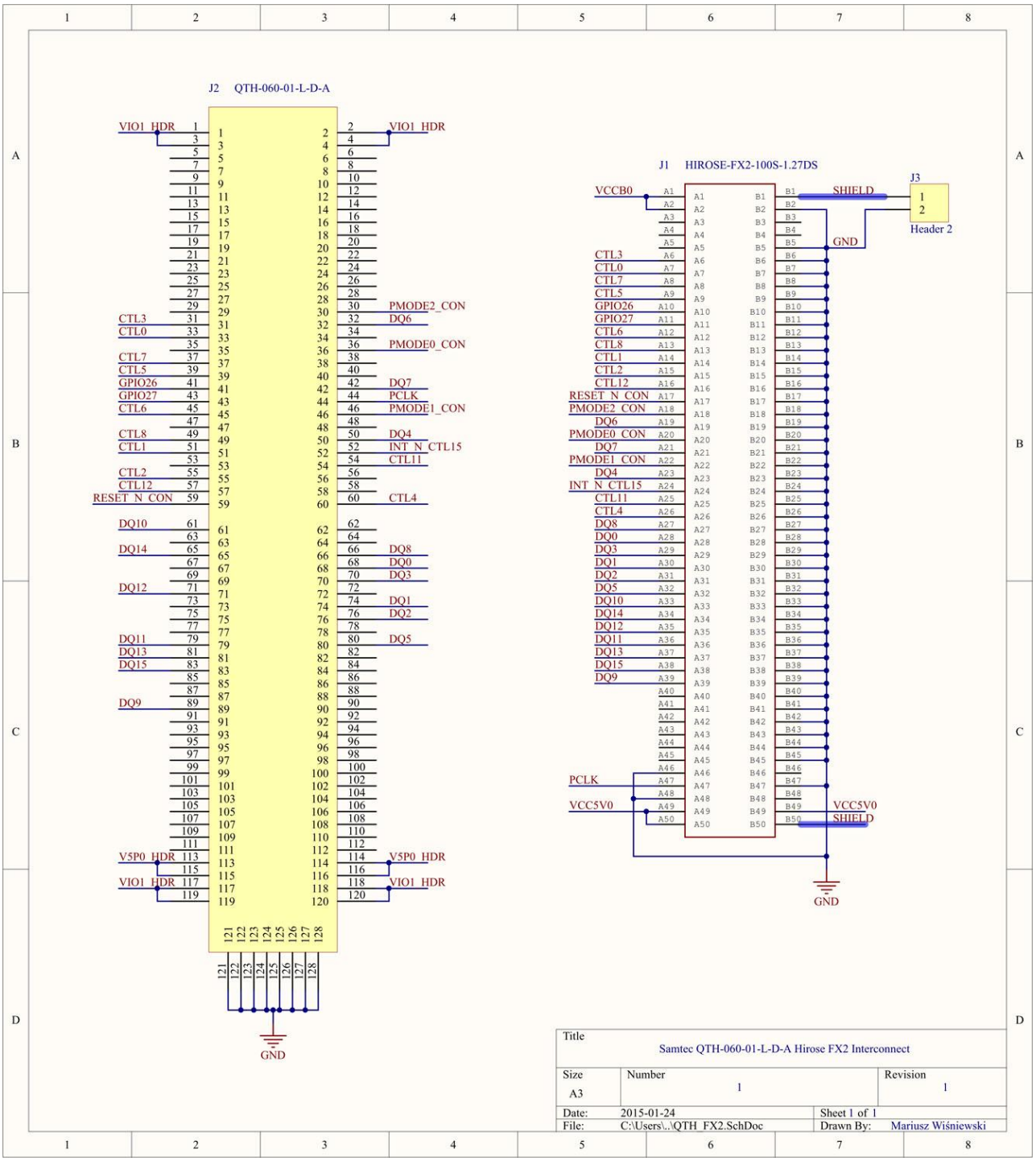


Rys. 2. Schemat działania interfejsu Slave Fifo

Synchroniczny interfejs Slave Fifo dzięki 2-bitowej linii adresowej jest w stanie uzyskać dostęp do maksymalnie czterech wątków. Jest możliwość rozszerzenia tej liczby, jednak wiąże się to z potrzebą użycia 5-bitowej linii adresowej. Pociąga to za sobą rzecz jasna konsekwencje, np. dodatkowe opóźnienia podczas monitorowania stanu flagi oraz po zmianie adresu. W omawianej pracy ten przypadek nie jest rozpatrywany, nie zachodziła taka potrzeba.

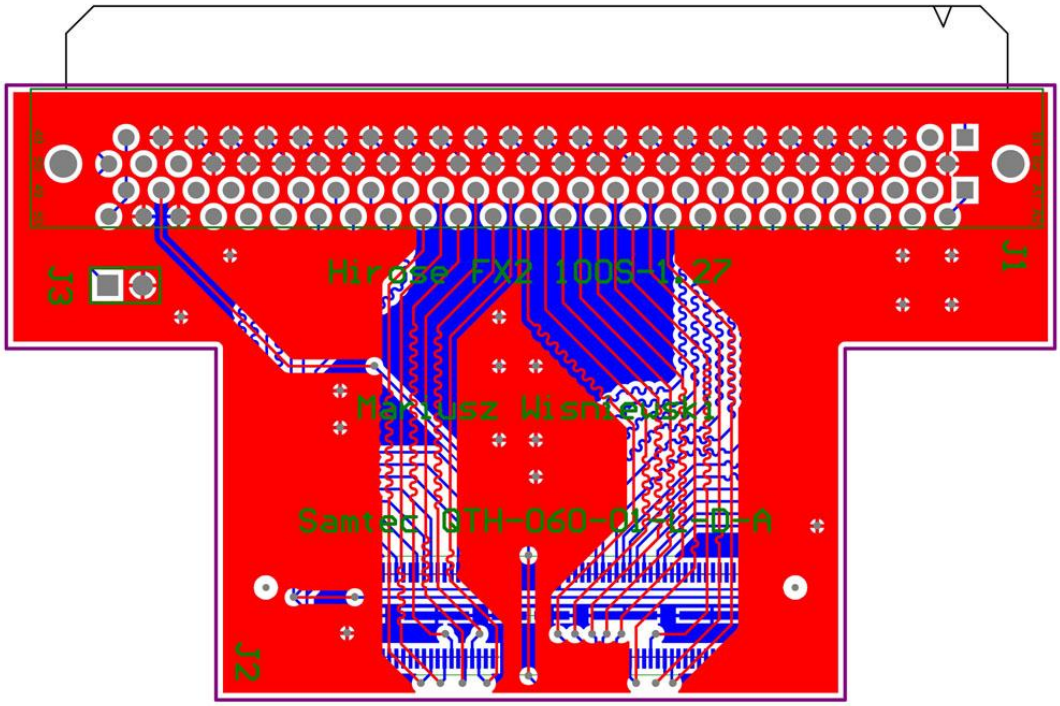
3.2. Płytka łącząca dwa zestawy uruchomieniowe

Firma Cypress nie posiada w swojej ofercie złącza, które pozwoli na sprzęgnięcie ze sobą obu używanych zestawów w pracy. W sklepie internetowym widnieje jedynie płytka CYUSB3ACC-002, która jest przeznaczona dla układów posiadających złącze FMC (Xilinx LPC). Należy dodać, że ten produkt nie jest obecnie już produkowany.



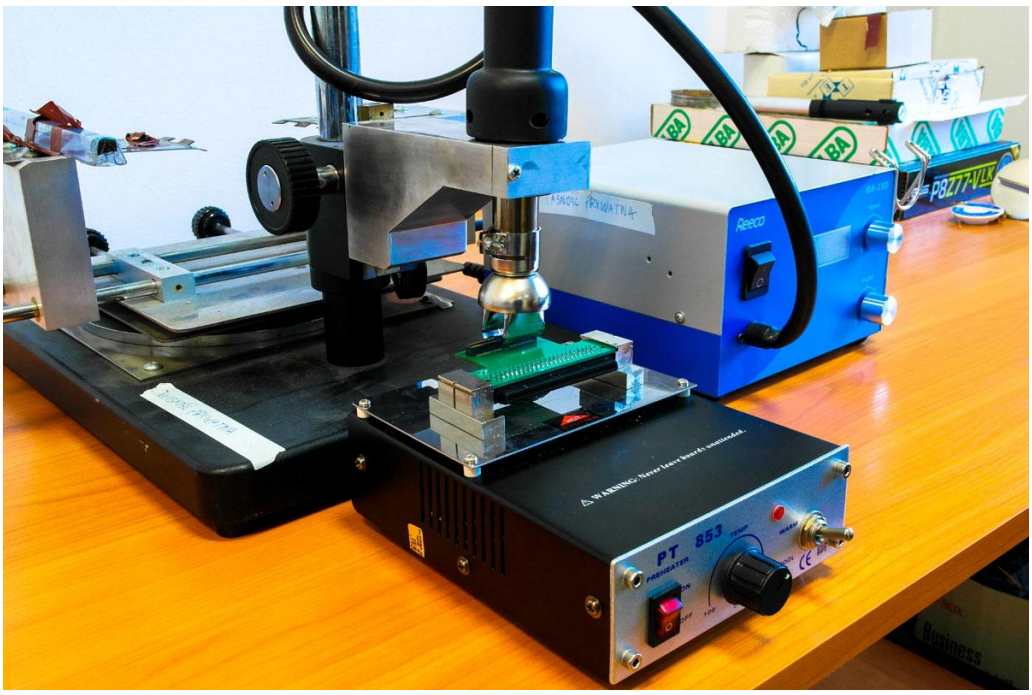
Rys. 3. Schemat elektryczny płytki

Schemat elektryczny oryginalnej płytki oraz dokumentacje elektryczne zestawu Spartan 3E posłużyły jako baza do zaprojektowania własnej przejściówki, opartej na złączach Hirose FX2 oraz Samtec QTH-060-01-L-D-A. Schemat ideowy oraz projekt PCB został wykonany w programie Altium Designer, ze względu na możliwości, które oferuje to oprogramowanie. Wszystkie ścieżki (oprócz linii zegarowej) posiadają identyczną długość. Ma to na celu zlikwidowania ewentualnych opóźnień na magistrali danych i wyeliminowanie efektu propagacji sygnałów. Płytka jest dwuwarstwowa.



Rys. 4. Projekt PCB płytki

Po wygenerowaniu w Altium Designerze wszystkich warstw i otworów projekt został wysłany do firmy Techno-Service, która specjalizuje się w wykonywaniu obwodów drukowanych. Po otrzymaniu gotowej płytki rozpoczęto przygotowania do montażu złącza QTH-060-01-L-D-A metodą na gorące powietrze.

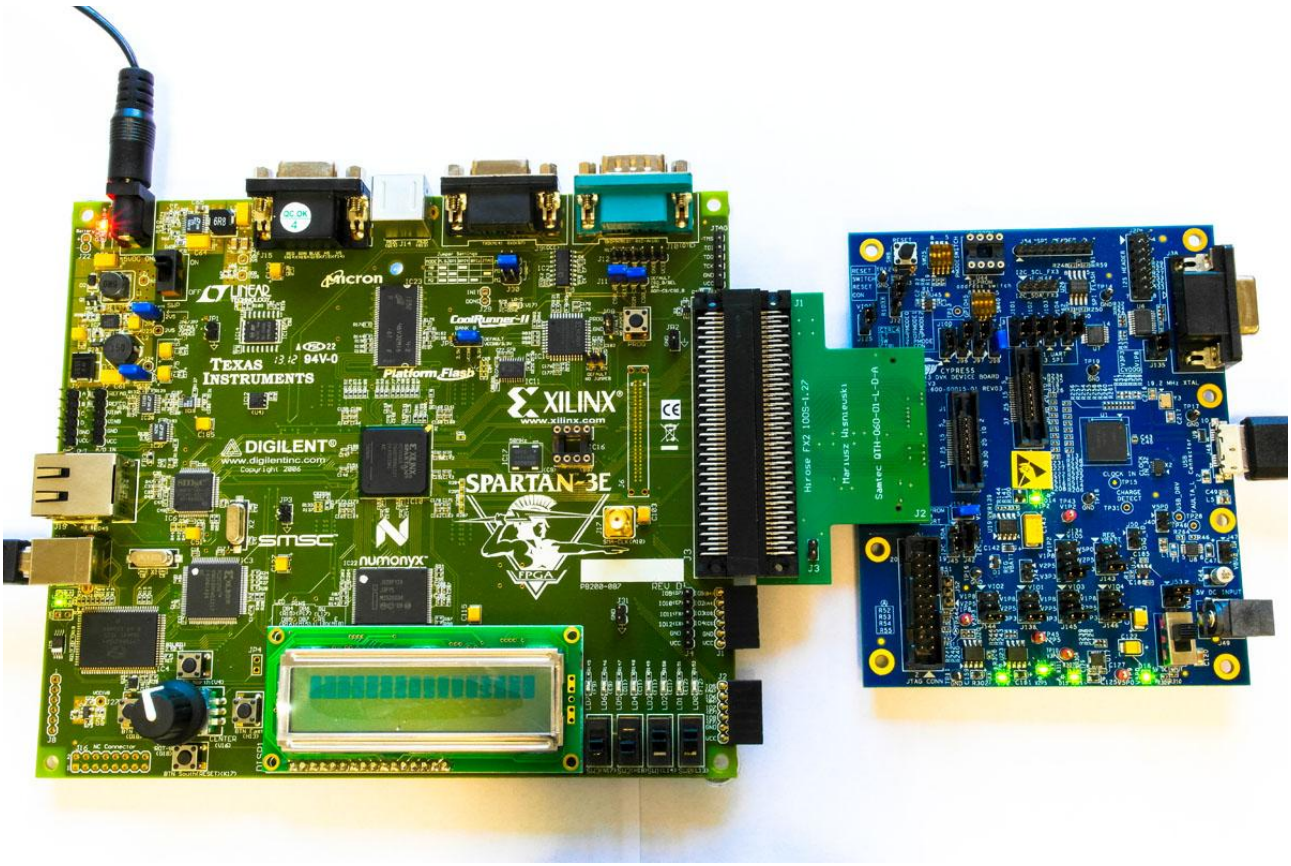


Rys. 5. Proces montażu elementów gorącym powietrzem. Widoczny podgrzewacz oraz lutownica hot-air

Do lutowania metodą na gorące powietrze posłużyły następujące narzędzia:

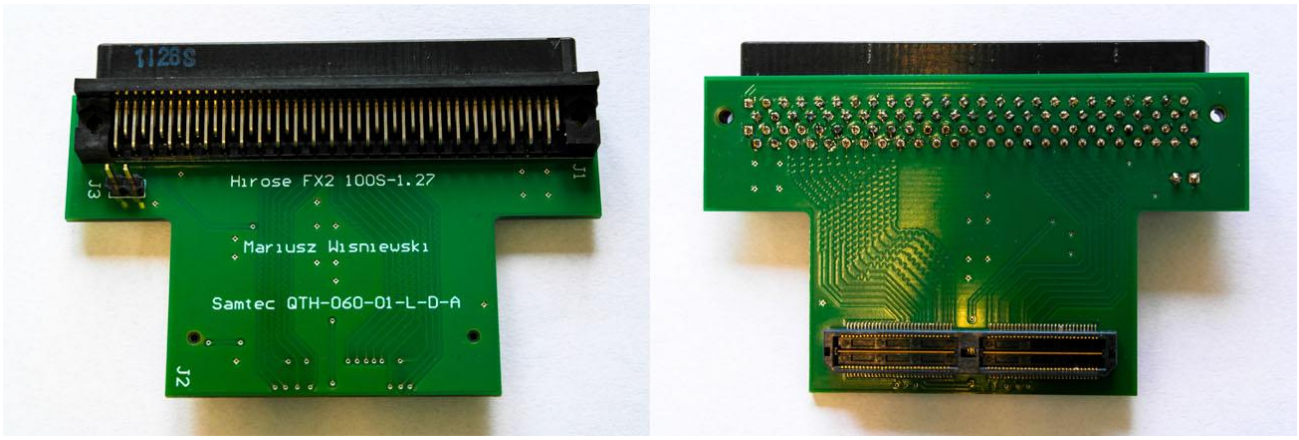
- podgrzewacz PT 853
- stacja hot-air Reeco RA-150

Poniżej przedstawiono fotografię efektu końcowego, czyli gotową płytkę sprzęgającą dwa układy.



Rys. 6. Oba układy po sprzęgnięciu ze sobą

Zdjęcia samej płytki z obu stron:



Rys. 7. Po lewej złącze Hirose FX2, po prawej Samtec QTH-060-01-L-D-A

3.3. Programowalna maszyna stanów GPIF II

GPIF II to programowalna maszyna stanów, która wprowadza możliwość implementacji dowolnego standardu komunikacji (własnego lub wcześniej opracowanego przez korporacje). Może funkcjonować jako podmiot nadrzędny lub podrzędny (w pracy GPIF II podlega urządzeniu głównemu, czyli FPGA).

Posiadane funkcje:

- funkcjonowanie jako urządzenie nadrzędne lub podrzędne
- oferuje 256 programowalnych stanów
- wspiera 8, 16 i 32-bitową równoległą szynę danych
- maksymalna częstotliwość pracy to 100 MHz
- daje możliwość konfiguracji 14 niezależnych sygnałów przy 32-bitowej szynie danych
- daje możliwość konfiguracji 16 niezależnych sygnałów przy 8 lub 16-bitowej szynie danych

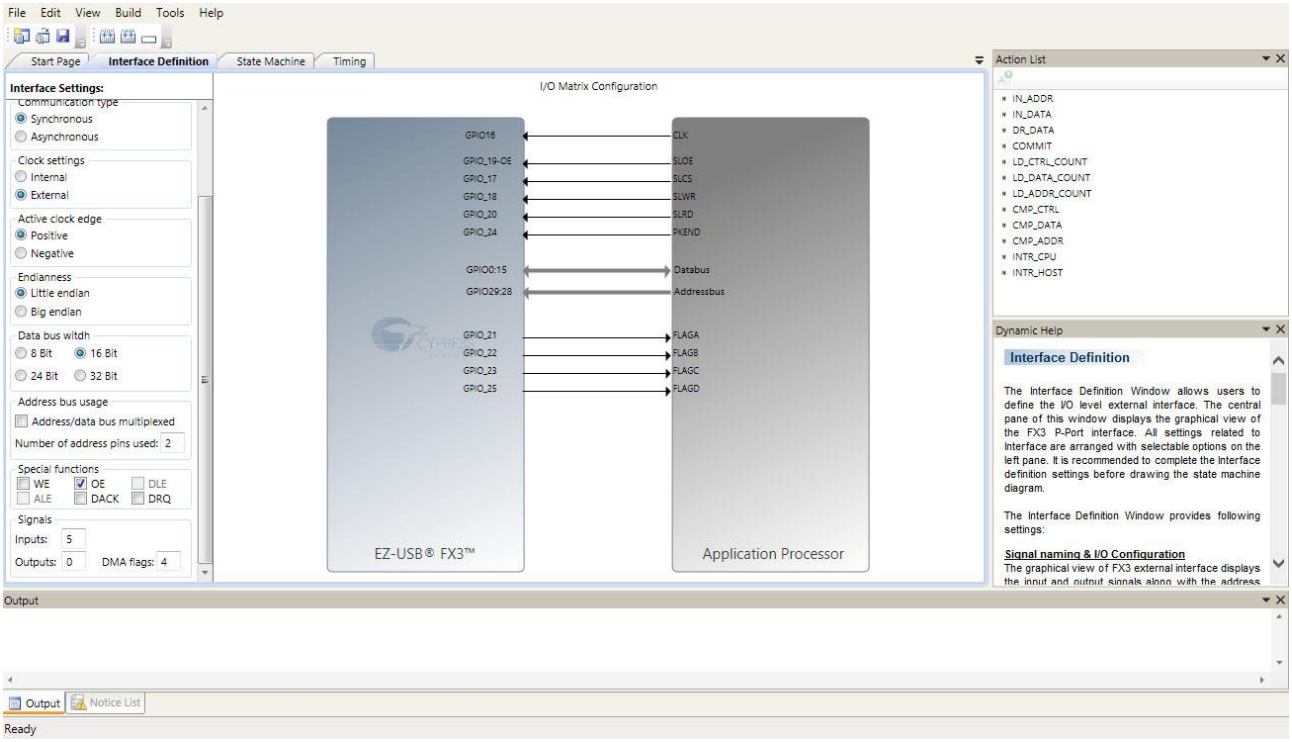
Zachowanie maszyny stanów jest zdefiniowane w deskryptorze, składa się on głównie z zestawu programowalnych rejestrów konfiguracyjnych. W kontrolerze EZ-USB FX3 została dla niego przydzielona pamięć 8 kB.

3.4. Narzędzie GPIF II Designer

W tym rozdziale przedstawiono konfigurację interfejsu Slave Fifo w programie GPIF II Designer, począwszy od rozpoczęcia projektu aż do konfiguracji częściowych flag (B i D).

3.4.1. Utworzenie i edycja sygnałów interfejsu

Rozpoczęto od importu projektu *sync_slave_fifo_2bit* przygotowanego przez firmę Cypress. Jednak w tym przypadku nie ma możliwości zmiany najważniejszych parametrów interfejsu, więc należy zapisać projekt jako edytowalny (*File - Save Project As Editable*). Po otworzeniu właściwego projektu ukazują się nam pełne możliwości edycyjne programu GPIF II Designer.



Rys. 8. Okno programu GPIF II Designer

W lewej części okna wybierając odpowiednie opcje można dostosować projekt do własnych potrzeb. Przeprowadzone zmiany:

- typ interfejsu: podrzędny (slave)
- komunikacja: synchroniczna
- zegar: zewnętrzny
- reakcja na zbocze: stan wysoki
- kolejność bajtów: cienkokońcowość (little endian)
- liczba bitów szyny danych: 16
- liczba bitów szyny adresowej: 2
- sygnałów wejściowych: 5
- flag DMA: 4

Następnie należało zmienić domyślne nazwy na docelowe (np. SLCS, FLAGA) dla zwiększenia czytelności projektu. Każdy sygnał może mieć osobną konfigurację oraz przypisany inny pin na procesorze FX3, przyjęto poniższe ustawienia:

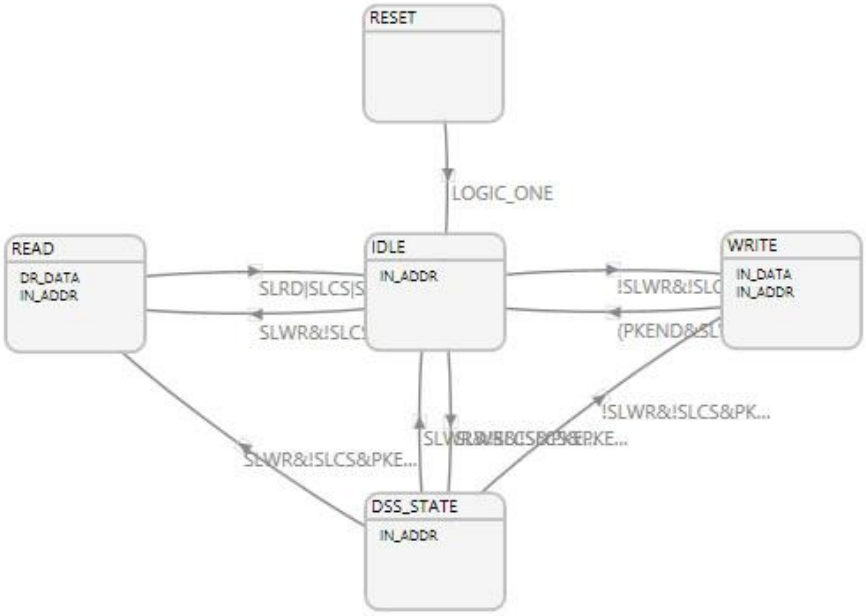
Tab. X. Konfiguracja sygnałów interfejsu Slave Fifo

Nazwa sygnału	Logiczne "1"	Pin FX3
CLK	Stan niski	16
SLOE		19
SLCS		17
SLWR		18
SLRD		20
PKTEND		24
Szyna danych	Nie dotyczy	0:15

Szyna adresowa	Nie dotyczy	29:28

3.4.2. Edycja maszyny stanów

Standardowo maszyna stanów obejmowała 7 stanów, zmniejszono tę liczbę do 5 stanów.



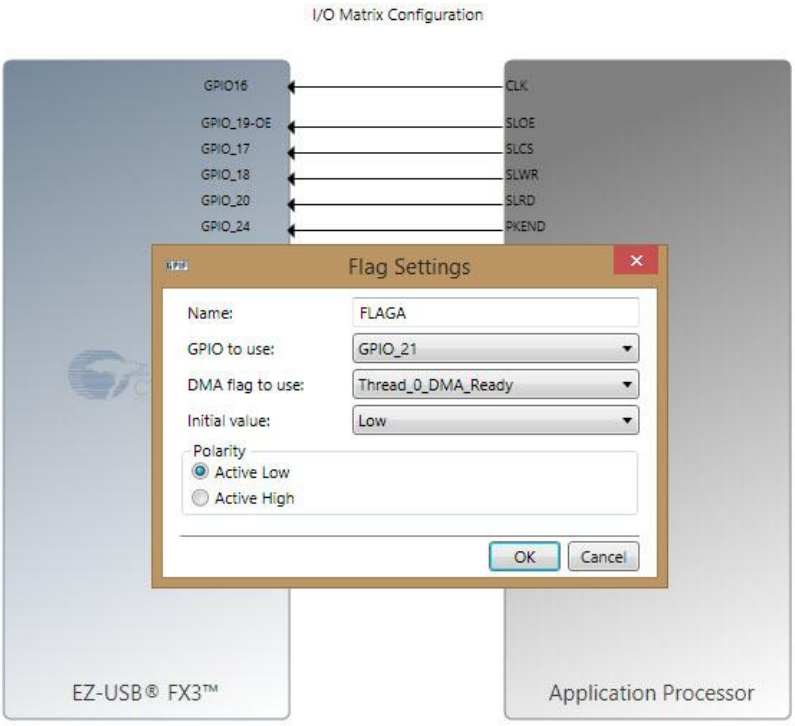
Rys. 9. Maszyna stanów Slave Fifo

Można ją utworzyć od podstaw, jednakże oparto się na projekcie, który zaimplementowała firma Cypress.

3.4.3. Edycja flag pełnych oraz częściowych

W tym miejscu została przedstawiona jedynie praktyczna strona realizacji flag pełnych i częściowych. Niezbędna teoria jest zawarta w następnym dziale poświęconym urządzeniu FX3.

W tym projekcie wykorzystano 4 gniazda, zatem były potrzebne 4 flagi. Dwie z nich (A i C) są flagami pełnymi, a B i D flagami częściowymi.



Rys. 10. Przyporządkowanie flagi do wątku

Na nazwie flagi należy wybrać ustawienia sygnału (*DMA Flag Settings*) oraz przyporządkować jej właściwy wątek. Tabela przedstawia konfigurację wszystkich czterech flag:

Tab. X. Konfiguracja flag w GPIF II Designerze

Pin FX3	Nazwa sygnału	Logiczne "1"	Wątek
21	Flaga A	Stan niski	Thread_0_DMA_Ready
22	Flaga B		Thread_0_DMA_Watermark
23	Flaga C		Thread_3_DMA_Ready
25	Flaga D		Thread_3_DMA_Watermark

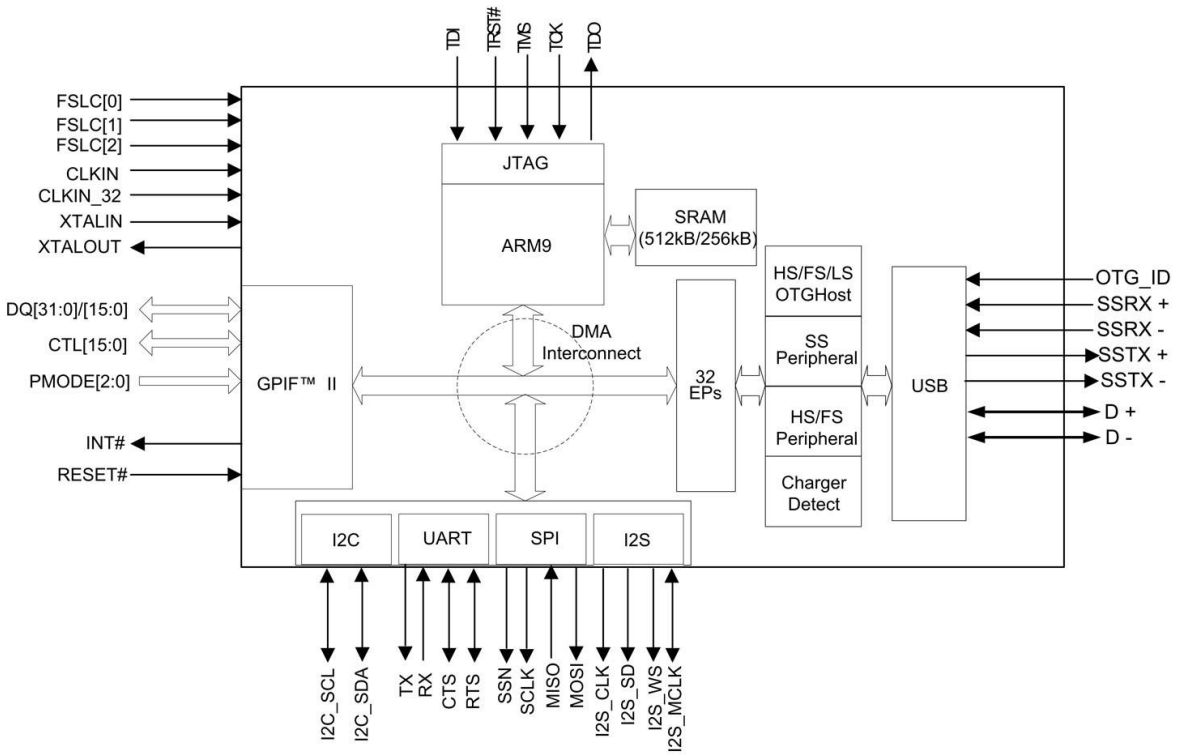
4. Cypress EZ-USB FX3

Urządzenie EZ-USB FX3 (w tej pracy często skraca się nazwę do FX3) firmy Cypress jest kontrolerem USB, który umożliwia dodanie funkcjonalności magistrali USB 3.0 do dowolnego procesora lub układu. Procesor FX3 został oparty na architekturze ARM9. Oferuje on łączność z zewnętrznymi urządzeniami poprzez magistrale zarówno szeregowe jak i równoległe.

Główną funkcją FX3 jest przesył danych o wysokiej przepustowości pomiędzy hostem USB (komputerem), a urządzeniem zewnętrznym, np. FPGA, kamera czy skaner. Moc procesora na rdzeniu ARM9 pozwala również na manipulację danymi bez większych obciążeń. Jednakże w tej pracy nie korzysta się z tych możliwości, gdyż celem nadrzędnym jest dokładna transmisja danych bez jakichkolwiek ingerencji w pakiety.

Poza standardowymi magistralami takimi jak I2C, SPI, UART czy I2S FX3 oferuje programowalną maszynę stanów GPIF II (II w nazwie oznacza drugą generację przygotowaną przez Cypressa). Dzięki temu możliwe jest podłączenie do FX3 układów programowalnych bez względu na producenta (Xilinx, Altera).

Schemat budowy FX3 zaczerpnięty z dokumentacji producenta:



Rys. 11. Schemat blokowy FX3

Opis opcje oferowanych przez kontroler USB:

- interfejs USB - kompatybilność z USB 2.0 (wszystkie prędkości, standard) oraz z USB 3.0, do 16 wejściowych oraz 16 wyjściowych endpointów
- GPIF II, szerzej omówione w rozdziale 3.3

- 32-bitowy procesor ARM926EJ-S oparty na rdzeniu ARM9, 512 kB pamięci SRAM
- interfejs JTAG - 5-pinowy interfejs umożliwiający debugowanie programu
- UART - interfejs wspiera komunikacje w trybie full-duplex (dane są przesyłane w obu kierunkach jednocześnie bez spadku transferu), sygnały TX, RX, CTS oraz RTS, prędkość od 300 bitów/s do 4608 Kb/s
- I2C - FX3 działa jako urządzenie nadrzedne (master), zatem pozwala na dołączenie urządzeń podrzędnych (slave). FX3 może pobierać program podczas startu z pamięci EEPROM, która komunikuje się poprzez magistralę I2C. Wspierane są 3 częstotliwości zegarowe: 100 kHz, 400 kHz oraz 1 MHz
- I2S - magistrala umożliwiająca przesyłanie danych audio, służy do podłączania zewnętrznych urządzeń nadawczych lub odbiorczych
- SPI - wspierana jest magistrala operująca na maksymalnej częstotliwości 33 MHz
- zasilanie - każdy blok peryferyjny może być zasilany z osobnego źródła o różnym napięciu
- zegar - wewnętrzny zegar taktuje z częstotliwością 19.2 MHz, jest możliwość dołączania zewnętrznych zegarów do 52 MHz

4.1. Teoria FX3 - wątki, gniazda, DMA, flagi oraz deskryptory

Rozdział ma za zadanie przybliżyć teoretyczną stronę realizacji interfejsu Slave Fifo. Do tego niezbędne jest zaznajomienie się z poniższymi terminami.

4.1.1. Gniazda

Gniazdo jest punktem połączeniowym pomiędzy blokiem peryferyjnym oraz pamięcią RAM FX3. Każdy blok (np. USB, GPIF, UART czy SPI) posiada określoną liczbę gniazd z nim związanych. Liczba gniazd odpowiada liczbie niezależnych danych przechodzących przez gniazdo (np. jedno gniazdo odpowiadające transferowi danych z USB do GPIF, a drugie z GPIF do USB).

4.1.2. Deskryptory DMA

Deskryptor DMA (Direct Memory Access - bezpośredni dostęp do pamięci) jest zestawem rejestrów umieszczonych w pamięci RAM FX3. Przetrzymuje on informacje o adresach, wielkości buforów DMA oraz zawiera wskaźniki do następnych deskryptorów. Komplet tych wskaźników tworzy łańcuch deskryptorów DMA.

4.1.3. Bufor DMA

Bufor DMA jest sekcją w pamięci RAM, służy do pośredniego zapisu i przetrzymywania danych przysyłanych dzięki urządzeniu FX3. Bufory są zaalokowane, podobnie jak deskryptory, w pamięci RAM przez oprogramowanie FX3, ich adresy są przechowywane w deskryptorach DMA.

4.1.4. Wątki GPIF II

Pojedynczy wątek GPIF II to dedykowana ścieżka danych w bloku maszyny stanów, który łączy zewnętrzne peryferia z GPIF-em poprzez gniazda (rys. X). Gniazdka mogą się komunikować dzięki zdarzeniom (events) oraz przerwaniom procesora FX3 (interrupts), konfiguracja odbywa się w oprogramowaniu.

Przykład: niech nastepuje transfer danych z bloku GPIF II do bloku USB. Gniazdo GPIF może poinformować gniazdo USB, że zostało zapełnione danymi w buforze DMA oraz gniazdo USB może poinformować o wolnym miejscu w dedykowanym buforze DMA. Taka implementacja jest nazwana automatycznym kanałem DMA, procesor nie bierze udziału w jakichkolwiek zmianach danych podczas przesyłu.

Jednakże gniazdo GPIF jest w stanie zasygnalizować wywołania przerwania przez procesor, gdy bufor DMA zostanie zapełniony. Wtedy procesor odpowiada za przekazanie tej informacji gniazdu USB. W odpowiedzi gniazdo USB może przekazać do procesora sygnał o wywołaniu przerwania informującego o tym, że bufor DMA jest pusty. To działanie pociąga za sobą ponowne poinformowanie gniazda GPIF przez CPU o dostępności gniazda USB. Opisana implementacja nosi nazwę manualnego kanału DMA, używany jest, gdy procesor ma za zadanie modyfikować dane.

4.1.5. Konfiguracja kanału DMA

d

4.1.6. Konfiguracja flag - pełne oraz częściowe

d

d

5. Spartan 3E

Spartan 3E to rodzina układów programowalnych FPGA firmy Xilinx, która została zaprojektowana w celu zaspokojenia rynku na potrzebę uzyskania układów o bardzo dużej pojemności w przystępnej cenie. W tej pracy wykorzystano układ XC3S500E, który posiada 10476 programowalnych bramek. Największy układ z tej rodziny posiada ich 33192, a najmniejszy 2160. XC3S500E klasyfikuje się w środkowym miejscu tabeli jeśli chodzi o zaawansowanie układu. Poniżej przedstawiono najważniejsze cechy układu oraz płytki ewaluacyjnej:

- standard logiczny: 3,3 V
- moduł Cyfrowego Zarządzania Zegarem - 4 bloki
- maksymalna ilość wejść/wyjść - 232
- możliwość konfiguracji po załączeniu zasilania za pomocą Xilinx 4 Mbit Flash PROM
- zewnętrzny zegar 50 MHz
- złącze Hirose FX2 do szybkiej transmisji danych
- cztery przełączniki wykorzystywane do zmiany trybu transmisji
- wyświetlacz LCD ze sterownikiem HD44780 działający w trybie 4-bitowym

5.1. Środowisko projektowe

Do syntezy i implementacji projektu wykorzystano pakiet WebPack ISE 14, który jest dostępny do pobrania za darmo na stronie producenta[1]. Jest to najnowsza wersja środowiska programistycznego udostępniona przez firmę Xilinx. WebPack ISE to potężny pakiet narzędziowy, który standardowo zawiera wszystkie niezbędne moduły, między innymi: syntezer języka HDL, PlanAhead (planowanie, rozkład i dostosowanie opcji pinów wejściowych/wyjściowych) czy Xilinx Impact (programowanie układu docelowego lub układu wykorzystującego plik PROM do wstępnej konfiguracji FPGA).

5.2. Utworzenie projektu, przypisanie pinów

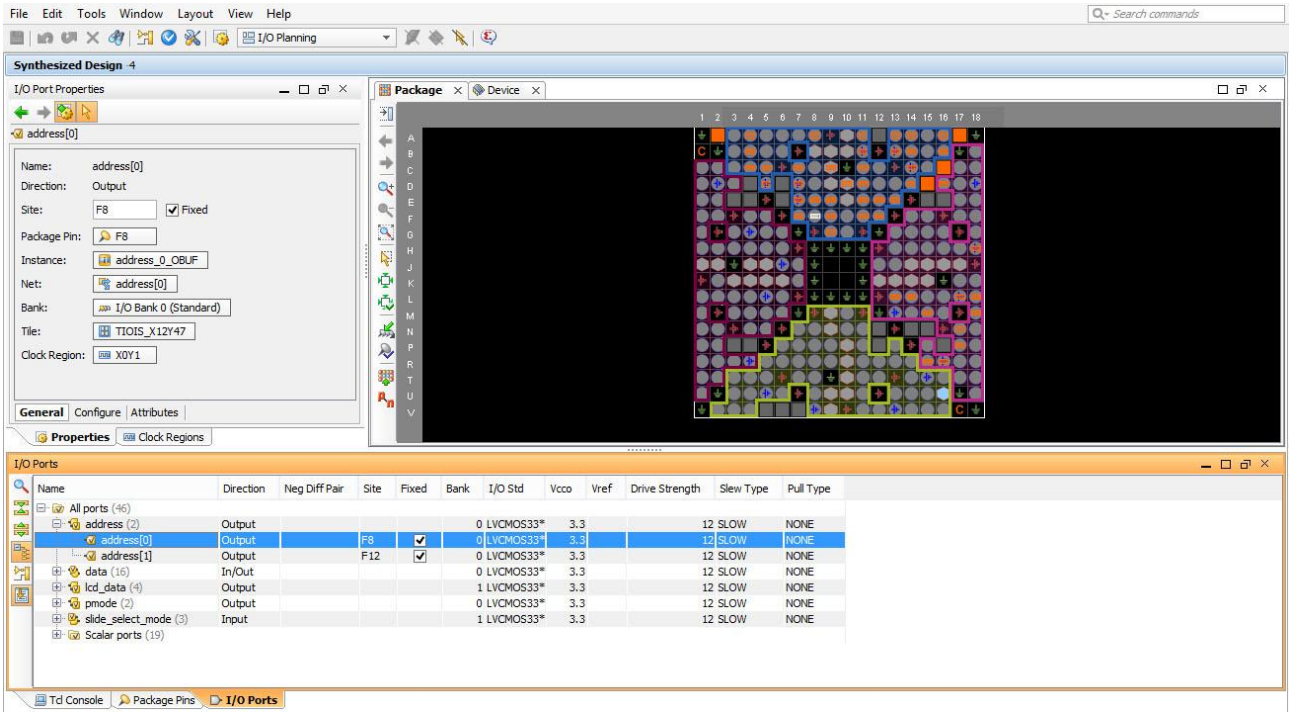
Nowy projekt utworzono wybierając opcję w menu programu *File - New Project*. Podano nazwę projektu "Slave Fifo" oraz wybrano miejsce na dysku komputera w celu ulokowania plików projektu. W opcjach szczegółowych wybrano z rozwijanej listy następujące parametry układu:

- **Rodzina**: Spartan 3E
- **Układ**: XC3S500E
- **Obudowa**: FG320
- **Prędkość**: 4
- **Synteza**: XST (VHDL/Verilog)
- **Język programowania**: VHDL

Następnie dodano do projektu pliki źródłowe z kodem napisanym w języku VHDL klikając prawym klawiszem myszy na opcję *New source* widoczną w drzewie projektu.

[1] http://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/design-tools/v14_1.html

Kolejnym ważnym krokiem w konfiguracji projektu było utworzenie pliku o rozszerzeniu ucf, który zawiera przypisania pinów do zadeklarowanych portów układu. Posłużono się modułem Xilinx PlanAhead (Rys. 1).



Rys. 12. Xilinx FlanAhead - widok programu

W części *I/O Ports* są dostępne wszystkie wejścia i wyjścia zdefiniowane w części deklaracyjnej projektu (entity). Wybierając konkretny sygnał można w zakładce I/O Port Properties dokonano przypisania sygnału. Po zaznaczeniu wszystkich sygnałów (*All ports*) i kliknięciu prawym przyciskiem myszy wybrano opcję *Configure I/O Ports*. Zmieniono domyślny standard z LVCMOS18 na LVCMOS33 (do portów doprowadzone jest znamionowe napięcie 3,3 V), moc sygnału z 4 na 12. Jedynie porty odpowiadające sygnałom dostępnym na złączu Hirose FX2 (wykorzystywane do komunikacji z kontrolerem USB) posiadają prędkość Fast, reszta portów działa w standardowym trybie Slow. Po zapisaniu ustawień moduł automatycznie wygeneruje plik ucf, który jest niezbędny do syntezy projektu. Poniżej zamieszczono tabelę z przypisaniami każdego użytego pinu.

Tab. 1. Przypisania pinów

Nazwa sygnału	Przypisanie	Opis
clock50	C9	Wewnętrzny zegar 50 MHz
reset_from_slide	N17	Reset układu
slide_select_mode[0]	L13	Wybór trybu transmisji
slide_select_mode[1]	L14	jw.
slide_select_mode[2]	H18	jw.
led_buffer_empty_show	F9	Status pamięci FIFO
lcd_e	M18	Sygnał Enable LCD
lcd_rs	L18	Sygnał Register Select LCD
lcd_rw	L17	Sygnal Read/Write LCD
lcd_sratаflash_disable	D16	Wyłączenie modułu FLASH LCD

lcd_data[0]	R15	4-bitowa linia danych
lcd_data[1]	R16	jw.
lcd_data[2]	P17	jw.
lcd_data[3]	M15	jw.
clock100_out	D10	Sygnał zegarowy do FX3
flaga	A13	Flaga inf. o stanie buforu FX3
flagb	C5	jw.
flagc	E7	jw.
flagd	F7	jw.
pktend	D5	Informacja o krótkim pakiecie
sloe	C7	Sygnał Output/Enable FX3
slwr	D7	Sygnał Write FX3
slcs	A4	Sygnał Chip Select FX3
slrd	B4	Sygnał Read FX3
address[0]	F8	Adresowanie wątków FX3
address[1]	F12	jw.
data[0]	A14	Linia danych Slave FIFO
data[1]	C14	jw.
data[2]	D14	jw.
data[3]	B14	jw.
data[4]	E11	jw.
data[5]	A16	jw.
data[6]	E9	jw.
data[7]	C11	jw.
data[8]	B13	jw.
data[9]	G9	jw.
data[10]	B16	jw.
data[11]	B11	jw.
data[12]	C4	jw.
data[13]	A11	jw.
data[14]	E13	jw.
data[15]	A8	jw.
pmode[0]	D11	Sygnały do debugowania
pmode[1]	F11	jw.
reset_to_fx3	E8	Sygnał resetujący FX3z FPGA
reset_from_fx3	E12	Sygnał resetujący FPGA z FX3

5.3. Cyfrowe Zarządzanie Zegarem (DCM - Digital Clock Manager)

FPGA jest głównym układem (master) w interfejsie Slave Fifo, więc kontroler EZ-USB FX3 jest układem podrzędnym (slave). Aby przesyłanie danych przebiegało bezproblemowo firma Cypress zaleca następującą konfigurację:

- EZ-USB FX3 powinien być taktowany zegarem o częstotliwości 400 MHz, pociąga to za sobą skonfigurowanie pętli PLL, aby przekształciła doprowadzony sygnał zegarowy 27 MHz do docelowej wartości
- doprowadzenie do programowalnej maszyny stanów GPIF II sygnału zegarowego z układu głównego (FPGA) o częstotliwości 100 MHz

Pierwszy podpunkt wykonuje się za pomocą kilku instrukcji wykorzystując API Cypressa (szerzej jest to omówione w poprzednim rozdziale). Natomiast uzyskanie 100 MHz z 50 MHz (tyle wynosi częstotliwość zegara na płytce uruchomieniowej FPGA) wiąże się z wykorzystaniem gotowego modułu przygotowanego przez firmę Xilinx, a mianowicie DCM (Digital Clock Manager - Cyfrowe Zarządzanie Zegarem).

Pojedynczy DCM składa się z 4 części: syntezer częstotliwości DFS (Digital Frequency Synthesier), pętli DLL (Delay Locked Loop), programowanego przesuwnika fazy (Phase Shifter) oraz zespołu logiki (Status Logic). Moduł zapewnia integrację zaawansowanych możliwości taktowania układu programowalnego. Co za tym idzie, boki DCM w rodzinie Spartan 3 rozwiązują wiele problemów (zwłaszcza związanych z uzyskiwaniem dużych częstotliwości):

- mnożenie lub dzielenie doprowadzonego sygnału zegarowego - synteza kompletnie nowej wartości częstotliwości
- poprawianie zbocza sygnału, eliminacja efektu propagacji
- kontrola 50% wypełnienia
- przesuwanie fazy sygnału

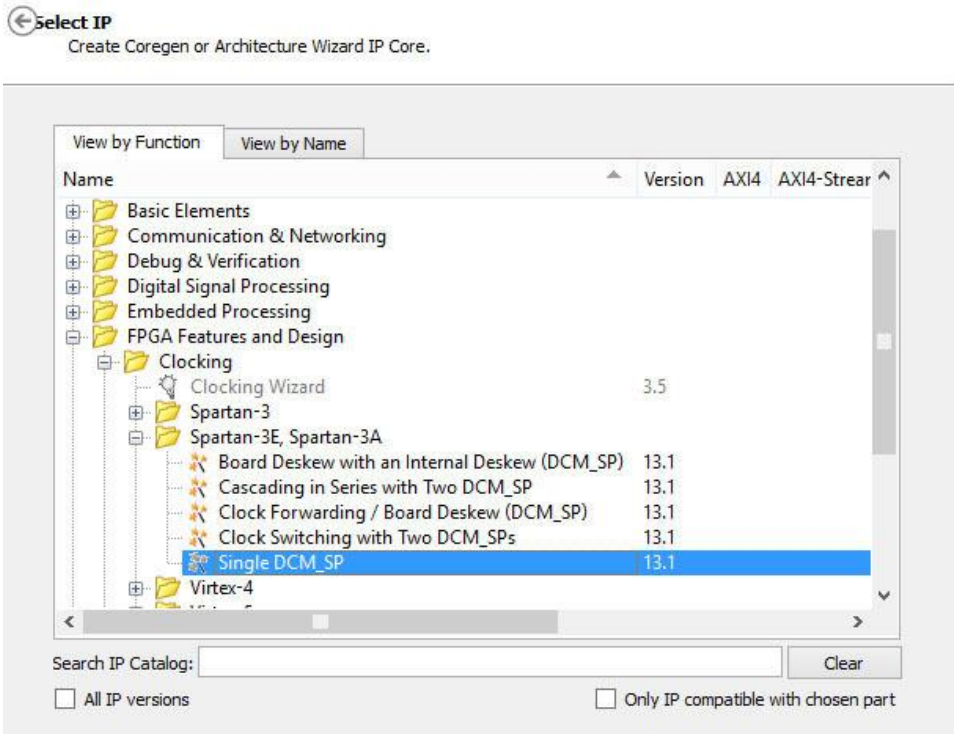
Celem wykorzystania modułu DCM było uzyskanie 100 MHz, a zatem częstotliwości dwa razy większej od oryginalnej. Nie skorzystano zatem z opcji wyboru mnożników i dzielników, a zaznaczono opcję CLK2FX - jest to opcja znacznie wygodniejsza. Proces konfiguracji DCM (rys. 2 oraz rys 3):

- dodanie nowego źródła w drzewie projektu
- wybór opcji IP (CORE Generator & Architecture Wizard)
- FPGA Features and Design - Clocking - Spartan 3E - Single DCM_SP
- podanie wartości "50 MHz" w polu Input Clock Frequency, zaznaczenie opcji "zewnętrzny zegar" (CLKIN Source - External)
- Zaznaczenie opcji CLK2X oraz RST

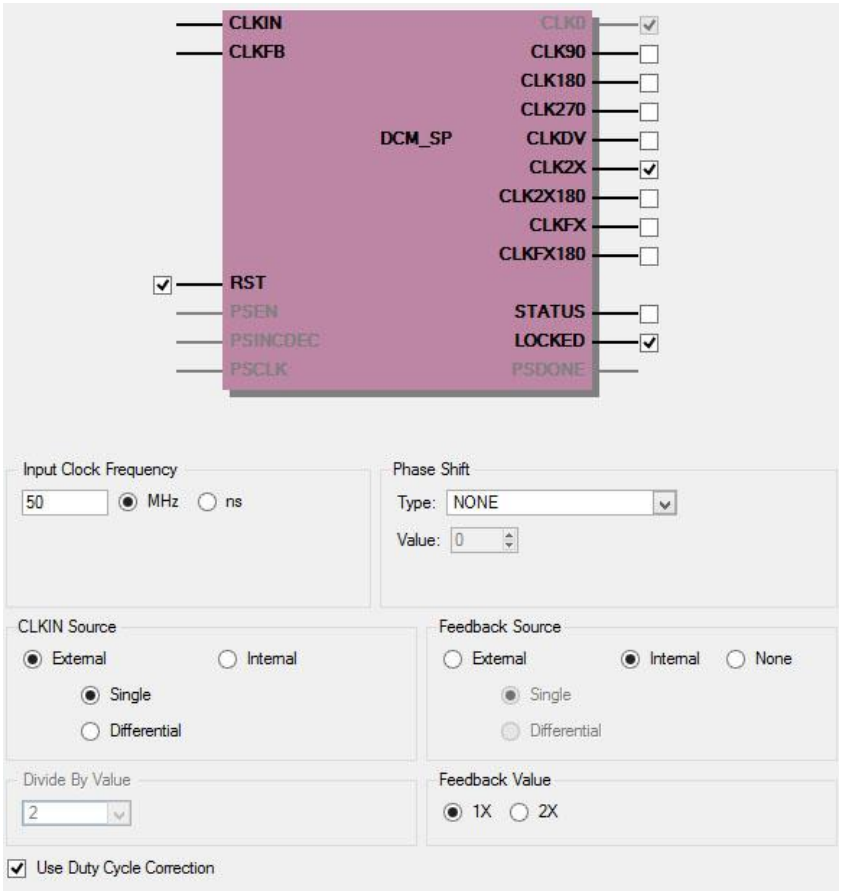
Po wygenerowaniu modułu należy zaznaczyć go i wybrać z listy *View HDL Instantiation Template* w celu uzyskania kodu, który można dołączyć do głównego. Są to sekcje *component* (deklaracja jednostki projektowej) oraz *port map* (mapowanie sygnałów). W drugiej sekcji dokonujemy przypisywania wchodzących oraz wychodzących sygnałów do konkretnego modułu.

Tab. 2. Opis sygnałów modułu DCM

Nazwa sygnału	Typ	Opis sygnału
CLKIN_IN	wejście	zewnętrzny zegar 50 MHz
RST_IN	wejście	resetowanie modułu
CLKIN_IBUFG_OUT	wyjście	globalny bufor
CLK0_OUT	wyjście	50 MHz po korekcji zbocza
CLK2X_OUT	wyjście	100 MHz po korekcji zbocza



Rys. 13. Wybór DCM z listy IP Cores



Rys. 14. Konfiguracja bloku DCM

5.4. Pamięć FIFO

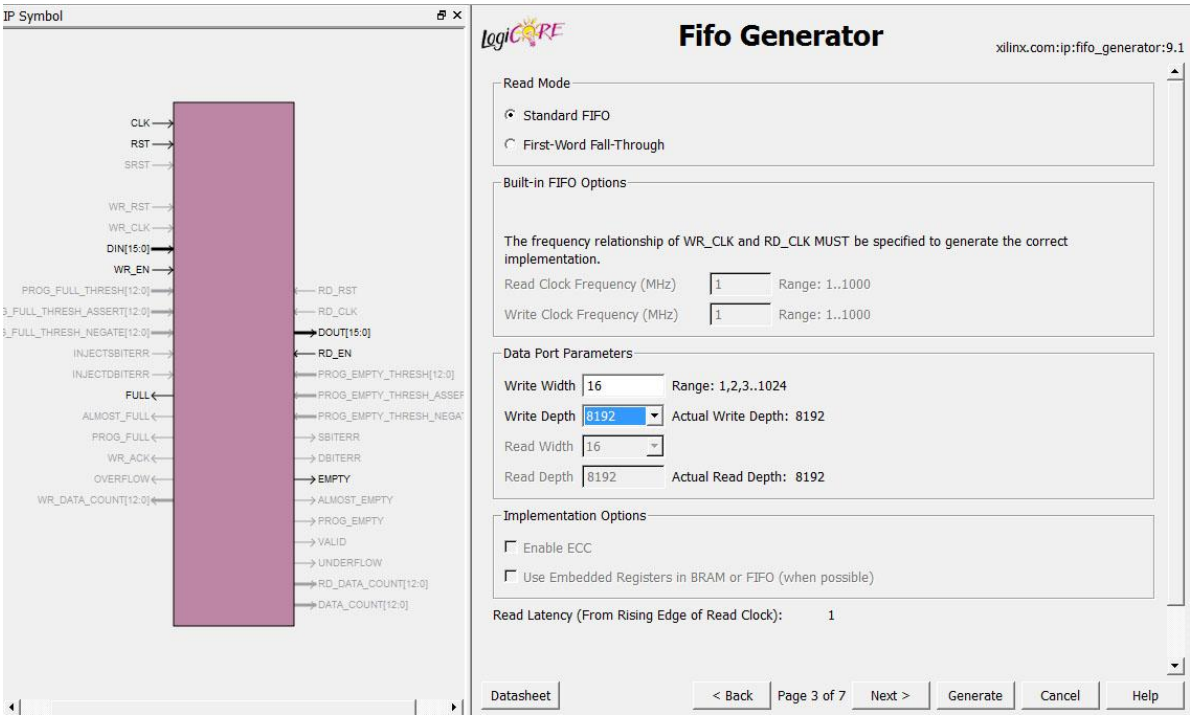
W celu implementacji pętli (Loopback Transfer, czyli wysyłanie i odbiór identycznych pakietów danych w komputerze) niezbędna jest pamięć FIFO (First In, First Out). Posłużono się modułem *Fifo Generator* zrealizowanym przez firmę Xilinx dostępnym (podobnie jak DCM) w katalogu IP Core. Konfiguracja (Rys. 4) przebiegła następująco:

- dodanie nowego źródła w drzewie projektu, wybranie *IP Cores* oraz podanie nazwy modułu
- wybranie *Memories & Storage Elements - Fifo Generator* w wersji 9.1
- port wejściowy oraz wyjściowy mają identyczną pojemność: szerokość 16 bitów, głębokość 8192 bitów

Po zatwierdzeniu ustawień postąpiono analogicznie jak w przypadku DCM-a, czyli skopiowano do pliku zawierającego transfer pętli wygenerowany kod. Sygnały modułu:

Tab. 3. Opis sygnałów modułu pamięci FIFO

Nazwa sygnału	Typ	Opis sygnału
clk	wejście	zegar 100 MHz
rst	wejście	resetowanie pamięci
din	wejście	16-bitowe wejście na dane
wr_en	wejście	pozwolenie na zapis danych
rd_en	wejście	pozwolenie na odczyt danych
dout	wyjście	16-bitowe wyjście danych
full	wyjście	sygnalizacja pełnej pamięci
empty	wyjście	sygnalizacja pustej pamięci



Rys. 15. Konfiguracja pamięci FIFO

5.5. Ogólna struktura programu

Program napisano w języku opisu sprzętu VHDL. Został on podzielony na indywidualne moduły, każdy z nich jest maszyną o skończonej maszynie stanów:

- slave_fifo_main
- lcd_controller
- slave_fifo_stream_write_to_fx3
- slave_fifo_stream_read_from_fx3
- slave_fifo_loopback

Jak zostało wcześniej nadmienione, doprowadzamy sygnał zegarowy o częstotliwości 50 MHz. Wybór transmisji wybieramy za pomocą trzech przełączników umiejscowionych po prawej stronie wyświetlacza LCD. Czwarty przełącznik służy do resetowania układu. Poniżej przedstawiono tabelę z możliwymi ustawieniami przełączników.

Tab. X. Wybór trybu działania przez przełączniki

	Przełącznik 4	**Przełącznik 3**	**Przełącznik 2**	**Przełącznik 1**	**Opis działania**
Stan	Niski	Niski	Niski	Niski	Stan bezczynności
	Niski	Niski	Niski	Wysoki	Pętla
	Niski	Niski	Wysoki	Niski	Ciągłe wysyłanie danych z PC do FPGA
	Niski	Wysoki	Niski	Niski	Ciągłe wysyłanie danych z FPGA do PC
	Wysoki	Niski	Niski	Niski	Reset
	Wysoki	Wysoki	Dowolny	Dowolny	Reset
	Niski	Wysoki	Wysoki	Dowolny	Stan bezczynności

Wyświetlacz LCD informuje użytkownika, w którym module obecnie się znajduje. Ponadto dla trzech trybów również wyświetla odbierane/wysyłane dane (16 bitów).

Kolejne podrozdziały mają za zadanie zaprezentować sposób działania każdego modułu.

5.5.1. Główny program - Slave Fifo Main

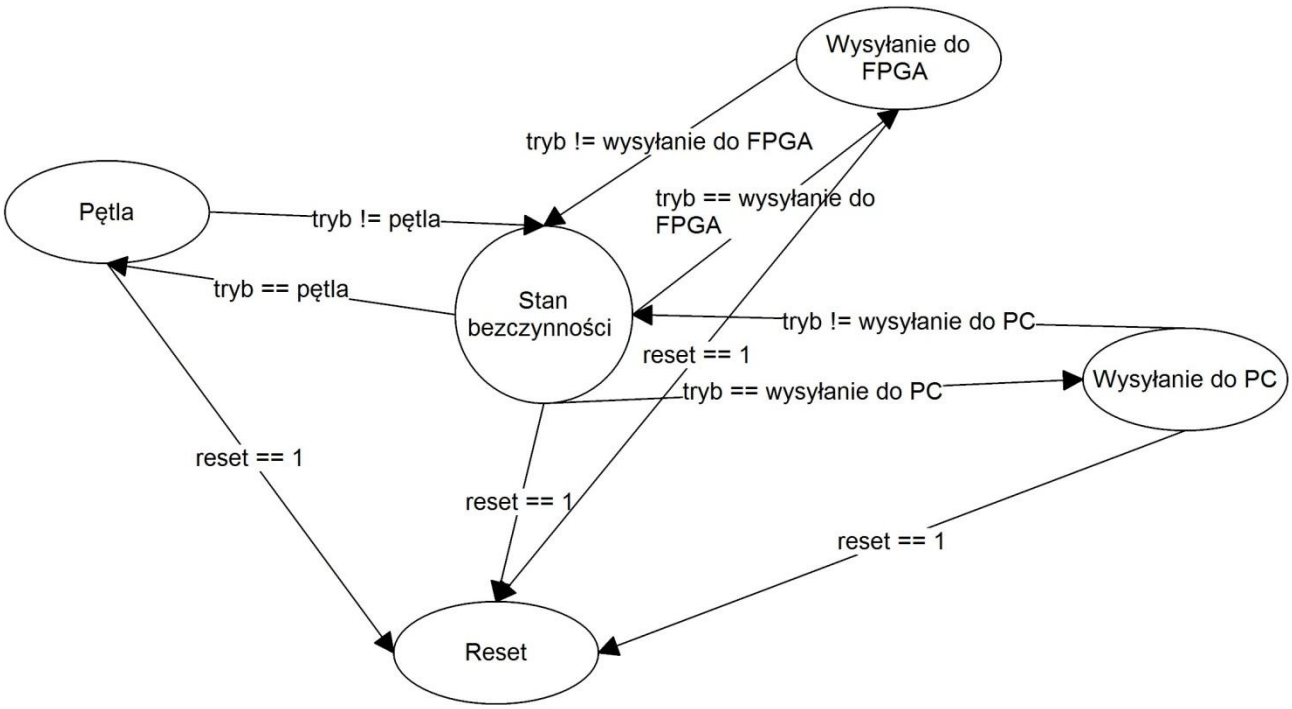
Główny program zaczyna się od deklaracji jednostki (entity). Zdefiniowane są wszystkie sygnały wyjściowe i wejściowe (opisane wcześniej w tab. X). W architekturze zdefiniowane są stałe, pomocnicze sygnały oraz zmienne maszyny stanów. Następnie zostały zmapowane wszystkie komponenty. Napisano funkcję *vector_to_string*, która ma za zadanie konwersję zmiennej typu *std_logic_vector* na *string*. Jest ona wykorzystywana do konwersji wchodzących lub wychodzących danych na format umożliwiający wyświetlanie dwóch bajtów w formie pojedynczych bitów na ekranie LCD.

W sekcji głównej programu znalazły się procesy, które w zależności od wybranej transmisji danych manipulują stanami głównych sygnałów, tzn. sloe, slrd, slcs czy sloe (ich znaczenie zostało omówione w rozdziale trzecim). Pozostałe procesy odpowiadają odbieraniu lub wysyłaniu danych i monitorowaniu flag maszyny stanów GPIF II. Na końcu programu umieszczono główną maszynę stanów odpowiadającą za przechodzenie do wybranego trybu transmisji.

Stanu (zmienne) głównego programu:

- idle_state (stan bezczynności)
- loopback_state (pętla)
- stream_read_from_fx3_state (ciągłe wysyłanie danych do FPGA)
- stream_write_to_fx3_state (ciągłe wysyłanie danych do PC)

Schemat graficzny przedstawia zasadę działania.



Rys. 16. Maszyna stanów głównego programu

Analizując powyższy diagram można dojść do wniosku, że nie można przejść z trybu np. ciągłego wysyłania danych z PC do FPGA do pętli pomijając stan bezczynności (idle). Adres aktualnego wątku jest zależny od trybu, w którym znajduje się maszyna stanów. Jedynym wyjątkiem jest przejście do stanu resetu, przebiega ono asynchronicznie i niezależnie od obecnego trybu pracy FPGA.

W programie głównym znajduje się jeszcze obsługa ekranu LCD, która została również podzielona na poszczególne etapy, a co za tym idzie - użyto maszyny stanów. Oto one:

- idle (stan bezczynności)
- start (wyzerowanie licznika)
- clearscr (wyczyszczenie ekranu z poprzedniego napisu oraz wyzerowanie licznika)

- move1 (przejście do 1 miejsca w 1 linii)
- send1 (wysłanie 2 bajtów danych)
- move2 (przejście do miejsca w 2 linii)
- send2 (wysłanie 2 bajtów danych)

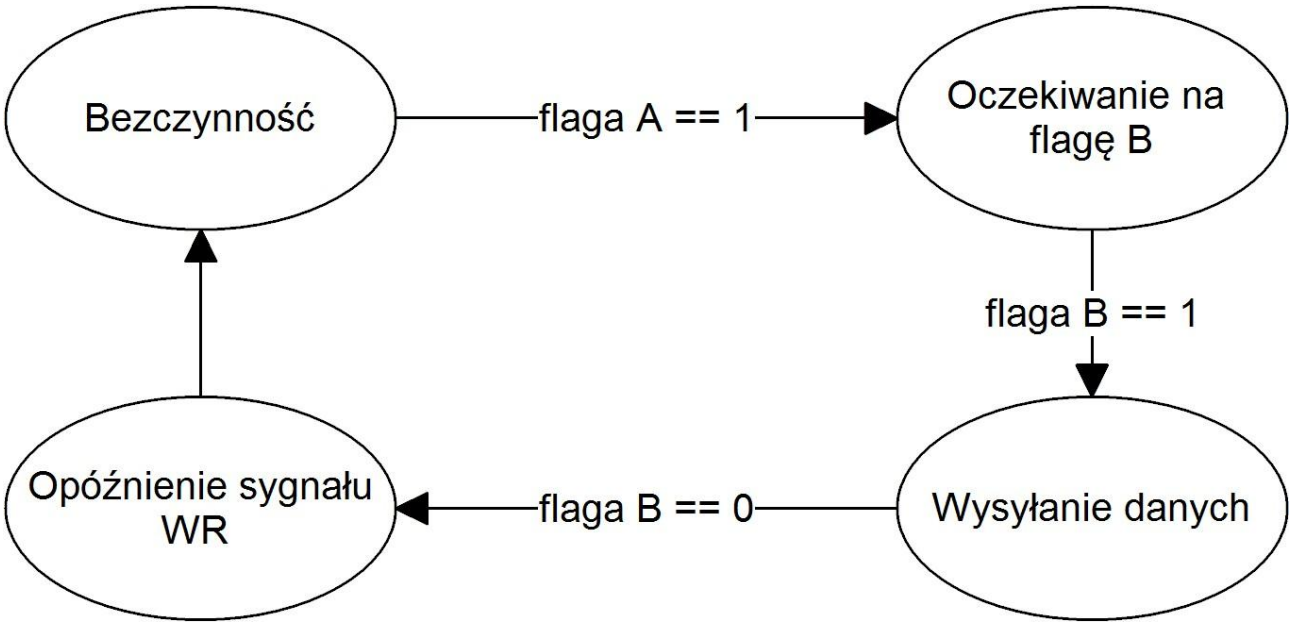
Za dokładną obsługę 4-bitowej linii danych oraz pozostałych sygnałów ekranu LCD odpowiada moduł *lcd_controller*.

5.5.2. Ciągłe wysyłanie danych do komputera - Stream Write to FX3

Po deklaracji jednostki projektowej, stałych i sygnałach pomocniczych znajduje się deklaracja maszyny stanów odpowiadająca za poprawne działanie trybu ciągłej wysyłki danych do komputera.

- idle
- wait_flagb
- write
- wr_delay

Działanie prezentuje poniższy diagram:



Rys. 17. Schemat działania ciągłego zapisu do PC

Maszyna stanów odpowiada za monitorowanie sygnałów flag pochodzących z kontrolera USB (dostępność buforów DMA) oraz zmienia stany sygnałów związanych z transmisją danych.

W tym trybie manipuluje się jedynie stanem slwr, czyli sygnałem odpowiedzialnym za zapis danych.

- slcs = 0 (wartość stała)
- sloe = 1 (wartość stała)
- slrd = 1 (wartość stała)

- slwr = 0 gdy znajdujemy się w stanie stream_in_write oraz flaga B jest równa 1, w innych przypadkach slwr = 1

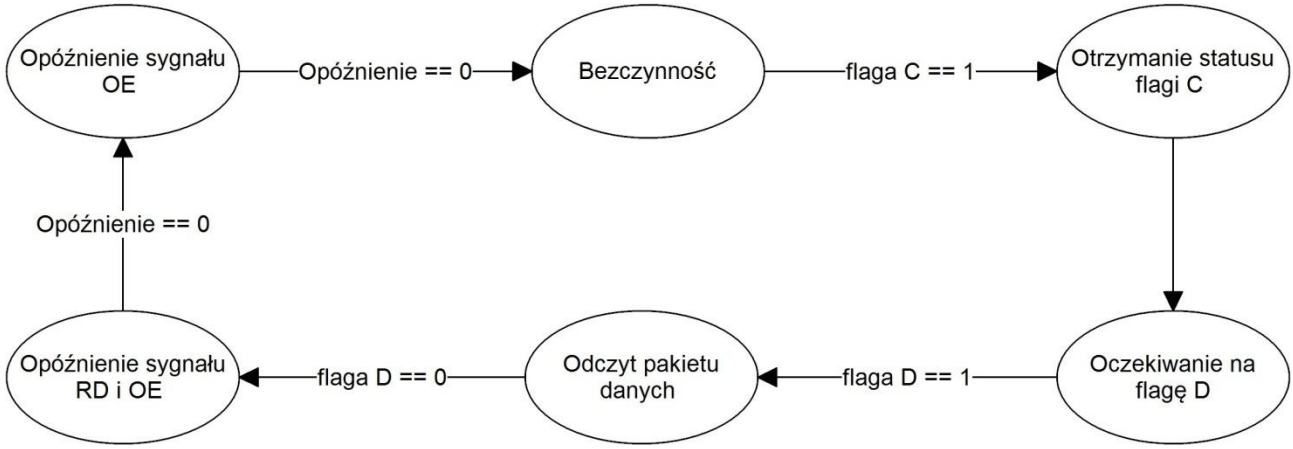
Postanowiono, że program będzie wysyłał do komputera identyczny pakiet danych zapisany w formacie bitowym: 0101011101001101. Gdy przekształci się tę liczbę na format szesnastkowy otrzyma się 0x574D. Liczba 0x57 odpowiada literze "W", a 0x4D literze "M" w kodzie ASCII.

5.5.3. Ciągłe odbieranie danych do komputera - Stream Read from FX3

Analogicznie do poprzednich działów ten tryb ma własną maszynę stanów. Oto poszczególne etapy:

- idle
- flagc_rcvd
- wait_flagd
- read
- read_rd_oe_delay
- read_oe_delay

Zasadę działania jest zamieszczona w poniższym diagramie:



Rys. 18. Schemat ciągłego odbierania danych z PC

Odczytany pakiet danych jest wyświetlany w czasie rzeczywistym na wyświetlaczu LCD, jednak nie jest on nigdzie zapisywany (pamięć FIFO jest użyta tylko w trybie pętli).

Na osoby akapit zasługuje omówienie opóźnień generowanych po odczycie danych. Zgodnie z zasadami (opisanymi w dokumentacji firmy Cypress odnośnie interfejsu Slave FIFO) w przypadku użycia częściowych flag jest konieczne, aby FX3 monitorował sygnał Read jeszcze przez kolejne 3 takty zegara (oe_delay) po standardowym opóźnieniu 2 cykli zegara (rd_oe_delay).

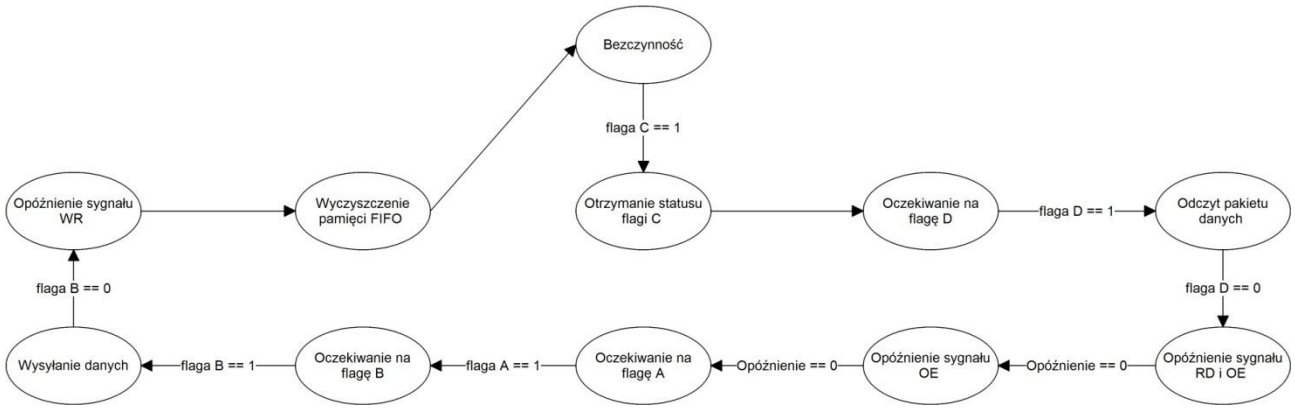
Przebywając w tym trybie program modyfikuje stany tylko dwóch sygnałów odpowiedzialnych za zapis, są to:

- slcs = 0 (wartość stała)
- slwr = 1 (wartość stała)

- sloe = 0 gdy znajdujemy się w stanie read oraz w obu opóźnieniach, w pozostałych przypadkach sloe = 1
- slrd = 0 gdy znajdujemy się w stanie read oraz w pierwszym opóźnieniu (rd_oe_delay), w pozostałych przypadkach slrd = 1

5.5.4. Pętla - Loopback Transfer

Ostatni z zaimplementowanych rodzajów przesyłania danych jako jedyny wykorzystuje (omówioną w rozdziale 5.4) pamięć FIFO. Pakiet danych odebrany z magistrali USB jest zapisywany w buforze danych, a następnie przy zwolnieniu się miejsca w kanałach DMA następuje pobranie tego pakietu, wyczyszczenie pamięci oraz odesłanie danych z powrotem do komputera. Pętla jest połączeniem dwóch poprzednich transmisji, zatem diagram jest najbardziej rozbudowany.



Rys. 19. Schemat maszyny stanów pętli

Maszyna stanów w kodzie VHDL zawiera wymienione poniżej elementy:

- idle
- flagc_rcvd
- wait_flagd
- read
- read_rd_oe_delay
- read_oe_delay
- wait_flaga
- wait_flagb
- write
- write_wr_delay
- flush_fifo

W tym przypadku monitoruje się wszystkie cztery flagi dostępu do buforów DMA oraz manipuluje stanami trzech sygnałów:

- slcs = 0 (wartość stała)
- slwr = 0 gdy znajdujemy się w stanie stream_in_write oraz flaga B jest równa 1, w innych przypadkach slwr = 1

- sloe = 0 gdy znajdujemy się w stanie read oraz w obu opóźnieniach, w pozostałych przypadkach sloe = 1
- slrd = 0 gdy znajdujemy się w stanie read oraz w pierwszym opóźnieniu (rd_oe_delay), w pozostałych przypadkach slrd = 1

Zasada stosowania opóźnień po odczycie pakietu jest identyczna jak w przypadku transmisji ciągłego odczytu danych.

6. Przedstawienie wyników oraz efektów działania interfejsu Slave Fifo

ddd