

**AKADEMIA GÓRNICZO-HUTNICZA
IM. STANISŁAWA STASZICA W KRAKOWIE**

Wydział Informatyki, Elektroniki i Telekomunikacji
Katedra Informatyki



PROJEKT INŻYNIERSKI
**Uniwersalny moduł sprzętowego
przetwarzania danych oparty na FPGA**

DOKUMENTACJA TECHNICZNA

Autor: Krzysztof Papciak
Opiekun: dr inż. Jacek Długopolski

Spis treści

1. Wstęp	3
2. Dziedzina problemu.....	4
2. 1. Sformułowanie zadania projektowego.....	4
2. 2. Układy FPGA	4
2. 3. Rodzina układów FPGA Cyclone III firmy Altera	5
2. 4. Uzasadnienie wyboru układu FPGA Altera Cyclone III i jego zastosowanie w projekcie.....	8
3. Projekt schematu modułu SNF-0 opartego o układ FPGA.....	9
3. 1. Ogólny opis modułu.....	9
3. 2. Układ FPGA i jego otoczenie	10
3. 3. Pamięć SSRAM	12
3. 4. Port szeregowy	13
3. 5. Port PS2	14
3. 6. Przyciski i diody LED.....	14
3. 7. Porty wejścia/wyjścia ogólnego przeznaczenia i wyprowadzenia napięć zasilających	15
3. 8. Wyjście VGA2.....	16
3. 9. Wyjście VGA1, przetwornik cyfrowo analogowy i bufor obrazu	17
3. 10. Moduł zasilający	19
4. Projekt i wykonanie obwodu drukowanego modułu SNF-0.....	21
4.1. Projekt obwodu drukowanego	21
4.2. Wykonanie płytki obwodu drukowanego i montaż elementów	22
4.2. Uruchomienie i testowanie modułu SNF-0	22
5. Projekt i wykonanie przykładowego specjalizowanego procesora graficznego.....	23
5. 1. Kontroler klawiatury PS2	24
5. 2. Moduł kontroli skalowania i obrotu	24
5. 3. Moduł pamięci i struktura danych modelu 3D	25
5. 4. Moduł przekształceń geometrycznych	25
5. 5. Kontroler wyświetlania	29
5. 6. Generator odcinków	31
5. 7. Kontroler VGA.....	31
5. 8. Eksport pliku z programu 3DS Max do pamięci układu FPGA	32
6. Spis rysunków	33

1. Wstęp

Dokument ten zawiera opis techniczny projektu inżynierskiego *Uniwersalny moduł sprzętowego przetwarzania danych oparty na FPGA*, realizowanego na Akademii Górnictwo-Hutniczej w Krakowie.

Drugi rozdział opisuje dziedzinę problemu i wymagania, jakie mają być realizowane przez moduł uruchomieniowy SNF-0. W dalszej jego części krótko opisano układy FPGA, następnie scharakteryzowano rodzinę układów *Cyclone III* firmy *Altera*. Na koniec uzasadniono jej wybór i opisano jej zastosowanie w projekcie inżynierskim.

Rozdział trzeci opisuje schemat elektryczny modułu SNF-0. Omówione zostały poszczególne elementy projektu i przedstawione fragmenty schematów.

W rozdziale czwartym przedstawiono proces projektowania płytki PCB oraz jej wykonanie, uruchomienie i testowanie.

Ostatni rozdział przedstawia szczegóły implementacyjne specjalistycznego procesora graficznego wyświetlającego grafikę 3D zbudowaną z linii.

2. Dziedzina problemu

W tym rozdziale przedstawiono cele zadania projektowego oraz po krótkie opisano układy FPGA ze szczególnym uwzględnieniem rodziny *Cyclone III* firmy *Altera*. W ostatnim podrozdziale uzasadniono zastosowanie FPGA tej rodziny

2. 1. Sformułowanie zadania projektowego

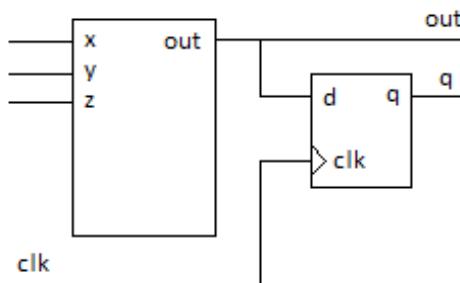
Celem projektu inżynierskiego było stworzenie uniwersalnej platformy do sprzętowego przetwarzania danych, zrealizowanej w oparciu o układ FPGA. Moduł docelowo ma służyć do łatwego projektowania, uruchamiania i testowania procesorów zaprojektowanych w językach opisu sprzętu, takich jak VHDL i Verilog.

Dodatkowym celem wykonanej pracy było zaprojektowanie i wykonanie przykładowego procesora, prezentującego możliwość platformy. Co za tym idzie, należało także przygotować biblioteki pozwalające na podstawową obsługę peryferiów, takich jak porty wejścia/wyjścia oraz umożliwienie dwukierunkowej komunikacji z użytkownikiem i komputerem.

2. 2. Układy FPGA

FPGA (*field programmable gate array*) jest rodzajem programowalnego układu logicznego, którego głównym podzespołem jest dwuwymiarowa macierz elementów logicznych łączonych ze sobą za pośrednictwem przełącznic. Każda z jednostek logicznych może zostać skonfigurowana do wykonywania prostej operacji, natomiast programowalny przełącznik dostosowany do tworzenia odpowiedniego połączenia między poszczególnymi komórkami logicznymi. Projekt systemu oparty na układzie FPGA może zostać zaimplementowany poprzez specyfikację funkcji każdej z komórek logicznych i selektywną konfigurację programowalnych przełączników.

Projektowanie dużego systemu opartego o układ FPGA jest skomplikowanym procesem, złożonym z wielu zaawansowanych przekształceń, wykorzystujących algorytmy optymalizacyjne. Do tego celu wymagane jest oprogramowanie, które automatyzuje niektóre z tych zadań. Do zaprojektowania opisywanego systemu użyto oprogramowania *Quartus II Web Edition* firmy *Altera*, w wersji 13.1.



Rys. 1. Schemat koncepcyjny jednostki logicznej opartej o tablicę LUT

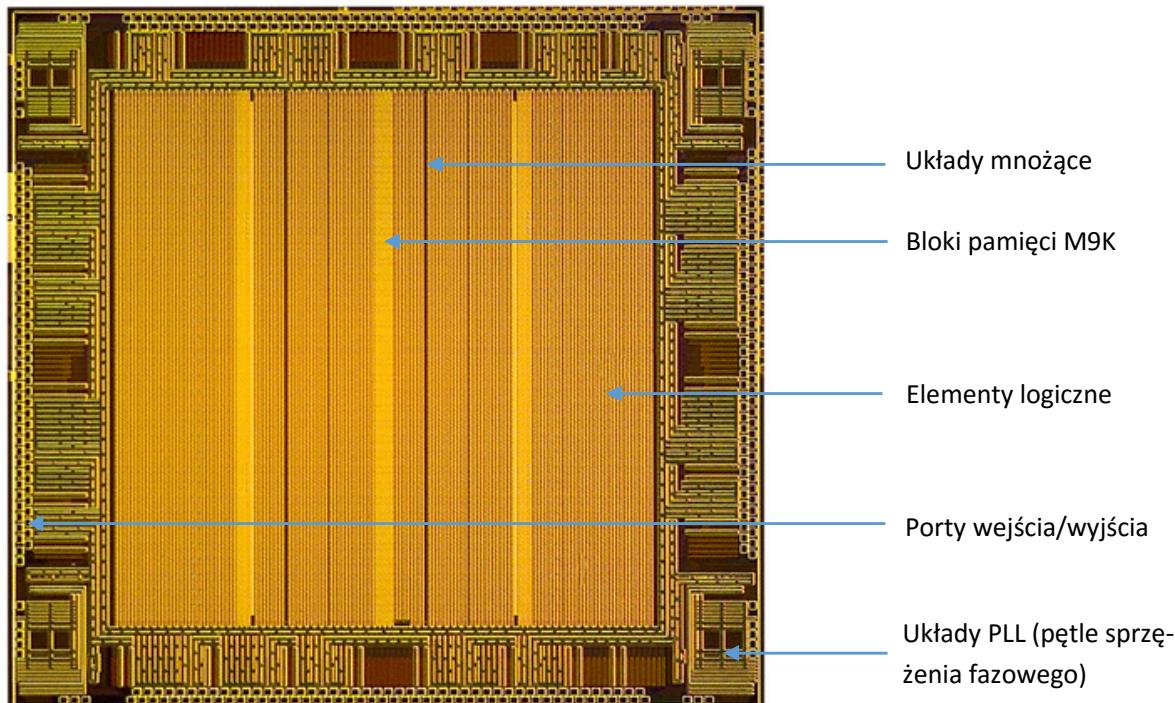
Podstawowym elementem budującym układ FPGA jest element logiczny (LE), który zawiera mały konfigurowalny obwód kombinacyjny z przerzutnikiem typu D (flip-flop). Najpopularniejszą metodą implementacji konfigurowalnego obwodu kombinacyjnego jest zastosowanie tzw. *Look-up table* (LUT). Jest

to pamięć implementująca prostą funkcję logiczną. Każdemu stanowi n wejście przyporządkowany jest odpowiedni stan wyjścia. Schemat koncepcyjny jednostki logicznej opartej na LUT przedstawiony został na rysunku 1.

2. 3. Rodzina układów FPGA Cyclone III firmy Altera

Do wykonania projektu inżynierskiego wybrano układ FPGA *Altera Cyclone III*.

Rodzina układów *Cyclone III* charakteryzuje się między innymi niskim poborem mocy. Układy tej rodziny posiadają od ok. 5 tys. do 200 tys. elementów logicznych oraz od 0.5Mb do 8Mb pamięci RAM.



Rys. 2. Struktura układu FPGA rodziny Cyclone II i opis podstawowych elementów

Na rysunku 2 przedstawiono zdjęcie przedstawiające strukturę układu FPGA Altera Cyclone III. Widoczne są między innymi bloki układów mnożących, elementy pamięci RAM oraz struktury wejścia/wyjścia.

Elementy i bloki logiczne w rodzinie układów Cyclone III

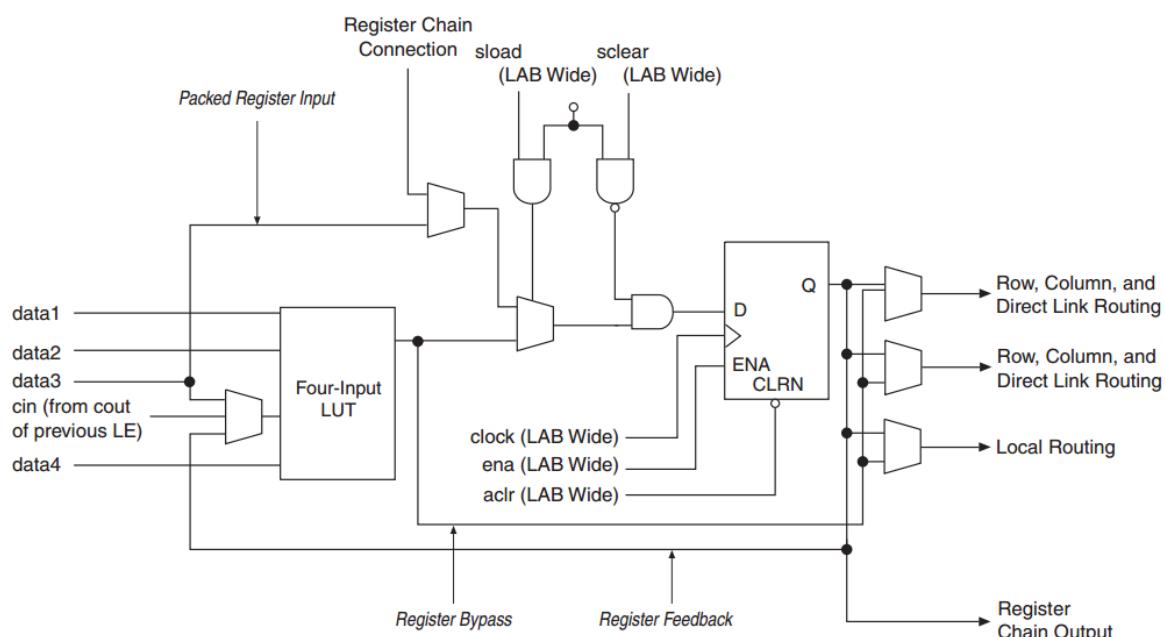
W układach rodziny *Cyclone III* bloki logiczne (*logic array blocks*) składają się z 16 elementów logicznych oraz bloku kontrolnego. Blok logiczny (LE) jest najmniejszą jednostką strukturalną w układzie FPGA rodziny *Cyclone III*.

Każdy element logiczny posiada 4 wejścia oraz czterowęjsiową tablicę LUT (*look-up table*), rejestr i logikę wyjściową. Jest to więc generator, na którym można zaimplementować każdą funkcję 4 zmiennych logicznych.

Rejestr, w każdym bloku logicznym, może zostać skonfigurowany do działania, jako przerzutnik następujących typów: D, T, JK lub SR. Każdy z rejestrów posiada sygnał danych, zegarowy, sygnał aktywacji wejścia zegarowego oraz wejście kasujące jego aktualny stan.

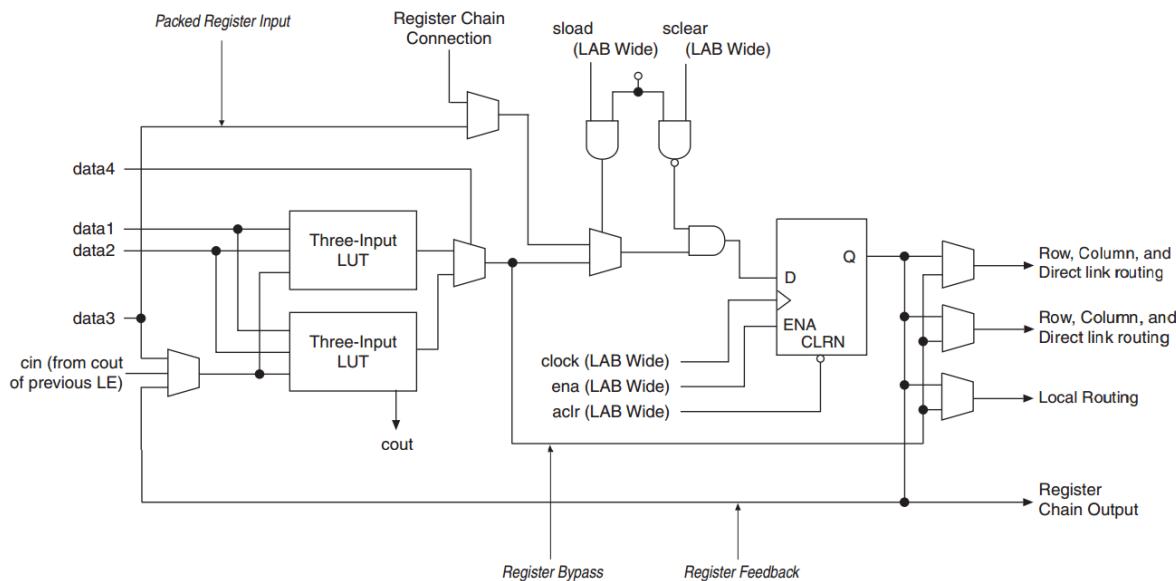
Elementy logiczne w układach *Altera Cyclone III* mogą pracować w jednym z dwóch dostępnych trybów: normalnym (*normal mode*) i arytmetycznym (*arithmetic mode*). Każdy z nich inaczej używa zasoby elementu logicznego. Oprogramowanie *Quartus II* automatycznie wybiera odpowiedni tryb dla popularnych funkcji, takich jak liczniki, sumatory, układy odejmujące.

Tryb normalny jest odpowiedni dla podstawowych aplikacji logicznych i funkcji kombinacyjnych. W tej konfiguracji, 4 wejścia z bloku logicznego kierowane są do wejść jednostki LUT.



Rys. 3. Element logiczny układu Cyclone II w trybie normalnym
(źródło: Altera Cyclone III Device Handbook. Volume 1)

Tryb arytmetyczny jest wykorzystywany do tworzenia sumatorów, liczników, akumulatorów i komparatorów. Element logiczny w trybie arytmetycznym implementuje 2-bitowy pełny sumator i prosty łańcuch przeniesień i zapożyczeń. Kompilator programu *Quartus II* automatycznie tworzy logikę tego łańcucha podczas przetwarzania.



Rys. 4. Element logiczny układu Cyclone II w trybie arytmetycznym
(źródło: Altera Cyclone III Device Handbook. Volume 1)

Bloki pamięci w rodzinie układów Cyclone III

Układy FPGA rodziny Cyclone III posiadają bloki pamięci M9K, które mogą spełniać różne funkcje, takie jak pamięć o swobodnym dostępie (RAM), rejesty przesuwne, pamięć tylko do odczytu (ROM) oraz kolejki FIFO. Bloki M9K mają pojemność 8192 bitów (9216 bitów wliczając bity parzystości) i mogą pracować w jednej z kilku konfiguracji, obejmującej następujące szerokości magistrali danych: 1, 2, 4, 8, 9, 16, 18, 32 i 36 bitów.

Bloki pamięci w układach *Cyclone III* posiadają sygnały kontrolne między innymi umożliwiające włączanie odczytu, zapisu oraz aktywację wejścia zegarowego. Do odczytu i zapisu mogą zostać użyte osobne sygnały taktujące, a więc zapis może przebiegać z inną częstotliwością niż odczyt. Możliwe jest także zastosowanie trybu *Dual-Port*, który pozwala na jednoczesny odczyt i zapis do pamięci dla różnych lokacji.

Dzięki tym funkcjom pamięci układu *Cyclone III*, wygodne jest przeprowadzenie wielu współbieżnych operacji, a także operacji wymagających różnych częstotliwości taktowania dla odczytu i zapisu (np. wyświetlanie grafiki przy użyciu wyjścia VGA).

18-bitowe jednostki mnożące w układzie FPGA Cyclone III

Układ FPGA *Cyclone III* posiada dedykowane jednostki, które mogą wykonywać mnożenie liczb bez użycia dodatkowych elementów logicznych. Mogą one działać w jednym z dwóch trybów: mnożnika 18 x 18 lub dwóch mnożników 9 x 9 bitów. Dla liczb większych niż 18 bitów, oprogramowanie *Altera Quartus II* automatycznie łączy ze sobą bloki układów mnożących. Nie ma ograniczenia wielkości czynników, jednak proces mnożenia przebiega wolniej dla większych liczb.

Oprócz dedykowanych bloków mnożników, można wykorzystać programowe układy mnożące oparte o bloki pamięci M9K i tablice LUT elementów logicznych.

Pętle sprzężenia fazowego w układach rodziny Cyclone III

Pętle sprzężenia fazowego (Phase Locked Loop – PLL) są wykorzystywane do uzyskania wymaganej częstotliwości taktowania z podstawowego sygnału zegarowego. Układ FPGA Cyclone III posiada maksymalnie cztery jednostki PLL. Każda z nich ma 5 wyjść, które mogą generować różne, jednak zależne od siebie częstotliwości. Oprogramowanie *Quartus II* może automatycznie dobrać odpowiednie ustawienia układu PLL do generowania zadanej częstotliwości sygnału wyjściowego.

2. 4. Uzasadnienie wyboru układu FPGA Altera Cyclone III i jego zastosowanie w projekcie

Układy FPGA pozwalają na łatwe prototypowanie. Dzięki możliwości wielokrotnego przeprogramowania, umożliwiają projektantom tworzenie projektów, które później mogą trafić do produkcji seryjnej, jako specjalizowane układy scalone (*Application Specific Integrated Circuit - ASIC*). W przeciwieństwie do nich układy FPGA pobierają więcej mocy i są wolniejsze.

Dzięki swej specyficznej budowie, układy FPGA pozwalają na projektowanie systemów współbieżnych. Może to znacznie przyspieszyć rozwiązywanie niektórych zadań, które stanowiłyby duży problem dla klasycznych procesorów. Ciągłe rosnąca ilość podstawowych elementów logicznych, zwiększanie ilości bloków pamięci i stosowanie specjalnych modułów, takich jak układy mnożące, pozwala na tworzenie coraz bardziej skomplikowanych projektów.

Podstawowym założeniem opisywanego projektu inżynierskiego było stworzenie modułu pozwalającego na tworzenie prototypów układów przetwarzania danych, między innymi specjalistycznych procesorów. Wykorzystanie do tego celu układu FPGA pozwala na łatwe i szybkie tworzenie takich projektów. Możliwe jest tworzenie układów intensywnie wykonujących współbieżne obliczenia, np. procesorów wielordzeniowych lub jednostek generujących grafikę.



Rys. 5. Układ FPGA Cyclone III na płytce modułu SNF-0

na tworzenie układów logicznych taktowanych z maksymalną częstotliwością ok. 400MHz. Dostępnych jest 128 portów wejścia i wejścia/wyjścia ogólnego przeznaczenia.

Jako główny element modułu SNF-0 wybrano układ FPGA *EP3C40* firmy *Altera* z rodziną *Cyclone III*, który wyposażony jest w 39600 elementów logicznych. Liczba tych jednostek pozwala na tworzenie bardziej skomplikowanych projektów. Do dyspozycji projektanta *Altera* oddała 126 bloków pamięci *M9K*, które łącznie mają pojemność ok 1.16 megabitu. Układ *EP3C40* posiada 4 pętle sprzężenia fazowego (PLL) i 126 18-bitowych jednostek mnożących.

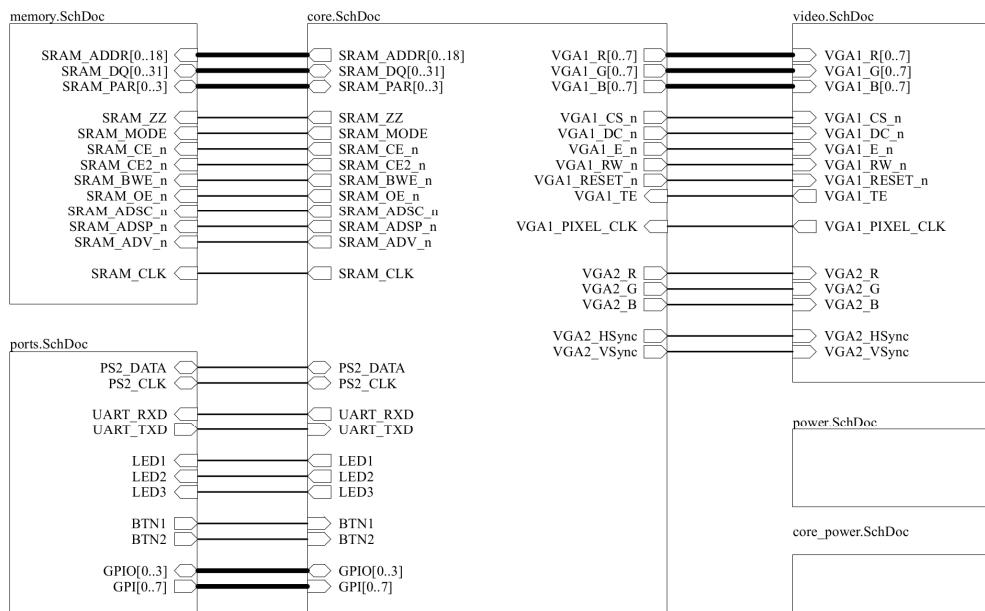
Wybrany układ FPGA znajduje się w 240-nóz- kowej obudowie PQFP. Ze względu na stosun- kowo duże rozmiary, układ ten ma ósmą (naj- wolniejszą) klasę szybkości. Pozwala jednak na tworzenie układów logicznych taktowa-

3. Projekt schematu modułu SNF-0 opartego o układ FPGA

W rozdziale trzecim omówiono poszczególne elementy schematu modułu SNF-0. Przedstawiono funkcje pełnione przez podzespoły wykorzystane do jego budowy i opisano możliwości, które ze sobą niosą.

3. 1. Ogólny opis modułu

Głównym elementem modułu SNF-0 jest układ FPGA *Altera Cyclone III*. Do niego podłączone są wyrowadzenia portów PS2 oraz portu szeregowego za pośrednictwem konwertera USB. Zestaw wyposażono także w dwa złącza VGA. Jedno z nich posiada bufor obrazu, który pozwala na asynchroniczne generowanie grafiki oraz przetwornik cyfrowo-analogowy zapewniający 24-bitową głębię kolorów. Wszystkie elementy modułu SNF-0 zasilane są przy pomocy trzech stabilizatorów impulsowych podłączonych do napięcia wejściowego 5V.



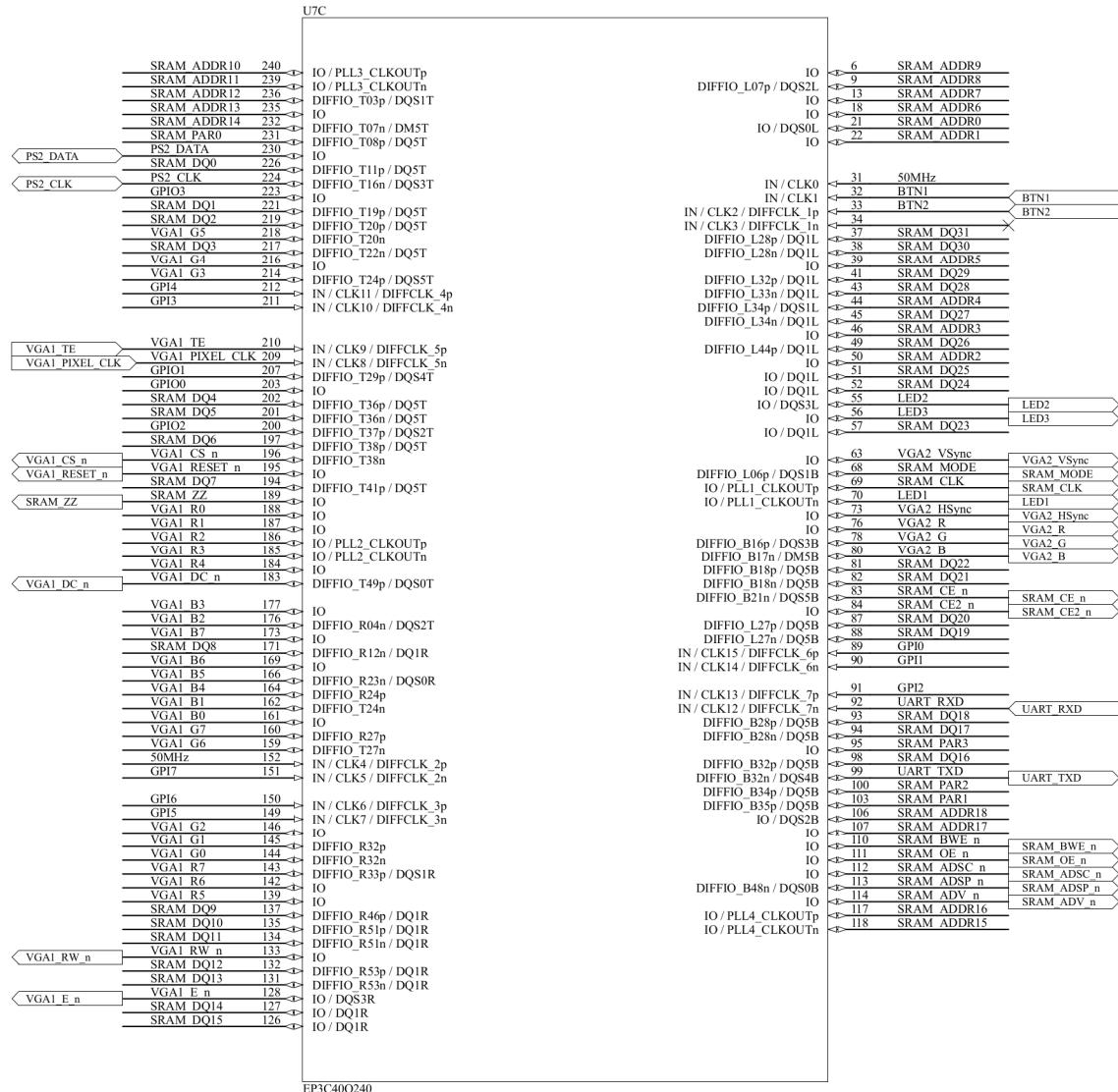
Rys. 6. Podział i połączenie dokumentów schematu modułu SNF-0

Schemat obwodu drukowanego zaprojektowano w programie *Altium Designer*. Jest on podzielony na 6 bloków (rys. 6). Plik *memory.SchDoc* przedstawia połączenia pamięci SRAM z układem FPGA oraz jej otoczenie. *core.SchDoc* zawiera schemat połączeń układu FPGA z innymi elementami modułu SNF-0 oraz układ programowania, natomiast *core_power.SchDoc* jego połączenia z napięciami zasilającymi. W pliku *ports.SchDoc* przedstawiono połączenie portów wejścia/wyjścia, PS2, portu szeregowego i przycisków oraz diod LED. Plik *video.SchDoc* prezentuje sposób podłączenia portów VGA oraz układ bufora obrazu i przetwornika cyfrowo-analogowego. W dokumencie *power.SchDoc* znajduje się schemat układu zasilania.

W poniższych podrozdziałach przedstawiono opis poszczególnych obwodów modułu SNF-0. Pełny zestaw schematów znajduje się na nośniku CD dołączonym do dokumentacji oraz w repozytorium projektu pod adresem: https://github.com/papciakk/FPGA_Board w folderze *doc*, w pliku *schematic.pdf*.

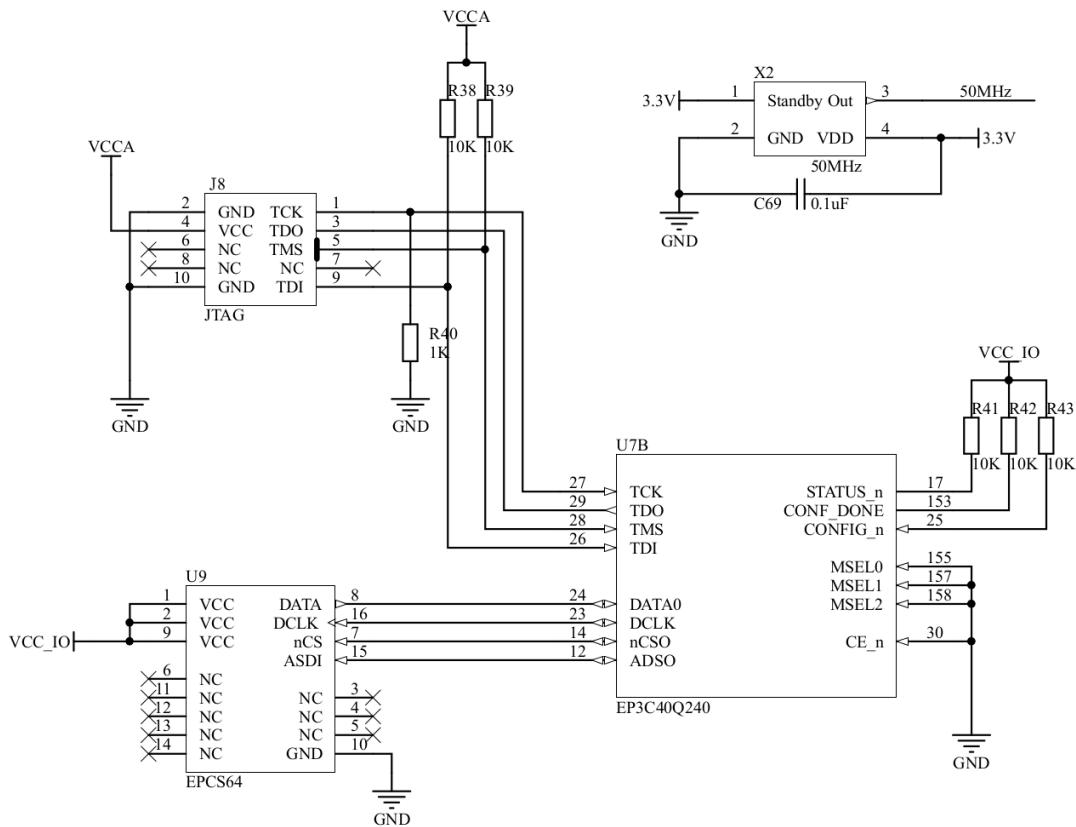
3. 2. Układ FPGA i jego otoczenie

Rysunek 7 prezentuje podłączenie portów wejścia/wyjścia układu FPGA *Cyclone III* do poszczególnych podzespołów modułu.



Rys. 7. Schemat podłączenia portów wejścia/wyjścia układu FPGA

Układ FPGA modułu SNF-0 programowany jest za pomocą programatora zgodnego z Altera USB-Blaster. Konfiguracja wykonywana jest w trybie JTAG. Domyślnie konfiguracja układu FPGA zanika po wyłączeniu zasilania. Moduł SNF-0 daje możliwość wykorzystania układ szeregowej pamięci EEPROM, który może ją przechowywać (w obecnym stanie nie jest wykorzystywana).

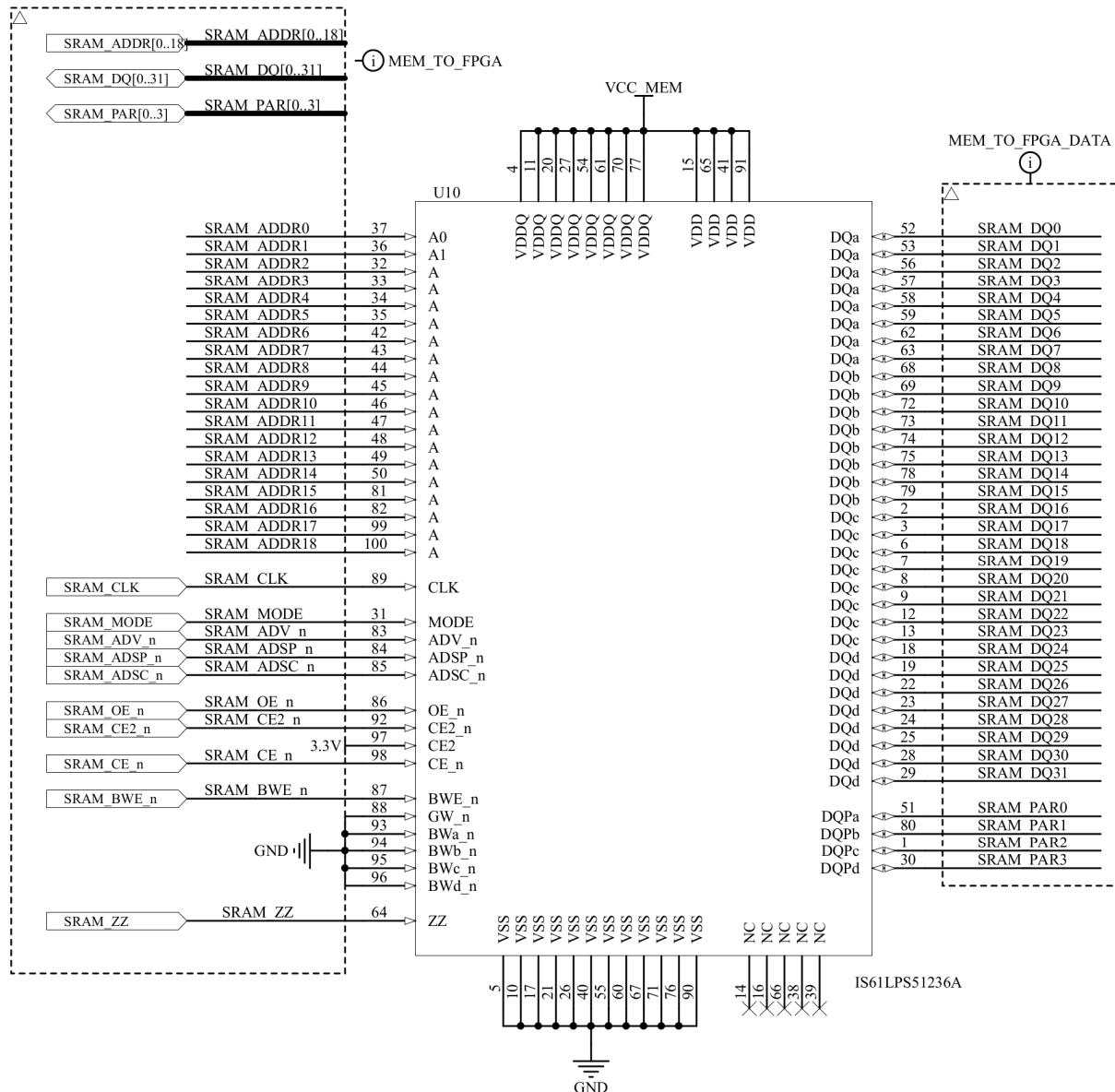


Rys. 8. Schemat układu programowania i taktowania modułu SNF-0

Na rysunku 8 pokazano schemat układu programowania i oscylator generujący główny sygnał zegarowy modułu SNF-0. Złącze programatora zgodnego z USB-Blaster zaznaczono symbolem J8. Układ U9 to szeregowa pamięć EEPROM mogąca przechowywać konfiguracje układu FPGA po odłączeniu zasilania. U7B to wyprowadzenia układu FPGA odpowiedzialne za programowanie. Piny *MSEL[0..2]* konfiguruają tryb programowania układu *Cyclone III*. Na schemacie symbolem X2 oznaczony został oscylator MEMS generujący sinusoidalny sygnał taktujący o częstotliwości 50MHz, który podłączony jest do wejść zegarowych układu FPGA.

3. 3. Pamięć SSRAM

Pamięć SSRAM (*Synchronous Static Random Access Memory*) jest podłączona do układu FPGA poprzez specjalizowany interfejs pamięci zewnętrznej, który udostępnia rodzina FPGA *Cyclone III* firmy Altera.

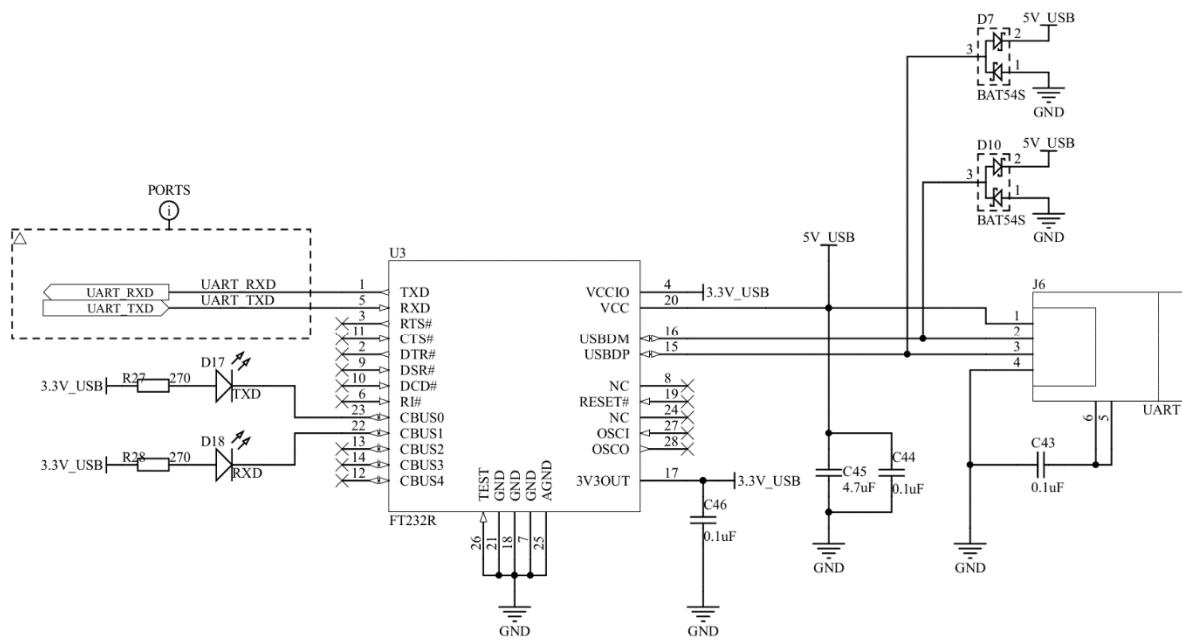


Rys. 9. Schemat podłączenia pamięci SSRAM do układu FPGA

Układ ten oznaczony jest na schemacie (rysunek 9) symbolem U10. Pamięć SRAM komunikuje się z układem FPGA używając 19-bitowej magistrali adresowej i 36-bitowej szyny danych. Dodatkowo wykorzystane są sygnały sterujące umożliwiające użycie *Burst Mode*, w którym dane przesypane są pakietami, co znacznie przyspiesza komunikację. Sygnał zegarowy pamięci U10 generowany jest bezpośrednio przez układ FPGA. Pozwala to na jego dowolne dostosowywanie i synchronizację transmisji danych. Układ pamięci *IS61LPS51236* użyty w projekcie może pracować z maksymalną częstotliwością 200MHz.

3. 4. Port szeregowy

Podstawowym portem komunikacyjnym wykorzystywany w module uruchomieniowym SNF-0 jest port szeregowy zgodny ze standardem RS-232, podłączany do komputera za pomocą konwertera z portu USB. Zastosowanie tego sposobu komunikacji pozwala standardowo na osiągnięcie prędkości transmisji na poziomie ok. 10.5 KB/s, co powinno wystarczyć do realizacji podstawowych projektów.



Rys. 10. Schemat interfejsu portu szeregowego

Układ U3 (*FT232R* firmy *FTDI Chips*), przedstawiony na rysunku 10 jest zintegrowanym interfejsem USB do portu szeregowego, który oprócz podstawowej funkcjonalności, posiada także między innymi zintegrowaną pamięć EEPROM, generator kwarcowy i stabilizator o niskim spadku napięcia.

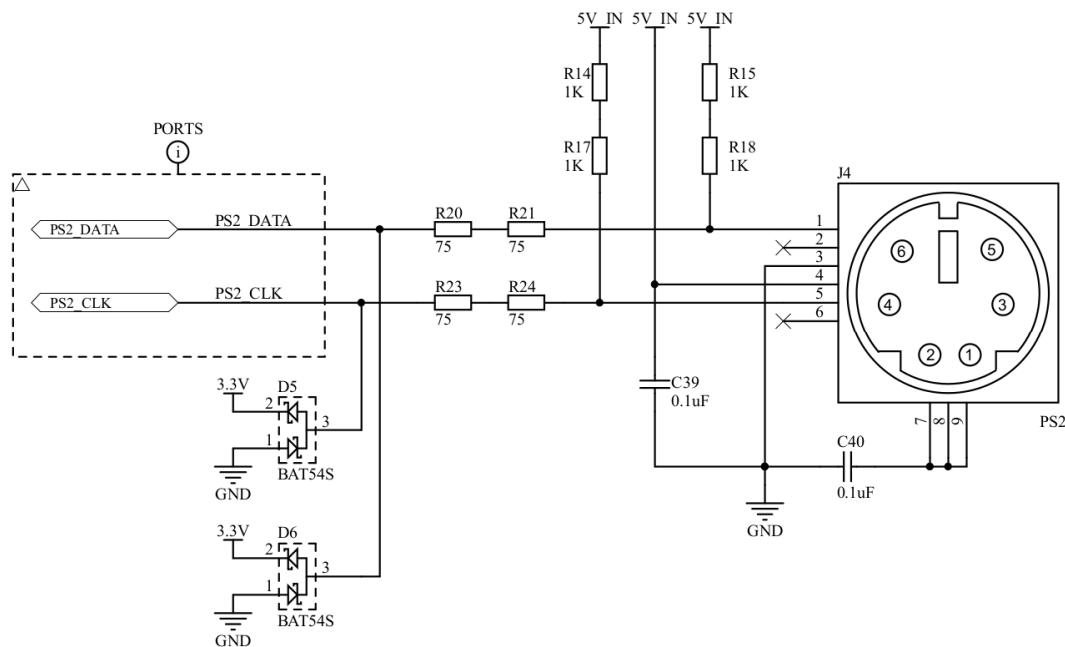
Układ firmy *FTDI Chips* oferuje także 5 konfigurowalnych portów ogólnego przeznaczenia. W opisywanej aplikacji podłączone są do diod LED, które wskazują trwającą transmisję (D17) i odbiór (D18). Interfejs USB do portu szeregowego oraz diody wskazujące transmisję danych zasilane są bezpośrednio z portu USB przy użyciu wewnętrznego stabilizatora, który obniża napięcie z 5V do 3.3V. Wykorzystano proste zabezpieczenie przy użyciu diod D7 i D10, które redukują skoki napięcia w razie przepięć elektrostatycznych lub pojawiennia się zbyt wysokiego lub ujemnego napięcia na portach transmisji danych standardu USB.

Ze względu na dużą integrację, układ firmy *FTDI Chips* pozwala na łatwe wykorzystanie, jako konwerter USB – port szeregowy. Dzięki niemu nie jest potrzebne użycie standardowego gniazda RS-232, które we współczesnych komputerach jest rzadkością. Zbędna staje się także konwersja poziomów logicznych, która byłaby potrzebna przy zastosowaniu klasycznego rozwiązania.

Układ *FT232R* wymaga zainstalowania sterowników, które można pobrać ze strony producenta. Jest tam także dostępne oprogramowanie, które pozwala na konfigurację ustawień układu, zapisanych w pamięci EEPROM. Możliwe jest między innymi ustalenie prezentowanej nazwy interfejsu, czy też konfiguracja portów ogólnego przeznaczenia.

3. 5. Port PS2

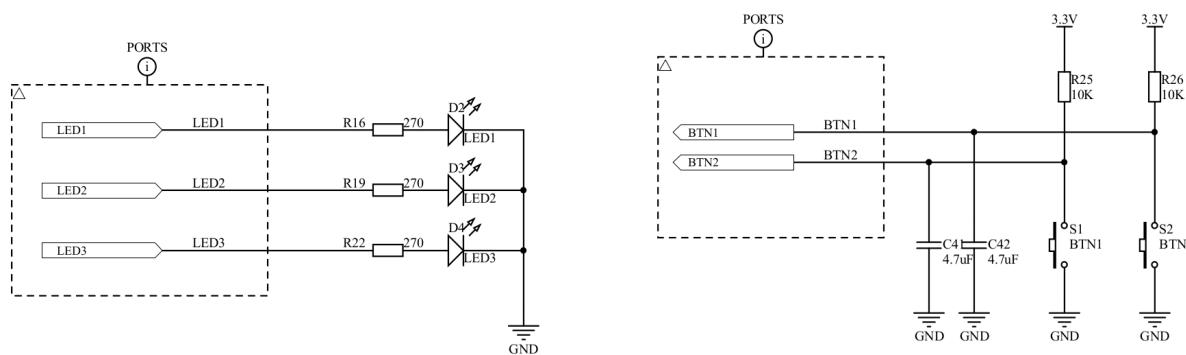
Rysunek 11 przedstawia schemat podłączenia portu PS2 do układu FPGA, które zostały zabezpieczone przed skokami napięcia za pomocą diod D5 i D6. Zastosowanie tego portu umożliwia łatwe podłączenie do modułu SNF-0 klawiatury lub myszki zgodnej ze standardem PS2.



Rys. 11. Schemat podłączenia portu PS2

3. 6. Przyciski i diody LED

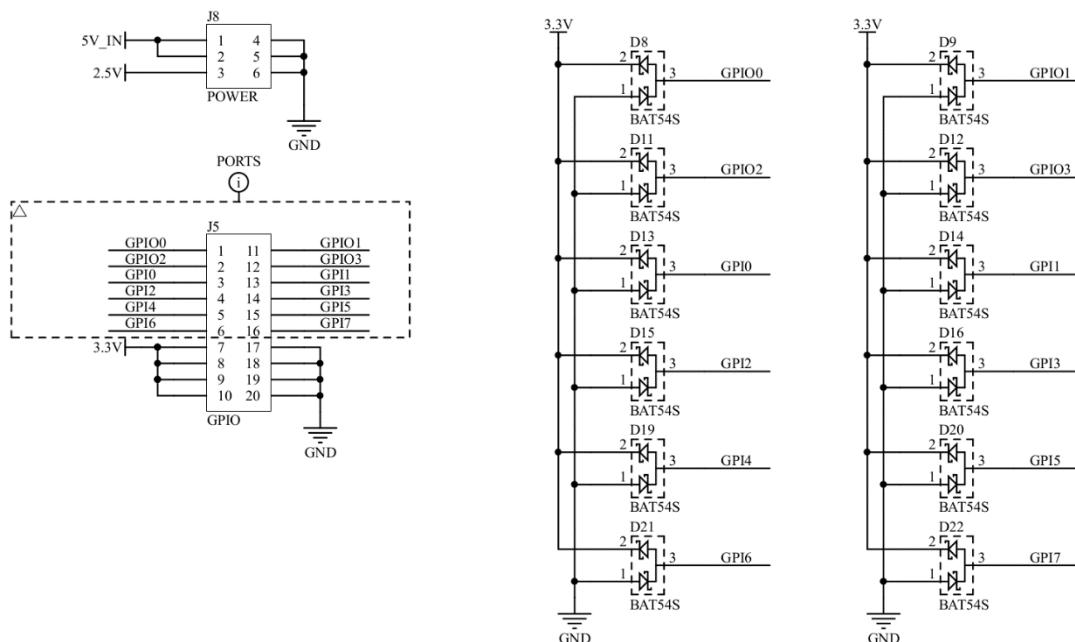
Elementami zestawu SNF-0 są 3 diody LED oraz 2 przyciski, posiadające prosty filtr redukujący niepożądane efekty związane z drgiem styków. Zarówno diody, jak i przyciski mogą zostać wykorzystane w projektach w dowolny sposób, pozwalając na interakcję z użytkownikiem. Schemat podłączenia omawianych elementów pokazano na rysunku 12.



Rys. 12. Schemat podłączenia diod LED i przycisków do układu FPGA

3. 7. Porty wejścia/wyjścia ogólnego przeznaczenia i wyprowadzenia napięć zasilających

Dostępne są 4 porty wejścia/wyjścia i 8 portów wejściowych ogólnego przeznaczenia (przedstawione na rysunku 13). Wyprowadzono także 4 napięcia zasilania o wartości 3.3V, dwa o wartości 5V i jedno 2.5V. Porty zostały zabezpieczone diodami chroniącymi układ FPGA przed skokami napięcia.



Rys. 13. Schemat portów wejścia/wyjścia ogólnego przeznaczenia

Wyprowadzenia wejścia/wyjścia, podłączone bezpośrednio do układu FPGA, mogą służyć do komunikacji z zewnętrznymi modułami. Wejścia mogą zostać także wykorzystane do dostarczania sygnału zegarowego do taktowania logiki zbudowanej na układzie FPGA.

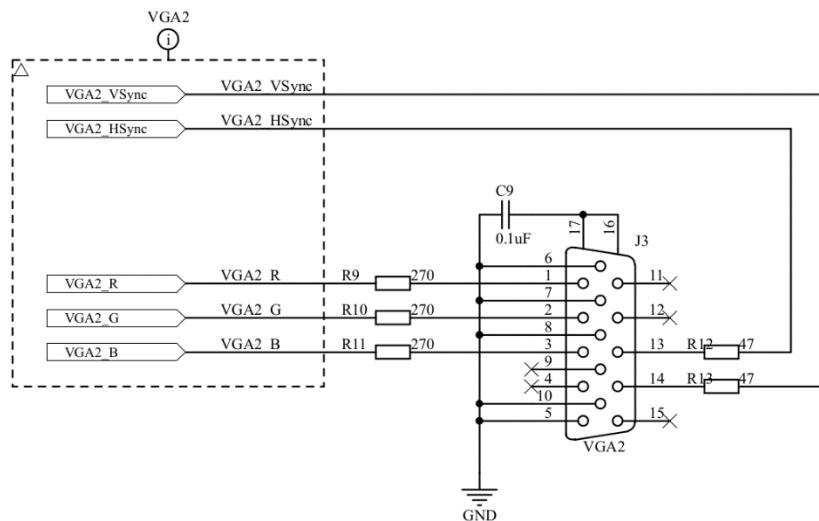
Dzięki stosunkowo dużej szybkości kontrolera wejścia/wyjścia w układzie *Altera Cyclone III*, możliwe jest używanie dostępnych wyprowadzeń, jako portów sygnałowych standardu SPI lub podłączenie do nich zewnętrznego modułu Ethernet.

W chwili obecnej do komunikacji w module SNF-0 dostępny jest port szeregowy, którego szybkość transmisji jest niewielka. Dzięki wykorzystaniu portów wejścia/wyjścia możliwe będzie w przyszłości rozbudowanie podsystemu komunikacji.

Wyprowadzono porty napięć dostępnych na płytce obwodu drukowanego modułu SNF-0, które można będzie wykorzystać do zasilania zewnętrznych podzespołów i modułów.

3. 8. Wyjście VGA2

Wyjście wideo VGA2 posiada podstawową funkcjonalność wyświetlania obrazu w 8 kolorach. Jego zaletą jest jednak łatwość w użyciu i możliwość zastosowania praktycznie dowolnej rozdzielczości (przykładowe rozdzielczości możliwe do uzyskania przedstawiono w tabeli 1). Wymagane jest, aby częstotliwość generowania pikseli była możliwa do uzyskania za pomocą układu FPGA modułu SNF-0. Port VGA2 posiada 3 jednobitowe sygnały kolorów oraz sygnały synchronizacji poziomej i pionowej.



Rys. 14. Schemat wyjścia VGA2

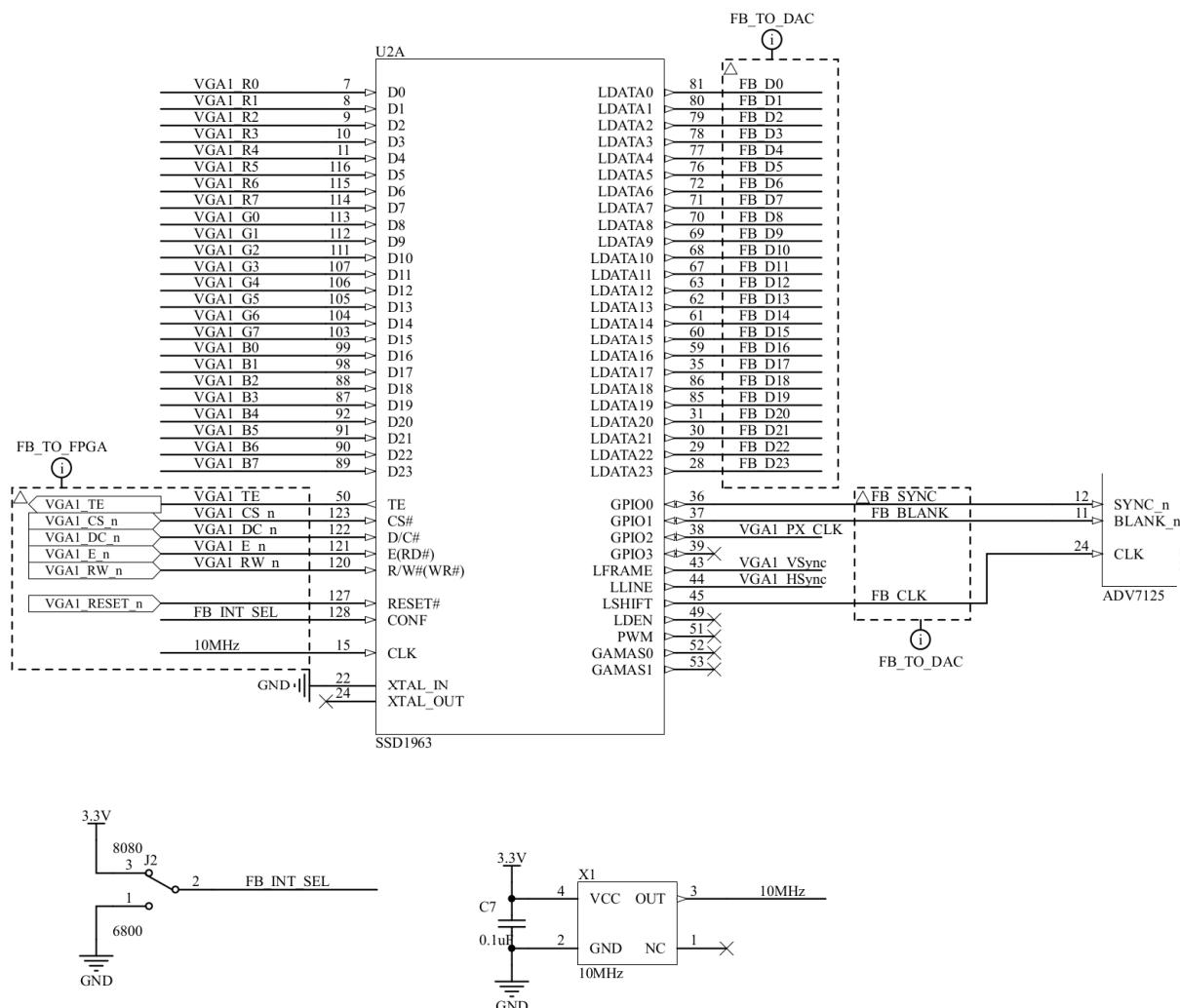
Na rysunku 14 zaprezentowano połączenie wyjścia VGA2 do układu FPGA.

rozdzielcość	częstotliwość odświeżania ekranu	częstotliwość zegara pikseli
640 x 480	60 Hz	25.175 MHz
640 x 480	75 Hz	31.5 MHz
800 x 600	60 Hz	40.0 MHz
800 x 600	75 Hz	49.5 MHz
1024 x 768	60 Hz	65.0 MHz
1024 x 768	75 Hz	78.8 MHz
1280 x 1024	60 Hz	108.0 MHz
1280 x 1024	75 Hz	135.0 MHz

Tab. 1. Przykładowe parametry obrazu możliwe do wyświetlenia przy pomocy złącza VGA2

3. 9. Wyjście VGA1, przetwornik cyfrowo analogowy i bufor obrazu

Wyjście VGA1 zostało wyposażone w dodatkowe układy, pozwalające na uzyskanie rozszerzonych funkcjonalności. Dzięki zastosowaniu bufora obrazu możliwe jest przesyłanie danych do wyświetlenia w wybranej kolejności lub też selektywne nadpisanie już obecnych wartości koloru. Pamięć obrazu pozwala także na aktualizowanie danych z dowolną częstotliwością, co umożliwia generowanie złożonej grafiki. Nie byłoby możliwe wyświetlenie obrazu o takiej ilości kolorów i rozdzielczości przy wykorzystaniu bezpośredniego podłączenia portu VGA do układu FPGA, która dostępna jest przy zastosowaniu zewnętrznego układu.



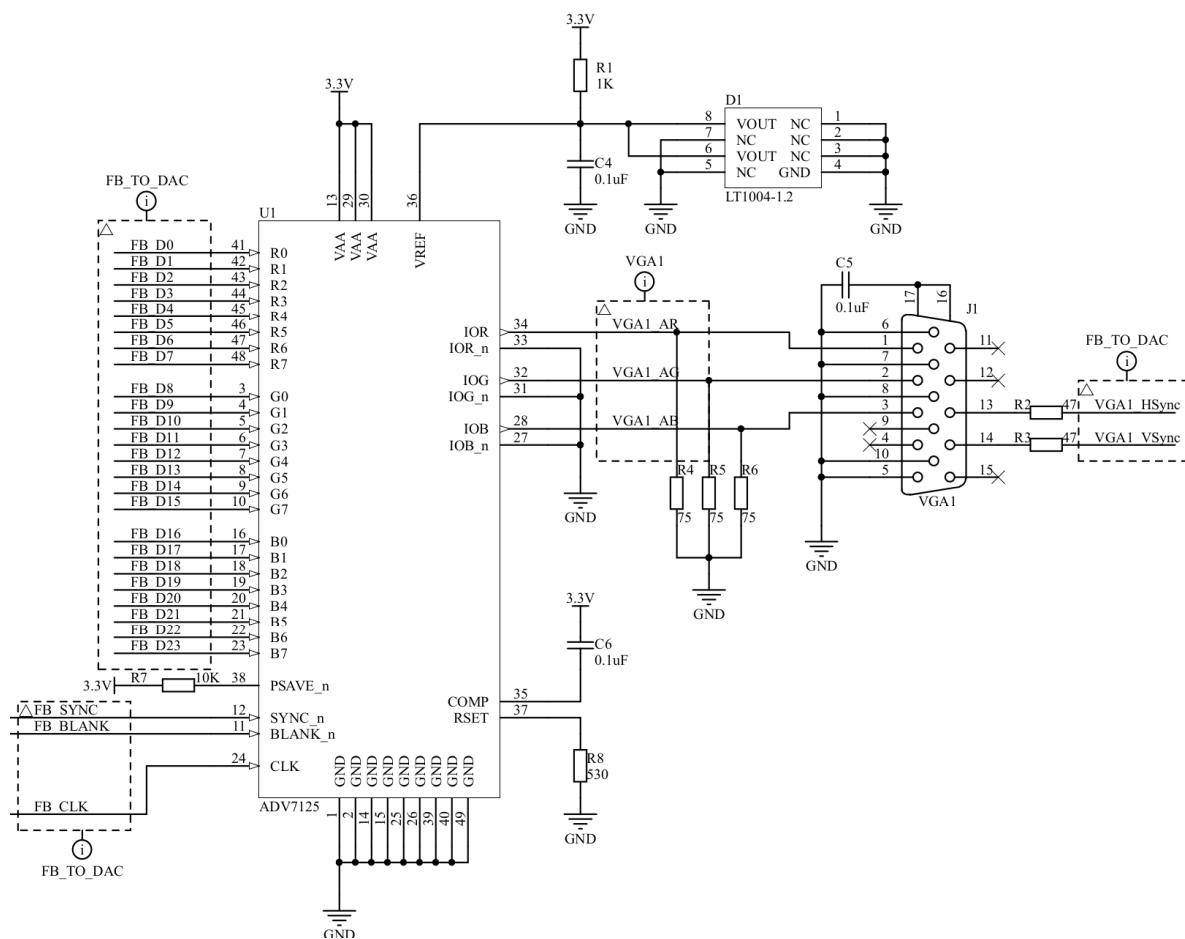
Rys. 15. Schemat połączeń układu bufora obrazu i jego otoczenia

Układ U2A (SSD1963 firmy Solomon Systech), przedstawiony na rysunku 15, jest buforem obrazu o pojemności 1215KB ze zintegrowanym kontrolerem wyświetlania. Dostępna pamięć pozwala na przechowanie obrazu o rozdzielczości do 864 x 480 pikseli przy 24-bitowej głębi koloru. Układ został wyposażony w równoległy interfejs komunikacyjny, wspierający 2 tryby (tryb 6800 i 8080) i różne szerokości magistrali danych (8, 9, 16 lub 24-bitów). Tryb interfejsu ustawiany jest za pomocą zworki J2. W otoczeniu układu kontrolera obrazu znajduje się oscylator kwarcowy (X1) o częstotliwości 10MHz, który stanowi dokładniejszą alternatywę do wbudowanego generatora sygnału zegarowego. SSD1963 wymaga dwóch różnych napięć: 1.2V -do zasilania logiki układu i 3.3V dla sygnałów wejścia/wyjścia.

Oprócz podstawowej funkcjonalności, omawiany element posiada kilka przydatnych funkcji pozwalających na manipulację wyświetlonym obrazem takich, jak: możliwość manipulowania jasnością, kontrastem i nasyceniem, obrót obrazu oraz odbicie lustrzane.

Układ SSD1963 został skonstruowany, jako kontroler obrazu dla wyświetlaczy LCD nieposiadających własnej pamięci RAM. Pomimo tego faktu, interfejs sterownika firmy *Solomon Systech* nadaje się do wykorzystania z wyjściem VGA. Sygnały komunikacyjne w wyświetlaczaach ciekłokrystalicznych, które wspiera układ SSD1963 są zgodne z tymi używanymi w standardzie VGA.

Dzięki wbudowanej pętli sprzężenia fazowego (*PLL*) możliwe jest ustawienie odpowiedniej częstotliwości zmiany kolorów kolejnych pikseli oraz parametrów licznika synchronizacji pionowej i poziomej. Konfiguracja układu odbywa się za pomocą dostępnych interfejsów poprzez wydawanie odpowiednich poleceń i ustawianie wymaganych wartości wbudowanych rejestrów.



Rys. 16. Schemat wyjścia VGA1 i przetwornika cyfrowo-analogowego

Kolejnym ważnym elementem podsystemu wyświetlania VGA1 jest układ 3-kanałowego przetwornika cyfrowo-analogowego U1 (ADV7125 firmy *Analog Devices*). Jest to specjalizowany konwerter przeznaczony do zastosowań związanych z wyświetlaniem obrazu o trzech składowych koloru.

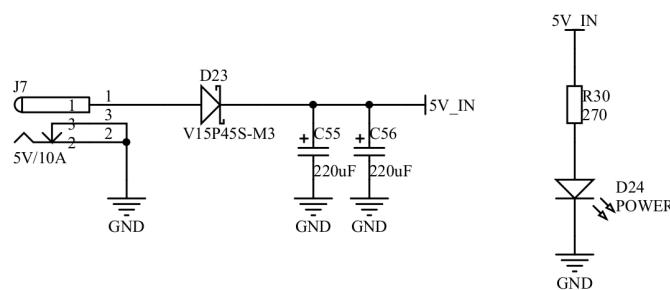
Dla każdego kanału dostępna jest 8-bitowa magistrala wejściowa, przez którą przesyłane są dane o kolorze. Na każdym z trzech wyjść analogowych, podpiętych do złącza VGA, pojawia się sygnał o odpowiednim poziomie napięcia, który steruje jasnością aktualnie wyświetlonego subpixela.

Wykorzystano zewnętrzny układ napięcia referencyjnego 1.235V, którego głównym elementem jest dioda Zenera D1 o dużej dokładności.

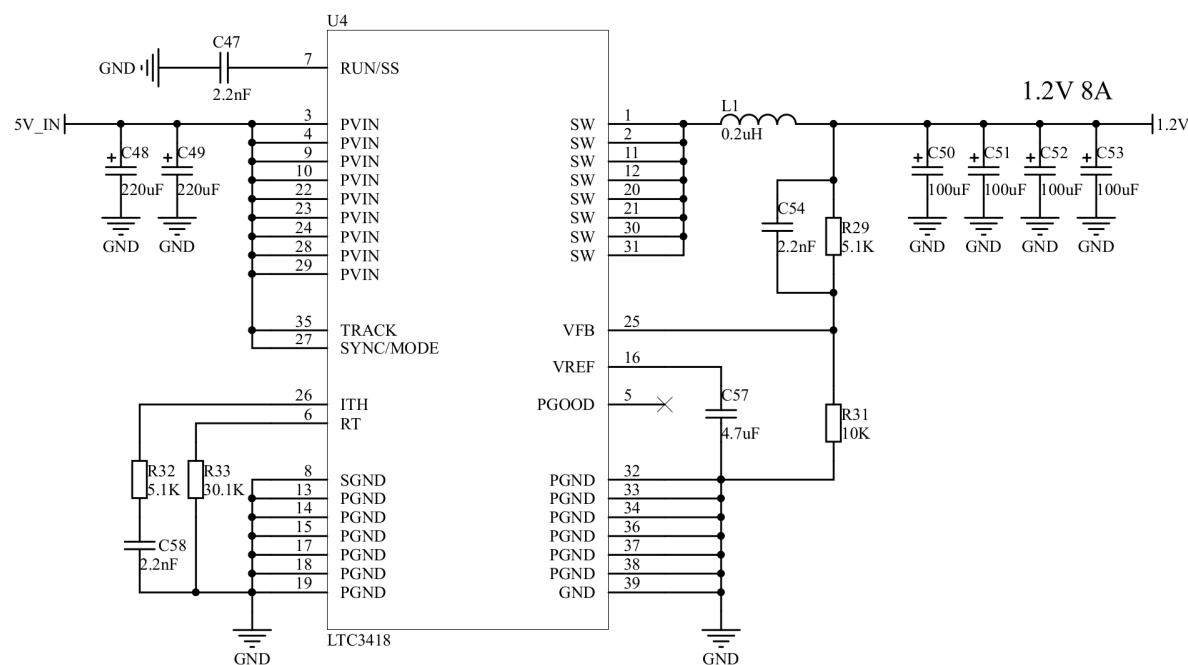
Podsystem wyświetlania obrazu ze złączem VGA1 pozwala na tworzenie projektów wyświetlających zaawansowaną grafikę w 24-bitowej głębi kolorów. Wadą tego rozwiązania jest jednak maksymalna rozdzielczość możliwa do uzyskania, wynosząca 640 x 480 pikseli. Schemat omawianego obwodu pokazano na rysunku 16.

3. 10. Moduł zasilający

Aby spełnić wymagania napięciowe i prądowe układów modułu SNF-0, zaprojektowano układ zasilania składający się z trzech stabilizatorów impulsowych, które obniżają napięcie wejściowe 5V do wartości użytecznych dla elementów modułu.



Rys. 17. Schemat złącza zasilania i elementów peryferyjnych (najczęściej złącza USB), która zabezpiecza przed podłączeniem napięcia o dodatkowym źródle, a także wykrywa obecność napięcia zasilania. Rysunek 17 pokazuje dwa typy złączy zasilania.

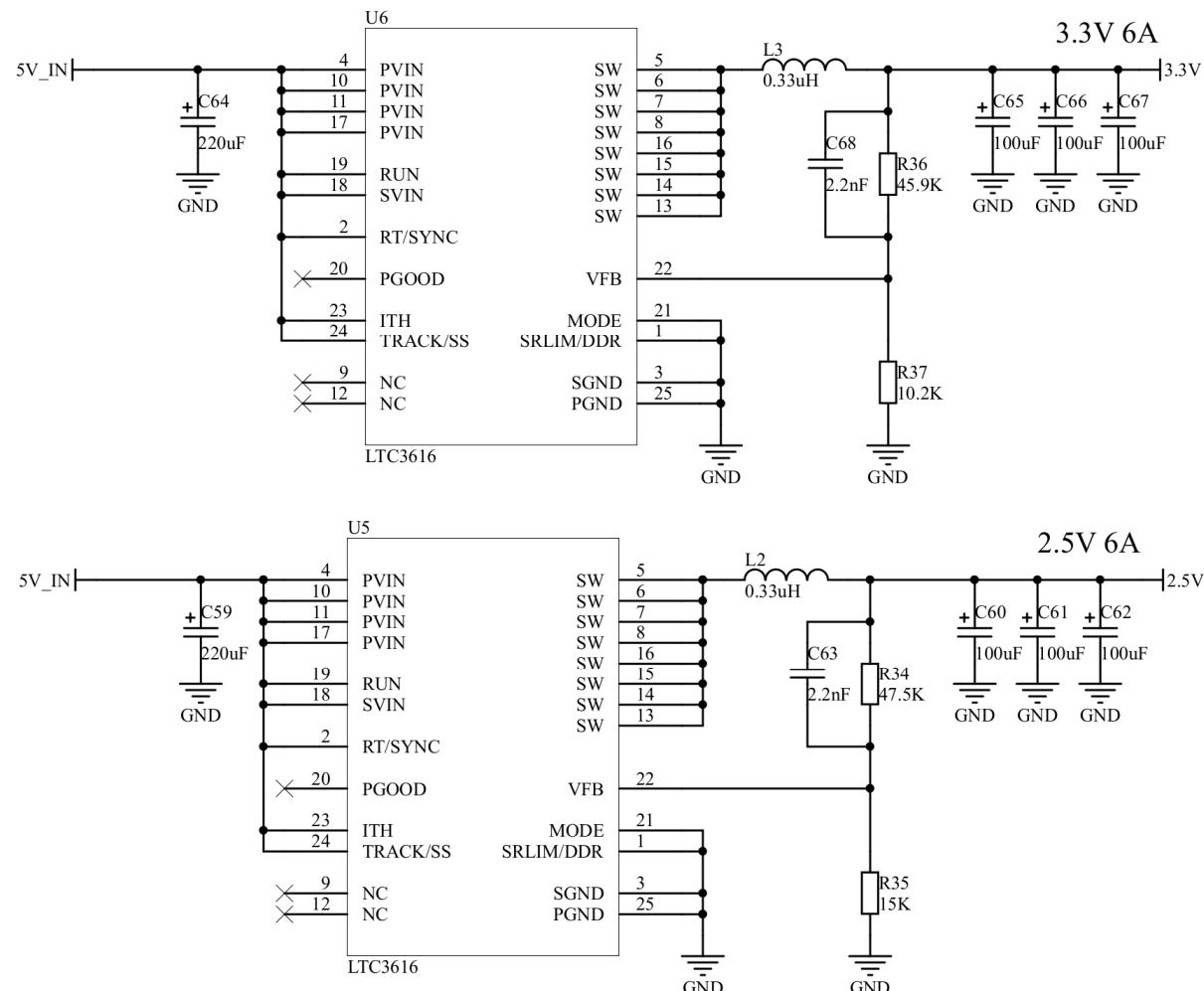


Rys. 18. Schemat stabilizatora impulsowego, obniżającego napięcie do 1.2V

Do zbudowania podsystemu zasilania wykorzystano stabilizatory impulsowe firmy *Linear Technology*, charakteryzujące się między innymi niskim spadkiem napięcia pod obciążeniem i dużą wydajnością prądową. Niebagatelnym warunkiem wyboru stabilizatorów impulsowych *LTC3616* i *LTC3418* była mała ilość elementów dyskretnych potrzebnych do ich uruchomienia i łatwa dostępność.

Układy stabilizujące napięcie użyte w projekcie posiadają wbudowane tranzystory kluczujące. Do ich poprawnego działania potrzebne są tylko kondensatory, cewka i rezystory konfigurujące wartość napięcia wyjściowego i częstotliwość pracy układu.

Rysunek 18 przedstawia schemat stabilizatora impulsowego obniżającego napięcie do wartości 1.2V, które używane jest przede wszystkim do zasilania logiki układu FPGA, a także, jako jedno z napięć bufora obrazu. Układ stabilizatora U4 jest w stanie zasilić odbiornik pobierający maksymalnie 8A. Rezystory R29 i R31 tworzą dzielnicę napięcia, który służy do konfiguracji napięcia wyjściowego. Opornik R33 ustawia częstotliwość pracy układu na ok. 2.24MHz.



Rys. 19. Schemat stabilizatorów impulsowych, obniżających napięcie do 2.5V i 3.3V

Na rysunku 19 przedstawiono stabilizatory impulsowe, które na wyjściu dają napięcie 3.3V oraz 2.5V. Pierwsza wartość napięcia wykorzystywana jest między innymi do zasilania pamięci SSRAM, kontrolera wejścia/wyjścia układu FPGA. Napięcie 2.5V jest używane w FPGA do zasilania układów PLL. W obu przypadkach wykorzystano te same układy, pozwalające na podłączenie obciążenia pobierającego prąd o natężeniu 6A. Wartości rezystorów R36 i R37 oraz R34 i R35 ustalają napięcie wyjściowe. Układy U5 i U6 pracują z domyślną częstotliwością.

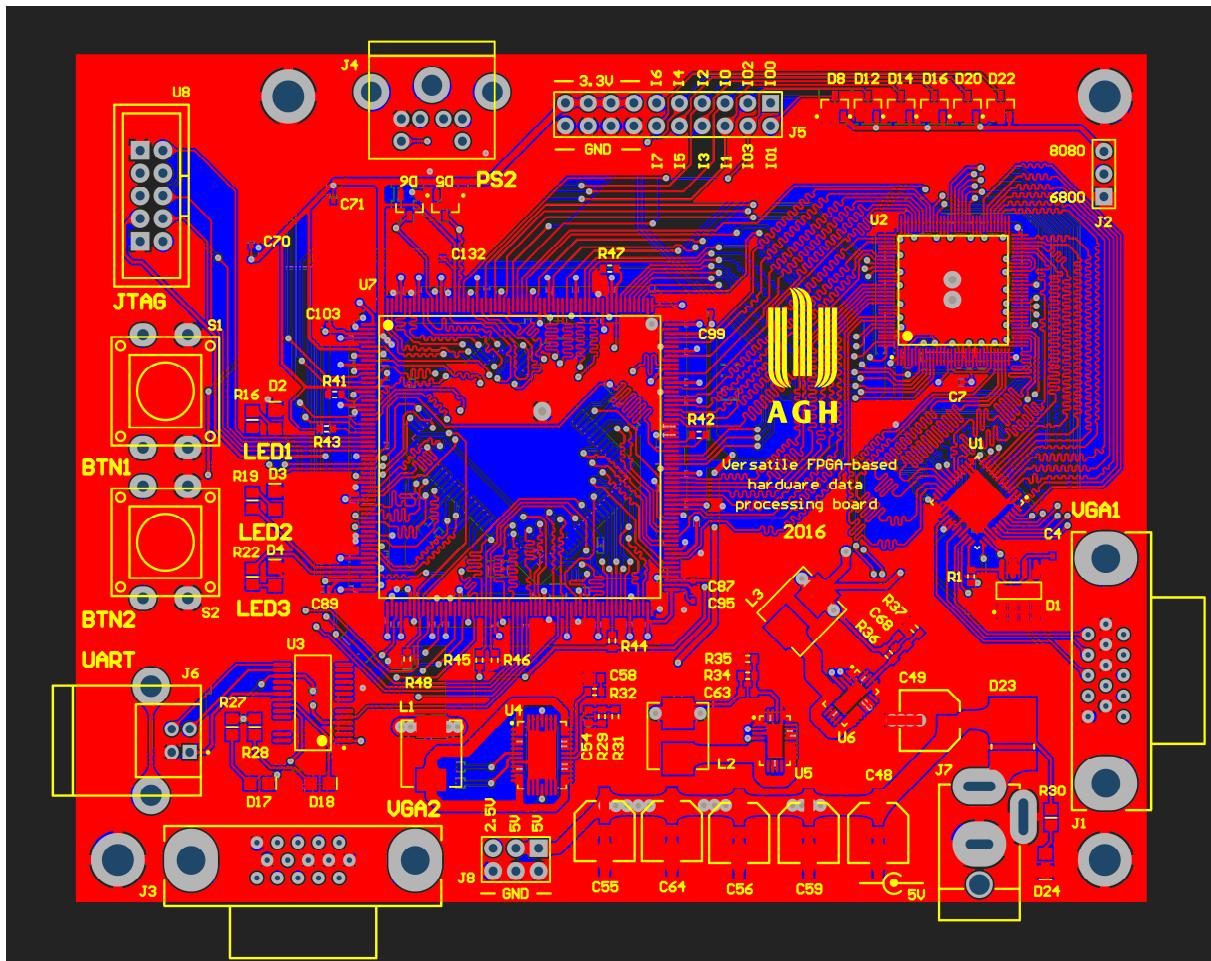
4. Projekt i wykonanie obwodu drukowanego modułu SNF-0

W poniższym rozdziale został opisany proces projektowania obwodu drukowanego modułu SNF-0. Przedstawiono szczegóły techniczne projektu i zastosowane rozwiązania. Dalsze podrozdziały prezentują proces montażu i uruchomienia modułu.

4. 1. Projekt obwodu drukowanego

Projekt obwodu drukowanego modułu SNF-0 został wykonany w module *PCB Editor* programu *Altium Designer*. W początkowej fazie prac przygotowano prototyp, który pozwolił na oszacowanie możliwości i wymagań odnośnie struktury i aspektów technicznych obwodu drukowanego. Więcej w tym temacie napisano w dokumentacji procesowej.

Porty wejścia/wyjścia, przyciski i kontrolki zostały ułożone na brzegach płytki, umożliwiając łatwy dostęp. W centralnej części PCB znajduje się układ FPGA, po prawej stronie rozbudowany moduł wyświetlania VGA1, natomiast u dołu umieszczono układ zasilania.



Rys. 20. Warstwy obwodu drukowanego modułu SNF-0 (czerwony – górna warstwa, niebieski – dolna)

Wykorzystano technologię *length tuning* dla dopasowania długości magistrali komunikacyjnych między układem FPGA i pamięcią oraz FPGA i buforem obrazu. Połączenia między układem SSD6319 i przetwornikiem cyfrowo-analogowym także zostały dopasowane pod względem długości. Pozwoli to na wyrównanie czasów opóźnień w przesyiale danych i uniknięcie związań z tym problemów przy wykorzystaniu wysokich częstotliwości. Starano się także, aby ważne połączenia były jak najkrótsze. Odpowiednio zwiększo grubość ścieżek układu zasilania, którymi może płynąć prąd o wysokim natężeniu. Zastosowano opisy na obu stronach płytki PCB, które ułatwiały montaż elementów i stanowią podstawowy opis dostępnych portów i kontrolek. W miejscach, w których nie ma ścieżek i elementów zastosowano wypełnienie obszarem miedzi podłączonym do masy, w celu redukcji szumów i ułatwienia dostępu masy do wymaganych miejsc. Gotowy projekt PCB (górną warstwą) przedstawiono na rysunku 20.

4. 2. Wykonanie płytki obwodu drukowanego i montaż elementów

Wykonanie płytki obwodu drukowanego zostało zlecone firmie zajmującej się tego typu usługami. Po skompletowaniu wszystkich elementów i odebraniu płytki, przystąpiono do jej montażu. Zaprojektowano i zamówiono także prostą obudowę ze szkła akrylowego, zabezpieczającą elementy przed prześcieradłami elektrostatycznymi i urazami mechanicznymi, która składa się z dwóch części montowanych za pomocą plastikowych wsporników i śrubek.

Montaż elementów wykonano przy użyciu stacji lutowniczej wyposażonej w standardową lutownicę grotową i dmuchawę na gorące powietrze. Zastosowanie urządzenia *hot air* między innymi ułatwiło montaż niektórych małych elementów (np. generatora kwarcowego).

Obwód drukowany ma wymiary 9,5 x 12 cm i został wykonany z laminatu o grubości 1,5 mm, pokrytego 35 mikrometrową warstwą miedzi. Zastosowano cynowanie metodą HASL (*hot air solder leveling*).

4. 3. Uruchomienie i testowanie modułu SNF-0

Testowanie modułu SNF-0 odbywało się bezpośrednio po montażu każdego podsystemu. Najpierw przygotowano układ zasilania i upewniono się, że napięcia przez niego wytwarzane są odpowiednio stabilne. Przetestowano spadki napięcia pod obciążeniem.

Następnie przylutowano generator kwarcowy i sprawdzono czy sygnał zegarowy na jego wyjściu spełnia wymagania. Kolejnym etapem było zmontowanie interfejsu USB do portu szeregowego i przetestowanie komunikacji z komputerem. Po przylutowaniu innych wymaganych elementów dyskretnych, przystąpiono do montażu układu FPGA.

Po zmontowaniu całego modułu, sprawdzono możliwość programowania układu *Cyclone III* i przygotowano prostą konfigurację testową. Następnie, po ustawieniu odpowiedniego stanu wyjść, sprawdzono oscyloskopem poziomy napięć na nich.

Po przejściu wstępnych testów, przygotowano konfiguracje testujące podstawowy port VGA2, działanie przycisków, diod, złącza PS2 i portu szeregowego.

Następnie przystąpiono do uruchamiania docelowego projektu specjalizowanego procesora graficznego wyświetlającego prostą grafikę 3D zbudowaną z linii. Jego opis znajduje się w kolejnym rozdziale.

5. Projekt i wykonanie przykładowego specjalizowanego procesora graficznego

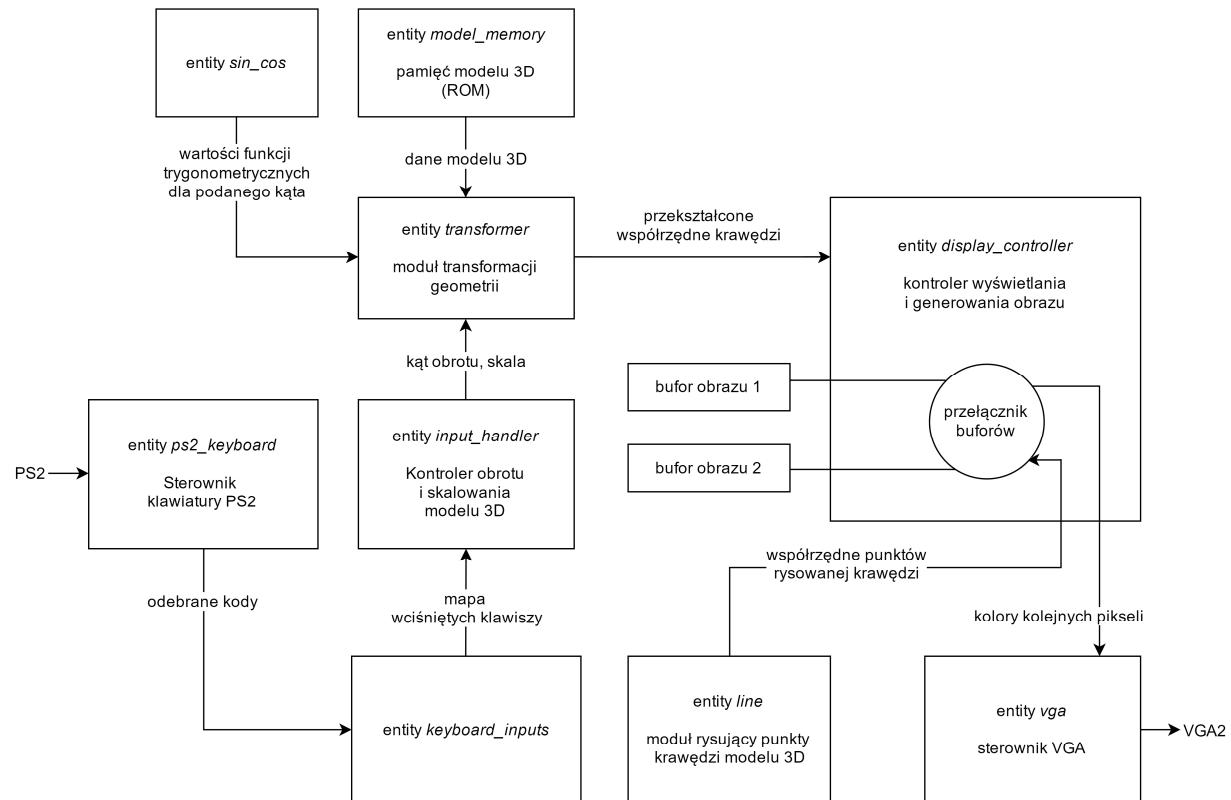
Główym celem projektu inżynierskiego było zbudowanie platformy umożliwiającej tworzenie procesorów zaprojektowanych w językach opisu sprzętu. Aby zaprezentować możliwości modułu SNF-0 wykonano przykładowy specjalizowany procesor graficzny służący do wyświetlania prostej grafiki 3D zbudowanej z linii. Został on napisany w języku VHDL.

Projekt wykorzystuje wyjście *VGA2* do wyświetlania jednokolorowego obrazu. Ograniczenie rozdzielczości i ilości barw związane jest z niewielką pamięcią wewnętrzną układu FPGA, która częściowo jest wykorzystywana przez projekt, jako bufory obrazu (zastosowano podwójne buforowanie).

Do kontroli przekształceń rysowanej bryły wykorzystywana jest klawiatura podłączona do portu PS2. Użytkownik może sterować obrotem oraz skalowaniem obiektu 3D.

Procesor graficzny posiada konfigurację w postaci geometrii bryły zapisaną na stałe w jego kodzie, która może jednak być w łatwy sposób podmieniana poprzez dostarczenie danych w odpowiednim formacie i ponowną komplikację projektu.

Napisano skrypt eksportu modelu 3D z programu *3DS Max*, który generuje plik **.vhd*, zawierający odpowiednią konfigurację procesora graficznego.

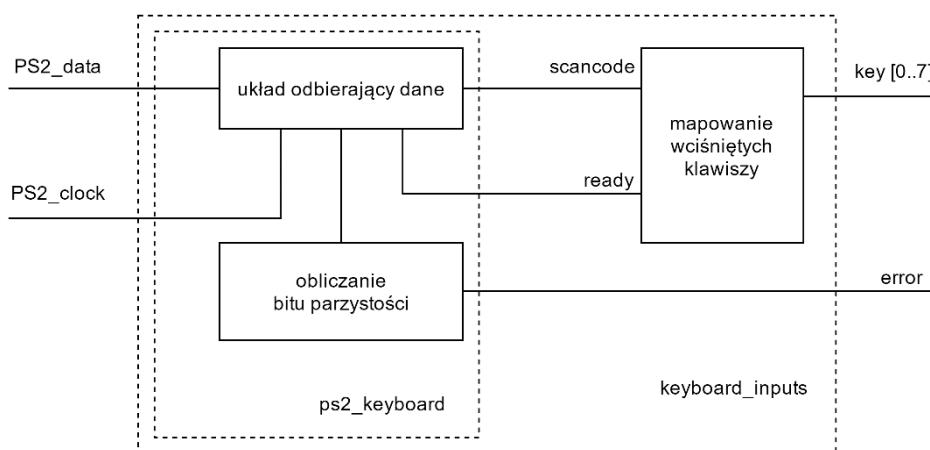


Rys. 21. Schemat blokowy specjalizowanego procesora graficznego

Na rysunku 21 pokazano schemat blokowy procesora graficznego. Poniżej przedstawiono szczegółowy opis jego poszczególnych jednostek.

5. 1. Kontroler klawiatury PS2

PS2 jest interfejsem pozwalającym na podłączenie klawiatury lub myszki do komputera, posiadającego 6-pinowe złącze Mini-DIN. Urządzenie wyposażone w gniazdo PS2 musi dostarczyć klawiaturze lub myszce napięcie zasilające o wartości 5V. Komunikacja przebiega przy użyciu 2 sygnałów: zegarowego i linii danych. Sygnał zegarowy portu PS2 ma częstotliwość od 10 do 16.7 KHz. Przesył danych poprzedzany jest bitem startowym (stan niski). Następnie transmitowany jest jeden bajt danych, bit parzystości i bit końcowy. Po skończeniu przesyłania oba sygnały wracają do stanu wysokiego. Odebrane dane reprezentują część tzw. *scan code*. Kod naciśnięcia klawisza (*make code*) zazwyczaj składa się z jednego lub dwóch bajtów. *Break code* sygnalizuje zwolnienie przycisku klawiatury, który oprócz informacji o kodzie klawisza, zawiera dodatkowy bajt o wartości *F0*.



Rys. 22. Schemat blokowy kontrolera klawiatury PS2

Kontroler klawiatury (przedstawiony na rysunku 22) składa się z dwóch jednostek: *ps2_keyboard* i *keyboard_inputs*. Do sterownika podłączone są sygnały portu PS2, które synchronizowane są przy użyciu rejestrów i filtrowane. Odebrane dane zapisywane są do rejestru przesuwnego, taktowanego zegarem *PS2_clock*. Układ odbierający dane posiada także licznik, który odmierza ilość cykli do zakończenia pobierania bajtu. Kiedy transmisja zostanie wykonana, ustawiana jest flaga *ready*. Do sprawdzenia poprawności odebranych danych wykorzystywany jest bit parzystości.

Druga część jednostki kontrolera klawiatury *keyboard_inputs* odpowiedzialna jest za zmapowanie odebranych bajtów na flagi określające wciśnięcie klawisza. Każdy bit wyjścia *key* zawiera informację, czy wciśnięto klawisz odpowiadający jego indeksowi.

5. 2. Moduł kontroli skalowania i obrotu

Jednostka modułu kontroli skalowania i obrotu (*input_handler*) wykorzystywana jest do kontroli przekształceń wyświetlanej bryły 3D. Na podstawie tablicy wciśniętych klawiszy, przekazanej z modułu *keyboard_inputs*, zmieniany jest kąt obrotu wokół każdej z osi oraz skalowanie wyświetlanego obiektu.

Moduł posiada licznik, inkrementowany, gdy ramka obrazu VGA zaczyna być rysowana, który powoduje opóźnienie aktualizacji przekształceń modelu 3D. Wartości kątów obrotu i skali przekazywane są do jednostki transformacji geometrycznych.

5. 3. Moduł pamięci i struktura danych modelu 3D

Do przechowywania danych geometrycznych trójwymiarowego modelu, który ma zostać wyświetlony na specjalizowanym procesorze graficznym, wykorzystano bloki pamięci układu FPGA, skonfigurowane do pracy w trybie ROM. Jednostka projektowa *model_memory* wykorzystywana jest do synchronicznego odczytu danych. W pakiecie *model* znajduje się wyeksportowana struktura bryły 3D.

Geometria, przechowywana w blokach ROM, reprezentowana jest przez tablicę 96-bitowych wektorów. Każdy element składa się z trójwymiarowych współrzędnych końców krawędzi modelu. Składowe współrzędne reprezentowane są przez 16-bitowe liczby całkowite.

Wszystkie dalsze obliczenia dostosowane są do liczb z przedziału -8191 do 8191. Jest to wystarczająca dokładność, aby uzyskać zadowalające efekty przy dobrym wykorzystaniu dostępnych zasobów pamięciowych i logicznych. Wydobycie składowych z elementu tablicy ma miejsce w module przekształceń geometrycznych.

Każdy model 3D, przed użyciem w opisywanym projekcie, musi zostać wyeksportowany z programu *Autodesk 3DS Max* za pomocą przygotowanego skryptu. Eksporter ten tworzy plik **.vhd*, który zawiera dane modelu. Więcej na ten temat w podrozdziale 5. 11.

5. 4. Moduł przekształceń geometrycznych

Wstęp

Jednostka projektowa wykonująca transformacje geometryczne jest najważniejszym modelem projektu. Jej funkcją jest obliczanie przekształceń obrotu i skalowania modelu 3D, a także rzutowanie trójwymiarowych współrzędnych na płaszczyznę wyświetlaną na monitorze przy pomocy złącza VGA.

Na wejściu jednostki podawane są kąty obrotu wokół każdej z osi oraz współczynnik skalowania bryły, a także sygnały sterujące z modułu kontrolera wyświetlania. Jednostka przekształceń geometrycznych posiadainstancję pamięci modelu, przechowującą dane określające współrzędne krawędzi tworzących model 3D. Na wyjściu pojawiają się współrzędne x_0, y_0, x_1, y_1 kolejnych przekształconych krawędzi. Cały proces jest synchronizowany przez kontroler wyświetlania.

Jednostka dostarczająca wartości funkcji trygonometrycznych

Do obliczeń trygonometrycznych potrzebne są wartości funkcji sinus i cosinus kątów obrotu bryły. Dostarczane są one przez jednostkę *sin_cos*. Podstawowym elementem tego modułu jest tablica, zawierająca wartości funkcji sinus dla całkowitych kątów od 0° do 90° . Dokładność ta okazała się dostarczać zadowalających efektów wizualnych obrotu bryły.

Poniżej przedstawiono sposób wyznaczania wartości funkcji trygonometrycznych sinus i cosinus dla kątów w zakresie 0° do 360° . Tablica *sin_lut* zawiera 90 obliczonych wcześniej wartości. Do zmiennych wyjściowych *sin_out* i *cos_out* przypisywana jest wartość funkcji dla danej wejściowej *angle*.

```

if angle ≥ 270° then
    sin_out ← -sin_lut[360° - angle]
    cos_out ← sin_lut[angle - 270°]
elseif angle ≥ 180 then
    sin_out ← -sin_lut[angle - 180°]
    cos_out ← -sin_lut[270° - angle]
elseif angle ≥ 90° then
    sin_out ← sin_lut[180° - angle]
    cos_out ← -sin_lut[angle - 90°]
else
    sin_out ← sin_lut[angle]
    cos_out ← sin_lut[90° - angle]
end if

```

Działanie modułu transformacji

Podstawowym elementem jednostki transformacji jest rozbudowana maszyna stanów, która steruje wykonaniem całego przekształcenia. W początkowej fazie przygotowywane są wartości funkcji trygonometrycznych dla poszczególnych kątów. Następnym krokiem jest dokonanie obliczeń potrzebnych do wykonania obrotu i skalowania. Końcowym etapem jest oczekiwanie, aż linia, o końcach w przekształconych współrzędnych, zostanie narysowana. Powyższe operacje powtarzane są, aż cała scena zostanie przygotowana do wyświetlenia. Szczegółowy opis kolejnych kroków maszyny stanów opisano w poniżej tabeli.

Nazwa stanu	Następny stan	Opis
<i>request_sincos_x</i>	<i>get_sincos_x</i>	Przekazanie kąta obrotu wokół osi x do modułu <i>sin_cos</i>
<i>get_sincos_x</i>	<i>request_sincos_y</i>	Zapisanie wartość funkcji sinus i cosinus dla wcześniejszej podanego kąta
<i>request_sincos_y</i>	<i>get_sincos_y</i>	Przekazanie kąta obrotu wokół osi y do modułu <i>sin_cos</i>
<i>get_sincos_y</i>	<i>request_sincos_z</i>	Zapisanie wartość funkcji sinus i cosinus dla wcześniejszej podanego kąta
<i>request_sincos_z</i>	<i>get_sincos_z</i>	Przekazanie kąta obrotu wokół osi z do modułu <i>sin_cos</i>

<i>get_sincos_z</i>	<i>cast_edge_to_32bit</i>	Zapisanie wartość funkcji sinus i cosinus dla wcześniej podanego kąta
<i>cast_edge_to_32bit</i>	<i>apply_x_rot</i>	Skopiowanie danych aktualnej krawędzi do tymczasowej struktury do obliczeń o 32-bitowych wartościach
<i>apply_x_rot</i>	<i>apply_y_rot</i>	Wykonanie obliczeń skutkujących obrótom obiektu wokół osi x
<i>apply_y_rot</i>	<i>apply_z_rot</i>	Wykonanie obliczeń skutkujących obrótom obiektu wokół osi y
<i>apply_z_rot</i>	<i>apply_scale</i>	Wykonanie obliczeń skutkujących obrótom obiektu wokół osi z
<i>apply_scale</i>	<i>inc_line_cnt</i>	Przeskalowanie współrzędnych krawędzi przez podany współczynnik
<i>inc_line_cnt</i>	<i>edge_draw_wait</i> – jeżeli pozostały krawędzie do wyświetlenia <i>next_wait</i> – jeżeli wszystkie krawędzie są narysowane	Zwiększenie licznika krawędzi i ustawienie flagi <i>edge_ready</i> – gdy pozostały jeszcze krawędzie do przekształcenia, wyzerowanie licznika i ustawienie bitu wyjściowego <i>rendered</i> w przeciwnym wypadku
<i>edge_draw_wait</i>	<i>cast_edge_to_32bit</i>	Oczekiwanie na pojawienie się zlecenia przekształcenia kolejnej krawędzi od kontrolera wyświetlania (po skończeniu jej rysowania). Obliczenia wartości funkcji sinus i cosinus dla wymaganych kątów nie zmieniają się, transformowana jest tylko krawędź, wskażywana przez licznik <i>inc_line_cnt</i> .
<i>next_wait</i>	<i>request_sincos_x</i> – jeżeli ustalono flagę <i>update_rot_request</i> <i>cast_edge_to_32bit</i> – w przeciwnym wypadku	Oczekiwanie na sygnał rozpoczęcia kolejnej transformacji krawędzi modelu 3D. Jeżeli ustaliona jest flaga świadcząca o zmianie któregoś z kątów obrotu (<i>update_rot_request</i>), aktualizowane są wartości funkcji trygonometrycznych. W przeciwnym wypadku, rozpoczyna się przekształcanie kolejnych krawędzi.

Oprócz logiki sekwencyjnej implementującej wyżej opisaną maszynę stanów, zastosowano układy kombinacyjne.

Pierwszy z nich jest odpowiedzialny za wyodrębnienie poszczególnych współrzędnych obu końców krawędzi z 96-bitowej wartości przechowywanej w pamięci modelu. Zostaje ona podzielona na 12 szesnastobitowych liczb.

Drugi układ konwertuje przekształcone współrzędne do przestrzeni 2D i dokonuje odpowiedniego skalowania według algorytmu przedstawionego poniżej. Wartości $x0$, $y0$, $x1$, $y1$ (w przedziale [-8191, 8191]) oznaczają współrzędne końców linii do narysowania, natomiast struktura *edge* zawiera koординaty przekształconej krawędzi w 3D.

```
x0 ← edge.z0 / 64 + 640 / 2  
y0 ← edge.y0 / 64 + 480 / 2  
x1 ← edge.z1 / 64 + 640 / 2  
y1 ← edge.y1 / 64 + 480 / 2
```

Przekształcenia geometryczne

Omawiany procesor graficzny wykonuje dwa rodzaje przekształceń bryły 3D: rotacje wokół osi x, y i z oraz skalowanie.

Dla uproszczenia projektu i optymalizacji liczby wymaganych zasobów układu FPGA, wszystkie transformacje wykonywane są na szesnastobitowych liczbach całkowitych, wykorzystując operacje mnożenia i dodawania. Na każdym etapie wykonywania przekształceń odrzucane są najmniej znaczące bity wyniku poprzez operację przesunięcia arytmetycznego w prawo. Początkowa bryła posiada wierzchołki odpowiednio przeskalowane na etapie eksportu tak, aby znajdowały się w przedziale -8191 do 8191. Dobrały wartości przesunięć na każdym etapie obliczeń w taki sposób, aby końcowy wynik również był z takiego przedziału.

Podstawą wykorzystanego algorytmu skalowania i obrotu były operacje macierzowe (przedstawione na rysunku 23). Wektor współrzędnych wierzchołka jest mnożony przez kolejne macierze transformacji. Przekształcono działania tak, aby wystarczyło obliczenie sumy iloczynów. Rozdzielono obliczanie poszczególnych transformacji w celu zmniejszenia opóźnień w układzie logicznym.

(a) $R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & \sin \theta & \cos \theta \end{bmatrix}$ $R_y(\theta) = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix}$ $R_z(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta & 0 \\ \sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	(b) $S(s_x, s_y, s_z) = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & s_z \end{bmatrix}$ (c) $V_{out} = R_x R_y R_z S \begin{bmatrix} x \\ y \\ z \end{bmatrix}$
--	--

Rys. 23. (a) Macierze obrotu dla poszczególnych współrzędnych, (b) macierz skalowania, (c) złożenie przekształceń i obliczenie wektora wynikowego

5. 5. Kontroler wyświetlania

Jednostka kontrolera wyświetlania jest kolejnym ważnym modelem specjalizowanego procesora graficznego. Jej głównym zadaniem jest zarządzanie przebiegiem transformacji wejściowych krawędzi, rysowania linii i zapisem do odpowiedniego bufora obrazu.

Zastosowano technikę podwójnego buforowania polegającą na tym, że scena renderowana jest do jednego z nich, podczas gdy z drugiego pobierane są kolejne piksele do wyświetlenia na ekranie. Po skończeniu procesu, bufory zostają zamienione. Dzięki zastosowaniu tej metody, uniknięto efektu migotania rysowanej sceny i zwiększo czas przeznaczony na wygenerowanie obrazu.

Jako pamięć obrazu, wykorzystano układy 2-PORT RAM z biblioteki *Megafunctions*, dostępnej w oprogramowaniu *Altera Quartus II*. Dzięki możliwości użycia osobnego zegara do odczytu i zapisu, możliwe było zsynchronizowanie wyświetlania obrazu przez złącze VGA, przy jednoczesnym zachowaniu wyższego taktowania generowania sceny (zapis: 50MHz, odczyt: 25MHz).

Przechowywany obraz ma 1-bitową głębię koloru. Zastosowanie 2 buforów zajęło ok. 50% jednostek pamięci M9K dostępnych w układzie FPGA. Do przełączania buforów stworzono odpowiednią logikę zsynchronizowaną ze sterownikiem VGA. Przełączania pamięci do zapisu i wyświetlania następuje po skończeniu wyświetlania obrazu.

Proces odczytu danych obrazu oblicza adres pamięci RAM, na podstawie współrzędnych aktualnie wyświetlonego piksela, dostarczonych przez moduł kontrolera VGA. Dane koloru piksela pobierane są z bufora i przekazywane do kontrolera VGA. Po skończeniu przesyłania danych koloru, następuje zamiana buforów.

Proces zapisujący do pamięci RAM jest nieco bardziej skomplikowany. Zbudowany został, jako maszyna stanów. Jej opis znajduje się w poniższej tabeli.

Nazwa stanu	Następny stan	Opis
<i>start_draw</i>	<i>clear</i>	Aktualny adres bufora zapisu ustawiany jest na 0, w stan wysoki przechodzi flaga <i>start</i> , która informuje o rozpoczęciu generowania sceny jednostce transformującej geometrię, która przygotowuje pierwszą krawędź.
<i>Clear</i>	<i>start_transform</i> – po skończeniu czyszczenia bufora	Następuje przypisanie koloru tła do wszystkich pikseli bufora zapisu.
<i>start_transform</i>	<i>transform_wait</i> – jeżeli nie przekształcono jeszcze wszystkich krawędzi <i>frame_rendered</i> – w przeciwnym przypadku	Ustawienie flagi zlecającej rozpoczęcie transformacji kolejnej krawędzi, jeżeli jeszcze nie wszystkie zostały przekształcone.
<i>transform_wait</i>	<i>load</i> – jeżeli scena nie została wyrenderowana i krawędź dostępna jest przekształcona krawędź <i>frame_rendered</i> – jeżeli scena została wyrenderowana	Oczekiwanie na przekształcenie krawędzi przez moduł <i>transformer</i> .
<i>Load</i>	<i>init_draw</i>	Skopiowanie przekształconych współrzędnych do rejestrów generatora linii.
<i>init_draw</i>	<i>draw_edge</i>	Rozpoczęcie rysowania linii
<i>draw_edge</i>	<i>start_transform</i> – jeżeli cała linia została wygenerowana	Pobieranie kolejnych współrzędnych pikseli z generatora linii i ich zapis w pamięci obrazu.
<i>frame_rendered</i>	<i>start_draw</i> – po zamianie buforów	Oczekiwanie na zlecenie zamiany buforów od procesu wyświetlającego obraz na monitorze VGA i wykonanie tej akcji.

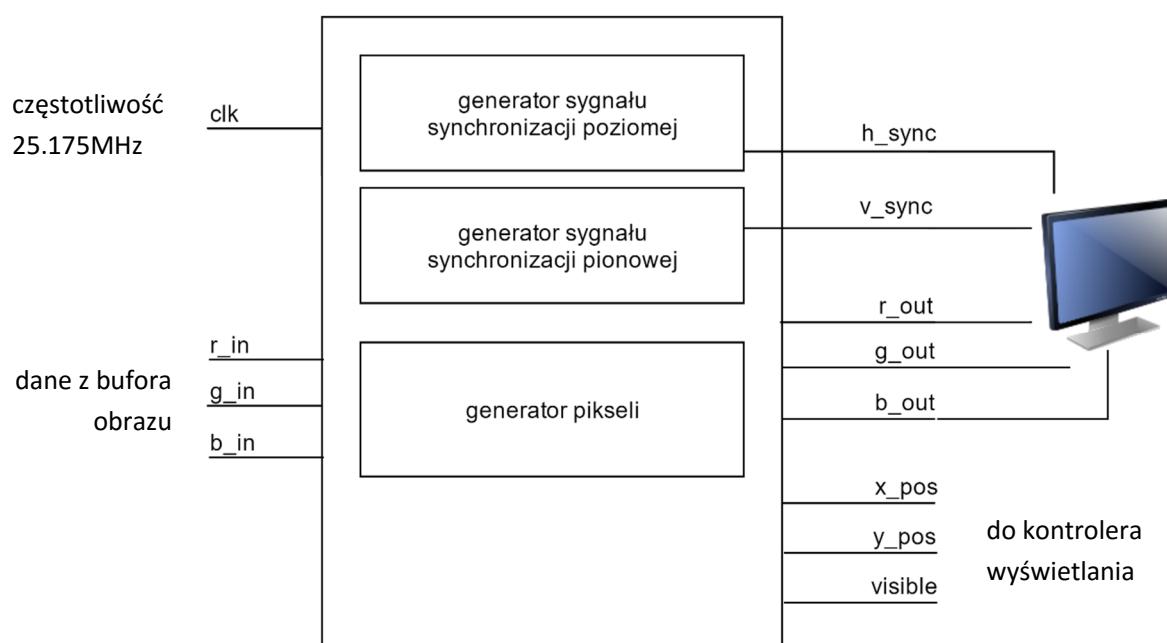
5. 6. Generator odcinków

Do generowania odcinków (krawędzi bryły 3D) zastosowano algorytm Bresenhama. Po zleceniu rysowania linii, moduł generuje współrzędne kolejnych pikseli i ustawia flagę rysowania. Do synchronizacji zastosowano prostą maszynę stanów. Sterowanie jednostką generatora odbywa się z poziomu kontrolera wyświetlania. Obliczanie współrzędnych pikseli zrealizowano przy pomocy logiki kombinacyjnej.

5. 7. Kontroler VGA

Jednostka projektowa kontrolera VGA na wyjściu, oprócz wartości kolorów, generuje sygnały synchronizacji poziomej i pionowej. Dostępne są także rejesty *x_pos* i *y_pos*, które zawierają współrzędne aktualnie wyświetlonego piksela. Jeżeli generowany obraz jest aktualnie widoczny flaga *visible* jest ustawiana. Na wejściu kontrolera podawany jest sygnał zegarowy oraz wartość koloru piksela.

Najważniejszym elementem jednostki projektowej *vga* są generatory sygnałów *h_sync* i *v_sync* sterowane licznikami taktowanymi zegarem *clk* o częstotliwości 25.175 MHz. Podłączenie modułu przedstawiono na rysunku 24.



Rys. 24. Jednostka kontrolera VGA

5. 8. Eksport pliku z programu 3DS Max do pamięci układu FPGA

Program eksportujący model 3D z programu *Autodesk 3DS Max* został napisany w języku skryptowym tego środowiska (*MaxScript*). Język ten pozwala na łatwy dostęp do elementów sceny wczytanych do oprogramowania firmy *Autodesk*, a także umożliwia operację na plikach. Wykorzystanie *MaxScript* pozwoliło w łatwy sposób przygotować model 3D do zapisania w pamięci układu FPGA.

Po załadowaniu skryptu eksportu, pojawia się przycisk z napisem *Export scene objects*, po którego naciśnięciu wykonywany jest zapis sceny do pliku **.vhd*.

Po uruchomieniu, skrypt wyświetla dialog wyboru ścieżki do zapisu. Następnie wykonywane są następujące operacje dla obiektów sceny: przeskalowanie bryły tak, aby współrzędne jej wierzchołków mieściły się w przedziale od -1 do 1, konwersja do trybu *EditPoly* i zapis danych do pliku.

W pliku **.vhd* tworzona jest struktura pakietu (*package*) języka VHDL, w której znajdują się dwie stałe: przechowująca ilość krawędzi oraz tablica z danymi. Podczas zapisu, skrypt eksportera pobiera współrzędne wszystkich krawędzi obiektów sceny, skaluje je przez 8191 i tworzy z nich łańcuch znaków złożony z 12 liczb szesnastobitowych. Wartości zapisane są w postaci szesnastkowej.

Tak przygotowany plik **.vhd* może być w łatwy sposób użyty w projekcie specjalizowanego procesora graficznego.

6. Spis rysunków

Rys. 1. Schemat koncepcyjny jednostki logicznej opartej o tablicę LUT	4
Rys. 2. Struktura układu FPGA rodziny Cyclone II i opis podstawowych elementów	5
Rys. 3. Element logiczny układu Cyclone II w trybie normalnym	6
Rys. 4. Element logiczny układu Cyclone II w trybie arytmetycznym.....	7
Rys. 5. Układ FPGA Cyclone III na płytce modułu SNF-0	8
Rys. 6. Podział i połączenie dokumentów schematu modułu SNF-0.....	9
Rys. 7. Schemat podłączenia portów wejścia/wyjścia układu FPGA.....	10
Rys. 8. Schemat układu programowania i taktowania modułu SNF-0	11
Rys. 9. Schemat podłączenia pamięci SSRAM do układu FPGA.....	12
Rys. 10. Schemat interfejsu portu szeregowego	13
Rys. 11. Schemat podłączenia portu PS2	14
Rys. 12. Schemat podłączenia diod LED i przycisków do układu FPGA	14
Rys. 13. Schemat portów wejścia/wyjścia ogólnego przeznaczenia	15
Rys. 14. Schemat wyjścia VGA2.....	16
Rys. 15. Schemat połączeń układu bufora obrazu i jego otoczenia	17
Rys. 16. Schemat wyjścia VGA1 i przetwornika cyfrowo-analogowego.....	18
Rys. 17. Schemat złącza zasilania i elementów peryferyjnych	19
Rys. 18. Schemat stabilizatora impulsowego, obniżającego napięcie do 1.2V	19
Rys. 19. Schemat stabilizatorów impulsowych, obniżających napięcie do 2.5V i 3.3V	20
Rys. 20. Warstwy obwodu drukowanego modułu SNF-0 (czerwony – górna warstwa, niebieski – dolna) ..	21
Rys. 21. Schemat blokowy specjalizowanego procesora graficznego.....	23
Rys. 22. Schemat blokowy kontrolera klawiatury PS2	24
Rys. 23. (a) Macierze obrotu dla poszczególnych współrzędnych, (b) macierz skalowania, (c) złożenie przekształceń i obliczenie wektora wynikowego	29
Rys. 24. Jednostka kontrolera VGA	31