

ZACHODNIOPOMORSKI UNIWERSYTET TECHNOLOGICZNY
W SZCZECINIE

Wydział Informatyki



Łukasz Stoleman

Kierunek Informatyka

**Dekoder sygnału wzorca czasu DCF77
z wykorzystaniem radia programowalnego
i FPGA**

Praca dyplomowa inżynierska
napisana pod kierunkiem
dra Remigiusza Olejnika
w Katedrze Architektury Komputerów
i Telekomunikacji

Szczecin, 2016

Streszczenie

Praca dyplomowa obejmuje badania zagadnień przetwarzania sygnałów, w szczególności radiowych. Prezentowany projekt ma na celu zastosowanie techniki programowalnego radia w środowisku FPGA. Implementacja sprzętowa to zaprojektowany przez autora schemat ideowy oraz płytką drukowaną, w której użyto układów radiowych, zadaniem których jest wstępna obróbka oraz przetworzenie sygnału z postaci analogowej do cyfrowej. Implementacja programowa to projekt FPGA w języku opisu sprzętu Verilog, w którym zapisano algorytm przetwarzający sygnał wejściowy. Ostatnim etapem jest aplikacja PC w języku Python, która interpretuje przesłane przez układ dane.

Abstract

The thesis covers research on issues of digital signal processing, especially wireless and radio signals. The project has a goal to apply software defined radio technique in FPGA environment. Hardware implementation designed by the author consists schematic design and a printed circuit board in which he uses several integrated circuits which aim is to preliminary processing of a signal from analog to digital form. Software implementation is a FPGA project in Verilog hardware description language in which an algorithm of processing input signal is written. Finally, an application in Python language reads data sent by PCB to a PC and converts it into readable form.

Oświadczenie

Oświadczam, że przedkładaną pracę dyplomową napisałem samodzielnie. Oznacza to, że przy pisaniu pracy poza niezbędnymi konsultacjami, nie korzystałem z pomocy innych osób, a w szczególności nie zlecałem opracowania pracy lub jej części innym osobom oraz nie przypisałem sobie autorstwa istotnego fragmentu lub innych elementów cudzego utworu lub ustalenia naukowego. Potwierdzam też zgodność wersji papierowej i elektronicznej złożonej pracy.

Mam świadomość, że poświadczenie nieprawdy będzie w tym przypadku skutkowało cofnięciem decyzji o wydaniu dyplomu.

Łukasz Stoleman

Spis treści

Wstęp	5
Cel i zakres pracy	5
Rozdział 1. Software Defined Radio	7
1.1. Teoria	7
Rozdział 2. DCF77	9
2.1. Sygnał i jego właściwości	9
2.2. Kodowanie nadawanego komunikatu	9
Rozdział 3. Sprzęt – projekt	14
3.1. Moduły sprzętowe	14
3.1.1. Płytki bazowa FPGA	14
3.1.2. Schemat ideowy	15
3.1.3. Płytki drukowana	19
Rozdział 4. Oprogramowanie – moduły, algorytmy	21
4.1. Architektura programowa	21
4.2. Moduły	22
4.2.1. PLL	22
4.2.2. FIR	23
4.2.3. CIC	24
4.2.4. Goertzel	26
4.2.5. UART	29
4.2.6. Aplikacja PC	29
4.3. Symulacja	30
Zakończenie	36
Zawartość płyty	37
Słownik terminów	38
Spis rysunków	40
Spis tabel	41
Bibliografia	42

Wstęp

Przez ostatnie lata można zaobserwować gwałtowny rozwój usług mobilnego dostępu do danych. Wciąż zwiększana przepustowość oraz jednoczesna ilość obsługiwanych użytkowników wymaga zastosowania odpowiednich metod technicznych w celu ich prawidłowej realizacji.

Cegiełką, która jest podstawą w budowie tych systemów jest prezentowana w niniejszej pracy technologia programowalnego radia cyfrowego, obecna na rynku cywilnym od początków lat 90. XX wieku (na przykład w formie telefonii GSM), a przez wojsko używana już dwie dekady wcześniej. Jej zastosowanie znacząco upraszcza projektowanie systemów radiowych a także umożliwia znaczące obniżenie kosztów poprzez przeniesienie większości przetwarzania sygnału z obszaru analogowego — sprzętowego, do cyfrowego — oprogramowania. Rezultatem jest znaczne przyspieszenie prac inżynierskich nad projektem, łatwość poprawianie błędów (nawet na wyprodukowanym już urządzeniu) a także zmniejszenie rozmiarów wynikowego PCB. Z drugiej strony takie podejście do problemu wymaga dużych nakładów obliczeniowych czyli szybkich procesorów, lecz wraz z rozwojem coraz bardziej zaawansowanych technik półprzewodnikowych stało się to możliwe. Przykładem były telefony firmy Nokia, np. model 5110 wydany w roku 1998 — zawierał układ ASIC (zintegrowany procesor ARM z układem DSP), odpowiadający za przetwarzanie danych.

Obecnie, za sprawą takich projektów jak np. USRP [16] czy HackRF [18] znacznie zwiększyła się popularność techniki SDR i właściwie każdy może mieć do niej łatwy dostęp. Mnogość projektów [61] pozwala na wybraniu i dostosowanie sprzętu do własnych potrzeb.

Cel i zakres pracy

Tematem obejmującym niniejszą pracę dyplomową jest analiza zagadnienia radia programowalnego, projekt sprzętu oraz symulacja prototypu cyfrowego systemu odbioru i dekodowania transmisji radiowej wzorca czasu DCF77. System odbiorczy

powinien być zdolny do odebrania sygnału radiowego oraz takiego jego przetworzenia, aby możliwe było odczytanie nadawanego komunikatu.

Zakres pracy obejmuje następujące podmoduły:

- **Sprzęt** – celem było zaprojektowanie alternatywnej do już istniejących platformy wielokrotnego użytku do odbioru sygnałów radiowych
- **Oprogramowanie** – celem było zaimplementowanie jednego z istniejących algorytmów używanych w technice SDR

W ramach realizacji, projekt zostanie zaprogramowany w środowisku FPGA. Docelowym modulem jest płytką *DE0-Nano* firmy *Terasic*. Samo środowisko programistyczne wynika z zastosowanego modułu prototypowego — jest to *Altera Quartus Prime* w wersji 15.1.

Praca składa się z trzech części:

- Pierwszy rozdział opisuje założenia techniki SDR.
- Drugi rozdział przedstawia system nadawania sygnału czasu DCF77.
- Trzeci rozdział przedstawia część sprzętową: płytkę prototypową z układem FPGA i płytkę prototypową będącą modulem radiowym.
- Czwarty rozdział skupia się nad proponowaną architekturą oprogramowania, zawiera przedstawienie wykorzystanych modułów oraz prezentuje symulacje które powstały w celu przetestowania poprawności zaprojektowanego oprogramowania.

Przy tworzeniu pracy wykorzystane były następujące oprogramowanie i narzędzia CAD:

- **Eagle PCB 7.4** – program do projektowania płytek drukowanych
- **Altera Quartus Prime 15.1** – środowisko projektowania dla układów firmy Altera
- **Modelsim 10.4d Altera Edition** – symulator języków opisu sprzętu (dołączony do środowiska Quartus Prime)
- **Qucs 0.0.18** – open-source symulator analogowych obwodów elektronicznych
- **Python 3.4.3** – język programowania wysokiego poziomu ogólnego przeznaczenia
- **Matplotlib 1.4.3** – biblioteka dla języka Python umożliwiająca rysowanie wykresów

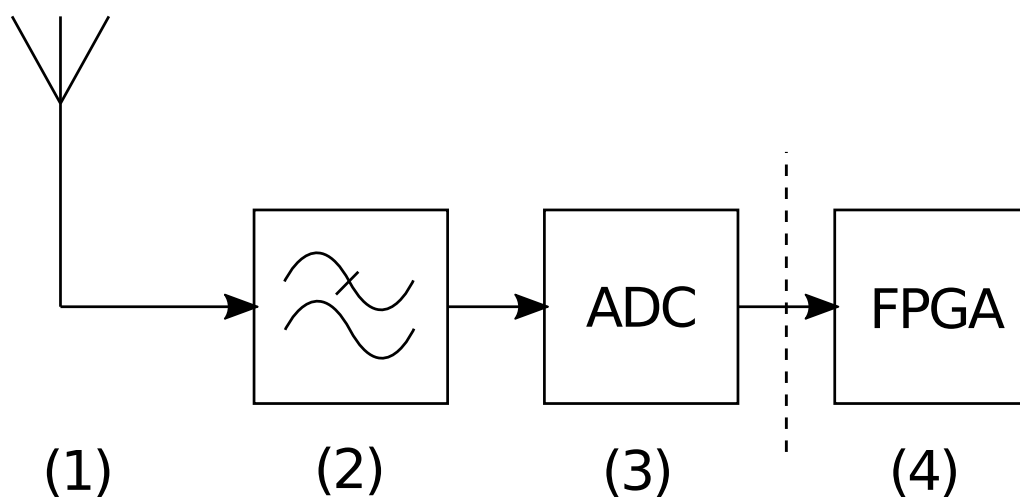
Rozdział 1

Software Defined Radio

Rozwój technologii produkcji układów scalonych wielkiej skali integracji umożliwił rozpoczęcie innowacyjnych projektów. Jednym z nich z pewnością można nazwać ideę radia programowalnego. Realizacja praktyczna rozpoczęła się w latach 80. XX wieku [27]. Dziesięć lat później pojęcie to zostało opisane teoretycznie [38]. Realizacja praktyczna była możliwa dzięki wykorzystaniu procesora DSP TMS320 firmy Texas Instruments. Dzisiaj każdy zainteresowany może korzystać z tej technologii.

1.1. Teoria

Rysunek 1.1 przedstawia typowy sposób działania radia programowalnego.



Rysunek 1.1: Idea radia programowalnego

Pierwszym elementem (1) jest antena. Jej wybór determinowany jest zakładanym przez projektantów systemu zakresem częstotliwości wejściowych, kierunkowością, selektywnością i szerokopasmowością.

Drugi element (2) to filtry wejściowe. Doprowadzenie pełnego pasma do przetwornika analogowo-cyfrowego spowodowałoby powstanie zjawiska aliasingu [56,57].

Z tego powodu należy zastosować filtry analogowe o odpowiedniej charakterystyce, które zapobiegają przedostawaniu się niepożądanych części pasma radiowego do dalszych części systemu.

Kolejnym (3) elementem systemu jest przetwornik analogowo-cyfrowy. Umożliwia on odwzorowanie sygnałów analogowych na sygnały cyfrowe możliwe do analizowania i przetwarzania przez logikę cyfrową [29, 56].

Ostatnim (4) elementem systemu jest układ akwizycji danych. Ten element jest odpowiedzialny za odbiór danych z przetwornika ADC oraz następne przetworzenie sygnału. W tym miejscu można zastosować dokładniejsze, bardziej selektywne algorytmy filtracji oraz inne procesy operujące na sygnale tak, aby jak najlepiej dostosować go do oczekiwań projektantów. Na schemacie ten blok podpisano jako FPGA, gdyż ten będzie reprezentowany w niniejszej pracy. Jednak nic nie szkodzi na przeszkodzie aby był to inny układ (na przykład typu DSP, szeroko stosowany w komunikacji komórkowej).

Cały schemat jest uproszczonym przykładem SDR, nakreśla jednak jego główne właściwości. Przerywana strzałka na obrazku symbolicznie rozdziela część sprzętową (po lewej stronie) od części programowej przetwarzania sygnału.

Rozdział 2

DCF77

DCF77 jest to nazwa sygnału czasu nadawany na falach długich. Częstotliwość została zarejestrowana w ITU w 1953 a system rozpoczął działanie w 1959 roku — pierwotnie jako wzorzec częstotliwości. W roku 1979 dołączono zmodulowaną informację o aktualnej godzinie. Wysoka stabilność sygnału, kontrolowana przez zegary atomowe powoduje, że sygnał jest uznawany za wzorzec odniesienia czasu w Niemczech.

Sygnał jest kontrolowany przez *Physikalisch-Technische Bundesanstalt* (narodowy instytut metrologii Republiki Federalnej Niemiec) i nadawany z miejscowości Manfliegen — około 25 km na południowy zachód od Frankfurtu nad Menem. Moc nadajnika to 50 kW a zasięg użyteczny sygnału to około 2000 km w promieniu od nadajnika [12] (rys. 2.1). Czas nadawany przez DCF77 może być także stosowany bez zmian na terenie Polski ze względu na jednakową strefę czasową.

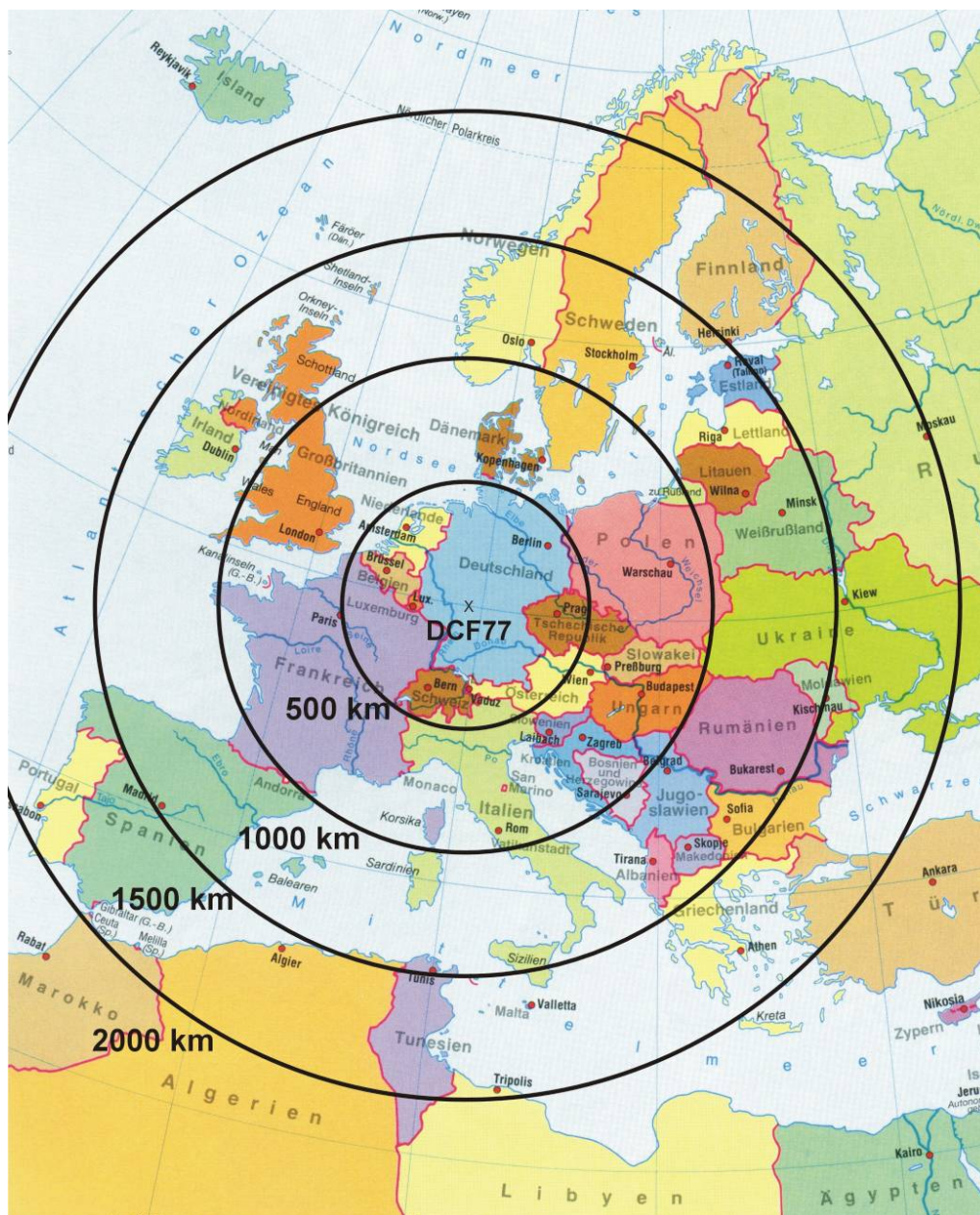
2.1. Sygnał i jego właściwości

Częstotliwość nośna sygnału wynosi 77,5 kHz jest modulowana amplitudowo z sekundowymi znacznikami. Z początkiem każdej sekundy amplituda jest redukowana na czas 0,1 s lub 0,2 s — z wyjątkiem ostatniej sekundy każdej minuty, co jest identyfikatorem początku kolejnej minuty (rys. 2.4).

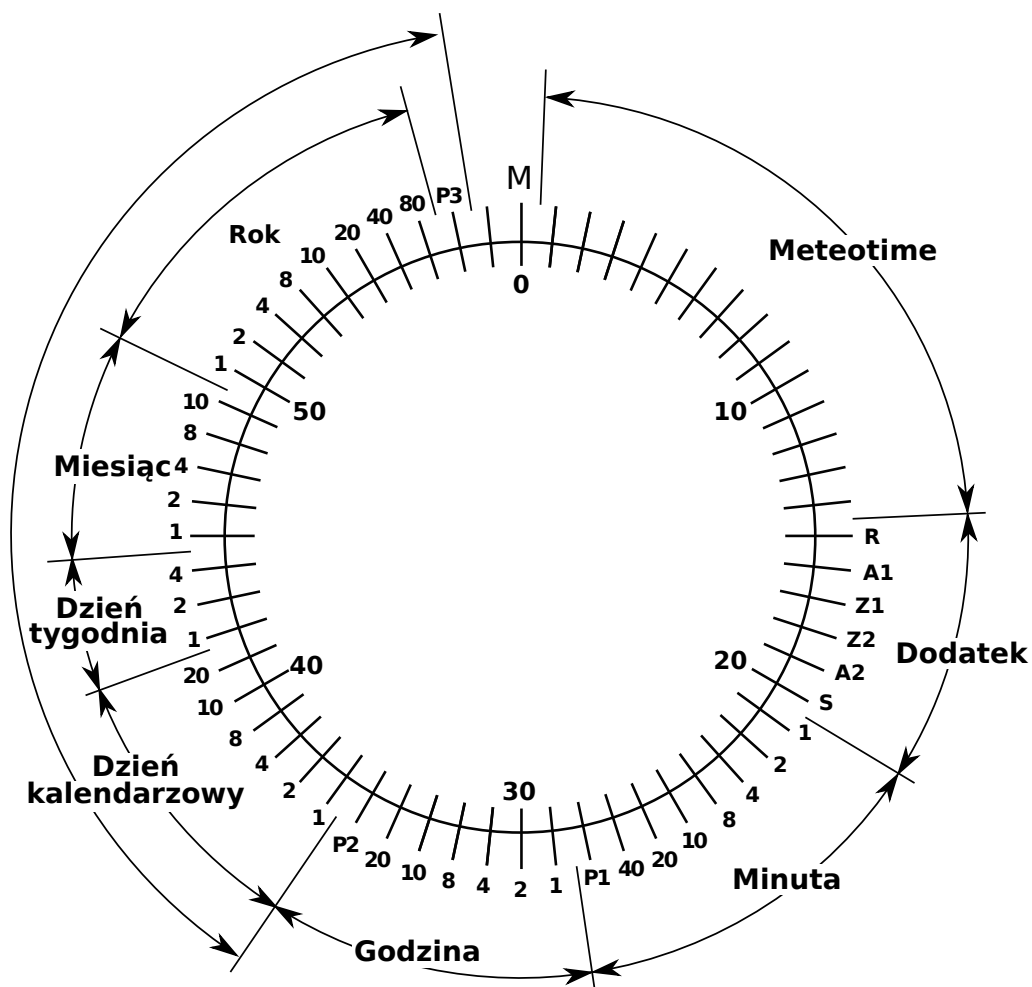
Z początkiem każdej sekundy amplituda sygnału jest zmniejszana do 15% na 0,1 s lub 0,2 s [12].

2.2. Kodowanie nadawanego komunikatu

Każda zmiana amplitudy oznacza wartość w systemie binarnym, brany jest pod uwagę czas tej zmiany. I tak — zmiana na okres 0,1 s uważana jest za binarne zero a 0,2 s — binarną jedynkę. Co każdą minutę transmitowane są sygnały minuty, godziny, dnia, miesiąca i roku w systemie BCD — czas jest nadawany dla nadchodzącej



Rysunek 2.1: Zasięg sygnału DCF77 (źródło: [12])



Rysunek 2.2: Wykres zegarowy sygnału DCF77

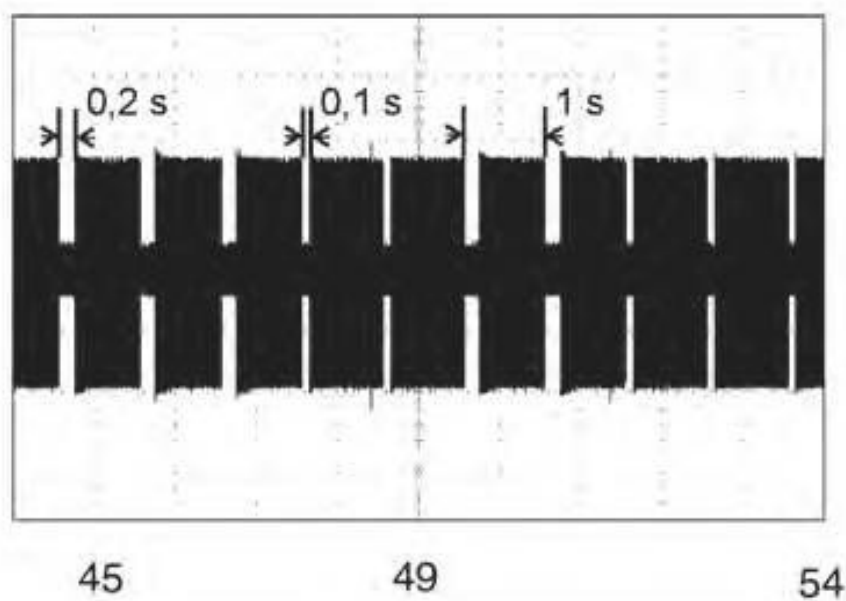
minuty. Całą zawartość komunikatu ilustruje rysunek 2.2, w którym przedstawiono nadawane bity w formie tarczy zegarowej.

Komunikat, tak jak wspomniano powyżej, jest podzielony na kilka grup, co bardziej szczegółowo przedstawia tabela 2.1. Początek nowego komunikatu jest oznaczony przez brak modulacji (rys. 2.3).

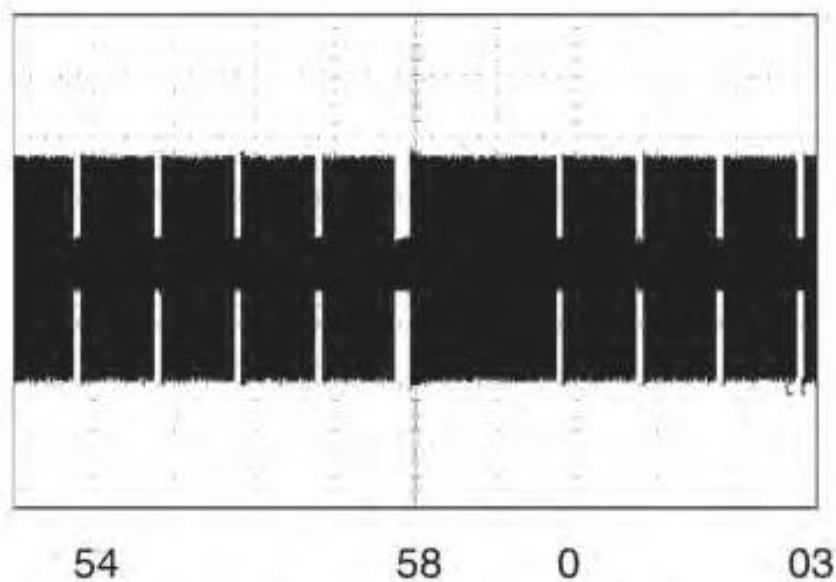
Ciekawą informacją dodatkową w sygnale jest komunikat *Meteotime*. Jest to informacja pogodowa dla ponad osiemdziesięciu obszarów Europy. Dla Polski wybrano 3 obszary, które jednak nie obejmują całości kraju.

Bit	Symbol bitu	Grupa
0	M	początek minuty (zawsze 0)
1-14	–	informacja pogodowa (Meteotime)
15	R	nieprawidłowe działanie nadajnika
16	A1	nadchodzący czas letni (przez godzinę przed ustaleniem)
17	Z1	czas zimowy
18	Z2	czas letni
19	A2	nadchodząca sekunda przestępna (przez godzinę przed ustaleniem)
20	S	początek sygnału czasu (zawsze 1)
21-27	–	minuty (pierwszy bit najmłodszy)
28	P1	bit parzystości minuty
29-34	–	godziny (pierwszy bit najmłodszy)
35	P2	bit parzystości godziny
36-41	–	dzień miesiąca (pierwszy bit najmłodszy)
42-44	–	dzień tygodnia (pierwszy bit najmłodszy)
45-49	–	miesiąc
50-57	–	rok (dwie cyfry)
58	P3	bit parzystości daty (bity 36-57)
59	–	znacznik nowej minuty (brak modulacji)

Tabela 2.1: Poszczególne bity sygnału DCF77



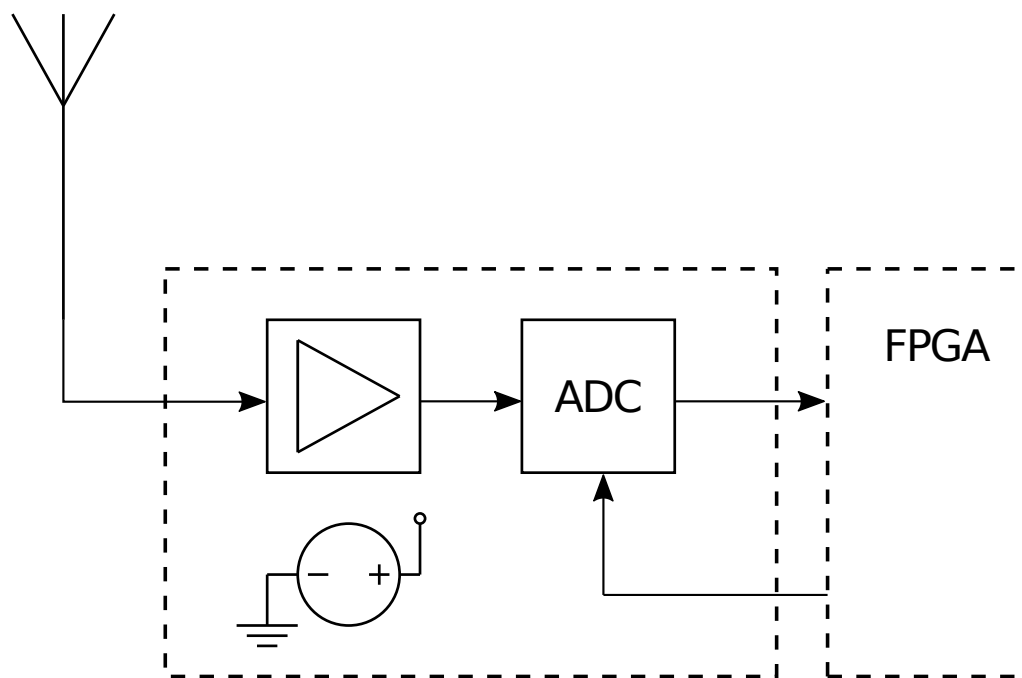
Rysunek 2.3: Obwiednia sygnału DCF77 (źródło: [12])



Rysunek 2.4: Znacznik nowej minuty DCF77 (źródło: [12])

Rozdział 3

Sprzęt – projekt



Rysunek 3.1: Architektura modułowa części sprzętowej

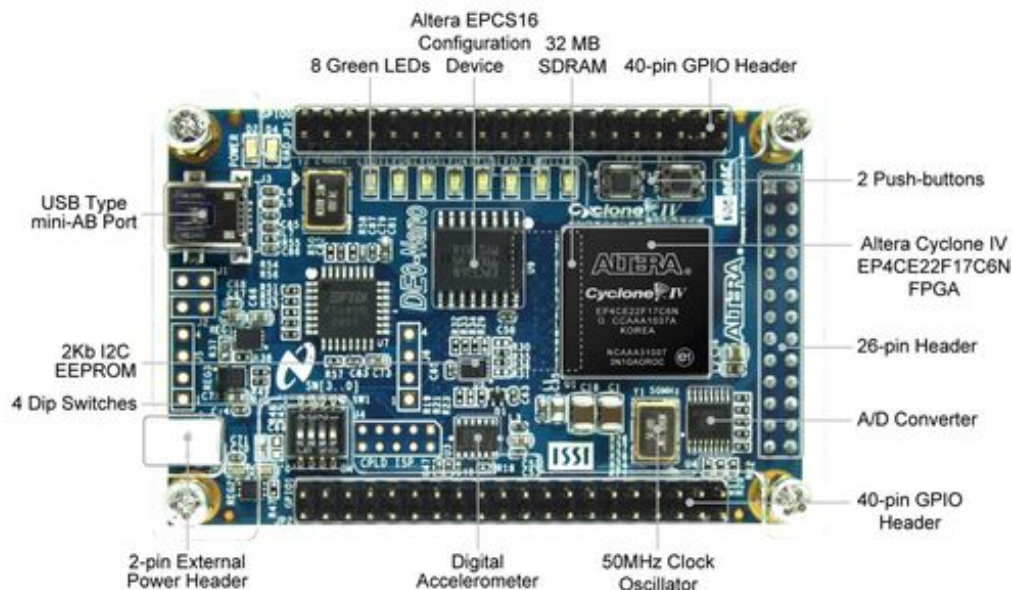
Sekcja sprzętowa (rys. 3.1) składa się z dwóch modułów: z płytki drukowanej zaprojektowanej przez autora, będącej częścią wstępną pobierania danych (front-end) oraz z modułu prototypowego FPGA firmy Terasic (po prawej).

3.1. Moduły sprzętowe

3.1.1. Płytki bazowa FPGA

Projekt jest zbudowany na podstawie, i jako płytki bazowej używa zestaw startowy *Terasic DE0-Nano* [53].

Moduł zawiera ponad 22 tysiące programowalnych elementów logicznych, 500 kilobajtów wbudowanej pamięci, 66 układów mnożenia, 4 generatory częstotliwości PLL oraz 72 porty wejścia/wyjścia zgrupowane w dwa gniazda typu 2x40 pin



Rysunek 3.2: Moduł prototypowy *Terasic DE0-Nano* (źródło: [55])

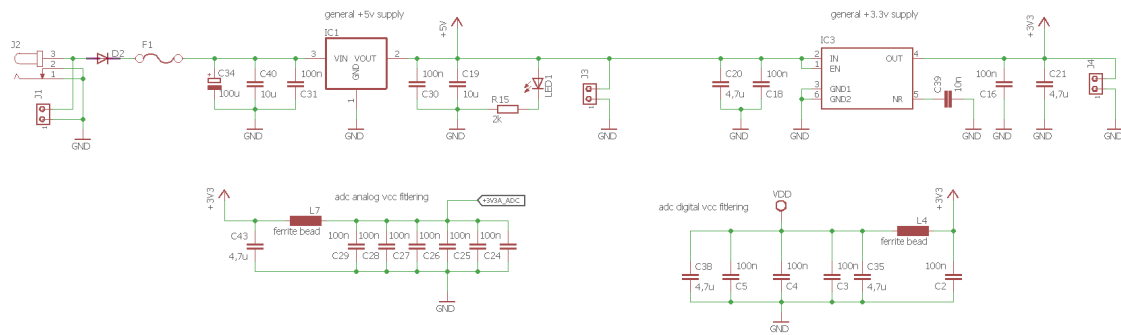
(rys. 3.2). Dodatkowo (co nie jest używane w tym przypadku) posiada także 8 diod LED, 32 megabajty pamięci dynamicznej typu SDRAM, akcelerometr i przetwornik analogowo-cyfrowy o szybkości próbkowania 50 ksps [55].

3.1.2. Schemat ideowy

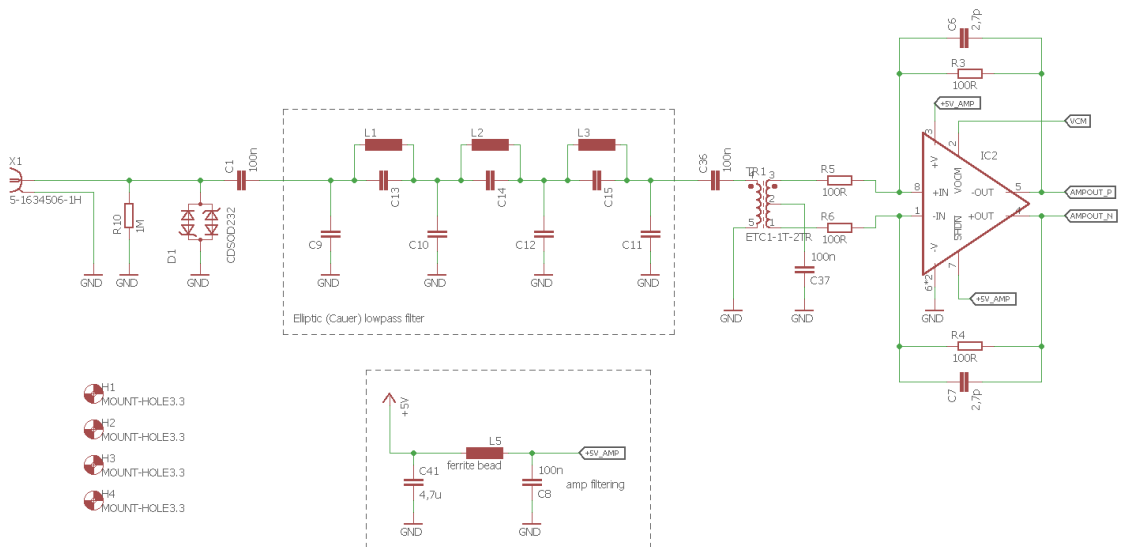
Schemat ideowy autorskiego modułu (rys. 3.3) był projektowany w metodologii analizy zstępującej (od ogółu do szczegółu, top-down). Wpierw wyróżniono trzy główne elementy, tj. zasilanie, wejście sygnału radiowego oraz jego przetwarzanie (rys. 3.1). W następnym kroku uszczegóławiano je. Wyróżnione moduły zostały opisane poniżej.

Sekcja zasilania

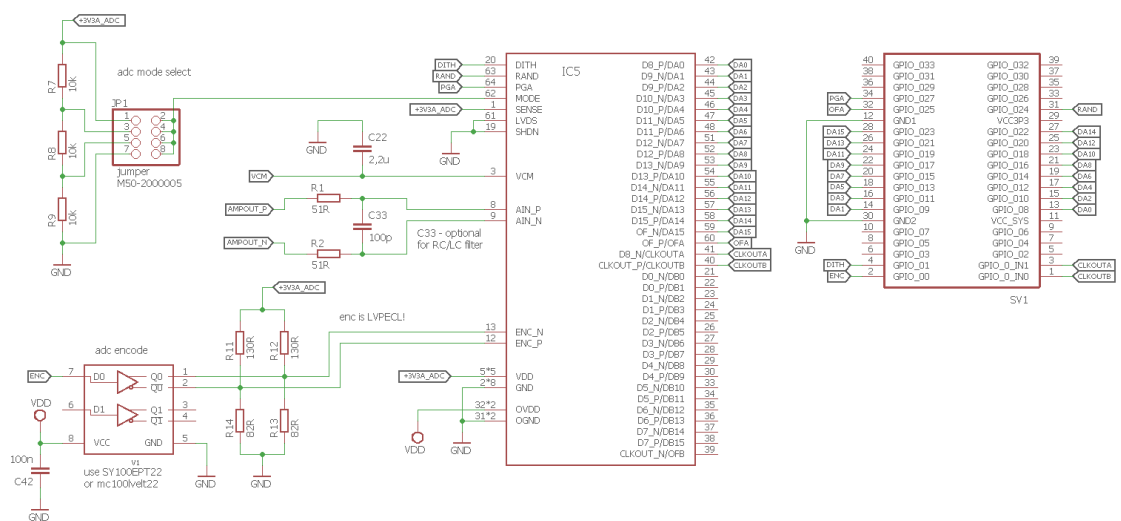
Blok zasilający układy scalone przedwzmacniacza oraz przetwornika ADC. Założeniem była uniwersalność zasilania. W tym celu zastosowano zasilanie dwustopniowe: w pierwszym stopniu wejściowe napięcie z zakresu od 6 do 20 V zostaje przetworzone przez liniowy regulator o niskim spadku napięcia do wartości 5 V. Następnie to napięcie zostaje skierowane do kolejnego liniowego regulatora o niskim spadku napięcia; tutaj napięcie wyjściowe to 3,3 V. Wszystkie układy są filtrowane kondensatorami odsprężającymi [24, 39], dodatkowo zielona dioda LED wskazuje na stan modułu.



(a) Zasilanie



(b) Część analogowa



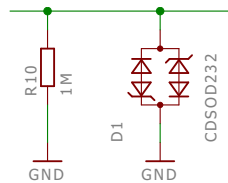
(c) Część cyfrowa

Rysunek 3.3: Schemat ideowy modułu prototypowego

Sekcja formowania sygnału radiowego

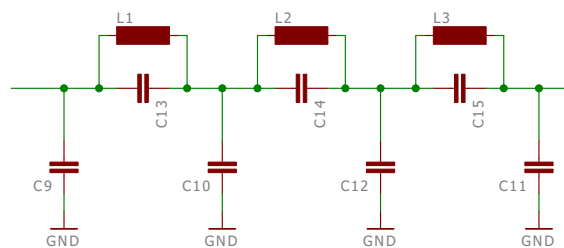
W tej sekcji znajdują się elementy pośredniczące pomiędzy anteną a ADC. W literaturze ten blok nazywany jest „Analog front-end” (AFE) lub „RF front-end”. Jest odpowiedzialny za wstępne przygotowanie sygnału i jest krytycznym elementem z tego względu, gdyż wszelkie dalsze przetwarzanie sygnału jest wykonywane na drodze cyfrowej.

Sygnał radiowy jest wprowadzany do układu poprzez złącze BNC. Następnie sygnał prowadzony jest poprzez układ chroniący przed przepięciem [48, 49] (rys. 3.4), do zaprojektowanego filtra (rys. 3.5).



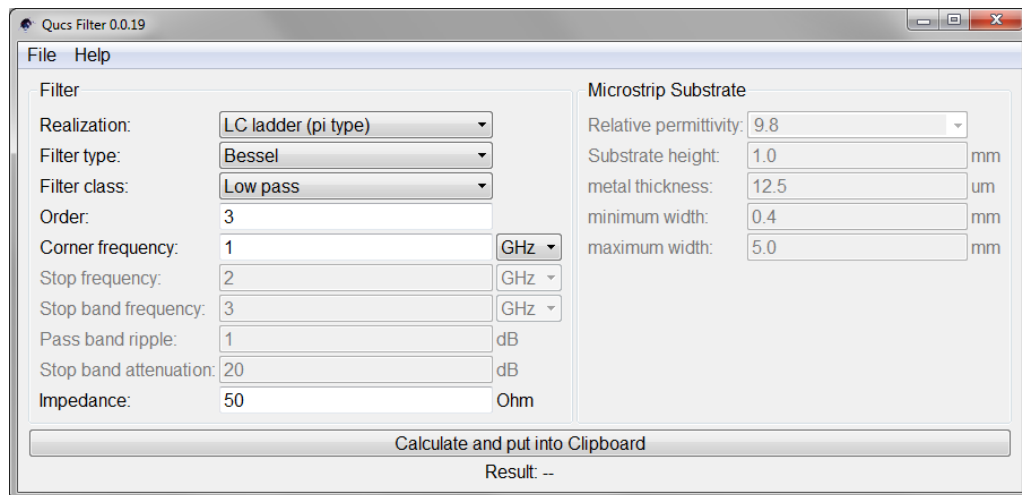
Rysunek 3.4: Układ chroniący przed przepięciem

Filtr został zaprojektowany w programie Qucs [45]. Program ten jest darmowym programem typu SPICE i pozwala na analizę i symulację układów analogowych. Posiada wtyczkę do projektowania filtrów (rys. 3.6) użytą do wygenerowania i symulacji charakterystyki filtra. W tym przypadku zaprojektowano filtr eliptyczny, typ π , dolnoprzepustowy o częstotliwości granicznej 55 MHz i tłumieniu w paśmie zaporowym 59 dB (rys. 3.7). Wybrano filtr typu eliptycznego ze względu na jego ostre przejście pomiędzy pasmem przepuszczania i zaporowym oraz możliwość dokładnego określenia tętnień i szerokości pasma przejściowego.

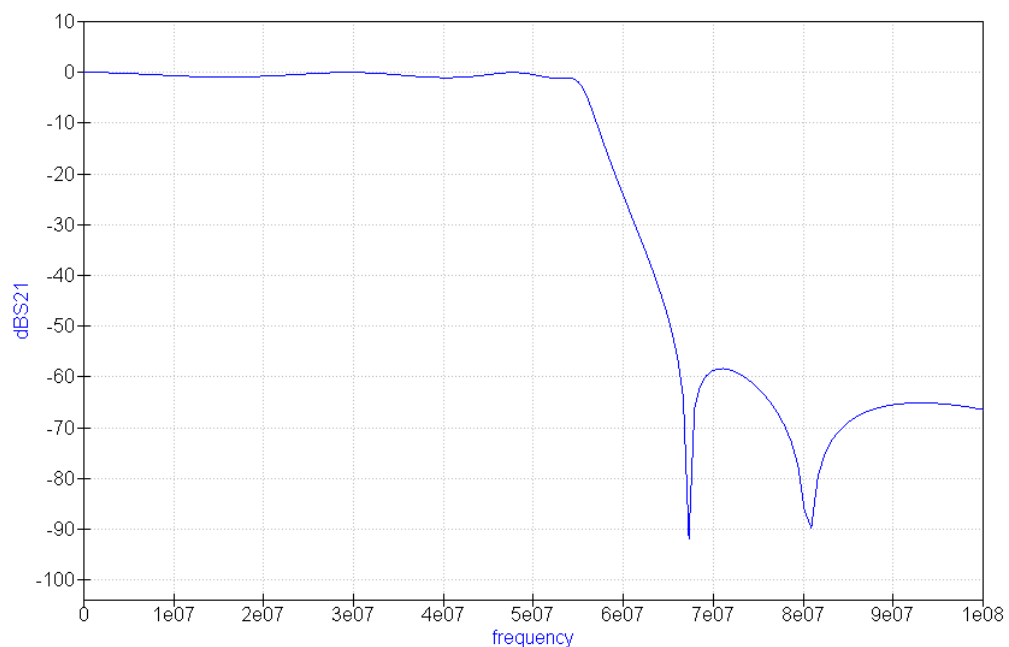


Rysunek 3.5: Filtr antyaliasingowy

Po filtracji, sygnał jest kierowany do symetryzatora, który zamienia sygnał niesymetryczny na symetryczny a następnie do przedwzmacniacza LTC6405 zasilanego napięciem 5 V.



Rysunek 3.6: Qucs – projektowanie filtrów



Rysunek 3.7: Qucs – symulowana charakterystyka zaprojektowanego filtru

Sekcja cyfrowa

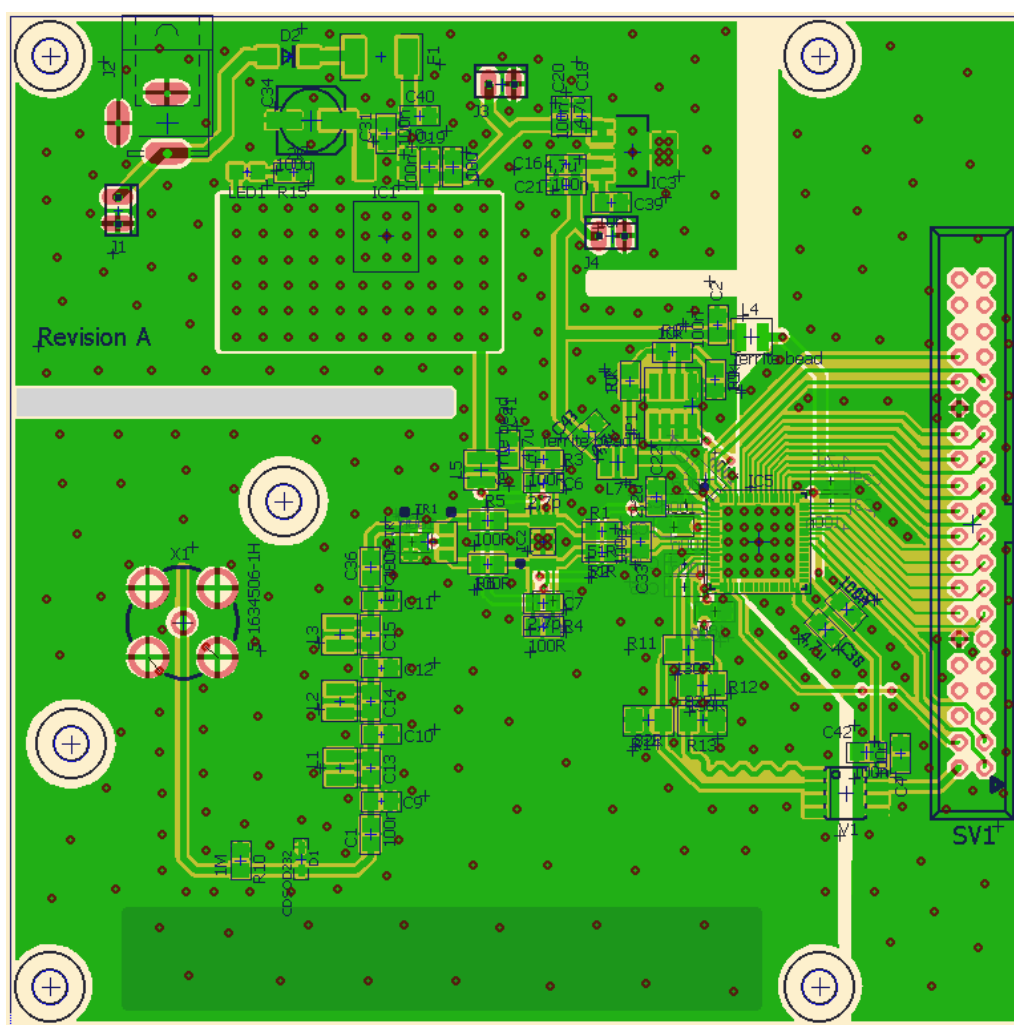
W tej części sygnał jest wstępnie przetworzony przez sekcję formowania sygnału radiowego i dopasowany do układu przetwornika analogowo-cyfrowego LTC2208. Następnie sygnał jest próbkowany przez przetwornik i z drugiej strony może być odebrany przez moduł prototypowy FPGA. Dodatkowo, w tej sekcji istnieją też dodatkowe elementy takie jak zworka wyboru trybu działania ADC oraz układ SY100EPT22 który odpowiada za przetwarzanie z poziomu napięć TTL na sygnał różnicowy, odpowiedni do taktowania przetwornika [58]. Ostatecznie, sygnał cyfrowy

przesyłany jest szesnastoma liniami danych i poprzez złącze krawędziowe może być pobrane przez docelowy układ (tutaj – moduł FPGA).

3.1.3. Płytką drukowaną

Jak wspomniano wcześniej, schemat ideowy został podzielony na logiczne części. Z tego wynikł schemat projektu prototypu płytki drukowanej oraz jego podział, również na trzy części. Na rys. 3.9 zaznaczono je następująco: sekcja zasilania – niebieski, sekcja wejścia sygnału radiowego – żółty oraz sekcja cyfrowa – czerwony.

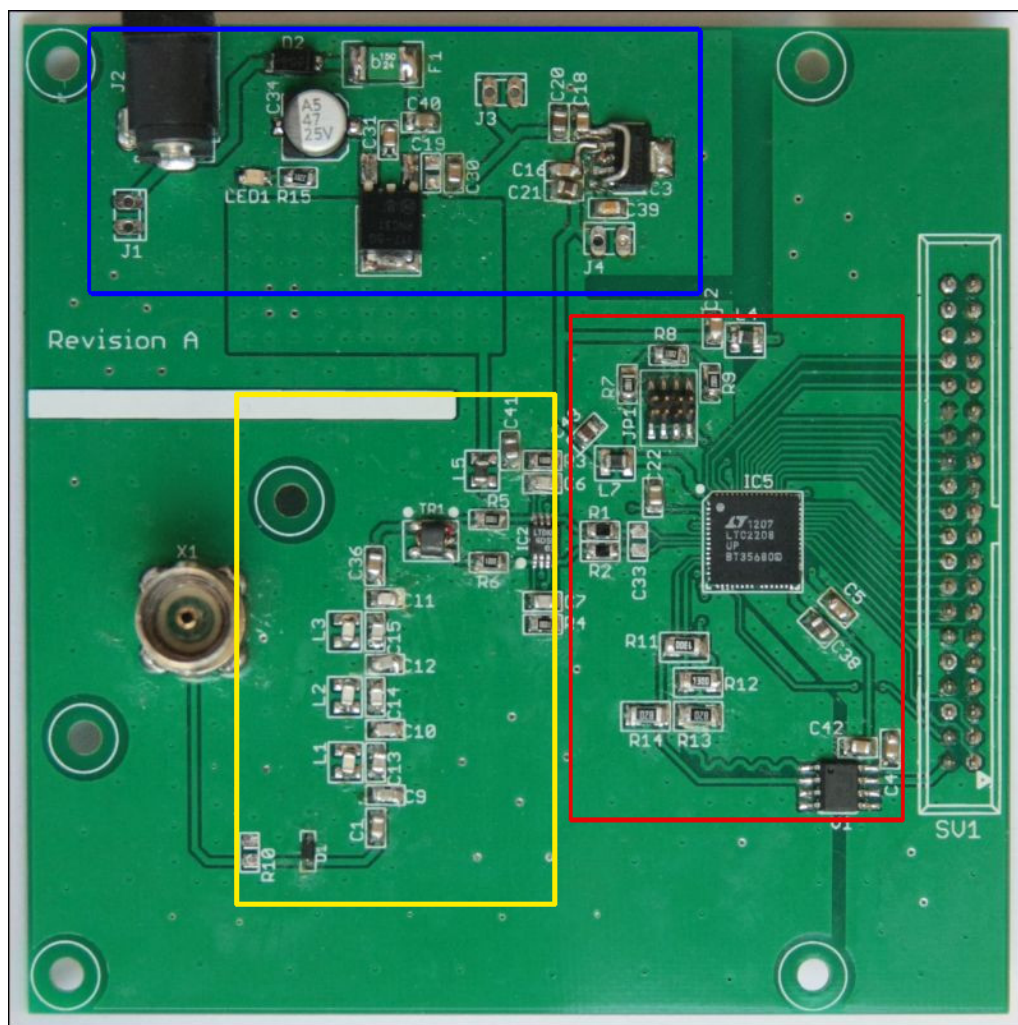
Schemat ideowy został zastosowany do zaprojektowania płytki drukowanej (rys. 3.8) a następnie do jej fizycznego wykonania oraz zmontowania (rys. 3.9).



Rysunek 3.8: Płytką drukowaną — rendering

Podczas projektowania wzorowano się na istniejących, zrealizowanych projektach [1, 40–42].

Zostały poczynione następujące założenia w kwestii zaprojektowania PCB:



Rysunek 3.9: Płytką drukowana — wykonanie fizyczne

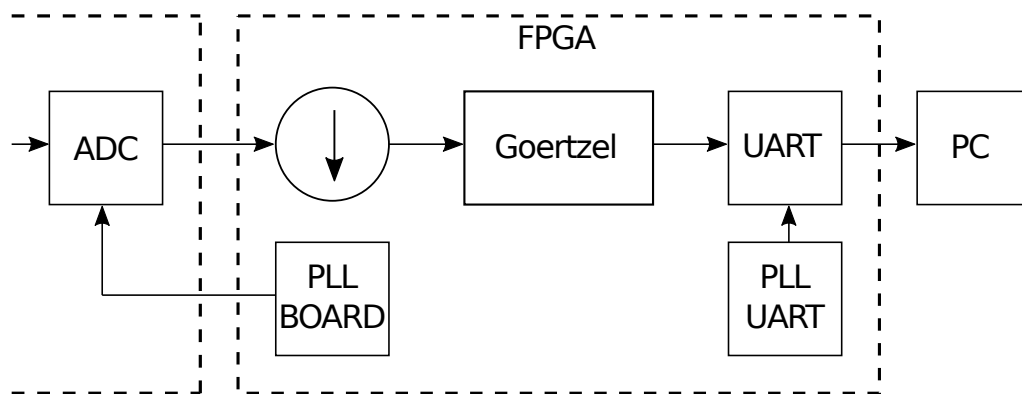
- **Dwuwarstwowa płytka drukowana** – płytka ma na celu przenosić sygnał radiowy. Z tego względu zdecydowano się na wykonanie płytki dwuwarstwowej, z jedną warstwą masy.
- **Elementy SMD** – Płytkę od początku projektowano z myślą o elementach SMD (rys. 3.8). Było to założenie narzucone przez same układy przetwornika i przedwzmacniacza, które posiadają mały raster wyprowadzeń.

Projekt został wykonany w zewnętrznej firmie [15]. Tak więc możliwe było uzyskanie wysokiej jakości oraz dokładności połączeń przez co montaż powierzchniowy był możliwy do zrealizowania. Przylutowanie elementów o bardzo małym rastrze oraz takich, które wymagały zaawansowanych narzędzi, zlecono zewnętrznej firmie. Pozostałe elementy zostały zlutowane przez autora. Wynikowa wersja modułu została zaprojektowana z myślą o efektywnym oraz logicznym rozkładzie komponentów wraz z zasadami projektowania [43].

Rozdział 4

Oprogramowanie – moduły, algorytmy

4.1. Architektura programowa



Rysunek 4.1: Architektura modułowa części programowej

Rysunek 4.1 przedstawia architekturę części programowej urządzenia. Autor wybrał styl projektowania top-down w celu implementacji powyższej architektury. Działanie poszczególnych modułów było badane, rozpoznawane, implementowane a następnie testowane. Kolejnym krokiem było poszczególne łączenie modułów w całość.

Symbol strzałki w dół zawartej w okręgu symbolizuje decymację sygnału. W tym bloku duża częstotliwość z jaką przesyłane są próbki danych z przetwornika ADC jest obniżana. Rolę tego bloku może pełnić filtr FIR o zaprojektowanej charakterystyce lub dolnoprzepustowy filtr CIC.

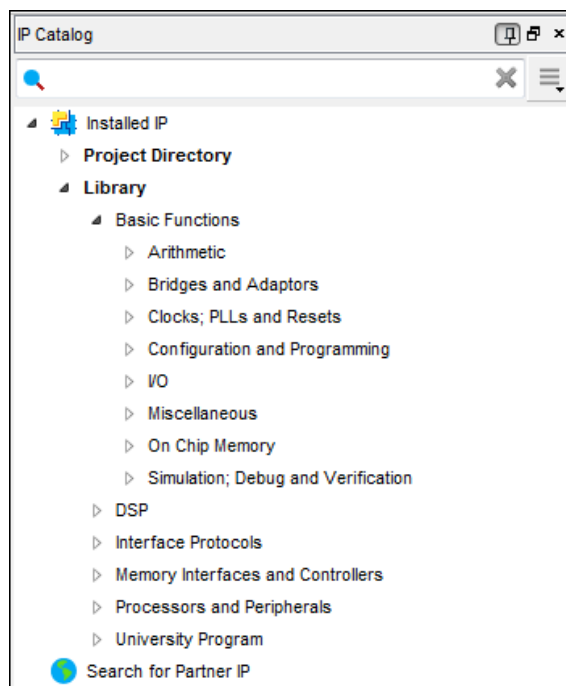
Każdy moduł udostępnia sygnały (porty) wejścia/wyjścia dla sterowania i interakcji z modułem. Zdefiniowano następujące moduły:

- **reset_state** – moduł resetu. Zarządza globalnym ustawieniem resetowania innych modułów tego wymagających
- **pll_board** – moduł PLL. Jego celem jest przyjęcie zegara o częstotliwości 50 MHz i wygenerowanie na jego podstawie zegara 130 MHz, który taktuje PCB z układem ADC

- **pll_uart** – moduł PLL. Jego celem jest przyjęcie zegara o częstotliwości 50 MHz i wygenerowanie na jego podstawie zegara 115,2 kHz taktującego moduł „uart”
- **fir_first** – moduł filtra FIR. Odpowiedzialny za wstępne filtrowanie oraz decymację sygnału wejściowego
- **cic** – moduł filtra CIC. Odpowiedzialny za wstępne filtrowanie oraz decymację sygnału wejściowego
- **goertzel** – moduł algorytmu Goertzela. Jest to moduł obliczania pojedynczego prążka transformaty Fouriera za pomocą algorytmu Goertzela
- **uart** – moduł RS232 – część odpowiedzialna za przesyłanie obliczonych wyników do komputera PC

4.2. Moduły

Przestawione w poprzedniej sekcji wybrane moduły zostaną teraz szczegółowiej opisane — są to moduły zarządzania zegarem, filtrujące, moduł algorytmu Goertzela oraz moduł RS232.



Rysunek 4.2: Katalog bloków IP w programie Quartus

4.2.1. PLL

Moduł, który jest udostępniany przez firmę Altera do swoich produktów w postaci „IP Core’a” — fragmentu kodu, który został napisany przez firmę Altera i jest

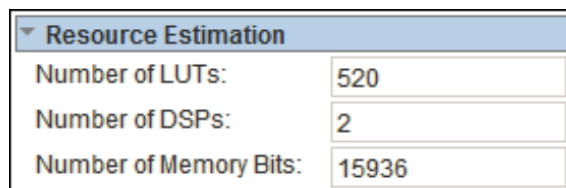
udostępniany innym programistom. W środowisku Quartus Prime moduły IP można znaleźć w zakładce „IP Catalog” (rys. 4.2).

Konfiguracja modułu PLL rozpoczyna się od znalezienia pozycji „ALTPLL” w katalogu IP. Podstawowa konfiguracja zakłada wpisanie częstotliwości wejściowej zegara oraz zdefiniowanie częstotliwości wyjściowej. Kreator przy tym przekazuje nam odpowiedni komunikat dotyczący tego, czy zadane wartości są prawidłowe i moduł będzie mógł być prawidłowo zaimplementowany w strukturze FPGA. W podsumowaniu wybieramy pliki utworzone przez kreator, które będą użyte przy syntezie i ewentualnej symulacji [4, 10].

W pracy dyplomowej tej funkcji użyto w modułach „pll_board” oraz „pll_uart”. Pierwszy z nich służy do wygenerowania częstotliwości 130 MHz taktującej układ przetwornika analogowo-cyfrowego, drugi — do użycia przez moduł obsługi portu szeregowego „uart”.

4.2.2. FIR

Moduł również udostępniany przez firmę Altera w postaci IP Core pod nazwą „FIR II”. Moduł daje dużo większe możliwości konfiguracyjne aniżeli „PLL”. Można skonfigurować wiele zmiennych, między innymi prędkość zegara, wielkość magistrali wejścia i wyjścia. Zakładka „Coefficients” pozwala na wprowadzenie współczynników filtra oraz wygenerowanie wykresu (rys. 4.4).

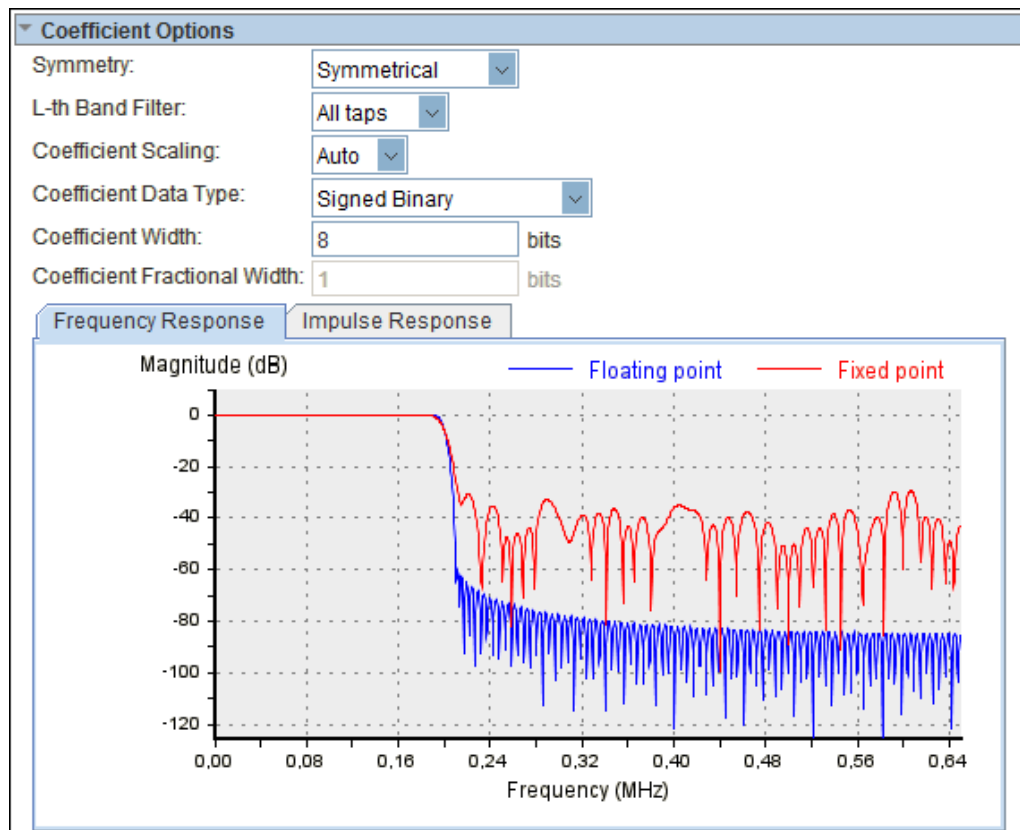


▼ Resource Estimation	
Number of LUTs:	520
Number of DSPs:	2
Number of Memory Bits:	15936

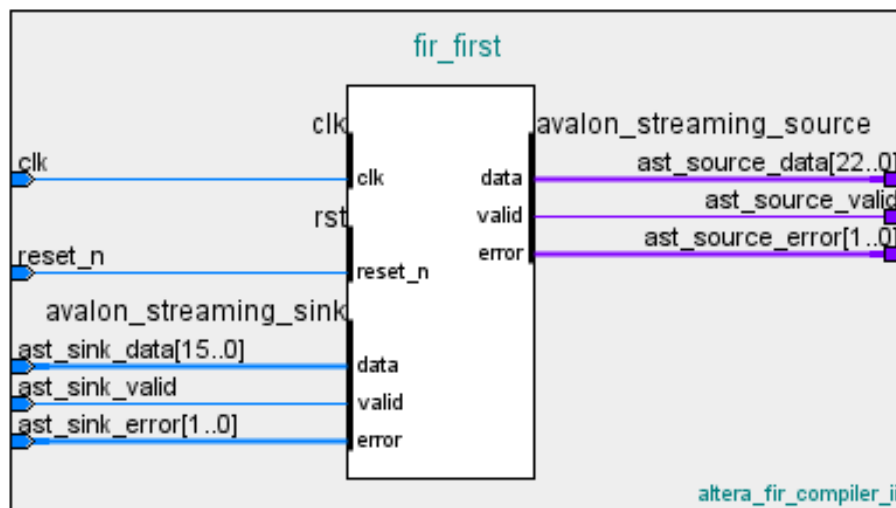
Rysunek 4.3: Przewidywanie zajętości zasobów logicznych projektowanego modułu

Istotnym wskaźnikiem jest okno „Resource estimation” (rys. 4.3). Pozwala ono stwierdzić jak dużo zasobów logicznych będzie potrzebować projektowany moduł. Jest to o tyle istotne, że podczas symulacji wszystkie wymagania będą spełnione. Jednak przy próbie syntezy układu środowisko Quartus z pewnością poinformuje o braku dostępnych zasobów logicznych niezbędnych do zsyntezowania projektu w układzie FPGA. Ma to miejsce na przykład przy dość dużej ilości współczynników filtra [8, 10].

Ostatecznie, wygenerowany moduł filtra ma sygnały podobne do tych na rys. 4.5.



Rysunek 4.4: Współczynniki w kreatorze i symulacja charakterystyki

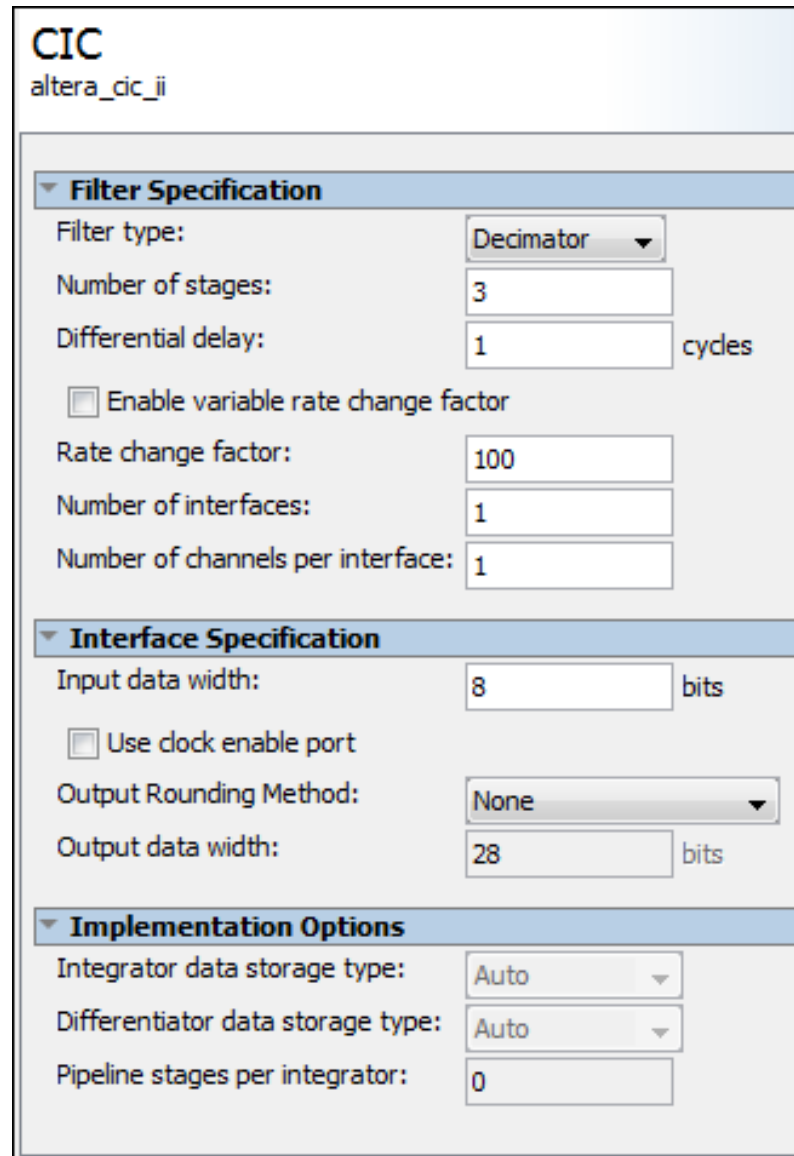


Rysunek 4.5: Przykładowe sygnały modułu „FIR II”

4.2.3. CIC

Kolejny moduł udostępniany jako „IP Core”. Podobnie jak „FIR” daje duże możliwości konfiguracji (rys. 4.6).

Filtr typu CIC, jest podklasą filtrów typu FIR [21]. Zalety jego użycia obejmują między innymi znaczne uproszczenie obliczeń w stosunku do filtra FIR — jedyne



The image shows the configuration window for the 'CIC' module, specifically the 'altera_cic_ji' instance. The window is divided into three main sections: Filter Specification, Interface Specification, and Implementation Options.

Filter Specification

- Filter type: Decimator (dropdown)
- Number of stages: 3 (text input)
- Differential delay: 1 (text input) cycles
- ☐ Enable variable rate change factor
- Rate change factor: 100 (text input)
- Number of interfaces: 1 (text input)
- Number of channels per interface: 1 (text input)

Interface Specification

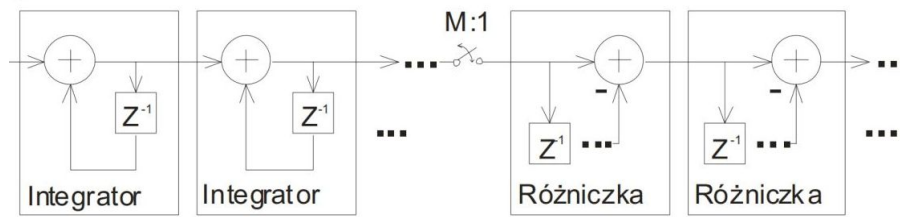
- Input data width: 8 (text input) bits
- ☐ Use clock enable port
- Output Rounding Method: None (dropdown)
- Output data width: 28 (text input) bits

Implementation Options

- Integrator data storage type: Auto (dropdown)
- Differentiator data storage type: Auto (dropdown)
- Pipeline stages per integrator: 0 (text input)

Rysunek 4.6: Konfiguracja modułu „CIC”

działania arytmetyczne to dodawanie oraz opóźnienie sygnału (rys. 4.7). Jest to bardzo korzystne obliczeniowo, gdyż wymaga niewielkiej ilości zasobów logicznych. Dodatkowym atutem jest charakterystyka dolnoprzepustowa filtra. Jest ona spowodowana decymacją wkomponowaną w strukturę filtra i będąca jego integralną częścią. Także ta właściwość spowodowała zainteresowanie wykorzystaniem tego typu filtra w zastosowaniach cyfrowej zmiany częstotliwości w radiu programowalnym [28].



Rysunek 4.7: Filtr CIC – budowa (źródło: [28])

4.2.4. Goertzel

Algorytm Goertzla pozwala na wykrywanie częstotliwości, podobnie jak transformata Fouriera. Jednak zbiór wartości funkcji ogranicza się do jednej, pojedynczej częstotliwości a nie jak w przypadku transformaty Fouriera, do widma. Z tego powodu, jest bardziej korzystne obliczeniowo korzystać z tego algorytmu podczas wykrywania pojedynczej częstotliwości [11]. Najczęściej stosowana aplikacja to dekodowanie DTMF.

Podstawowa wersja algorytmu zwraca część rzeczywistą i urojoną sygnału, tak jak transformata Fouriera. Możliwe jest więc obliczenie zarówno siły jak i fazy sygnału. Tutaj zastosowano jednak uproszczone (zoptymalizowane) obliczanie siły sygnału. Tracimy przy tym informacje o fazie, lecz nie jest ona potrzebna w tym przypadku.

Kod źródłowy algorytmu Goetzela w języku Python ma się następująco:

Listing 4.1: Algorytm Goertzla – Python

```

1 | N = len(samples)
2 | k = int((N*freq)/sample_rate)
3 | w = 2.0 * np.pi / N * k
4 | cosine = np.cos(w)
5 | coeff = 2 * cosine
6 | q0 = 0.0
7 | q1 = 0.0
8 | q2 = 0.0
9 | for num in range(N):
10 |     q0 = coeff * q1 - q2 + samples[num]
11 |     q2 = q1
12 |     q1 = q0
13 |
14 | result = q1*q1 + q2*q2 - q1*q2*coeff

```

Przed implementacją algorytmu z listingu 4.1 trzeba zdefiniować kilka wartości:

- Szybkość samplowania „**sample_rate**” — Wybrana arbitralnie poprzez szybkość wyjścia danych z filtra drugiego stopnia. Ze względu na to, że zegar taktujący

ADC ma 130 MHz a pierwszy stopień filtra decymuje o krotność 100, to szybkość samplowania wynosi 1,3 MHz.

- Wielkość bloku „N” — Analogicznie do transformaty Fouriera, ten wskaźnik ma znaczenie przy obliczaniu rozdzielczości. Została wybrana wielkość bloku równa 520.
- Numer prążka „k” — liczba typu całkowitego obliczana na podstawie docelowej częstotliwości, która będzie wykrywana. Z tego względu trzeba rozważyć wybrać wartości szybkości samplowania i wielkość bloku. Dla danych podanych wyżej obliczony numer prążka to $\frac{N \cdot target_freq}{sample_rate} = 31$ gdzie *target_freq* (częstotliwość poszukiwana) ma wartość 77500 Hz (77,5 kHz)

Kod z listingu 4.1 został zaimplementowany w języku Verilog (plik goertzel.v) i został podzielony na dwie części: listing 4.2 przedstawia pętlę (tłumaczenie kodu z linii 9-12 listingu 4.1), listing 4.3 określa zapisywanie wyniku (zmienna *result* z linii 14 listingu 4.1). Stała *coeff* została obliczona oraz na stałe zakodowana w kodzie źródłowym w formacie double floating point (64 bity) standardu IEEE-754

Listing 4.2: Algorytm Goertzla – pętla

```

1  always @(posedge clock)
2  begin
3      if (sample_counter != sample_counter_test)
4          i <= 0;
5      else
6          i <= i+1;
7
8      sample_counter_test <= sample_counter;
9
10     if ( (i == 0) && (sample_counter == 0) )
11     begin
12         q0 <= 64'd0;
13         q1 <= 64'd0;
14         q2 <= 64'd0;
15     end
16
17     case (i)
18     0:
19         begin
20             int32fp64conv_i1_aclr <= 1;
21             fpmult_i1_aclr <= 1;
22             fpadd_i1_aclr <= 1;
23             fpsub_i1_aclr <= 1;
24         end
25     10: int32fp64conv_i1_aclr <= 0;
26     20: fpmult_i1_aclr <= 0;
27     30: fpsub_i1_aclr <= 0;
28     40: fpadd_i1_aclr <= 0;
```

```

29      94: q0 <= fpadd_i1_result;
30      95: q2 <= q1;
31      96: q1 <= q0;
32  endcase
33 end

```

Kod 4.2 modeluje automat sekwencyjny i jest podzielony na dwa bloki. W pierwszym bloku (do linii nr 15) następuje inkrementacja licznika. Jeśli rozpoczęło się obliczanie nowego bloku (linia 10), to resetujemy zmienne $q0$, $q1$, $q2$. Drugi blok — instrukcja *case* — modeluje układ kombinacyjny, w którym ma miejsce reagowanie na zmiany wartości licznika. Działanie to ma na celu poprawne obliczenie wartości $q0$, gdyż zdecydowano się na realizowanie obliczeń zmiennoprzecinkowych i wykorzystanie do tego celu modułów IP Core firmy Altera, dla których czas wykonania obliczeń to w najgorszym przypadku 7 cykli zegara [9]. Wykres 4.10 przedstawia wyniki symulacji modułu 4.2, wykres 4.11 — modułu 4.3.

Kolejny kod z listingu 4.3 modeluje automat obliczający wartość wyniku i działa analogicznie do kodu z listingu 4.2.

Listing 4.3: Algorytm Goertzla – obliczanie wyniku

```

1  always @(posedge clock_sample)
2  begin
3      sample_reg <= sample;
4      sample_counter <= sample_counter+1;
5
6      case (sample_counter)
7          10:
8              begin
9                  fpmult_i2_1_aclr <= 0;
10                 fpmult_i2_2_aclr <= 0;
11                 fpmult_i2_3_aclr <= 0;
12             end
13         20: fpmult_i2_4_aclr <= 0;
14         30: fpadd_i2_aclr <= 0;
15         40: fpsub_i2_aclr <= 0;
16         50: ready <= 1;
17         51: ready <= 0;
18         519:
19             begin
20                 sample_counter <= 0;
21                 q0_tmp <= q0;
22                 q1_tmp <= q1;
23                 q2_tmp <= q2;
24                 fpmult_i2_1_aclr <= 1;
25                 fpmult_i2_2_aclr <= 1;
26                 fpmult_i2_3_aclr <= 1;
27                 fpmult_i2_4_aclr <= 1;

```

```
28 |             fpadd_i2_aclr <= 1;  
29 |             fpsub_i2_aclr <= 1;  
30 |         end  
31 |     endcase  
32 | end
```

4.2.5. UART

Moduł odpowiedzialny za przesyłanie danych z płytki FPGA do komputera protokołem RS232. Jest interfejsem pomiędzy płytką a komputerem. Ze względu na brak obecności jakiegokolwiek interfejsu w module prototypowym wybrano ten standard jako najprostszy w implementacji.

Komunikacja w tym standardzie niesie ze sobą wiele niedogodności, jedną z których jest mała prędkość przesyłu. W tej aplikacji wynosi ona 115,2 kB/s i jest standardową dla RS232. Kolejną niedogodnością jest przestarzałość standardu; konieczne jest stosowanie przejściówek oraz dodatkowych układów pośredniczących. Obecnie, najlepszym interfejsem w pośredniczeniu pomiędzy komputerem a modułem FPGA jest zastosowanie Ethernetu. To rozwiązanie zapewnia wymaganą prędkość przesyłu danych oraz rozwiązuje problem dostępności elementów przez swoją wszechobecność oraz popularność.

Zaprogramowany moduł jest prostą maszyną stanów reagującą na asynchroniczny sygnał obecności danych, który aktywuje synchroniczną maszynę stanów, działającą w takt zegara 115,2 kHz. Po przesłaniu wszystkich bitów danego bajtu maszyna stanów przechodzi w stan bezczynności, zwalnia sygnał zajętości i oczekuje na następne dane.

4.2.6. Aplikacja PC

Program napisany w języku Python odpowiada za przetworzenie danych przesłanych z modułu FPGA do komputera. Wynikiem są poszczególne bity sygnału oraz zdekodowany z nich czas.

Algorytm działania polega na, po pierwsze, znalezieniu znacznika nowej minuty (co jest sygnalizowane długim impulsem w stanie wysokim) a następnie dekodowaniu bitów aż do następnego znacznika nowej minuty. Decyzja czy bit jest jedynką czy zerem jest ustalana na podstawie długości stanu niskiego, tak jak zostało to przedstawione teoretycznie w rozdziale 2.

4.3. Symulacja

Dla sprawdzenia poprawności działania, wszystkie moduły zostały symulowane i testowane w symulatorze Modelsim Altera Edition, wersja 10.4d. Ta wersja symulatora jest ulepszona względem poprzednich i zawiera możliwość symulowania projektów mieszanych (Mixed-language, mixed-HDL simulation), tzn. zbudowanych jednocześnie z kodu Verilog i VHDL. Było to niezmiernie użyteczne przy symulacji modułów filtra FIR, gdzie wytworzony przez kreator IP kod jest właśnie kodem mieszanym.

Symulacja polegała na napisaniu skryptów (.do files) w języku TCL, jednostek testowych (testebench) w języku Verilog oraz skryptu w języku Python. Skrypty symulacyjne zostały rozdzielone na moduły. Jest to podyktowane złożonością projektu oraz długim czasem symulacji w sytuacji symulacji całego projektu. Zamiast tego stosowano wspomniany podział na moduły — każda jednostka testowa czyta plik wejściowy, przetwarza zawarte w nim dane oraz zapisuje te dane do pliku wyjściowego. Następnie plik wyjściowy trafia jako wejście do kolejnego modułu i algorytm się powtarza.

Sama procedura symulacji rozpoczyna się od zapisu bitowego pełnej minuty na podstawie czasu. Konwersji zadanego czasu na komunikat w formie bitów dokonuje skrypt w języku Python (rys. 4.8). Dalszym krokiem jest wytworzenie zmodulowanej fali nośnej w formacie DCF77 (rys. 4.9). Następnie następuje już właściwy proces przetwarzania sygnału – stosowane są filtry decymujące oraz algorytm Goertzla (rys. 4.10, rys. 4.11). Na końcu następuje zdekodowanie nadawanego komunikatu oraz interpretacja danych w aplikacji PC w języku Python (rys. 4.12, rys. 4.13).

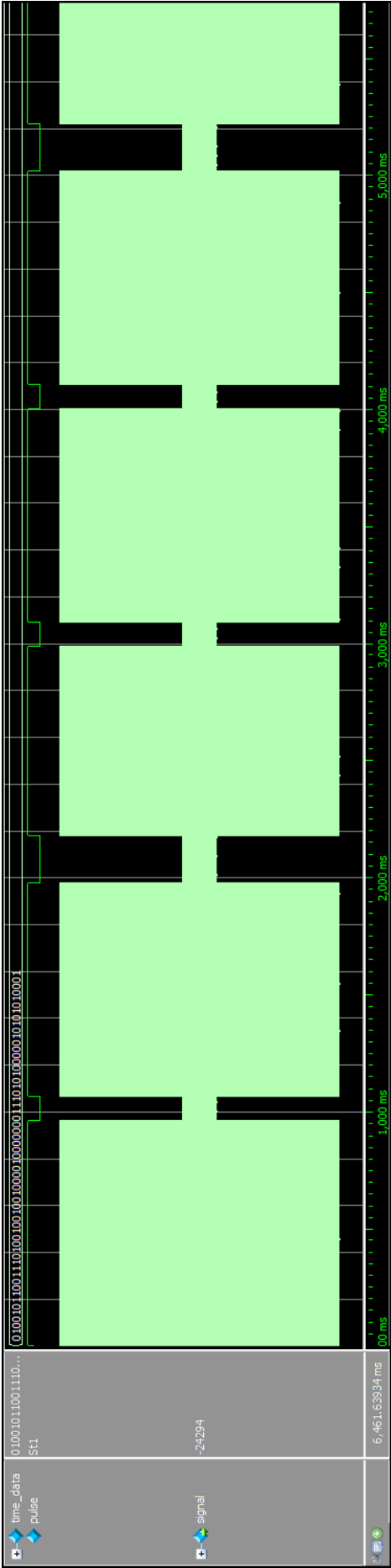
Symulowano, testowano i weryfikowano w całości moduły napisane własnoręcznie. Moduły IP testowano jedynie w celu sprawdzenia, czy dobrane parametry i współczynniki filtrów są właściwe („All the IP cores have been thoroughly tested and verified[...]”, źródło: [2]).

```
λ python encode_bits.py
encoded time:
2015-08-17 00:04:00
Normal antenna
No time change
Summer time
No leap second

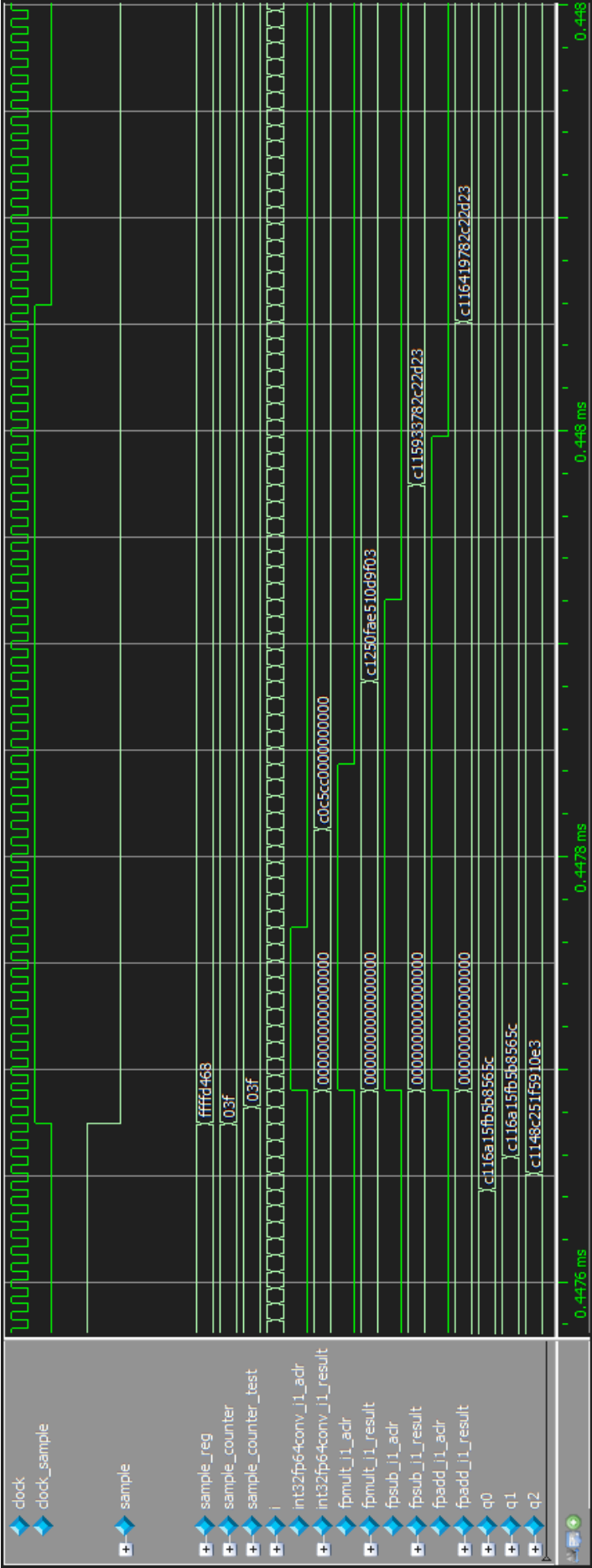
dcf77 message bits:
0000000000000000000010100100001000000011101010000010101010001

message length: 59
```

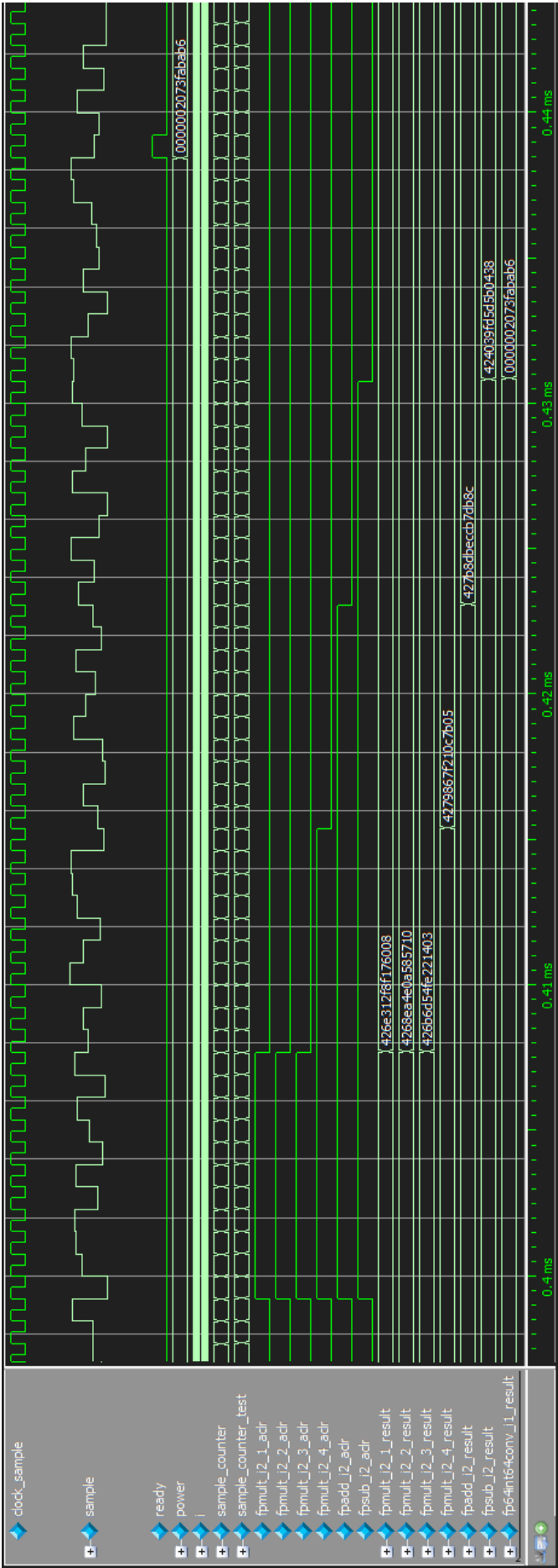
Rysunek 4.8: Generowanie komunikatu bitowego DCF77 dla zadanej godziny



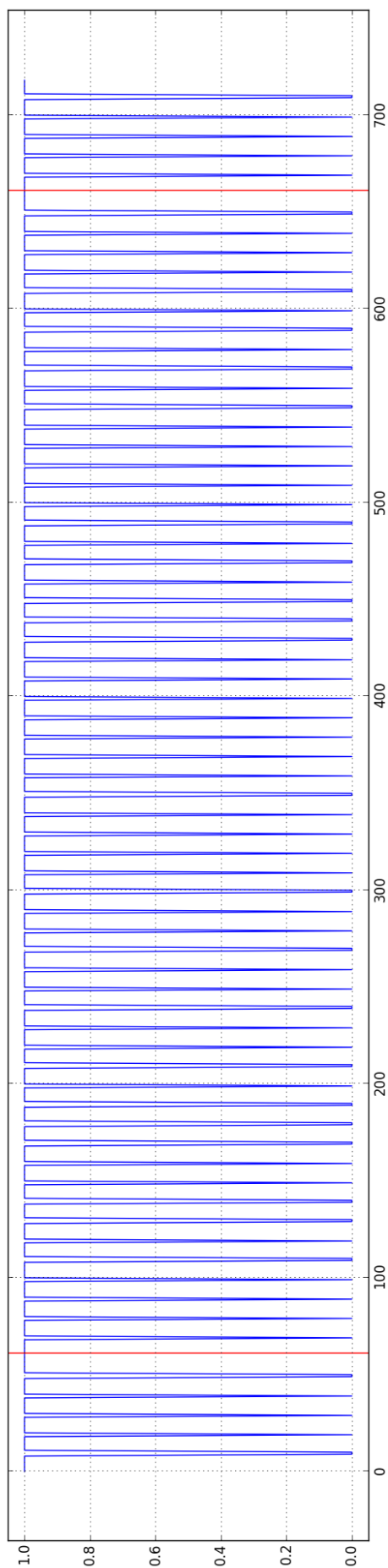
Rysunek 4.9: Generowanie fali nośnej DCF77



Rysunek 4.10: Algorytm Goertzla – pętla (symulacja przepływu danych)



Rysunek 4.11: Algorytm Goertzla – wyniki (symulacja przepływu danych)



Rysunek 4.12: Wyszukiwanie początku nowej minuty oraz dekodowanie bitów

```
λ python decode_bits.py
Input bits:
01001011001110100100100100001000000011101010000010101010001

Decoded time:
2015-08-17 00:04
Normal antenna
No time change
Summer time
No leap second
```

Rysunek 4.13: Dekodowanie godziny z odebranego komunikatu bitowego DCF77

Rezultaty symulacji pokazują, że programowane moduły działają prawidłowo a pierwotny cel pracy jakim było stworzenia dekodera DCF77 został osiągnięty. Jak widać na rysunku 4.9, wygenerowano sygnał wejściowy bez szumu. Dla zastosowania systemu w warunkach realistycznych należałoby przeprowadzić również symulacje z różnymi poziomami szumu w sygnale. Ta niedogodność nie umniejsza jednak doboru algorytmu; ulepszyć należałoby głównie aplikację dekodującą na PC. W takim przypadku algorytm Goertzela podawałby jedynie bardziej rozmyte wartości, zaś zadaniem dekodera odpowiadającego za wyszukiwanie początku komunikatu oraz kwantyzację do wartości 0 lub 1 (rys. 4.12, 4.13) byłoby ich prawidłowe rozróżnienie.

Zakończenie

Przedstawiono metody zaimplementowania przykładowego algorytmu SDR jakim jest niewątpliwie algorytm Goertzla, użyty w aplikacji dekodowania czasu DCF77. Zbudowano i symulowano model nadajnika (generator fali nośnej), odbiornika (filtry, algorytm Goertzla) oraz dekodera (aplikacja PC).

Wydajność rozwiązania sprzętowego przewyższa rozwiązanie programowe chociażby ze względu na równoległe działanie wielu modułów, których zaimplementowanie w komputerze PC wiązałoby się z potrzebą zastosowania większej ilości zasobów.

Prezentowane rozwiązanie nie jest z pewnością optymalnym; w dalszych pracach nad układem należałoby zoptymalizować algorytm Goertzla, dodać do płytki wydajniejszy interfejs np. Ethernet oraz zaimplementować interfejs do popularnego programu SDR, na przykład GNU Radio. Umożliwi to szersze wykorzystanie prezentowanego systemu, niż tylko w aplikacji „dekodowania sygnału wzorca czasu DCF77”.

Zawartość płyty

Praca dyplomowa, wyniki symulacji, kody źródłowe, projekty i skompilowane pliki umieszczono na płycie w następującym porządku:

- **/hardware** – wszelkie pliki związane z projektem fizycznym PCB
- **/software** – pliki źródłowe implementacji sprzętowej, w tym:
 - **/hdl/src** – pliki źródłowe projektu programu Altera Quartus Prime, w którym skompilowano wsad dla FPGA
 - **/hdl/simulation** – pliki źródłowe w języku Python oraz skrypty ModelSim używane do symulacji
- **/thesis** – pliki źródłowe pracy w formacie \LaTeX

Słownik terminów

ADC (*ang. Analog to Digital Converter*) — przetwornik analogowo-cyfrowy.

ARM (*ang. Advanced RISC Machine, dawniej Acorn RISC Machine*) — rodzina procesorów produkowana przez firmę ARM Ltd., szeroko wykorzystywana obecnie w urządzeniach wbudowanych.

ASIC (*ang. Application Specific Integrated Circuit*) — układ elektroniczny zaprojektowany do pełnienia z góry określonego działania.

BCD (*ang. Binary-Coded Decimal*) — sposób zapisu cyfr liczby dziesiętnej polegający na użyciu jej 4 młodszych bitów w reprezentacji binarnej.

BNC (*ang. Bayonet Neill-Concelman*) — złącze stosowane z kablem koncentrycznym, często spotykane w aparaturze pomiarowej (np. oscyloskopach).

CAD (*ang. Computer Aided Design*) — zastosowanie komputera w projektowaniu technicznym.

CIC (*ang. Cascaded Integrator-Comb*) — filtr cyfrowy będący połączeniem integratora, bloku zmiany częstotliwości próbkowania oraz różniczki.

DCF77 — Niemiecki sygnał czasu nadawany na falach długich.

DSP (*ang. Digital Signal Processing, Digital Signal Processor*) — przetwarzanie sygnałów (lub) procesor specjalizowany posiadający dodatkowe funkcje przydatne przetwarzaniu sygnałów, np. mnożenie z akumulacją.

DTMF (*ang. Dual Tone Multi Frequency*) — system sygnalizacji pomiędzy urządzeniami telefonicznymi.

DVB-T (*ang. Digital Video Broadcasting – Terrestrial*) — standard cyfrowej telewizji naziemnej.

FIR (*ang. Finite Impulse Response*) — filtr cyfrowy o skończonej odpowiedzi impulsowej.

FPGA (*ang. Field Programmable Gate Array*) — programowalny układ logiczny o dużym stopniu scalenia.

GSM (*ang.* Global System for Mobile Communications) — najpopularniejszy standard telefonii komórkowej drugiej generacji.

ITU (*ang.* International Telecommunication Union) — organizacja ustanowiona przez ONZ w celu standaryzacji i regulowania rynku tele- i radiokomunikacyjnego.

PC (*ang.* Personal Computer) — komputer osobisty.

PCB (*ang.* Printed Circuit Board) — płytka z izolacyjnego materiału na której są umieszczone układy elektroniczne oraz ścieżki łączące je elektrycznie.

PLL (*ang.* Phase Locked Loop) — układ elektroniczny działający na zasadzie sprzężenia zwrotnego, zwykle służący do automatycznej regulacji częstotliwości.

SDR (*ang.* Software Defined Radio) — system radiowy w którym większość tradycyjnych elementów odbiorczych (lub nadawczych) została zastąpiona oprogramowaniem.

SMD (*ang.* Surface-Mount Devices) — rodzaj elementów elektronicznych nie posiadających drutów połączeniowych a jedynie metalowe końcówki, które umożliwiają przylutowanie elementu na płytkę drukowaną bez potrzeby wiercenia otworów.

SPICE (*ang.* Simulation Program with Integrated Circuits Emphasis) — oprogramowanie symulujące analogowe układy elektroniczne oraz schematy ideowe.

Spis rysunków

1.1	Idea radia programowalnego	7
2.1	Zasięg sygnału DCF77	10
2.2	Wykres zegarowy sygnału DCF77	11
2.3	Obwiednia sygnału DCF77	13
2.4	Znacznik nowej minuty DCF77	13
3.1	Architektura modułowa części sprzętowej	14
3.2	Moduł prototypowy Terasic DE0-Nano	15
3.3	Schemat ideowy modułu prototypowego	16
3.4	Układ chroniący przed przepięciem	17
3.5	Filtr antyaliasingowy	17
3.6	Qucs – projektowanie filtrów	18
3.7	Qucs – symulowana charakterystyka zaprojektowanego filtra	18
3.8	Płytką drukowaną — rendering	19
3.9	Płytką drukowaną — wykonanie fizyczne	20
4.1	Architektura modułowa części programowej	21
4.2	Katalog bloków IP w programie Quartus	22
4.3	Przewidywanie zajętości zasobów logicznych projektowanego modułu	23
4.4	Współczynniki w kreatorze i symulacja charakterystyki	24
4.5	Przykładowe sygnały modułu „FIR II”	24
4.6	Konfiguracja modułu „CIC”	25
4.7	Filtr CIC – budowa	26
4.8	Generowanie komunikatu bitowego DCF77 dla zadanej godziny	30
4.9	Generowanie fali nośnej DCF77	31
4.10	Algorytm Goertzla – pętla (symulacja przepływu danych)	32
4.11	Algorytm Goertzla – wyniki (symulacja przepływu danych)	33
4.12	Wyszukiwanie początku nowej minuty oraz dekodowanie bitów	34
4.13	Dekodowanie godziny z odebranego komunikatu bitowego DCF77	35

Spis tabel

2.1	Poszczególne bity sygnału DCF77	12
-----	---	----

Bibliografia

- [1] Ahlstrom, J. *HiQSDR: a Software Defined Radio Transceiver*. <http://hiqsdr.com/>.
- [2] Altera. *IP Evaluation and Purchase*. <https://www.altera.com/support/support-resources/download/megacore-ip.html>.
- [3] Altera. *AN455: Understanding CIC Compensation Filters*. https://www.altera.com/en_US/pdfs/literature/an/an455.pdf, 2007.
- [4] Altera. *ALTPLL (Phase-Locked Loop) IP Core User Guide*. https://www.altera.com/literature/ug/ug_altp11.pdf, 2014.
- [5] Altera. *CIC IP Core User Guide*. https://www.altera.com/en_US/pdfs/literature/ug/ug_cic.pdf, 2014.
- [6] Altera. *Cyclone IV Device Handbook*. <https://www.altera.com/literature/hb/cyclone-iv/cyclone4-handbook.pdf>, 2014. Volume 1, 2, 3.
- [7] Altera. *NCO IP Core User Guide*. https://www.altera.com/en_US/pdfs/literature/ug/ug_nco.pdf, 2014.
- [8] Altera. *FIR II IP Core User Guide*. https://www.altera.com/literature/ug/ug_fir_compiler_ii.pdf, 2015.
- [9] Altera. *Floating-Point IP Cores User Guide*. https://www.altera.com/en_US/pdfs/literature/ug/ug_altfp_mfug.pdf, 2015.
- [10] Altera. *Introduction to Altera IP Cores*. https://www.altera.com/literature/ug/ug_intro_to_megafunctions.pdf, 2016.
- [11] Banks, K. *The Goertzel Algorithm*. <http://www.embedded.com/design/configurable-systems/4024443/The-Goertzel-Algorithm>.
- [12] Bauch, A., Hetzel, P., Piester, D. *Time and Frequency Dissemination with DCF77: From 1959 to 2009 and beyond*. *PTB-Mitteilungen*, 119(3), 2009.
- [13] Bieńkowski, Z. *Poradnik ultrakrótkofalowca*. Wydawnictwa Komunikacji i Łączności, Warszawa, 1988.
- [14] Choraś, R. *Przetwarzanie sygnałów*. Wydawnictwa Uczelniane Uniwersytetu Technologiczno-Przyrodniczego, Bydgoszcz, 2014.
- [15] Elecrow. *PCB Prototyping*. <http://www.elecrow.com/>.
- [16] Ettus Research. *USRP Software Defined Radio*. <https://www.ettus.com/product>.
- [17] Frąc, C. *O sygnałach bez calek*. Pracownia Wydawnicza „ElSet”, Olsztyn, 2012.

- [18] Great Scott Gadgets. *HackRF One*. <https://greatscottgadgets.com/hackrf/>.
- [19] Hajduk, Z. *Wprowadzenie do języka VERILOG*. Wydawnictwo BTC, Legionowo, 2009.
- [20] Hanzo, L. *Adaptive wireless transceivers turbo-coded, turbo-equalized and space-time coded TDMA, CDMA, and OFDM systems*. Wiley, New York, USA, 2002.
- [21] Hogenauer, E. An economical class of digital filters for decimation and interpolation. *IEEE Transactions on Acoustics Speech and Signal Processing*, 29, 1981.
- [22] Horowitz, P. *Sztuka elektroniki*. Wydawnictwa Komunikacji i Łączności, Warszawa, 2009.
- [23] Hołubowicz, W. *Systemy radiowe z rozpraszaniem widma, CDMA: teoria, standardy, aplikacje*. Motorola Polska, Poznań, 1998.
- [24] Intersil. *AN1325: Choosing and Using Bypass Capacitors*. <https://www.intersil.com/data/an/an1325.pdf>, 2011.
- [25] Jackowski, S. *Telekomunikacja*. Politechnika Radomska, Radom, 2005.
- [26] Janoś, T. *Vademecum teleinformatyka III: komunikacja mobilna, bezpieczeństwo, technologie i protokoły sieciowe*. IDG Poland, Warszawa, 2004.
- [27] Johnson, P. *New research lab leads to unique radio receiver*. *E-Systems Team*, 5(4):6–7, 1985.
- [28] Kapruziak, M., Olech, B. *Dedykowany procesor do cyfrowej kowersji częstotliwości w dół. Poznańskie Warsztaty Telekomunikacyjne*, (9):261–265, 2004.
- [29] Kester, W. *Przetworniki A/C i C/A: teoria i praktyka*. Wydawnictwo BTC, Legionowo, 2012.
- [30] Kwiatkowski, W. *Wstęp do cyfrowego przetwarzania sygnałów*. Wojskowa Akademia Techniczna, Warszawa, 2003.
- [31] Leśnicki, A. *Technika cyfrowego przetwarzania sygnałów*. Wydawnictwo Politechniki Gdańskiej, Gdańsk, 2014.
- [32] Lipiński, W. *Modulacja, kodowanie i transmisja w systemach telekomunikacyjnych*. Wydawnictwo Uczelniane Politechniki Szczecińskiej, Szczecin, 2001.
- [33] Lipiński, W. *Transmisja danych w telefonii komórkowej*. Wydawnictwo Uczelniane Politechniki Szczecińskiej, Szczecin, 2003.
- [34] Majewski, J. *Układy FPGA w przykładach*. Wydawnictwo BTC, Warszawa, 2007.
- [35] Linear Technology. *LTC2208 - 16-Bit, 130Msps ADC*. <http://www.linear.com/product/LTC2208>.
- [36] Linear Technology. *LTC6405 - 2.7GHz, 5V, Low Noise, Rail-to-Rail Input Differential Amplifier/Driver*. <http://www.linear.com/product/LTC6405>.
- [37] Linear Technology. *Application Notes for Thermally Enhanced Leaded Plastic Packages*. <http://www.linear.com/docs/42139>, 2012.
- [38] Mitola, J. *Software Radios: Survey, Critical Evaluation and Future Directions*. *IEEE Aerospace and Electronic Systems Magazine*, 8(4):25–36, 1993.

- [39] Murata. *Application Manual for Power Supply Noise Suppression and Decoupling for Digital ICs*. <http://www.murata.com/~media/webrenewal/support/library/catalog/products/emc/emifil/c39e.ashx>, 2010.
- [40] Opendous. *Upconverter 125 MHz: Passive Upconverter Hardware Design for SDR*. <https://github.com/opendous/Upconverter1v3>.
- [41] OpenHPSDR. *Hermes – a DUC/DDC transceiver*. <http://openhpsdr.org/hermes.php>.
- [42] OpenHPSDR. *Mercury – a direct sampling front end*. <http://openhpsdr.org/mercury.php>.
- [43] Ott, H. *Partitioning and Layout of a Mixed-Signal PCB. Printer Circuit Design & Fab*, 2001.
- [44] Purczyński, J. *Wybrane zagadnienia teorii sygnałów*. Wydawnictwo Uczelniane Politechniki Szczecińskiej, Szczecin, 2000.
- [45] Qucs. *Quite Universal Circuit Simulator*. <http://qucs.sourceforge.net/>.
- [46] Redmayne, D. *A Short Course in PCB Layout for High-Speed ADCs*. <http://www.linear.com/solutions/1809>.
- [47] Schulze, H. *Theory and applications of OFDM and CDMA wideband wireless communications*. Wiley, Chichester, England, 2005.
- [48] Skyworks. *PIN Limiter Diodes in Receiver Protectors*. <http://www.skyworksinc.com/uploads/documents/200480C.pdf>, 2008.
- [49] Skyworks. *RF Diode Design Guide*. http://www.skyworksinc.com/downloads/literature/RF_Diode_Design_Guide.pdf, 2011.
- [50] Smith, S. *Cyfrowe przetwarzanie sygnałów: praktyczny poradnik dla inżynierów i naukowców*. Wydawnictwo BTC, Warszawa, 2007.
- [51] Szabatin, J. *Podstawy teorii sygnałów*. Wydawnictwa Komunikacji i Łączności, Warszawa, 2003.
- [52] Țariov, A. *Wprowadzenie do modelowania sygnałów telekomunikacyjnych w środowisku Matlab-Simulink*. Wydawnictwo Uczelniane Politechniki Szczecińskiej, Szczecin, 2008.
- [53] Terasic. *DE0-Nano Development and Education Board*. <http://de0-nano.terasic.com.tw/>.
- [54] Terasic. *DE0-Nano Schematic*. ftp://ftp.altera.com/up/pub/Altera_Material/Boards/DE0-Nano/DE0_Nano_Datasheets.zip, 2012.
- [55] Terasic. *DE0-Nano User Manual*. ftp://ftp.altera.com/up/pub/Altera_Material/Boards/DE0-Nano/DE0_Nano_User_Manual.pdf, 2012.
- [56] Texas Instruments. *SLAA510: High-Speed, Analog-to-Digital Converter Basics*. <http://www.ti.com/lit/an/sl原因510/sl原因510.pdf>, 2011.
- [57] Texas Instruments. *SLAA594A: Why Oversample when Undersampling can do the Job?* <http://www.ti.com/lit/an/sl原因594a/sl原因594a.pdf>, 2013.

- [58] Vectron. *Signal Types and Terminators*. http://www.vectron.com/products/literature_library/Signal_Types_and_Terminations.pdf, 2014.
- [59] Weedman, K. *Verilog Class Lectures*. <http://verilog.openhpsdr.org/>.
- [60] Wesołowski, K. *Podstawy cyfrowych systemów telekomunikacyjnych*. Wydawnictwa Komunikacji i Łączności, Warszawa, 2003.
- [61] Wikipedia. *List of software-defined radios*. https://en.wikipedia.org/wiki/List_of_software-defined_radios.
- [62] Wilkinson, B. *Układy cyfrowe*. Wydawnictwa Komunikacji i Łączności, Warszawa, 2003.
- [63] Zieliński, T. *Cyfrowe przetwarzanie sygnałów: od teorii do zastosowań*. Wydawnictwa Komunikacji i Łączności, Warszawa, 2007.
- [64] Zieliński, T., Korohoda, P., Rumian, R. *Cyfrowe przetwarzanie sygnałów w telekomunikacji: podstawy, multimedia, transmisja*. Wydawnictwo Naukowe PWN, Warszawa, 2014.