

# Projekt SYCYF

Zespół nr 12

Jędrzej Joniec

Michał Podgajny

Maciej Antosz

Sebastian Skrzek

Michał Kamiński

Politechnika Warszawska, Telekomunikacja

9 czerwca 2021

## Historia zmian

Wersja	Data	Autor	Opis zmian
1.0	12.03.2021	JJ, PM, AM, SS	Pierwsza wersja raportu Etapu 1
1.1	14.03.2021	KM	Rozszerzono opis rozdziału 2.3 o opisy etapów Double Diamond Design Thinkingu i teorii psychologicznej
1.2	15.03.2021	PM	Poprawa równomiernego rozłożenia tekstu w dokumencie w celu poprawy czytelności
1.3	15.03.2021	JJ	Dodanie do spisu metod metody "5 why" z uzasadnieniem wyboru tej metody
2.1	06.04.2021	JJ, PM, AM, SS, KM	Rozpoczęcie etapu 2
2.2	07.04.2021	JJ, PM, AM, SS, KM	Dopracowanie podpunktów 3.2 i 3.4
2.3	07.04.2021	JJ, MK	Stworzenie tabelki zaangażowania
3.1	25.04.2021	MP	Poprawa błędów zasugerowana przez prowadzącego projekt
3.2	27.04.2021	JJ, PM, AM, SS, KM	Opracowanie etapu 3

3.2	22.05.2021	JJ, PM, AM, SS, KM	Opracowanie etapu 4
4.0	05.06.2021	JJ, PM, AM, SS, KM	Opracowanie etapu 5

# Spis treści

<b>Historia zmian</b>	1
<b>1. Wstęp</b>	6
<b>2. Organizacja prac</b>	6
2.1. Zarządzanie projektem	6
2.2. Design Thinking	6
2.2.1. Metody	8
2.2.2. Narzędzia	9
<b>3. Informacje podstawowe</b>	10
3.1. Szczegółowe objaśnienie zadania	10
3.1.1. Dane uzyskane z treści zadania	11
3.1.2. Prawdopodobny obiekt, dla którego wykonujemy pomiar	12
3.1.3. Problemy związane z pomiarem	12
3.2. Budowa urządzenia	13
3.2.1. Uproszczony schemat budowy systemu pomiarowego	13
3.2.2. Przykładowe technologie wykonania procesora pomiarowego (DSP)	13
3.2.3. Zadania procesora pomiarowego	13
3.2.4. PLD	13
3.2.5. FPGA	14
3.2.6. Języki opisu sprzętu	14
3.2.7. Przykładowe rodzaje RAM-u	14
3.2.8. CPU	14
3.3. Metody pomiarów odległości za pomocą lasera	15
3.3.1. Rodzaje dalmierzy	15
3.3.2. Dalmierz laserowy	15
3.3.3. Dalmierz fazowy	15
3.3.4. Interferometria	16
3.3.5. Dalmierz impulsowy	16
3.3.6. Analiza przydatności metod	16
3.4. Metody interpretacji wyników	17
3.4.1. Podstawowe dane na temat odebranego sygnału przez odbiornik	17
3.4.2. Metody wykrycia momentu dotarcia impulsu do odbiornika	17
3.4.3. Analiza widma	18
3.4.4. Transformata Fouriera	18
3.4.5. Algorytm FFT i jego odmiany	19
3.5. Poszukiwanie prostszej metody analizy	19
3.5.1. Korelacja	20
<b>4. Koncepcja</b>	20
4.1. Wybór metody : Korelacja	20
4.1.1. Zebranie metod	20
4.1.2. Zalety/wady metody korelacji	20
4.1.3. Zalety/wady metody FFT	21
4.1.4. Ostateczna decyzja	21
4.2. Budowa systemu	21
4.2.1. Schemat	21
4.2.2. Objaszenie wstępne działania elementów	22
4.3. Przykład ilustrujący działanie systemu	23
4.3.1. Pomiar Ziemia - Księżyca	23
4.3.2. Przepływ informacji podczas pomiaru (procedura pomiarowa)	23
4.3.3. Wnioski jakie można wyciągnąć z takiego pomiaru	23
4.4. Algorytm	24
4.4.1. Pseudokod	24
4.5. Symulacja w środowisku Python	24
4.5.1. Model referencyjny	24
4.5.2. Wykonanie z użyciem biblioteki numpy	25
4.5.3. Wykonanie z bez użycia bibliotek	26
4.6. Oczekiwania wobec przeprowadzonej analizy	27
4.6.1. Dane do testowania	27
4.6.2. Scenariusze testowe	27

4.6.3. Wyniki testów modelu referencyjnego . . . . .	27
4.6.4. Wnioski . . . . .	29
<b>5. Implementacja . . . . .</b>	<b>29</b>
5.1. Sposób realizacji systemu . . . . .	29
5.1.1. Ostateczna architektura systemu . . . . .	29
5.1.2. Moduły . . . . .	30
5.1.3. Korelator . . . . .	31
5.1.4. Podmoduł tima . . . . .	33
5.1.5. Podmoduł regPrzes . . . . .	34
5.1.6. MyOut . . . . .	38
5.1.7. Detektor - sposób detekcji momentu odbicia . . . . .	39
5.1.8. Moduł timer . . . . .	40
5.1.9. Współpraca układu DSP z procesorem . . . . .	41
5.1.10. Zastosowana pamięć . . . . .	41
5.1.11. Przykład przepływu danych po układzie . . . . .	41
5.2. Wstępne testy . . . . .	41
5.2.1. Wydruki programu ModelSim . . . . .	41
5.2.2. Omówienie szybkości . . . . .	42
5.2.3. Omówienie wymaganych zasobów . . . . .	43
5.3. Ocena rozwiązania . . . . .	44
5.3.1. Ocena szybkości rozwiązania . . . . .	44
5.3.2. Ocena zajętości wymaganych zasobów . . . . .	44
5.4. Scenariusze testowe . . . . .	45
5.4.1. Scenariusz 1 . . . . .	45
5.4.2. Scenariusz 2 . . . . .	45
5.4.3. Scenariusz 3 . . . . .	45
5.4.4. Działania w razie uzyskania negatywnych wyników testów benchtesta . . . . .	46
5.5. Weryfikacja funkcjonalna . . . . .	46
5.5.1. Założenia testbencha . . . . .	46
5.5.2. Moduł wejściowy - plik do z wektorami testowymi . . . . .	46
5.5.3. Moduł generatora . . . . .	48
5.5.4. Moduł monitora . . . . .	48
5.5.5. Właściwy benchtest . . . . .	49
5.5.6. Wydruki z programu ModelSim . . . . .	50
5.5.7. Ocena poprawności działania . . . . .	50
<b>6. Uruchomienie . . . . .</b>	<b>50</b>
6.1. Parametry sprzętu na którym dokonano uruchomienia . . . . .	51
6.1.1. Model płytka . . . . .	51
6.1.2. Parametry . . . . .	52
6.2. Budowa urządzenia testowego . . . . .	52
6.2.1. Elementy . . . . .	52
6.2.2. Funkcje przycisków . . . . .	53
6.2.3. Funkcje kontrolek . . . . .	53
6.2.4. Wyświetlacz 7-segmentowy . . . . .	53
6.3. Budowa układu testowego . . . . .	54
6.3.1. Schemat . . . . .	54
6.3.2. Moduł główny . . . . .	54
6.3.3. Układ DSP . . . . .	56
6.3.4. Przyciski . . . . .	56
6.3.5. ROM . . . . .	59
6.3.6. Układ przeliczający działający w dwóch trybach . . . . .	61
6.3.7. Wyświetlacz 7-segmentowy . . . . .	62
6.3.8. Zamiana liczby binarnej na dziesiętną . . . . .	64
6.3.9. Obsługa wyświetlacza multipleksowanego . . . . .	65
6.3.10. Komunikacja błędów . . . . .	67
6.4. Dane wejściowe . . . . .	69
6.4.1. Przetworzenie danych wejściowych . . . . .	69
6.4.2. Postać ROMu w Verilogu . . . . .	70
6.4.3. Oczekiwane działanie modułu imitującego wyjście lasera . . . . .	71

6.5. Uruchomienie . . . . .	71
6.5.1. Procedura uruchomienia na płytce . . . . .	71
6.5.2. Procedura testu . . . . .	72
6.5.3. Opis uzyskanych rezultatów . . . . .	73
<b>7. Podsumowanie . . . . .</b>	<b>75</b>
7.1. Wnioski . . . . .	75
7.1.1. Raport na temat zajętości zasobów . . . . .	75
7.1.2. Raport na temat maksymalnej częstotliwości . . . . .	75
7.1.3. Co udało się wykonać . . . . .	76
7.1.4. Czego nie udało się wykonać . . . . .	76
7.1.5. Przyczyny . . . . .	76
7.2. Zaangażowanie poszczególnych studentów . . . . .	76
<b>Literatura . . . . .</b>	<b>77</b>

## 1. Wstęp

W pierwszym etapie projektu celem było stworzenie zespołu i wstępna organizacja warsztatu pracy. Podstawą organizacji takowego warsztatu jest obranie długoterminowego planu pracy, który umożliwia pełne wykonanie projektu oraz stworzenie kompletnego i dopracowanego rozwiązania problemu, który jest tematem projektu czyli **"Próba pomiaru odległości do pewnego obiektu"**.

Pracę postanowiono podzielić na **pięć** etapów. Poszczególne etapy realizacji projektu obejmują:

- I Etap wstępny – stworzenie zespołu i organizacja warsztatu pracy,
- II Etap zdobywania informacji – analiza literatury, istniejących metod, zebranie wiedzy teoretycznej związanej z tematem projektu,
- III Etap opracowania koncepcji – szukanie rozwiązań, opracowanie koncepcji rozwiązania na podstawie zdobytej wiedzy w poprzedni etapach, opracowanie prostego modelu referencyjnego i danych do testowania
- IV Etap implementacji – na tym etapie będą rozwijane i rozbudowywane koncepcje projektowe docelowego systemu, modelowane elementy systemu w HDL, weryfikowane funkcjonalnie, integrowane oraz oceniane prototypy,
- V Etap uruchomienia – wdrożenie projektu, uruchomienie na docelowej platformie, przetestowanie według wcześniej opracowanych scenariuszy testowych.

Prace wykonane w ramach każdego etapu będą opisane w oddzielnym raporcie oraz w raporcie końcowym.

Raport będzie zawierał opisy w których będą wykorzystane rysunki, wykresy i tabele dla ułatwienia zrozumienia koncepcji Autorów.

Postępy prac należy będą szczegółowo i skrupulatnie opisywane w historii zmian tak, aby opiekun projektu mógł śledzić wkład poszczególnych członków zespołu w realizację projektu.

## 2. Organizacja prac

- podejście Design Thinking (w wersji Double Diamond lub innej),
- organizacja warsztatu pracy, dobór narzędzi (Overleaf, Microsoft Teams, GitHub, itp.)

### 2.1. Zarządzanie projektem

Naszym projektem zarządzają wszyscy członkowie naszego pięcioosobowego zespołu. Mamy wspólny dostęp do naszych repozytoriów na platformie Overleaf i GitHub. Spotykamy się raz w tygodniu w terminie najbardziej dogodnym dla wszystkich, między spotkaniami każdy z nas szuka informacji które mogą pomóc nam w wykonaniu projektu. Podczas naszych spotkań wymieniamy się naszymi pomysłami i wykonujemy wymogi etapowe. Wybraliśmy wspólnie lidera, Jędrzeja Jońca który motywuje nas do pracy i dba o organizację naszego zespołu.

### 2.2. Design Thinking

Do realizacji myślenia projektowego wybrano metodę Double Diamond Design Process. Korzysta się w niej z dwóch wzajemnie przenikających się podejść do rozwiązywania problemów.

Pierwszym z nich jest **myślienie dywergencyjne** (rozbieżne). Twórca terminu, Joy Paul Guilford, definiuje je jako to stawiające nacisk na kreatywność grupy i generujące wiele możliwych sposobów rozwiązywania problemu w dość krótkim czasie[1]. Charakteryzuje się spontanicznością, otwartością, korzystaniem z wyobraźni i nielinowością, wobec czego dobrze się nim kierować na początkowych stadiach danego zadania, gdy jest wiele niewiadomych. Każda idea rozwiązania może być dobra, jednak zbyt długi czas generowania rozwiązań może skutkować zastojem.

Z kolei **myślienie konwergencyjne** (zbieżne) jest pozbawione większego procesu twórczego. Wykorzystuje się je w sytuacjach, gdzie trzeba znaleźć jedną, konkretną odpowiedź na pytanie, czyli np. w testach wielokrotnego wyboru. Wykorzystywanie umiejętności analitycznych, krytyczne myślenie czy podejmowanie decyzji są jego cechami charakterystycznymi[2]. W złożonych projektach korzysta się z niego w drugim stadium zadania, gdzie po wyeksplorowaniu problemu nadchodzi czas na wybranie najodpowiedniejszego rozwiązania.

Double Diamond Design Process składa się z 4 etapów, wykorzystujących naprzemiennie myślenie rozbieżne i zbieżne:

— **1. Etap odkrywania (discover), myślenie dywergencyjne**

Pierwszy etap modelu polega na zrozumieniu istoty problemu oraz poznawaniu różnych zmiennych mających wpływ na niego i jego możliwym rozwiązaniu[3]. Celem jest odkrycie przeszkód i obszarów możliwości w obszarze badań. W projekcie będzie on obejmował zrozumienie postawionego zadania poprzez analizę literatury i istniejących metod oraz zebranie wiedzy teoretycznej związanej z tematem projektu.

— **2. Etap rozwijania (define), myślenie konwergencyjne**

Etap rozwijania polega na przefiltrowaniu wszystkich informacji uzyskanych z pierwszego etapu i ich rozwinięcie, a ostatecznie odnalezieniu tego jedynego rozwiązania. Właściwe osiągnięcie tego jest ważne, ponieważ będzie to definiujący cel, do którego projekt będzie dążyć. Tutaj będzie on odnosił się do szukania rozwiązań i opracowania koncepcji rozwiązania na podstawie wiedzy zdobytej ze wcześniego etapu. Ponadto, zostanie stworzony prosty model referencyjny i dane do testowania.

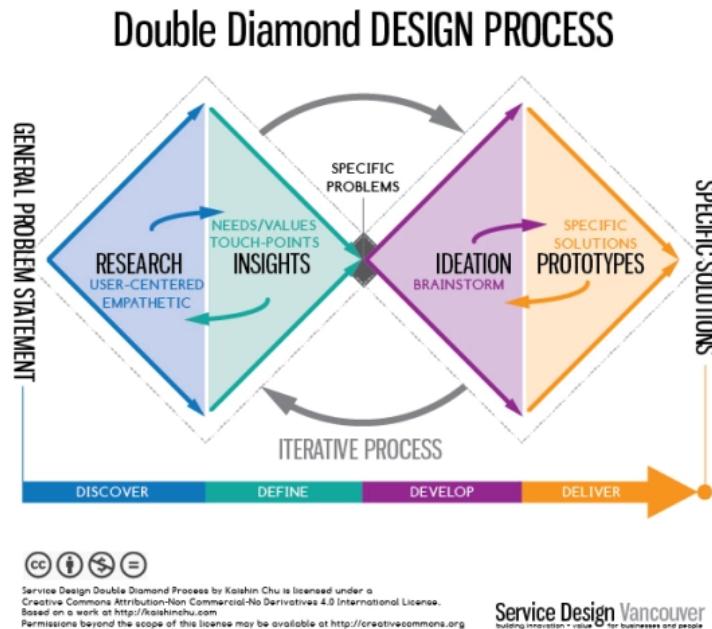
— **3. Etap definiowania (develop), myślenie dywergencyjne**

Celem tego etapu jest opracowanie jak największej liczby pomysłów dotyczących sposobu rozwiązania problemu, który zdefiniowano na etapie 2. Te pomysły powinny być proponowane niezależnie od ich wykonalności - kluczowa jest tu ilość i zakres myślenia.[4] Gdy jest wiele pomysłów, konieczne jest zawężenie ich do tych, które mogą najlepiej rozwiązać postawiony problem. Kryteria „najlepszego potencjału” należy zdefiniować w oparciu o wykonalność, zróżnicowanie / wyjątkowość, czy szybkość wdrażania. Pomyśl o największym potencjale są następnie przenoszone do następnego etapu. W trakcie trwania tego etapu projektu zostaną rozwinięte i rozbudowane koncepcje projektowe docelowego systemu, zamodelowane elementy systemu w HDL, zweryfikowana funkcjonalność, a prototypy zintegrowane i ocenione.

— **4. Etap dostarczania (deliver), myślenie konwergencyjne**

W ostatnim etapie zostaną zamienione najlepsze pomysły w prototypy - wstępny / wstępny zarys / model rozwiązania problemu zdefiniowanego na etapie 2[5]. Następnie testowana będzie zdolność prototypu do rozwiązywania problemu. Wnioski z tego testu są następnie wykorzystywane, a prototyp jest udoskonalany i ponownie testowany. Ten proces będzie

powtarzany, dopóki prototyp nie będzie całkowicie rozwiązywać problem w satysfakcjonujący sposób. Tutaj dostarczanie będzie polegało na wdrożeniu projektu, uruchomieniu na docelowej platformie i przetestowaniu według wcześniej opracowanych scenariuszy testowych



Rysunek powyżej przedstawia schemat Design Thinkingu "Double Diamond Design Process"

### 2.2.1. Metody

W celu realizacji naszego projektu wybraliśmy trzy metody projektowe. Zdecydowanie najważniejszą z nich jest metoda **diamond thinking**, pozwala nam ona dzięki podzieleniu procesu realizacji na cztery części realizować nasze założenia etap po etapie.

Kolejną metodą jest **brainstorming** (burza mózgów) podczas naszych spotkań wymieniamy się wszystkimi pomysłami jakie przyjdą nam do głowy i staramy się wybrać takie które pomogą nam zrealizować projekt. Pozwala to wszystkim członkom zespołu zaangażować się w projekt a także wspólnie podjąć najlepsze decyzje.[6]

Trzecią wybraną przez nas metodą jest **5 why** (5 dlaczego), polega ona na zadawaniu sobie pytania "dlaczego?" tak długo aż nie znajdziemy przyczyny naszego problemu. Będzie ona szczególnie przydatna w sytuacji w której dojdziemy do ściany i nie będziemy w stanie ruszyć z naszym projektem dalej. Zachęca ona każdego członka zespołu do dzielenia się swoimi wątpliwościami i pomysłami, bez zrzucania winy na innych. [7]

Ostatnią mimowolnie wybraną metodą jest metoda **trial and error method** (metoda prób i błędów), która w oczywisty sposób jest obecna w trakcie wykonywania naszych zadań. Próbujemy implementować nasze rozwiązanie w różny sposób ze świadomością że jeśli niektóre z nich okażą się nieefektywne lub nieskuteczne to możemy je poprawić. [8]

### 2.2.2. Narzędzia

- **Microsoft Teams** – spotkania, kontakt z prowadzącymi, przekazywanie raportów;

Microsoft Teams jest narzędziem wybranym przez naszą uczelnię do prowadzenia zajęć i kontaktowania się z wykładowcami. Uznano, że nie należy szukać nowego narzędzia aby nie wprowadzać niepotrzebnego zamieszania zarówno koordynatorom jak i członkom zespołu. Dzięki temu uzyskano jedną platformę do kontaktowania się zarówno w obrębie członków zespołu jak i innych osób koordynujących pracę nad powyższym projektem. [9]

- **Overleaf** – łatwe, wspólne tworzenie raportu

Narzędzie Overleaf umożliwia sprawne przetwarzanie tekstów w formacie **TEX** (umożliwiającym szybką obróbkę dużej ilości tekstu), w przeglądarce. Dzięki temu nie jesteśmy ograniczeni oprogramowaniem zainstalowanym na komputerze/ innym urządzeniu które mamy pod ręką i możemy na bieżąco nanosić poprawki do dokumentów/ raportów, uzupełniać je o świeże pomysły. Pozwala to na skrócenie czasu potrzebnych na sporządzenie stosownych dokumentów projektowych, niezbędnych do udokumentowania pracy zespołu [10]

- **Środowisko Python** – tworzenie kodu dzięki szerokim bibliotekom poszerzającym i ułatwiającym nam poszukiwania rozwiązania;

Środowisko Python (zupełnione o środowisko **Anaconda**) daje nam dostęp do wielu bibliotek pozwalających dokonywać obliczeń na dużych zestawach danych, opracowywać prototypy rozwiązań, w bardzo krótkim czasie, którego nie da się osiągnąć w innych bardziej zaawansowanych językach programowania (np Java). Dzięki temu proces opracowywania algorytmów przetwarzania danych, które będą musiały być zaimplementowane w sprzęcie zostanie znacznie przyspieszony, przez co będzie można poświęcić więcej czasu na udoskonalenie implementacji sprzętowej HDL [11]

- **GitHub** – przechowywanie kodu źródłowego, zdalne wprowadzanie zmian, łatwy dostęp dla całego zespołu.

GitHub umożliwia łatwy i szybki dostęp do kodu, zarówno do programów wykoanych w Python-ie jak i kodów implementacji sprzętowej wykonanych w HDL-u (**Verilog HDL**), uodprawniając się na awarie komputerów itp., dzięki czemu niwelujemy ryzyko nieodwracalnej straty efektów naszej pracy. Dodatkowo GitHub umożliwia łatwe przetwarzanie zespołowe kodu

- **Quartus i ModelSim Altera** – narzędzie umożliwiające wykonanie sprzętowej implementacji rozwiązania problemu.

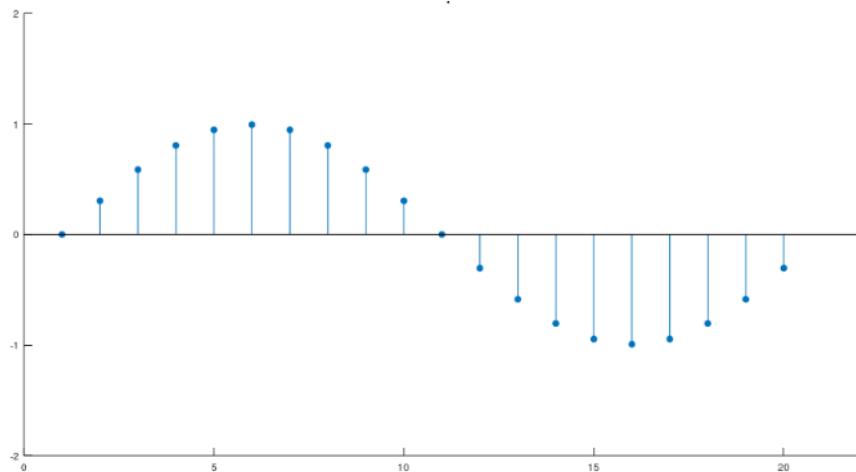
Narzędzie Intela służące do programowania układów FPGA, uruchamiania implementacji sprzętowych na płytach testowych itp. Jest niezbędne do wykonania implementacji sprzętowych rozwiązań. [12] [13]

### 3. Informacje podstawowe

#### 3.1. Szczegółowe objaśnienie zadania

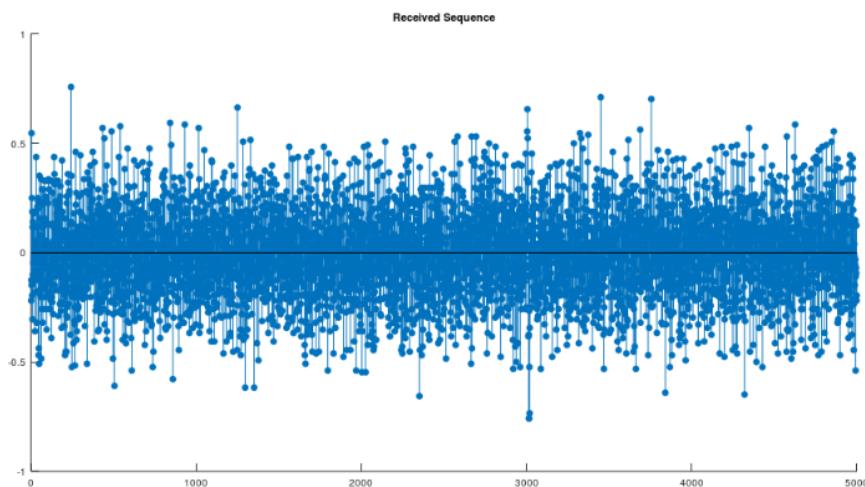
Zespół dostał zadanie wykonania systemu cyfrowego służącego do pomiaru odległości do pewnego obiektu przy użyciu danych odebranych przez odbiornik (sygnału), zapisanego w ośmiu bitach. Odbiornik oczekuje na odbity sygnał nadany przez laser, który ma być poddany analizie, która finalnie ma dać odległość do danego obiektu

Najpierw w ciągu nieznanego czasu  $t_0$  wysłano wiązkę laserową zmodulowaną sygnałem sinusoidy z szybkością  $10^6$  próbek na sekundę. Sygnał ten miał okres  $T = 20$  próbek. Każda próbka trwa według założeń  $t = \frac{1s}{10^6} = 0,000001s = 1\mu s$ , zatem okres sygnału wynosi  $T = 20 \mu s$ , co daje nam częstotliwość  $f = 50 \text{ kHz}$



Rysunek przedstawia wykres zmodulowanego sygnału sinusoidalnego wysłanego przez laser w kierunku obiektu

Potem odczekano **2,5 sekundy**. Następnie rejestrację światła odbitego prowadzono przez  $t_p = 5$  milisekund z szybkością  $10^6$  próbek/sekundę i rozdzielczością 8 bitów na próbce. Odebrany sygnał zaprezentowano na wykresie poniżej.



Rysunek przedstawia wykres odebranego sygnału przez odbiornik

Widoczny jest wysoki poziom zaszumienia sygnału, co będzie utrudniać pomiar tejże odległości. Na podstawie tych danych należy:

- - opracować metodę dokonania pomiaru na podstawie dostępnych danych,
- - zaimplementować sprzętowy moduł części cyfrowej (DSP Processor) systemu dokonującego pomiaru,
- - podać co jest prawdopodobnym obiektem i zmierzoną odległość od tego obiektu.

Propozycje metod, technologii, które można użyć do wykonania tego systemu, zostały znalezione i opracowane przez zespół podczas wykonania etapu II, którego rezultaty opisano poniżej w tej sekcji raportu.

### 3.1.1. Dane uzyskane z treści zadania

Dokonano analizy danych, które udostępniono wraz z zadaniem. Analizując etapy pomiaru zaprezentowanego powyżej, można wyciągnąć pewne wnioski.

Czas wysyłania sygnału  $t_0$  nie ma znaczenia, poszukujemy więc metody, która nie będzie brała pod uwagę czasu "nadawania" sygnału przez laser.

Najistotniejszą daną z perspektywy wyboru metod pomiarowych jest czas oczekiwania między wysłaniem sygnału a rozpoczęciem rejestracji. Wynosi on  $t = 2,5$  s. Światło w ciągu tego czasu może pokonać bardzo duże odcinki drogi. Analizując w ciągu **2,5 sekundy** dla prędkości światła w próżni  $c = 3 * 10^6$  m/s z wzoru [14] :

$$s = \frac{ct}{2}$$

Dochodziszy do obiektu, światło odbija się i wróciwszy do odbiornika, pokonuje  $s_c = 370$  **000 km**. Mamy więc do czynienia prawdopodobnie z pomiarem obiektu, który nie znajduje się na Ziemi. [14]

Następnie rejestracja **5000 próbek** była potrzebna prawdopodobnie do ustalenia bardziej precyzyjnej odległości od tego obiektu. Skoro jedna próbka trwa  $t_0 = 1$  **us**, możemy ustalić, że pomiar może dać nam dokładność nawet: [14]

$$s_p = \frac{ct_0}{2}$$

Daje nam dokładność  $\pm s_p = 150$  **m**. Jest to więc jak na skalę kosmiczną bardzo precyzyjny pomiar, pozwalający dokładnie rejestrować zmiany położenia tego obiektu względem ziemi itd. [14]

Potrzebna jest więc metoda, która umożliwi odebranie bardzo osłabionego sygnału (przez odległość jaką musi pokonać oraz przez inne zakłócenia odbierane przez odbiornik, co widać na sygnale odebranym). W etapie II poszukiwano metod pomiarowych i analizy danych pobranych przez odbiornik takich, które wydajnie pozwolą przeanalizować takie dane w krótkim czasie (dla  $10^6$  próbek na sekundę).

### 3.1.2. Prawdopodobny obiekt, dla którego wykonujemy pomiar

Na podstawie powyższych danych można stwierdzić odległość od jakiego obiektu jest mierzona. Kluczowym do stwierdzenia tego jest czas oczekiwania  $t_0=2,5$  s. **Jest to przybliżony czas płynięcia światła z Ziemią na Księżyc i z powrotem** [załącznik]. Badanie służy prawdopodobnie więc ustaleniu precyzyjnej odległości w danym momencie Ziemia - Księżyc. Odległość Ziemia - Księżyc wynosi pomiędzy 356,500 km a 406,700 km. Na podstawie czasu rejestracji światła możemy ustalić, że "zakres pomiarowy" systemu dla specyfikacji podanej w zadaniu wynosi od 375000 do 375750 km. Wynika to z obliczeń: [14]

$$s_0 = \frac{ct_0}{2}$$
$$s_1 = \frac{ct_0}{2} + \frac{ct_p}{2}$$

gdzie  $s_0$  to odległość minimalna, jaką można obliczyć przy ustalenach zadania, zaś  $s_1$  jest odlegością maksymalną. Różni się ona od pierwotnej o różnicę czasu wynikającą z czasu rejestracji sygnału odbitego. Oznacza to, że uzyskane dane mieszczą się w zakresie odległości Ziemia- Księżyc, co daje duże prawdopodobieństwo, że mamy do czynienia z takim pomiarem [15]

### 3.1.3. Problemy związane z pomiarem

Jak napisano powyżej, zadaniem zespołu jest opracowanie systemu, który wykona stosownego pomiaru. Poszukiwano więc urządzeń, technologii i algorytmów, które pomogą spełnić wymogi zadania oraz umożliwią wykonanie stosownego pomiaru.

Kluczowym problemem jest bardzo duży szum sygnału rejestrowanego przez odbiornik. Trudno zauważyc jakiekolwiek różnice w nim, które mogłyby nam pokazać, że wtedy odbiornik rejestruje odbity sygnał od obiektu. Będzie to wymagało znalezienia metody analizy sygnału, która pozwoli wyznaczenie n-tej próbki zarejestrowanego sygnału, w której zaczyna się rejestrowanie odbitego sygnału.

Kolejnym problemem jest znalezienie technologii, algorytmów, które dla próbkowania  $10^6$  próbek na sekundę dadzą nam szybką analizę danych dla rejestrowanego sygnału w czasie rzeczywistym. Klasyczny komputer nie przetworzy zaawansowanych metod analizy sygnału w tak krótkim czasie, co będzie wymagać specjalnej techniki wykonania tejże analizy

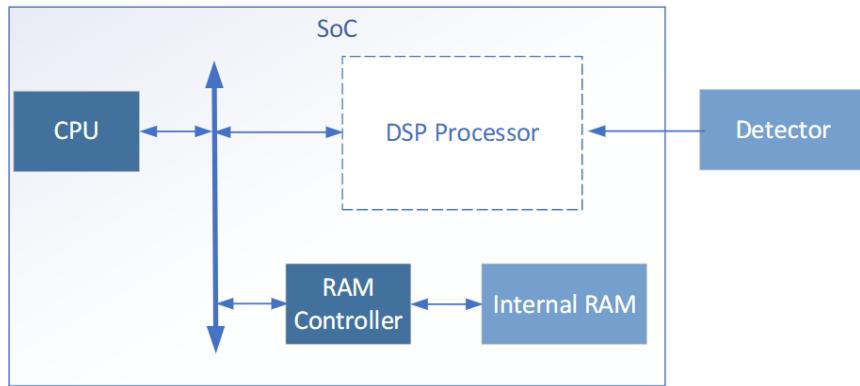
Dodatkowym problemem jest wybranie techniki samego odbioru sygnału, dzięki której będziemy mogli uzyskać taką postać sygnału, jaka została zaprezentowana na wykresie odebranego sygnału w poprzednich podpunktach. Potrzebny będzie specjalny typ urządzenia rejestrującego w taki sposób sygnał.

Dalszy wywód będzie dotyczył szukania metod, algorytmów i technologii, które mogą pomóc w rozwiązaniu postawionego problemu.

### 3.2. Budowa urządzenia

#### 3.2.1. Uproszczony schemat budowy systemu pomiarowego

Zespołowi zlecono wykonanie systemu o poniższej budowie:



Rysunek przedstawia schemat budowy systemu dokonującego pomiaru

#### 3.2.2. Przykładowe technologie wykonania procesora pomiarowego (DSP)

DSP[16] - Digital Signal Processor - jest to grupa układów przeznaczonych do obróbki cyfrowych sygnałów, na przykład akustycznych czy optycznych. Charakterystyczną cechą ich budowy jest to, że mają wstępnie zdefiniowany profil określonego rodzaju sygnału. Ich program posiada osobny obszar pamięci przeznaczony wyłącznie dla niego i dla danych. Umożliwiają równoczesne pobieranie instrukcji i danych poprzez sprzętowe wykonywanie operacji występujących najczęściej przy przetwarzaniu sygnałów (np. filtracji, transformacji Fouriera, korelacji wzajemnej) oraz potokowym przetwarzaniem instrukcji.

#### 3.2.3. Zadania procesora pomiarowego

Procesor pomiarowy powinien przede wszystkim, co jest oczywiste, przetwarzać sygnał na odległość, czyli znaleźć sposób interpretacji sygnału. W tym celu powinien reagować na polecenia głównego procesora oraz gromadzić używane do analizy dane na odpowiednio dobranym RAMie. Oczywiście, innym zadaniem będzie wysyłanie wyników jego pracy dalej

#### 3.2.4. PLD

PLD (ang. Programmable Logic Device) to układ elektroniczny o programowalnej strukturze. Charakteryzuje się tym, że służy do wykonywania czynności równolegle, gdy procesor wykonuje je szeregowo[17], co przekłada się między innymi na prędkość wykonywania zadań. Zbudowany jest on z ogromnej liczby pojedynczych komórek zwanych elementami logicznymi, a na pojedynczą komórkę składają się najczęściej bramki AND, OR, NOT i przerzutnik RS[18]. Konfigurowanie układu, zwane potocznie "programowaniem", polega na zarządzaniu połączeniami w elemencie logicznym pomiędzy przerzutnikami oraz bramkami, wykorzystując języki opisu sprzętu. Wyróżniamy między innymi układy CPLD, PAL oraz wybrany w tym projekcie FPGA.

### 3.2.5. FPGA

FPGA(ang.Field Programmable Gate Array), inaczej zwane bezpośrednio programowalną macierzą bramek- układ PLD charakteryzujący się najwyższym poziomem zaawansowania wśród wszystkich układów PLD[18]. Posiada on matrycę programowalnych bloków logicznych i konfigurowalnych połączeń między nimi. Prawie zawsze nie posiada pamięci stałej, wyłącznie ram RAM, więc po każdym włączeniu muszą zostać ponownie skonfigurowany[17][19][20].

### 3.2.6. Języki opisu sprzętu

Hardware Description Language, HDL(ang. język opisu sprzętu) oznacza język komputerowy służący do opisu i konfiguracji układów cyfrowych w technice cyfrowej[21]. Jest on przetwarzany przez kompilator(syntezator) Istnieje wiele języków, ale dwa z nich są najpowszechniejsze:

- **Verilog**- na ogół stosuje się go w projektowaniu i weryfikacji układów cyfrowych na poziomie rejestrów przeniesienia z abstrakcji[22]. Poza tym używa się go też w weryfikacji obwodów analogowych i obwodów sygnałów mieszanych, a także w projektowaniu obwodów genetycznych. Verilog składnią przypomina język C
- **VHDL**(Very High Speed Integrated Circuit Hardware Description Language)- przeważnie jest używany do dwóch odmiennych zadań: do symulacji projektów elektronicznych albo do ich syntezy. Synteza oznacza proces, w którym VHDL jest kompilowany i mapowany na technologię implementacji, taką jak FPGA[23]. VHDL składnią przypomina język Pascal.

### 3.2.7. Przykładowe rodzaje RAM-u

Pamięć RAM dzieli się na dwa typy:

- SRAM[24] - Static Random-Access Memory - statyczna pamięć o dostępie swobodnym, typ pamięci półprzewodnikowej zbudowanej przy użyciu sześciu tranzystorów, z czego cztery tworzą przerzutnik, a dwa są tranzystorami sterującymi. Przechowuje dane tak długo, jak dostarczane jest do niej zasilanie.
- DRAM[25] - Dynamic Random-Access Memory - pamięć ulotna rodzaju półprzewodnikowego, która przechowuje dane dzięki kolejnym kondensatorom znajdującym się w układzie scalonym. Poszczególne elementy układu to tranzystory MOS, z których jeden pełni funkcję kondensatora, a drugi steruje. Wymaga ona okresowego odświeżania.

Porównując te dwa typy pamięci, doszliśmy do wniosku, że najbardziej odpowiada nam pamięć rodzaju SRAM. Spowodowane jest to różnicami w szybkości działania, ponieważ szybkość DRAM waha się w okolicach od 60 do 100ns, a dla typu SRAM to tylko 20-40ns. Oba wyniki są zadowalające, ale w naszym przypadku różnica jest znacząca. DRAM ma przewagę nad SRAM jeśli chodzi o pojemność i cenę, lecz tutaj cena nie gra roli, a pojemność jest wystarczająca.[26]

### 3.2.8. CPU

CPU[27] - Central Processing Unit - jest to sekwencyjne urządzenie cyfrowe, które pobiera dane z pamięci operacyjnej, interpretuje je i wykonuje działania z nich pochodzące. Wykonują one ciągi operacji matematyczno-logicznych ze zbioru operacji podstawowych. Z wielu parametrów jakie posiada, podobnie jak w przypadku pamięci, największe znaczenie ma jego szybkość. Ważne jest również IPC - Instructions Per Cycle - ilość instrukcji wykonywanych w czasie jednego cyklu zegara taktującego.

Urządzenie takowe może być potrzebne do interpretowania wyników, które uzyskano z procesora DSP, może zarządzać nim czy wysyłać polecenia, które rozkażą układowi DSP wykonania kolejnego pomiaru. CPU umożliwia poszerzenie możliwości analizy pomiarów odległości itp. Można stworzyć oprogramowanie na ten procesor interpretując stosownie wyniki uzyskane z procesora DSP. Takie oprogramowanie będzie tańsze niż projektowanie nowego układu do analizy wyników układu pomiarowego.

### 3.3. Metody pomiarów odległości za pomocą lasera

Zadaniem zespołu było wykonanie urządzenia do pomiaru odległości za pomocą odbitej wiązki laserowej. Takie urządzenie jest bliskie koncepcji dalmierza laserowego, nasze urządzenie/system będzie więc bardzo w założeniach przypominało takową konstrukcję.

Uznano, że warto rozważyć podczas koncepcjalizacji systemu sprawdzone rozwiązania, dla tego w tej sekcji dokonano analizy rodzajów dalmierzy, która pozwoliła na wybranie takiej koncepcji dalmierza, która najbardziej pasuje do projektu.

Dalmierz takowy musi spełnić pewne wymagania:

- Do wykonania pomiaru musi uwzględniać podczas pomiaru jedynie dane:
  - $c$  - prędkość światła w ośrodku rozchodzącego się wiązki laserowej
  - $t_0, t_p, t$  - czasy biegnięcia sygnału w danych etapach pomiaru
- Musi nadawać się do pomiarów bardzo dużych odległości (ponad 100 000 km)
- Musi być przystosowany do rejestrowania bardzo osłabionych sygnałów
- Musi wykryć odbicie bardzo krótkiego impulsu o czasie  $t_0$  odbitego od obiektu.

#### 3.3.1. Rodzaje dalmierzy

Dalmierze to urządzenia służące do mierzenia odległości pomiędzy nimi a obiektem bez zmiany położenia. Można je podzielić ze względu na długość fali przenoszącej sygnały pomiarowe, będą to dalmierze: **ultradźwiękowe i elektromagnetyczne**[28]. My jesteśmy zainteresowani tymi drugimi ze względu na użycie lasera.

#### 3.3.2. Dalmierz laserowy

Jednym z rodzajów dalmierzy jest dalmierz laserowy. Jest to urządzenie, które pozwala na wyznaczenie odległości, wysyłając promień laserowy. Drogie dalmierze pozwalają uzyskać dużą dokładność pomiarów. Wśród dalmierzy laserowych można wyróżnić dalmierze opierające się o czas przelotu, przesunięcie fazowe fali światła, a także interferometrie.

#### 3.3.3. Dalmierz fazowy

Używając dalmierza fazowego należy wysłać okresowy sygnał o określonej częstotliwości. Zasada działania dalmierza fazowego polega na zbadaniu różnicy fazy między sygnałem wysyłanym a odbieranym. Odległość, używając takiego dalmierza, można wyrazić funkcją o wzorze: [29]

$$D = \frac{c}{2f} * \frac{\varphi}{2\pi}$$

gdzie:

$c$ -prędkość światła

$f$ -częstotliwość sygnału, który wysyłaliśmy

$\varphi$ -przesunięcie fazowe (wyrażone w radianach)

Taki dalmierz nie spełnia naszych zastosowań. Obiekt badany jest o wiele dalej oddalony niż długość światła nadawanego przez laser (6 km). Dalmierz takowy jest bardzo dobry do pomiaru odległości bliskich zaś dla pomiaru odległości "kosmicznych" jest bezużyteczny

Technika stosowana przy dalmierzach fazowych jest dokładna, same dalmierze są używane między innymi przez wojsko.

### 3.3.4. Interferometria

Dalmierze interferometryczne dokonują pomiaru odległości, wykorzystując zjawisko interferencji fal elektromagnetycznych. Są to dalmierze laserowe o największej precyzji, jednak są bardzo drogie i podatne na uszkodzenia.

Niestety, taki dalmierz również nie nadaje się do zadanego pomiaru. Pomiar interferometru nie uwzględnia danych pobieranych przez detektor.

### 3.3.5. Dalmierz impulsowy

Używając dalmierza impulsowego, wysyłamy impuls elektromagnetyczny. Dalmierz oblicza odległość na podstawie czasu, w jaki impuls dotarł do celu i wrócił z powrotem do urządzenia. Odległość, używając takiego dalmierza, można wyrazić funkcją o wzorze:

$$D = \frac{c*t}{2}$$

gdzie:

c-prędkość światła

t-czas od momentu wypuszczenia impulsu do jego powrotu

Taki dalmierz spełnia nasze wymagania i jego koncepcja przyda się podczas konceptualizowania systemu cyfrowego do zadanego pomiaru.

### 3.3.6. Analiza przydatności metod

Po powyższej analizie można dojść do wniosku, że system będzie bardzo wyspecjalizowanym dalmierzem impulsowym, przystosowanym do wykrywania bardzo osłabionych sygnałów. Urządzenie więc będzie odbiegać od koncepcji "klasycznego" dalmierza laserowego, ale podstawy działania oraz obliczania odległości są bardzo podobne w obu urządzeniach. Metoda obliczania odległości pokrywa się z ustaleniami z podpunktu 3.1.1

Dla takiego urządzenia niezbędne będzie opracowanie specjalnej metody interpretacji wyników, która umożliwi "wychwycenie" takich sygnałów. Analiza takowych metod zawarta jest w następnej sekcji.

Stosując w pomiarach odległości dalmierze elektromagnetyczne, musimy uwzględniać wpływ błędu przypadkowych i systematycznych, które obciążają wyniki tych pomiarów. Ogólnie można powiedzieć, że błędy te związane są z samym dalmierzem oraz z wpływem środowiska na sygnał pomiarowy.

### 3.4. Metody interpretacji wyników

#### 3.4.1. Podstawowe dane na temat odebranego sygnału przez odbiornik

Po wybraniu metody nadawania i rejestracji należało znaleźć metodę interpretacji sygnału odebranego przez rejestrator. Warto więc przeanalizować, jaki rodzaj danych może być rejestrowany przez rejestrator założony w zadaniu

Istotne jest to, że wartości sygnału są wyrażone w 8 bitach, co nie jest bardzo precyzyjne. Zaważy to na metodzie interpretacji wyniku; sygnał jest wyraźnie zaszumiony, dlatego "wydobycie" potrzebnych informacji będzie trudne oraz będzie wymagało specjalnych metod, które mogą dawać niejednoznaczne wyniki.

Sygnał był rejestrowany z częstotliwością 1 MHz, czyli analiza sygnału musi być bardzo szybka. Niezbędnym będzie poszukanie specjalistycznego algorytmu umożliwiającego w bardzo krótkim czasie wykonanie koniecznych przekształceń sygnału itp. w celu uzyskania przydatnych danych do stwierdzenia, w którym momencie sygnał wrócił do układu pomiarowego.[30]

#### 3.4.2. Metody wykrycia momentu dotarcia impulsu do odbiornika

Głównym celem analizy sygnału jest wykrycie momentu, w którym odbity od odbiektu mierzonego sygnał dotrze do detektora. W tym celu należy podzielić sygnał składający się z 5000 zarejestrowanych próbek na mniejsze fragmenty tak, aby analizować drobniejsze partie danych oraz wychwycić tę partię danych, w której znajdzie się najbardziej interesująca zmiana w sygnale.

Sygnały można podzielić na wiele sposobów. Przykładowe pomysły stworzone przez zespół:

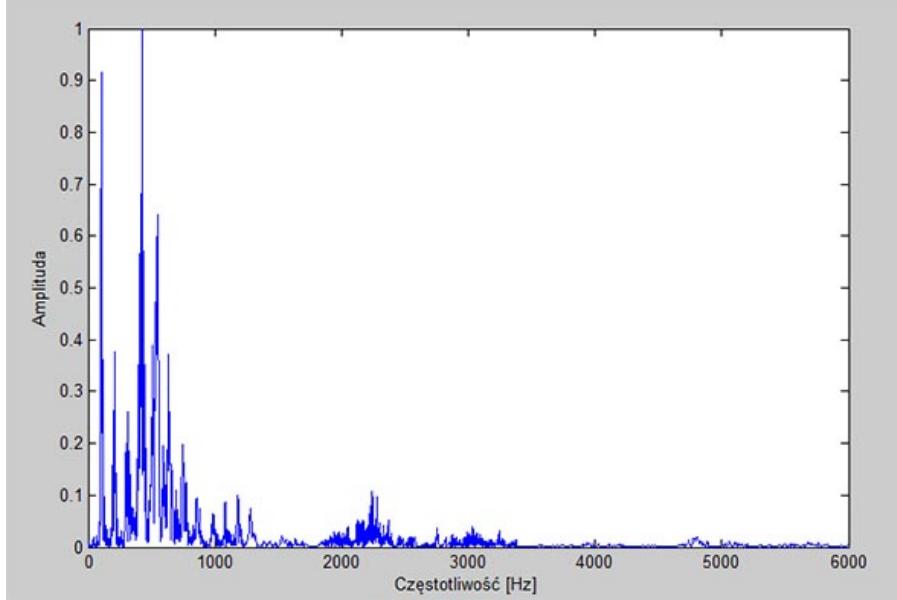
Założymy pobrane próbki są ponumerowane od najstarszej do najmłodszej od 0 do 4999

- Pobieranie fragmentów składających się z fragmentów zaczynających się z elementów o numerach od  $n * 20$  do  $(n + 1) * 20$  (mnożymy przez 20, gdyż sygnał, który został wysłany przez laser, ma częstotliwość 50 kHz dla częstotliwości rejestrowania przez detektor wynoszącą 1 MHz), dzieląc niejako sygnał na "plasterki". Gdy zmiana zostanie wykryta w danym przedziale, oznacza to, że sygnał dotarł do detektora w przedziale  $n$ -ów  $([n * 20; (n + 1) * 20 - 1])$ . Założymy  $n \in [0, 250]$
- Pobieranie fragmentów składających się z fragmentów zaczynających się z elementów o numerach od  $n$  do  $n + 20$  (gdyż sygnał, który został wysłany przez laser, ma częstotliwość 50 kHz dla częstotliwości rejestrowania przez detektor wynoszącą 1 MHz). Założymy  $n \in [0, 5000 - 20] = [0, 4979]$ . Ta metoda w teorii daje bardziej precyzyjne wyznaczenie odległości, gdyż zmianę stwierdzamy dla konkretnego  $n$ -a, a nie dla przedziału  $n$ -ów

Następnie należało znaleźć metody, które umożliwiają stwierdzenie interesujących zmian dla danego fragmentu sygnału. Trzeba podkreślić, że sygnał rejestrowany przez odbiornik jest wysoce zaszumiony, stąd należy szukać metody, która wykryje zmiany związane z sygnałem emitowanym przez laser w obliczu licznych zakłóceń. Do "wydobycia" składowych sinuśnid (gdyż sygnał wyemitowany przez laser jest sinusem) przydatna jest analiza widmowa sygnału.

### 3.4.3. Analiza widma

Analiza widma pozwala na podstawie sygnału stwierdzić, z jakich składowych sygnałów sinusoidalnych dany sygnał się składa. Dzięki tej metodzie można wykryć składowe częstotliwości, a więc pozwala odnaleźć czy sygnał składa się z konkretnej danej częstotliwości. Gdy poziom szukanej częstotliwości  $f = 50$  kHz skoczy, oznacza to, że dotarł do odbiornika szukany sygnał, na podstawie którego będzie można dokonać pomiaru [31]



Przykładowy wynik analizy widmowej [32]

### 3.4.4. Transformata Fouriera

Do wyszukania częstotliwości  $f = 50$  kHz w zaszumionym sygnale potrzebujemy transformaty Fouriera, aby uzyskać rządane widmo.

Transformata Fouriera pozwala zamienić sygnał z  $f(t)$  z dziedziny czasu na dziedzinę częstotliwości  $F(\omega)$ , gdzie  $\omega$  jest łatwo zamienialne na częstotliwość za pomocą wzoru:

$$\omega = 2\pi f$$

zatem:

$$\frac{\omega}{2\pi} = f$$

W ten sposób można "nasłuchiwać" skoków poziomu dla danej częstotliwości.

Przekształcenie to odbywa się za pomocą formuły: [33]

$$G(f) = \int_{-\infty}^{\infty} g(t) e^{-2\pi jft} dt$$

Potrzebna jest jednak uproszczona forma tego twierdzenia do realizacji w systemie cyfrowym. Precyzyjne wyliczanie transformaty Fouriera jest bezcelowe dla tego systemu. System ten ma odnaleźć skok poziomu dla częstotliwości sygnału nadawanego przez odbiornik, a nie wyznaczyć prezycyjną matematycznie transformaty Fouriera. Poza tym takowe obliczanie wymagałoby bardzo rozbudowanego i finalnie niewydajnego układu procesora DSP.

Zatem konieczny jest jednak algorytm wyspecyfikowany do szybkiego wykonania tejże analizy dla przypadku danych nieokreślonych konkretnym wzorem, która da szybkie efekty. Do tego przydadzą się algorytmy szybkiej transformaty Fouriera (FFT), lub jego pochodne, które zostaną opisane w punkcie poniższym. [34]

### 3.4.5. Algorytm FFT i jego odmiany

Do wykonania układu może być potrzebny algorytm wykonujący szybko transformatę Fouriera, który umożliwia szybkie analizowanie sygnału wejściowego. Umożliwia on wydajne i szybkie wykonanie transformaty Fouriera, którą da się zaimplementować w układzie cyfrowym DSP. [34]

Są różne rodzaje algorytmów FFT:

- - Cooley'a – Tukey'ego (DFT) - algorytm szybkiej transformacji Fouriera (FFT). Wyraża dyskretną transformację Fouriera (DFT) o dowolnej złożonej wielkości  $N = N_1 N_2 N = N_1 N_2$  w członach mniejszych DFT wielkości  $N_1 N_1 N_2, N_2$ , rekurencyjnie, w celu ograniczenia czasu obliczeń do  $O(N \log N), O(N \log N)$ , szczególnie w przypadku  $N$  będącego liczbą wysoce złożoną (liczbą gładką)[35]
- IFFT to szybki algorytm do wykonywania odwrotnej (lub wstępnej) transformaty Fouriera (IDFT), która cofa proces DFT. IDFT sekwencji  $F_n$ ,[36]
- RFFT - algorytm szybkiej transformaty Fouriera ignorujący wartości zespolone sygnałów (nie istnieją dla niego częstotliwości ujemne itd.) stąd pierwsza litera R czyli Real input [37]

Poniżej opisano szybką transformatę Fouriera Cooley'a – Tukey'ego

Oto pseudokod algorytmu FFT :

DFT dla danych w postaci  $(x_0, x_s, x_{2s}, \dots, x_{(N-1)s})$ :

```
X0, ..., N - 1 = ditfft2 (x, N, s)

ditfft2 (x, N, s):
if N = 1 then
    X0 = x0 \\ przypadek bazowy trywialnego rozmiaru DFT = 1
else
    X0, ..., N/2-1 = ditfft2 (x, N/2, 2s)
    XN/2, ..., N-1 = ditfft2 (x+s, N/2, 2s)

    for k = 0 to N/2 - 1 do \\ laczenie DFT dwoch polowek
        do pelnego DFT
            pi = 3,14
            t = Xk
            Xk = t+exp(-2*pi/N k) Xk+N/2
            Xk+N/2 = t-exp(-2*pi*i/N k) Xk+N/2

    end for

end if
```

Algorytmy te są dostępne w pythonowej bibliotece `scipy.py`, stąd łatwo było prototypować rozwiązania z algorytmami tego typu. [38]

### 3.5. Poszukiwanie prostszej metody analizy

Analiza FFT mimo tego, że jest skuteczna, dla zastosowań tego projektu jest zbyt wyszukana, gdyż do określenia odległości nie potrzeba tylu danych ile uzyskujemy dzięki tejże analizie. Postanowiono poszukać metody więc prostszej i łatwiejszej w implementacji a wypadkowo szybszej, która będzie mogła pracować z większą częstotliwością.

### 3.5.1. Korelacja

Korelacja jest miarą podobieństwa lub wzajemnej zależności. Korelacja sygnałów x i y oznacza, że wzrostowi x towarzyszy wzrost y. Jeśli przeważa sytuacja odwrotna mówimy o korelacji ujemnej.

Korelację oblicza się jako sumę wartości obu sygnałów przemnożonych przez siebie. Analiza korelacyjna ma ważne znaczenie w diagnostyce technicznej, w teorii przetwarzania i przesyłu sygnałów. Stosując analizę korelacyjną można z mieszaniny sygnału informacyjnego i zakłóceń wydzielić sygnał synchronizacji niezbędny do odbioru przesłanej informacji. Analizę korelacyjną stosuje się w teorii modulacji do skutecznego kodowania informacji. Funkcja autokorelacji służy do określenia dynamiki zmian sygnału, do wykrywania sygnałów okresowych z „zaszumionych” sygnałów pomiarowych co jest niezwykle ważne przy obróbce sygnałów diagnostycznych.

## 4. Koncepcja

### 4.1. Wybór metody : Korelacja

#### 4.1.1. Zebranie metod

Podczas etapu III zebrano metody analizy sygnału w celu rozstrzygnięcia która z nich daje najlepsze efekty zarówno pod względem wydajnościowym jak i prostoty układu. Wybrano dwie metody:

- Analiza FFT
- Metoda korelacji

W tym rozdziale postanowiono rozszerzyć, która z tych metod jest najlepsza dla tego projektu oraz przedstawić wstępne rozwiązanie.

Aby tego dokonać zebrano wady i zalety obu metod i analizowano, które z wad są łatwiejsze a które gorsze do pogodzenia, a które zalety są bardziej kuszące pod względem wyboru metody analizy.

#### 4.1.2. Zalety/wady metody korelacji

Lp	Wady	Zalety
1	Wymaga „generowania” wzorca sygnału	Bardzo efektywna
2		Prosta w implementacji
4		Prosty algorytm
3		Potencjalne wysokie taktowanie układu
3		
5		Niski koszt wykonania układu

#### 4.1.3. Zalety/wady metody FFT

Lp	Wady	Zalety
1	Skomplikowana w implementacji	Odwzorowanie w teorii sygnałów (transformata Fouriera)
2	Powolna	Precyzyjna
4	Jest zbyt precyzyjna jak na nasze potrzeby: nie potrzebuje szczegółów my aż tylu do pomiaru	Jak na wykonywane obliczenia bardzo efektywny algorytm (DFT)
3	Wyższy koszt wykonania układu	
5	Potencjalnie niższa maskymalna częstotliwość taktowania układu	

#### 4.1.4. Ostateczna decyzja

Po szczegółowym przemyśleniu metod analizy sygnału wybrano korelację. Metoda ta jest najprostsza w implementacji z wcześniej analizowanych. Układ który będzie realizował tą metodę będzie o wiele prostszy niż analizator widma. Dodatkowo będzie mógł pracować z większą częstotliwością i przez mniejszą zawiłość będzie bardziej odporny na wady konstrukcyjne (co jest istotne dla początkujących projektantów).

Te ustalenia pozwoliły zespołowi przejść do dalszego etapu projektowania czyli opracowania koncepcji takiego systemu.

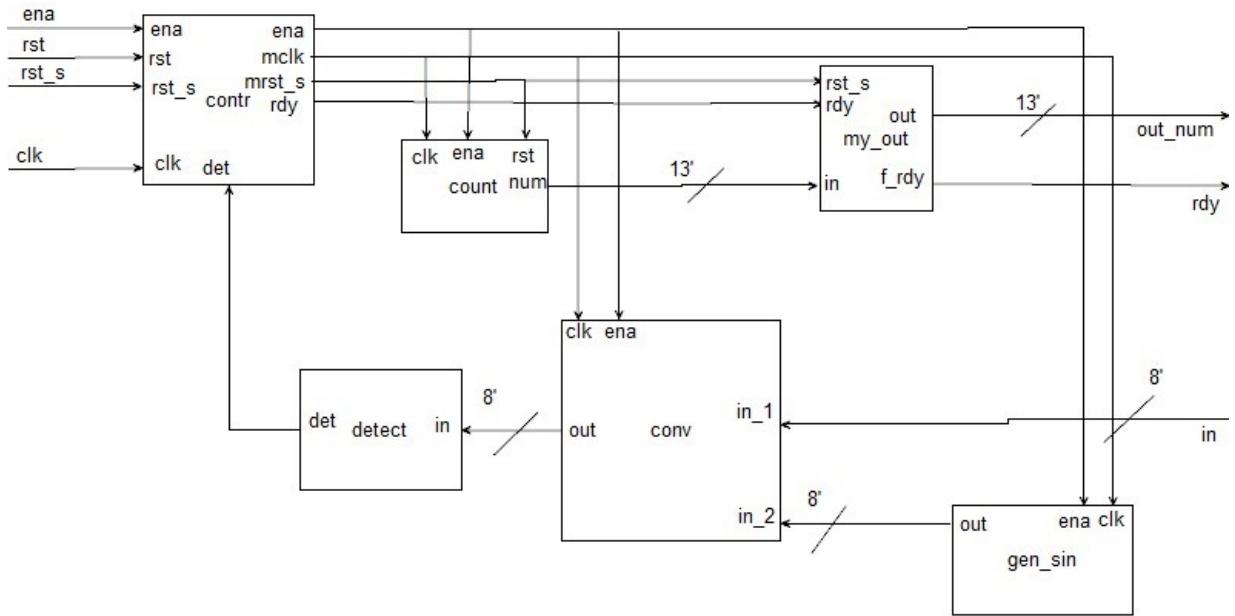
### 4.2. Budowa systemu

Etap III obejmuje opracowanie blokowej budowy systemu wykrywającego odległość od danego obiektu. Dzięki ustaleniom z Etapu II, uzyskanej wówczas wiedzy, można było wstępnie opracować budowę systemu z bloków funkcjonalnych. W kolejnych etapach będzie rozważana ich bardziej szczegółowa budowa wraz z implementacją w spręciu (układ FPGA).

#### 4.2.1. Schemat

Zgodnie z koncepcją projektowania systemów cyfowych w ramach syntezy strukturalnej (data-flow), postanowiono sporządzić schemat systemu, który wstępnie przybliża działanie właściwego systemu pomiarowego.

Schemat poniżej:



Schemat ogólny budowy procesora DSP służącego do pomiaru odległości

Ważnym założeniem jest to, że uznano, że przeliczanie z czasu na odległość może być wykonyane już przez program procesora.

#### 4.2.2. Objaśnienie wstępne działania elementów

Kluczem do zrozumienia działania systemu, jest objaśnienie działania poszczególnych elementów układu, tak aby ich rola w układzie była czytelna.

##### 1. *contr* (Kontroler)

Kontroler odpowiada za sterowanie poszczególnymi elementami systemu pomiarowego. Zarządza rozdzieleniem zegara, przekazuje od np procesora proźby o restart układu, blokuje układ itp. Pośredniczy w operacji przekazania informacji o wykryciu odbitego światła, przekazując informacje z modułu *detect* do *my\_out*, przy okazji wyłączając (*ena* = 0) układ. Dzięki temu informacja z układu trafia do procesora

##### 2. *count* (Licznik próbek)

Odpowiada za przeliczanie ilości próbek przeanalizowanych zanim dotrze sygnał *rst* (resest synchroniczny) zerujący licznik. Dzięki niemu jest możliwe odliczenie czasu który mija od włączenia rejestracji aż do dotarcia do modułu "my\_out" sygnału *rdy*, który zatrząskuje wartość, przez co dzięki sygnałowi "out\_num" jest widoczna dla np procesora. Wartość ta jest kluczowa dla szacowania odległości od detektora do obiektu

Sygnały licznika są 13-bitowe gdyż analizujemy próbki w przedziale 0-5000, liczby 13-bitowe umożliwiają przetwarzanie liczb w zakresie 0-8191.

### 3. *my\_out* (Wyjście układu)

Umożliwia komunikację między układem a otoczeniem zatrzaskując wartość licznika w odpowiednim momencie (gdy dotrze do niego sygnał rdy) umożliwiając procesorowi itp. dalszą analizę oszacowanych danych. Przekazuje również dalej sygnał rdy informujący o zakończeniu pomiaru

### 4. *detect* (Odpowiada za detekcję momentu dotarcia do odbiornika)

Odpowiada za detekcję w sygnałe przetworzonegm przez moduł "conv", który świadczy o dotarciu światła do odbiornika. Moduł ten wtedy wysyła sygnał do kontrolera (out) aby rozpocząć proces zakończenia pomiaru( zatrzaśnięcie wartości licznika układu itd.)

### 5. *conv* (Element odpowiadający za realizację korelacji dwóch sygnałów)

Element ten odpowiada za wykonanie operacji korelacji dwóch sygnałów wejściowych wyrażonych w 8-bitach (zmodulowanego sinusa i sygnału odbieranego przez odbiornik). Moduł jest kontrolowany przez zegari sygnał ena, pochodzących z modułu kontrolnego.

### 6. *gen\_sin* (Element odpowiadający za generowanie sygnału sinusoidalnego do obiektów)

Moduł ten odpowiada za emulowanie sygnału zmodulowanego sinusa potrzebnego do obliczeń modułu "conv". Moduł jest kontrolowany przez zegari sygnał ena, pochodzących z modułu kontrolnego.

## 4.3. Przykład ilustrujący działanie systemu

Aby zaprezentować działanie systemu, posłużono się przykładem opisany poniżej

### 4.3.1. Pomiar Ziemia - Księżyca

W trakcie pomiaru Ziemia-Księżyca sygnał wysłany z Ziemi odbija się od księżyca i wraca zniekształcony na ziemie

### 4.3.2. Przepływ informacji podczas pomiaru (procedura pomiarowa)

Podczas takiego pomiaru, przepływ informacji w naszym systemie zaczyna się od dotarcia sygnału do modułu conv. Następnie trafia do modułu detect który wysyła go do kontrolera (moduł contr) w celu rozpoczęcia procesu zakończenia pomiaru, z kontrolera trafia do *my\_out*.

### 4.3.3. Wnioski jakie można wyciągnąć z takiego pomiaru

Dzięki uzyskanemu numerowi próbki można oszacować odległość Ziemia-Księżyca. Skoro sygnał był próbkowany z cz.  $f = 1 \text{ MHz}$  oznacza to że jedna próbka trwała  $1 * 10^6 \text{ s}$ . Starczy przez ten czas przemnożyć numer próbki aby określić odległość (Ustalenia z etapu II) : Odległość, używając takiego dalmierza, można wyrazić funkcją o wzorze:

$$D = \frac{c*t}{2}$$

gdzie:

c-prędkość światła

t-czas od momentu wypuszczenia impulsu do jego powrotu

Obliczenia te powinno przekazać już procesorowi aby nie powodować dodatkowych niepotrzebnych opóźnień układowi.

#### 4.4. Algorytm

Dzięki ustaleniom z etapu II można było przejść do opracowania algorytmu systemu (opisu funkcjonalnego), który umożliwi przetworzenie sygnału na informację o odległości. Algorytm jest jedynie opisem zachowania, nie uwzględnia fizycznej budowy układu, zatem jego założenia mogą odbiegać od założeń układu fizycznego.

##### 4.4.1. Pseudokod

Poniżej znajduje się pseudokod przedstawiający zachowanie systemu:

```
pobrane = pobierz dane(z_wysjcia)
wyslane = pobierz dane(sinus_50kHz)

odpowiedz = double [z_wysjcia]

for n : n < dlugosc_z(pobrane)
    suma -> 0;

    for k : k > n-dlugosc_z(sent) , k <= n:
        k_bezwzględne -> k - (n-dlugosc_z(sent))

        jeżeli k >= 0:
            suma += odebrany[k] * wyslany[k_bezwzględne]

    odp[n] -> (suma)

n -> 0;
for probka z odp:
    if wartosc_bezwzględna(probka) > wartosc_minimalna:
        wyslij_rdy()
        break
    n -> n + 1
```

#### 4.5. Symulacja w środowisku Python

Aby zweryfikować konstrukcje systemu należało wykonać testy modelu referencyjnego. Użyto do tego celu środowiska Python zarówno implementując korelację sprzętowo jak i algorytmem:

##### 4.5.1. Model referencyjny

Środowisko PyCharm służy zespołowi do testowania systemu, którego zachowanie zakodowano w języku Python. Taki program, model referencyjny, umożliwia zaprojektowanie systemu zanim przejdzie się do projektowania sprzętu (projektowanie sprzętu w sposób algorytmiczny jest niefektywne, a projektowanie układów jest nienaturalne dla człowieka stąd potrzebne są

narzędzia ułatwiające ten proces).

Model referencyjny zamiast rzeczywistych sygnałów będzie przetwarzał ich reprezentacje, pliki tekstowe będące ciągami skwantownych sygnałów. Wynik takowej analizy też nie zostanie wysłany w modelu testowym jako sygnał ready, a wyświetli się jako komunikat informującym o dotarciu sygnału oraz o numerze próbki, w której dotarł sygnał.

Takowy model można zamieniać np. za pomocą analizy RTL itp. do formy w pełni wydajnego wyspecjalizowanych elementów układu scalonego. Program (będący modelem referencyjnym) zostanie opracowany na podstawie wcześniejszych opracowanych algorytmów.

#### 4.5.2. Wykonanie z użyciem biblioteki numpy

Model referencyjny wykonano w środowisku Python. Pierwsze podejście wykonano z użyciem biblioteki *numpy* zawartej w tymże środowisku, aby wykonać przez nią operację korelacji (*numpy.convolve*). Dane wyjściowe pobrano z plików z zidentyfikowanym sygnałem, dołączonych do zadania projektowego przez prowadzącego przedmiot. Detektor sygnału w tym algorytmie został ustawiony referencyjnie na 2.5 aby przetestować algorytmie

Poniżej znajduje się kod takowego rozwiązania:

```
import io
import math

import matplotlib.pyplot as plt
import numpy as np

def pobierz dane(plik):
    file = open(plik, "r")
    dane = []

    for elementy in file:
        try:
            liczba = float(elementy)
            dane.append(liczba)
        except:
            print(" ", end="")

    file.close()
    return dane

received = pobierz dane("received8.txt")
sent = pobierz dane("sent8.txt")

korelacja = np.convolve(received, sent)

fig = plt.figure()
```

```

plt.plot(korelacja)

plt.show()

n = 0;
for probka in korelacja:
    if math.fabs(probka) > 2.5:
        print("Zarejestrowano sygnał w próbce nr "+str(n))
        break
    n= n + 1

```

Następnym krokiem była zamiana funkcji (*numpy.conclove*) na algorytm korelacji sporządzony własnoręcznie

#### 4.5.3. Wykonanie z bez użycia bibliotek

Poniżej zawarto kod sporządzony na podstawie wcześniej opracowanych schematów blokowych zamieniając funkcje *numpy.conclove* na algorytm

```

import io
import matplotlib.pyplot as plt
import math

def pobierz dane(plik):
    file = open(plik, "r")
    dane = []

    for elementy in file:
        try:
            liczba = float(elementy)
            dane.append(liczba)
        except:
            print(" ", end="")

    file.close()
    return dane

received = pobierz dane("received8.txt")
sent = pobierz dane("sent8.txt")

odp = []

for n in range(len(received)):
    suma = 0;

    for k in range(n-len(sent),n):
        k_bezwzgledne = k - (n-len(sent))

        if k >= 0:

```

```

suma += received[k] * sent[k_bezwzgledne]

odp.append(suma)

fig = plt.figure()

plt.plot(odp)

plt.show()

n = 0;
for probka in odp:
    if math.fabs(probka) > 2.5:
        print("Zarejestrowano sygnał w probce nr "+str(n))
        break
    n= n + 1

```

Dzięki takowemu programowi można było dokonać testów modelu referencyjnego. Kody źródłowe tych programów znajdują się w repozytorium git projektu na portalu GitHub [39]

## 4.6. Oczekiwania wobec przeprowadzonej analizy

### 4.6.1. Dane do testowania

Do testowania użyto danych dołączonych do zadania projektowego: 2 pliki tekstowe z sygnałem skwantyzowanym do 8 bitów. Są to:

- *received8.txt* - plik zawierający przetworzony sygnał odebrany przez detektor
- *sent8.txt* - plik zawierający przetworzony sygnał "nadany" przez laser

Oba te pliki znajdują się w zadaniu projektowym jak i w repozytorium projektu. Programy (będące modelami referencyjnymi) są przystosowane do zamiany takiej postaci do formy użytecznej dla przetwarzania algorytmicznego. Niemniej dane te trzeba będzie później podczas opracowywania testbench-a przetworzyć do formy zrozumiałej dla systemu programującego układy FPGA, aby dało się testować rozwiązanie sprzętowe.

Poniżej opisano scenariusze testowe, które trzeba przewidzieć aby usprawnić proces usprawniania rozwiązania projektowego:

### 4.6.2. Scenariusze testowe

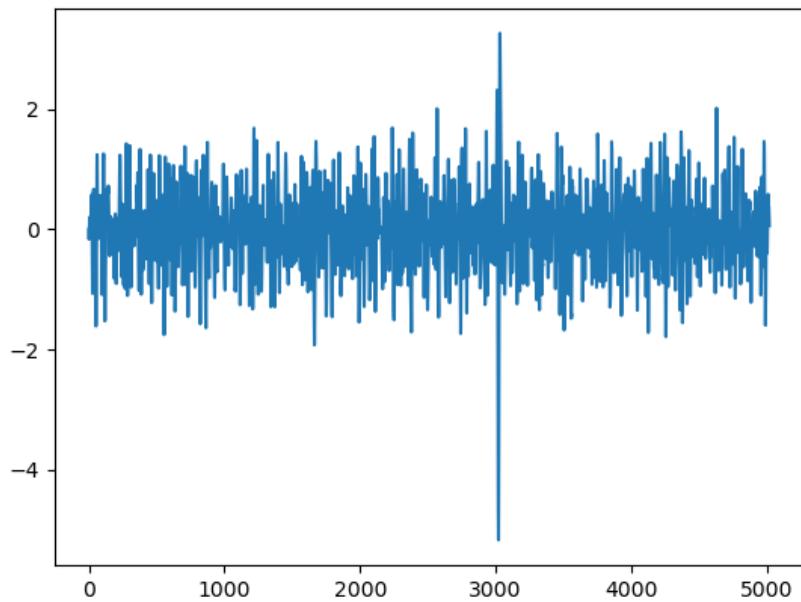
W celu sprawdzenia czy nasz system działa poprawnie, możemy zmierzyć odległość między ziemią a ISS, wysyłamy sygnał o znanych parametrach, który odbija się od ISS i wraca do nas. Nasz system analizuje sygnał, oczekujemy że dzięki tej analizie jesteśmy w stanie zmierzyć odległość między ziemią a ISS. Jeśli będzie bliska prawdziwej odległości ziemi od ISS, test zakończy się sukcesem.

### 4.6.3. Wyniki testów modelu referencyjnego

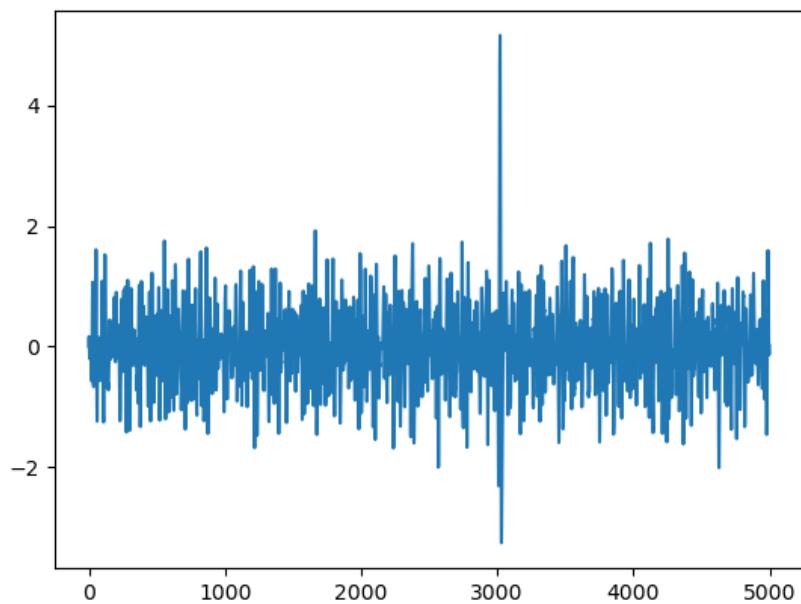
Po wykonaniu testów modelu referencyjnego w środowisku Python przetworzony sygnał zarówno przez program stosujący rozwiązań biblioteczne jak i stosujący algorytm jednoznacznie wskazały miejsce (w czasie), w którym sygnał (zmodulowane światło lasera) wrócił do odbiornika. Jest to wzmacnianie sygnału widoczne jako dużo większy niż średnia "peak" w miejscu

ok 3000 próbki (dokładnie 3017).

Poniżej znajdują się wykresy sygnału po przetworzeniu operacją korelacji. Na podstawie tych wykresów można uznać, że detektor sygnału znajdujący się w module może być czuły po przekroczeniu wartości bezwzględnej 2.5 tak jak w programach będących modelami referencyjnymi.



Wykres przedstawiający przebieg sygnału dla rozwiązania stosującego bibliotekę *numpy* do obliczeń korelacji [39]



Wykres przedstawiający przebieg sygnału dla rozwiązania stosującego własnoręcznie napisany program do obliczeń korelacji [39]

Różnice wynikają z nieco innych algorytmów przetwarzanych do korelacji, nadal jednak wykres 1 jest odbiciem symetrycznym od osi OX, i nadal potwierdzają tą samą odległość od obiektu.

#### 4.6.4. Wnioski

Metoda zaproponowana przez zespół oraz system mają duże szanse powodzenia podczas pomiaru i implementacji w sprzęcie. Względna prostota budowy oraz jednoznaczność wyników pozwaliło zespołowi na przejście do etapu implementacji w sprzęcie rozwiązania projektu, co zostanie wykonane w etapie IV.

### 5. Implementacja

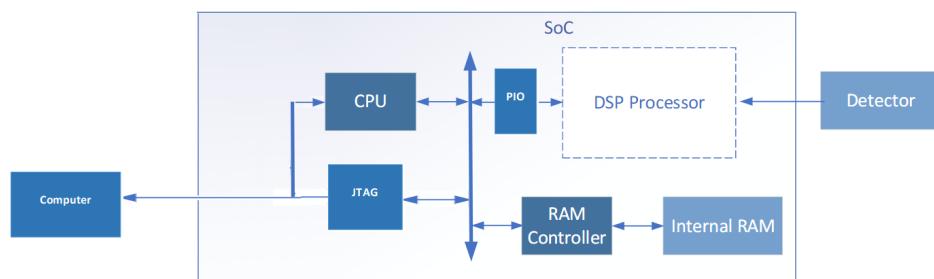
**Uwaga!** Z powodu problemów technicznych na termin etapu IV wykonano jedynie procesor DSP. Na GitHubie [39] znajduje się również projekt podłączony do procesora, jednak program Eclipse nie mógł wykonać projektu, z powodu licznych niuansów technicznych, których rozwiązanie nie było możliwe przed terminem. Moduł procesora zostanie wykonany na etap V i przetestowany w ramach uruchomienia.

#### 5.1. Sposób realizacji systemu

Podczas opracowywania systemu cyfrowego realizującego pomiar odległości należało opracować ostateczną architekturę systemu:

##### 5.1.1. Ostateczna architektura systemu

Na potrzeby realizacji projektu zaprojektowano poniższą architekturę systemu:



Schemat przedstawiający ogólną architekturę systemu [39]

Jako elementów będących blokami użyto dobranych dzięki narzędziu Platform Designer gotowych elementów soft IP Core. Skróciło to czas projektowania układu do minimum, gdyż bez tego potrzebne byłoby opisanie za pomocą języka Verilog od zera procesora RAM-u układów PIO i JTAG, które przetwarzaliby skutecznie dane wygenerowane przez DSP Procesor, prekazywałyby dane do komputera w zrozumiałym dla niego sposób oraz byłyby na tyle efektywne aby opłacało się dodawać je do sprzętu

Lista modeli dobranych elementów:

- **CPU** Nios II/e - wersja darmowa procesora soft-core Nios

- **JTAG JTAG** UART Intel FPGA IP - moduł odpowiadający za komunikację procesora/układu z komputerem. Dzięki niemu zostaną przekazane dane o odległości przeliczone przez procesor do komputera oraz wyświetcone np. w konsoli
- **PIO PIO** (Parallel I/O) Intel FPGA IP - odpowiada za komunikację procesora CPU z DSP, przekazuje dane o odległości (numer próbki w której wykryte jest odbicie) wraz z sygnałem rdy, tak aby procesor przeliczył te dane na odległość w kilometrach oraz mógł wykonywać inne operacje na tej wielkości. Przekazuje on 15-bitową informację o czasie dotarcia próbki (pierwszy bit sygnał rdy sygnaлизujący dotarcie sygnału i pozostałe 14 bitów na numer próbki) do procesora.
- **RAM Controller oraz Internal RAM** On-Chip Memory (RAM or ROM) Intel FPGA IP - w układzie zaimplementowano 16 kB RAM na potrzeby obliczeń procesora CPU
- **DSP Processor** - układ opracowany przez zespół projektowy

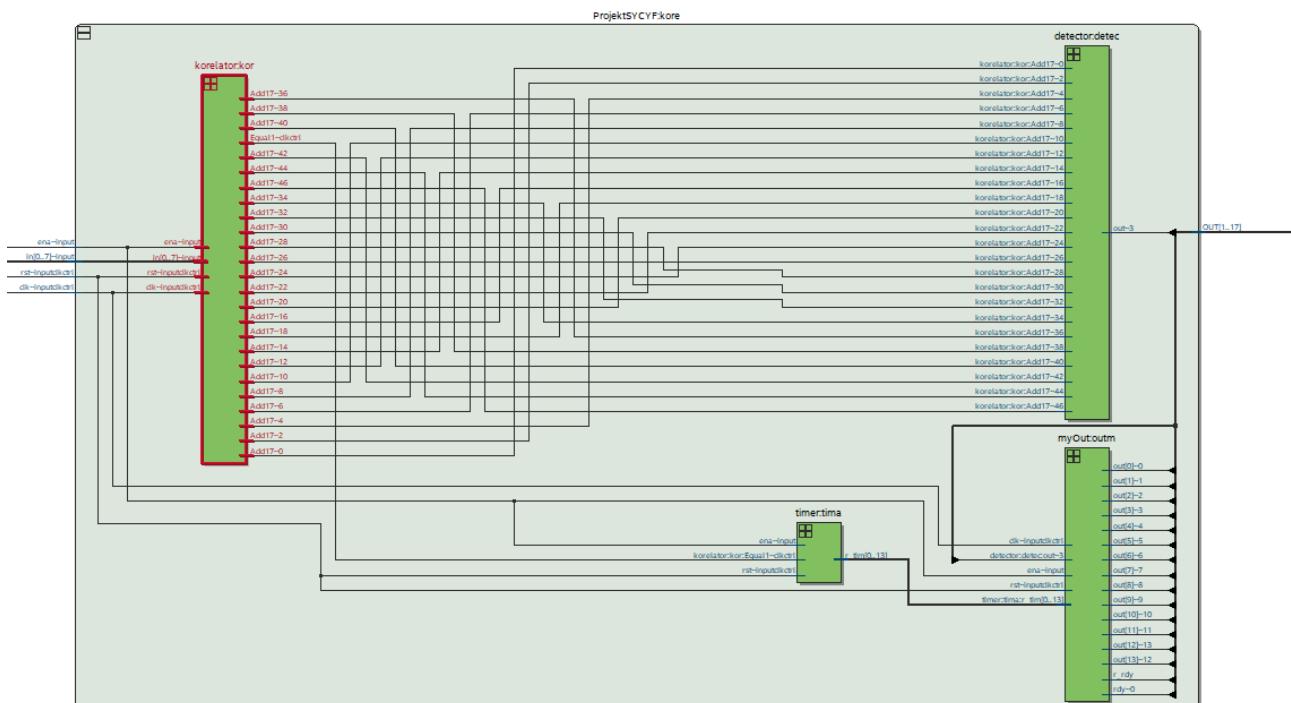
Układ jest taktowany zegarem 50 Mhz oraz wewnętrznie pewne elementy układu DSP Procesor zegarem 1 MHz.

### 5.1.2. Moduły

W tej sekcji opisano budowę procesora DSP - układu który został od zera zaprojektowany na potrzeby skrócenia potencjalnie dużych obliczeń korelacji, które mogłyby uczynić procesor taktowany jedynie zegarem 50 Mhz dla analizy sygnału odbieranego, próbkowanego częstotliwością 1 Mhz, bezużytecznym

Architekturę tego procesora uproszczono względem pierwotnej koncepcji, gdyż okazało się, że pewne elementy niepotrzebnie spowalniały ten układ (w nim zespołowi zależało na prędkości a nie na precyzyjne spełnienie wstępnej koncepcji)

Poniżej znajduje się schemat architektury procesora DSP, wykonującego procedurę korelacji



Schemat przedstawiający architekturę procesora DSP wykonany przez moduł pakietu Quartus Technology Map Viewer [39]

Układ składa się z 3 głównych bloków:

- Korelator - układ odpowiadający za wykonanie korelacji sygnału wejściowego z sygnałem sinusa o okresie  $t_p = 5$  ms oraz przekazujący wynik do modułu *detekor*
- MyOut - odpowiada za zatrzaśnięcie wyniku korelacji i przekazanie informacji do modułu PIO systemu o czasie dotarcia sygnału
- detektor - sprawdza wcześniej określone warunki dotarcia sygnału i w razie ich spełnienia przekazuje tą informację za pomocą sygnału *detect* aby moduł MyOut zatrzasnął wartość timera
- timer - który odlicza czas od rozpoczęcia rejestracji przez układ DSP, przekazuje tą informację do modułu *MyOut* aby w przypadku wykrycia sygnału czas ten został zatrzaśnięty

Kod układu zewnętrznego z połączonymi modułów znajduje się poniżej:

```
module ProjektSYCYF(
    input ena ,
    input rst ,
    input clk ,
    input signed [7:0] rec ,
    output signed [14:0] tim ,
    output rdy
);

    wire clk0 ;
    wire signed [23:0] lout ;
    wire signed [13:0] ltim ;
    wire detect ;

    korelator kor (.ena(ena) ,.rst(rst) ,.clk(clk) ,.rec(rec) ,.out(lout)
    ,.clk0(clk0));
    timer tima (.ena(ena) ,.rst(rst) ,.clk(clk0) ,.tim(ltim));

    detector detec (.in(lout) ,.out(detect));

    myOut outm (.ena(ena) ,.rst(rst) ,.clk(clk) ,.in(ltim) ,.detect(detect)
    ,.rdy(rdy) ,.out(tim));

endmodule
```

Sygnal *ena* może służyć np dla detektora aby on włączał podczas pracy ten układ. Na razie założono, że ten sygnał będzie uruchamiany za pomocą switcha na płytce FPGA.

### 5.1.3. Korelator

Sercem układu DSP jest układ wykonujący kluczową operację - korelację. Oto kod tego układu. Moduł ten jest połączony z podmodułem - rejestrem przesuwającym, aby imitować okno dla tej procedury układu DSP.

Układ dodatkowo składa się z logiki kombinacyjnej obliczającej operacje DSP (mnożenia i dodawania), na podstawie wartości zatrzaśniętych przez rejestr. Układ ma jednoczesny dostęp do każdej wartości zatrzaśniętej w tym rejestrze.

Dzięki temu na wyjście może być wypuścić na wyjściu 24-bitową korelację z 8-bitowego sygnału. Sygnał ten trafia do układu detektoea który ”podejmuje decyzje” o zatrzaśnięciu wartości

Kod układu znajduje się poniżej :

```
module korelator(
    input ena ,
    input rst ,
    input clk ,
    input signed [7:0] rec ,
    output signed [24:0] out ,
    output clk0
);

wire clk_0 ;
wire [4:0] tim ;

wire signed [7:0] out0 ;
wire signed [7:0] out1 ;
wire signed [7:0] out2 ;
wire signed [7:0] out3 ;
wire signed [7:0] out4 ;
wire signed [7:0] out5 ;
wire signed [7:0] out6 ;
wire signed [7:0] out7 ;
wire signed [7:0] out8 ;
wire signed [7:0] out9 ;
wire signed [7:0] out10 ;
wire signed [7:0] out11 ;
wire signed [7:0] out12 ;
wire signed [7:0] out13 ;
wire signed [7:0] out14 ;
wire signed [7:0] out15 ;
wire signed [7:0] out16 ;
wire signed [7:0] out17 ;
wire signed [7:0] out18 ;
wire signed [7:0] out19 ;

wire signed [7:0] sin0 = $signed(8'b00000000);
wire signed [7:0] sin1 = $signed(8'b00100111);
wire signed [7:0] sin2 = $signed(8'b01001011);
```

```

wire signed [7:0] sin3 = $signed(8'b01100111);
wire signed [7:0] sin4 = $signed(8'b01111001);
wire signed [7:0] sin5 = $signed(8'b01111111);
wire signed [7:0] sin6 = $signed(8'b01111001);
wire signed [7:0] sin7 = $signed(8'b01100111);
wire signed [7:0] sin8 = $signed(8'b01001011);
wire signed [7:0] sin9 = $signed(8'b00100111);
wire signed [7:0] sin10 = $signed(8'b00000000);
wire signed [7:0] sin11 = $signed(8'b11011001);
wire signed [7:0] sin12 = $signed(8'b10110101);
wire signed [7:0] sin13 = $signed(8'b10011001);
wire signed [7:0] sin14 = $signed(8'b10000111);
wire signed [7:0] sin15 = $signed(8'b10000001);
wire signed [7:0] sin16 = $signed(8'b10000111);
wire signed [7:0] sin17 = $signed(8'b10000111);
wire signed [7:0] sin18 = $signed(8'b10110101);
wire signed [7:0] sin19 = $signed(8'b11011001);

counter tima (.ena(ena),.rst(rst),.clk(clk),.tim(tim),.clk_0(clk_0));
regPrzes rega(.ena(ena),.rst(rst),.clk(clk_0),.rec(rec),.out0(out0),
.out1(out1),.out2(out2),.out3(out3),.out4(out4),.out5(out5),.out6(out6),
.out7(out7),.out8(out8),.out9(out9),.out10(out10),.out11(out11),
.out12(out12),.out13(out13),.out14(out14),.out15(out15),.out16(out16),
.out17(out17),.out18(out18),.out19(out19));

assign out = out1 * sin1 + out2 * sin2 + out3 * sin3 + out4 * sin4
+ out5 * sin5 + out6 * sin6 + out7 * sin7 + out8 * sin8 +
out9 * sin9 + out10 * sin10 + out11 * sin11 + out12 * sin12 +
out13 * sin13 + out14 * sin14 + out15 * sin15 + out16 * sin16 +
out17 * sin17 + out18 * sin18 + out19 * sin19;
assign detect = (out > 80000 || out <-80000) ? 1 : 0;

assign clk0 = clk_0;

endmodule

```

#### 5.1.4. Podmoduł tima

Odpowiada za dzielenie zegrara. Zamienia częstotliwość 50 Mhz płytka na 1 Mhz do takto-wania rejestrów aby w odpowiednim tempie zapełniał komórki tego układu

Kod modułu:

```

module counter(
    input ena,
    input clk,
    input rst,

```

```

        output [4:0]  tim ,
        output  clk_0
    );

reg [4:0]  r_tim=0;
reg [4:0]  n_tim=0;

always@ (posedge clk , posedge rst)begin
    if( rst ) r_tim <= 5'b0;
    else if( ena) r_tim <= n_tim ;
end

always@ (*) begin

    n_tim = r_tim + 1;
    if( n_tim==20) n_tim = 0;
end

assign tim = r_tim;
assign clk_0 = (r_tim == 0) ? 1 : 0;
endmodule

```

### 5.1.5. Podmoduł regPrzes

Moduł ten odpowiada za imitacje okna dla korelacji. Sygnał na podstawie którego jest korelacja ma 20 próbek zatem można było stworzyć rejestr przesuwający który pobiera dane z częstotliwością  $f_{min} = 1$  MHz oraz cały czas jest dzięki niemu dostępna pamięć tego rejestrów (20 wartości \* 8 bitów = 160 bitów rejestrów). Dane te są wykorzystywane przez logikę kombinacyjną do wyciągnięcia z tych danych korelacji dla danej zawartości tego rejestrów.

Kod układu znajduje się poniżej:

```

module regPrzes (
    input  ena ,
    input  clk ,
    input  rst ,
    input  signed [7:0]  rec ,

    output signed [7:0]  out0 ,
    output signed [7:0]  out1 ,
    output signed [7:0]  out2 ,
    output signed [7:0]  out3 ,
    output signed [7:0]  out4 ,
    output signed [7:0]  out5 ,
    output signed [7:0]  out6 ,
    output signed [7:0]  out7 ,
    output signed [7:0]  out8 ,
    output signed [7:0]  out9 ,
    output signed [7:0]  out10 ,

```

```

        output signed [7:0] out11 ,
        output signed [7:0] out12 ,
        output signed [7:0] out13 ,
        output signed [7:0] out14 ,
        output signed [7:0] out15 ,
        output signed [7:0] out16 ,
        output signed [7:0] out17 ,
        output signed [7:0] out18 ,
        output signed [7:0] out19
);

reg [19:0] bit1 ;
reg [19:0] bit2 ;
reg [19:0] bit3 ;
reg [19:0] bit4 ;
reg [19:0] bit5 ;
reg [19:0] bit6 ;
reg [19:0] bit7 ;
reg [19:0] bit8 ;

reg [19:0] mbit1 ;
reg [19:0] mbit2 ;
reg [19:0] mbit3 ;
reg [19:0] mbit4 ;
reg [19:0] mbit5 ;
reg [19:0] mbit6 ;
reg [19:0] mbit7 ;
reg [19:0] mbit8 ;

reg signed [7:0] outr ;

always@(posedge clk ,posedge rst)begin
    if( rst)begin
        bit1 <= 20'b0;
        bit2 <= 20'b0;
        bit3 <= 20'b0;
        bit4 <= 20'b0;
        bit5 <= 20'b0;
        bit6 <= 20'b0;
        bit7 <= 20'b0;
        bit8 <= 20'b0;
    end
    else if(ena)
    begin
        bit1 <= mbit1 ;
        bit2 <= mbit2 ;
        bit3 <= mbit3 ;
        bit4 <= mbit4 ;
    end
end

```

```

        bit5 <= mbit5;
        bit6 <= mbit6;
        bit7 <= mbit7;
        bit8 <= mbit8;
    end
end

always@(*) begin
    mbit1 = bit1 << 1;
    mbit1[0] = rec[0];
end

always@(*) begin
    mbit2 = bit2 << 1;
    mbit2[0] = rec[1];
end

always@(*) begin
    mbit3 = bit3 << 1;
    mbit3[0] = rec[2];
end

always@(*) begin
    mbit4 = bit4 << 1;
    mbit4[0] = rec[3];
end

always@(*) begin
    mbit5 = bit5 << 1;
    mbit5[0] = rec[4];
end

always@(*) begin
    mbit6 = bit6 << 1;
    mbit6[0] = rec[5];
end

always@(*) begin
    mbit7 = bit7 << 1;
    mbit7[0] = rec[6];
end

always@(*) begin
    mbit8 = bit8 << 1;
    mbit8[0] = rec[7];
end

```

```

assign out0 = {bit8[0], bit7[0], bit6[0], bit5[0],
bit4[0], bit3[0], bit2[0], bit1[0]};
assign out1 = {bit8[1], bit7[1], bit6[1], bit5[1],
bit4[1], bit3[1], bit2[1], bit1[1]};
assign out2 = {bit8[2], bit7[2], bit6[2], bit5[2],
bit4[2], bit3[2], bit2[2], bit1[2]};
assign out3 = {bit8[3], bit7[3], bit6[3], bit5[3],
bit4[3], bit3[3], bit2[3], bit1[3]};
assign out4 = {bit8[4], bit7[4], bit6[4], bit5[4],
bit4[4], bit3[4], bit2[4], bit1[4]};
assign out5 = {bit8[5], bit7[5], bit6[5], bit5[5],
bit4[5], bit3[5], bit2[5], bit1[5]};
assign out6 = {bit8[6], bit7[6], bit6[6], bit5[6],
bit4[6], bit3[6], bit2[6], bit1[6]};
assign out7 = {bit8[7], bit7[7], bit6[7], bit5[7],
bit4[7], bit3[7], bit2[7], bit1[7]};
assign out8 = {bit8[8], bit7[8], bit6[8], bit5[8],
bit4[8], bit3[8], bit2[8], bit1[8]};
assign out9 = {bit8[9], bit7[9], bit6[9], bit5[9],
bit4[9], bit3[9], bit2[9], bit1[9]};
assign out10 = {bit8[10], bit7[10], bit6[10], bit5[10],
bit4[10], bit3[10], bit2[10], bit1[10]};
assign out11 = {bit8[11], bit7[11], bit6[11], bit5[11],
bit4[11], bit3[11], bit2[11], bit1[11]};
assign out12 = {bit8[12], bit7[12], bit6[12], bit5[12],
bit4[12], bit3[12], bit2[12], bit1[12]};
assign out13 = {bit8[13], bit7[13], bit6[13], bit5[13],
bit4[13], bit3[13], bit2[13], bit1[13]};
assign out14 = {bit8[14], bit7[14], bit6[14], bit5[14],
bit4[14], bit3[14], bit2[14], bit1[14]};
assign out15 = {bit8[15], bit7[15], bit6[15], bit5[15],
bit4[15], bit3[15], bit2[15], bit1[15]};
assign out16 = {bit8[16], bit7[16], bit6[16], bit5[16],
bit4[16], bit3[16], bit2[16], bit1[16]};
assign out17 = {bit8[17], bit7[17], bit6[17], bit5[17],
bit4[17], bit3[17], bit2[17], bit1[17]};
assign out18 = {bit8[18], bit7[18], bit6[18], bit5[18],
bit4[18], bit3[18], bit2[18], bit1[18]};
assign out19 = {bit8[19], bit7[19], bit6[19], bit5[19],
bit4[19], bit3[19], bit2[19], bit1[19]};

assign out = outr;

endmodule

```

### 5.1.6. MyOut

Moduł *MyOut* odpowiada za komunikację układu na zewnątrz przekazując zatrzasniętą wartość na zewnątrz układu do PIO. Aby zatrzasnąć wartość do modułu musi dotrzeć sygnał *detect*.

Moduł ten wysyła dodatkowo sygnał *rdy* na zewnątrz który informuje procesor o wykryciu sygnału. Układ stanowi swoisty bufor dla sygnału *detect*.

Kod znajduje się poniżej:

```
module myOut(
    input ena ,
    input rst ,
    input clk ,
    input signed [14:0] in ,
    input detect ,

    output rdy ,
    output signed [14:0] out
);

reg [13:0] n_out;
reg [13:0] r_out;

reg r_rdy;
reg n_rdy;

always@(posedge clk , posedge rst) begin
    if( rst ) begin
        r_out <= 0;
        r_rdy <= 0;
    end else if( ena ) begin
        r_out <= n_out;
        r_rdy <= n_rdy;
    end
end

always@(*) begin
    n_rdy = r_rdy;
    n_out = r_out;

    if( detect ) begin
        n_out = in ;
        n_rdy = 1;
    end
end
```

```

assign out = n_out;
assign rdy = n_rdy;

endmodule

```

### 5.1.7. Detektor - sposób detekcji momentu odbicia

Detektor sprawdza warunek konieczny, aby uznać, że korelacja jest na tyle wysoka, żeby uznać, że wtedy sygnał dotarł do detektora. Aby tego dokonać trzeba było ustalić jaka wartość będzie minimalną, aby ten warunek spełnić.

Ustalono, że wektory 8-bitowe są to liczby w kodzie U2 w zakresie od -1 do 1 oznacza to 7 bitów na wartość i jeden na znak (można więc to zamienić na zakres od -127 do 127). Wektory testowe zamieniono więc do formy całkowitej aby mnożąc wartość ułamkową z pliku tekstowego na wartość całkowitą.

Oczekiwana wartość wg testów to ok 5 zatem gdy mnożymy dwie liczby które mają 7 cyfr po przecinku (gdziż wartości zostały zakodowane kodem 8 bitowym U2), zatem mnożenia będą miały 14 miejsc po przecinku. Trzeba założyć, że poprzez dodawanie, do którego doszło podczas korelacji potrzeba będzie jeszcze więcej bitów. Na zaś założono, że takowa korelacja może mieć 24-bity.

Maksymalna liczba 14-bitowa w kodzie NKB to 16 384 - możemy założyć, że dla naszej korelacji ta wartość jest równa wartości liczonej na kartce = 1. Skoro przewidywaną wartością dla maksymalnego skoku jest ok 5 oznacza to, że żeby uznać korelację, za tą odpowiadającą dotarciu do odbiornika należy pomnożyć wcześniejszej ustaloną wartość \*5 co daje nam  $x_{szukane} = 81920$ . Dla uniknięcia wahań sygnału uznano, że wartość korelacji sygnału (zaokrąglając ustaloną wartość) odbitego wynosi w przedziale  $(-\infty, -80000] \cup [80000, \infty)$ .

Dzieki tym ustaleniom opracowano układ sprawdzający za pomocą operacji logicznych, czy wartość korelacji znajduje się w ustalonym przedziale, który w założeniu ma spełniać warunki dotarcia sygnału do odbiornika.

Kod detektora opracowanego na podstawie powyższych zaleceń :

```

module detector(
    input signed [23:0] in ,
    output out
);

assign out = (in > 80000 || in < -80000) ? 1 : 0;

endmodule

```

Zrewidowano wcześniejsze ustalenia (że odległość ma równa być 3017, gdyż założono, że warunkiem koniecznym jest osiągnięcie zbioru spoza  $[-4,4]$ , okazał się ten zbiór jednak zbyt szerokim aby wynik był wiarygodny). Zatem po tym ustalono, że oczekiwany numer próbki w którym korelacja osiągnie wartość rzadzaną wynosi  $x_{szukane} = 3001$

### 5.1.8. Moduł timer

Timer oblicza numer próbki, która jest analizowana w danym momencie i przesyła ten czas do modułu MyOut. Timer ten zakłada wartość pierwotną równą -20, gdyż dla pierwszych 20 odliczeń tego licznika, uzyskujemy niepełne okno (z pustymi wartościami, które nie są wiarygodne). Planuje się aby to procesor analizował takowe błędy i w razie wykrycia sygnału w trakcie pierwszych 20 odliczeń, na wyjściu był czas ujemny co program procesora interpretowałby jako błąd.

Timer jest taktowany zegarem 1Mhz, gdyż z taką częstotliwością jest taktowany sygnał wejściowy procesora DSP.

Wartość timera jest wyrażona w kodzie U2 dla liczb 14-bitowych (1 bit znaku i 13 wartości), żeby mogła zmieścić się wartość sygnału, gdyż przewidziano, że do detektora w strumieniu dojdzie 5000 próbek (maksymalna wartość 8128)

Kod znajduje się poniżej:

```
module timer(
    input ena ,
    input clk ,
    input rst ,
    output signed [14:0] tim
);

reg signed [13:0] n_tim ;
reg signed [13:0] r_tim ;

always@ (posedge clk , posedge rst )begin
    if (rst) r_tim <=-20;
    else if (ena) r_tim <= n_tim ;
end

always@ (*) begin
    n_tim = r_tim + 1;
end

assign tim = r_tim ;
endmodule

endmodule
```

### **5.1.9. Współpraca układu DSP z procesorem**

**Uwaga :** Kod dla procesora zostanie (na termin etapu IV) opracowany zgodnie z informacją zawartą na początku .

Procesor będzie przetwarzał informacje przekazanie przez PIO (układ I/O odpowiadający za odbiór informacji z procesora DSP). Tak aby wyciągnąć kluczowe informacje na temat odbitego sygnału. Chodzi przede wszystkim o:

- Przeliczanie na podstawie wzorów sporządzonych podczas Etapu II (Informacje ogólne) z numeru próbki na odległość w kilometrach
- Przekazywanie do komputera informacji o przeliczonej odległości
- Informowanie o błędach przez moduł JTAG (ujemny numer próbki)

Program zostanie wykonany w języku C przy użyciu wewnętrznych bibliotek środowiska Eclipse Nios. Program zosatnie napisany w pakiecie NIOS II Software Build Tools for Eclipse

### **5.1.10. Zastosowana pamięć**

W układzie zastosowano układ pamięci ”On-Chip Memory (RAM or ROM) Intel FPGA IP” o rozmiarze 16 kiB, aby pomieścić pamięć operacyjną oraz program, który będzie wykonywał powyższe zadania. Aby to miejsce wystarczyło ograniczono do minimum rozmiar programu stosując szablon programu w pakiecie Eclipse ”Hello World Simple”.

Układ ten umożliwia zachowywanie w pamięci liczb niezbędnych do wykonania obliczeń odległości detekcji błędu i innych rzeczy związanych z wykonaniem procesora, komunikującego się przez JTAG z komputerem, aby dokonywać operacji, które nie są tak zasobocenne i nie trzeba ich implementować w sprzęcie.

### **5.1.11. Przykład przepływu danych po układzie**

Podczas takiego pomiaru, przepływ informacji w naszym systemie zaczyna się od dotarcia sygnału do modułu conv. Następnie trafia do modułu detect który wysyła synał detect do modułu ”MyOut”, kończąc pomiar oraz zatrzaskując informacje o czasie dotarcia sygnału który jest przekazywany przez PIO do procesora. Procesor na podstawie programu przetwarza

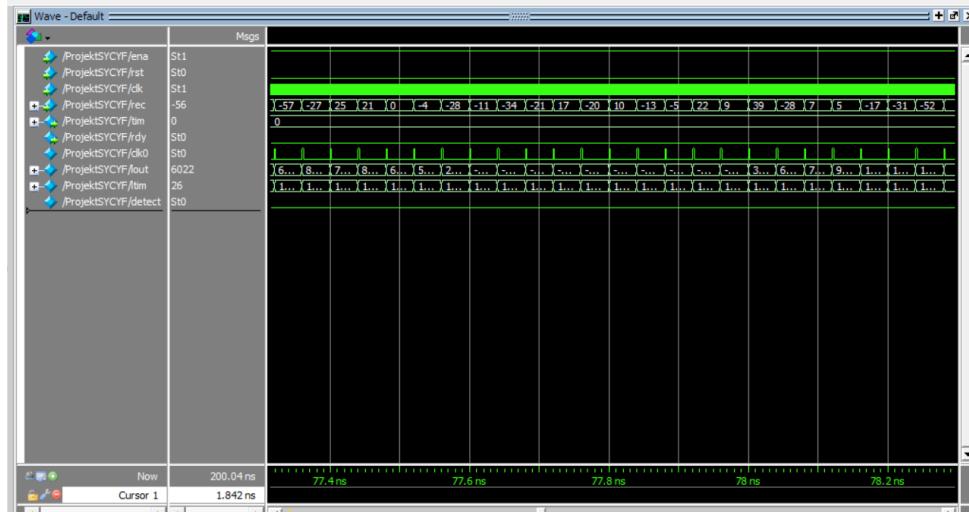
## **5.2. Wstępne testy**

Wydruk programu wykonano dla jedynie procesora DSP. Wyniki wydajnościowe zostaną zaprezentowane zarówno dla ”czytego” modułu DSP jak i dla całego systemu

### **5.2.1. Wydruki programu ModelSim**

Wykonano testy za pomocą wcześniej przygotowanych wektorów testowych aby wstępnie ocenić czy układ spełnia założenia. Wyniki te zostaną później potwierdzone benchtestem.

Poniżej znajduje się wykres działania symulacji układu w programie ModelSim Altera:



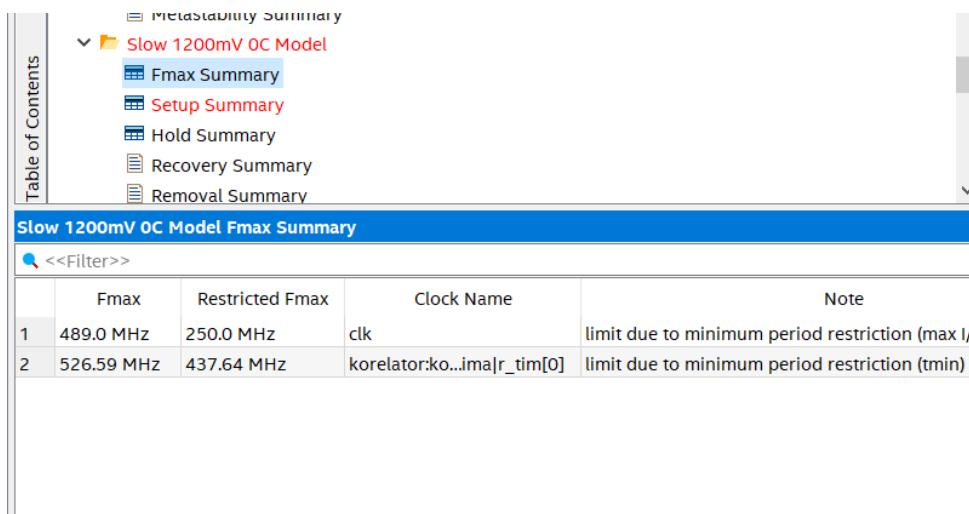
Wykres symulacji [39]

Układ działa zgodnie z oczekiwaniami w całej dziedzinie czasu analizy. Nie występują nie-pszodziewane peaki, a sygnały są zgodne z oczekiwaniami (counter nadaje co 20 okresów zegara itd.). Sygnały są czytelne, wartości zgodne z oczekiwaniami.

### 5.2.2. Omówienie szybkości

Na podstawie raportów wygenerowanych dla komplikacji układu z i bez procesora uzyskano wyniki:

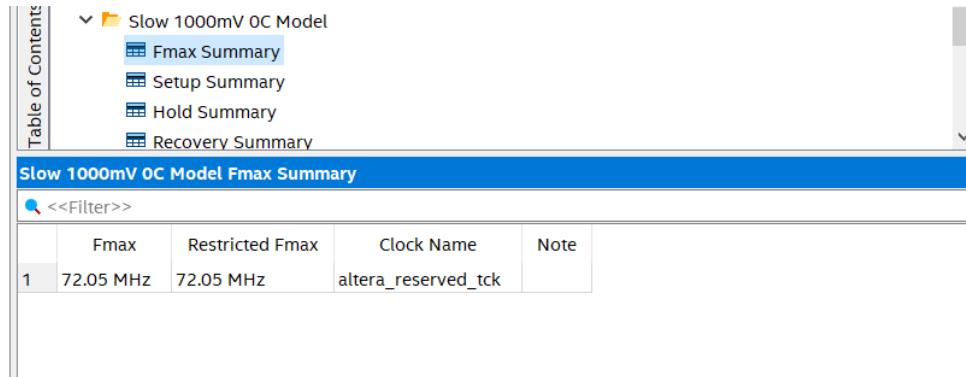
1) Dla "gołego" DSP:



Dane na temat maksymalnej częstotliwości [39]

Gołe DSP osiągnęło bardzo dobry wynik jak na układ FPGA jeżeli chodzi o częstotliwość (ponad 450 MHz). Taki układ w zupełności wystarcza do realizacji zadania projektowego, gdyż będzie powodował małe opóźnienie a zatem potencjalnie nawet powolny procesor nie spowoduje spadku tej częstotliwości poniżej 50 Mhz, czyli częstotliwości taktowania płytki, zatem jest bardzo małe ryzyko powstania zjawiska zwane *hazardem*, przez co jest duża szansa na stworzenie wydajnego układu

2) Dla DSP wraz z całym systemem cyfrowym



Fmax	Restricted Fmax	Clock Name	Note
1 72.05 MHz	72.05 MHz	altera_reserved_tck	

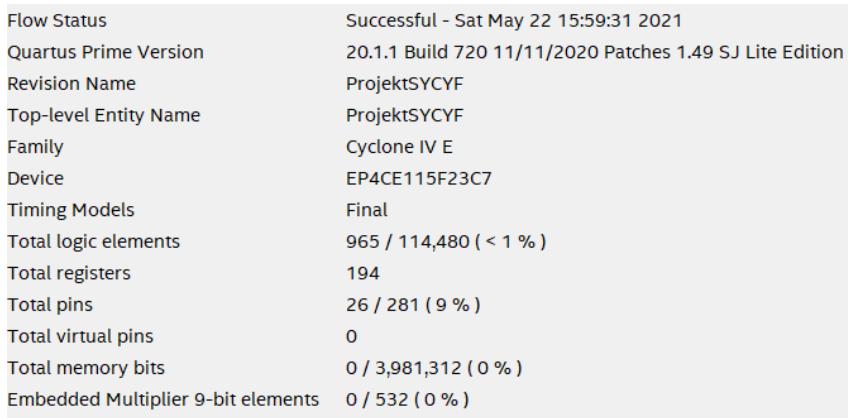
Dane na temat maksymalnej częstotliwości [39]

Zgodnie z oczekiwaniami  $f_{max}$  zmieścił się w przedziale powyżej 50 Mhz, zatem powyższa implementacja układu DSP nadaje się do zastosowania nawet z darmową wersją procesora NIOS. Daje to solidne podstawy pod to aby uznać to rozwiązanie za wydajne i odpowiednie dla naszych zastosowań.

### 5.2.3. Omówienie wymaganych zasobów

Na podstawie raportów wygenerowanych dla komplikacji układu z i bez procesora uzyskano wyniki:

1) Dla "gołego" DSP:



Flow Status	Successful - Sat May 22 15:59:31 2021
Quartus Prime Version	20.1.1 Build 720 11/11/2020 Patches 1.49 SJ Lite Edition
Revision Name	ProjektSYCYF
Top-level Entity Name	ProjektSYCYF
Family	Cyclone IV E
Device	EP4CE115F23C7
Timing Models	Final
Total logic elements	965 / 114,480 ( < 1 % )
Total registers	194
Total pins	26 / 281 ( 9 % )
Total virtual pins	0
Total memory bits	0 / 3,981,312 ( 0 % )
Embedded Multiplier 9-bit elements	0 / 532 ( 0 % )
Total PLLs	0 / 4 ( 0 % )

Dane na temat zajętości zasobów [39]

Procesor DSP zajmuje ok 960 bloków logicznych, dla naszej specyfikacji. Jest to poniżej 1% zasobów docelowej platformy - płytki FPGA Cyclone IV E EP4CE115F23C7. Taki układ więc zmieści się w docelowym układzie FPGA, zatem śmiałko można go zastosować do budowy systemu. Warto było jednak sprawdzić również zajętość zasobów wraz z procesorem aby ocenić, czy ostateczny wynik spełnia założenia projektowe

Samo DSP zajmuje 2) Dla DSP wraz z całym systemem cyfrowym

Flow Status	Successful - Sat May 22 15:26:40 2021
Quartus Prime Version	20.1.1 Build 720 11/11/2020 Patches 1.49 SJ Lite Edition
Revision Name	ProjektSYCYFProcesor
Top-level Entity Name	ProjektSYCYFProcesor
Family	Cyclone IV E
Device	EP4CE115F23C9L
Timing Models	Final
Total logic elements	2,650 / 114,480 ( 2 % )
Total registers	1165
Total pins	23 / 281 ( 8 % )
Total virtual pins	0
Total memory bits	142,336 / 3,981,312 ( 4 % )
Embedded Multiplier 9-bit elements	0 / 532 ( 0 % )
Total PLLs	0 / 4 ( 0 % )

Dane na temat zajętości zasobów [39]

Układ po dołączeniu pozostałych komponentów zajmuje tym razem ok 2% (2650 bloków logicznych) zasobów docelowego układu FPGA. Układ taki spełnia założenia zarówno jeżeli chodzi o częstotliwość jak i zasoby zajmowane przez układ. Ilość pamięci wewnętrznej wykorzystywanych przez projekt jest również jedynie ułamkiem dostępnej na płytce (142 kB / 3 MiB) dokładnie 4 %. Oznacza to że zasoby dostępne na platformie docelowej w zupełności wystarczają do wykonania takowego układu

### 5.3. Ocena rozwiązania

#### 5.3.1. Ocena szybkości rozwiązania

Rozwiązanie jest wystarczająco szybkie aby można było zastosować go na docelowej platformie. Procesor DSP wykazuje nadzwyczaj dużą szybkość działania, zaś układ cyfrowy mimo, że stosunkowo ciężki, nadal mieści się w granicach częstotliwości, które muszą być spełnione, aby nie doszło do zjawiska *hazardu*. Układ wskazuje bardzo wysoki potencjał jeżeli chodzi o zadanie, jakie zostało przed nim postawione. Wstępne testy nie wskazują znaczących anomalii związanych z stanem nieustalonym wyjścia itp (np niespodziewane peaki).

Układ prawdopodobnie mógłby być szybszy. Dodatkowo nadal nie opracowano programu w C który będzie wykonywał zadania wspomniane wcześniej. Szybkość układu nie jest wszystkim, co definiuje ten układ, by móc powstawać problemy z zbyt wolnym programem wewnętrz. Dodatkowo mała pamięć może spowodować, że trzeba będzie podzielić program na mniejsze składowe, wypadkowo spowalniając go. Niemniej wstępne testy pokazują, że układ nadaje się do zaleconych zadań.

#### 5.3.2. Ocena zajętości wymaganych zasobów

Układ zajmuje zaledwie ułamek zasobów, którymi dysponuje docelowy układ. Podobnie pamięć to jedyne 4 % dostępnej. Oznacza to że taki układ FPGA mógłby teoretycznie zawierać wiele takich układów. Zasoby mieścią się na układzie zarówno procesor DSP jak i cały układ wraz z procesorem i układami wyjścia wejścia (I/O). Układ przez to wykazuje duży potencjał jeżeli chodzi o dalszą rozbudowę pod nowe zadania.

Mimo to architektura mogłabybyć prawdopodobnie bardziej dopracowana, aby zajmowała jeszcze mniej bloków. Niemniej to jest jedynie ułamek zasobów dostępnych dla zespołu zatem układ nadaje się do realizacji zadań.

#### **5.4. Scenariusze testowe**

Przed wykoniem testu wykonano przewidywania dotyczące scenariuszy testowych dotyczących tego przypadku

##### **5.4.1. Scenariusz 1**

- Nazwa : Pomiar dla 2000 próbek
- Warunki wstępne :
  - Podłączenie modułu do detektora
  - Uruchomienie procesora
- Kroki wykonania :
  - Uruchomienie detektora
  - Oczekanie 2s.
  - Uruchomienie układu DSP
  - Oczekiwanie na wynik po 2000 próbkach
- Oczekiwany rezultat :

Nie otrzymamy pożądanego wyniku, zatrzaśnięta wartość pozostanie- nadal zerem nie pojawi się sygnał rdy, co równoważne jest tym że nic się nie stało. Sygnał nie zdażył powrócić do odbiornika, zatem wynik nie będzie zgodny z oczekiwaniem

##### **5.4.2. Scenariusz 2**

- Nazwa : Pomiar dla czasu oczekiwania zamiast 1s to 2s
- Warunki wstępne :
  - Podłączenie modułu do detektora
  - Uruchomienie procesora
- Kroki wykonania :
  - Uruchomienie detektora
  - Oczekanie 1s.
  - Uruchomienie układu DSP
  - Oczekiwanie na wynik po 5000 próbkach
- Oczekiwany rezultat :

Nie otrzymamy pożądanego wyniku, zatrzaśnięta wartość pozostanie nadal zerem nie pojawi się sygnał rdy, co równoważne jest tym że nic się nie stało. Sygnał nie zdażył powrócić do odbiornika, zatem wynik nie będzie zgodny z oczekiwaniem. Gdyby poczekano jeszcze jedną sekundę uzyskanoby prawdopodobnie wynik

##### **5.4.3. Scenariusz 3**

- Nazwa : Pomiar dla pełnego zakresu czasu
- Warunki wstępne :
  - Podłączenie modułu do detektora
  - Uruchomienie procesora
- Kroki wykonania :
  - Uruchomienie detektora
  - Oczekanie 2s.
  - Uruchomienie układu DSP

- Oczekiwanie na wynik po 5000 próbkach
- Oczekiwany rezultat  
Uzyskanie prawidłowego wyniku czyli  $t_o = 3001$  czasu trwania próbki oraz zatrzaśnięty sygnał rdy. Równoważne jest to prawidłowemu pomiarowi.

#### 5.4.4. Działania w razie uzyskania negatywnych wyników testów benchtesta

W razie niepowodzenia testu należałoby szukać przyczyny błędu. Warto było jednak ustalić procedurę, która pomogła zespołowi poprawiać projekt wcześniej (wersje wcześniejsze nie były wystarczająco dopracowane, aby dawały dobre wyniki).

- Sprawdzano czy wykonano prawidłowy plik do
- Sprawdzano czy połączenia między elementami benchtestu są prawidłowe
- Jeżeli porzednie elementy nie dały odnalezienia błędu oznacza to, że błąd jest głębiej zagnieżdżony i wymaga on dogłębnej inspekcji kodu
- Jeżeli błąd został odnaleziony ponawiano test.
- Sprawdzić poprawność wektorów testowych
- Sprawdzić czy czas symulacji był odpowiedni (czy obejmuje wszystkie 5000 próbek)
- Jeżeli powyższe punkty dały efekt w postaci znalezienia problemu należało powtórzyć test
- Jeżeli powyższe punkty nie dały efektu w postaci odnalezienia błędów oznacza to, że implementacja jest błędna

### 5.5. Weryfikacja funkcjonalna

#### 5.5.1. Założenia testbencha

Testbench jest modułem testującym układ, który umożliwia testowanie układów bez zmysłnego oglądania przebiegu, po wpuszczeniu w programie symulującym np ModelSim, oglądania przebiegów.

Układ benchtesta musi sprawdzić czy zostanie zatrzaśnięta wartość 3001, czyli ta, która została ustalona wcześniej.

#### 5.5.2. Moduł wejściowy - plik do z wektorami testowymi

Przed opracowaniem benchtesta należało przetworzyć wektory podane w formacie decymalnym na takie, które będą zrozumiałe przez układ oraz są na tyle prawdopodobne aby przypominały te jakie może wysyłać układ detektora.

Zgodnie z ustaleniami z opisu modułu detektra przetworzono wekotry mnożąc je przez 128 i kodując te wartości w kodzie U2 aby uzyskać ciąg wektorów testowych. Dodatkowo napisano imitację zegara procesorowego.

Do wykonania tych wektorów tesowych użyto środowiska Python, aby przetworzyć pliki txt (z etapu III), na wektory wejściowe dla FPGA.

Do realizacji tego zadania wykorzystano poniższy program napisany w języku Python:

```
linijki = []
```

```

def decimalToBinary(n):
    return bin(n).replace("0b", "")

def presentValue(el, num, name):
    value = int(el * 128);
    tex = ""

    if value >= 0:
        tex = decimalToBinary(value) + ""
    else:
        tex = decimalToBinary(value & 0xff) + ""
    for n in range(8 - len(tex)):
        tex = "0" + tex

    time = num * 40
    linijki.append("force "+name+" "+tex+" "+str(time))

plik = open(r"received8.txt", "r")
plik2 = open(r"sent8.txt", "r")

lista= []
listasin= []

for elementy in plik:
    try:
        lista.append(float(elementy))
    except Exception:
        print("")

for elementy in plik2:
    try:
        listasin.append(float(elementy))
    except Exception:
        print("")

plik.close()
plik2.close()

num = 0
sinn = 0
linijki.append(r"restart -nowave -force")
linijki.append(r"add wave *")

lin = 0
for el in lista:

```

```

presentValue( el ,num," rec" )

num = num + 1

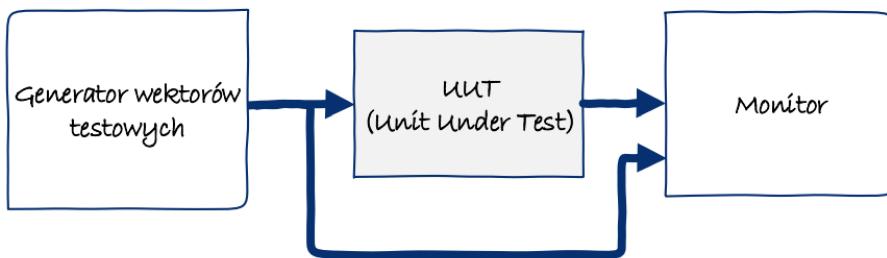
linijk1.append(" run " + str(num*40+40))
write = open(" output.txt ","w")
for linia in linijk1:
    write.write(linia+"\n")

write.close()

```

Analiza za pomocą takiego formatu wektorów jest może i powala ale bardzo precyzyjna i pozwala na wykrycie nawet drobnych defektów. Nadal emulacja układów cyforwych na bardzo niskim poziomie (bramek itp.) jest bardzo zasobozerna, przez co testy potrafiły trwać nawet 1h.

Następnie wykonano moduł benchtestu. Elementy opisane poniżej oraz sposób ich współpracy opisuje schemat poniższy:



Schemat benchtestu [39]

### 5.5.3. Moduł generatora

Moduł generatora benchtestu będą imitować wcześniej wygenerowane wektory z pliku do. Włączenie funkcji za pomocą komendy `do` [nazwa pliku do] daje nam możliwość z odpowiednio przygotowanym monitorem pełnej automatyzacji testu. Benchtest z użyciem komponentu jako generatora byłby problematyczny ze względu na ograniczenia narzędzia Quartus pod względem wykonywanej ilości iteracji oraz powtóżeń dla logiki kombinacyjnej (ilość wektorów jest bardzo duża a projektowane specjalnie do tego celu ROM-u jest w przypadku opracowania pliku do zbędną czynnością, pochłaniającą cenny czas przy projektowaniu układu)

### 5.5.4. Moduł monitora

Monitor będzie służył do wykrycia czy w odpowiednim momencie dotrze spodziewany sygnał, czyli numer próbki 3001, który jest spodziewaną wartością. Sprawdzany jest czy dla wartości 3001 spełniony jest warunek detektora z modułu DSP oraz wyświetla komunikat gdy ten test zostanie zakończony pozytywnie. Kod monitora znajduje się poniżej:

```
module monitor(
```

```

    input  clk ,
    input [13:0]  tim ,
    input  rdy
);

always@(posedge clk) begin
    if (tim == 3001 && rdy) begin
        $display ("Test zakończył się sukcesem");
    end
end
endmodule

```

Gdy test stwierdzi poprawność działania układ w pętli zacznie wyświetlać się komunikat *”Test zakończył się sukcesem”*.

Kod znajduje się również w repozytorium projektowym

### 5.5.5. Właściwy benchtest

Aby wykonać benchtest należało te trzy moduły połączyć. Kod modułu połączonego znajduje się poniżej:

Kod znajduje się również w repozytorium projektowym

```

module ProjektSYCYFTest(
    input  clk ,
    input  rst ,
    input  ena ,
    input [7:0]  rec
);

    wire [13:0]  tim ;
    wire  rdy ;

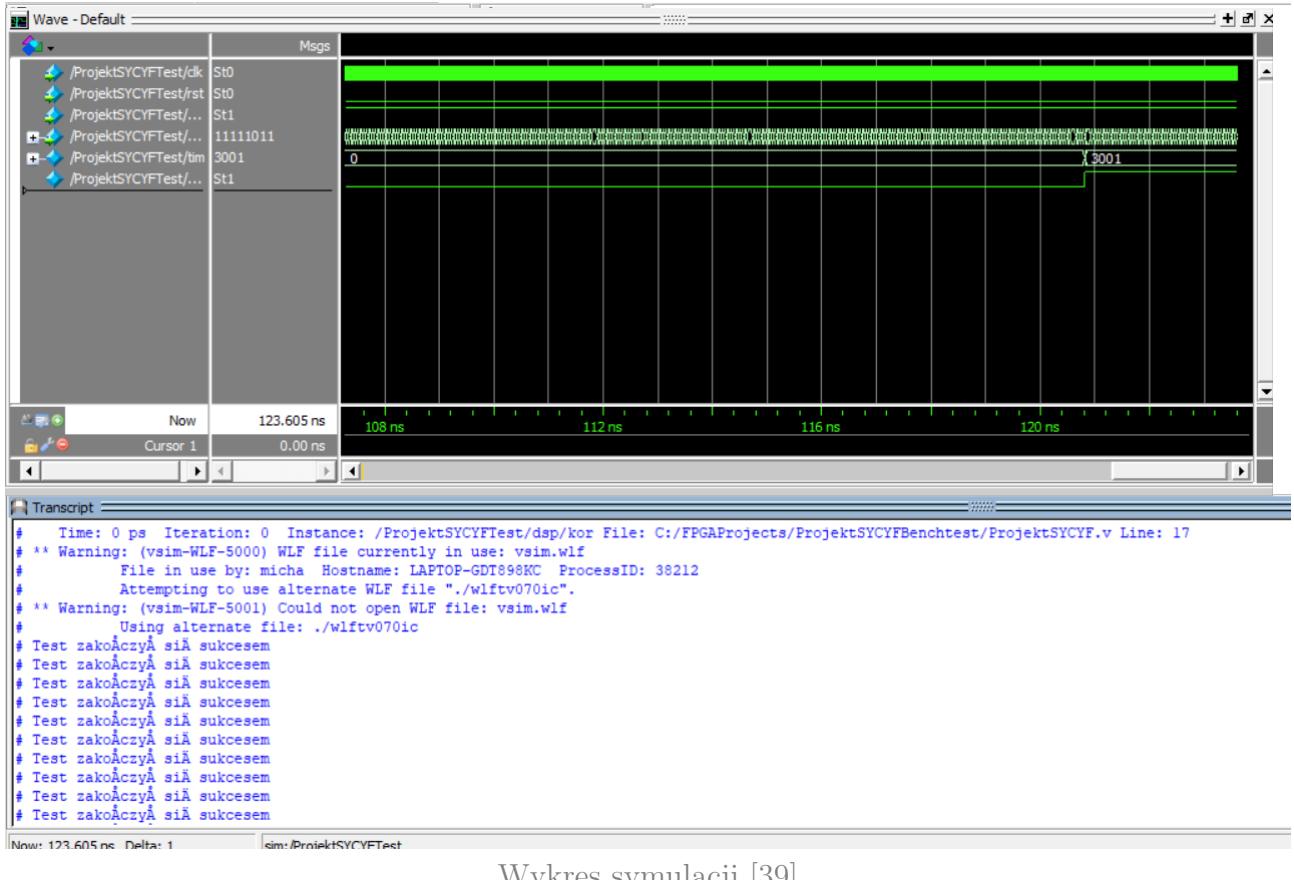
    ProjektSYCYF  dsp (.clk(clk), .rst(rst), .rec(rec),
    ..ena(ena), .tim(tim), .rdy(rdy));
    monitor  monit (.clk(clk), .tim(tim), .rdy(rdy));

endmodule

```

Skoro procesor jest takowany zegarem  $f_m = 50$  MHz oznacza to że precyza symulacji musi wynosić 10 ms / 10 ps, (połowa okresu), aby można było zasymulować zegar.

### 5.5.6. Wydruki z programu ModelSim



Wykres symulacji [39]

Uzyskano wynik zgodny z powyższymi oczekiwaniami : pętla nieskończona napisów świadczących o powodzeniu wykonanego testu. Dodatkowo przebiegi pokazują że zatrzaśnięto wartość 3001 co potwierdza, że układ działa prawidłowo

### 5.5.7. Ocena poprawności działania

Benchtest pokazuje powodzenie testu. Wartość 3001 została przyjęta przez monitor i zatwierdzona jako poprawna. Oznacza to, że specyfikacja sprzętowa wykonuje się poprawnie przynajmniej dla pierwotnych wektorów testowych. Daje to powód do tego aby zacząć przystosowywać układ do uruchomienia na prawdziwej platformie (płytki FPGA).

## 6. Uruchomienie

Etap V polega na uruchomieniu układu na docelowym sprzęcie. Ze względu na brak dostępu do płytki dostępnej na uczelni, użyto mniej zaawansowanego zamiennika, który nadal umożliwia uruchomienie zaprojektowanego wcześniej układu.

Zrezygnowano z użycia procesora na rzecz układów imitujących jego funkcje. Dzięki temu uzyskano układ prostszy. Zrobiono to, gdyż uznano, że jedyną operacją, jaką w tym przypadku musi wykonać procesor, jest mnożenie, dzięki któremu uzyskujemy przeliczenie z klatek na kilometry. Uznamo to za niewystarczająco wyrafinowaną operację, aby angażować do tego cały procesor, użyto więc układu mnożącego, o którym w dalszych podpunktach.

Wszelki kod zamieszczony w tym rozdziale znajduje się również w repozytorium projektowym na portalu GitHub.com.

## 6.1. Parametry sprzętu na którym dokonano uruchomienia

### 6.1.1. Model płytki

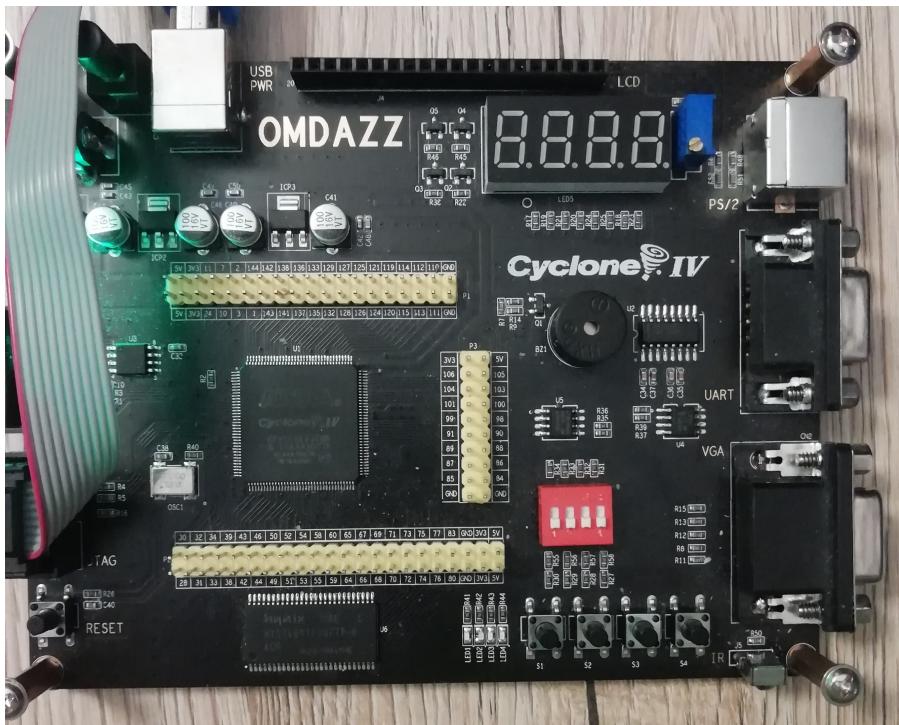
Mniej wyrafinowanym zamiennikiem płytki testowej DE2\_115 w przypadku tego zespołu był również Cyclone IV E, ale model **EP4CE6E22**. Płytnka ta udostępnia o wiele mniej zasobów niż pierwotny sprzęt docelowy, ale nadal jest to wystarczające pod potrzeby tego projektu.

Płytnka ta ma dostępny jedynie zegar 50MHz (tylko takiego używamy w projekcie).

Wybór ten uzasadniono tym, iż była to jedyna dostępna płytka FPGA przeznaczona do testów układów napisanych w języku opisu sprzętu VerilogHDL. Mimo stosunkowo małej ilości zasobów płytka wykazuje nadzwyczaj duży potencjał, jeżeli chodzi o implementowanie układów cyfrowych.

Układ ten docelowo zmieścił wraz z układami dodatkowymi, które są potrzebne do ręcznej obsługi płytki ok 25% całości zasobów. Wynika z tego, że nawet w teorii najsłabszej wersji tego sprzętu, można nadal zmieścić cztery w pełni funkcjonale układy przeznaczone do wykonania projektu.

Poniżej znajduje się zdjęcie powyższego modelu płytka:



Płytnka EP4CE6E22 [32]

### 6.1.2. Parametry

Poniższa tabela przedstawia parametry fizyczne płytki EP4CE6E22:

Częstotliwość sygnału zegara systemowego pochodzącego z oscylatora na płytce	50 MHz
Blok logiczny (LE's)	6,272
Dostępne piny	92
Wbudowana pamięć	256 Kbit

Według wcześniejszych ustaleń ”czyste” DSP zajmuje niewielki ułamek nawet i tego sprzętu. Można było więc dojść do wniosku, że ten sprzęt prawdopodobnie będzie wystarczający na potrzeby tego projektu.

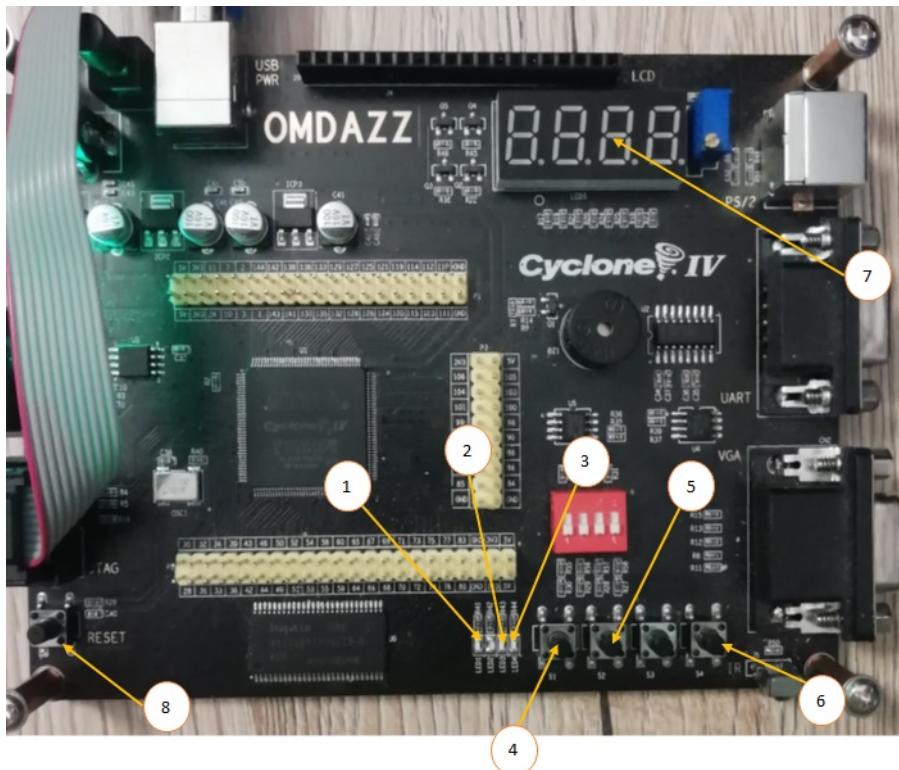
Następny rozdział opisuje przystosowanie tej platformy do uruchomienia projektu

## 6.2. Budowa urządzenia testowego

Na potrzeby uruchomienia na płytce projektowanego DSP, należało doprojektować dodatkowe układy oraz skonfigurować płytę w taki sposób, aby była możliwa manualna obsługa sprzętu przez potencjalnego użytkownika. Ten rozdział opisuje konfigurację elementów pod kątem obsługi układu oraz jego testów.

### 6.2.1. Elementy

Poniższe zdjęcie pokazuje elementy układu wraz z oznaczeniami numerycznymi elementów. Poniżej znajdują się opisy funkcjonalne każdego z elementów pod kątem obsługi sprzętu testowego.



Płytkę EP4CE6E22 wraz z oznaczeniami funkcji poszczególnych elementów

### **6.2.2. Funkcje przycisków**

Do obsługi układu przystosowano przyciski tak, aby było możliwe ręczne kontrolowanie procesu pomiaru:

4) Przycisk ten służy do zmiany trybów pomiarów, z trybu spowolnionego do trybu normalnego i na odwroć, w którym widać, że przetworzenie nie jest procesem, który jest momentalnym. Można go nazwać trybem 'podgladowym', który służy zespołowi do wychwycenia niuansów, które mogłyby nie być uwzględnione przez symulator ModelSim Altera. Głównym celem umieszczenia trybu spowolnionego jest to, aby proces pomiaru był widoczny gołym okiem, gdyż dla tempa 1MHz nie da się tego uchwycić. Tryb spowolniony spowalnia układ pomiarowy DSP wraz z układem emitującym laser 1024 razu, czyli do prędkości pomiaru z 1MHz do prędkości ok 1kHz.

5) Jest to przycisk imitujący włącznik układu. Jego naciśnięcie resetuje układ i zmienia status sygnału ena w układzie w zależności. Gdy ena jest w stanie wysokim, czyni ten stan stanem niskim, a gdy w stanie niskim, czyni ten stan stanem wysokim.

6) Jest to tryb podglądu odległości. Podczas naciskania tego przycisku pomiar wyświetlony za pomocą wyświetlacza 7-segmentowego zamienia się z numeru próbki na odległość jaką światło pokonuje w tym czasie. Aby dokonać pomiaru żądanej odległości w tym projekcie, trzeba do tej odległości dodać 37500 km (taką odległość jaką światło pokonuje przez 2s).

8) Przycisk RESET - służy do resetowania platformy pomiarowej.

### **6.2.3. Funkcje kontrolek**

Aby był widoczny stan układu, przystosowano do tego celu kontrolki :

1) Sygnalizuje prędkość pomiaru. Gdy dioda jest zapalona, oznacza to tryb spowolniony pomiaru (1 kHz), a gdy zgaszona- tryb normalny (1MHz). Na podstawie tej kontrolki można dowiedzieć się o aktualnie ustawionym trybie pomiaru oraz świadomie zmieniać ten tryb.

2) Sygnalizuje uruchomienie sprzętu, czyli gdy sygnał ena w sprzęcie jest w stanie wysokim.

3) Sygnalizuje zakończenie pomiaru. Zapala się gdy sygnał gotowości układu pomiarowego DSP jest w stanie wysokim; oznacza to zakończenie pomiaru oraz to, że wynik wyświetlony na wyświetlaczu jest tym ostatecznym.

### **6.2.4. Wyświetlacz 7-segmentowy**

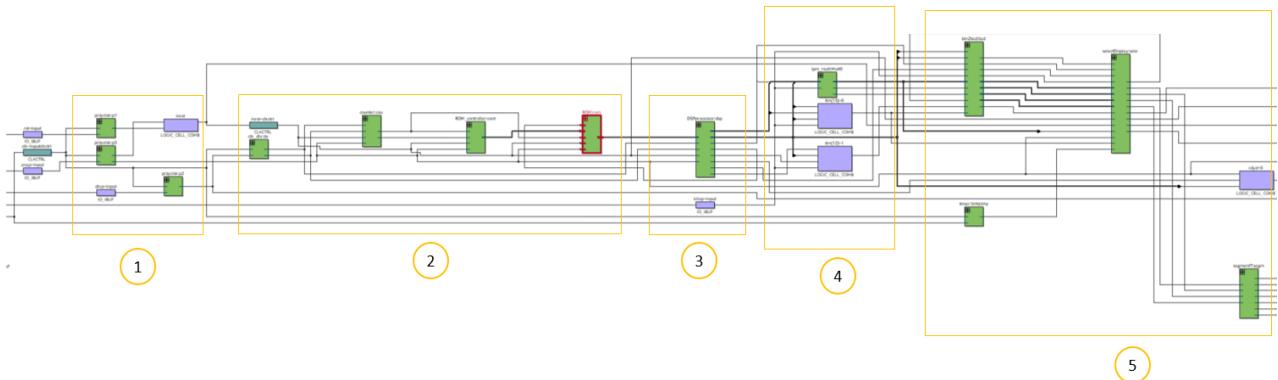
Wyświetlacz 7-segmentowy złożony z 4 bloków służy do wyświetlania wyników, które generuje ten układ. W założeniach:

- Wyświetla numer próbki, w której dotarł sygnał odbity
- Przy odpowiednich ustawieniach wyświetla odległość, jaką światło pokonuje w czasie trwania testu od początku do wychwycenia numeru próbki
- Wyświetla literę E przy wykryciu błędu

### 6.3. Budowa układu testowego

#### 6.3.1. Schemat

Poniżej znajduje się schemat budowy wewnętrznej układu pomiarowego. Numerami oznaczono bloki funkcjonalne, które mają określone role, w analizie sygnału



Uproszczony schemat budowy systemu utworzony przez narzędzie Technology Map Viewer  
Elementy opisano poniżej:

- 1) **Moduł wejścia** - odpowiada za prawidłową obsługę przycisków tak, aby były odporne na drgania płytka itd., aby jednoznacznie wykrywać wciśnięcia przycisków.
- 2) **Moduł imitera lasera** - odpowiada za imitowanie działania lasera, który nadaje sygnał o częstotliwości 1 MHz na podstawie danych umieszczonych w ROM-ie. Te dane zostaną użyte przez procesor DSP do przetworzenia i analizy.
- 3) **Moduł DSP** - procesor DSP. Jego funkcje są objaśnione w poprzednim rozdziale - Implementacja.
- 4) **Moduł obliczeniowy** - odpowiada za przeliczenia numeru próbki na odległość.
- 5) **Moduł wyjścia** - odpowiada za wyświetlenie statusu i odległości za pomocą kontrolerów i wyświetlacza 7-segmentowego.

Dzięki połączeniu tych 5 modułów układ może być obsługiwany manualnie oraz testowany na platformie docelowej. Dalsze podpunkty opisują szczegółowo poszczególne moduły.

#### 6.3.2. Moduł główny

Powyższy układ elementów zrealizowano za pomocą głównego kodu opisanego w języku Verilog. Poniżej znajduje się główny nadzędny moduł układu opisujący kluczowe połączenia w układzie:

```
module ProjektSYCYF(  
    input rst ,  
    input clk ,  
    input divp ,  
    input enap ,  
    input kilop ,
```

```

        output [6:0] sgm,
        output [3:0] sel,
        output rdyd,
        output divd,
        output enad
);

// Modul wejscia

wire clk1;

clk_div dv(.clk(clk),.clk1(clk1),.indiv(indiv));

wire press;
wire divpress;
wire enapress;
wire enarpress;

przycisk p1(.mode(0),.in(rst),.clk(clk),.press(press));
przycisk p2(.mode(1),.in(divp),.clk(clk),.press(divpress));
przycisk p3(.mode(1),.in(enap),.clk(clk),.press(enapress),.rspress(enarpress))

wire inrst = ~press || ~enarpress;
wire indiv = ~divpress;
wire ena = ~enapress;
wire kilo = ~kilop;

wire [12:0] addr;
wire clk0;
wire signed [7:0] rec;

// Modul przetwornika
counter cou(.ena(ena),.clk(clk1),.rst(inrst),.clk_0(clk0));

ROM rom(.clk(clk0),.addr(addr),.out(rec));
ROM_controller cont(.ena(ena),.clk(clk0),.rst(inrst),.addr(addr));

DSPprocessor dsp(.ena(ena),.rst(inrst),.clk(clk1),.rec(rec),.tim(timx),.rdy(rdy));

// Wykonanie przemnozen

wire signed [13:0] timx;
wire signed [26:0] tim1 = timx*13'b0010011001100;
wire signed [13:0] tim2 = tim1 >> 13;

```

```

wire signed [13:0] tim = (kilo)? tim2 : timx;

// Modul wyjścia
wire[15:0] out;
wire[3:0] num;
wire clk1khz;

bin2bcd bcd (.bin(tim), .bcd(out));

timer1kHz khz (.clk(clk), .clk0(clk1khz));

selectDisplay sele (.in(out), .clk0(clk1khz), .rst(inrst), .out(num), .sel(sel), .enad(enad));
segment7 segm (.bcd(num), .seg(sgm));

assign rdyd = ~(rdy && ena);
assign divd = divpress;
assign enad = enapress;
endmodule

```

Dane zgodnie z oczekiwaniami przepływają w poniższy sposób. Najpierw układ imitujący laser generuje sygnał na podstawie zakodowanych w pamięci ROM (o rozmiarze ok 64 kb), sygnał odpowiadający działaniu prawdziwego lasera, który w teorii miał być użyty w tym projekcie. Dane te trafiają do układu DSP i są przetwarzane poprzez operacje *korelacji* oraz obliczana jest klatka, w której światło wraca do lasera. Te układy są taktowane kontrolowanym przez dzielnik zegara. On odpowiada za ustawianie trybów spowolnionego oraz normalnego.

Następnie sygnał trafia do modułu obliczeniowego, który na podstawie sygnału nadanego za pomocą modułu wejścia oblicza numer próbki bądź odległość. Dane te trafiają do układu wyjścia, który odpowiada za wyświetlenie uzyskanych rezultatów za pomocą wyświetlacza 7-segmentowego albo kontrolek.

### 6.3.3. Układ DSP

Układ DSP jest tym samym układem, który został opracowany w etapie IV (Implementacja). Układ ten odpowiada za przetworzenie sygnału za pomocą operacji korelacji oraz zatrzaśnięciu numeru próbki, dla którego zostały spełnione warunki i dla których korelacja wskazuje, że światło dotarło do odbiornika. Numer próbki jest przekazywany jak w poprzednich podpunktach opisano dalej.

### 6.3.4. Przyciski

Aby obsługiwać przyciski dostępne na płytce, trzeba było stworzyć specjalny moduł obsługujący te komponenty. Ważne jest to, że te przyciski nie posiadają żadnych filtrów itd., które odpowiadałyby za przetwarzanie sygnałów potencjalnie niepożądanych (peaki itd.). Ważnym było więc, aby stworzyć taki układ odbiorczy, który zabezpieczy wnętrze układu przed wszelkimi zakłóceniami, potencjalnie tworzonymi przez nieprzetworzony sygnał przycisków. W niektórych przypadkach (np tryb przełączenia z kilometrów na numer próbki), można było to pominąć, gdyż ta zmiana nie tworzy zagrożenia dla danych zatrzaśniętych w sprzęcie. W pozostałych

przypadkach wymagane było stworzenie modułu, który zabezpieczy sygnały wewnętrz.

Układ, który opracowano do tego celu, nazwano roboczo "pompką". Układ działa w ten sposób, że gdy przycisk nie jest wciśnięty, licznik dodaje wartości, a gdy jest, odejmuje. Wtedy, gdy wartości znajdują się w żądanym przedziale :

- Gdy wartość licznika jest większa niż 24000 zmienia się wartość switcha.
- Gdy wartość jest mniejsza niż 10000 sygnał jest zerowany

Wskaźnik **mode** służy do określenia tego, czy przycisk ma być zatrzaszkowany; ma działać jak np. włącznik światła w pokoju lub ma działać jak przycisk w pilocie, wykrywać jedynie wciśnięcie. 0 oznacza tryb niezatrzaskujący, a 1 oznacza tryb zatrzaskujący.

Poniżej znajduje się kod:

```
module przycisk(
    input mode,
    input in,
    input clk,
    output press,
    output spress,
    output rpress
);

reg [17:0] lic = 18'b0;
reg [17:0] nlic = 18'b0;
reg snd = 1'b1;
reg nsnd = 1'b1;

reg switch = 1'b1;
reg nswitch = 1'b1;

always@ ( posedge clk ) begin
    snd <= nsnd;
    lic <= nlic;
    switch <= nswitch;
end

always@ ( * ) begin
    // Pompka przycisku
    nsnd = snd;
    nlic = 8'd0;
    nswitch = switch;

    if (in) begin
        if (nlic < 18'd25000) begin
```

```

        nlic = lic + 1;
    end

    if( nlic > 18'd24000) begin
        if( ~snd ) begin
            nswitch = ~switch;
        end

        nsnd = 1;
    end

end else
begin
    if( nlic > 0) begin
        nlic = lic - 1;
    end

    if( nlic < 18'd10000) begin
        nsnd = 0;
    end
end

if( ~mode) begin
    nswitch = nsnd;
end
end

assign rspress = snd;
assign press = switch;
assign spress = in;

endmodule

```

W ten sposób rozwiazuje się problem polegający na tym, że na sprzecie mamy odwrócone znaczenie sygnałów; stan wysoki oznacza przycisk niewciśnięty a stan niski przycisk wcisnięty. Dalej sygnały z tych modułów są przetwarzane za pomocą operacji *NOT* tak, aby uzyskać sygnały, które mogą być odbierane przez układy wewnętrzne. Ten moduł służy jako swoisty bufor danych wejściowych. Logicznym jest, że jest potrzebny taki moduł, gdyż jest potrzebna izolacja między światem zewnętrznym a wnętrzem układu. Jest więc to element pewnego rodzaju protokołu komunikacyjnego między światem a wnętrzem.

Odwracanie tych wartości opisano poniżej:

```

wire press;
wire divpress;
wire enapress;

```

```

wire enarpssess;

przycisk p1(.mode(0),.in(rst),.clk(clk),.press(press));
przycisk p2(.mode(1),.in(divp),.clk(clk),.press(divpress));
przycisk p3(.mode(1),.in(enap),.clk(clk),.press(enapress)),.rspress(er);

wire inrst = ~press || ~enarpssess;
wire indiv = ~divpress;
wire ena = ~enapress;
wire kilo = ~kilop;

```

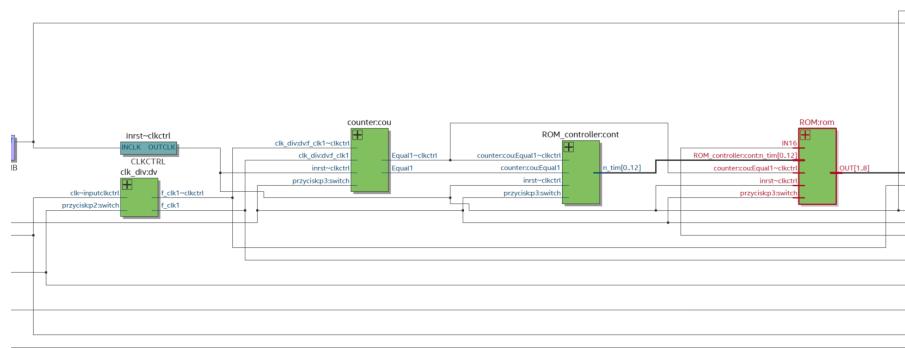
Sygnal rpress odpowiada w module przycisku za wykrywanie jedynie wciśnięcia przycisku nawet w przypadku układu ustawionego na tryb zatrzaszkiwania.

Ważnym aspektem projektowania układu było połączenie przycisku uruchamiającego z resetem. Oznacza to, że układ domyślnie podczas odblokowania jest resetowany, imitując w ten sposób przycisk zasilania.

### 6.3.5. ROM

Kolejnym modułem wchodzącym w skład układu jest moduł **imitera lasera**. Służy on do, jak nazwa wskazuje, imitowania sygnału nadawanego przez laser. Układ nadaje dane na podstawie ROMU, w którym jest zapisane wejście danych. Sposób przetworzenia tych danych do formy ROMU opisano w poniższych podpunktach.

Schemat budowy imitera znajduje się poniżej:



## Budowa modułu imitującego laser w układzie testowym [39]

Układ imitera składa się z 3 elementów

- **Podmodułu counter** służącego jako dzielnik częstotliwości, aby układ nadawał z częstotliwością 1 MHz, a nie z tą, jaką nadaje bezpośrednio płytka (50 MHz). Na układ ten wpływa też dzielnik zegara odpowiadający za tryby pracy układu. W trybie spowolnionym generuje on sygnał zegarowy o częstotliwości 1 kHz. Dzięki temu imiter może spełniać rolę przewidzianą w projekcie. Kod podmodułu znajduje się poniżej:

```
module counter(
```

```

        input  clk ,
        input  rst ,
        output [4:0]  tim ,
        output  clk_0
    );

reg [4:0]  r_tim=0;
reg [4:0]  n_tim=0;

always@ (posedge clk or posedge rst)begin
    if (rst) r_tim <= 5'b0;
    else if (ena) r_tim <= n_tim;
end

always@ (*) begin
    n_tim = r_tim + 1;
    if (n_tim==20) n_tim = 0;
end

assign tim = r_tim;
assign clk_0 = (r_tim == 0) ? 1 : 0;
endmodule

```

- Właściwego **ROM-u** imitowanego przez zaimplementowany RAM z danymi elementami (wg. implementacji pokazanej przez Technology Map Viewer). Zawiera on dane na temat imitowanego sygnału.
  - **Kontroler ROM-u.** Jest to licznik, który generuje kolejne adresy z częstotliwością nadaną przez podmoduł *counter* tak, aby finalnie na wyjściu powstał strumień danych, który będzie przetwarzany przez procesor DSP.
- Kod tego modułu znajduje się poniżej:

```

module ROM_controller(
    input  clk ,
    input  rst ,
    input  ena ,
    output [12:0]  addr
);

reg [12:0]  f_tim=1;
reg [12:0]  n_tim=0;

always@ (posedge clk or posedge rst)begin
    if (rst) f_tim <=13'b1;

```

```

        else if (ena) f_tim <= n_tim;

    end

    always@(*) begin
        if (f_tim < 5000) begin
            n_tim = f_tim + 1;
        end
    end

    assign addr = n_tim;

endmodule

```

Dzięki imiterowi powstaje sygnał wejściowy do modułu DSP, który imituje sygnał nadawany przez laser. Dalej tak, jak opisano w poprzednim rozdziale, układ przetwarza ten sygnał i wysyła go do modułów obliczeniowych.

### 6.3.6. Układ przeliczający działający w dwóch trybach

Sygnał wysłany przez moduł DSP, poprzez przetworzenie sygnału imitera, musi być przetworzony tak, aby możliwe było zinterpretowanie tych danych przez użytkownika. Zaprojektowano więc podmoduł, który na podstawie sygnału przycisku przełączania z sygnału kilometrów na sygnał lub numeru próbki, przekazuje do modułu wyjścia stosowne dane. Układ ten więc działa jak multiplekser. Wykorzystano tutaj arytmetykę stałoprzecinkową (*fixed point*), przez co przeliczenie nieco odbiega od policzenia tego na np. kalkulatorze. Do przemnożenia wykorzystano 13-bitowe rozwinięcie liczby 0,15 (gdyż taka odległość pokonuje światło tam i z powrotem, odbijając się przez czas  $1 \mu s$ ).

Przeliczenie tej odległości na kalkulatorze (tak jak w testach  $3001 * 0,15 = 450,15$ ), zaś układ wyrzuca 449. Uwzględniając jednak skalę, w jakiej wykonywany jest pomiar oraz zastosowaną technikę obliczeń, jest to błąd niewielki. W dalszych implementacjach tego układu można byłoby popracować nad dalszym usprawnieniem tego podmodułu, niemniej obliczenia uzyskane dzięki niemu są wystarczające.

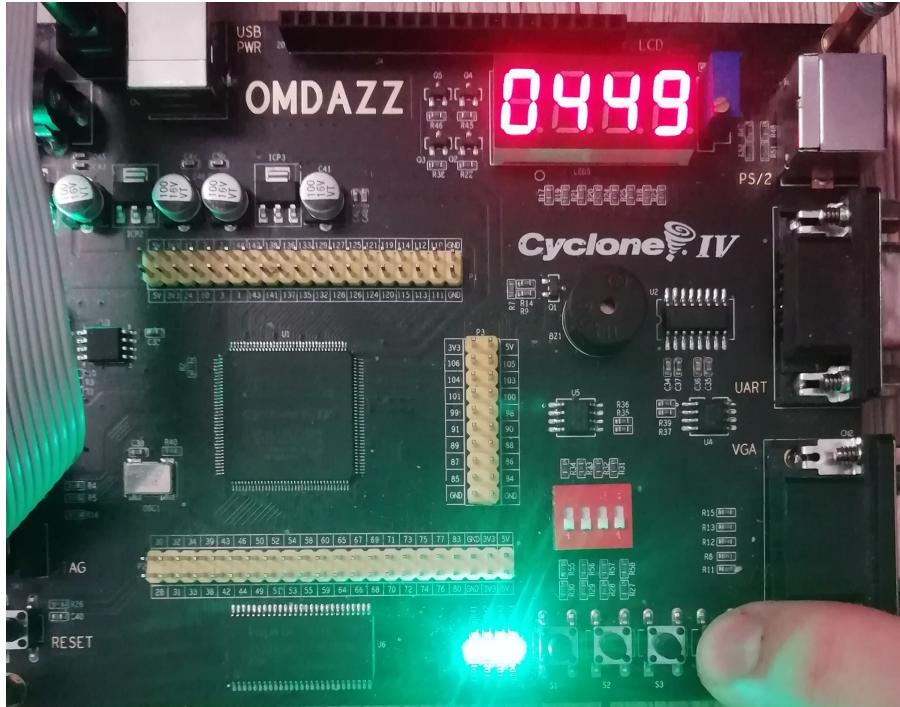
Poniżej znajduje się kod tego modulu:

```

// Wykonanie przemnozen
wire signed [13:0] timx;
wire signed [26:0] tim1 = timx*13'b0010011001100;
wire signed [13:0] tim2 = tim1 >> 13;
wire signed [13:0] tim = (kilo)? tim2 : timx;

```

Przesunięcie w lewo o 13 jest potrzebne do tego, aby pozbyć się części ułamkowej. Dzięki temu modułowi uzyskano taki odczyt w układzie. Jest on zgodny z oczekiwaniami:



Wyświetlenie dodatkowej drogi jaką pokonuje światło poza pierwszymi 2 sek

### 6.3.7. Wyświetlacz 7-segmentowy

Do wyświetlania danych służy 4-blokowy 7-segmentowy wyświetlacz pozwalający wyświetlać liczby z zakresu od 0-9999, uwzględniając ewentualne przecinki. Do obsługi tego modułu służy układ **segment7**, służący do przetworzenia znaków zakodowanych w kodzie 4-bitowym, na ich formy, które mogą być przetworzone przez wyświetlacz. Uznanano, że dla oszczędności zasobów w układzie powinny znajdować się maksymalnie małe formy pojedynczych znaków, zdecydowano się więc na taką formę kodowania. Oto spis znaków dostępnych jako swoisty *charset* dla tego systemu cyfrowego.:

Znak	Oznaczenie HEX	Oznaczenie BIN
0	0	0000
1	1	0001
2	2	0010
3	3	0011
4	4	0100
5	6	0101
6	6	0110
7	7	0111
8	8	1000
9	9	1001
E	a	1010

Kod modułu znajduje się poniżej. Jest to prosta tablica z zapisanymi wartościami 7-bitowymi znaków:

```

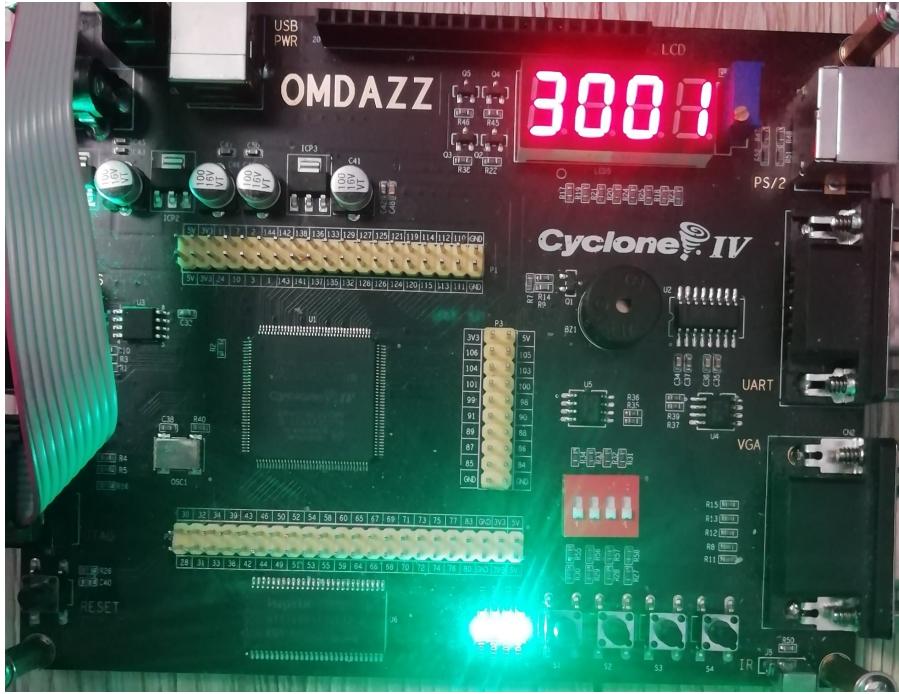
module segment7(
    input [3:0] bcd,
    output reg [6:0] seg
);

    always @(bcd)
    begin
        case (bcd) // case statement
            0 : seg = 7'b1000000;
            1 : seg = 7'b1111001;
            2 : seg = 7'b0100100;
            3 : seg = 7'b0110000;
            4 : seg = 7'b0011001;
            5 : seg = 7'b0010010;
            6 : seg = 7'b0000010;
            7 : seg = 7'b1111000;
            8 : seg = 7'b0000000;
            9 : seg = 7'b0010000;
            10: seg = 7'b0000110;
        default : seg = 7'b1111111;
    endcase
end

endmodule

```

Sygnał E służy do wyświetlenia błędu na wyświetlaczu. Z powodu dostępu do multipleksowanego wyświetlacza 7-segmentowego użyto tylko jednego takiego układu, nie każdego odzielnego dla każdej komórki wyświetlacza. Dzięki temu przetwarzaniu uzyskano wyświetlenie odległości takie jak na zdjęciu poniżej:



Wyświetlanie przez system numeru klatki [32]

### 6.3.8. Zamiana liczby binarnej na dziesiętną

Aby było możliwe wyświetlenie danych na takim wyświetlaczu, potrzebne jest przetworzenie danych z formy binarnej w kodzie U2 na formę BCD. Ponieważ dla wartości ujemnych przewidziano sygnał błędu (E), oznacza to, że implementowano jedynie wartości nieujemnych.

Do tego celu zastosowano algorytm Double Dabble, czyli algorytm ustępu do 3, małowymagającego algorytmu do przetwarzania liczb z formy dziesiętnej na formę BCD, która nadaje się do takiego przetwarzania w naszym sprzęcie.

Poniżej znajduje się kod takiego modułu realizującego procedurę double dabble:

```
module bin2bcd (
    input signed [13:0] bin ,
    output reg [15:0] bcd
);
```

```
reg [3:0] i ;
reg merr = 0;
//Double Dabble
always @(bin)
begin
```

```

bcd = 0;
for (i = 0; i < 14; i = i+1)
begin
  bcd = {bcd[14:0], bin[13-i]};

  if (i < 13 && bcd[3:0] > 4)
    bcd[3:0] = bcd[3:0] + 3;
  if (i < 13 && bcd[7:4] > 4)
    bcd[7:4] = bcd[7:4] + 3;
  if (i < 13 && bcd[11:8] > 4)
    bcd[11:8] = bcd[11:8] + 3;
    if (i < 13 && bcd[15:12] > 4)
      bcd[15:12] = bcd[15:12] + 3;
  end
  if (bin < 0) bcd = 16'b111111111111010;
end
assign err = merr;
endmodule

```

Układ takowy ma stosunkowo duże opóźnienie, ale nadal układ wraz z platformą testową ma  $f_{max}$  na poziomie 120 MHz, co absolutnie wystarcza dla naszych zastosowań. Sygnał na wyjściu nadaje oddzielone od siebie 4 cyfry w formacie 4 bitowym - daje to łącznie sygnał 16-bitowy.

### 6.3.9. Obsługa wyświetlacza multipleksowanego

Ponieważ zespół miał do czynienia z wyświetlaczem multipleskowanym potrzebny był moduł, który w pętli równomiernie rozkładałby liczby między komórki wyświetlacza, aby uzyskać niemigający i wyrazisty obraz. Aby diody mogły się wygasnąć między przejściami oraz uzyskać czytelny obraz, taki wyświetlacz musi być taktowany wolnym zegarem (na podstawie metody prób i błędów uzyskano częstotliwość na poziomie 1 kHz, tak aby obraz był jednoznaczny bez widocznych zbędnych artefaktów, np. pozostałości poprzednio niedogaszonego wcześniej sygnału itd.).

Opracowano więc kolejny dzielnik zegara, który spowalnia sygnał zegarowy do ok 762 Hz (65536 raza =  $2^{16}$ ). Kod tego dzielnika znajduje się poniżej:

```

module timer1kHz(
  input clk,
  output clk0
);

reg [16:0] f_tim=0;
reg [16:0] n_tim=0;

reg f_clk_0=0;
reg n_clk_0=0;

always@ (posedge clk) begin
  f_clk_0 <= n_clk_0 ;
  f_tim <= n_tim ;
end

```

```

always@(*) begin
    n_tim = f_tim + 1;
    n_clk_0 = f_clk_0;

    if (n_tim == 0) begin
        n_clk_0 = ~f_clk_0;
    end
end

assign clk0 = n_clk_0;

endmodule

```

Dzięki temu można było przejść do projektowania właściwego multipleksera połączonego z licznikiem, który rodziela sygnał 16-bitowy nadany przez układ zamiany liczby binarnej na liczbę BCD. Układ ten jest taktowany zegarem, z powyższego dzielnika i przekazuje sygnał do modułu *segment7*, który następnie jest przekazywany do wyświetlacza. Moduł ten generuje też sygnał do wyświetlacza multiplekowanego, który wybiera daną komórkę w wyświetlaczu.

Ponownie tutaj wyświetlacz 7-segmentowy odbiera odwrócone bity - (0 - lampka zapalona, 1 - lampka zgaszona). Tryb wybierania komórek wyświetlacza nie jest zoptymalizowany do 2-bitów, a wybiera się tę komórkę bezpośrednio. Do każdej komórki jest przyporządkowany oddzielny bit, przez co, aby wyświetlić liczbę na komórce nr 2, trzeba nadać sygnał 4-bitowy do selektora o wartości  $4'b1101$ .

Do tych zadań sporządzono układ **selectDisplay**, którego kod znajduje się poniżej:

```

module selectDisplay(
    input [15:0] in ,
    input clk0 ,
    input rst ,
    input ena ,

    output [3:0] out ,
    output [3:0] sel
);

reg [3:0] f_out=0;
reg [3:0] n_out=0;

reg [3:0] f_sel=0;
reg [3:0] n_sel=0;

reg [2:0] f_sn=0;
reg [2:0] n_sn=0;

always@(posedge clk0 or posedge rst)begin
    if (rst)begin

```

```

        f_sn <= 3'd0;
    end else
    begin
        f_sn <= n_sn;
    end
end

always@(*) begin
    n_sn = f_sn + 1;
end

always@(*) begin
    if(~ena || rst) n_sel = 4'b1111;
    else
        case(n_sn)
            0: n_sel = 4'b1110;
            1: n_sel = 4'b1110;
            2: n_sel = 4'b1101;
            3: n_sel = 4'b1101;
            4: n_sel = 4'b1011;
            5: n_sel = 4'b1011;
            6: n_sel = 4'b0111;
            7: n_sel = 4'b0111;
        endcase
end

always@(*) begin
    case(n_sn)
        0: n_out = in[3:0];
        1: n_out = in[3:0];
        2: n_out = in[7:4];
        3: n_out = in[7:4];
        4: n_out = in[11:8];
        5: n_out = in[11:8];
        6: n_out = in[15:12];
        7: n_out = in[15:12];
        default: n_out = 4'b1111;
    endcase
end

assign out = n_out;
assign sel = n_sel;
endmodule

```

### 6.3.10. Komunikacja błędów

Według założeń z poprzednich rozdziałów, układ powinien informować o potencjalnych błędach (np. ujemna klatka). Jest to możliwe, gdyż taką wartość założono projektując procesor

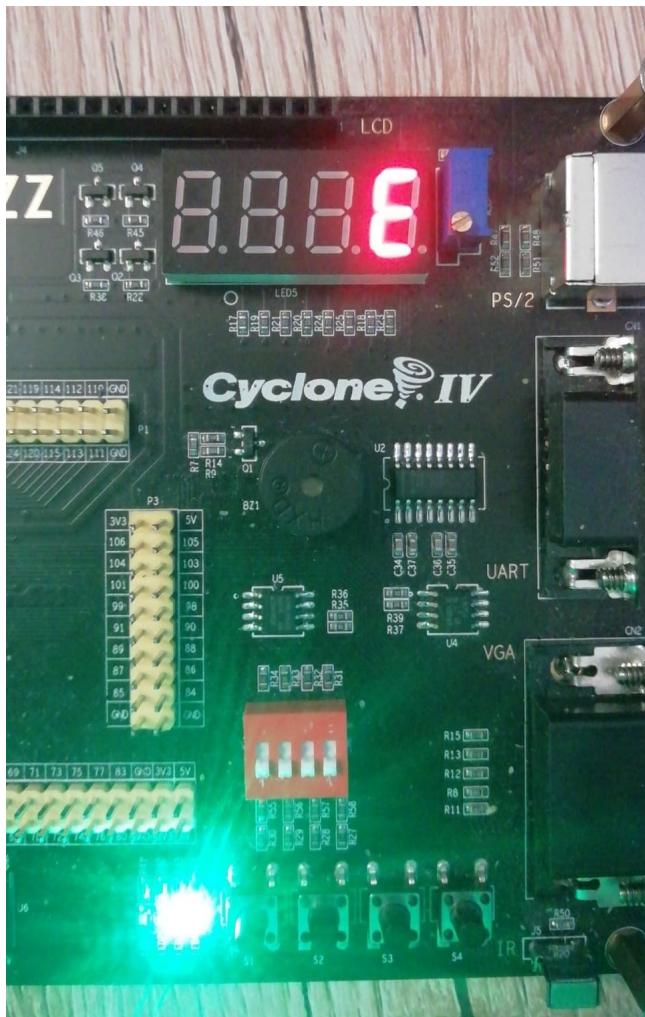
DSP. Jednym z celów tego układu było opracowanie modułu, który wychwyciłby ten błąd. Fragmentem wykrywającym to znajduje się w module *Double Dabble*, który przedstawia się w poniższy sposób:

```

...
    bcd [11:8] = bcd [11:8] + 3;
                                if ( i < 13 && bcd [15:12] > 4 )
    bcd [15:12] = bcd [15:12] + 3;
end
                                if ( bin < 0) bcd=16'b111111111111010 ;
end
assign err= merr;
endmodule

```

W trakcie przetwarzania, gdy wykryto ujemną wartość, przekazywany do multipeleksera jest sygnał E, który wyświetla na wyświetlaczu mutipleksowanym literę E tak jak na zdjęciu poniżej.



Wyświetlanie błędu przez system [32]

Dzięki temu układ informuje o potencjalnych błędach. Nadal jednak nie ma takiej blokady dla przeliczeń bitowych (na kilometry), więc po wciśnięciu przycisku przełączającym na tryb kilometrów wyświetli się nieprawidłowa wartość. Nadal jednak po jego puszczeniu wyświetla się błąd, co jest podstawą do zignorowania takiego wyniku.

## 6.4. Dane wejściowe

### 6.4.1. Przetworzenie danych wejściowych

Aby utworzyć ROM, użyty w układzie imitera lasera, postanowiono utworzyć plik .v z zawartością takowego ROM-u. Do przetworzenia sygnałów zapisanych w formacie dziesiętnym z przecinkiem do formy takiej, aby z niej utworzyć ROM w formacie **fixed point**, trzeba było zastosować algorytm. Do tego użyto załączonego również do repozytorium tego projektu skryptu w języku *Python*. Dzięki temu otrzymano skrypt w Verilog HDL, który zawiera stosowny ROM.

Poniżej znajduje się kod tego skryptu, pozwalający na takowe przetworzenie:

```
linijki = []
index = 0;

def decimalToBinary(n):
    return bin(n).replace("0b", "")

def presentValue(el, num, name):
    value = int(el * 128);
    tex = ""

    if value >= 0:
        tex = decimalToBinary(value) + ""
    else:
        tex = decimalToBinary(value & 0xff) + ""
    for n in range(8 - len(tex)):
        tex = "0" + tex

    time = num * 40
    linijki.append(tex)

plik = open(r"received8.txt", "r")
plik2 = open(r"sent8.txt", "r")

lista= []
listasin= []

for elementy in plik:
    try:
        lista.append(float(elementy))
    except Exception:
        print("")

for elementy in plik2:
    try:
        listasin.append(float(elementy))
```

```

except Exception :
    print("")

plik.close()
plik2.close()

num = 0
sinn = 0

lin = 0

for el in lista:

    presentValue( el ,num," rec" )

    num = num + 1

write = open("output.txt","w")
for linia in linijki:
    write.write(linia+"\n")

write.close()

```

Dzięki temu skryptowi uzyskano układ w języku opisu sprzętu, który mógł zostać wstawiony do kodu.

#### 6.4.2. Postać ROMu w Verilogu

Dzięki uruchomieniu skryptu w środowisku Python 3.3.8, uzyskano kod w języku opisu sprzętu Verilog HDL, w poniższej postaci. Postanowiono nie wstawiać całego kodu, gdyż zajmuje on ok 5000 linii, a z perspektywy analizy jedynymi istotnymi danymi są początek i koniec kodu, dzięki któremu możemy analizować daną strukturę takiego pliku.

Kod wygenerowany przez skrypt został zaprezentowany poniżej:

```

module ROM(
    input  clk ,
    input [12:0]  addr ,
    output signed [7:0]  out
);

    reg signed [7:0]  m_out;

    always@( posedge  clk )
        case( addr )
            13'd0: m_out<=8'b11100100 ;
            13'd1: m_out<=8'b11110000 ;
            13'd2: m_out<=8'b01000110 ;
            13'd3: m_out<=8'b11101101 ;

```

```

13'd4: m_out<=8'b00100000;
13'd5: m_out<=8'b11101110;
13'd6: m_out<=8'b11100111;
13'd7: m_out<=8'b00011000;
...
13'd4987: m_out<=8'b11101101;
13'd4988: m_out<=8'b00010010;
13'd4989: m_out<=8'b00010011;
13'd4990: m_out<=8'b11111001;
13'd4991: m_out<=8'b11111100;
13'd4992: m_out<=8'b10111011;
13'd4993: m_out<=8'b111110010;
13'd4994: m_out<=8'b00010000;
13'd4995: m_out<=8'b11110111;
13'd4996: m_out<=8'b111110100;
13'd4997: m_out<=8'b00010000;
13'd4998: m_out<=8'b11100100;
13'd4999: m_out<=8'b11100111;
default: m_out <=8'b0;
endcase
assign out = m_out;
endmodule

```

Dzięki temu plikowi można było wstawić taki plik do układu jako pamięć ROM, która może być użyta jako źródło danych dla emitera.

#### 6.4.3. Oczekiwane działanie modułu imitującego wyjście lasera

Oczekiwany skutkiem "wstawienia" tego ROM-u jest nadawanie sygnału takiego jak podczas przeprowadzania testbencha czystego układu w poprzednim etapie (IV Implementacja). Na płytce powinno uzyskać, zarówno w trybie spowolnionym, jak i normalnym, zapaloną diodę gotowości i liczbę 3001, oznaczającą numer próbki.

### 6.5. Uruchomienie

W tym podrozdziale opisano kluczowy fragment tego etapu- uruchomienie na płytce układu.

#### 6.5.1. Procedura uruchomienia na płytce

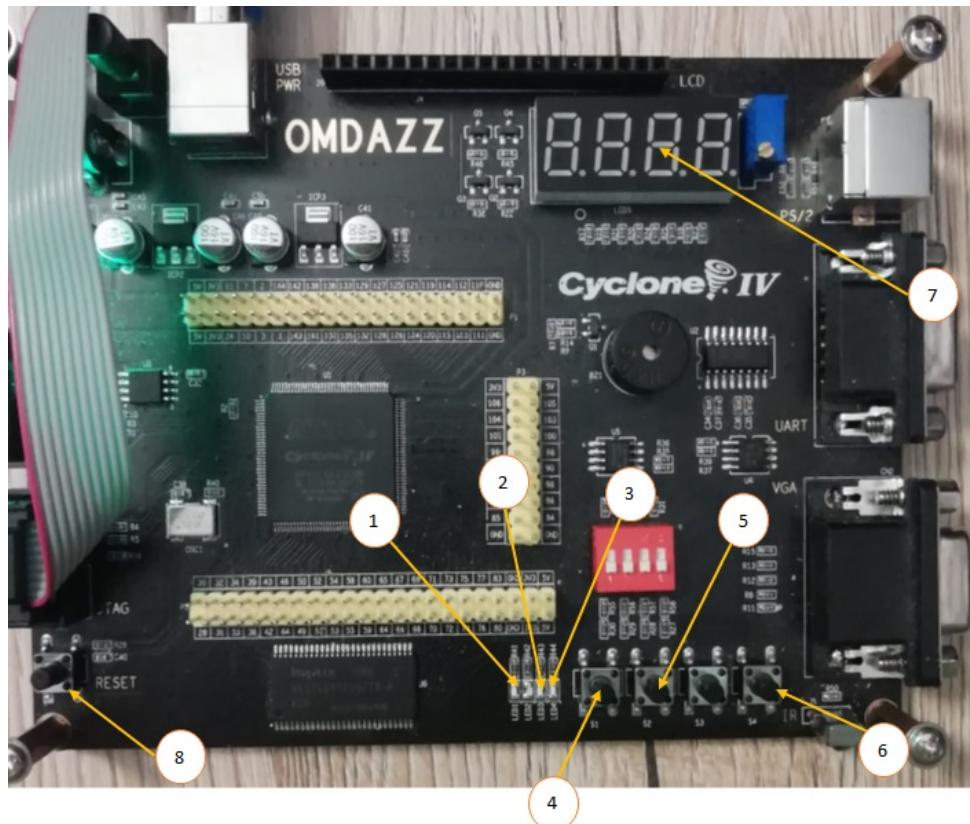
Po skompilowaniu projektu w programie Quartus wykonano następujące kroki:

- Uruchomiono narzędzie Programmer w programie Quartus.
- Podłączono do komputera płytkę EP4CE6E22 oraz wybrano ją w programie za pomocą funkcji "Hardware Setup" za pomocą urządzenia USB-Blaster
- Sprawdzono, czy komputer prawidłowo rozpoznał model płytka.
- Zaznaczono funkcję Program/Configure.
- Kliknięto Start.

Następnie należało odczekać moment, aby komputer za pomocą USB-Blastera zaprogramował nasz układ programowalny (Pasek ładowania cały zapełni się na zielono z napisem "100% (Successful)"). Za pomocą JTAGa zaprogramowano płytkę. Następnie zespół przeszedł do właściwych testów na sprzęcie.

### 6.5.2. Procedura testu

Po uruchomieniu płytki zespół przeszedł do właściwych testów. Poniżej zajduje się płytką wraz z oznaczeniami komponentów, aby łatwiej było analizować kolejne kroki testu:



Płytkę wraz z oznaczeniami komponentów

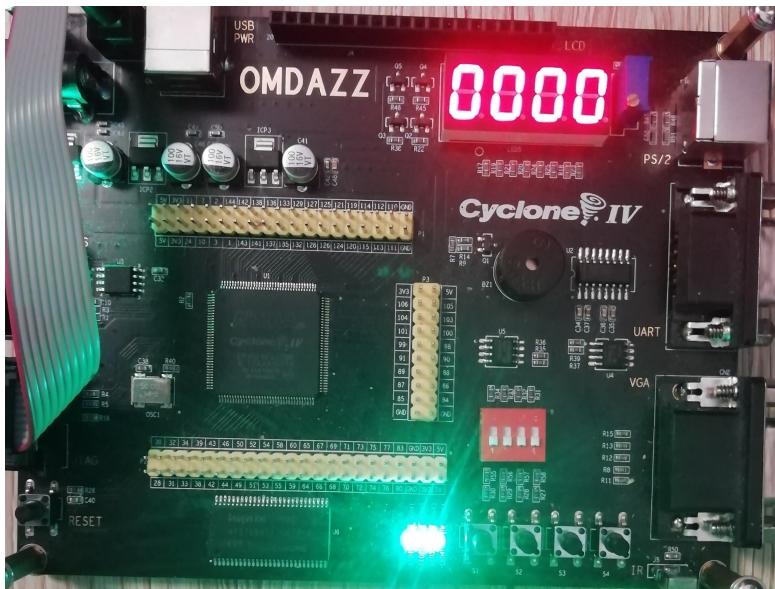
Układ uruchomiono za pomocą przycisku 5. Układ momentalnie wyświetlił pewną wartość oraz zasygnalizował zakończenie pomiaru (dioda **nr 3**). Następnie wciśnięto przycisk zmiany trybu z numeru klatki na kilometry (przycisk **nr 6**). Po zweryfikowaniu wartości wciśnięto przycisk zmiany trybu na spowolniony (za pomocą przycisku **nr 4**). Po zapaleniu się lampki sygnalizującej tryb spowolniony (lampka **nr 1**), zresetowano układ za pomocą przycisku **8**. Następnie odczekano pewien moment, aż do uzyskania w trybie spowolnionym wyniku. Ponownie też wykonano sprawdzenie przeliczenia odległości z numeru próbki, wciskając przycisk **nr 6**.

Dzięki temu wykonano żądane testy. Następnie wyłączono układ za pomocą przycisku **nr 5**, a następnie zespół przeszedł do analizy sfotografowanych podczas testu kroków tego testu.

### 6.5.3. Opis uzyskanych rezultatów

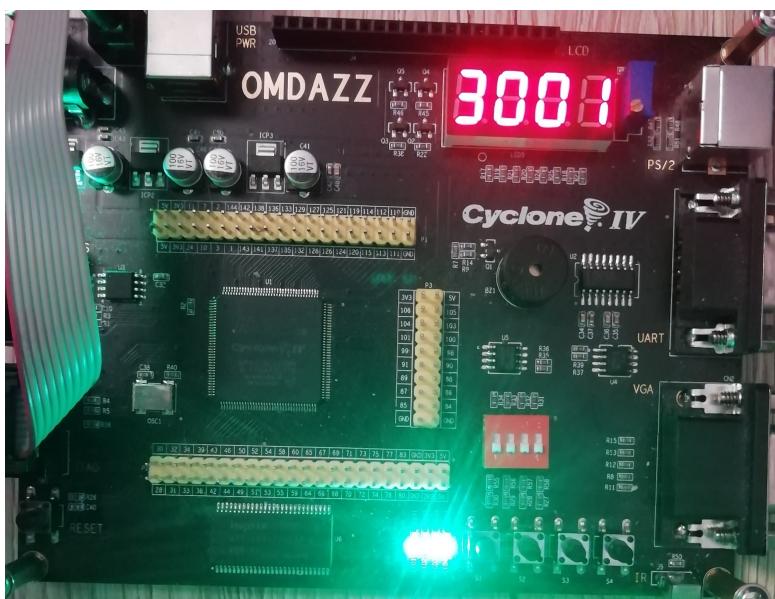
Dzięki powyższym krokom uzyskano wyniki testów. Test dokumentowano za pomocą fotografii wykonanych na telefonie. Po uruchomieniu układu uzyskano na wyświetlaczu 7-segmentowym poniższe rezultaty.

Wyświetliła się wartość **3001**. Następnie dokonano odczytu odległości przeliczanej przez układ. Uzyskano wartość 449. Następnie powtórzono testy w trybie spowolnionym, uzyskując identyczne wyniki (3001 próbka, 449 km). Uzyskano te wyniki po wyraźnie dłuższym czasie niż wcześniej; zamiast momentu test ten trwał ok 1,5 s. W trakcie oczekiwania wyświetlona na ekranie była wartość 0000, tak jak na zdjęciu poniżej:

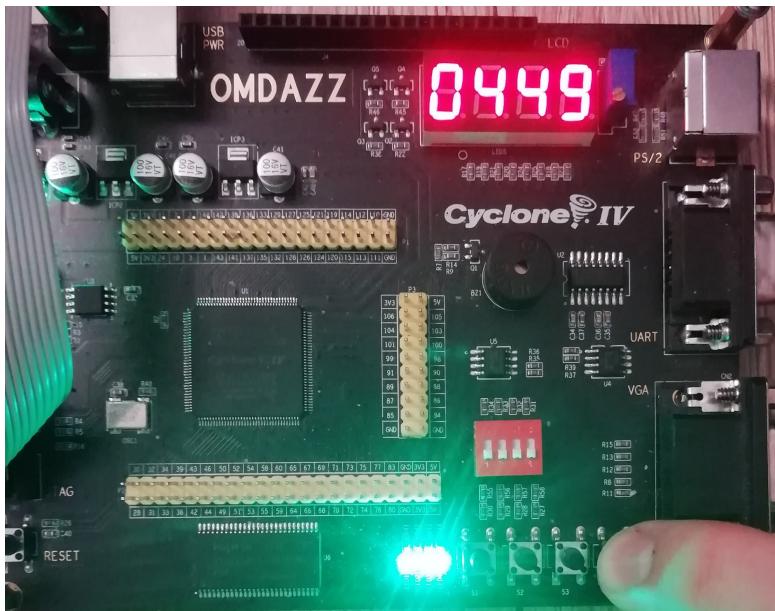


Oczekiwanie

Poniżej znajdują się zdjęcia z testów wykonanych w trybie spowolnionym:



Wynik testu pomiaru klatki



Wynik testu pomiaru odległości dodatkowej

## 7. Podsumowanie

### 7.1. Wnioski

Uruchomienie dało spodziewane efekty. Oznacza to, że układ działa prawidłowo i nie posiada istotnych wad technicznych, które mogłyby wpływać na błędny pomiar. Wynik jest zgodny z wynikami wcześniejszego benchtestu, wykonanego w etapie poprzednim (IV). Oznacza to, że uzyskano prawidłowy układ, do realizacji zadania postawionego zespołowi.

Wykonano też pozostałe zadania :

- Wyznaczono potencjalny obiekt do którego wykonywano pomiar (Księzyc)
- Udało się wykonać symulacje potwierdzające ustalenia układu
- Udało się wykonać modele referencyjne tego układu.

Wykonanie projektu można uznać więc za sukces zespołu projektowego.

#### 7.1.1. Raport na temat zajętości zasobów

Na podstawie poniższego raportu na temat zajętości zasobów, układ DSP wraz z układami wspomagającymi pomiar zajmuje 1387 bloków logicznych co stanowi 22% zasobów, w tym 295 rejestrów. Układ wykorzystuje 19 pinów wyjścia do komunikacji z komponentami. Wykorzytano również 64 kb (8 kB) pamięci wbudowanej.

Jest to jedynie 1/5 całości układu FPGA. Oznacza to że można w tym układzie zmieścić około 4 takich układów do wykonywania pomiarów, aby wystarczyło zasobów. Zaprojektowany przez zespół układ nadaje się do implementacji na tym sprzęcie.

Poniżej znajduje się raport komplikacji na temat zajętości zasobów:

Flow Status	Successful - Sat Jun 05 10:31:22 2021
Quartus Prime Version	20.1.1 Build 720 11/11/2020 Patches 1.49 SJ Lite Edition
Revision Name	ProjektSYCYF
Top-level Entity Name	ProjektSYCYF
Family	Cyclone IV E
Device	EP4CE6E22C8
Timing Models	Final
Total logic elements	1,387 / 6,272 ( 22 % )
Total registers	295
Total pins	19 / 92 ( 21 % )
Total virtual pins	0
Total memory bits	65,536 / 276,480 ( 24 % )
Embedded Multiplier 9-bit elements	0 / 30 ( 0 % )
Total PLLs	0 / 2 ( 0 % )

Raport na temat zajętości zasobów

#### 7.1.2. Raport na temat maksymalnej częstotliwości

Na podstawie poniższego raportu oceniono  $f_{max}$  (maksymalna bezpieczna częstotliwość taktowania) na 115,96 MHz. Ponieważ układ jest taktowany zegarem 50 MHz, umożliwia to umieszczenie tego układu na płytce o takim taktowaniu. Zaprojektowany przez zespół układ nadaje się więc do implementacji na tym sprzęcie

Poniżej znajduje się raport komplikacji na temat maksymalnej częstotliwości taktowania.

	Fmax	Restricted Fmax	Clock Name	Note
1	115.96 MHz	115.96 MHz	clk	
2	158.98 MHz	158.98 MHz	ROM_controll...ntjf_tim[10]	
3	254.58 MHz	254.58 MHz	timer1kHzkhz f_clk_0	
4	358.55 MHz	238.04 MHz	DSPprocess...majr_tim[0]	limit due to minimum period restriction (tmin)
5	414.08 MHz	402.09 MHz	clk_div:dv f_clk1	limit due to minimum period restriction (tmin)

### Raport na temat wyznaczania maksymalnej częstotliwości

Na większą część spadku tej częstotliwości w porównaniu z tym układem wpływa dołożenie nowych podukładów przystosowujących ten układ do obsługi manualnej oraz zaimplementowaniem dużej ilości funkcji dodatkowych (np. tryb spowolniony czy czytanie z ROM-u). Gdyby nie to, układ mógłby być szybszy, niemniej nawet taki nadaje się w zupełności do wykonania zadania pomiaru dużych odległości.

#### 7.1.3. Co udało się wykonać

- Wykonanie sprawnego układu
- Realizacja zadania pomiaru tejże odległości
- Wyznaczenie potencjalnego obiektu dla którego mierzono odległość (Księżyc)
- Wykonanie prototypu sprzętu obsługującego przyciski diody itp.

#### 7.1.4. Czego nie udało się wykonać

- Nie udało się lepiej zoptymalizować całego układu (prawdopodobnie da się uzyskać wyższą częstotliwość układu niż 115 MHz).
- Nie udało się wykonać w pełni obsługi błędów (Np wyświetlanie przy błędzie ujemnej odległości obliczanej odległości).
- Nie udało się wykonać takiego układu wraz z sprzęttem docelowym (z prawdziwym laserem, w który emitowałby takie sygnały jak w założeniu zadania i jakie są emitowane przez imitator lasera)
- Nie udało się wykonać precyzyjniejszego pomiaru odległości przez moduł obliczeniowy (zamiast 449 uzyskano 450km).

#### 7.1.5. Przyczyny

- Nie wykonano testów na docelowej płytce FPGA, gdyż zespół nie posiadał takowego sprzętu przy sobie (zdalna realizacja zajęć), niemniej wykonano testy na zamienniku.
- Brak w obecnym momencie stosownej wiedzy (zespół składa się z początkujących "programistów" FPGA)

## 7.2. Zaangażowanie poszczególnych studentów

Poniższa tabela reprezentuje zaangażowanie każdego z twórców raportu:

Etap	Jędrzej Joniec	Michał Kamiński	Michał Podgajny	Maciej Antosz	Sebastian Skrzek
1.	20%	20%	20%	20%	20%
2.	20%	20%	20%	20%	20%
3.	20%	20%	20%	20%	20%
4.	20%	20%	20%	20%	20%
5.	20%	20%	20%	20%	20%
całość	20%	20%	20%	20%	20%

## Literatura

- [1] TeachThought. 3 Modes Of Thinking: Lateral, Divergent Convergent Thought. <https://www.teachthought.com/critical-thinking/3-modes-of-thought-divergent-convergent-thinking/>.
- [2] Fred Wilson. Convergent Thinking vs. Divergent Thinking: Why only Planning for a Project isn't always enough? <https://www.ntaskmanager.com/blog/convergent-thinking-vs-divergent-thinking/>, 2020.
- [3] JustInMind. The Double Diamond model: what is it and should you use it? <https://www.justinmind.com/blog/double-diamond-model-what-is-should-you-use/>, 2018.
- [4] Design Council. What is the framework for innovation? Design Council's evolved Double Diamond. <https://www.designcouncil.org.uk/news-opinion/what-framework-innovation-design-councils-evolved-double-diamond>, 2019.
- [5] Nicholas Lee. Introducing Design Thinking. <https://www.northstarhub.com/posts/introducing-design-thinking>, 2020.
- [6] John Hammersley and John Lees-Miller. 6 Successful Methods For Brainstorming Idea Creation. <https://morethandigital.info/en/6-successful-methods-brainstorming-idea-creation/>, 2021.
- [7] Kanbanize. 5 Whys: The Ultimate Root Cause Analysis Tool. <https://kanbanize.com/lean-management/improvement/5-whys-analysis-tool>.
- [8] Wikipedia. Trial and error. [https://en.wikipedia.org/wiki/Trial\\_and\\_error#:~:text=Trial%20and%20error%20is%20a,until%20the%20practicer%20stops%20trying.](https://en.wikipedia.org/wiki/Trial_and_error#:~:text=Trial%20and%20error%20is%20a,until%20the%20practicer%20stops%20trying.), 2021.
- [9] Microsoft. Microsoft Teams. <https://www.microsoft.com/en-ww/microsoft-teams/group-chat-software>, 2021.
- [10] John Hammersley and John Lees-Miller. OverLeaf. <https://www.overleaf.com/>, 2021.
- [11] Python Software Foundation. Python 3.9.2. <https://www.python.org/>, 2021.
- [12] Intel. Download Center for FPGAs. <https://fpgasoftware.intel.com/?edition=lite>, 2021.
- [13] Terasic. Altera DE2-115 Development and Education Board. [https://www.terasic.com.tw/cgi-bin/page/archive\\_download.pl?Language=English&No=502&FID=cd9c7c1fea2467c58c9aa4cc02131af](https://www.terasic.com.tw/cgi-bin/page/archive_download.pl?Language=English&No=502&FID=cd9c7c1fea2467c58c9aa4cc02131af), 2020.
- [14] OpenStax Poland. Rozchodzenie się światła. <https://cnx.org/contents/u2KTPvIK@6.16:Zv6FJYpb@4/1-1-Rozchodzenie-si%C4%99-%C5%9Bwiat%C5%982a>, 2021.
- [15] Tim Sharp. How Far is the Moon? <https://www.space.com/18145-how-far-is-the-moon.html>, 2017.
- [16] Wikipedia. Procesor sygnałowy. [https://pl.wikipedia.org/wiki/Procesor\\_sygna%C5%82owy](https://pl.wikipedia.org/wiki/Procesor_sygna%C5%82owy), 2021.
- [17] eXtronic. Co to jest FPGA? <http://extronic.pl/content/33-co-to-jest-fpga>.
- [18] Adam Bemski. Kurs FPGA – 2 – podstawowe pojęcia, porównanie z uC. <https://forbot.pl/blog/kurs-fpga-podstawowe-pojecia-roznice-wzgledem-uc-id7061>, 2017.
- [19] Clive Maxfields. The design warrior's guide to FPGAs: devices, tools and flows, 2004.
- [20] Marcin Karbowniczek. FPGA od początku do końca - część pierwsza. <https://elektronikab2b.pl/technika/1315-fpga-od-poczatku-do-konca-czesc-pierwsza>, 2007.
- [21] Język opisu sprzętu - Hardware description language. [https://pl.qaz.wiki/wiki/Hardware\\_description\\_language](https://pl.qaz.wiki/wiki/Hardware_description_language).
- [22] Verilog - Verilog Verilog - <https://pl.qaz.wiki/wiki/Verilog>. <https://pl.qaz.wiki/wiki/Verilog>.
- [23] VHDL - VHDL. <https://pl.qaz.wiki/wiki/VHDL>.
- [24] Wikipedia. Pamięć statyczna. [https://pl.wikipedia.org/wiki/Pami%C4%99%C4%87\\_statyczna](https://pl.wikipedia.org/wiki/Pami%C4%99%C4%87_statyczna), 2021.
- [25] Wikipedia. Pamięć dynamiczna. [https://pl.wikipedia.org/wiki/Pami%C4%99%C4%87\\_dynamiczna\\_\(informatyka\)](https://pl.wikipedia.org/wiki/Pami%C4%99%C4%87_dynamiczna_(informatyka)), 2020.
- [26] The ATP Group. Understanding RAM and DRAM Computer Memory Types. <https://www.atpinc.com/blog/computer-memory-types-dram-ram-module>, 2018.
- [27] Grzegorz Kowal. Co to jest CPU. Co należy wiedzieć? <https://dobrykomputer.info/poradnik/co-to-jest-cpu-co-nalezy-wiedziec>, 2015.
- [28] Elektroniczne Techniki Pomiarowe, Notatki w internecie. Dalmierze elektromagnetyczne – zasada działania i klasyfikacja. [https://zasoby1.open.agh.edu.pl/dydaktyka/automatyka/c\\_elektroniczna\\_techn\\_pomiarowa/w12.htm](https://zasoby1.open.agh.edu.pl/dydaktyka/automatyka/c_elektroniczna_techn_pomiarowa/w12.htm).
- [29] Zhu Jingguo Ren Jianfeng Xiao Fang Meng zhe Li Menglin1, Xie Wenfeng. THE ANALYSIS OF DIGITAL PHASE-SHIFT MEASURING METHODS FOR DUAL-FREQUENCY LASER RANGE FINDER. <https://www.imeko.org/publications/tc14-2014/IMEKO-TC14-2014-27.pdf>, 2014.
- [30] kartex.eu. Dalmierze elektromagnetyczne. <https://www.google.pl/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwijub2nyevvAhWpAhAIHeZaALUQFjAHegQIBhAD&url=http%3A%2F%2Fkartex.eu%2Findex.php%3Faction%3Dtpmod%3Bd1%3Dget105&usg=A0vVaw0mCc1JkSd56J7FhSUtqYvU,->.
- [31] PWN. Analiza spektralna. <https://dobrykomputer.info/poradnik/co-to-jest-cpu-co-nalezy-wiedziec>, 2021.
- [32] Wikipedia.com. Widmo osiem.jpg. [https://pl.m.wikipedia.org/wiki/Plik:Widmo\\_osiem.jpg](https://pl.m.wikipedia.org/wiki/Plik:Widmo_osiem.jpg), 2014.
- [33] The fourier transform .com. <https://www.thefouriertransform.com/transform/fourier.php>, 2011.
- [34] cp-algorithms.com. Fast Fourier transform. <https://cp-algorithms.com/algebra/fft.html>, 2021.

- [35] Wikipedia. Cooley–Tukey FFT algorithm. [https://en.wikipedia.org/wiki/Cooley%E2%80%93Tukey\\_FFT\\_algorithm](https://en.wikipedia.org/wiki/Cooley%E2%80%93Tukey_FFT_algorithm), 2021.
- [36] Pythonic Perambulations. Understanding the FFT Algorithm. [http://dsp-book.narod.ru/FFTBB/0270\\_PDF\\_C14.pdf](http://dsp-book.narod.ru/FFTBB/0270_PDF_C14.pdf), 2013.
- [37] CRC Press LLC. Cooley–Tukey FFT algorithm. [http://dsp-book.narod.ru/FFTBB/0270\\_PDF\\_C14.pdf](http://dsp-book.narod.ru/FFTBB/0270_PDF_C14.pdf), 2000.
- [38] SciPy.org. Fourier Transforms (scipy.fft). <https://docs.scipy.org/doc/scipy/reference/tutorial/fft.html>, 2021.
- [39] GitHub.com. Repozytorium projektu SYCYF. <https://github.com/mpdg837/ProjektSYCYF>, 2014.