AKADEMIA GÓRNICZO-HUTNICZA IM. STANISŁAWA STASZICA W KRAKOWIE

**WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI,
INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ**

Praca dyplomowa

Implementacja algorytmów cyfrowego przetwarzania sygnałów w układzie heterogenicznym Intel Cyclone V SoC

Implementation of digital signal processing algorithms in the heterogeneous Intel Cyclone V SoC system

Autor: *Dominik Bachurski*
Kierunek studiów: *Mikroelektronika w Technice i Medycynie*
Opiekun pracy: *dr inż. Paweł Skrzypiec*

Kraków, 2024

Uprzedzony o odpowiedzialności karnej na podstawie art. 115 ust. 1 i 2 ustawy z dnia 4 lutego 1994 r. o prawie autorskim i prawach pokrewnych (t.j. Dz.U. z 2006 r. Nr 90, poz. 631 z późn. zm.): „Kto przywłaszcza sobie autorstwo albo wprowadza w błąd co do autorstwa całości lub części cudzego utworu albo artystycznego wykonania, podlega grzywnie, karze ograniczenia wolności albo pozbawienia wolności do lat 3. Tej samej karze podlega, kto rozpowszechnia bez podania nazwiska lub pseudonimu twórcy cudzy utwór w wersji oryginalnej albo w postaci opracowania, artystycznego wykonania albo publicznie zniekształca taki utwór, artystyczne wykonanie, fonogram, wideogram lub nadanie.", a także uprzedzony o odpowiedzialności dyscyplinarnej na podstawie art. 211 ust. 1 ustawy z dnia 27 lipca 2005 r. Prawo o szkolnictwie wyższym (t.j. Dz. U. z 2012 r. poz. 572, z późn. zm.): „Za naruszenie przepisów obowiązujących w uczelni oraz za czyny uchybiające godności studenta student ponosi odpowiedzialność dyscyplinarną przed komisją dyscyplinarną albo przed sądem koleżeńskim samorządu studenckiego, zwanym dalej «sądem koleżeńskim».", oświadczam, że niniejszą pracę dyplomową wykonałem(-am) osobiście i samodzielnie i że nie korzystałem(-am) ze źródeł innych niż wymienione w pracy.

Serdecznie dziękuję . . . tu ciąg dalszych podziękowań np. dla promotora, żony, sąsiada itp.

Spis treści

Spis symboli i skrótów

DSP Digital Signal Processing

DSP Digital Signal Processor

FPGA Field Programmable Gate Array

HPS Hard Processor System

FFT Fast Fourier Transform

DMA Direct Memory Access

OCM On-Chip Memory

FIR Finite Impulse Response

LUT Look Up Table

ARM Advanced RISC Machine

CPU Central Processing Unit

GPU Graphics Processing Unit

SoC System on Chip

AVMM Avalon Memory-Mapped

AVST Avalon Streaming

TEA Tiny Encryption Algorithm

Wstęp

Wprowadzenie

Cyfrowe przetwarzanie sygnałów, DSP jest dziedziną inżynierii zajmującą się metodami przetwarzania oraz analizy sygnałów cyfrowych. Źródłem sygnałów cyfrowych mogą być między innymi przetworniki analogowo-cyfrowe (A/C) oraz generatory sygnałów. Najpopularniejszymi przykładami sygnałów analogowych konwertowanych do domeny cyfrowej są sygnały odbierane ze źródeł audio oraz sygnały wideo.

Algorytmy przetwarzania sygnałów znajdują szerokie zastosowanie w wielu dziedzinach techniki, takich jak telekomunikacja (np. w systemach komunikacji mobilnej), obróbka dźwięku (np. w equalizerach) oraz bezpieczeństwo danych (np. szyfrowanie). Przykładowe algorytmy wykorzystywane we wspomnianych zastosowaniach obejmują transformacje takie jak Fast Fourier Transform (FFT), umożliwiające analizę sygnałów w dziedzinie częstotliwości; filtry DSP, zarówno liniowe, jak i nieliniowe, stosowane do eliminacji zakłóceń oraz optymalizacji jakości sygnału; oraz algorytmy szyfrowania, które zabezpieczają dane podczas transmisji.

Algorytmy przetwarzania sygnałów, mimo upływu lat (prace nad cyfrowym przetwarzaniem sygnałów rozpoczęły się w latach 50. XX wieku), pozostają tematem bardzo aktualnym, budzącym ciekawość wśród naukowców i inżynierów. Zastosowania DSP rozwijają się dynamicznie, obejmując coraz nowsze obszary, co czyni tę dziedzinę niezbędną w rozwoju współczesnej technologii.

Cel pracy

Celem niniejszej pracy była konstrukcja systemu do przetwarzania sygnałów cyfrowych w opraciu o system heterogeniczny, łączący w jednym układzie scalonym mikroprocesor ARM oraz układ FPGA. Wykorzystanie takiego rozwiązania umożliwiło dystrybucję zadań pomiędzy aplikacje zarządzane przez system operacyjny Linux oraz układ cyfrowy zaimplementowany w FPGA.

Motywacją do realizacji projektu była potrzeba weryfikacji możliwości wykorzystania systemu heterogenicznego do przetwarzania sygnałów pochodzących z np. przetworników analogowo-cyfrowych (A/C) w czasie rzeczywistym. W wielu zastosowaniach przetwarzanie sygnałów wymaga możliwości dostosowania architektury systemu do specyficznych potrzeb aplikacji, czego mogą nie oferować inne

rozwiązania oparte np. o mikrokontrolery lub procesory sygnałowe, Digital Signal Processor (DSP). Ponadto dzięki integracji układu FPGA z HPS, użytkownik ma możliwość zdalnego zarządzania systemem oraz monitorowania przetwarzanych sygnałów, co zwiększa elastyczność i wygodę pracy. Umożliwia to szybkie dostosowywanie parametrów przetwarzania do konkretnych potrzeb, co jest istotne w wielu aplikacjach.

System ten ma na celu nie tylko zaprezentowanie w praktyce możliwości przetwarzania sygnałów, ale również stworzenie podstawy do dalszego rozwoju o kolejne funkcjonalności, takie jak implementacja nowych algorytmów DSP, rozszerzenie interfejsu użytkownika, czy integracja z innymi systemami przetwarzania danych.

Zakres pracy

Zakres pracy obejmował:

- opracowanie systemu do transmisji danych zaimplementowanego w FPGA;
- implementację w systemie algorytmów DSP;
- przygotowanie środowiska uruchomieniowego w Yocto;
- stworzenie oprogramowania w języku Python;
- zaprojektowanie aplikacji internetowej.

W projekcie opracowano wydajny system transmisji danych oparty na FPGA, który umożliwia przesył i odczyt danych pomiędzy HPS a pamięciami w FPGA, z wykorzystaniem bezpośredniego dostępu do pamięci, Direct Memory Access (DMA). Implementacja obejmowała algorytmy DSP, w tym filtr Finite Impulse Response (FIR) z możliwością dynamicznej zmiany współczynników, moduł FFT oraz algorytm enkrypcji danych. Stworzone oprogramowanie w języku Python, oparte na bibliotece unittest, umożliwiało testowanie i weryfikację funkcjonalności systemu, a także obsługę zadań po stronie FPGA. Ponadto zaprojektowano aplikację internetową, która umożliwia użytkownikowi zdalną interakcję z systemem.

Struktura pracy

1. Wprowadzenie

1.1. Systemy heterogeniczny

Historia systemów heterogenicznych sięga końca XX wieku, gdy projektanci zaczęli dostrzegać potencjał połączenia różnych typów jednostek obliczeniowych w jednym systemie, aby uzyskać wyższą efektywność i wydajność. Początkowo były to systemy zawierające różne procesory, takie jak procesory ogólnego przeznaczenia (CPU) i procesory sygnałowe (DSP), które współpracowały, by przyspieszyć intensywne obliczenia związane z przetwarzaniem sygnałów. W kolejnych dekadach nastąpił rozwój FPGA oraz akceleratorów GPU, co pozwoliło na dalszą poprawę wydajności systemów obliczeniowych i wykorzystywanie ich w coraz bardziej zaawansowanych zastosowaniach. Taka integracja pozwala na maksymalizację efektywności wykonywania specyficznych zadań obliczeniowych, wykorzystując moc poszczególnych jednostek w optymalny sposób.

Architektura heterogeniczna powstała z potrzeby radzenia sobie z ograniczeniami wydajnościowymi i energetycznymi, które napotykają klasyczne systemy jednorodne. Jednoukładowe systemy mikroprocesorowe często okazują się niewystarczające w kontekście intensywnych obliczeń, wymagających zarówno dużej szybkości, jak i minimalnego zużycia energii. Systemy heterogeniczne, poprzez elastyczne przydzielanie zasobów, pozwalają uzyskać optymalne efekty energetyczne i wydajnościowe. Kluczowym aspektem ich projektowania jest efektywna współpraca pomiędzy różnymi komponentami, co wymaga odpowiedniego zarządzania pamięcią, synchronizacji oraz skutecznej komunikacji między jednostkami obliczeniowymi.

Wykorzystanie systemów heterogenicznych przynosi liczne korzyści, ale również wyzwania związane z programowaniem i optymalizacją. Tworzenie oprogramowania dla różnych typów architektur wymaga zrozumienia specyfiki działania każdego komponentu oraz wiedzy o tym, jak je obsłużyć, co może skomplikować proces. Rozwój systemów heterogenicznych wpływa również na architekturę nowych urządzeń i sposób ich projektowania. Integracja wielu typów układów w jednym chipie, jak w przypadku SoC, pozwala na miniaturyzację oraz bardziej efektywne zarządzanie zasobami, co sprawia, że technologia heterogeniczna ma potencjał, by stać się standardem w przyszłości, zwłaszcza w sektorach wymagających wysokiej wydajności i niskiego zużycia energii.

1.1.1. System on Chip (SoC)

System on Chip (SoC) to zintegrowany układ scalony, który łączy wszystkie elementy kompletnego systemu na jednym chipie. SoC obejmuje procesor oraz inne jednostki obliczeniowe, takie jak FPGA, DSP, GPU które umożliwiają heterogeniczne przetwarzanie danych, czyli współpracę różnych typów komponentów obliczeniowych w jednym systemie. Dodatkowo SoC często zawiera wbudowane moduły komunikacyjne, które zapewniają wymianę danych z innymi urżadzeniami np. poprzez interfejsy UART, USB, SPI czy I2C. Ponadto SoC posiadają wiele układów pomocniczych, które zwiększają funkcjonalność systemu. Są to między innymi:

– Układy zarządzania energią, które zapewniają efektywne zasilanie i minimalizują zużycie energii.

– Przetworniki analogowo-cyfrowe (A/D) oraz cyfrowo-analogowe (D/A), które umożliwiają interakcję z komponentami analogowymi.

– Pamięci takie jak RAM i pamięci nieulotne (np. flash).

1.1.2. FPGA

FPGA to programowalny układ scalony, który można dynamicznie rekonfigurować, co czyni go idealnym do szybkiego prototypowania, emulacji oraz eksploracji nowych architektur. Oferuje wyższą elastyczność w porównaniu do ASIC, a także charakteryzuje się lepszą oszczędnością energetyczną w porównaniu do GPU przy zachowaniu zadowalającej wydajności obliczeniowej.

Układy FPGA w odróżnieniu od procesorów, oferują możliwość jednoczesnego przetwarzania wielu strumieni danych, co pozwala na znaczne przyspieszenie złożonych operacji, zniwelowanie opóźnień oraz wyższą przepustowość systemu. Równoległe przetwarzanie sprawia, że FPGA przodują w zastosowaniach, gdzie przetwarzanie wymaga jednoczesnej analizy wielu strumieni danych, np. w DSP.

Ważnym atutem FPGA jest precyzyjna kontrola zegara i deterministyczne przetwarzanie. Języki opisu sprzętu, takie jak VHDL czy SystemVerilog, umożliwiają projektantom dokładne określenie momentów wykonywania operacji, co zapewnia przewidywalność. Dzięki temu FPGA gwarantują deterministyczność w aplikacjach wymagających precyzyjnego zarządzania czasem, co jest kluczowe w przypadku np. filtrów FIR, gdzie dane są przetwarzane i transmitowane w równych odstępach czasowych.

1.1.3. HPS

Systemy heterogeniczne często łączą FPGA z wbudowanymi procesorami HPS, które pozwalają na elastyczne zarządzanie zadaniami w czasie rzeczywistym. Procesor wbudowany w HPS odpowiada za zarządzanie i konfigurację układów FPGA oraz synchronizację operacji między jednostkami obliczeniowymi. Dzięki temu możliwe jest wykorzystanie procesora do kontroli całego systemu i zarządzania interfejsami zewnętrznymi przy zachowaniu korzyści z możliwości obliczeniowych i równoległego przetwarzania FPGA, co przydaje się np. w przetwarzaniu sygnałów czy analizie dużych zbiorów danych.

1.2. Algorytmy DSP

1.3. Linux dla systemów wbudowanych

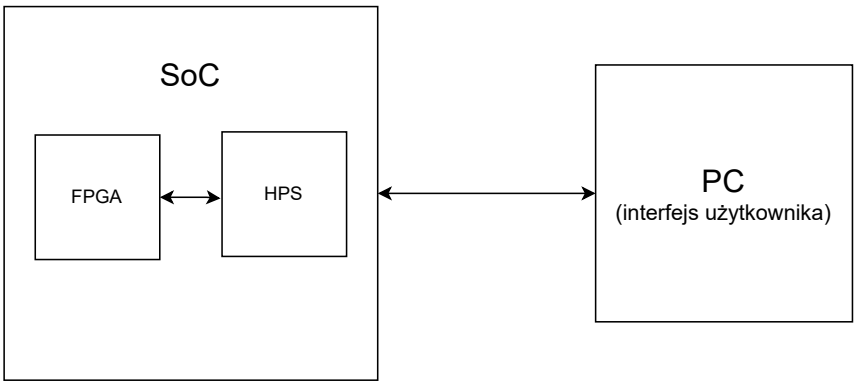
W dobie dynamicznego rozwoju systemów wbudowanych, inżynierowie potrzebują narzędzi które umożliwią im dostosowanie systemów operacyjnych do specyficznych potrzeb. Yocto Project to platforma open source, która zdobywa coraz większą popularność wśród inżynierów, oferując zestaw narzędzi do tworzenia indywidualnych dystrybucji systemu Linux. Takie rozwiązanie umożliwia tworzenie systemów dostosowanych do specyficznych wymagań aplikacji.

Yocto Project wyróżnia się także zdolnością do pracy z różnymi architekturami sprzętowymi, co czyni go uniwersalnym rozwiązaniem w wielu zastosowaniach. Modularność projektu pozwala na dodawanie i usuwanie pakietów, co zwiększa kontrolę nad funkcjonalnością systemu. W rezultacie inżynierowie mogą dostarczać wydajne i zoptymalizowane rozwiązania, które odpowiadają na potrzeby zmieniającego się rynku technologii wbudowanych.

W kontekście współczesnych wyzwań technologicznych, Yocto Project oferuje szereg narzędzi do zarządzania wersjami i aktualizacjami, co zapewnia długoterminowe wsparcie i stabilność, co jest istotne dla bezpieczeństwa aplikacji wbudowanych. Dzięki tym zaletom inżynierowie mogą skupić się na innowacjach i rozwijaniu swoich projektów, zamiast na rozwiązywaniu problemów związanych z wydajnością.

2. Architektura

Ogólna architektura zaimplementowanego rozwiązania została zaprezentowana na rysunku 2.1.



Rys. 2.1. Diagram przedstawiający uproszczoną architekturę systemu

2.1. Platforma

Platformą wybraną do realizacji pracy jest system heterogeniczny DE0-Nano-SoC, którego glównym elementem był układ Intel Cyclone V SoC.Zawiera on FPGA oraz HPS oparty na procesorze ARM, co umożliwia podział zadań między hardware i software.

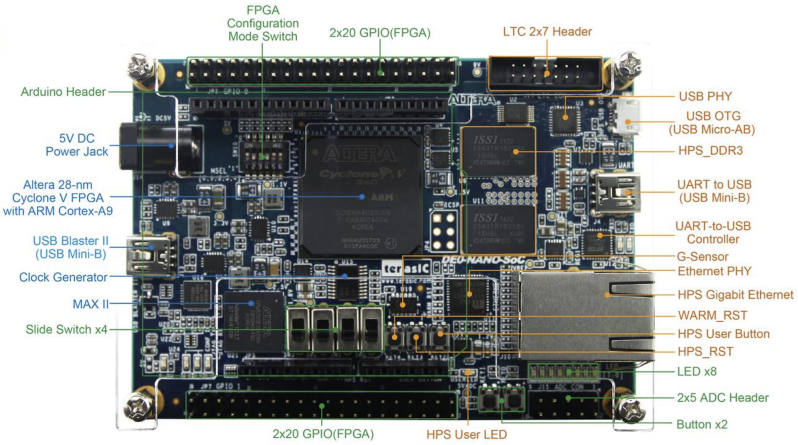
Układ FPGA w Cyclone V SoC oferuje programowalne zasoby, takie jak Look Up Table (LUT), bloki DSP oraz pamięci RAM, które umożliwiają implementację szerokiego zakresu funkcji, od prostych operacji po bardziej zaawansowane algorytmy. Zintegrowany z FPGA system HPS, oparty na dwurdzeniowym procesorze ARM Cortex-A9 z zegarem 925 MHz, komunikuje się z FPGA za pomocą mostków AXI i AXI Lightweight.

Decyzja o wyborze tej konkretnej platformy była podyktowana kilkoma istotnymi aspektami. Przede wszystkim heterogeniczna architektura systemu, dobrze sprawdza się w zastosowaniach DSP, co opisano dokładniej w sekcji 1.1.

Ponadto układ posiada gniazdo na kartę micro SD, na którą można wgrać obraz całego systemu. Karta SD pełni funkcję głównego nośnika, z którego ładowany jest system przy starcie. Dzięki wykorzystaniu wbudowanego slotu możemy uruchomić procesor wykorzytując niemal każdy system operacyjny

wspierający architekturę ARM, a korzystając z narzędzi takich jak Yocto jesteśmy w stanie skonfigurować system zgodnie z naszymi wymaganiami.

Złącze Ethernet w układzie zapewnia dostep do połaczenia sieciowego, co umożliwia zdalne zarządzanie oraz wymianę danych między systemem a zewnętrznymi urządzeniami. W kontekście opisywanej architektury, interfejs został wykorzystany, aby umożliwić połączenie sieciowe niezbędne do komunikacji z zaprojektowaną aplikacją internetową.



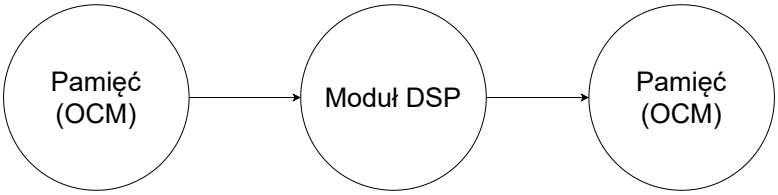
Rys. 2.2. Układ DE0-Nano-SoC (Cyclone V)

Ze względu na przystępną cenę i elastyczność konfiguracji, układ ten znajduje szerokie zastosowanie w projektach wymagających kompromisu między wydajnością a kosztami, szczególnie w środowiskach akademickich. Niniejsza praca zawiera przykłady tego, jak efektywnie wykorzystać ten układ w implementacji algorytmów przetwarzania sygnałów.

2.2. Architektura Sprzętowa

Architektura sprzętowa systemu została zaimplementowana z wykorzystaniem układu FPGA, który pełni rolę centralnego komponentu przetwarzającego dane. Rysunek 2.3 przedstawia uproszczoną architekturę systemu, ukazującą główne jego elementy. System składa się z dwóch pamięci On-Chip Memory (OCM), z których jedna jest odpowiedzialna za przechowywanie danych wejściowych przekazywanych z HPS, natomiast druga - danych wyjściowych po przetworzeniu.

Dane wejściowe są przesyłane do modułu DSP za pomocą kontrolera DMA i interfejsu ST-MM. Cała transmisja przez moduł DSP odbywa się w sposób strumieniowy, co oznacza, że zarówno dane wejściowe, jak i wyjściowe są przesyłane w czasie rzeczywistym. W module DSP realizowane są operacje przetwarzania sygnałów, w tym filtracja przy użyciu filtru FIR oraz szyfrowanie danych. Przetworzone dane są przesyłane do drugiej pamięci OCM poprzez interfejs MM-ST, skąd mogą zostać odczytane przez hosta.



Rys. 2.3. Diagram przedstawiający architekturę sprzętową

2.2.1. Interfejsy

W systemie wykorzystano dwa interfejsy do transmisji danych: Avalon Memory-Mapped (AVMM) oraz Avalon Streaming (AVST).

Avalon MM jest interfejsem pamięciowym, który sprawdza się w sytuacjach, gdy zachodzi potrzeba bezpośredniego odczytu lub zapisu danych do pamięci. Umożliwia transfer w postaci tzw. burstów, co oznacza, że pozwala na przesyłanie wielu paczek danych w kolejnych taktach zegara.

Avalon Streaming pozwala na przesył danch w sposób ciągły bez konieczności adresacji każdej paczki. Interfejs sprawdza się aplikacjach bazujących na przetwarzaniu w czasie rzeczywistym, wymagających szybkiego transferu dużych ilości danych.

Wybór pomiędzy tymi interfejsami zależy od specyficznych potrzeb aplikacji – Avalon MM jest odpowiedni do operacji bazujących na transakcjach do pamięci, podczas gdy Avalon Streaming lepiej sprawdza się w przypadku transmisji danych pomiędzy modułami, które mają je przetworzyć.

2.2.2. Zaprojektowane moduły DSP

System został wyposażony w dwa moduły DSP – filtr FIR o zmiennych współczynnikach oraz moduł szyfrowania danych oparty na algorytmie Tiny Encryption Algorithm (TEA).

Moduł filtru FIR umożliwia dynamiczną zmianę współczynników z poziomu oprogramowania, co pozwala na realizację różnych typów filtracji sygnałów w czasie rzeczywistym. Dzięki tej funckjonalności, filtr może pełnić różne typy filtracji, takie jak:

- Low Pass (dolnoprzepustowy): Wygładzanie sygnału przez eliminację wysokich częstotliwości.
- Bandpass (pasmowoprzepustowy): Przepuszczanie sygnałów z określonego zakresu częstotliwości.
- Moving Average (średnia krocząca): Wygładzanie danych poprzez uśrednianie kolejnych wartości sygnału.

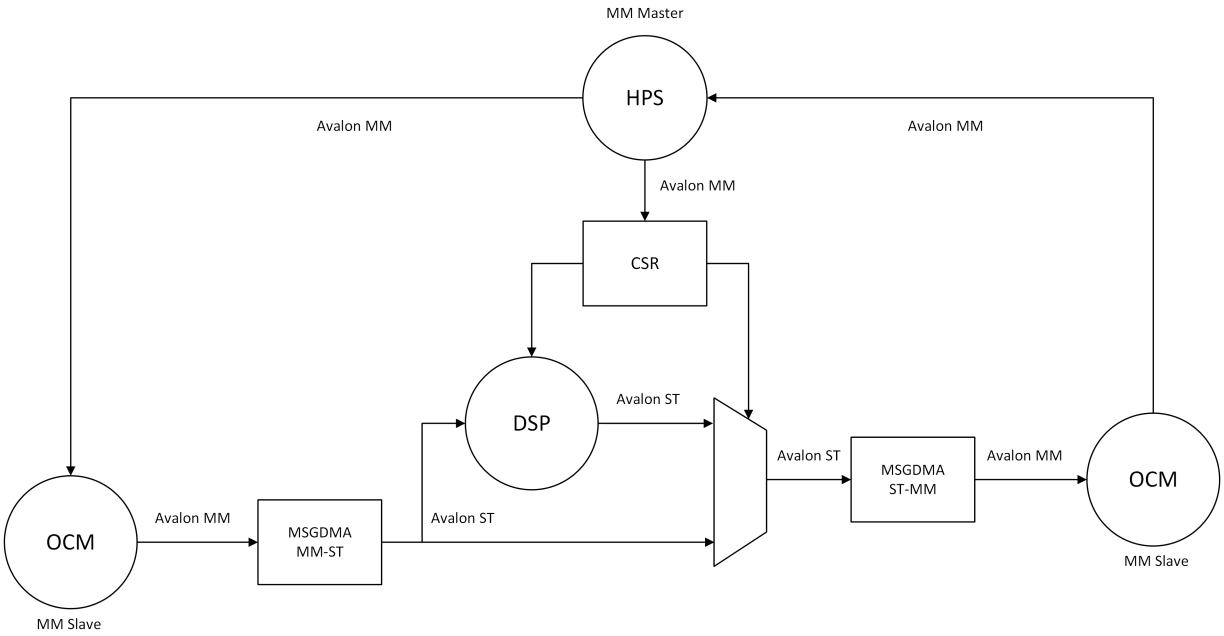
Modularność współczynników filtru umożliwia dostosowanie parametrów filtracji w zależności od aktualnych wymagań aplikacji, co znacząco zwiększa przydatność modułu w różnych zastosowaniach.

Moduł enkrypcji oparty na algorytmie TEA zapewnia bezpieczne szyfrowanie danych w systemie. TEA jest lekkim algorytmem szyfrowania blokowego, którego najważniejszymi cechami są prostota

implementacji oraz wysoka prędkość przetwarzania. Algoytm operuje na danych wejściowych o szerokości 32 bitów (w zaimplementowanym systemie szerokość wejść została ograniczona do 32 bitów) i 128-bitowym kluczu. Ze względu na swoją prostotę i niskie zapotrzebowanie na zasoby, TEA jest idealnym wyborem do zastosowań, w których wymagana jest efektywność przy zachowaniu podstawowego poziomu bezpieczeństwa.

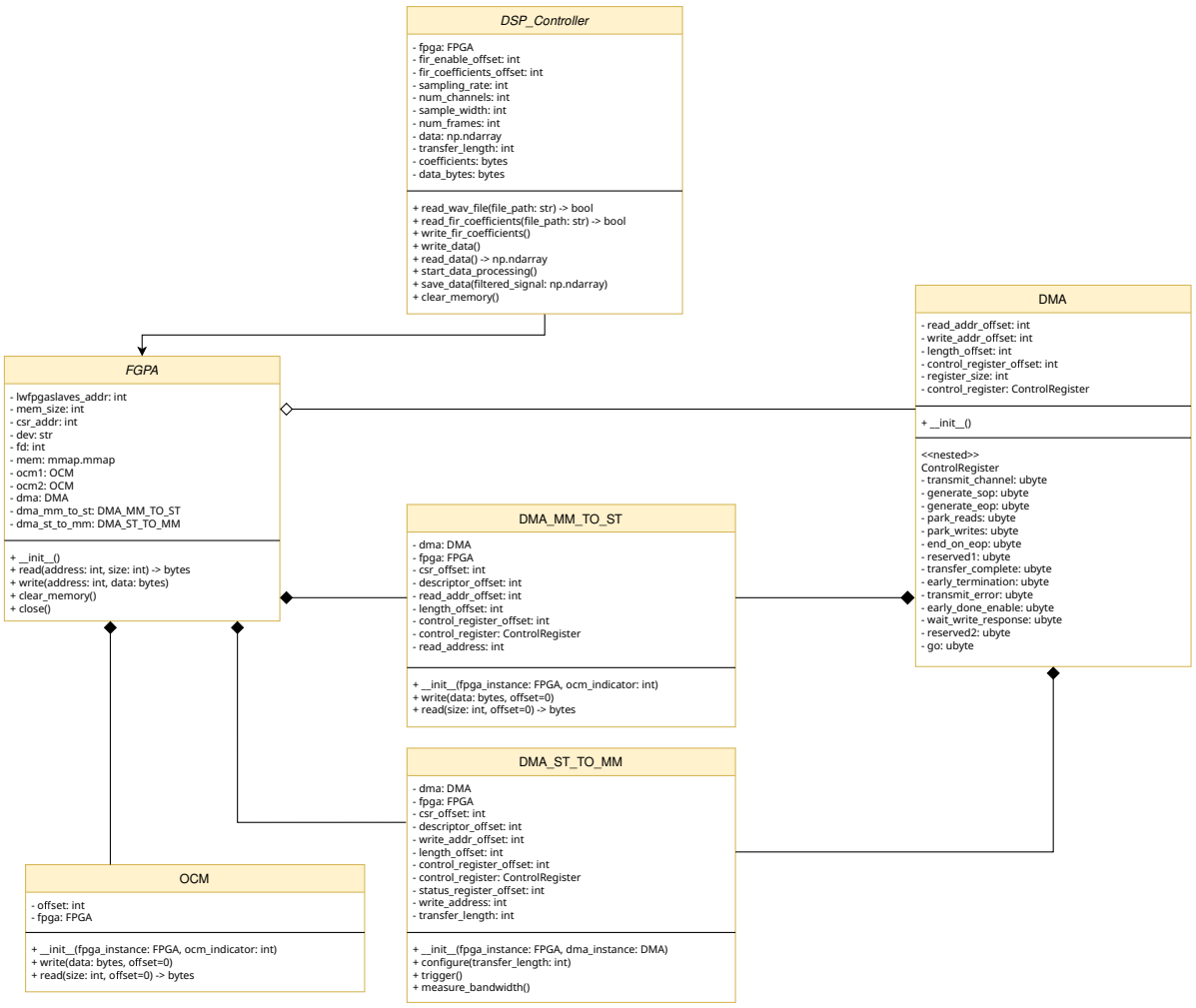
2.3. Architektura Programowa

3. Hardware



Rys. 3.1. Diagram przedstawiający architekturę sprzętową systemu

4. Software



Rys. 4.1. Diagram klas