Technika cyfrowa

Analizator spektrum zrealizowany na układzie FPGA

Przy użyciu płytki DE10-Standard i układu Altera FFT IP Core

Karol Szustakowski

Informatyka Rok 2
AGH WIET
4 V 2021

Spis treści

1 Wstęp

Celem projektu jest zrealizowanie analizatora spektrum na układzie FPGA. W tym przypadku użyta została płytka Terasic DE10-Standard, wyposażona w układ Cyclone V.

1.1 Założenia projektu

Wykonany analizator powinien działać w następujący sposób:

- Sygnał dźwiękowy powinien być doprowadzony do układu poprzez wejście LINE-IN lub MIC-IN zestawu.
- Jednocześnie, sygnał wejściowy powinien być automatycznie przekierowany na wyjście SPEAKER czy LINE-OUT.
- Układ powinien pobrać z wbudowanego w płytkę kodeku Wolfson WM8731 informację na temat amplitudy dźwięku.
- Informacja ta następnie przekazywana jest do układu realizującego szybką transformatę Fouriera.
- Wynik transformaty jest odpowiednio analizowany, a następnie przekazywany do podukładu realizującego wyjście/wizualizację dźwięku.
- Podukład realizujący wyjście przygotowuje dane wejściowe do przesłania je do paska diód LED WS2812B.
- Każda kolejna dioda LED umieszczona na pasku oznacza kolejną, wyższą częstotliwość. Zapalenie się diody oznacza pojawienie się danej częstotliwości w sygnale. Samo wystąpienie częstotliwości i amplituda odpowiadającej jej fali odpowiedzialne są za intensywność świecenia diody.
- W zależności od częstości pojawiania się danej częstotliwości w sygnale, diody LED zmieniają kolor, od czerwonego dla rzadko pojawiających się częstotliwości, poprzez kolor zielony, do koloru niebieskiego dla często pojawiających się częstotliwości.

1.2 Wykorzystane zewnętrzne układy

W projekcie zostały wykorzystane następujące układy nieskonstruowane przeze mnie:

- Audio Controller for DE-series Boards - Intel FPGA University Program. Jest to układ będący niejako uchwytem do danych wystawianych przez umieszczony na płytce kodek Wolfson WM8731 pod kątem rozdzielczości danych, sposobu dostarczania informacji, czy też kierunku przesyłu danych.
- I2C bus for audio and video configuration on DE-series boards - Intel FPGA University Program. Układ ten realizuje konfigurację opisanego powyżej kodeku, pod kątem częstotliwości próbkowania, przesyłu danych na linii LINE-IN - LINE-OUT, oraz sposobu reprezentacji danych.

- Audio Clock for DE-series Boards - Intel FPGA University Program. Jest to układ PLL realizujący redukcję częstotliwości zegara wejściowego do częstotliwości, z którą pracuje kodek audio.
- FFT - Altera Corporation. Jest to układ realizujący szybką transformatę Fouriera na danych wejściowych (w tym przypadku danych wejściowych audio).
- ALTSQRT - Intel FPGA Integer Arithmetic IPCores User Guide. Jest to układ realizujący funkcję pierwiastka kwadratowego na danych wejściowych.

1.3 Sprzęt potrzebny do odtworzenia projektu

Do skonstruowania i odtworzenia projektu wymagane są następujące produkty:

- Płytka Terasic DE10-Standard.
- Kabel audio mini jack 3.5mm męsko-męski.
- Taśma LED WS2812B o długości 20 diód.
- Źródło dźwięku ze wzmacniaczem (końcowy układ jest oparty o wejście LINE-IN).
- Oprogramowanie Intel Quartus.

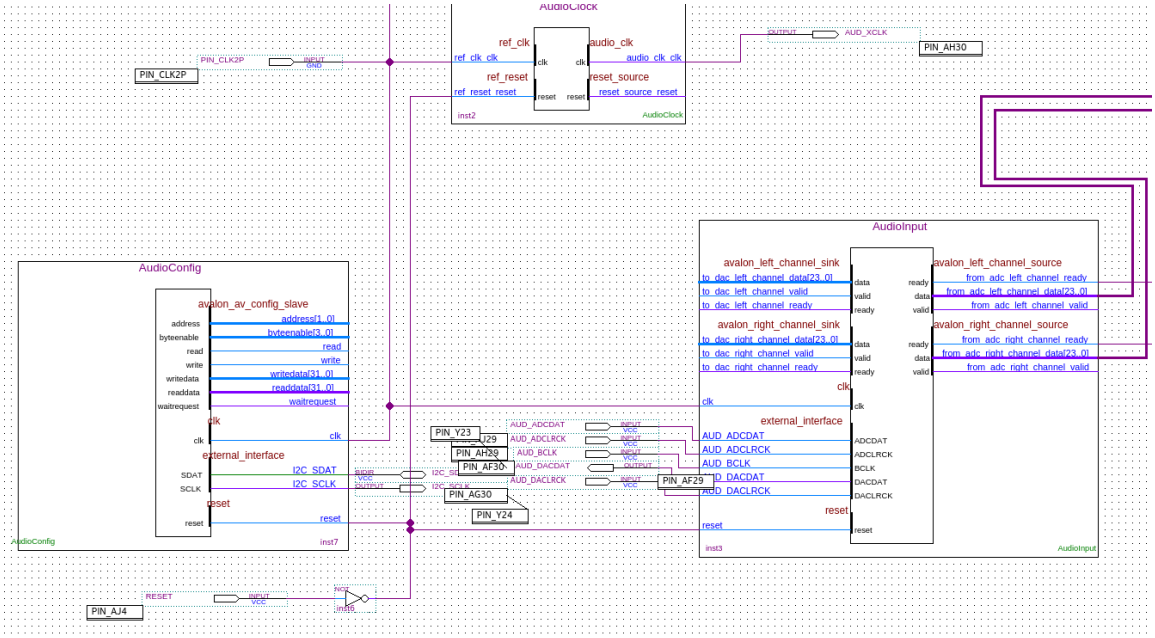
1.4 Dodatkowe oprogramowanie

W trakcie implementacji projektu, użyteczne są następujące podprogramy:

- Simulation Waveform Editor - przydatny do analizy danych wypisywanych przez sterownik diód i sprawdzania ich poprawności.
- SignalTap - przydatny do kontrolowania wartości wyjść FFT podczas dopasowywania pewnych stałych w programie, oraz do generalnego sprawdzania poprawności działania FFT.

2 Wejście audio

Ta sekcja poświęcona jest konfiguracji wejść audio do układu oraz konfiguracji tych wejść.



Rysunek 1: Fragment układu realizujący wejście audio

2.1 Konfiguracja kodeka

Kodek został skonfigurowany w następujący sposób:

- częstotliwość próbkowania - 96 kHz;
- dane 24-bitowe;
- automatyczne przesyłanie (bypass) wejścia LINE-IN do wyjścia LINE-OUT;
- używanie wejścia LINE-IN jako wejście, z którego pobierane będą dane.

W celu zapewnienia poprawnego działania układu konieczne jest podłączenie go do zegara (zegar ten jest taki sam jak główny zegar całego systemu - 50 MHz). Konieczne jest również podłączenie sygnału reset, który został podpięty poprzez bramkę logiczną NOT do pinu AJ4, odpowiadającemu jednemu z przycisków fizycznych na płytce.
Do zapewnienia prawidłowej funkcjonalności konieczne jest również połączenie wyjść I2C konfiguratora do odpowiednich pinów układu FPGA - wyjście I2C_SDAT odpowiada pinowi Y23 układu FPGA, a wyjście I2C_SCLK pinowi AG30.

2.2 Układ realizujący wejście

Układ realizujący wejście został skonfigurowany w następujący sposób:

- typ prezentacji danych - streaming;
- szerokość danych wyjściowych - 24-bitowe;
- kierunkowośc danych - in/out.

Typ danych streaming zapewnia, że dane pojawiające sie na wyjściu są kompletną próbką amplitudy w danej chwili czasu, a więc wykres czasowego przebiegu wartości wyjścia tego układu (który jest 24-bitowym wektorem reprezentującym liczbę ze znakiem), jest przebiegiem sygnału audio.
Typ streaming nie jest zastosowany bez powodu - użyta w projekcie transformata Fouriera wymaga podawania danych w kompletnych próbkach, więc jest to najwygodniejszy do użycia typ prezentacji uzyskanych danych, ponieważ może być bezpośrednio użyty jako wejście transformaty.

Tak samo, jak w przypadku konfiguratora, konieczne jest podłączenie sygnału zegarowego oraz wejścia reset. Oprócz tego, należy połączyć następujące wejścia/wyjścia układu z następującymi pinami układu FPGA:

- wejście układu - AUD_ADCDAT - AJ29
- wejście układu - AUD_ADCLRCK - AH29
- wejście układu - AUD_BCLK - AF30
- wejście układu - AUD_DACLRCK - AG30
- wyjście układu - AUD_DACDAT - AF29

Powyższe wyprowadzenia wynikają z dokumentacji płytki DE10 Standard.

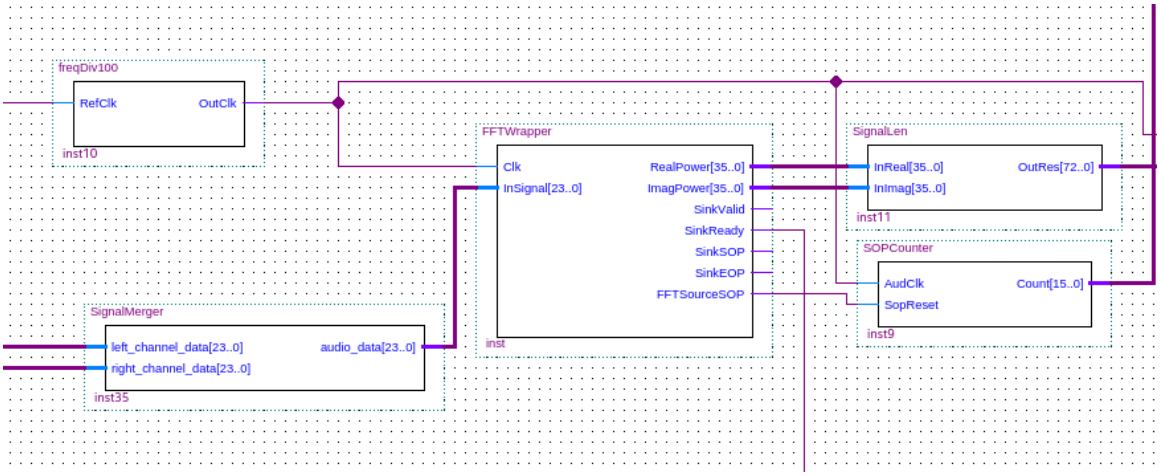
2.3 Zegar audio

Zegar audio jest układem PIL, na wyjściu wytwarzającym sygnał zegarowy o częstotliwości 12.8 MHz, jest to częstotliwość wymagana do prawidłowego funkcjonowana kodeku i powinna być wyprowadzona na pin AH30.

2.4 Uśrednianie sygnału

Czynności opisane w tej podsekcji nie powinny być stosowane w realnym sprzęcie. Ponieważ wejście LINE-IN w przypadku testowanego sprzętu okazało się bardzo zaszumione, zdecydowano się na uśrednienie sygnału z lewego oraz prawego kanału, oraz wartość uśrednioną podać na wejście transformaty.
Czynność taka sprawi, że dla dźwięków o znacznej różnicy w fazie między kanałami, mogą wystąpić artefakty oraz błędy w przetwarzaniu dźwięku, jednak dla dźwięków relatywnie zgodnych w fazie, podejście takie redukuje szum wejściowy.
Przykładowo, jeśli na wejściu pojawi się szum sinusoidalny przesunięty w fazie na jednym kanale o dokładnie 180 stopni względem drugiego, to dodanie amplitud obu kanałów spowoduje wyzerowanie się końcowej wartości. Jednak, jeśli sygnał jest zgodny w fazie na obu kanałach, dodanie wartości amplitud i podzielenie ich przez dwa zwraca taką samą wartość, jak amplituda jednego z sygnałów.

3 Transformata Fouriera oraz układy wspomagające



Rysunek 2: Fragment układu realizujący transformatę Fouriera

3.1 Częstotliwość próbkowania oraz konfiguracja transformaty

Do analizy sygnałów audio konieczne jest uzyskanie sporej rozdzielczości transformaty Fouriera. W tym przypadku założeniem było uzyskanie rozdzielczości około 10 Hz. W celu uzyskania takiego wyniku użyto transformaty 2048-punktowej oraz dzielnika częstotliwości, który zmniejszał częstotliwość wejściową (50 MHz) 2440-krotnie.

Uzyskano więc częstotliwość pomiaru rzędu: $50\mathbf{MHz}/2048/2440 \approx 10.006\mathbf{Hz}$
Sam podział częstotliwości wejściowej wykonano na prostym dzielniku liczącym do wartości 1220.

Ponieważ zastosowano transformatę 2048-punktową, a używane jest tylko wejście rzeczywiste transformaty, zgodnie z twierdzeniem o próbkowaniu maksymalna częstotliwość, którą można mierzyć wynosi 10 kHz. Ludzkie ucho jest w stanie usłyszeć wyższe dźwięki, lecz rzadko pojawiają się one w utworach muzycznych, maksymalna wartość rzędu 10 kHz jest więc wystarczająca.

Finalnie, konfiguracja modułu FFT prezentuje się następująco:

- transformata 2048-punktowa,
- transformata dwukierunkowa - wymóg platformy,
- reprezentacja danych typu fixed-point,
- typ prezentacji danych - variable streaming,
- szerokość wejścia - 24 bity,
- szerokość wyjścia - 36 bitów.

3.2 Moduł FFTController

Transformata do poprawnego działania wymaga pewnych dodatkowych kroków - po pierwsze, moduł musi uzyskać informacje, kiedy zaczyna się oraz kończy pakiet danych. W tym celu stworzono moduł FFTController, który co 2048 taktów zegara wysyła sygnał SOP - start of packet, informujący transformatę o rozpoczęciu transmisji pakietu danych. Również co 2048 taktów wysyłany jest sygnał EOP - end of packet, informujący o końcu pakietu. Naturalnie, sygnał SOP od sygnału EOP dzieli 2047 taktów.
Zadaniem modułu FFTController jest więc generowanie sygnałów SOP oraz EOP.

3.3 Moduł FFTWrapper

Nieco więcej zadań spełnia moduł FFTWrapper. Instancjonuje on zarówno moduł samej transformaty, oraz moduł FFTController, łącząc je ze sobą.
Transformata po uruchomieniu sprzętu nie jest gotowa do pracy, konieczne jest wystawienie na wejście RESET transformaty stanu wysokiego - tym zadaniem również zajmuje się moduł FFTWrapper, odczekując określoną liczbę cyklów a następnie ustawiając stan wysoki na wejście RESET modułu transformaty.
Moduł ustawia też sztywno następujące wejścia transformaty:

- sink_error - ustawiono na zero - w tym układzie nie korzysta się z kontroli błędów;
- source_ready - ustawiono na 1 (true) - w układzie nie pojawia się analizowanie wyjścia FFT wymagające większej ilości czasu, więc układ jest zawsze gotowy na zaakceptowanie wyjścia FFT;
- fftpts_in - ustawiono na wartość 2048 (wektor rozmiaru 12) - w tym przypadku ilość punktów transformaty nie zmienia się w trakcie działania programu;

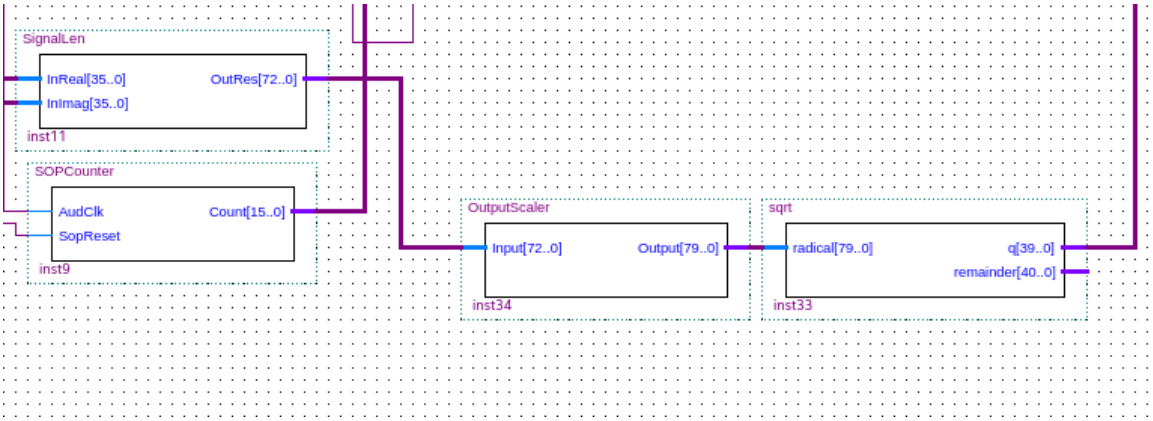
Moduł wyprowadza następujące wejścia:

- Clk - wejście zegarowe - na wejście podłączony jest sygnał zegarowy po podzieleniu dzielnikiem opisanym powyżej;
- InSignal - wejście audio - w tym przypadku podłączone do wyjścia modułu uśredniającego dwa kanały audio;
- SinkReady - wejście informujące, czy wejście audio do transformaty jest poprawne i gotowe do przesyłu danych, zostało podłączone do wyjścia *_channel_ready układu reprezentującego kodek, jednak formalnie nie jest ono potrzebne z powodu dużej częstotliwości próbkowania kodeku względem częstotliwości próbkowania transformaty.

Oraz następujące wyjścia:

- RealPower - wyjście rzeczywiste transformaty o szerokości 36 bitów;
- ImagPower - analogicznie, wyjście urojone transformaty;
- FFTSourceSOP - sygnał SOP wejściowy transformaty nie jest synonimiczny z sygnałem SOP wyjścia transformaty - bowiem sygnał wyjściowy jest przesunięty o pewną wartość cykli. Konieczne jest więc poinformowanie następujących po module transformaty układów analizujących jej wyjście, kiedy pojawia się pierwsza próbka.

4 Wstępne operacje na sygnale wyjściowym FFT



Rysunek 3: Fragment układu wykonujący wstępne operacje na wyjściu FFT

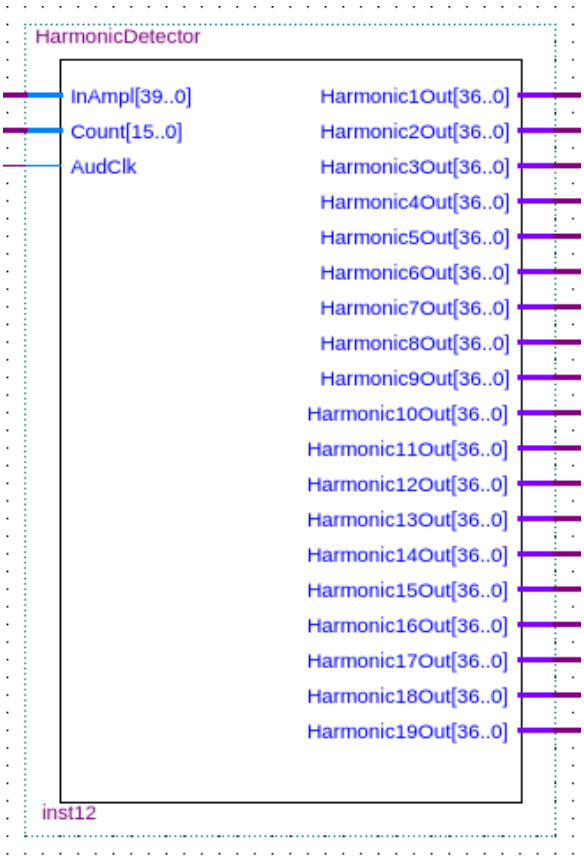
4.1 Obliczanie wyjściowej wartości dla próbki

Dla zastosowania opisanego w tym dokumencie, nie jest konieczna znajomość osobno wyjść urojonych oraz rzeczywistych modułu transformaty, a jedynie ich moduł. Obliczaniem tejże długości zajmuje się moduł SignalLen, który podnosi do kwadratu wartości rzeczywiste oraz urojone, a następnie dodaje je do siebie.
Tak dodane wartości następnie są skalowane poprzez moduł OutputScaler, który zwiększa rozmiar wektora wejściowego, do rozmiaru którego wymaga układ sqrt, wykonujący operację pierwiastkowania danych wejściowych.
Formalnie, zastosowanie układu pierwiastkującego nie jest konieczne - bowiem funkcja pierwiastek jest monotonicznie rosnąca, a więc nie zmienia samej charakterystyki informacji, jednak niewykonanie operacji pierwiastkowania utrudnia analizowanie danych w kolejnych układach, ponieważ wartości, które pojawiają się dla rzeczywistego sygnału bardzo szybko rosną dla niewielkich zmian amplitudy sygnału audio.

4.2 Obliczanie indeksu próbki

Moduł transformaty nie informuje, która próbka znajduje się aktualnie na wyjściu, dostarcza jedynie sygnał FFTSourceSOP, który informuje o rozpoczęciu transmisji całego pakietu składającego się z 2048 próbek wyjściowych.
W celu uzyskania numeru próbki skonstruowano więc moduł SOPCounter, który odlicza od zera do 2047, resetując wartość na zero po napotkaniu sygnału FFTSourceSOP. Wyjście tego układu to obliczony indeks informujący następujące dalej układy o indeksie aktualnej próbki.

5 Analiza wyjścia FFT - moduł HarmonicDetector



Rysunek 4: Układ realizujący operację analizy danych wyjściowych transformaty.

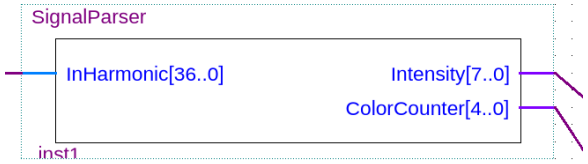
5.1 Analiza danych wyjściowych FFT

Dane, które zostały przetworzone przez moduł transformaty oraz moduły wstępnych operacji na danych wyjściowych wędrują następnie do układu HarmonicDetector (wejście InAmpl). Moduł ten realizuje aktualizowanie przedziałów częstotliwościowych.
Każdy przedział częstotliwości ma zdefiniowany numer początkowego i końcowego indeksu próbki w transformacie Fouriera. Na przykład, pierwszy przedział obejmuje próbki o indeksach 1-1 (włącznie), a więc próbka o indeksie 1 z transformaty fouriera będzie analizowana dla tego przedziału w układzie HarmonicDetector; z kolei indeksy próbek np. dla siódmego przedziału to 9-12, a więc przedział siódmy będzie odpowiadał za przedziały częstotliwości w zakresie 90-120Hz (ponieważ rozdzielczość transformaty wynosi 10Hz).

Próbki z każdego kolejnego przedziału są analizowane po kolei, moduł HarmonicDetector zapisuje największą wartość spośród próbek odpowiadających danemu przedziałowi, a pod koniec analizowania ostatniej z próbek (dla danego przedziału), ustawia na wyjście dla danego przedziału tę właśnie maksymalną wartość. Wartość ta nie zmieni się podczas analizowania próbek z kolejnych przedzia-

łów.
Wyjścia tego układu reprezentują kolejne przedziały analizowane przez moduł, w aktualnej wersji modułu przedziałów jest 19, a więc układ zdolny jest do dalszego przetwarzania 19 rozłącznych przedziałów częstotliwości.

5.2 Przygotowanie danych do przesłania - moduł SignalParser



Rysunek 5: Parser informacji z danego przedziału.

Kolejnym krokiem analizy jest odpowiednie przygotowanie danych do przesłania na taśmę LED. Założono następującą reprezentację danych:

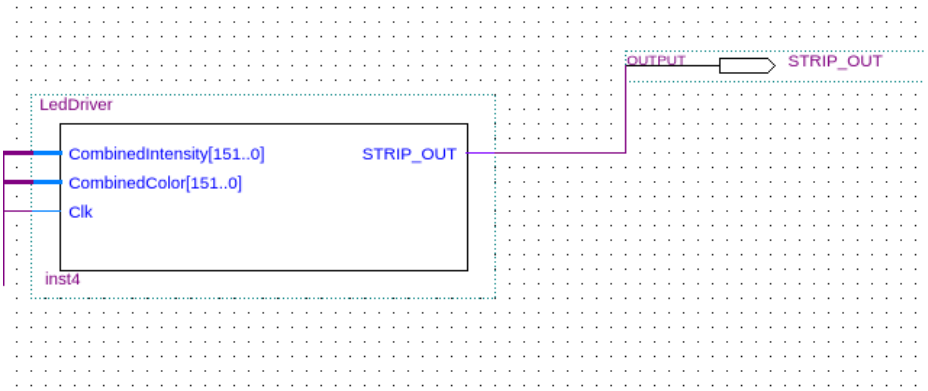
- każda dioda świeci z różną intensywnością, definiowaną przez 8-bitową liczbę *Intensity*,
- w zależności od tego, jak często dany przedział częstotliwości prezentuje wartość amplitudy większą od zadanego progu, kolor diody zmienia się z czerwonego, poprzez zielony, do niebieskiego, reprezentuje to zmienna *ColorCounter*, o zakresie wartości 0-16.

Moduł SignalParser realizuje powyższe założenia - przede wszystkim pobiera sygnał wyjściowy z modułu HarmonicDetector, odejmuje od niego pewną wartość reprezentującą szum, oraz dzieli tak, aby pozostająca wartość leżała w przedziale 0-255 odpowiadającym zakresowi zmiennej *Intensity*. Jeśli amplituda sygnału po przeskalowaniu jest większa niż 255, to jest ona obcinana do tej wartości. Następnie, korzystając z obliczonej wartości *Intensity* obliczana jest wartość *ColorCounter* kolejno poprzez inkrementację, bądź dekrementację zmiennej licznikowej reprezentującej ilość wystąpień danej częstotliwości.

5.3 Zmiana reprezentacji danych - moduł DataSerializer

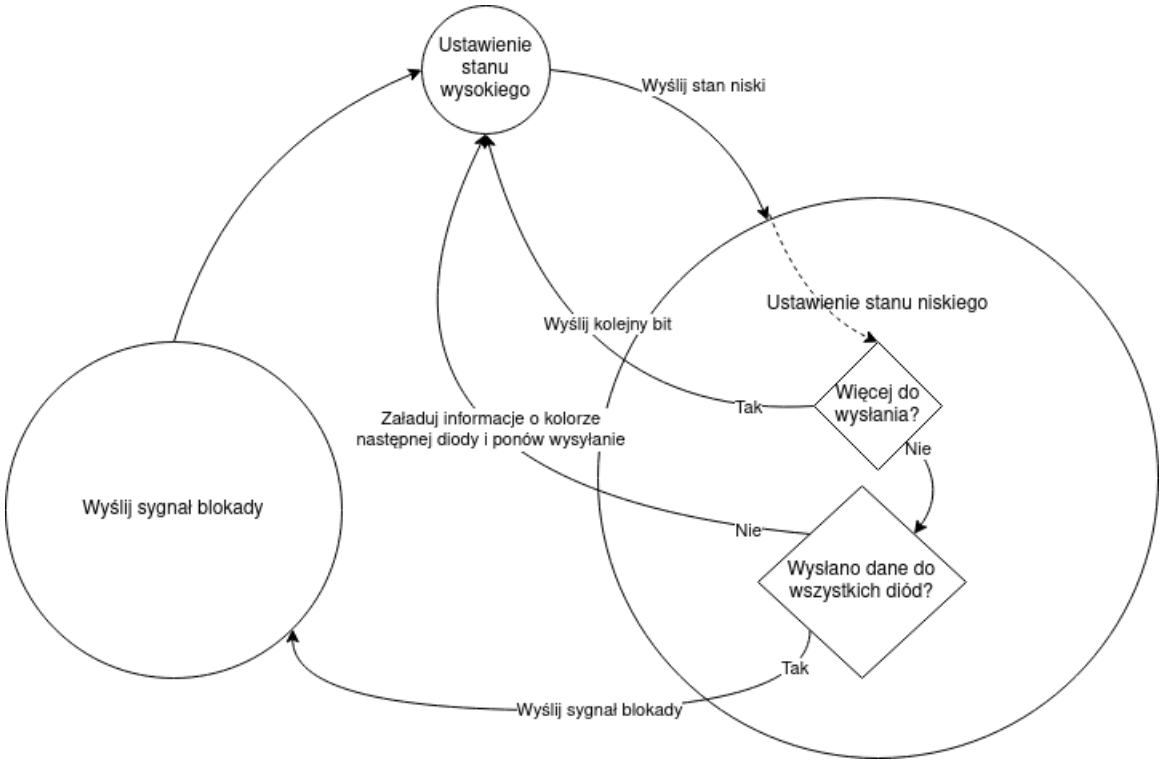
Ponieważ z punktu widzenia oprogramowania zajmującego się sterowaniem taśmą LED łatwiejsze jest analizowanie danych dostarczonych w sposób szeregowy, zastosowany został moduł DataSerializer, który serializuje dane wyjściowe z dziewiętnastu modułów SignalParser do dwóch 152-bitowych wektorów.

6 Prezentacja danych - sterowanie paskiem LED



Rysunek 6: Moduł odpowiedzialny za sterowanie paskiem LED.

Po uzyskaniu odpowiedniego formatu danych konieczne jest przesłanie ich do taśmy LED WS2812B. Moduł za to odpowiedzialny, LedDriver, działa na podstawie maszyny stanu, którą można odpisać poniższym poglądowym schematem:



Rysunek 7: Moduł odpowiedzialny za sterowanie paskiem LED.

Taśma LED wymaga wysłania dla każdej diody, szeregowo, następujących po sobie stanów wysokich i niskich, których czasy trwania są odpowiedzialne za logiczną jedynkę lub zero.
Moduł LedDriver za każdą iteracją oblicza kolor oraz intensywność diody LED, która odpowiada danemu przedziałowi częstotliwości, a następnie wysyła bity w sposób specyficzny dla zaprezentowanej taśmy LED. Po wysłaniu wszystkich bitów dla danej diody, inkrementowany jest wewnętrzny licznik i sytuacja powtarza się dla następnej z diód, aż do przesłania informacji do każdej z nich. Następnie, wysyłany jest sygnał LATCH, "informujący"diody o końcu transmisji, skutkiem czego wyświetlany jest odpowiedni kolor o odpowiedniej, obliczonej wcześniej, intensywności. Wszystkie informacje wysyłane są za pomocą pojedynczego, wyjściowego pinu danych.

7 Podsumowanie

Konstrukcja układu zaprezentowanego w tym dokumencie nie jest skomplikowana, jeśli używane są dostarczone moduły do szybkiej transformaty Fouriera, której implementacja prawdopodobnie byłaby najbardziej czasochłonna. Mimo to, większość czasu wymaganego na implementację należy poświęcić na zapoznanie się ze środowiskiem Intel Quartus oraz zasadą funkcjonowania układów FPGA, jak również dokumentacją używanych modułów.