Sprawozdanie z projektu SPEKTROP
Blok elektroniki spektrometru obrazującego

Creotech Instruments S.A.
Centrum Badań Kosmicznych PAN

23 maja 2014

Spis treści

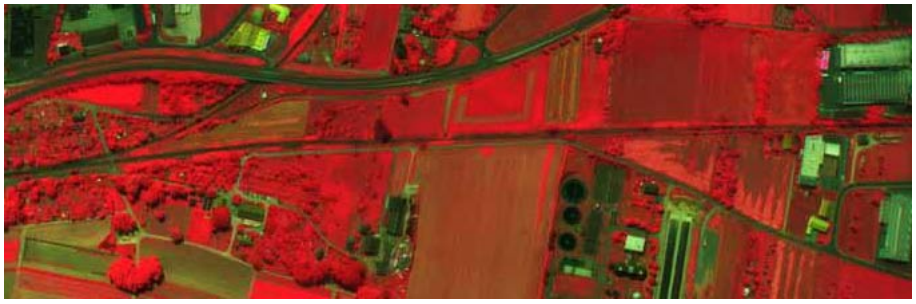
Spis rysunków

Rozdział 1

Wstęp

Poniższy dokument jest sprawozdaniem z projektu bloku elektroniki spektrometru obrazującego SPEKTROP zaprojektowanego przez Creotech Instruments S.A. na zlecenie Centrum Badań Kosmicznych Polskiej Akademii Nauk. Zawarto w nim specyfikację systemu, sposób działania oraz opis zastosowanych rozwiązań sprzętowych i programowych.

Projekt SPEKTROP jest systemem kamery spektralnej, umieszczonej na samolocie bądź UAV[1] pozwalającym na rejestrację obrazu lądu jednocześnie w wielu długościach fali. Takie rozwiązanie pozwala na analizę powierzchni różnych rodzajów obszarów jak lasy, jeziora, pola uprawne. Czy analizę możliwości wystąpienia złóż.



Rysunek 1.1: Przykładowy obraz z kamery spektralnej[4]

Sercem zaprojektowanego urządzenia jest układ SoC Zynq Z7045, który łączy w sobie układ FPGA Kintex 7 oraz dwurdzeniowy procesor ARM Cortex-A9. System pozwala na rejestrację danych z czujnika CMV4000 z maksymalną prędkością 100 FPS przy minimalnym czasie ekspozycji wynoszącym $\sim$ 10 ms. Ponadto system umożliwia rejestrację parametrów lotu niezależnie od akwizycji obrazu oraz z możliwością przyporządkowania danych do poszczególnych klatek. Dzięki czemu możliwa jest późniejsza analiza zebranych danych. Zebrane dane są zapisywane na dysku/ach HDD poprzez interfejs SATA zaimplementowanym w logice FPGA przy pomocy transceiverów GTX.

Sterowanie i komunikacja z systemem odbywa się poprzez interfejs Ethernet przy pomocy dedykowanej aplikacji działającej pod systemem Linux.

[1] UAV - Unmanned Aerial Vehicle

Rozdział 2

Specyfikacja

2.1 Czujnik CMOS

- Detektor: CMV4000 [2] albo CIS 1910F [3].
- Tryby pracy czujnika:
  - Podstawowy: czas ekspozycji (EXP[1]) 1-10 ms, 100 FPS[2]
  - Manualny: dowolnie długi czas ekspozycji z mniejszą ilością FPS (np. 5-10 FPS przy EXP $\sim$ 1-2 ms
- Rozdzielczość bitowa - 16 bit/piksel, ENOB[3]:8-9 bitów
- Zakres temperatury pracy: $-10^oC \div +50^oC$

2.2 Platforma sprzętowa

- Xilinx Zynq Evaluation Board ZC706 [1]

2.3 System rejestracji parametrów lotu

- rejestracja parametrów lotu na podstawie systemów GPS oraz IMU[4] z zapewnieniem synchronizacji czasowej względem akwizycji klatek i niezależnie
  - minimalny okres zapisu parametrów - 0.1 s
- zapis czasu pomiaru (timestamp) i czasu akwizycji parametrów lotu
- nośnikiem danych jest dysk twardy (SSD/HDD) z interfejsem SATA

2.4 Tryby pracy

- Ręczny - ustawiany przez operatora przy pomocy komputera PC (operator leci w samolocie)
- Autonomiczny - posiadający następujące funkcje:
  - wyliczanie ekspozycji i częstotliwości akwizycji klatek na podstawie parametrów lotu dostarczanych przez systemy sterujące samolotu/UAV
  - sprawdzanie poprawności ekspozycji
  - możliwość wyzwalania pomiarów na podstawie osiągnięcia pozycji wskazanej przez GPS

[1] EXP - Exposure
[2] FPS -Frames Per Second
[3] ENOB - Effective Number of Bits
[4] IMU - Internal Management Unit

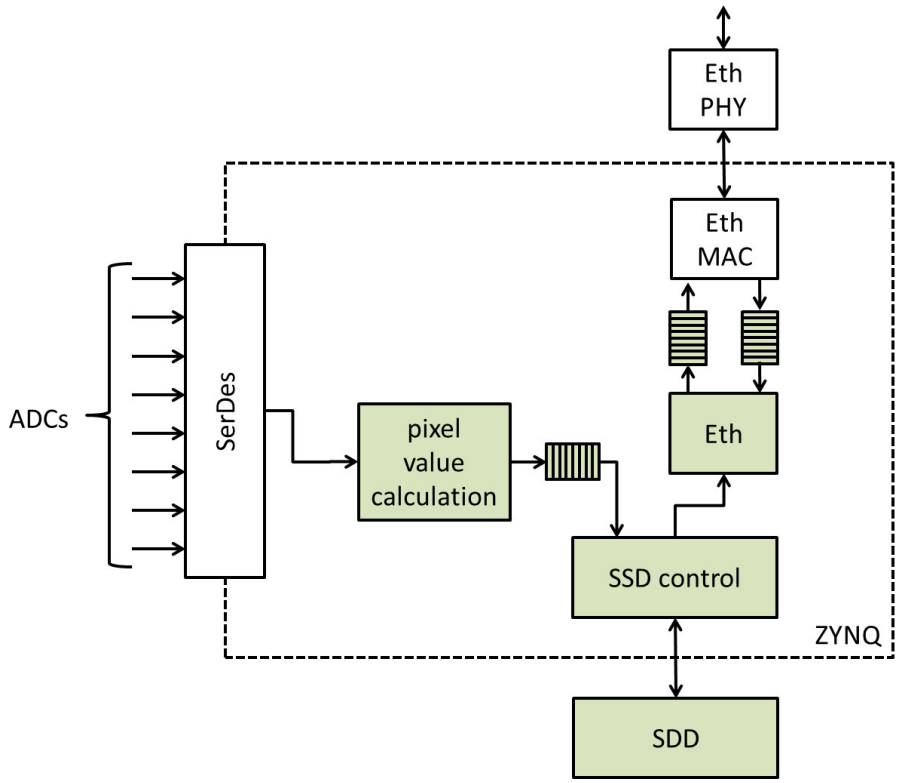
2.5 Wymagania elektryczne

Napięcie zasilania 24 V albo 12 V.

2.6 Wymagania mechaniczne

Wymiary PCB: 85 mm x 160 mm.

2.7 Proponowany schemat funkcjonalny systemu



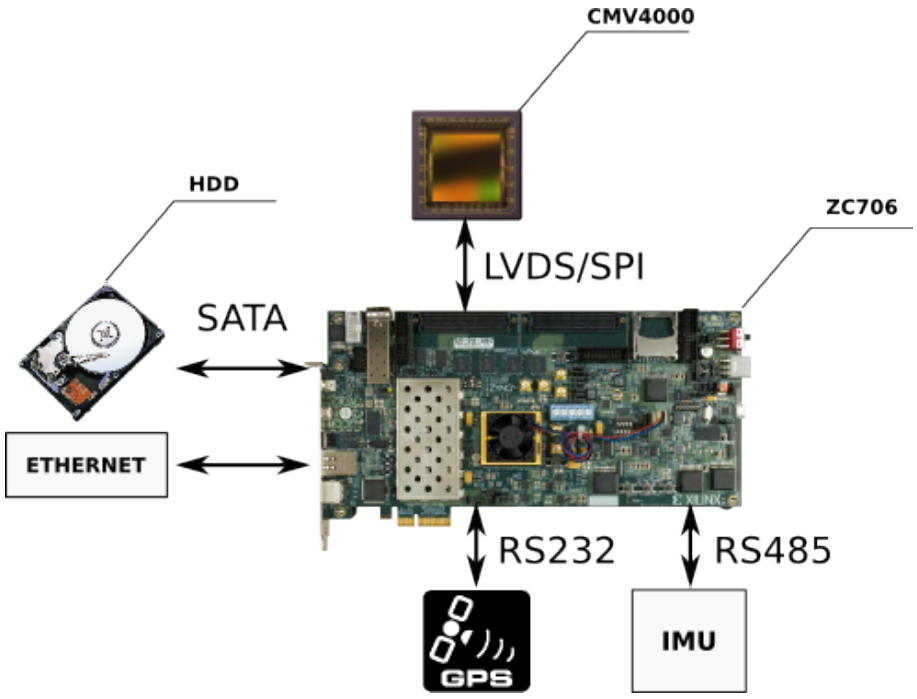
Rysunek 2.1: Schemat blokowy systemu

Rozdział 3

Koncepcja realizacji

Poniższy rozdział opisuje koncepcję realizacji bloku elektroniki projektu SPEKTROP. Rysunek [4.1] przedstawia główne elementy systemu:

- płytę ewaluacyjną ZC706
- czujnik CMOSIS MV4000
- interfejs SATA
- interfejs Ethernet
- połączenie z systemem GPS i IMU



Rysunek 3.1: Główne elementy systemu

Całość funkcjonalności systemu jest realizowana za pomocą układu SoC Zynq. Akwizycja danych i zapis na dysku poprzez interfejs SATA zostanie zrealizowany w logice programowalnej, natomiast kontrola i zarządzanie w dwurdzeniowym procesorze ARM Cortex A9.

Logika programowalna pozwala na przesył i prostą analizę dużej ilości danych. Czujnik CMV4000 wysyła dane do układu Zynq szeregowo 8 liniami LVDS, dane w FPGA są deserializowane i przesyłane poprzez wewnętrzne magistrale do IP SATA.

Zarządzanie jest realizowane głównie za pomocą jednego rdzenia ARM, na którym uruchomiony jest system czasu rzeczywistego FreeRTOS wraz z obsługą stosu TCP/IP. Drugi rdzeń odpowiada za obsługę przerwania IP AXI VIDEO DMA.

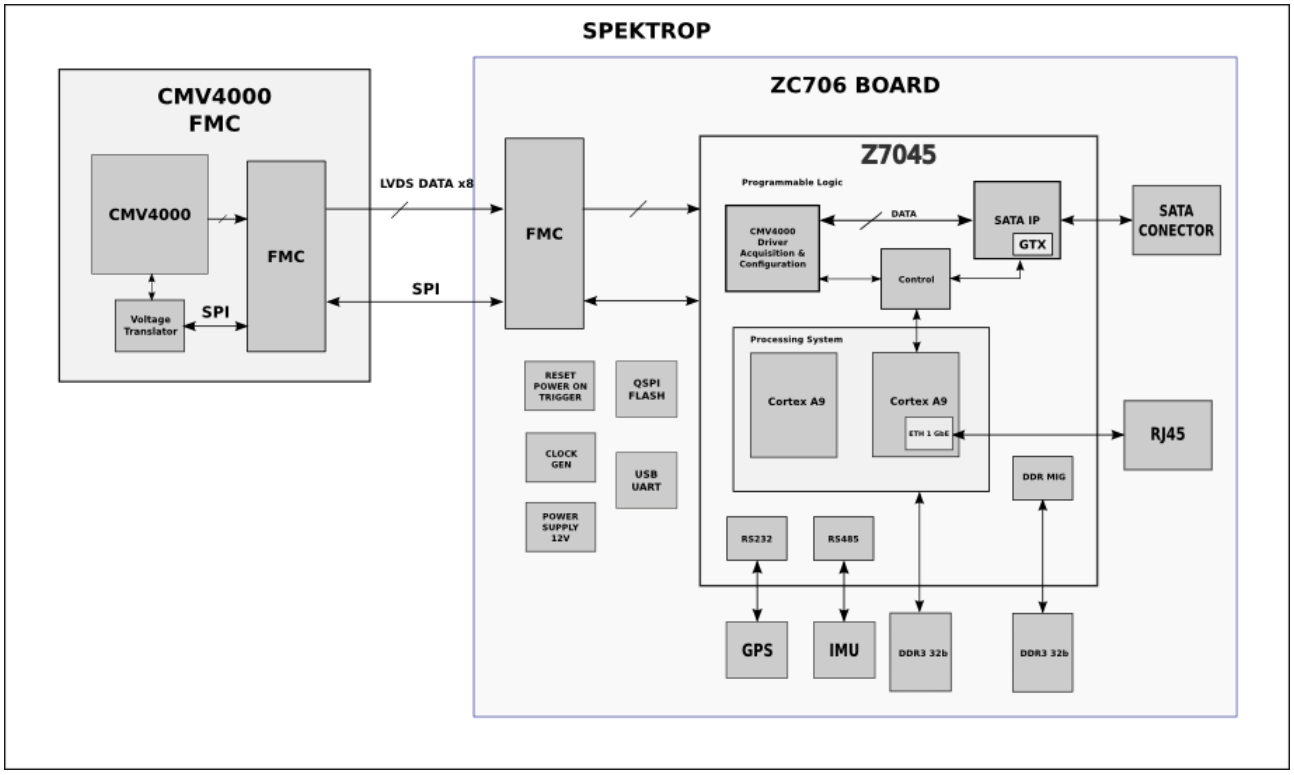
Po akwizycji każdej klatki, zapisywane są parametry lotu związane z daną klatka oraz dotyczące samej akwizycji. Niezależnie od tego na karcie pamięci zapisywane są dane z parametrów lotu oraz dane diagnostyczne systemu.

Rozdział 4

Realizacja

Niniejszy rozdział zawiera opis techniczny realizacji bloku elektroniki do projektu SPEKTROP. W pierwszej części opisano wykorzystany układ SoC Zynq oraz czujnik CMV4000. Następnie przedstawiona została

Rysunek [4.1] zawiera schemat blokowy systemu. System składa się z modułu ewaluacyjnego układu Zynq ZC706, oraz z modułu FMC z czujnikiem CMOSIS CMV4000.



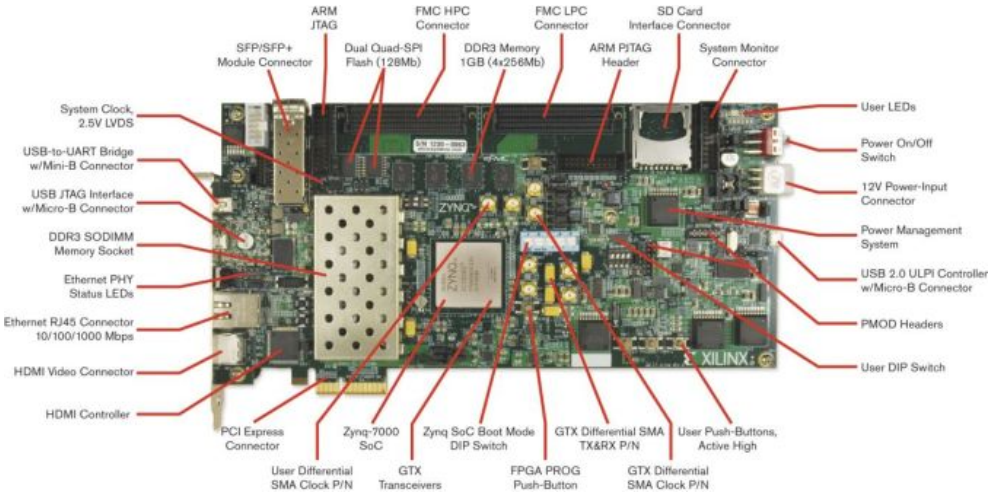
Rysunek 4.1: Główne elementy systemu

4.1 Platforma sprzętowa

System oparty jest o moduł ewaluacyjny ZC706 posiadający wszystkie podstawowe komponenty sprzętowe potrzebne do realizacji zaawansowanych systemów przetwarzania.

Moduł posiada następujące komponenty:

- Układ SoC
  - Zynq-7000 XC7Z045 FFG900
- Konfiguracja



Rysunek 4.2: Elementy płyty ewaluacyjnej ZC706

  - 2X16MB Quad SPI Flash
  - SDIO
  - PC4 i JTAG

- Pamięć
  - DDR3 Component Memory 1GB (PS)
  - DDR3 SODIM Memory 1GB (PL)
  - 2X16MB Quad SPI Flash
  - IIC - 1 KB EEPROM

- Interfejsy komunikacyjne
  - PCIe Gen2x4
  - SFP+ and SMA Pairs
  - GigE RGMII Ethernet (PS)
  - 1 CAN with Wake on CAN (PS)
  - USB OTG 1 (PS) - Host USB
  - IIC Bus Headers/HUB (PS)
  - USB UART (PS)

- Komponenty wideo
  - HDMI 8 color RGB 4.4.4 1080P-60 OUT
  - HDMI IN 8 color RGB 4.4.4

- Złącza we/wy
  - FMC LPC
  - FMC HPC
  - Pmod dualny i pojedynczy
  - Dostęp do I2C
  - USB OTG 1 (PS) - Host USB
  - IIC Bus Headers/HUB (PS)
  - USB UART (PS)

- Sygnały zegarowe
  - 33MHz Zegar systemowy
  - 200MHz PL Oscylator
  - złącza SMA dla zewnętrznych sygnałów zegarowych
  - referencyjne do GTX
  - OBSAI/CPRI – SFP+
  - EXT Config CLK
- Sterowanie
  - 2 User Push Buttons/Dip Switch, 2 User LEDs
  - 3 User Push Buttons, 2 User Switches, 8 User LEDs
  - IIC access to 8 I/O
  - IIC access to a WTClock
- Zasilanie
  - 12 V
  - możliwość pomiaru prądu linii zasilających

4.2 Układ SOC Zynq

Układy z serii Zynq 7000 stanowią najnowszą generację układów typu SOC FPGA – stanowią połączenie na jednym kawałku krzemu programowalnych układów logicznych FPGA serii 7 z dwurdzeniową procesorem ARM Cortex A9 oraz szeroką paletą peryferiów takich jak Ethernet, SPI, I2C, CAN, USB, itp. Połączenie takie upraszcza tworzenie oraz obniża koszty złożonych systemów wbudowanych. Układ Zynq 7020 używany w projekcie posiada 32 bitowy interfejs do pamięci DDR3 współdzielony pomiędzy procesorem, a logiką programowalną. Zapewnia to bardzo dużą przepustowość danych pomiędzy procesorem a FPGA. Od strony logiki programowalnej dostęp do pamięci realizowany jest przez 4 złącza AXI HP (wysokiej przepustowości). Maksymalna, teoretyczna przepustowość pojedynczego kanału HP wynosi 1200MB/s w każdą stronę. Maksymalna, teoretyczna, całkowita przepustowość pamięci DDR3 w układzie wynosi 4264 MB/s. Dodatkowo układ posiada po dwa złącza AXI GP (ogólnego przeznaczenia) master i slave między procesorem a układem FPGA o przepustowości 600MB/s w każdą stronę. Najczęściej wykorzystywane są one do programowania rejestrów peryferiów zrealizowanych w logice programowalnej. Układ Zynq w wersji 7020 ma 85 tysięcy programowalnych komórek logicznych, 106400 przerzutników, 220 układów DSP oraz 560KB BRAM. Rdzenie ARM Cortex pracują z zegarem 667MHz.

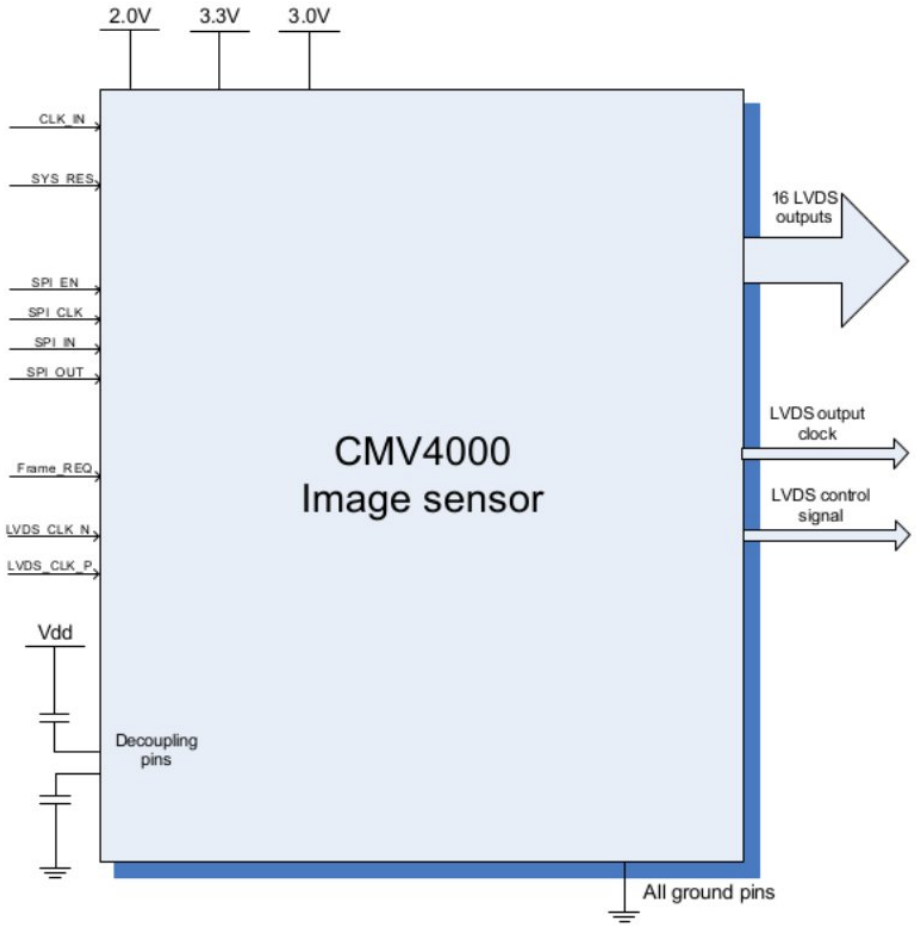
4.2.1 Architektura APU

Każdy z rdzeni posiada po 32KB pamięci podręcznej instrukcji oraz danych, układ MMU, FPU oraz NEON SIMD. Rdzenie dzielą między sobą pamięć podręczną drugiego poziomu o rozmiarze 512 KB, kontroler przerwań UART, OCM oraz pamięć DDR.

4.3 Sensor CMV4000

Sensor użyty w projekcie to przetwornik video o rozmiarze 1", wykonany w technologii CMOS. Charakteryzuje się rozdzielczością 2048x2048 punktów i maksymalną szybkością przetwarzania ramek obrazu o wartości 180 klatek/s przy zegarze o częstotliwości 480 MHz (maksymalna częstotliwość zegara dla tego sensora). Sensor posiada 2 wejścia zegarowe – zegar systemowy oraz zegar LVDS. Programowanie rejestrów sensora odbywa się przy pomocy interfejsu SPI. Dane z przetwornika wystawiane są na 16 (lub mniej w zależności od ustawień) wyjść LVDS. Dodatkowo sensor posiada jeszcze 1 wyjście sterujące LVDS, na które wystawiane są sygnały sterujące strumieniem danych oraz 1 wyjście zegara LVDS

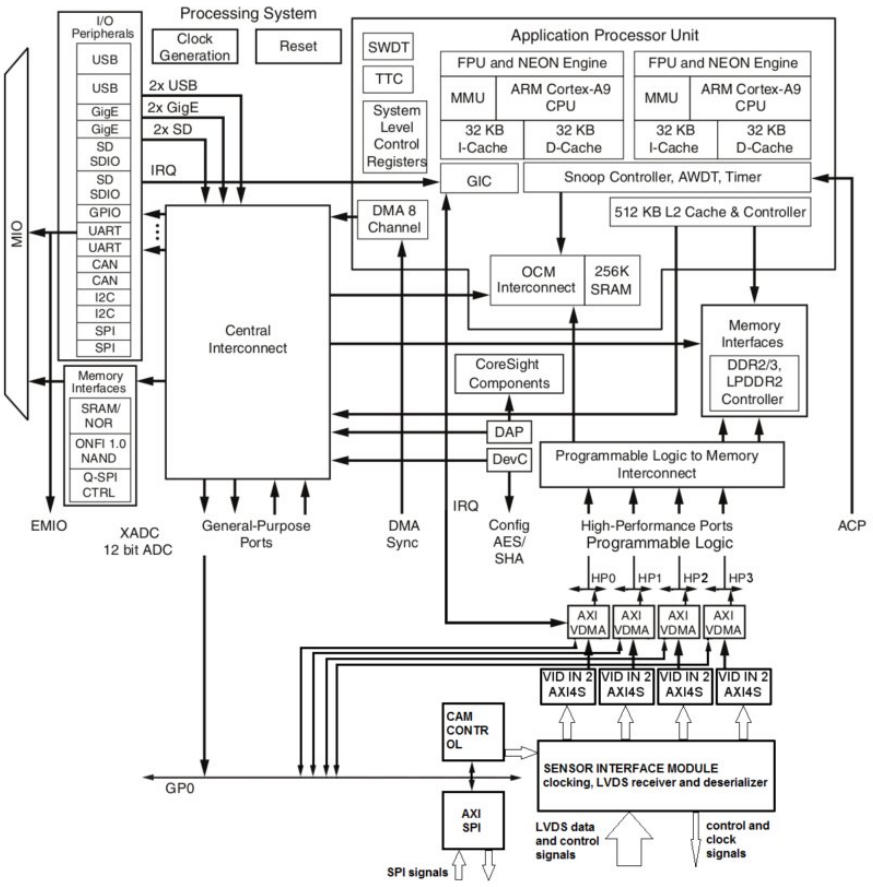
danych. Dane wystawiane są z częstotliwością równą połowie częstotliwości zegara podanego na wejście zegarowe LVDS na obu zboczach zegara (DDR). Dane na poszczególnych liniach LVDS są wysyłane przez sensor w postaci szeregowej – ze współczynnikiem 10:1 lub 12:1 (w zależności od ustawień przetwornika AC). Wysyłanych jest równolegle 16 serializowanych fragmentów ramki obrazu – po jednym na każde wyjście LVDS. Aby odtworzyć pierwotną postać obrazu, należy najpierw zdeserializować, a następnie przegrupować dane otrzymane od sensora.



Rysunek 4.3: Schemat połączeń sensora CMV4000

4.4 Część sprzętowa (logika programowalna)

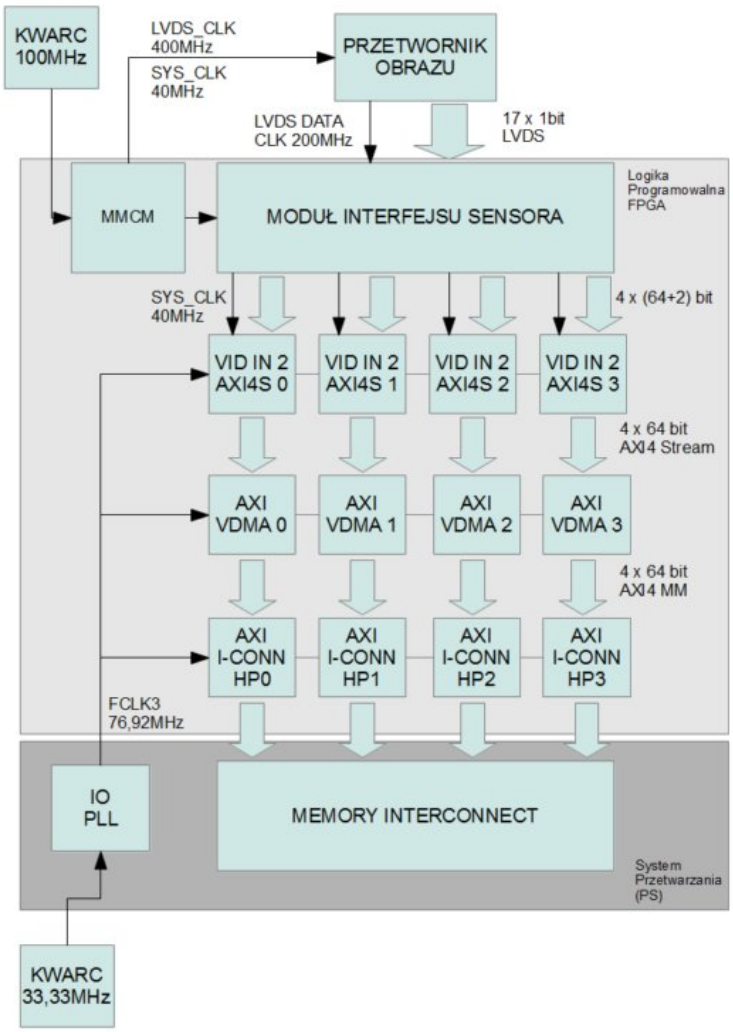
Część sprzętowa projektu składa się z pliku konfiguracyjnego układu SOC FPGA wygenerowanego na podstawie list połączeń utworzonych przez kreatory IP – core oraz na podstawie kodu VHDL. Zadaniem modułu sprzętowego jest wysyłanie sygnałów sterujących do sensora oraz odbieranie danych z sensora, odpowiednie ich przetwarzanie i zapisywanie w pamięci RAM systemu. Dalszym przetwarzaniem zajmuje się część programowa projektu, działająca na dwóch rdzeniach procesora ARM Cortex A9.



Rysunek 4.4: Schemat konfiguracji sprzętowej układu Zynq

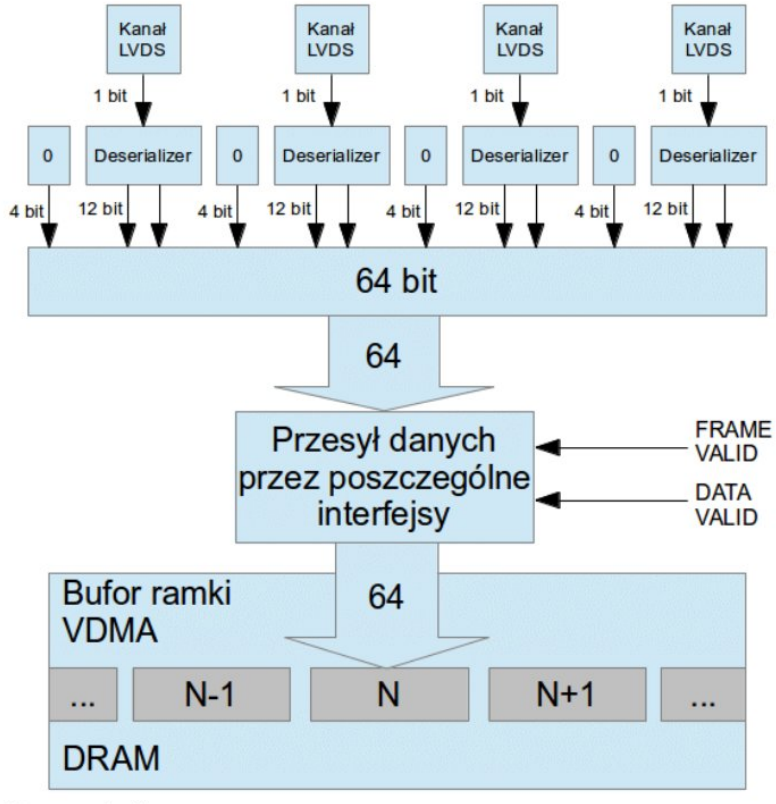
W części sprzętowej wykorzystano zarówno gotowe rdzenie IP dostarczane nieodpłatnie przez firmę Xilinx jak i własny kod VHDL. Jak widać na rysunku 2, układ został skonfigurowany w taki sposób, aby wykorzystać wszystkie dostępne łącza wysokiej przepustowości (HP0 - HP3) pomiędzy logiką programowalną, a kontrolerem pamięci DDR RAM. Pozwala to na optymalne dopasowanie się do szerokości szyny kontrolera pamięci, która wynosi 64 bity. Zdeserializowane dane z sensora po uzupełnieniu zerami mają szerokość równą 4x64bity. Wykorzystano także jeden port ogólnego przeznaczenia (GP0), do komunikacji z rdzeniami IP. Maksymalna przepustowość danych pomiędzy sensorem, a interfejsem FPGA wynosi: 180kl./s*2048*2048*10bit = 7200Mbit/s. Przepustowość ta jest sztucznie zwiększana na ścieżce pomiędzy modułem interfejsu LVDS, a rdzeniami AXI VIDEO DMA, ponieważ linie danych na magistralach AXI muszą mieść szerokość równą wielokrotności 8 bitów. Rozmiar próbki jest więc uzupełniany zerami do szerokości 16 bitów w module interfejsu LVDS i w takiej postaci przesyłany do rdzeni VID IN 2 AXI4S. Zwiększa to wykorzystywaną część przepustowości do maksymalnie 11520Mbit/s = 1440MB/s. W projekcie ta wartość jest nieco mniejsza, ponieważ sensor nie jest taktowany z maksymalną częstotliwością.

4.4.1 Przepływ danych w FPGA



Rysunek 4.5: Przepływ danych z czujnika do logiki

Rysunek [4.5] przedstawia schemat modułów sprzętowych przetwarzających i przesyłających dane obrazowe z sensora. Na rysunku pominięto interfejs SPI oraz interfejsy AXI4 Lite. W projekcie wykorzystano 2 układy PLL do generacji zegarów dla systemu – jeden zasilany z kwarcu procesora i generujący zegar 76,92 MHz dla rdzeni IP oraz drugi zasilany z kwarcu przeznaczonego dla FPGA i generujący zegary o różnych częstotliwościach (wyszczególnionych poniżej) dla Modułu interfejsu sensora.



Rysunek 4.6: Deserializacja i buforowanie danych

Na rysunku 4 przedstawiono uszczegółowienie jednej z 4 ścieżek przepływu danych obrazu w systemie. Na każdą ścieżkę przypadają 4 z 16 kanałów danych LVDS. 1 kanał sterujący LVDS jest współdzielony pomiędzy ścieżkami (dane sterujące są takie same dla wszystkich kanałów danych). Jak wynika z przedstawionego rysunku oraz sposobu przesyłania danych przez sensor [1], dane w pamięci nie są układane w kolejności odpowiadającej ich położeniu w obrazie i muszą zostać uporządkowane przed generowaniem podglądu lub zapisu do formatu obrazowego. Porządkowanie danych odbywa się programowo.

4.4.2 Wykorzystywane interfejsy komunikacyjne

- AXI4 – Stream
- AXI4 – Lite
- AXI4 – Memory Mapped

4.4.3 Domeny zegarowe

W projekcie wykorzystywane są następujące domeny zegarowe:

- FCLK_CLK3 – 76,92308 MHz, zegar systemowy, generowany w PLL układu Zynq
- SYS_CLK – 40MHz, główny zegar modułu interfejsu sensora, główny zegar sensora oraz zegar pikseli po deserializacji, generowany w PLL układu Zynq
- SPI_CLK – FCLK_CLK3/32, zegar SPI generowany w rdzeniu AXI SPI z głównego zegara systemowego

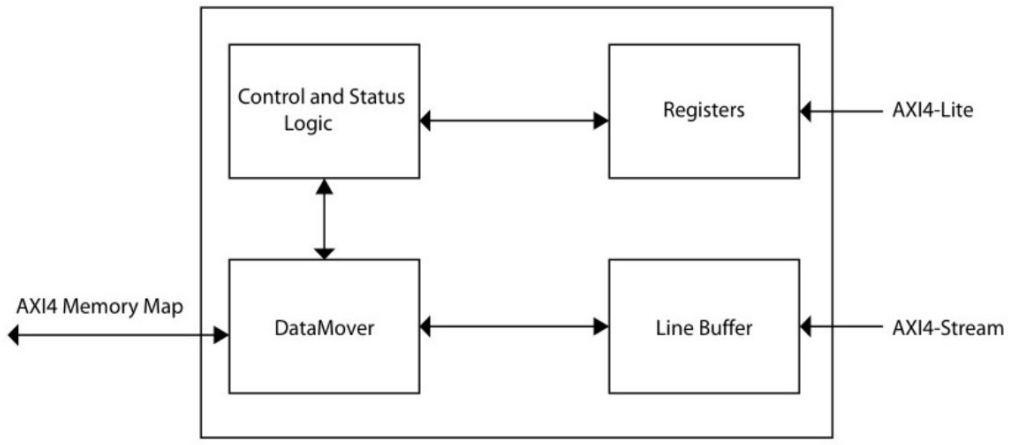
- ZEGAR DANYCH LVDS – 200MHz, zegar odbiorczy danych LVDS, generowany w PLL układu Zynq
- ZEGAR LVDS SENSORA – 400MHz, zegar nadajnika LVDS sensora, generowany w PLL układu Zynq

Sensor CMV4000 generuje dodatkowo własny sygnał zegarowy danych LVDS, o częstotliwości równej połowie częstotliwości wejściowego zegara LVDS, ale nie jest on wykorzystywany w projekcie.

4.4.4 Opis modułów IP

AXI Video Direct Memory Access 5.04.a

AXI VDMA jest modułem własności intelektualnej dostarczanym nieodpłatnie przez firmę Xilinx wraz z licencją na oprogramowanie projektowe ISE Design Suite do układów SOC oraz FPGA tej firmy. Moduł ten oferuje dwukierunkowe łącze wysokiej przepustowości pomiędzy peryferiami o interfejsie zgodnym z protokołem AXI4 – Stream Video, a pamięcią (w tym przypadku pamięcią DDR RAM). Zarządzanie modułem odbywa się poprzez zapisywanie oraz odczytywanie zestawu rejestrów poprzez interfejs AXI4 – Lite. Moduł obsługuje interfejs AXI4 Memory Map o szerokościach 32, 64, 128, 256, 512 oraz 1024 bitów oraz interfejs AXI4 – Stream o szerokościach do 1024 bitów, przy czym szerokość ta musi być wielokrotnością 8. Rzeczywista szerokość szyny do pamięci wynosi 64 bity, w związku z czym większe szerokości są dzielone na 64 bitowe transfery. AXI VDMA ma możliwość przechowywania w pamięci zewnętrznej do 32 ramek obrazu. Moduł zapewnia podstawową możliwość obsługi błędów – błędne wartości podstawowych parametrów pracy są wykrywane, a informacja o nich jest zapisywana na poszczególnych bitach odpowiednich rejestrów. AXI VDMA może generować przerwanie na zaprogramowanym przez użytkownika zdarzeniu takim jak wystąpienie błędu, odliczenie zadanej ilości klatek lub odliczenie zadanego opóźnienia względem początku klatki.



Rysunek 4.7: Przepływ danych w magistralach AXI

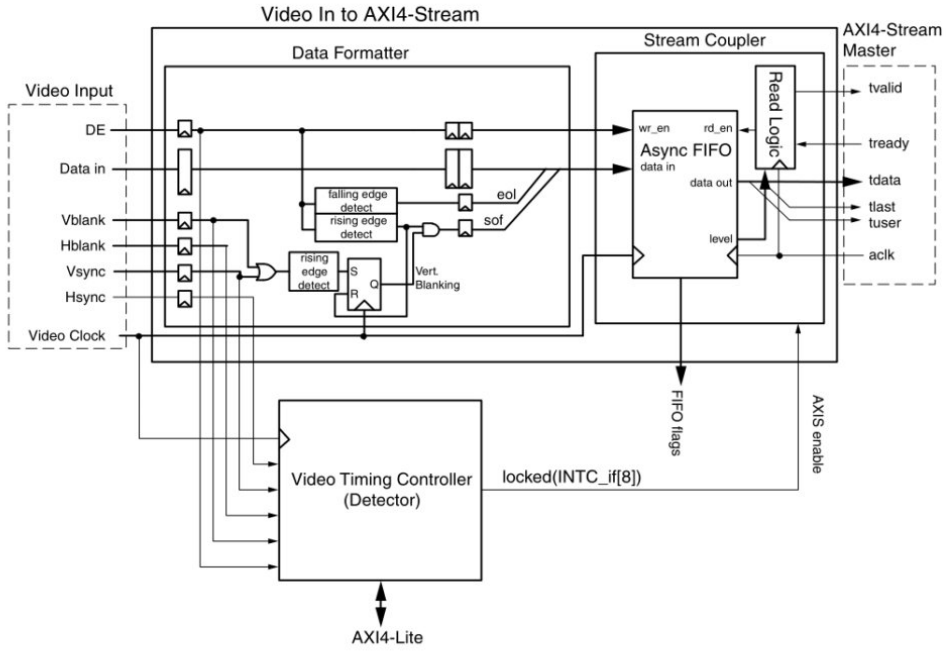
Video In to AXI4 – Stream 2.01.a

Podobnie jak AXI VDMA moduł ten jest udostępniany nieodpłatnie przez firmę Xilinx. Jego zadaniem jest dostarczanie interfejsu pomiędzy źródłami cyfrowego, równoległego sygnału wideo (np. z przetworników wideo, interfejsu HDMI, DVI itp.), a magistralą protokołu AXI4 – Stream Video. Moduł zapewnia

przejście pomiędzy dwoma domenami zegarowymi – zegarem pikseli sygnału wideo oraz zegarem magistrali AXI4 Stream. Obsługiwana jest większość formatów cyfrowego wideo, z długościami próbek na kanał do 16 bitów. Moduł posiada także wejściową kolejkę FIFO, która pozwala na zastosowanie zegara AXI, o częstotliwości mniejszej od zegara pikseli. Jest to możliwe, ponieważ protokół AXI4 Stream przesyła tylko aktywne piksele obrazu.

Minimalnym zestawem sygnałów sterujących wideo potrzebnych do poprawnego działania modułu jest para DE, Vsync. Sygnały te są niezbędne do ustalenia granic końca linii oraz początku nowej ramki. W projekcie do wejścia DE został podłączony sygnał Data Valid sensora, a do wejścia Vsync sygnał Frame Valid, uprzednio zdeserializowane z kanału sterującego LVDS. W projekcie nie jest wykorzystywany moduł Video Timing Controller.

Sensor CMV4000 wystawia sygnały Frame Valid oraz Data Valid na tym samym zboczu zegara, natomiast jak widać na rysunku 5, moduł Video In to AXI4 Stream wymaga, by sygnał Vblank lub Vsync miał ustaloną wartość, zanim pojawi się DE. W związku z tym zaimplementowano dodatkowy moduł, który opóźnia sygnał Data Valid o jeden takt zegara w stosunku do Frame Valid.



Rysunek 4.8: Schemat połączenia w magistrali AXI4-Stream

AXI SPI 1.02.a

Moduł Xilinx AXI Serial Peripheral Interface (SPI) zapewnia komunikację z urządzeniami posiadającymi szeregowy interfejs urządzeń peryferyjnych [4]. W projekcie wykorzystywany jest do programowania przetwornika obrazu CMV 4000. Komunikacja odbywa się synchronicznie przy pomocy 4 standardowych sygnałów SPI:

- Master Out Slave In (MOSI)
- Master In Slave Out (MISO)
- Serial Clock (SCK)
- Slave Select (SS)

Wewnętrzne rejestry modułu są programowane przez użytkownika poprzez interfejs AXI4 Lite. Ogólnie przyjętym standardem jest, że sygnał Slave Select jest aktywny poziomem niskim. W przypadku użytego sensora występuje sytuacja odwrotna – Slave Select jest aktywny poziomem wysokim, co wymagało stworzenia „przejściówki" w VHDL odwracającej wartość sygnału Slave Select.

AXI Interconnect 1.06.a

Moduł Xilinx AXI Interconnect łączy 1 lub więcej urządzeń nadrzędnych (master) AXI4 MM do 1 lub więcej urządzeń podrzędnych (slave) AXI4 MM. W projekcie moduły te zapewniają połączenie 1 do 1 pomiędzy czterema modułami AXI VDMA (master), a czterema łączami wysokiej przepustowości HP do kontrolera pamięci (slave) oraz pomiędzy interfejsem AXI4 Lite (master), a łączem ogólnego przeznaczenia GP (slave). W przypadku bieżącej wersji układu Zynq, moduł ten zapewnia także konwersję pomiędzy wersjami 3 i 4 protokołu AXI. Sprzętowo, od strony układu przetwarzania (PS) obsługiwana jest wersja 3 protokołu, natomiast moduły własności intelektualnej korzystają z wersji 4. Szczegółowy opis architektury moduł można znaleźć w [5].

Cam – Control 1.00.a

Cam Control jest bardzo prostym pod-modułem stworzonym w ramach projektu, służącym do sterowania Modułem Interfejsu Sensora. Składa się on z jednego 32 – bitowego rejestru, programowanego poprzez interfejs AXI4 Lite. Opis poszczególnych bitów rejestru sterującego:

- 0 – frame request
- 1 – reset modułu interfejsu sensora

Moduł Interfejsu Sensora

Napisana w VHDL jednostka odpowiada za odbieranie i deserializację danych obrazowych z przetwornika oraz generację sygnałów zegarowych wewnętrznych oraz dla przetwornika. Jednostka została napisana na podstawie kodu VHDL firmy CMOSIS, służącego do obsługi sensora CMV4000 poprzez interfejs Camera Link. Kod firmy CMOSIS stworzony i przetestowany został na układzie Xilinx Virtex 4. Uruchomienie kodu na układzie Zynq i przystosowanie do potrzeb bieżącego projektu wymagało pewnych zmian. Moduł interfejsu składa się z następujących plików źródłowych VHDL:

- *module_1_stub.vhd* Główny plik projektu, odpowiada za integrację interfejsu sensora z modułami IP oraz systemem przetwarzania Zynq. Zawiera opis połączeń modułów z sygnałami wyprowadzonymi na zewnątrz układu FPGA.
- *tsc_mv1_top.vhd* Nadrzędny plik modułu interfejsu sensora. Zawiera instancje wszystkich pozostałych podmodułów.
- *tsc_mv1_clocking.vhd* Plik modułu generacji zegara, zawiera instancję modułu zegarowego Xilinx Clocking Wizard v3.6.0. Odpowiada także za generację sygnałów resetu dla pozostałych podmodułów interfejsu sensora.
- *tsc_mv1_control.vhd* Plik zawiera implementację głównego automatu sterującego modułem interfejsu LVDS.
- *tsc_mv1_datapath.vhd* Plik zawiera instancję odbiornika LVDS.
- *tsc_mv1_rx.vhd* Główny plik odbiornika LVDS, zawiera instancje deserializerów LVDS dla poszczególnych kanałów sensora oraz automat sterujący trenowaniem interfejsu LVDS. Dla zaoszczędzenia zasobów logiki programowalnej, zaimplementowany został jeden kontroler trenujący, współdzielony przez wszystkie kanały LVDS. Trenowanie odbywa się sekwencyjnie, każdego kanału po kolei.
- *tsc_mv1_ser2par.vhd* Plik zawierający opis interfejsu odbiorczego LVDS. Uproszczony schemat przedstawiony jest na rysunku 6.

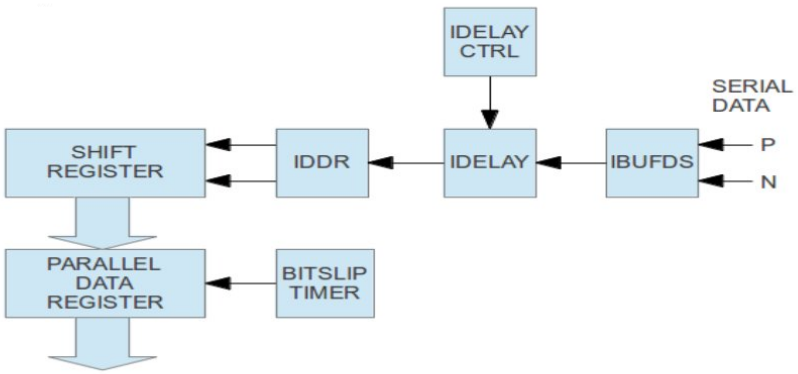
- *vid_sig_formatter.vhd* Plik zawierający opis modułu przesuwającego sygnały Data Valid i Frame Valid zgodnie z wymaganiami rdzenia Video in to AXI4 Stream.

SATA IP

Moduł SATA IP implementuje warstwy *Command,Transport,Link* interfejsu SATA [**?**] i udostępnia moduł łączący warstwę fizyczną z IP transcerivera GTX. Moduł warstwy fizycznej posiada maszynę stanów do obsługi sygnałów Out-Of-Band służących do inicjalizacji i synchronizacji połączenia interfejsu SATA. IP umożliwia połączenie z dyskami z interfejsem SATA 2 (Winchester) oraz z dyskami SSD FLASH. Moduł interfejsu składa się z następujących plików źródłowych:

- *command_layer.vhd* Implementacja warstwy sterującej interfejsu SATA.
- *crc.vhd* Moduł liczący sumę kontrolną CRC32.
- *ipcores.v* IP transceivera GTX, firmy Xilinx.
- *mux_21.v* Moduł multipleksera 2bit 2:1.
- *mux_41.v* Moduł multipleksera 32bit 4:1.
- *mux_161.v* Moduł multipleksera 32bit 16:1.
- *oob_control.v* Moduł obsługujący wymagania sygnałowe Out-Of-Band (OOB) dla inicjalizacji i synchronizacji interfejsu.
- *sata_core.v* Główny moduł SATA IP.
- *sata_link_layer.v* Moduł SATA IP implementujący warstwę połączeniową (link layer).
- *scrambler.v* Moduł implementujący skrambler danych dla protokołu SATA.

4.4.5 Interfejs odbiorczny LVDS



Rysunek 4.9: Interfejs odbiorczy LVDS

Interfejs odbiorczy LVDS dla każdego kanału składa się z szeregu bloków funkcjonalnych. Pierwszym elementem jest IBUFDS – bufor odpowiedzialny za odbiór elektrycznego sygnału różnicowego i jego zamianę na na sygnał pojedynczy. Następnym elementem jest programowalny bufor opóźniający IDELAY, którego zadaniem jest wyrównanie opóźnień poszczególnych linii danych i zapewnienie optymalnego punktu próbkowania danej na zboczu zegarowym. W układach FPGA serii 7 (a więc także Zynq)

bufor ten ma 32 stopniową skalę opóźnienia, a każdy krok opóźnia sygnał wejściowy o 78ps (Ref_clk = 200MHz) [9], co razem daje prawie 2,5ns opóźnienia (1 okres zegara 400MHz).

Blokiem współpracującym z IDELAY jest IDELAY CONTROL. Element ten na bieżąco kontroluje i kalibruje blok opóźniający uodporniając go na rozrzuty produkcyjne, zmiany temperatury oraz napięć zasilających (w pewnym zakresie) [8]. Po odpowiednim opóźnieniu sygnał trafia do bloku IDDR, który próbkuje sygnał na obu zboczach zegara danych i zamienia go na dwa sygnały wystawiane na pojedynczym zboczu. Następnie dane są kierowane do rejestru przesuwnego i rejestru wyjściowego, gdzie są zatrzaskiwane na sygnale wystawianym przez licznik modulo 10 (lub 12).

4.4.6 Trenowanie interfejsu odbiorczego LVDS

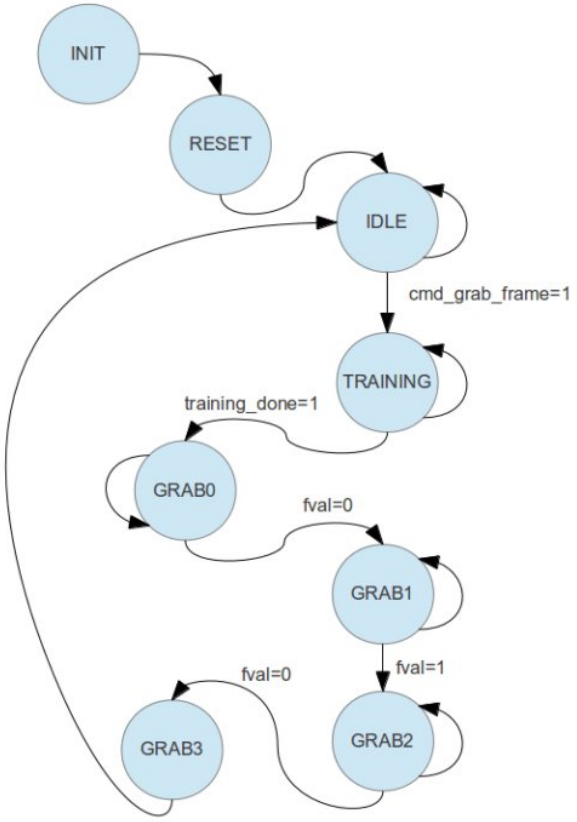
Ponieważ sensor nie generuje żadnych sygnałów ustalających początek kolejnego słowa w szeregowym strumieniu bitów, przed odbiorem danych interfejs musi zostać wytrenowany. Pierwszym krokiem algorytmu trenowania jest takie opóźnienie sygnałów z poszczególnych linii LVDS, aby znaleźć stabilny punkt próbkowania danych. Dokonuje tego automat sterujący, próbkujący wartość bitu słowa trenującego określoną ilość razy i porównując czy wartość bitu się zmieniła. Po każdej serii porównań, następuje opóźnienie sygnału o jeden krok i powtórzenie procedury od nowa. Po znalezieniu stabilnego regionu wartości bitu, wartość opóźnienia jest ustawiana tak, aby punkt próbkowania znalazł się w jego środku.

Następnym krokiem jest ustalenie początku słowa danych. Odbywa się to poprzez porównywanie odebranych danych ze znanym wzorcem i przesuwanie danych w rejestrze przesuwnym, aż do momentu kiedy nastąpi zgodność ze wzorcem. Wartość przesunięcia zostaje zapamiętana i wykorzystana do odbioru właściwych danych. Dokładny opis algorytmu trenowania można znaleźć w [6].

4.4.7 Główny automat sterujący

Zaprojektowano prosty 8 – stanowy automat sterujący, którego zadaniem jest kierowanie pracą modułu interfejsu sensora. Po uruchomieniu układu FPGA, automat znajduje się w stanie INIT, z którego przechodzi do stanu RESET. W stanie RESET generowany jest sygnał resetu dla sensora, który podtrzymywany jest przez pewną ilość taktów zegara, odliczaną przy pomocy licznika. Następnie automat przechodzi do stanu IDLE w którym pozostaje, aż do otrzymania poziomu wysokiego sygnału CMD_GRAB_FRAME.

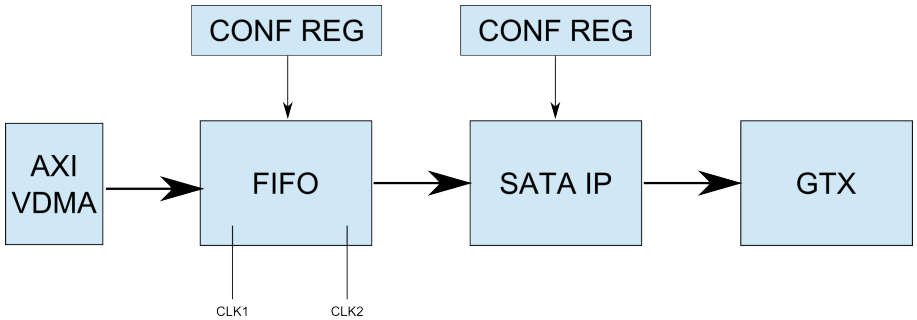
Po otrzymaniu sygnału odczytu klatki następuje trenowanie interfejsu LVDS oraz oczekiwanie na pojawienie się sygnału FRAME_VALID. Automat oczekuje sekwencji sygnału FRAME_VALID o postaci 010, a po jej pojawieniu się powraca do stanu IDLE. Rozwiązanie takie pozwala na trenowanie interfejsu przed każdą klatką obrazu oraz testowanie stabilności sygnałów sterujących, ponieważ niewystąpienie którejś z wartości spodziewanej sekwencji spowoduje zawieszenie się automatu. W wersji produkcyjnej systemu, należy rozważyć inne scenariusze trenowania oraz sterowania interfejsem tak, aby układ był możliwie najbardziej odporny na zakłócenia. Projekt był testowany pod kątem stabilności – układ pracował przez kilka godzin z uruchomionym odczytem danych z sensora – w trakcie trwania testu nie doszło do zawieszenia automatu.



Rysunek 4.10: Graf przejść głównego automatu sterującego

4.4.8 SATA IP

Zapis danych na dysk odbywa się przy pomocy bloku IP obsługującego interfejs SATA. IP zostało przystosowane do obsługi transceiveróg GTX [**?**], co pozwala na zapis danych z dużą prędkością. Dane obrazu z AXI Stream są dołączone na wejście kolejki FIFO, której wyjście jest dołączone do bloku IP interfejsu SATA. Do tego bloku dołączono również rejestry konfiguracyje, tak aby istniała możliwość sterowania logiką poprzez procesor ARM.

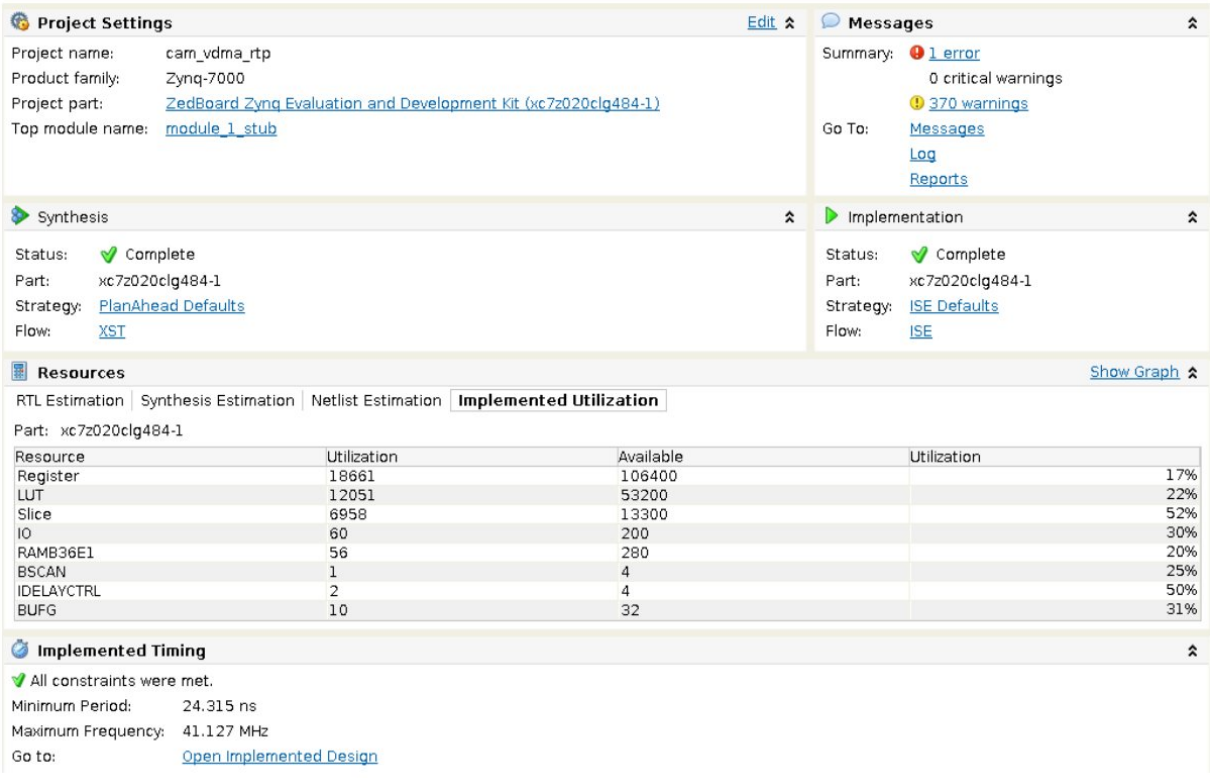


Rysunek 4.11: Schemat przesyłu danych na dysk twardy w logice programowalnej

4.4.9 Raport implementacji sprzętowej

Projekt napisano i zaimplementowano przy użyciu oprogramowania Xilinx ISE Desing Suite w wersji 14.6. Projekt części sprzętowej został zrealizowany w programie PlanAhead oraz Xilinx Platform Studio. Implementacja zakończyła się poprawnie, zużyto maksymalnie połowę dostępnych zasobów. Widoczny 1

błąd nie dotyczy implementacji projektu, a jest wynikiem błędu w oprogramowaniu Xilinx. Szczegółowe raporty opisujące każdy etap syntezy oraz implementacji dostępne są w katalogu projektu.



Rysunek 4.12: Raport implementacji sprzętowej w programie PlanAhead

4.5 Część programowa

Część programową projektu napisano w języku C za pomocą środowiska programistycznego Xilinx SDK, będącego częścią ISE Design Suite.

4.5.1 System operacyjny

Do sterowania sprzętem oraz komunikacji serwera z klientem wykorzystano prosty system operacyjny o otwartym kodzie źródłowym FreeRTOS [7] w wersji 7.0.2. Jest to system czasu rzeczywistego opracowany specjalnie dla systemów wbudowanych, charakteryzujący się prostotą, szybkością i bardzo małym wykorzystaniem zasobów. Wspiera wielozadaniowość, mechanizmy synchronizacji wątków oraz programowe liczniki czasu. Zapewnia podstawowe mechanizmy zarządzania pamięcią, zadania (wątki systemu) operują we wspólnej przestrzeni adresowej. FreeRTOS współpracuje z biblioteką lwIP, która zapewnia implementację stosu TCP/IP dla systemów wbudowanych, zaprojektowaną pod kątem minimalnego wykorzystania zasobów. W projekcie użyto bibliotekę lwIP w wersji 1.4.0. W projekcie wykorzystano 2 wersję algorytmu zarządzania stertą (heap_2.c). Pozwala ona na rezerwowanie i zwalnianie bloków pamięci. Nie łączy ze sobą sąsiednich wolnych bloków, więc nie nadaje się do zastosowań gdzie tworzone i zwalniane są wątki o zróżnicowanym rozmiarze sterty. W projekcie taka sytuacje nie występuje, więc wersja 2 algorytmu jest wyborem optymalnym. Zrezygnowano z używania funkcji bibliotecznych dostarczanych przez Xilinx (malloc i free), ponieważ podczas testowania okazało się, że zachowują się niestabilnie – czasami podczas uruchamiania programu zwracają błąd niemożliwości przydziału pamięci, mimo że pamięć wcześniej nie była zajęta.

4.5.2 Mapa przestrzeni adresowej

Poniżej przedstawiono mapę adresów podstawowych komponentów projektu. Mapa nie wyczerpuje wszystkich elementów, pełną mapę można znaleźć w plikach projektu Xilinx Platform Studio.

Pozycja	Komentarz	Adres bazowy	Adres końcowy
processing_system_7_0	Zakres adresów pamięci RAM	0x00000000	0x1FFFFFFF
axi_spi_0	axi_spi 1.02.a	0x42000000	0x4200FFFF
axi_vdma_3	axi_vdma 5.04.a	0x43000000	0x4300FFFF
axi_vdma_2	axi_vdma 5.04.a	0x43020000	0x4302FFFF
axi_vdma_1	axi_vdma 5.04.a	0x43040000	0x4304FFFF
axi_vdma_0	axi_vdma 5.04.a	0x43060000	0x4306FFFF
cam_control_0	cam_control 1.00.a	0x74400000	0x7440FFFF
VIDEO_BASEADDR	Zakres adresów bufora ramki VDMA	0x08000000	0x17FFFFFF
PRVW_ADDR0	Zakres adresów 1 ramki bufora podglądu	0x18000000	0x181FFFFF
PRVW_ADDR1	Zakres adresów 2 ramki bufora podglądu	0x18200000	0x183FFFFF

Rysunek 4.13: Mapa adresów komponentów projektu

4.5.3 Aplikacja serwera

Oprogramowanie serwera stworzone na potrzeby projektu wykorzystuje obydwa rdzenie ARM Cortex A9 w trybie AMP (asymmetric multiprocessing). Tryb AMP pozwala na uruchomienie na każdym rdzeniu procesora oddzielnego systemu operacyjnego. Na rdzeniu CPU0 uruchamiany jest FreeRTOS, natomiast CPU1 pracuje bez żadnego systemu operacyjnego w tzw. trybie bare metal. CPU0 odpowiada za:

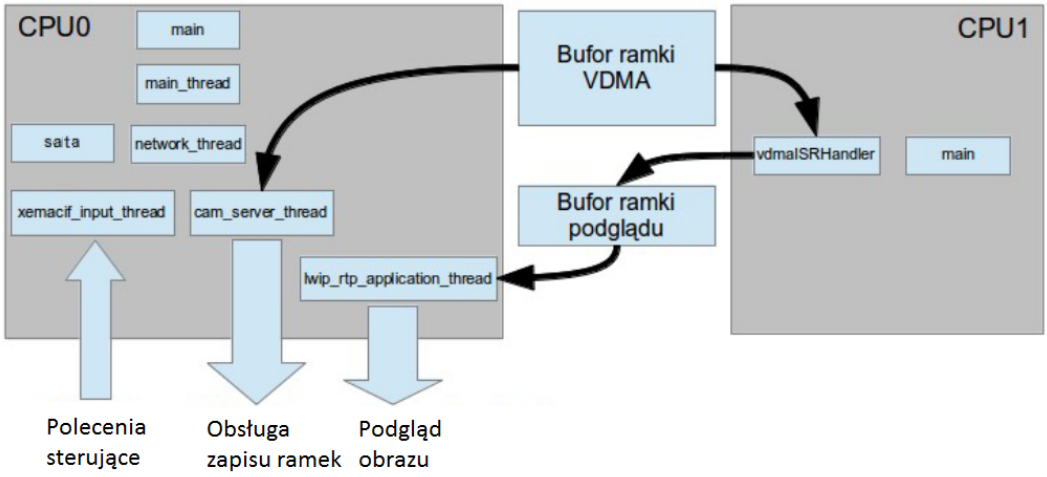
- Zarządzanie i programowanie układów peryferyjnych projektu w tym DMA
- Komunikację z klientem przez Ethernet używając stosu lwIP
- Sterowanie i kontrola zapisu obrazu na dysk twardy
- Komunikację z Internal Management Unit - systemem rejestracji parametrów lotu

CPU1 odpowiada za:

- Obsługę przerwania od rdzenia IP AXI VIDEO DMA
- Konwersję oraz podwójne buforowanie danych obrazu

Synchronizacja programów uruchomionych na poszczególnych rdzeniach procesora odbywa się za pomocą semaforów znajdujących się w pamięci OCM SRAM (przy wyłączonej pamięci podręcznej w obszarze jej adresów, zapewnia deterministyczny czas dostępu oraz mniejsze opóźnienie od pamięci

DRAM). Obydwa rdzenie korzystają ze wspólnej przestrzeni adresowej, co wymusza ostrożne zarządzanie obszarami pamięci oraz adresowaniem.



Rysunek 4.14: Schemat aplikacji serwera

Dane obrazu generowane przez część sprzętową projektu zapisywane są na bieżąco w pamięci DRAM w buforze ramki VDMA. Jednocześnie co pewną ilość zapisanych klatek obrazu, generowane jest w module AXI VDMA przerwanie podłączone do rdzenia procesora CPU1. Przerwanie to obsługiwane jest przez funkcję vdmaISRHandler(), której zadaniem jest przygotowywanie obrazu podglądu. Przed wysłaniem obrazu przez SATA, dane obrazu muszą zostać uporządkowane (sensor wysyła dane nie po kolei), opcjonalnie przekonwertowane na inny format oraz podwójnie zbuforowane tak, aby obraz podglądu był wolny od zniekształceń. Tak przygotowane dane są mapowane na pakiety i zapisywane za pomocą SATA IP do dysku twardego.

Adres IP płyty ZC706 jest zakodowany na stałe i wynosi 192.168.0.154:10001. Zmiana adresu na inny wymaga przekompilowania programu. Możliwe jest też zaimplementowanie obsługi DHCP.

Równolegle do innych wątków na CPU0 uruchomiony jest proces rejestracji parametrów lotu, który w czasie rzeczywistym zapisuje te dane do pamięci podręcznej na karcie SD. W zależności od konfiguracji istnieje możliwość przyporządkowania parametrów lotu do danej klatki. Podczas akwizycji ramki, zostaje wygenerowane przerwanie które dodaje aktualne dane o pozycji samolotu/drona, czasu wykonania pomiaru, parametrów obrazu, oraz wysokości na końcu ramki. Dane są zapisane w buforze zaraz za klatką i mają określoną strukturę dzięki czemu w prosty sposób można wydobyć te dane po zapisie na dysk twardy.

4.5.4 Komunikacja serwera z klientem

Komunikacja klienta z serwerem odbywa się przez Ethernet. Za odbiór i wykonanie rozkazów odpowiada wątek cam_server_thread. Rozkazy dla serwera mają postać łańcuchów znaków, podobnie jak ma to miejsce w protokole HTTP. Niektóre z rozkazów po łańcuchu znaków zawierają także pewną wartość liczbową zapisaną na określonej liczbie bitów, np. SET_EXPOSURE zawiera 4 bajtową wartość czasu naświetlania, reprezentowaną jako liczba całkowita bez znaku. Możliwe jest wysyłanie rozkazów dla serwera z dowolnego programu terminala, obsługującego protokół TCP/IP. Do poprawnego działania biblioteki lwIP musi być dodatkowo uruchomiony wątek xemacif_input_thread, który odpowiada za odbiór danych od PHY.

Rozkazy obsługiwane przez serwer:

- Zatrzymanie podglądu STOP_PREVIEW
- Wyzwolenie rejestracji serii klatek TRIG
- Wysłanie do klienta uprzednio zarejestrowanej serii klatek DLOAD

- Wyświetlenie w konsoli informacji diagnostycznych DBUG
- Uzbrojenie wyzwalacza ARM
- Ustawienie czasu naświetlania SET_EXPOSURE + 4 bajtowa liczba całkowita bez znaku (wartość czasu naświetlania)
- Zaprogramowanie rejestru sensora PROG_REG + 1 bajt adresu rejestru + 1 bajt wartości rejestru SPI
- Konfiguracja parametrów rejestracji lotu SET_FLIGHT_MODE + 8 bajtów konfiguracyjnych

4.5.5 Wykaz i opis plików źródłowych

app_cpu0_freertos

- *addr_defs.h* - Plik zawiera podstawowe parametry przetwarzania oraz adresy i zakresy pamięci semaforów i buforów ramki wideo.
- *app.h*, *app.c* Główny moduł aplikacji serwera – zawiera implementację wątków działających na CPU0.
- *cam_rtp.h*, *cam_rtp.c* Pliki zawierające implementację podzbioru protokołu RFC 4175, do strumieniowania podglądu na żywo. W obecnej wersji oprogramowania, ramki podglądu wysyłane są z częstotliwością ok. 1 klatki/s przy rozdzielczości 1024x1024 punkty. Obie wartości mogą zostać zmienione na dowolne, zgodne z protokołem RFC 4175 i nie przekraczające przepustowości łącza sieciowego.
- *cam_tools.h*, *cam_tools.c* Źródła z implementacją funkcji pomocniczych, takich jak konwersja RGB - YCbCr422.
- *cam_types.h* Deklaracje struktur danych używanych przy pakietyzacji RTP.
- *lwip_rtp.h*, *lwip_rtp.c* Implementacja funkcji do wysłania ramek podglądu przez ethernet.
- *platform_config.h* Definicje stałych konfiguracyjnych platformy Zynq.
- *xaxivdma_tools.h*, *xaxivdma_tools.c* Implementacja funkcji obsługujących VDMA (programowanie, diagnostyka) oraz SPI.
- *sata.c*, *sata.h* Obsługa i konfiguracja zapisu danych na dysk twardy.
- *imu.c*, *imu.h* Obsługa komunikacji z jednostką centralną UAV/samolotu, obsługa rejestracji parametrów lotu.
- *gps.c*,*gps.h* Driver modułu GPS do redundantnej lokalizacji UAV/samolotu.
- *main.c* Plik główny aplikacji działającej na rdzeniu CPU0.

app_cpu_1

- *app_cpu1.c* Plik główny aplikacji działającej na rdzeniu CPU1.
- *cam_tools.h*, *cam_tools.c*, *cam_types.h* Kopie plików

app_cpu0_freertos_bsp, *app_cpu1_bsp* BSP(Board Support Package) to podprojekty zawierające kody źródłowe sterowników, bibliotek systemowych, bibliotek obsługi sieci itd. W przypadku aplikacji cpu0, BSP zawiera także kod źródłowy systemu FreeRTOS. Podprojekty BSP są generowane automatycznie przez środowisko Xilinx SDK i konfigurowane przez użytkownika według potrzeb.

module_1_hw_platform

Platforma sprzętowa (HW platform) to podprojekt zawierający pliki konfiguracyjne modułu Zynq ładowane podczas programowania układu (ustawienia PLL, wejść/wyjść, adresy poszczególnych modułów sprzętowych itp.). Platforma jest generowana przez program Xilinx Platform Studio, na podstawie plików projektu Zynq

4.6 Aplikacja klienta

Zgodnie z założeniami, zadaniem programu klienta jest wysyłanie poleceń do serwera. Program klienta został napisany w języku C++ i jest przeznaczony do uruchamiania w systemie Linux. Na potrzeby programu stworzono bardzo prosty, tekstowy interfejs użytkownika przy pomocy biblioteki ncurses. Dane obrazowe są zapisywane na dysk przy pomocy biblioteki OpenCV. Obsługuje ona wiele popularnych formatów graficznych oraz charakteryzuje dużą szybkością działania.

Program klienta składa się z jednej klasy – CamClient, implementującej jako metody funkcje wysyłające poszczególne rozkazy do serwera. Liczba metod odpowiada liczbie rozkazów przedstawionej w jednym z poprzednich punktów. Wszystkie metody oprócz jednej działają jednostronnie – wysyłają rozkazy nie oczekując odpowiedzi. Ciężar poprawnego dostarczenia rozkazu spoczywa na łączu Ethernet i protokole TCP/IP (który wewnętrznie zawiera mechanizmy potwierdzenia dostarczenia wiadomości). Jedyną metodą dwukierunkową jest Download(), która po wysłaniu rozkazu dostarczenia danych obrazowych przez serwer, oczekuje na dane przez pewien określony czas (aktualnie 1s). W razie braku rozpoczęcia transmisji w wymaganym czasie generowany jest błąd.

4.6.1 Wykaz plików źródłowych

- main.cpp Główny plik aplikacji – zawiera instancję klasy CamClient, oraz implementację prostego interfejsu użytkownika.
- cam_tools.h, cam_tools.cpp Pliki z funkcjami pomocniczymi
- cam_client.h, cam_client.cpp Pliki zawierające implementację klasy CamClient.

Rozdział 5

Podsumowanie

Creotech Instruments S.A. był odpowiedzialny za realizację bloku elektroniki do projektu kamery spektralnej zleconego przez Centrum Badań Kosmicznych Polskiej Akademii Nauk. System został wykonany zgodnie z założeniami projektowymi i pozwala na rejestrację obrazów w czasie rzeczywistym z czujnika CMV4000 wraz z akwizycją parametrów lotu oraz zapisem danych na dysk twardy.

Przygotowana platforma wykorzystuje nowoczesny układ SoC Xilinx Zynq posiadający zalety za równo układu FPGA oraz zaawansowanego procesora 32 bitowego. Wykorzystany czujnik jest układem pozwalającym na bardzo szybką akwizycję obrazu nawet do 180 kl/s. Zapis obrazu wraz z parametrami lotu odbywa się poprzez dedykowany interfejs SATA. Sterowanie i konfiguracja systemu odbywa się poprzez interfejs Ethernet.

System jest urządzeniem gotowym do testów i dalszego rozwoju.

Bibliografia

[1] Xilinx ZC706 Evaluation Board http://www.xilinx.com/products/boards-and-kits/EK-Z7-ZC706-G.htm

[2] CMOSIS CMV4000 Sensor http://www.cmosis.com/products/standard_products/cmv4000

[3] CIS 1910F Sensor

[4] http://www.steadidrone.eu/uav-hexacopter-h6x-for-precise-agriculture/

[5] LogiCORE IP AXI Interconnect v1.06.a, Xilinx, DS768 December 18, 2012

[6] Application note for CMV2000 and CMV4000 , Interface Training , CMV2000-AN3-v1 , CMOSIS NV, 2009

[7] www.freertos.org, Real Time Engineers Ltd.

[8] 7 Series FPGAs SelectIO Resources User Guide , UG471 (v1.3), Xilinx, October 31, 2012

[9] Zynq-7000 All Programmable SoC (XC7Z010, XC7Z015, and XC7Z020): DC and AC Switching Characteristics, DS187 (v1.8), Xilinx, September 12, 2013 [**?**] http://www.serialata.org/