

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
Національний університет «Запорізька політехніка»

Інститут інформатики та радіоелектроніки  
Факультет комп'ютерних наук та технологій  
(повне найменування інституту, факультету)

Кафедра комп'ютерних систем та мереж  
(повне найменування кафедри )

**Пояснювальна записка**

до дипломного проєкту (роботи)

магістра

(ступінь вищої освіти (освітній ступінь))

на тему КОНФІГУРУЄМА РАДІОСИСТЕМА НА ПЛІС

---

Виконав: студент 5 курсу, групи КНТз-511м

спеціальності \_\_\_\_\_

123 «Комп'ютерна інженерія»

(код і найменування спеціальності)

Освітня програма (спеціалізація) \_\_\_\_\_

«Комп'ютерна інженерія»

Немінський М.І.

(прізвище та ініціали)

Керівник Грушко С.С.

(прізвище та ініціали)

Рецензент Єфименко М.В.

(прізвище та ініціали)

м. Запоріжжя  
2022 рік

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
**Національний університет «Запорізька політехніка»**  
(повне найменування вищого навчального закладу)

Інститут, факультет інформатики та радіоелектроніки, комп'ютерних наук і технологій  
Кафедра «Комп'ютерні системи та мережі»  
Ступінь вищої освіти (освітній ступінь) магістерський  
Спеціальність 123 Комп'ютерна інженерія  
(код і найменування)  
Освітня програма (спеціалізація) Комп'ютерна інженерія  
(назва освітньої програми (спеціалізації))

**ЗАТВЕРДЖУЮ**

Завідувач кафедри Кудерметов Р.К.

“ ” 2022 року

**З А В Д А Н Н Я**  
**НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА**

Немінський Максим Ігорович

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Конфігуруєма радіосистема на ПЛІС

керівник проєкту (роботи) Грушко Світлана Сергіївна, к. т. н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “07” листопада 2022 року №  
367

2. Строк подання студентом проєкту (роботи) 10 грудня 2022 року

3. Вихідні дані до проєкту (роботи) мова опису VHDL, мова опису VerilogHDL,  
інтегроване середовище розробки програмно-апаратних засобів QUARTUS,  
документація на стенд, документація SDR

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1) Огляд аналогів і вибір засобів проектування;

2) Розробка SDR приймача на ПЛІС;

3) Керівництво користувача.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

ПЛ1 - Функціональна схема SDR приймача супергетеродинного типу

ПЛ2 - Функціональна схема SDR приймача прямого перетворення.

Е1 - Стуркurna схема SDR приймача на ПЛІС

ПЛ3 - Структурна схема SDR приймача прямого перетворення

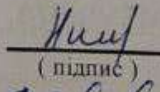
# 6. Консультанти розділів проекту (роботи)

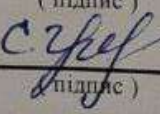
| Розділ       | Прізвище, ініціали та посада консультанта | Підпис, дата                                                                        |                                                                                     |
|--------------|-------------------------------------------|-------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------|
|              |                                           | завдання видав                                                                      | прийняв виконане завдання                                                           |
| 1-3          | Грушко С.С., к. т. н., доцент             |  |  |
| Нормконтроль | Щербак Н.В., ст. викладач                 |                                                                                     |                                                                                     |
|              |                                           |                                                                                     |                                                                                     |
|              |                                           |                                                                                     |                                                                                     |
|              |                                           |                                                                                     |                                                                                     |
|              |                                           |                                                                                     |                                                                                     |

7. Дата видачі завдання 01.10.2022 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів дипломного проекту (роботи)                                                                             | Строк виконання етапів проекту (роботи) | Примітка |
|-------|----------------------------------------------------------------------------------------------------------------------|-----------------------------------------|----------|
| 1     | Аналіз наявних методів та алгоритмів пошуку об'єктів, перегляд отриманих результатів, виділення недоліків та переваг | 10.10.2022 р.                           |          |
| 2     | Розробка структури системи                                                                                           | 15.10.2022 р.                           |          |
| 3     | Розробка алгоритму роботи системи                                                                                    | 30.10.2022 р.                           |          |
| 4     | Реалізація системи уточненого розпізнавання                                                                          | 01.11.2022 р.                           |          |
| 5     | Дослідження системи уточненого розпізнавання                                                                         | 20.11.2022 р.                           |          |
| 6     | Оформлення отриманих результатів у ПЗ                                                                                | 25.11.2022 р.                           |          |
| 7     | Оформлення графічного матеріалу                                                                                      | 01.12.2022 р.                           |          |
| 8     | Оформлення допоміжного матеріалу                                                                                     | 10.12.2022 р.                           |          |
|       |                                                                                                                      |                                         |          |
|       |                                                                                                                      |                                         |          |
|       |                                                                                                                      |                                         |          |
|       |                                                                                                                      |                                         |          |
|       |                                                                                                                      |                                         |          |

Студент  **М.І. Немінський**  
(підпис) (ініціали та прізвище)

Керівник проекту (роботи)  **С.С. Грушко**  
(підпис) (ініціали та прізвище)

## РЕФЕРАТ

ПЗ: 71 с., 31 рис., 3 табл., 1 додаток, 3 плакати, 1 креслення, 23 джерела.

ПРОГРАМУВАННЯ МІКРОКОНТРОЛЕРІВ, ПЛІС, ALTERA, FPGA, HDSDR, QUARTUS II, SDR , VHDL, VERILOGHDL.

Об'єкт дослідження – прийом та передача радіосигналів за допомогою ПЛІС.

Мета дослідження – розробити SDR приймач на ПЛІС.

Дипломна робота складається з трьох розділів.

У першому розділі розглядаються аналогі та засоби проектування, апаратне забезпечення, вибирається засіб проектування для роботи.

У другому розділі описано схему, показані основні типи даних та алгоритм роботи, основні частини лістингів та нюанси програмування ПЛІС та зроблені відповідні висновки .

У третьому розділі описано керівництво користувача, налаштування програм для роботи зі схемою управління.

В результаті виконання роботи розроблено SDR приймач на ПЛІС De0-Nano за допомогою мов опису апаратури VHDL, Verilog HDL, що дозволяє приймати сигнали у КВ діапазоні.

## ЗМІСТ

|                                                           |    |
|-----------------------------------------------------------|----|
| Вступ.....                                                | 8  |
| 1 Огляд аналогів і вибір засобів проектування .....       | 10 |
| 1.1 Altera De0-Nano .....                                 | 10 |
| 1.2 Artix-7 Fpga Ac701 .....                              | 13 |
| 1.3 Архітектура ПЛІС(FPGA).....                           | 14 |
| 1.4 Вибір програмного засобу .....                        | 19 |
| 1.5 Постановка Задачі .....                               | 27 |
| 2 Розробка SDR приймача на ПЛІС .....                     | 27 |
| 2.1 Software-Defined Radio.....                           | 27 |
| 2.3 Прийом та передача радіосигналів у ПЛІС.....          | 47 |
| 2.4 Розробка SDR на платі De0-Nano.....                   | 57 |
| 3 Керівництво користувача.....                            | 65 |
| 3.1 Програмне забезпечення Quartus II(Quartus Prime)..... | 65 |
| 3.2 Налаштування програми HDSDR для приймача SDR.....     | 66 |
| Висновки.....                                             | 70 |
| Перелік джерел посилання .....                            | 71 |
| Додаток А .....                                           | 73 |

Графічний матеріал:

Плакат 1 – Функціональна схема SDR приймача супергетеродинного типу

Плакат 2 – Структурна схема SDR приймача / передавача прямого перетворення архітектури DDC / DUC

Креслення 1 – Структурна схема SDR приймача на ПЛІС

Плакат 4 – Спрощена архітектура ПЛІС

## СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

|                |                                                                                                            |
|----------------|------------------------------------------------------------------------------------------------------------|
| АЦП            | – аналого-цифровий перетворювач                                                                            |
| НВІС           | – надвелика інтегральна схема                                                                              |
| ПЛІС           | – програмована логічна інтегральна схема                                                                   |
| ЦП             | – центральний процесор                                                                                     |
| API            | – прикладний програмний інтерфейс, Application Programming Interface                                       |
| ASIC           | – інтегральна схема для специфічного застосування, Application-specific integrated circuit                 |
| DAC            | – цифро-аналоговий перетворювач, Digital to Analog Converter                                               |
| DDR            | – подвоєна швидкість передачі даних, Double Data Rate                                                      |
| DATA_RATE      | – визначає режим вихідних даних (SDR/DDR)                                                                  |
| DATA_WIDTH     | – визначає розрядність шини даних на виході вузла (від 1 до 8)                                             |
| EDC            | – електричний проектний струм, Electrical Design Current                                                   |
| HID            | – клас пристроїв USB для взаємодії з людиною, Human Interface Device                                       |
| HS             | – горизонтальна синхронізація, Horizontal Synchronization                                                  |
| IP             | – міжмережевий протокол, Internet Protocol                                                                 |
| ISE            | – багатофункціональне рішення, що закриває повний спектр питань контролю доступу, Identity Services Engine |
| INTERFACE_TYPE | – тип інтерфейсу передачі                                                                                  |
| MATLAB         | – пакет прикладних програм для числового аналізу, а також мова програмування, Matrix Laboratory            |
| PLD            | – programmable logic device, програмований логічний пристрій                                               |
| SDK            | – набір засобів розробки, Software Development Kit                                                         |
| SOPC           | – поєднання ядра процесора із спеціальним обладнанням, System-On-A-Programmable-Chip                       |
| SPI            | – послідовний периферійний інтерфейс, Serial Peripheral Interface                                          |

SERDES\_MODE – режим підключення ISERDES у разі використання кількох серіалайзерів для отримання даних від одного контакту ПЛІС

USB – послідовний інтерфейс для підключення периферійних пристроїв до обчислювальної техніки, Universal Serial Bus

VHDL – мова опису апаратури інтегральних схем, Very high speed integrated circuits Hardware Description Language

Verilog HDL – мова опису апаратури, що використовується для опису та моделювання електронних систем

## ВСТУП

В даний час телекомунікаційні системи мають найважливішу роль у житті людей. У той час, коли удосконалювалися способи передачі даних, одночасно еволюціонувала і елементна база таких пристроїв. У різні проміжки часу з'являлися різні способи виділення корисного сигналу з робочого діапазону частот. З розвитком приймально-передавальних пристроїв з'явилася технологія SDR або Software-defined radio (програмно конфігуруєме радіо), такі пристрої можуть як приймати, так і передавати радіосигнал. Технологія SDR має наступні переваги:

- перегляд спектру сигналу на екрані ПК;
- фільтри, що налаштовуються програмно;
- функціонал для вимірювань сигналу;
- можливість настройки прийому / передачі різних модуляцій.

Подібні радіосистеми широко застосовуються для військових додатків та бездротових послуг зв'язку, оскільки дозволяють обслуговувати велику кількість радіопротоколів. Прикладом реалізації технології SDR у військовій сфері є проект ESSOR(European Secure Software-defined Radio) – проект програми Європейського Союзу «Структурна співпраця у сфері оборони та безпеки» (англ. Permanent Structured Cooperation, PESCO) з розробки програмно-конфігурованих радіосистем телекомунікацій для забезпечення інтероперабельного захищеного зв'язку в тактичному ланцюжку управління та в короткому, ультракороткому та дециметровому діапазонах хвиль.

Нині SDR використовуються реалізації простих радіомодемів, зокрема GSM, WiFi, WiMax. Згодом, SDR, можливо, стане основною технологією радіозв'язку. SDR є необхідною умовою для реалізації когнітивного радіо.

При розробці найкращим є тип DDC SDR, який дозволяє реалізувати мультистандартний прийом сигналів без особливої потреби змінювати аналогову частину приймача. Для реалізації таких пристроїв застосовуються АЦП

конверсного типу, що володіють достатньою розрядністю і великою частотою дискретизації, і ПЛІС, що здатні швидко обробити сигнал з АЦП і відправити на ПК для подальшої роботи з сигналом. Недоліки таких пристроїв:

- ціна;
- стаціонарна робота;
- складна алгоритмічна частина.

При використанні SDR приймачів відкриваються великі можливості для налагодження різних сигналів, експериментальних протоколів, а також тестування методів ЦОС. Виходячи з цього, розробка програмно конфігуруємої радіосистеми на ПЛІС типу FPGA з можливістю зміни прошивки ПЛІС є актуальною задачею.

# 1 ОГЛЯД АНАЛОГІВ І ВИБІР ЗАСОБІВ ПРОЕКТУВАННЯ

## 1.1 Altera DE0-Nano

Плата DE0-Nano представляє компактну платформу розробки FPGA, яка підходить для широкого спектру портативних дизайнерських проектів, таких як роботи та мобільні проекти. DE0-Nano ідеально підходить для використання з вбудованими програмними процесорами — він має потужний Altera Cyclone IV FPGA (з 22 320 логічними елементами), 32 Мб SDRAM, 2 Кб EEPROM і 64 Мб запам'ятовуючого пристрою послідовної конфігурації. Для підключення до реальних датчиків DE0-Nano містить 8-канальний 12-розрядний аналого-цифровий перетворювач National Semiconductor, а також 13-розрядний 3-осьовий акселерометр від Analog Devices. Плата DE0-Nano містить вбудований USB Blaster для програмування FPGA, і плата може живитися від цього порту USB або від зовнішнього джерела живлення. Плата містить роз'єми розширення, які можна використовувати для підключення різних дочірніх плат Terasic або інших пристроїв. Входи та виходи включають 2 кнопки, 8 користувальницьких світлодіодів і набір з 4 DIP-перемикачів, як показано на рисунку 1.1 [2].

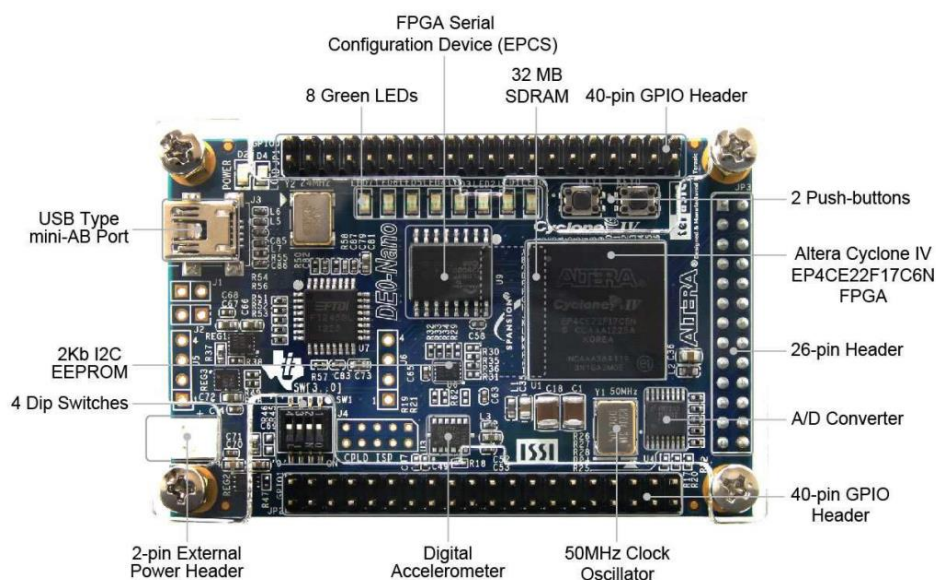


Рисунок 1.1 – Стенд DE0-Nano

### 1.1.1 Блок-схема плати DE0-Nano

Щоб забезпечити максимальну гнучкість для користувача, усі підключення виконуються через пристрій Cyclone IV FPGA. Таким чином, користувач може налаштувати FPGA для реалізації будь-якої системи, як показано на рисунку 1.2.

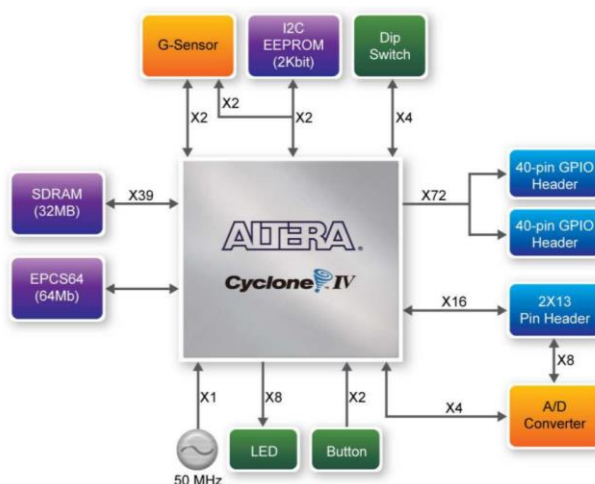


Рисунок 1.2 – Блок-схема DE0-Nano

### 1.1.2 Панель керування DE0-Nano

Плата DE0-Nano оснащена панеллю керування, яка дозволяє користувачам отримувати доступ до різних компонентів плати з головного комп'ютера. Головний комп'ютер зв'язується з платою через USB-з'єднання. Засіб можна використовувати для перевірки функціональності компонентів на платі або використовувати як засіб налагодження під час розробки коду RTL(register transfer level). Це означає, що ваш код VHDL описує, як дані перетворюються, коли вони передаються від реєстру до реєстру. Перетворення даних виконується комбінаційною логікою, яка існує між регістрами. На рисунку 1.3 показана концепція панелі керування DE0-Nano. На платі FPGA реалізована «Контрольна схема», яка виконує функції керування. Він спілкується з вікном панелі керування, яке активно на головному комп'ютері, через посилання USB Blaster. Графічний інтерфейс використовується для видачі команд схемі керування. Він

обробляє всі запити і здійснює передачу даних між комп'ютером і платою DE0-Nano [2].

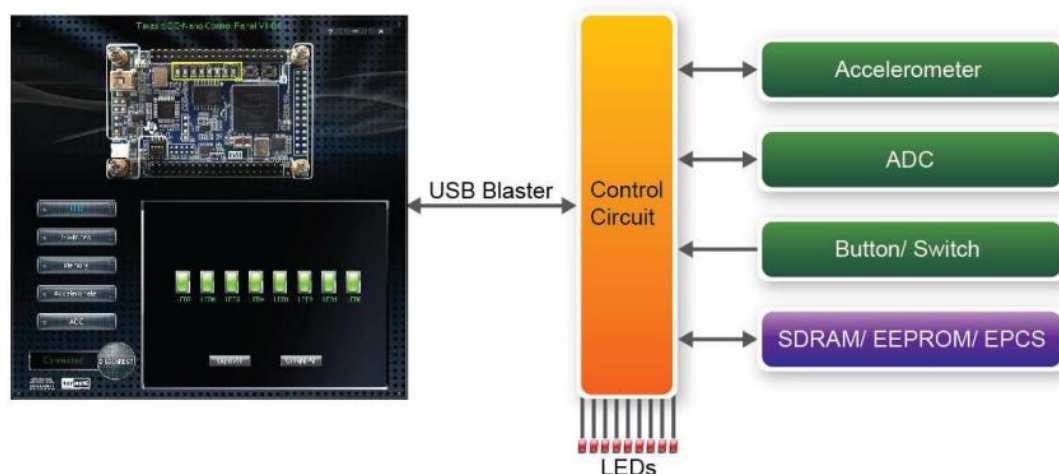


Рисунок 1.3 - Концепція панелі керування DE0-Nano

### 1.1.3 DE0-Nano конструктор системи (System Builder)

DE0-Nano System Builder — це утиліта для Windows, створена для допомоги користувачам у

створення проекту Quartus II для плати DE0-Nano за лічені хвилини.

Згенерований Квартус II, файли проекту включають:

- файл проекту Quartus II (.qpf);
- файл налаштувань Quartus II (.qsf);
- файл дизайну верхнього рівня (.v);
- файл Synopsys Design Constraints (.sdc);
- документ про призначення PIN-коду (.htm).

Надаючи вищезазначені файли, DE0-Nano System Builder допомагає запобігти виникненню ситуацій які схильні до помилок, коли користувачі вручну редагують файл дизайну верхнього рівня або розміщують призначення маркерів[2].

## 1.2 Artix-7 FPGA AC701

Плата AC701 підтримує два з п'яти режимів конфігурації FPGA серії 7:

- головна флеш-пам'ять SPI за допомогою вбудованої флеш-пам'яті Quad SPI;
- jtag за допомогою стандартного USB-кабелю від А до micro-B для підключення головного ПК до AC701, порт конфігурації плати або роз'єм плоского кабелю J4 Platform Cable USB/Parallel Cable IV.

Плата також має цільовий еталонний дизайн, який забезпечує розширений інтерфейс пам'яті та високопродуктивне послідовне підключення, як показано на рисунку 1.4. Дизайнери також можуть скористатися перевагами плати, щоб розширити вхід і вихід (I/O) за допомогою інтерфейсу FPGA Mezzanine Card (FMC) [14].

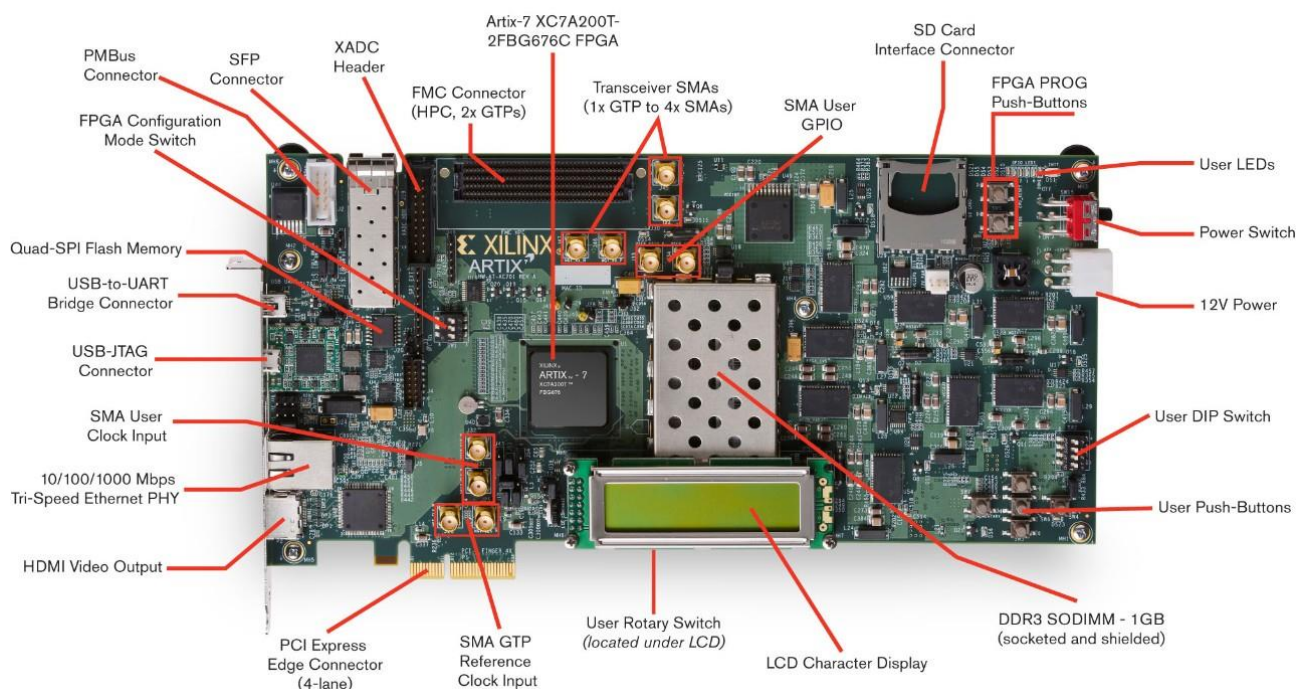


Рисунок 1.4 - Artix-7 FPGA AC701

Переваги:

- USB104 A7 поставляється в компактному форм-факторі PC/104;
- роботу напруги DDR можна налаштувати за допомогою перемикача;
- вбудований програматор Digilent JTAG;
- промисловий температурний пристрій.

Недоліки:

- Низькошвидкісний пристрій для основної ПЛІС;
- лише 16 МБ флеш-пам'яті NOR;
- роз'єм USB типу A, без USB C;
- немає Ethernet, тому вам знадобиться зовнішній PHY PMOD.

Саме через швидкісний пристрій для основної ПЛІС, а також присутність Ethernet, вибір пав на FPGA De0-Nano компанії Altera(Intel), а не на їх прямого конкурента Xilinx(AMD) [14].

### 1.3 Архітектура ПЛІС(FPGA)

Схематично мікросхема та шляхи проходження сигналів показані на рисунку 1.5. Мікросхема містить елементи вводу-виводу, логічні елементи та лінії зв'язку – рядки і стовбці. Спрощений механізм роботи пристрою на ПЛІС виглядає таким чином: сигнал через елементи вводу-виводу потрапляє в мікросхему, де в логічному елементі над ним виконуються необхідні операції. Потім по лініях зв'язку через елементи вводу-виводу сигнал виходить з мікросхеми [4].

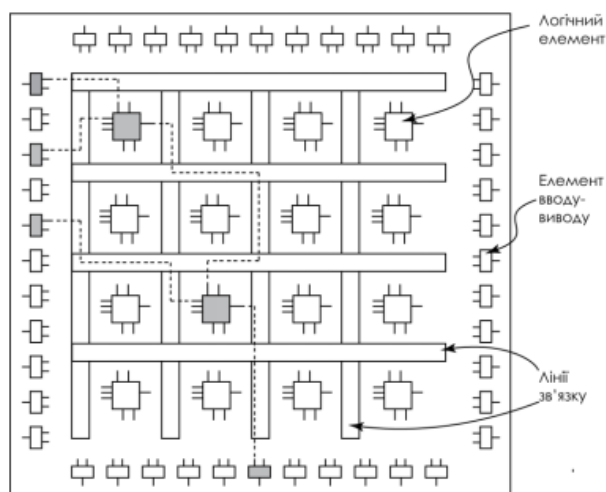


Рисунок 1.5 - Спрощена архітектура ПЛІС

### 1.3.1 Архітектура мікросхем CPLD

Архітектуру CPLD мікросхем розглянемо на прикладі сімейства MAX 3000 компанії Altera. На сьогоднішній день компанія Altera випускає два сімейства з класичною CPLD архітектурою: MAX 3000 та MAX 7000. Параметри мікросхем сімейства MAX 3000 наведені в таблиці 1.1.

Часові параметри, що використовуються в таблиці, мають наступне значення:  $t_{PD}$  – затримка проходження сигналу без тактування крізь мікросхему,  $t_{CO}$  – час між приходом тактового сигналу до появи сигналу на виході мікросхеми,  $f_{CNT}$  – максимальна тактова частота для тригерів.

Таблиця 1.1 – Параметри мікросхем MAX3000

| Параметр        | EPM3032A | EPM3064A | EPM3512A |
|-----------------|----------|----------|----------|
| Макрочарунок    | 32       | 64       | 512      |
| Логічних блоків | 2        | 4        | 32       |
| $t_{PD}$ , нс   | 4,5      | 4,5      | 7,5      |
| $t_{CO}$ , нс   | 3,0      | 3,1      | 4,7      |
| $f_{CNT}$ , МГц | 227,3    | 222,2    | 116,3    |

Як видно з рисунку 1.6 та 1.7 основою мікросхеми MAX 3000 є макрочарунка (Macrocell), яка складається з матриці розподілу термів (Product-Term Select Matrix) та конфігуруемого тригера (Programmable Register).

Для реалізації логічної функції на основі декількох макрочарунок одного логічного блока використовується логічний розширювач (Expander Product Terms). Розділяємий логічний розширювач формує інверсію терма і подає його значення на локальну програмуєму матрицю де він може використовуватись будь-якою макрочарункою логічного блоку. В матрицю з'єднань має в своєму складі надходять 36 сигналів з інших логічних блоків та 16 сигналів з розділяемого паралельного логічного розширювача. Тобто на основі одного логічного блоку можна реалізувати логічну функцію з 52 термів .

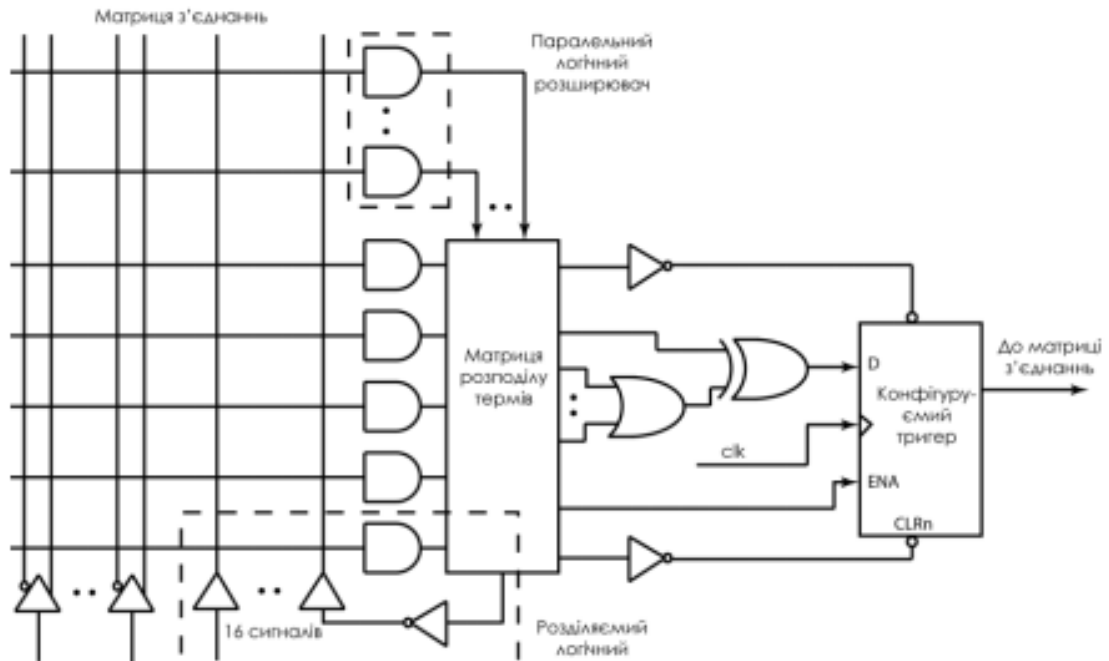


Рисунок 1.6 – Макрочарунка мікросхеми MAX 3000

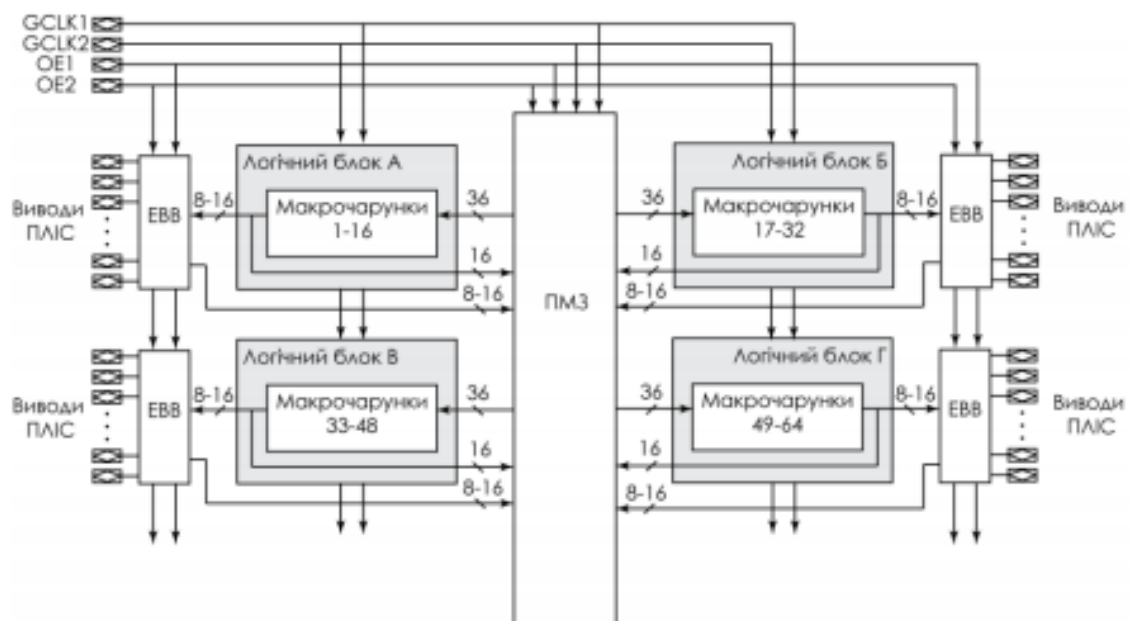


Рисунок 1.7 – Архітектура мікросхеми MAX 3000

Елементи вводу-виводу мають архітектуру, подібну до архітектури цих блоків в мікросхемах Cyclone III, які ми розглянемо далі [4].

### 1.3.2 Архітектура мікросхем сімейства Cyclone III

У таблиці 1.2 наведені основні параметри мікросхем сімейства Cyclone III.

Таблиця 1.2 – Параметри мікросхем сімейства Cyclone III

| Параметр                     | EP2C5   | EP2C70    |
|------------------------------|---------|-----------|
| Кількість логічних елементів | 4 608   | 68 416    |
| Кількість блоків пам'яті M4K | 26      | 250       |
| Загальний об'єм пам'яті, біт | 119 808 | 1 152 000 |
| Блоків ФАПЧ                  | 2       | 4         |
| Вбудованих помножувачів      | 13      | 150       |
| Виводів, дост. користувачу   | 158     | 622       |

Логічний блок (Logic Array Block – LAB) мікросхеми містить у собі шістнадцять логічних елементів, які з'єднані локальною матрицею з'єднань (Local interconnect), ланцюги сигналів керування логічним блоком, ланцюг переносу для логічних елементів блоку та ланцюг переносу з одного логічного блоку до іншого. На рисунку 1.8 показано два логічних блоки та з'єднання їх з глобальною матрицею з'єднань. Логічні блоки розташовані рядками та стовбчиками і з'єднуються між собою за допомогою глобальної матриці з'єднань (MultiTrack interconnect) .

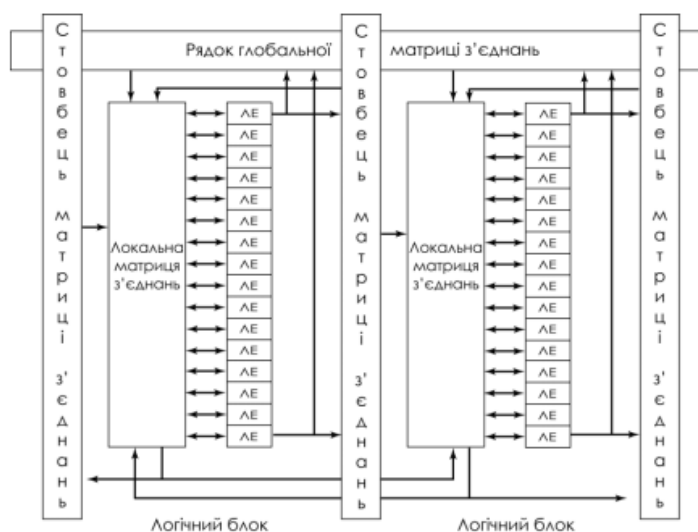


Рисунок 1.8 – Логічні блоки мікросхеми Cyclone III

Блоки фазового автопідстроювання частоти (ФАПЧ, Phaselocked loop, PLL) можуть використовуватись для множення та ділення частоти, зсув фази тактового сигналу відносно основної частоти та отримання програмованої шпаруватості тактового сигналу. Розглянемо основний принцип роботи системи ФАПЧ, структурна схема якої зображена на рисунку 1.9.



Рис. 1.9 – Структурна схема ФАПЧ в ПЛІС Cyclone III

Крім ліній програмування та живлення мікросхеми ПЛІС мають також виводи для підключення тактових сигналів. Сигнали з цих виводів підключаються до ліній глобального розповсюдження сигналу. Кількість спеціалізованих тактових сигналів залежить від типу корпусу мікросхеми і може дорівнювати 8 або 20. Архітектура мікросхем сімейства Cyclone III 30 сигналів по мікросхемі у порівнянні з проходженням сигналу через стандартні лінії зв'язку. Це дозволяє досягти майже одночасного спрацювання тригерів, які розташовані в різних місцях мікросхеми. Крім того джерелом для глобальних сигналів можуть слугувати також виводи подвійного використання DPCLK (Dual-Purpose Clock), кількість яких дорівнює 8 або 16, виходи вбудованих блоків ФАПЧ (PLL) та внутрішня логіка проекту. Підключення сигналів до ліній глобального розповсюдження може виконуватись або автоматично самим компілятором пакету Quartus II (опція Auto Global Clock в діалозі More Settings сторінки Fitter Settings меню Settings) або розробником за допомогою примітива Global як в графічному редакторі так і при використанні мови опису апаратури[4].

## 1.4 Вибір програмного засобу

Постійне ускладнення електронних пристроїв вимагає від розробників підвищення швидкості розробки, простого переходу між платформами проектування та наглядного опису проекту. Використання мов опису апаратури дозволяє забезпечити високорівневий опис складних систем, підтримку бібліотек проектування, сумісність коду, написаного для різних середовищ розробки. На сьогоднішній день найбільш часто використовуються дві мови опису апаратури: VHDL та Verilog HDL. Ці мови вважаються стандартними і підтримуються майже всіма засобами моделювання та синтезу пристроїв на ПЛІС. Крім цих мов існують менш поширені AHDL, Abel, SystemC, SystemVerilog, Verilog-ASM [18].

Ми будемо використовувати всі можливості середовища Quartus II та використовувати мови VHDL та Verilog HDL для різних цілей одного великого проекту, адже Quartus II – це інтегроване середовище для розробки проектів на ПЛІС фірми Altera.

### 1.4.1 Мова опису апаратури VHDL

Традиційне проектування зводиться до побудови логічних схем пристрою. На щастя в наш час вже не потрібно формувати складні логічні функції і вдаватися до метода пошуку їх мінімальних форм для побудови логічних схем. Досить скористатися одним з мов опису цифрових схем (VHDL або Verilog).

VHDL - мова опису інтегральних схем. Це незвичайний мову, як для програмістів, так і для розробників апаратури. Перш за все це пов'язано з тим, що це - мова паралельного програмування. Початківці програмісти часто зазнають труднощів при програмуванні на VHDL саме через не усвідомлення, що все написане вами виконується паралельно [17].

В ході програмування на VHDL вам доведеться створювати маленькі "мікросхеми" - складові частини майбутнього устрою. Вам раз по раз доведеться створювати сутності (entity) і архітектури кожної частини пристрою. По суті

описуються входи і виходи нашої мікросхеми всередині мікросхеми, архітектура ж реалізує його функції, виконуючи таким чином основне завдання мови VHDL.

Першим кроком до освоєння програмування ПЛІС на VHDL є установка САПР для розробки.

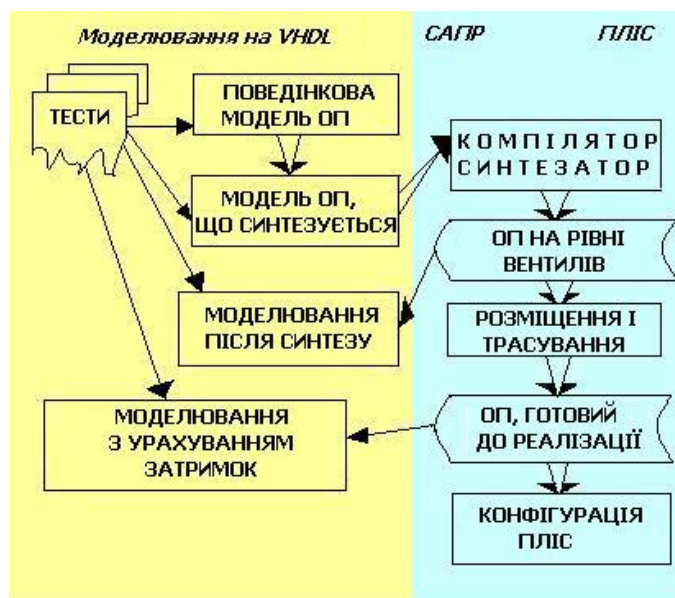


Рисунок 1.10 - VHDL як програмна система

VHDL (Very high speed integrated circuits Hardware Description Language) - дана мова призначений для опису проєктованих систем на схемотехнічному рівні проєктування і заміни класичного підходу до схемотехнічному проєктування на рівні окремих елементів. Мова дозволяє описувати цифрові системи на алгоритмічній рівні. За допомогою спеціального програмного забезпечення опис на мові VHDL перетворюється в схему на рівні найпростіших елементів цифрової електроніки. Опис на мові застосовуються VHDL як при проєктуванні замовних НВІС так і при проєктуванні цифрових систем на базі спеціальних пристроїв ПЛІС (Програмовані логічні інтегральні схеми) [17].

Інтерфейс - описує наше з вами пристрій, як чорний ящик з входами і виходами, тобто головна його задача показати які входи і виходи є у нашого пристрою для зв'язку із зовнішнім світом.

Архітектура - описує поведінку нашого пристрою або розкриває його внутрішню структуру, тобто в архітектурі описується алгоритм функціонування нашого пристрою.

Варто звернути увагу, що архітектура може бути описана в загальному випадку двома варіантами:

- поведінковим стилем (описується алгоритм роботи пристрою);
- структурним стилем (описується структура пристрою).

Поведінковий стиль зручно використовувати при описі елементів на низькому рівні ієрархії, а структурним на верхніх рівнях ієрархії, т.е. написати багато маленьких пристроїв поведінковим стилем, а потім описати що складаються з них пристрій структурним стилем, у вигляді зв'язків між ними див. рисунок 1.11.

Оператори мови - оператори в мові бувають послідовні, а бувають паралельні. Паралельні оператори вводяться для того, щоб відобразити паралельність протікають в залозі процесів. Але будь-який паралельний оператор можна замінити спеціальним паралельним оператором процесу з послідовними операторами всередині його, він як раз для цього і створений.

При допомозі паралельних операторів ми описуємо як би елементи схеми які можуть працювати одночасно, при моделюванні кожному паралельному оператору ставиться у відповідність свій процес. Далі я буду вважати кожен паралельний оператор процесом.

Сигнали - пов'язують між собою процеси. Вони є зовнішніми по відношенню до процесу, тобто процес може зчитувати сигнал і видавати значення в сигнал. По-цьому сигнали можуть оголошуватися тільки в області декларацій архітектури. Так само сигнали можуть зберігати значення, які необхідно передавати від процесу до процесу[17].

Змінні - змінні використовуються всередині паралельного оператора процесу і потрібні для опису алгоритму роботи процесу. Їх треба використовувати в тих випадках, коли не потрібно переносити інформацію від процесу до процесу.

Атрибути - це значення (характеристики), пов'язані з будь-якими об'єктами мови. Наприклад типами, сигналами, змінними і т.д.

Типи - множини значень з якими то загальними характеристиками. Характеризуються типи набором значень, які можуть приймати об'єкти (сигнал, змінні) даного типу, а так же набором операцій яке можуть виконуватися з об'єктами даного типу.

Бібліотеки та пакети - пакети становлять собою структури в яких зберігаються описи різних функцій, процедур, компонентів, типів, констант і т.д, все пакети збираються в бібліотеки, які в наслідку підключаються до проектів.

Проект на мові VHDL представляється сукупністю ієрархічно зв'язаних текстових файлів, які називаються проектними модулями. Типовий текст програми мовою VHDL має наступну структуру (див.рисунок 1.10): – декларація бібліотек; – інтерфейс об'єкта проекту; – архітектура об'єкта проекту.

Як показано на рисунку 1.11 на одному рівні описується інтерфейс об'єкта проекту та архітектура об'єкта проекту, плюс якщо потрібно до кожного інтерфейсу можна підключити потребується пакет. У свою чергу тіло архітектури складається з паралельних операторів, серед яких може бути оператор процесу, що містить послідовні оператори. Зверніть увагу, що сигнали декларуються на рівні архітектури, а змінні на рівні окремих процесів і використовуються всередині їх.

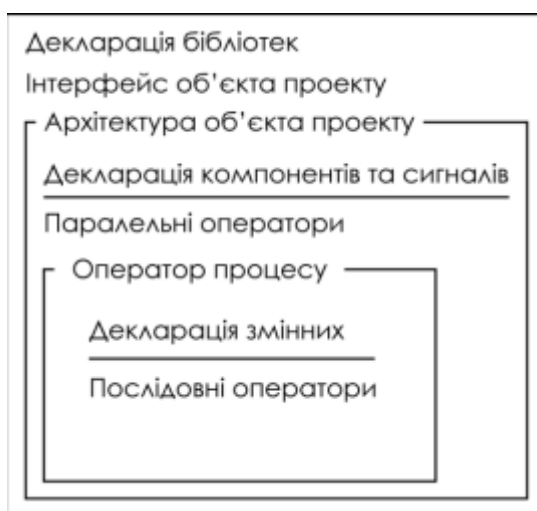


Рисунок 1.11 - Структура мови VHDL

VHDL є мовою з суворою типізацією. Це означає, що застосування різних типів даних і літералів в одній операції є помилкою. Засоби суворого контролю типів відіграють велику роль, оскільки дозволяють уточнювати наміри розробника. Мова використовується для представлення апаратних проектів у різноманітних варіантах, тому засоби контролю типізації даних мають тут особливо велике значення. Наприклад, вони дають розробнику можливість представити групу ліній (провідників) шини у вигляді масиву бітів або цілого числа [17].

#### **1.4.2 Мова опису апаратури Verilog HDL**

Verilog підтримує такі «стандартні» типи даних: ціле – integer (32-бітове зі знаком) та real – число з плаваючою точкою. Для моделювання також використовуються time (час), специфічний тип, що застосовується вбудованими функціями для моделювання часу, зазвичай 64-бітове ціле; event (подія) — у мові існує низка операторів та конструкцій для роботи з подіями. Створювати свої типи даних, як і VHDL, не можна. Під час розробки синтезованих моделей з перелічених типів використовується лише integer.

Сигнали, на відміну VHDL, бувають двох основних типів: «ланцюги» і «реєстри». Найпоширеніші їх описуються ключовими словами wire і reg відповідно. Проте слід, що засіб синтезу який завжди реалізує reg як тригера. Відмінність wire від reg у тому, що reg здатний зберігати присвоєне значення (працює як змінна у мовах програмування), а wire потрібно докладати безперервне вплив (driver). Тобто wire моделює провід, який перетворюється на невизначений стан при відключенні драйвера. Існують також wand, wor, tri0, tri1, triand, trior і trireg (це ланцюг, а не реєстр!) для моделювання різних типів ланцюгів (wand – wired and, tri0 – резистор до 0, trireg – ємність тощо) Але такі ланцюги зустрічаються рідко і тому в статті не розглядаються. Ідентифікатори Verilog є чутливими до реєстру написання і підкоряються звичайним правилам: не можуть починатися з цифри або знака \$ і можуть містити літери, цифри, «\$», «\_». Існують так звані escaped-ідентифікатори (в основному в структурних

моделях, отриманих після синтезу), які починаються з «\», містять будь-які символи та закінчуються пробілом або перекладом рядка [18].

На відміну від VHDL у Verilog уявлення сигналів реалізовано у мові (а чи не в бібліотеці). Усього існують чотири типи значень, які можуть приймати сигнали «ланцюг» та «регістр»: 0, 1, z, x. Перші три відповідають логічним рівням та стану з високим імпедансом. Четвертий (x) означає невизначений стан і використовується при моделюванні неініціалізованих сигналів, конфліктів (два виходи з протилежними станами з'єднані разом), метастабільних станів тригерів (при порушенні тимчасових співвідношень між входами даних та тактовим входом), іншими словами, у всіх випадках, коли симулятор не може визначити значення цього сигналу. У реальному приладі такого сигналу, звісно, немає [19].

Для запису багаторозрядних сигналів (констант) використовуються наступні конструкції: 1'bz - одnorozрядний високоімпедансний сигнал, 10'd1\_000 - десятирозрядне число 1000, записане в десятковій системі (символ "\_" ігнорується), 4'bх01z - чотирирозрядний сигнал з невизначеним високоімпедансним молодшим і другим і третім, рівними 1 і 0 відповідно. Тобто запис багаторозрядного сигналу являє собою розрядність, одинарну лапку "'" (не плутати з апострофом "'", що використовується в директивах), основу системи числення (b, o, d, h) та цифри, що використовуються в даній системі числення. У двійковій системі допустиме використання символів z та x. Символ підкреслення служить для полегшення запису та ігнорується. Використання констант без вказівки розрядності не бажано, оскільки за умовчанням константа сприймається довжиною 32 біта. Дані типу integer також можуть надаватися регістрам.

Основною структурною одиницею Verilog опису є module. Модуль відповідає entity в VHDL. Модуль описується ключовими словами module – endmodule. У файлі можна описати кілька модулів. Інші модулі можуть підключатися до ланцюгів модуля утворюючи ієрархічну структуру. При запуску Verilog симулятор будує ієрархічне дерево з усіх модулів, які виявлені у файлах, поданих на вхід симулятора, і знаходить модуль верхнього рівня. Якщо таких модулів кілька, відбувається помилка. Як правило, модуль містить список

портів - інтерфейсних сигналів, які служать для підключення його до інших модулів.

Порти бувають трьох типів:

- input – входи;
- output – виходи;
- inout – двонаправлені.

Входи та двонаправлені порти повинні мати тип wire, а виходи можуть бути як wire, так і reg [19].

### **1.4.3 Програмне забезпечення QUARTUS II(QUARTUS PRIME)**

Програмний пакет QuartusII(Quartus Prime) фірми Altera® є повною, набір інструментів, написаний середу проектування, легко адаптується до вимог конкретного проекту. Це комплексна середовище для розробки систем на програмованому кристалі (SOPC). Пакет Quartus II включає в себе всі утиліти, необхідні для роботи з мікросхемами FPGA і CPLD [3].

Наступна послідовність дій описує типовий порядок проектування в середовищі Quartus II за допомогою графічного інтерфейсу користувача:

- створення нового проекту, вказівка типу використовуваної мікросхеми або сімейства за допомогою команди New Project Wizard (меню File);
- створення вихідного файлу проекту на мовах Verilog HDL, VHDL, або Altera Hardware Description Language (AHDL) за допомогою текстового редактора (Text Editor). Крім того, можна створити блок-схему проекту в графічному редакторі (Block Editor) використовуючи символи, що представляють інші вихідні файли проекту або логічні елементи. За допомогою команди MegaWizard® Plug-In Manager (меню Tools) можна створювати різні варіанти мегафункцій і IP-ядер для включення їх в файл проекту;
- вказівка початкових налаштувань проекту за допомогою редактора призначень (Assignment Editor), діалогового вікна Settings (меню Assignments), редактора топології (Floorplan Editor), і / або застосовуючи фіксовані логічні блоки (LogicLock™);

- створення проекту на системному рівні за допомогою генератора систем на кристалі (SOPC Builder) або генератора систем ЦГЗ (DSP Builder);
- створення програмних файлів для процесорного ядра Nios® за допомогою редактора програмного забезпечення (Software Builder);
- синтез проекту за допомогою модуля аналізу і синтезу (Analysis & Synthesis);
- виконання функціонального моделювання проекту за допомогою симулятора (Simulator) і команди Generate Functional Simulation Netlist;
- виконання розміщення і трасування проекту за допомогою модуля трасування (Fitter);
- проведення попереднього аналізу споживаної потужності за допомогою програми PowerPlay Power Analyzer;
- проведення аналізу тимчасових затримок проекту за допомогою програми аналізатора тимчасових затримок (Timing Analyzer);
- виконання моделювання проекту з урахуванням тимчасових затримок за допомогою симулятора;
- поліпшення тимчасових характеристик проекту за допомогою повторного фізичного синтезу, використання фіксованих логічних блоків, налаштувань в діалоговому вікні Settings і в редакторі призначень;
- створення файлу для програмування мікросхеми за допомогою модуля асемблера (Assembler);
- програмування мікросхеми за допомогою утиліти вибору програм (Programmer) і обладнання Altera; або перетворення формату файлу для програмування;
- налагодження проекту за допомогою вбудованого логічного аналізатора (SignalTap® II Logic Analyzer), генератора контрольних точок (SignalProbe™).

Основною робочою одиницею в середовищі Quartus II є проект. Він створюється за допомогою спеціальної утиліти (New Project Wizard меню File).

При створенні нового проекту задається робоча директорія, призначається ім'я проекту і ім'я файлу верхнього рівня ієрархії. Додатково можна вказати

вихідні файли проекту, призначені для користувача бібліотеки, які використовуються САПР сторонніх фірм, вбрання сімейство мікросхем (або використовувати автоматичний вибір сімейства компілятором Quartus II) [3].

## **1.5 Постановка задачі**

Розробити SDR приймач на ПЛІС. Головне завдання ПЛІС - "захопити" цифровий сигнал з АЦП, перенести його в область низьких частот, відфільтрувати та відправити результат на комп'ютер. Для розробки схеми використовую мову опису апаратури VHDL, Verilog HDL і інтегроване середовище розробки програмно-апаратних засобів QUARTUS II (QUARTUS PRIME) .

## **2 РОЗРОБКА SDR ПРИЙМАЧА НА ПЛІС**

### **2.1 Software-defined radio**

Сам принцип побудови приймача, що складається з двох частин: невеликої схемної частини та програмного забезпечення. Ось на програмне забезпечення і покладено основне навантаження.

Насправді, комп'ютер моделює всі вузли приймача програмно. Причому робить це із найвищою якістю. Навіть живлення тепер можна взяти від USB-порту комп'ютера.

Отже, сигнал від антени подається через фільтр (можна і через ФНЧ) швидкодіючі ключі. Від цих ключів залежить багато, тому вони повинні бути хорошої якості (швидкодіючі, з малим перехідним та високим опором відключеного стану тощо). До виходів ключів підключено два операційні підсилювачі. До підсилювачів теж висуваються високі вимоги. Використовуючи

стандартну мікросхему синтезатора DDS (Direct Digital Synthesis – цифровий синтезатор прямого синтезу) та дільники з виходами сигналів, зрушеними на 90 градусів, можна перекрити той діапазон, який нам необхідний.

Розглянемо цифровий приймач SDR, з виходу операційних підсилювачів сигнал передається на лівий та правий канали аудіокарти. Все інше робить програма: обробляє сигнал, перебудовує приймач по діапазону (використовується керований DDS), моделює необхідні фільтри (причому з такими характеристиками, які в залізі не отримати), моделює роботу АРУ (набагато краще, ніж у «реальному» приймачі), має панорамний індикатор, що відображає спектрограму (на якому видно обстановку в ефірі), вимірює рівні сигналу та багато іншого. Тобто, використовуючи цифрову обробку сигналів, ми можемо реалізувати те, що хотілося б, а також регулювати в широкому діапазоні параметри.

Технологія SDR пропонує наявність панорами; при цьому оператору доступне візуальне спостереження в реальному часі на моніторі комп'ютера за сигналами в смузі частот, що визначається частотою дискретизації звукової карти (48, 96 або 192 кГц), а також покращувати функціональність SDR приймача шляхом удосконалення програмного забезпечення. Немає необхідності купувати щоразу новий апарат – достатньо лише завантажити та оновити програму.

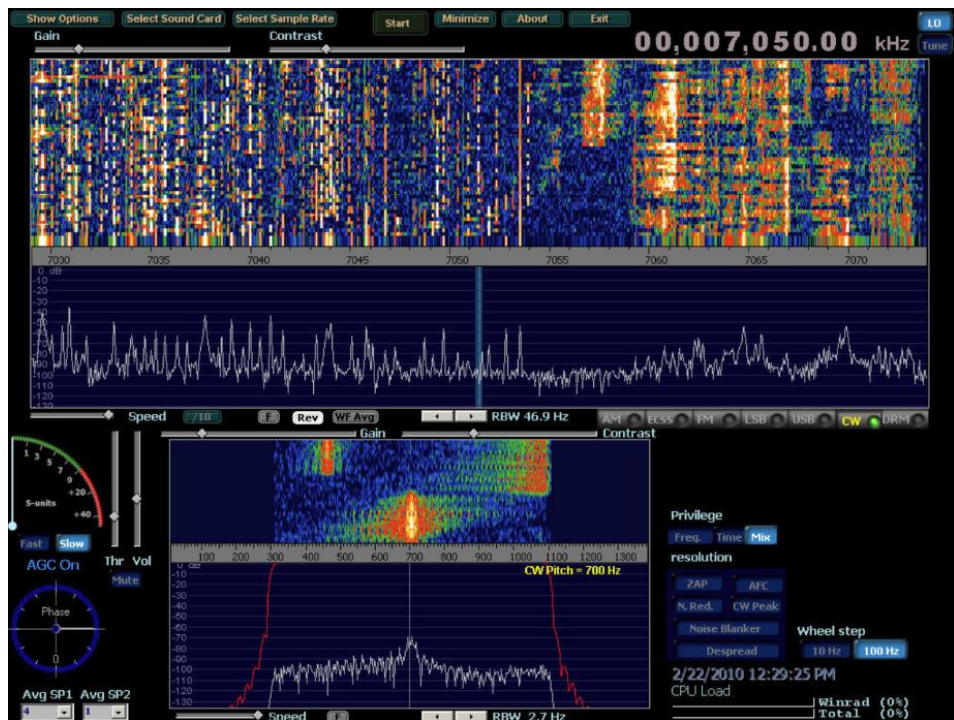


Рисунок 2.1 – Інтерфейс SDR приймача у програмі HDSDR

На рисунку 2.1 показано інтерфейс SDR приймача, поглянувши на рисунок можна собі уявити який потужний функціонал закладений в цій технології.

Крім моніторингу з дуже гарною наочністю, керуючі програми для SDR дозволяють здійснити високоякісний запис сигналу, що приймається, і ретранслювати його своєму кореспондентові, записати стан ефіру в смузі частот і згодом його відтворити, вибрати будь-яку смугу пропускання фільтра від 10Гц до 20кГц будь-які режими роботи.

Крім цього, SDR приймач - це ще й аналізатор спектру, що дозволяє оператору всебічно оцінювати сигнали радіостанцій, що приймаються. Особливі перспективи SDR техніка може дати тим, хто працює на діапазонах УКХ. Підключивши до SDR-трансівера УКХ-трансвертер, стає можливим спостерігати за наявністю та активністю радіостанцій у досить широкій смузі частот [10].

### 2.1.1 Види SDR

Існуючі SDR можна розділити на 3 види:

- вже застарілі моделі на базі звукової карти — оцифрування сигналу в них відбувається в ПК, а сигнал передається на лінійний вхід аудіокабелем.

Наразі вони давно зняті з виробництва, але іноді можуть з'явитися на барахолці. Брати за великим рахунком, ніякого сенсу, хіба що віддадуть задарма — ціна хорошої звукової карти перевищить ціну самого SDR;

- SDR, що мають вбудований АЦП та передають сигнали в ПК у цифровому форматі. Це більшість сучасних пристроїв середнього цінового діапазону. Вони побудовані за принципом гетеродинного прийому, тільки після перенесення частоти замість НЧ-блоку стоїть АЦП. Такі приймачі мають ширину смуги пропускання від 2 до 10 МГц, є різні моделі на різні частоти та діапазони (rtl sdr, SDRPlay, Airspy). Недолік будь-якого супергетеродинного приймача — наявність дзеркальних каналів прийому — оскільки фільтри неідеальні, станції приймаються там, де реально їх немає. Навіть якщо фільтри більш-менш непогані, сигнали потужних станцій все одно можуть пролазити і відтворюватися у вигляді перешкод;

- DDC (Direct Down Conversion) SDR. Це найсучасніша технологія на сьогоднішній день. Суть у тому, що гетеродин тут не потрібен - надшвидкий АЦП з частотою оцифрування порядку 100 млн семплів/с оцифровує безпосередньо вхідний сигнал з ефіру, що дозволяє мати прийом до частоти, що дорівнює половині частоти дискретизації, наприклад до 50 МГц. Бітовий потік на комп'ютер не передається, а обробляється в швидкодіючій ПЛІС на платі, і потрібна смуга (зазвичай до 6 МГц) передається в комп'ютер. Такий приймач не має дзеркальних каналів, і в ньому все гаразд.

Верхня межа частоти DDC-приймачів зазвичай обмежена 30-50 МГц, так як найбільш швидкодіючих АЦП у продажі або немає, або вони коштують космічних грошей (до речі, надшвидкі АЦП потрапляють в американські обмеження щодо постачання високотехнологічних електронних компонентів у країни третього світу) [11].

### **2.1.2 SDR приймачі супергетеродинного типу**

Алгоритм роботи такого типу приймача ґрунтується на перенесенні іспльзуюема сигналу на заздалегідь визначену проміжну частоту (ПЧ), де проводиться його обробка. Діапазон частот і селективні якості по сусідньому

каналу у приймачів, побудованих за даним типом є більш якісними, так як задаються при розрахунку схеми приймача. Найперші приймачі такого типу мали дискретизацію сигналу в каналі прийому 200Гц, параметр придушення з сусідніми каналами варіювалися в межах від -60дБ до -70дБ [10].

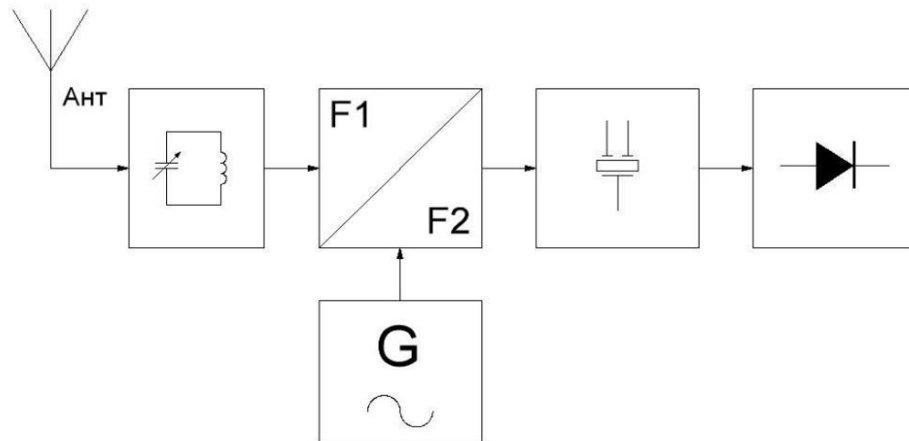


Рисунок 2.2 – Структурна схема SDR приймача супергетеродинного типу.

Але цей метод обробки сигналу має недоліки:

- віддзеркалення каналів прийому;
- максимальний діапазон частот.

Поява дзеркальних каналів пов'язано як з алгоритмом роботи прийом-ника, так і з похибкою роботи елементів приймального тракту, без яких обійтися не можна в такому типі приймачів . Кількість зеркальних каналів прийому може бути досить великим і залежить від кількості проміжних частот. Попередня обробка сигналу з антени при такій архітектурі побудови радіоприймача має основне завдання - придушення зеркальних каналів прийому.

З ростом інтеграції мікросхем з'явилися цифрові сигнальні процесори (DSP), їх почали впроваджувати у вхідній вузол обробки проміжної частоти. Це нововведення дозволило поліпшити селективність приємні- ков, а саме смугу фільтра від 50 Гц і параметр придушення з сусідніми каналами став доходити до -100дБ. У таких приймачах з'явилася можливість додати безліч нових функцій, такі як: функції з видалення шумів з каналу і перешкод, також з'явилася можливість додати додаткові демодулятори [10].

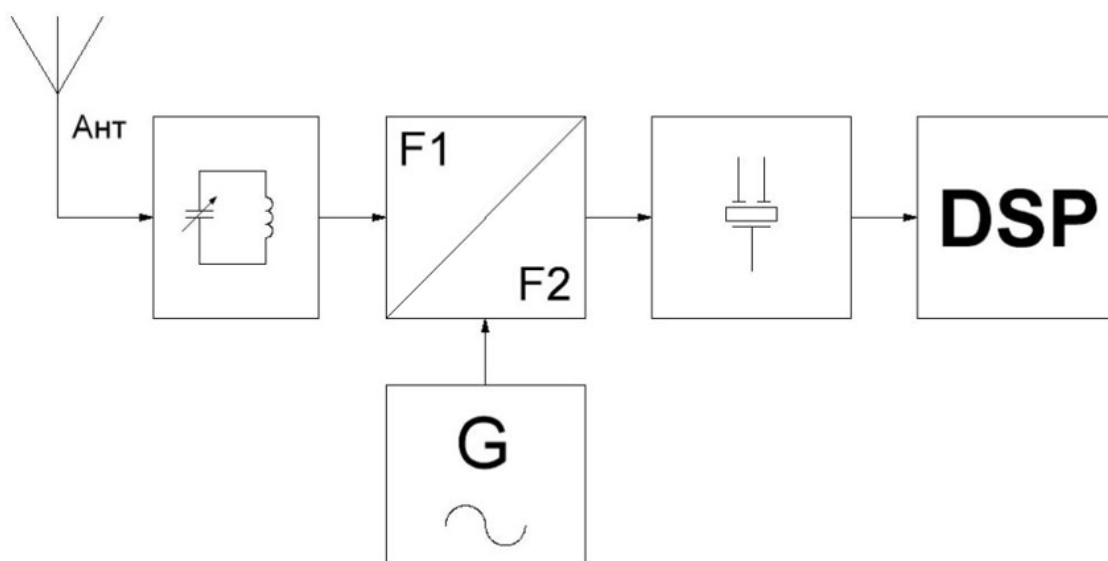


Рисунок 2.3 – Структурна схема SDR приймача супергетеродинного типу з використанням DSP в тракті обробки ПЧ.

Впроваджуючи в один пристрій кілька вхідних трактів і трактів обрамлення сигналів, виробники навчилися реалізовувати популярну в сьогодняшній день функцію, як відображення панорами спектра на робочому діапазоні. Лідером у використанні цієї технології була компанія ICOM.

Однак, коли із застосуванням DSP максимально покращилася селекція по сусідньому каналу прийому, але все одно існувала проблема з дзеркальним каналом. Також поряд з проблемою дзеркальних каналів з'явилася інша проблема: сигнали з антен передавачів стали досягати рівнів декількох сотень мілівольт, що почало викликати перевантаження вхідного тракту приймача.

В даний час в приймачах в яких використовується перенос НЧ сигналу з антени на проміжну частоту, параметр вхідних каскадів, переважних дзеркальні канали, має значення від -60дБ до -70дБ. Цього значення досить для того, щоб працювати в сучасному перевантаженому радіоефірі. Також на сьогоднішній день в таких приймачах існують такі проблеми:

- максимальні рівень сигналу, який можуть обробити прийомний вузол пристрою і вузол виконує цифрову обробку. Значення амплітуди сигналів

з радіоефіру можуть досягати рівнів декількох сотень мілівольт, якщо таке відбувається, то можливе блокування прийому сигналів пристроєм взагалі;

— діапазон амплітуд сигналів, які подаються на вхід приймача, які дозволяють тракту прийому сигналів працювати і не перевжась. Характерні параметри динамічного діапазону для приємопере- датчиків в даний час складають від 80дБ до 100дБ, що дозволяє ефективно працювати в радіоефірі в певному частотному діапазоні, навіть якщо в радіусі користувача буде передає радіоапаратура з потужністю 100 Вт.

Чимало важливим недоліком даної архітектури є шуми на виході радіоприймача, створювані самим радіоприймачем за умови, що в ньому присутній один або кілька вузлів прийому сигналу. Поява цих шумів обумовлено використанням аналогових елементів, таких як полупровод- ники. У підсумку, чим більше перетворюють елементів в приймальному тракті, тим більше може виникнути шумів на виході. Також можливо застосувати автоматичне регулювання підсилювачів тракту, що трохи вплине на загальний шум тракту, тому що кількість елементів посилення і перетворення не через менітся, а іноді і збільшується.

Ця проблема наочно проявляється у вигляді дратівливого шуму в навушниках або колонках, підключених до радіоприймача або інтегрованих в нього, навіть з не приєднаної антеною. При підключенні антени цей шум може бути не чути через шумів радіоефіру, але в такому випадку прослуховується радіоефір не матиме гарної якості. На прикладі рисунку 2.4 показана функціональна схема прийомо-передаючого пристрою, в якому підвищують перетворювачі частоти (ПЧ) в передавальному тракті, і знижують ПЧ в приймальному тракті. Прийомний тракт побудований за супергетеродина схемою з одноразовим перетворенням частоти[11].



Рисунок 2.4 - Функціональна схема SDR приймача супергетеродинного типу

Сигнал з антени через діплексер і смуговий фільтр (СФ1) подається на малошумний підсилювач високої частоти (ПВЧ), параметри якого в основному визначають співвідношення сигналу до шуму на виході приймача. З виходу ПВЧ сигнал надходить на один з входів змішувача (СМ1). На другий вхід СМ1 подається генерований сигнал певної частоти, що формується блоком синтезаторів частоти. Через фільтр проміжної частоти (ФПЧ) і посилювачі проміжної частоти (ППЧ) сигнал різницевої частоти, форміруємий на виході СМ1, надходить на вхід АЦП. Коефіцієнт посилення УПЧ виділяється напругою або кодовою комбінацією, ця характеристика залежить від типу використовуваної інтегральної мікросхеми. На виході управління системою автоматичного регулювання посилення (АРП), який формується на виході цифрового сигнального процесора (ЦСП). Вихідний сигнал ППЧ через аналого-цифровий перетворювач АЦП1 вводиться в ЦСП, де і здійснюється основна обробка сигналу: додаткова фільтрація, демодуляція, посилення сигналу і управління частотою синтезатора. Підсилювач, АЦП1 і ЦСП утворюють вузол АРУ, необхідний для стабілізації амплітуди прийнятих сигналів на виході вузла проміжної частоти, що дозволяє використовувати всю шкалу АЦП і забезпечити максимальне співвідношення сигналу до шуму. Демодульований в ЦСП сигнал

через цифро-аналоговий перетворювач ЦАП1 надходить на вихід приймача. Такий приймач виконує цифрову обробку сигналів у вузлі проміжної частоти. Модифікацією приймача є схема з не перестроюваним входним фільтром, смуга пропускання якого охоплює весь діапазон частот, що приймаються.

У передавальній частині приймача модульований сигнал переноситься в область високих частот за допомогою другого змішувача сигналів (СМ2), а ЦСП виконує функції формування модульованого сигналу, керування частотою несучого коливання і контролю вихідної потужності передавача. Модуючий сигнал через АЦП2 вводиться в ЦСП, в якому здійснюється модуляція сигналу на низькій частоті. Модульований сигнал через ЦАП2 і ФНЧ надходить на перший вхід см2, а на другий його вхід подається сигнал несучої частоти передавача. Смуговий фільтр (СФ) виділяє потрібну бічну смугу частот і ВЧ сигнал через підсилювач потужності (УМ) і дуплексер передається в антену. Контроль випромінюваної потужності ведеться за допомогою детектора потужності, що формує на виході напруга, пропорційне середньо-квадратичним значенням потужності на виході УМ [10].

В архітектурі приймачів супергетеродинного типу перетворення при- змінюється подвійне перетворення, іноді з потрібне або більше, для цього в тракці прийому додаються додаткові змішувачі, фільтр і підсилювач про- проміжній частоти. Прикладом радіоприймача, побудованого на кшталт супер- гетеродинного перетворення є більшість RTL-SDR приймачів. Їх дуже часто ви- користовують радіоаматори через низьку вартості самого пристрою і великої частоти дискретизації, яка досягається за допомогою зміни параметрів ФНЧ фільтрів. Але ці приймачі не позбавлені головних недоліків приймачів супергетеродинного типу:

- віддзеркалення каналів;
- необхідність наявності якісно налаштованих гетеродинів;
- необхідність наявності якісних фільтрів ФНЧ для ПЧ.

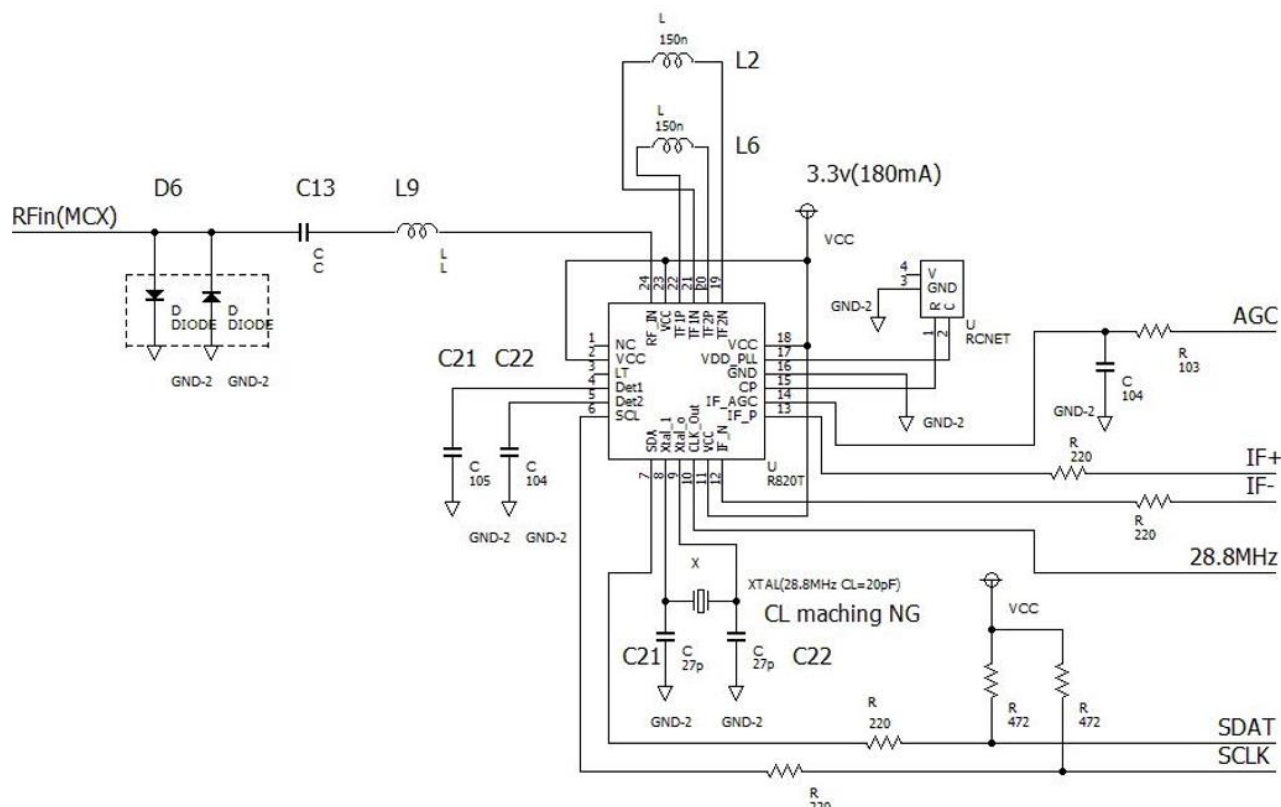


Рисунок 2.5 - Схема пристрою RTL2832U + R820T.

### 2.1.3 SDR приймачі прямого перетворення

Подолати більшість недоліків супергетеродинної архітектури приймачів дозволила архітектура прямого перетворення сигналів з радіоефіру в спектр звукових частот і перетворення отриманого сигналу фазовим способом. З вхідного вузла прийому сигналу забирається один з гетеродинов, і таким чином за допомогою всього лише одного умножителя і гетеродина сигнал переноситься на більш низьку проміжну частоту приблизно до 100 кГц і аналого-цифрова обробка сигналу відбувається засобами звукової карти комп'ютера. Подальша обробка сигналу виконується за допомогою програмного забезпечення комп'ютера, а саме демодуляція для прослуховування радіоефіру [10].

В архітектурі таких приймачів застосовується фазовий метод отримання сигналів на проміжну частоту. Для виконання методу застосовуються два ідентичних вузла прийому, зсунутих по фазі на 90 градусів, через це зеркальний

канал перебуває на 180 градусах щодо прямого сигналу, що дозволяє легко придушити його за допомогою програмних методів, рівень придушення становить від -100 дБ до -140 дБ. Селективність такого приймача досить висока, придушення сусіднього каналу становить від -100 дБ до -120 дБ.

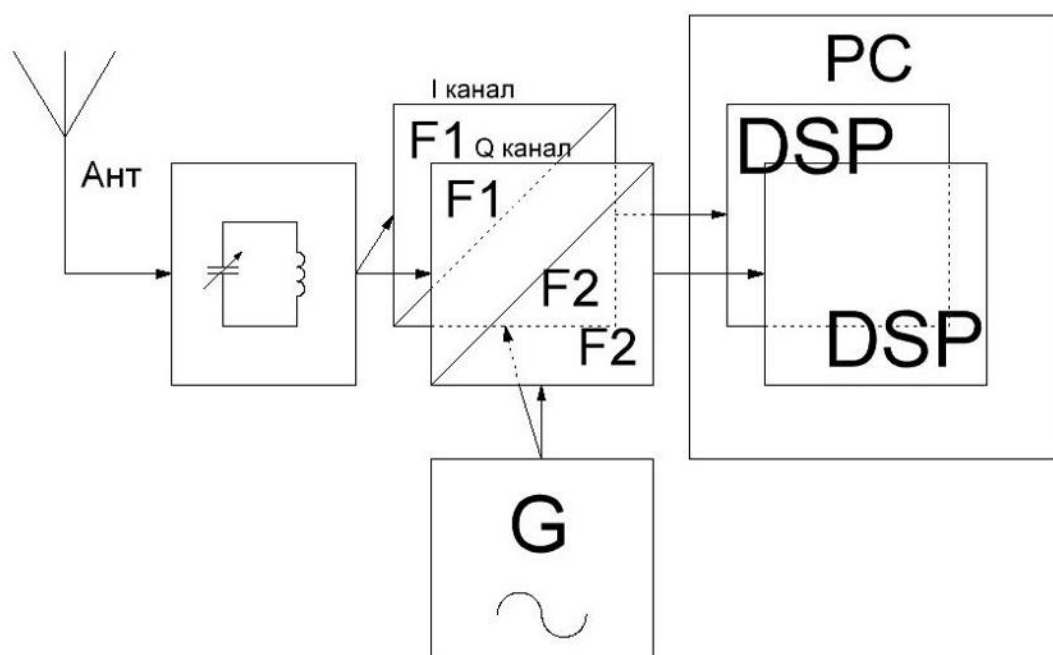


Рисунок 2.6 - Структурна схема SDR приймачів прямого перетворення.

Якщо у SDR приймачі прямого перетворення реалізований понижуючий програмний фільтр на частоту дискретизації 50-200Гц, то шуми стають практично не чути. Підвищується «прозорість» радіоефіру при використанні всього одного змішувача. За допомогою такої частоти дискретизації можна приймати навіть найслабші сигнали. Роботи з сигналом в комплексному вигляді дозволяє відображати весь спектр сигналу на екрані приймача, щоб можна було відразу знайти частоту на якій працює станція [11].

SDR технологія прямого перетворення має багато позитивних аспектів, але недоліки в ній все ж є. Хоч SDR приймач і вирішує безліч проблем класичних радіоприймачів, але робить це не до кінця. В першу чергу концепція перетворення сигналу на більш низьку частоту нікуди не пропала, тому в каналі прийому можуть виникати побічні канали. Вирішення цього завдання

покладається на смугові фільтри на вхідному каскаді прийому. Якщо використовувати діапазонні смугові фільтри (ДПФ) і фільтри низької частоти (ФНЧ) то можна домогтися придушення побічних каналів від -70дБ до -90дБ. Але практично використовуються ФНЧ і ДПФ 4-6 порядку з придушенням від -40дБ до -70дБ.

Найбільш складним завданням в SDR приймачах прямого перетворення є завдання по створенню ідентичності I і Q каналів перетворення сигналів з радіоефіру з певною шириною смуги і амплітудним діапазоном. Якщо в каналах перетворення виникає розбіжність фази або амплітуди хоча б на 2-4 градуси, то якість придушення побічних каналів падає від 20дБ до 40дБ. За більш ніж 15 річний розвиток SDR приймачів прямого перетворення такі розбіжності вдалося скорегувати програмним чином, що реалізовано в більшості програмних продуктів SDR.

У зв'язку з розвитком алгоритмів ЦОС і мікроелектроніки SDR приймачі з прямим перетворенням з гетеродином в архітектурі йдуть на другий план і займають нішу бюджетних приймачів / передавачів [10].

#### **2.1.4 Принцип архітектури DDC \ DUC**

Технологія SDR прямого перетворення з допомогою аналогових рішень стала основою для створення більш нової і просунутої технології. Таким чином, якщо застосовувати пряму оцифрування сигналу з радіоефіру і повністю цифрову обробку сигналу, то може вийти не тільки прослухати радіоефіру, а й побачити його на екрані в реальному часі.

SDR приймачі прямого перетворення, які виконують пряме оцифрування сигналу почали називати DDC від Digital Down-Converter. Алгоритм, виконуваних перетворення з цифри в аналог, називається DUC від Digital Up-Converter. Завдання з оцифрування сигналу майже повністю лягає на програмне забезпечення. Варто зазначити що DDC / DUC є окремою архітектурою SDR приймачів, що використовують програмні методи отримання сигналів на

проміжну частоту з мінімальною кількістю аналогових елементів пристрою, як показано на рисунку 2.7 [16].

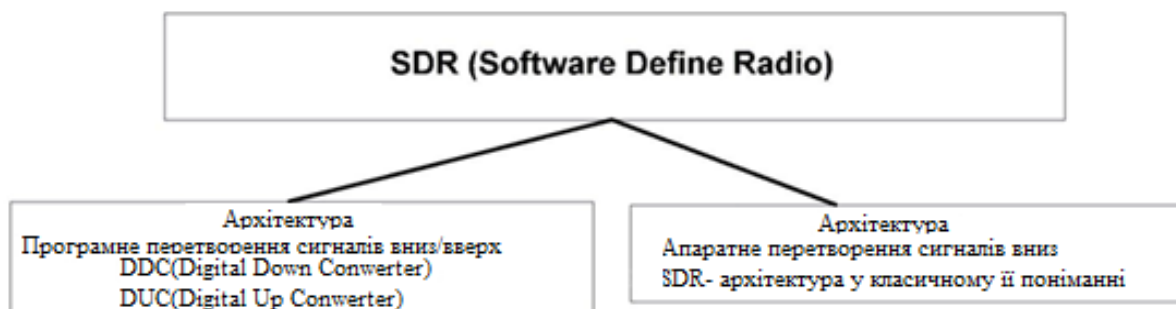


Рисунок 2.7 - Архітектури SDR приймачів.

Метод DDC SDR приймача у багато схожий з SDR прямого перетворення, але має відмінності. Основна відмінність DDC SDR від класичного SDR: Перенос сигналу на проміжну частоту в класичному SDR здійснюється за допомогою аналогового змішувача і гетеродина, після отримання сигналів на проміжну знижену частоту сигнал оцифровується за допомогою аудіо карти або АЦП і його можна прослухати. Якщо необхідно передати сигнал, всі операції виконуються в зворотну сторону. Селективність виявляється залежною від якості аудіо карти і може становити 48кГц, 96кГц або 192кГц. У рідкісних випадках модуль АЦП може знаходитися на платі приймача, що значного підвищує якість прийому сигналів. Є недолік у таких приймачів - це драйвер звукового пристрою Windows, робота по демодуляції сигналів, видалення шумів і відображення спектру виконується на персональному комп'ютері. У підсумку через недоліки аналого виття частини витрачається безліч часу і ресурсів для перетворення сигналу, як на передачу, так і на прийом, не дивлячись на уявну простоту схемного рішення класичного методу SDR, розробка не так проста.

За методом DDC SDR вхідний сигнал відразу оцифровується в діапазоні частот від 0 Гц до частоти дискретизації, для якої спроектована АЦП. На сьогоднішній день модулі АЦП здатні працювати при частоті дискретизації до 8 ГГц і більше, але вартість таких модулів досить висока. Існують дешеві модулі

АЦП з частотою дискретизації до 100 МГц, які цілком справляються з оцифруванням КВ діапазону [16].

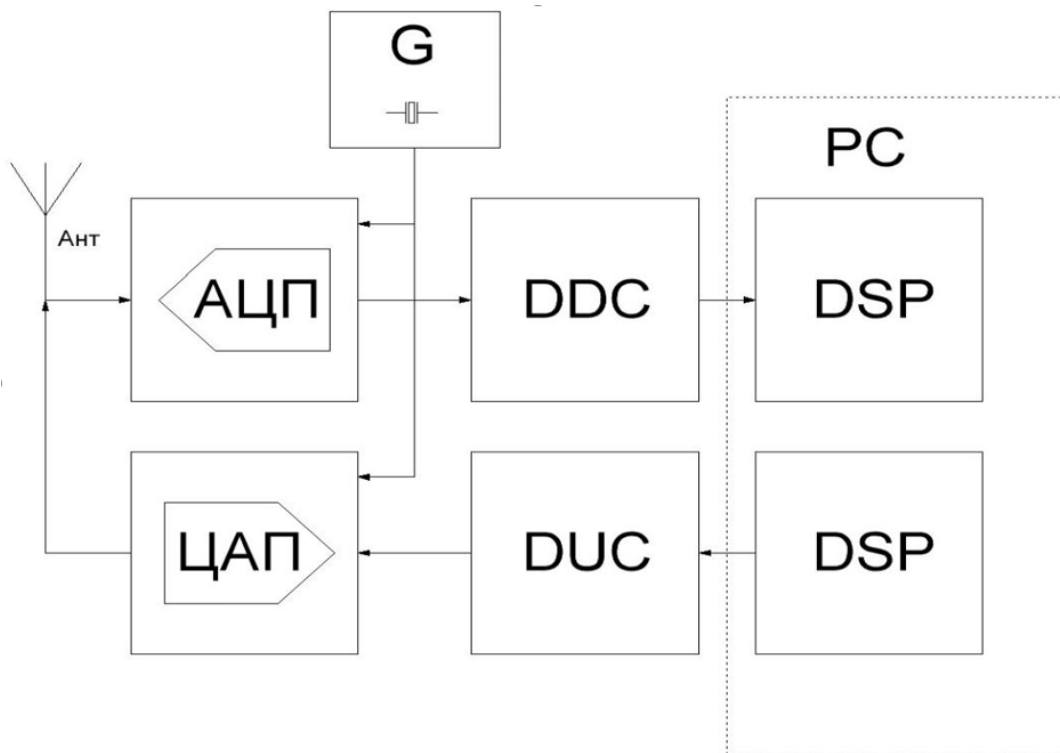


Рисунок 2.8 - Структурна схема SDR приймача / передавача прямого перетворення архітектури DDC / DUC

Після цифро-аналогового перетворення весь частотний діапазон від 0 Гц до 100 МГц (на прикладі АЦП з частотою дискретизації 100 МГц) у вигляді масиву цифрових даних может передаватися на комп'ютер, але цього частіше за все непотрібно, тому після АЦП дані найчастіше передають на високопродуктивну мікросхему ПЛІС, в завдання якої входить обробка такого більшого масиву даних. Саме ПЛІС виконує алгоритм DDC / DUC, реалізований в програмному вигляді, в якому виконується зниження або підвищення проміжної частоти. Під час виконання програми, ПЛІС може ви- виконуватися виділення потрібної смуги спектра і виконувати операції по передачі цих смуг на ПК групами, і з заданою швидкістю. На комп'ютері можуть виконуватися додатковий алгоритми по цифровій обробці сигналу і також демодуляція. При виконанні практичних

завдань зі спектром рідко потрібно виводити на екран весь діапазон частот від 0 Гц до 100 МГц. Найчастіше вимагають смуги шириною від 10 кГц до 50 кГц для виконання операції амплітудної модуляції (AM) сигналів [16].

При роботі з таким SDR приймачем можливо реалізувати передачу не-скільких потоків оброблених сигналів з ПЛІС меншого розміру. Таким про-разом можна відобразити на екрані комп'ютера два, три або більше одночасно працюючих каналу прийому - цей спосіб дозволяє створювати кілька «віртуальних приймачів» по всій смузі прийнятого сигналу.

Ключовим параметром в основі DDC SDR приймача є розрядність мікросхеми АЦП. На сьогоднішній день існують АЦП з частотою дискретизації в кілька гігагерц, але розрядність у більшості не перевищує 18 біт, навіть у найдорожчих моделей, доступних на ринку.

Для виконання завдання виведення на екран використовується алгоритм швидкого перетворення Фур'є (ШПФ). Для розрахунку задається максимальна кількість точок, найчастіше це значення береться з запасом - 4096 або 8192, дуже рідко більшу кількість. Загальна кількість використовуваних точок впливає на деталізацію сигналу при його розгляді. Чим більше задіяно точок, тим більше потрібно часу для розрахунку і отрисовка спектра. При виконанні цього алгоритму витрачається час менше, ніж якби весь потік інформації з АЦП надходив на ПК для розрахунку, тому і застосовується ПЛІС для виділення потрібних каналів в даний момент користувачеві, також алгоритм DDC виділяє потрібну кількість точок для каналів. За таким принципом виконується виведення сигналу на екран [16].

На рисунку 2.9 представлена структурна схема SDR приймача прямого перетворення. В даній схемі є вхідний фільтр (ВхФ) і підсилювач високої частоти (УВЧ), для реалізації такої схеми потрібно швидкий аналого-цифровий перетворювач (АЦП), оцифрований сигнал з антени якого вводиться в сигнальний процесор (ЦСП).

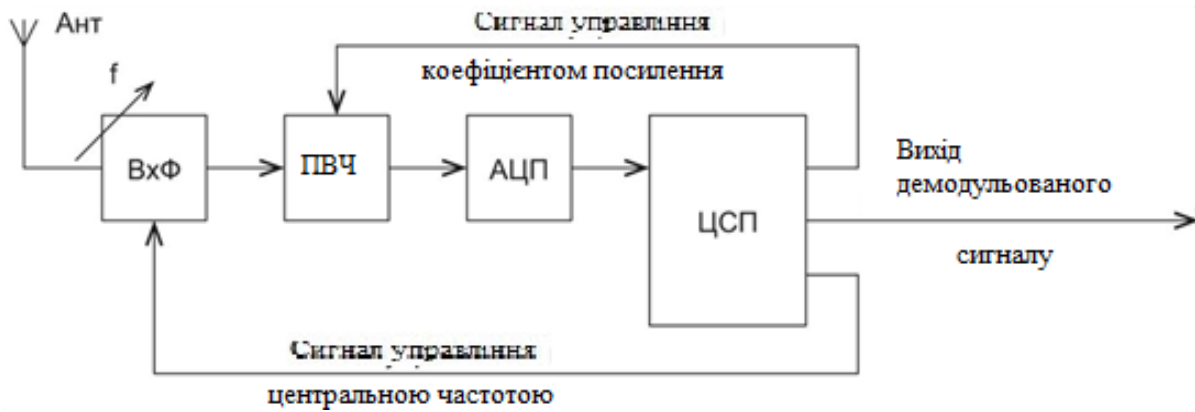


Рисунок 2.9 - Функціональна схема SDR приймача прямого перетворення.

Компанія Analog Devices випускає компоненти, що дозволяють реалізувати роботу SDR приймача на дискретних компонентах, такі як інтегральні мікросхеми (ІМС), що містять як аналогові, так і цифрові компоненти для реалізації таких систем [16].

Поєднання цифрової обробки і аналогової високочастотної схеми в одній ІМС приймача дозволяє значно скоротити витрати на необхідну обладнання і спростити реалізацію пристрою SDR. Прикладом такої ІМС є ІМС ADF7030, структурна схема наведена на рисунку 2.10.

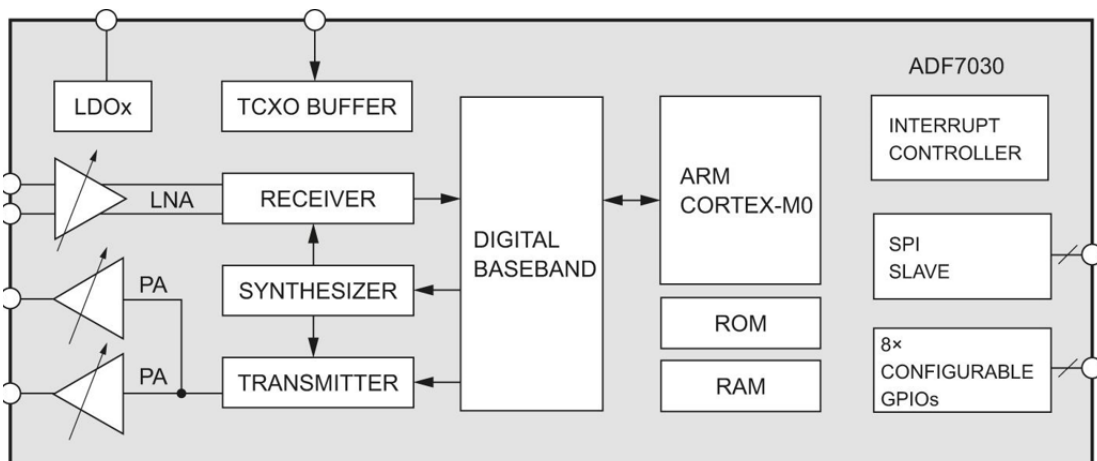


Рисунок 2.10 - Структурна схема ADF7030.

При реалізації DDC SDR приймача не обов'язково потрібні спеціальні ІМС для реалізації, досить мати швидкодіючу АЦП і ПЛІС, структурна схема такого пристрою представлена на рисунку 2.11.

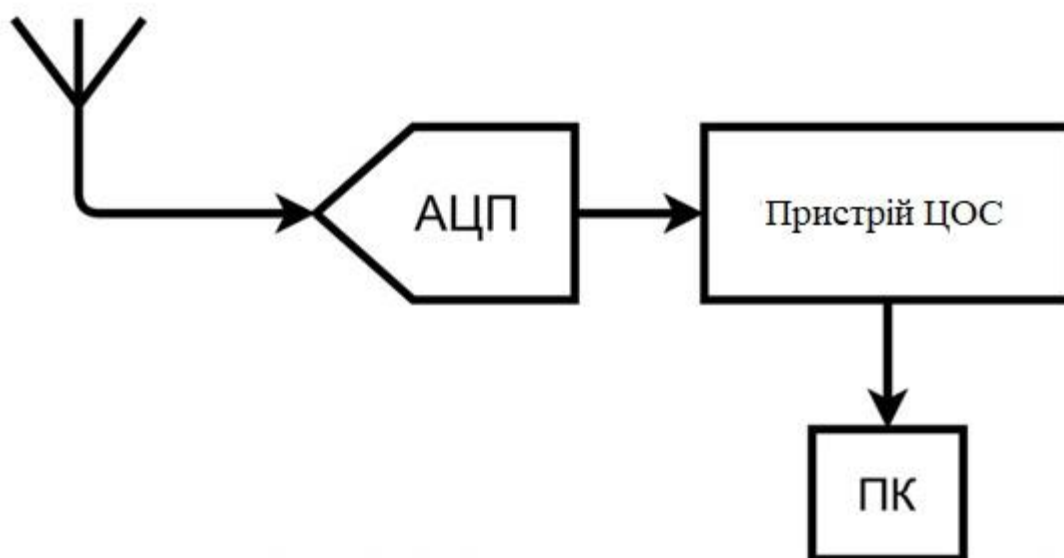


Рисунок 2.11 - Структурна схема DDC SDR приймача.

Після дослідження існуючих пристроїв, і типів реалізації SDR приймача, на перший план виходить DDC SDR архітектура приймача, як найбільш сучасний метод реалізації, хоча і супергетеродині архітектура все ще актуальна в наш час, але вона все одно вимагає глибоких знань в конструюванні фільтрів високих порядків і тонкій настройці генераторів, в тому числі великих фінансових вкладень [16].

## 2.2. Аналого-цифрові перетворювачі

АЦП застосовуються для широкого класу завдань радіотехніки – у будь-яких пристроях, де потрібно перетворити аналоговий сигнал на цифрову форму. Вони можуть використовуватися в аудіо- та відео-системах, застосовуються в задачах гідроакустики та гідрозв'язку (низька частота дискретизації, висока розрядність даних), в пристроях перетворення радіочастотного діапазону (схеми DDC/DUC), радіолокації та радіонавігації (високі частоти дискретизації, середня

розрядність даних). АЦП застосовуються в цифрових вольтметрах і мультиметрах, платах відеовходу комп'ютерів, відеокамерах, системах розпізнавання мови і звукових пристроях. Швидкі АЦП використовуються в осцилографах та спектроаналізаторах, застосовуються в лабораторному обладнанні та медичній техніці. Дуже часто АЦП використовується у завданнях збору та обробки даних на високих швидкостях[9].

Існує безліч типів АЦП, однак у рамках цієї статті ми обмежимося розглядом лише наступних типів:

- ацп паралельного перетворення (прямого перетворення, flash ADC);
- ацп послідовного наближення (SAR ADC);
- дельта-сигма АЦП (АЦП з балансуванням заряду).

Існують також інші типи АЦП, у тому числі конвеєрні і комбіновані типи, що складаються з декількох АЦП з (загалом) різною архітектурою. Однак наведені вище архітектури АЦП є найбільш показовими через те, що кожна архітектура займає певну нішу в загальному діапазоні швидкість-розрядність.

Найбільшою швидкістю і найнижчою розрядністю мають АЦП прямого (паралельного) перетворення. Наприклад, АЦП паралельного перетворення TLC5540 фірми Texas Instruments має швидкодію 40MSPS при розрядності всього 8 біт. АЦП цього типу можуть мати швидкість перетворення до 1 GSPS. Тут можна відзначити, що ще більшу швидкодію мають конвеєрні АЦП (pipelined ADC), проте вони є комбінацією декількох АЦП з меншою швидкістю і їх розгляд виходить за рамки цієї статті [8].

Середню нішу серед розрядність-швидкість займають АЦП послідовного наближення. Типовим значенням є розрядність 12-18 біт при частоті перетворення 100KSPS-1MSPS.

Найбільшої точності досягають сигма-дельта АЦП, що мають розрядність до 24 біт включно та швидкість від одиниць SPS до одиниць KSPS.

Ще одним типом АЦП, що знаходив застосування в недавньому минулому, є інтегруючий АЦП. Інтегруючі АЦП нині майже повністю витіснені іншими типами АЦП, але можуть зустрітися у старих вимірювальних приладах.

Як бачимо, мікросхеми АЦП – невід’ємна частина аналого-цифрових пристроїв. У сучасному світі дуже непросто знайти пристрій, в якому були б відсутні АЦП. Класичний вид АЦП представлений на рисунку 2.12 [8].

Ця мікросхема она містить два диференціальних аналогових входу VIN<sub>A</sub> і VIN<sub>B</sub>, вхід тактової частоти CLK, по інтерфейсу SPI, вихідну шину в управлінні паралельної форми D0-D13, точки напруги живлення різних вузлів мікросхеми, блок синхронізації SYNC та інші керуючі сигнали.

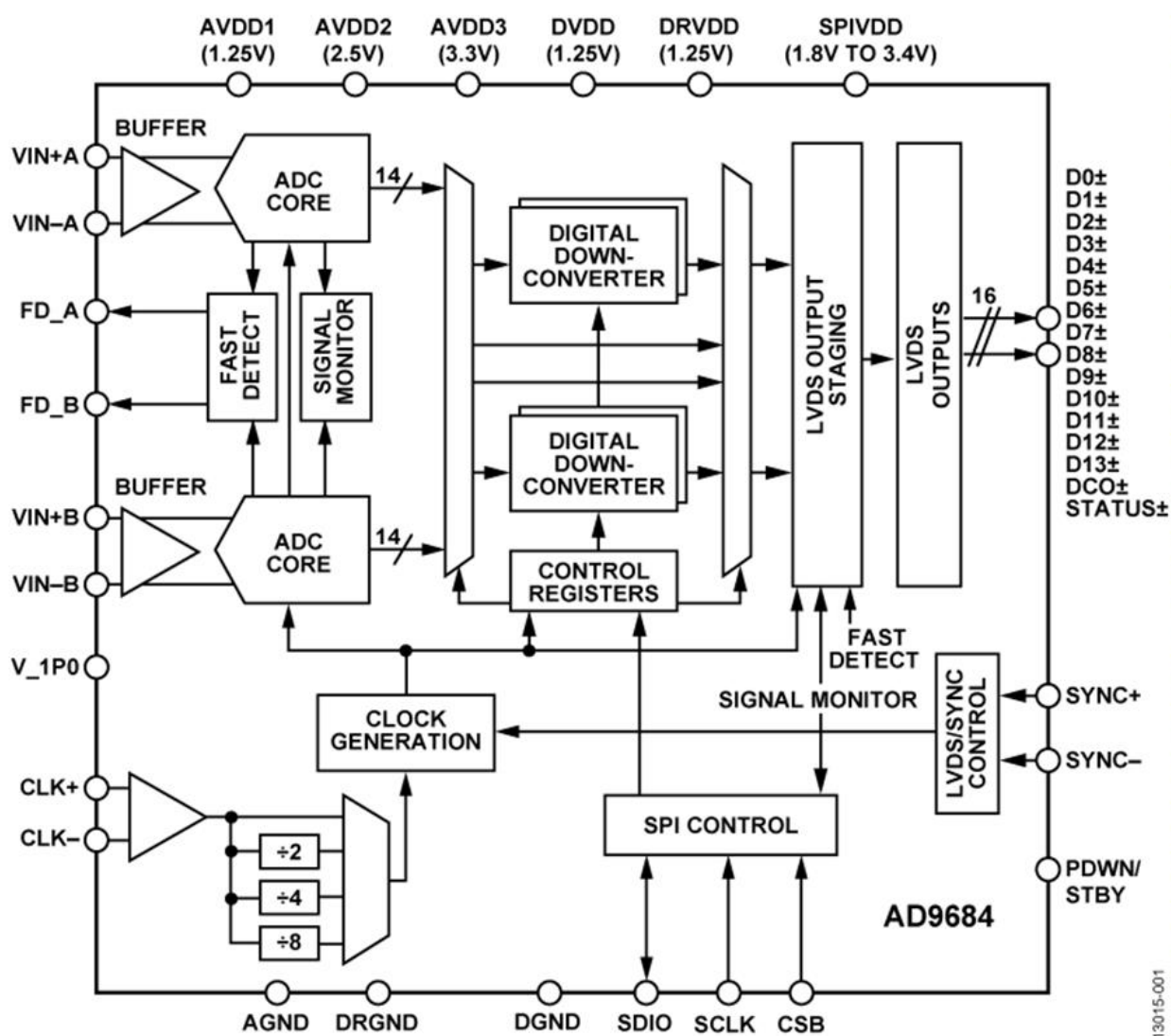


Рисунок 2.12 - Спрощена схема мікросхеми AD9684

Мікросхеми АЦП характеризуються набором визначених, від яких залежить кінцевий вибір параметрів користувача та який визначає область застосування АЦП. По цифровій частині – це швидкість передачі даних

(ефективна тактова частота), яка може знаходитися в діапазоні від кількох десятків кГц до кількох ГГц. Розрядність даних на виході АЦП – в сучасних пристроях від 8 до 32 біт даних. Состав аналогових характеристик – це співвідношення сигнал-шум (SNR) і динамічний діапазон, вільний від паразитних компонентів (SFDR), виражений, як правило, в децибелах (типові рівні SNR лежать в межах 70-80 дБ, а рівні SFDR в районі 90). дБс). Неважливими параметрами при виборі АЦП є – розсіювана потужність (Вт), кількість каналів в одному корпусі, інтерфейс обміну з цифровим узлом і, зрозуміло, вартість компонента [7].

Таблиця 2.1 – Сучасні швидкодіючі АЦП

| Ім'я        | Розрядність | Швидкість | Канали | SNR(dBFS) | SFDR(dBc) | Інтерфейс     | Ціна(\$) |
|-------------|-------------|-----------|--------|-----------|-----------|---------------|----------|
| AD9684      | 14          | 500       | 2      | 69.2      | 82        | LVDS Parallel | 272      |
| AD9652      | 16          | 310       | 2      | 75.4      | 94        | LVDS Parallel | 231      |
| AD9643      | 14          | 250       | 2      | 72        | 90        | LVDS Parallel | 132      |
| AD9467      | 16          | 250       | 1      | 76.4      | 96        | LVDS Parallel | 120      |
| AD9267      | 16          | 640       | 2      | 83        | 88        | Serial        | 41       |
| AD9239-250  | 12          | 250       | 4      | 65        | 77        | Serial        | 170      |
| AD9625-2500 | 12          | 2500      | 1      | 58.3      | 77        | JESD204B      | 735      |
| AD9691      | 14          | 1250      | 2      | 63.4      | 77        | JESD204B      | 693      |
| AD9680      | 14          | 1000      | 2      | 67.2      | 87        | JESD204B      | 585      |
| AD9234      | 12          | 1000      | 2      | 65.9      | 89        | JESD204B      | 238      |

Розглянемо сучасні швидкодіючі АЦП та їх характеристики на прикладі таблиці 2.1. Для зручності наведені найважливіші характеристики мікросхем, які розділені за типом інтерфейсу обміну з цифровим пристроєм і виведені за швидкістю передачі інформації (тактової частоти) від більшого значення до меншого.

У таблиці 2.1 представлено 4 топових мікросхеми, що передають дані в паралельній формі за стандартом LVDS. Розрядність даних 14-16, каналів 1-2, максимальна частота дискретизації 500МГц. Середні значення SNR і SFDR – 70 dBFS і 90 dBc, відповідно. Для деяких мікросхем із послідовною шиною передачі даних (Serial) характеристики практично такі. Для сучасних мікросхем АЦП, які використовують інтерфейс передачі JESD204B і підключаються до ПЛІС через гігабітні лінії, такі частоти значно вищі, але аналогові характеристики кілька вищі (з меншою мірою того, що розрядність цифрових даних нижче – 12 і 14 біт). У цій статті не буде розглядатися підключення АЦП до кристалів FPGA через гігабітні лінії. Я залишився на простому і добре вивченому варіанті підключення мікросхеми – послідовної та паралельної шини з інтерфейсами передачі даних типу LVTTTL, LVDS, LVCMOS, LVPECL. Зокрема, цікавий спосіб передачі даних по шині LVDS, який відрізняється простотою трасування друкованих плат, дешевим виконанням, низьким рівнем поміхи і розсіюваною потужністю, а також високою швидкістю передачі даних. Для прийому даних за послідовною шиною в ПЛІС використовуються серіалайзери / десеріалайзери. Для паралельної шини все набагато більше і дані приймаються як є [5].

Багато з модулів аналого-цифрового перетворення в стандарті FMC схожі і володіють практично ідентичними характеристиками, які визначають встановлені мікросхеми АЦП і ЦАП. Кінцевий вибір того чи іншого модуля визначається завданнями, які потрібно вирішити кінцевому користувачеві.

## 2.3 Прийом та передача радіосигналів у ПЛІС

Структурна схема приймача реалізованого мною показана на рисунку 2.13.

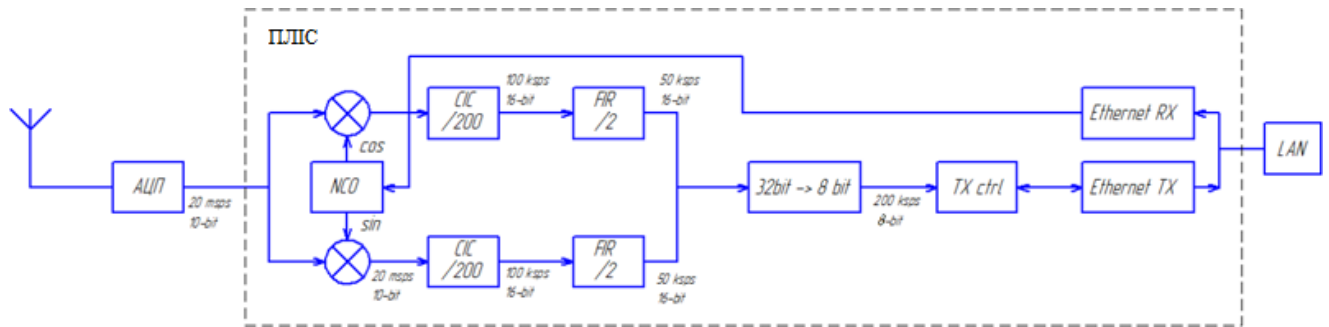


Рисунок 2.13 – Стурктура схема SDR приймача на ПЛС

Розглянемо послідовно компоненти, які проходить радіосигнал та цифрова інформація.

### 2.3.1 Підключення АЦП до ПЛС

Розглянемо підключення АЦП по паралельній та послідовній шині, з інтерфейсами передачі даних – LVTTTL, LVCMOS, LVDS. Як відомо, банки (групи пінів) введення/виведення ПЛС можуть конфігуруватися на різну напругу живлення, межі якого визначаються виробником кристалів. У сучасних ПЛС типу Xilinx діапазон напруг, що подаються, залежить від типу банку. Існує два основних типи банку (Xilinx) – HP (високопродуктивні) та HR (банки з великим вибором стандартів та живлення). Наприклад, у мікросхемах 7 серії Xilinx діапазон напруги для банків HR – від 1.2 до 3.3V, а для банків HP – від 1.2 до 1.8V. Крім цього, піни банків можуть програмуватися на широкий діапазон стандартних інтерфейсів передачі даних, але кінцевий вибір стандарту визначається двома правилами: напруга живлення банку ПЛС та спосіб підключення віддаленого пристрою та ПЛС [5].

З іншого боку, важливим фактором вибору напруги живлення банку ПЛС є сімейство кристалів. Так, для Kintex-7 можна використовувати стандарт LVDS 25 і живити банк напругою 2.5V, а для багатьох мікросхем ПЛС Virtex-7 стандарт підключення виключно LVDS 18, і банк запитується напругою 1.8V. У зв'язку з цим виникає проблема поєднання деяких аналогових модулів і несучих плат з різними сімействами ПЛС. Тому при покупці у стороннього виробника модуля АЦП та/або несучої плати потрібно переконатися, що вони сумісні за стандартом та напругою живлення. Як правило, інтерфейс передачі даних

прописується у файлі обмежень (UCF або XDC), а не у вихідних кодах проекту, що забезпечує гнучкість конфігурації та повторне використання файлів вихідних кодів в інших проектах. Структурна схема файлу верхнього рівня проекту (а також внутрішніх блоків, що використовуються для прийому даних від мікросхем АЦП за стандартом LVTTTL/LVCMOS/LVDS) представлена на наступному рисунку 2.13 [5].

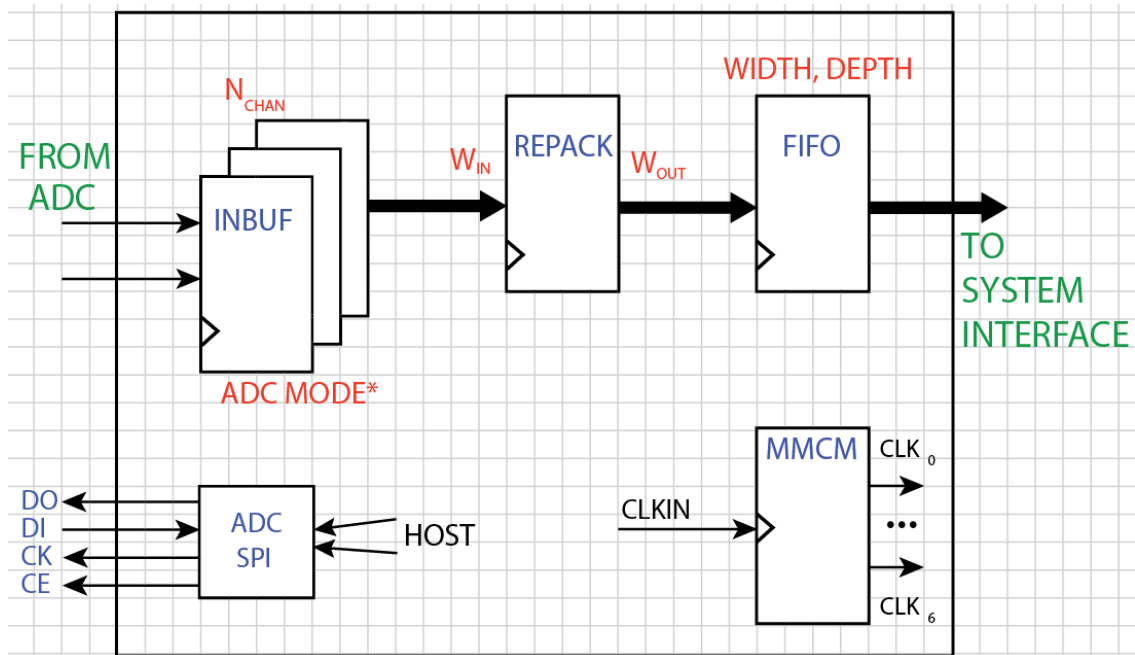


Рисунок 2.14 - Схема файлу верхнього рівня

Вона включає такі елементи:

- вхідний приймач;
- пакувальник даних;
- синхронізатор потоку даних (FIFO);
- блок керування мікросхемами АЦП.

Крім цих вузлів на верхньому рівні використовується модуль синтезу частоти на основі MMCM – стандартний ресурс ПЛІС. Також для керування затримками у вузлах IODELAY застосовується компонент IDELAYCTRL[6].

Основні параметри файлу верхнього рівня, від яких залежить конфігурація схеми ПЛІС:

- NCHAN - кількість незалежних каналів АЦП;

- DATA\_WIDTH - розрядність даних АЦП;
- FIFO\_ADDR - глибина FIFO:  $DEPTH = 2^{**}ADDR$ ;
- DATA\_RATE - режим прийому даних: SDR / DDR;
- PACK\_OUT - ширина вихідної шини пакувальника (системна шина);
- CLK\_SEL - вибір основного джерела тактової частоти для багатоканальної схеми;
- USE\_MMCM — використання MMCM замість BUFR для поділу частоти CLKDIV на SERDES;
- DIFF\_TERM — використання диференціальних вхідних сигналів (TRUE/FALSE);
- IOSTANDARD — стандарт напруги для LVDS ліній;
- DATA\_IOBDELAY\_TYPE - режим роботи вузла затримки IODELAY;
- DATA\_IOBDELAY\_VALUE — значення затримки за умовчанням;
- DIFF\_CLOCK — використання диференціального тактового сигналу на вході (TRUE/FALSE);
- OVR\_PRES - використання ліній переповнення АЦП (overflow);
- STR\_PRES - використання ліній сигналу зовнішнього старту АЦП (start);
- RESYNC\_OUT — увімкнення механізму первинної пересинхронізації даних на вхідному буфері (YES/NO).

Частина код файлу верхнього рівня приймача АЦП виглядає так:

```
xFIFO: CTRL_FIFO_CONFIG
generic map (
    DATA_WIDTH => PACK_OUT,
    ADDR_WIDTH => FIFO_ADDR
)
port map (
    reset => reset,
    wr_clk => clk_pack,
    rd_clk => sys_clk,
    data_i => do_pack,
```

```

data_o => do_fifo,
rd_en  => cs_fifo,
wr_en  => dv_pack,
empty  => ef_fifo,
full   => ff_fifo
);

```

Тут підключається вузол FIFO для синхронізації даних із системною шиною пристрою. Програмування затримок для IODELAY у багатоканальній та багаторозрядній схемі буфера виглядає наступним чином:

```

---- Data delays ----
x_dat_dd      <=      x_dat_do(conv_integer(unsigned(dl_chan)))      when
rising_edge(dl_clk);
x_dat_in(conv_integer(unsigned(dl_chan)))(conv_integer(unsigned(dl_muxs)))
<= dl_dat_in when rising_edge(dl_clk);
x_dat_ce(conv_integer(unsigned(dl_chan)))(conv_integer(unsigned(dl_muxs)))
<= dl_dat_ce when rising_edge(dl_clk);
x_dat_ld(conv_integer(unsigned(dl_chan)))(conv_integer(unsigned(dl_muxs)))
<= dl_dat_ld when rising_edge(dl_clk);

```

Де *dl\_chan* – вибраний канал АЦП, *dl\_muxs* – вибраний розряд АЦП. Це зроблено для незалежного програмування кожного біта даних. Розглянемо кожний вузол детально у пунктах 2.3.2, 2.3.4, 2.3.5, які наведено нижче.

### 2.3.2 Вхідний приймач

Містить первинні елементи підключення між зовнішніми сигналами друкованої плати та логікою ПЛІС. Це буфер IPAD від контакту ПЛІС до логіки вхідного буфера IOBUF. Залежно від інтерфейсу передачі буфер може бути диференціальний – IBUFDS і однополярний – IBUF. Для багатьох буферів можлива жорстка прив'язка до необхідного стандарту або можливе встановлення типу вихідного сигналу через атрибути. Наприклад, для IBUFDS за допомогою атрибуту IOSTANDARD можна заздалегідь у вихідних кодах визначити та задати стандарт «LVDS\_25»[6].

Слідом за буфером встановлюється вузол регулювання затримки проходження сигналу `IODELAY` для вирівнювання фронтів паралельної шини даних та первинного узгодження багатоканальних вузлів підключення АЦП до ПЛІС. `IODELAY` – програмований ресурс ПЛІС, величина затримки може бути фіксована і визначатися установкою за замовчуванням, або програмуватись в діапазоні від 0 до 31 умовних величин часу (цифра визначається даташитом на мікросхему ПЛІС). Програмування здійснюється примітивним чином за допомогою кількох сигналів. Для вузлів `IODELAY` необхідно підключати спеціальний вузол контролю затримок `IDELAYCTRL`, який у режимі реального часу калібрує затримки для кожного тактового регіону ПЛІС. Для усунення впливу температури кристала, напруги живлення та завантаженості ПЛІС, вузол `IDELAYCTRL` тактується незалежним сигналом `REFCLK`. Цей тактовий сигнал забезпечує рівну величину затримки (`TAP`) у всіх вузлах `IODELAY` кожного тактового регіону ПЛІС. Якщо в кожному тактовому регіоні потрібні свої величини підстроювання - в кристалі ПЛІС ініціалізуються кілька вузлів `IDELAYCTRL`. Тактовий сигнал `REFCLK` може подаватися із зовнішнього генератора, встановленого поза ПЛІС, або може бути отриманий шляхом синтезу частоти у вузлі `MMCM` кристала ПЛІС[6].

Слідом за вузлом затримки `IODELAY` можливе встановлення `FD` тригера, якщо режим передачі даних `DATA_RATE = "SDR"`. Цей тригер опціонально міститься у вхідний буфер і тактується частотою АЦП, що супроводжує дані з аналогового модуля. У режимі `DATA_RATE = DDR` тригер замінюється на ускладнений варіант `IDDR`, який має один вхід даних і два виходи. З виходу можлива видача даних у кількох режимах, але практично я використовував лише одне – це «`SAME_EDGE_PIPELINED`», у якому дані з виходу `Q1` і `Q2` видаються з одному й тому фронту тактового сигналу.

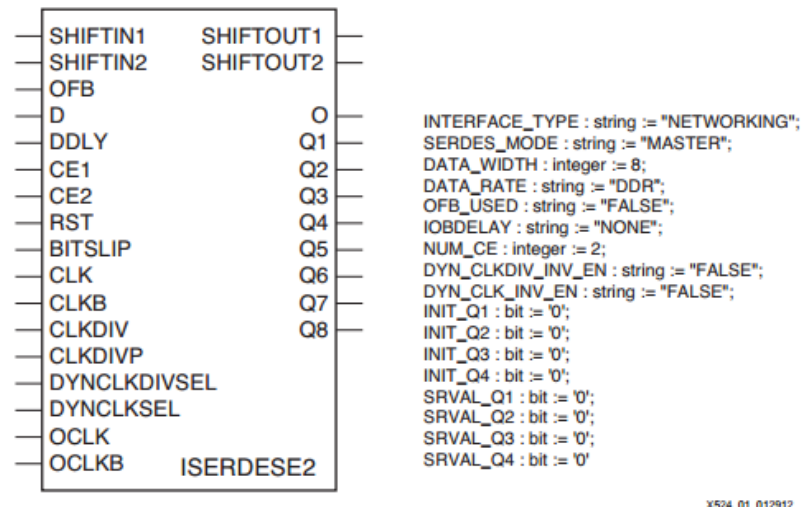


Рисунок 2.15 - Підключення вузла IODELAY до модуля десеріалізації даних ISERDES

Підключення вузла IODELAY до модуля десеріалізації даних ISERDES, який показано на рисунку 2.15. Вузол серіалайзера – складний елемент вхідного буфера ПЛІС, що містить наступні опції:

- DATA\_RATE – визначає режим вихідних даних (SDR/DDR);
- DATA\_WIDTH – визначає розрядність шини даних на виході вузла (від 1 до 8);
- INTERFACE\_TYPE – тип інтерфейсу передачі. Для підключення мікросхем АЦП використовується режим "NETWORKING";
- SERDES\_MODE – режим підключення ISERDES у разі використання кількох серіалайзерів для отримання даних від одного контакту ПЛІС.

Тактування вузла здійснюється декількома сигналами - це сигнал на високій частоті даних CLK і сигнал CLKDIV, поділений на число, що визначається параметром DATA\_WIDTH. Поділ може відбуватися поза ПЛІС, але на практиці використовуються синтезатори частоти в самому кристалі ПЛІС, які зветься MMCM. Таким чином, на вхід вузла ISERDES надходить послідовний сигнал на частоті CLK, а на виході вузла ISERDES виходить паралельний потік даних на зниженій частоті  $CLKDIV = CLK/DATA\_WIDTH$ . Тактову частоту можна розділити за допомогою регіонального буфера BUFR, значення

коефіцієнта розподілу BUFR\_DIVIDE = 1-8, або на вузлі синтезатора частоти MMCM, що програмується в широкому діапазоні[7].

Усі перелічені вище вузли (крім MMCM) входять у складі вхідного буфера ПЛИС і займають логічних ресурсів кристала. З виходу FD/IDDR/ISERDES дані надходять вже на логічні елементи – це тригер або FIFO невеликого розміру для первинної пересинхронізації даних на частоту, отриману після проходження частоти дискретизації АЦП у вузлі MMCM. У деяких випадках пересинхронізація не використовується, а дані прямо синхронізуються частотою дискретизації АЦП. На наступному малюнку наведено часову діаграму обробки послідовних даних з мікросхеми АЦП на пристрої ISERDES. Показано вхідну тактову частоту, послідовну форму вхідних даних та перетворені дані в паралельний код[7].

Таким чином, залежно від конфігурації, у вхідному буфері можливо три шляхи проходження сигналу, як показано на рисунку 2.16:

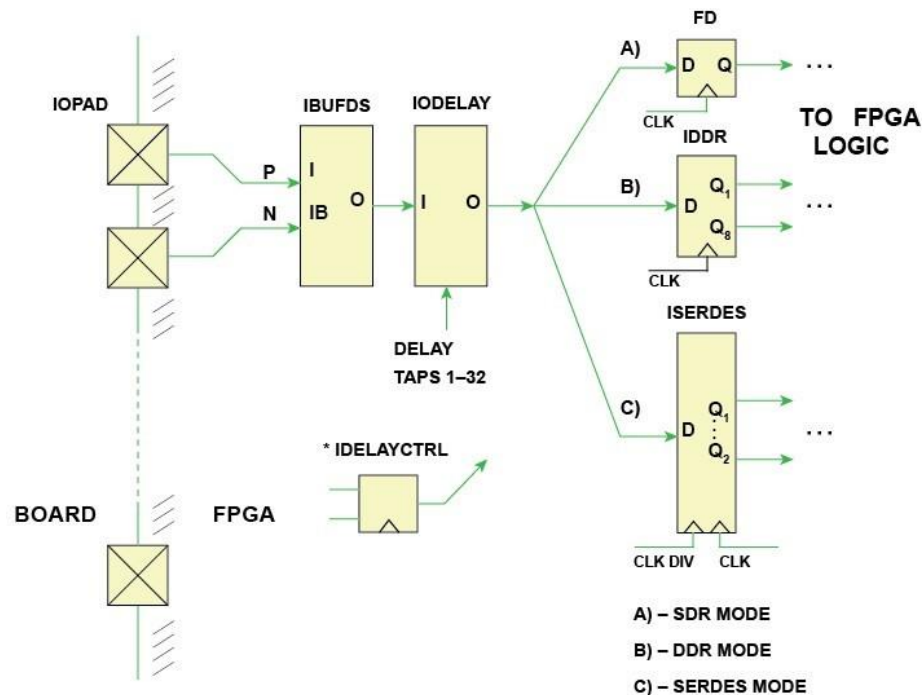


Рисунок 2.16 - Шляхи проходження сигналу

А) режим SDR - дані приймаються по паралельній шині, для кожного біта даних використовується свій буфер ПЛИС;

В) режим DDR – дані приймаються по паралельній шині, дані тактуються подвоєною частотою;

С) режим SERDES – дані приймаються по послідовній шині.

### 2.3.3 Пакувальник даних

Цей вузол використовується в одноканальних та багатоканальних схемах прийому даних від АЦП. Основне його призначення – упакувати дані у зручний формат внутрішньої шини, що використовується всередині ПЛІС. Розрядність внутрішньої шини визначається розробником чи корпоративним стандартом. Типові значення розрядності на виході пакувальника - 32, 64, 128, 256, 512 біт даних. У деяких випадках пакувальник не може бути налаштований. Це залежить від кількості вхідних каналів, розрядності вхідної та вихідної шини даних пакувальника. Наприклад, для чотириканального джерела з розрядністю вхідних даних = 16 біт неможливо організувати перепакувальник із вихідною шиною розрядністю менше  $WOUT = WIN * NCHAN = 16 * 4 = 64$  біт [8].

Для одноканального джерела даних з тією ж розрядністю на вході та виході дані будуть послідовно упаковані пачками по 16 біт у розрядну сітку 64 біти. Тобто перший відлік АЦП займе бітове поле [15:0], другий відлік – [31:16], третій – [47:32], а четвертий – [63:48]. Далі відліки циклічно займатимуть бітові поля від молодшого до старшого у вихідній сітці даних. Інший приклад. Використовується 2 канали даних, розрядності на вході та виході залишаються ті самі – 16 і 64 біт відповідно. Якщо система збору даних на віддаленому пристрої включає тільки один канал, упаковка відбувається, як показано в прикладі вище, причому неважливо, який з каналів включений. Якщо система збору включає обидва канали, дані упаковуються попарно від молодшої частини розрядної сітки до старшої. Перший відлік першого каналу займе позицію [15:0], перший відлік другого каналу займе позицію [31:16], другий відлік першого каналу – [47:32], другий відлік другого каналу – [63:48].

Це найзручніший варіант упаковки даних при багатоканальній обробці у разі відключення тих чи інших каналів передачі. Примітка: кількість каналів має бути кратною мірою двійки – 1, 2, 4, 8 і т.д. [8].

### 2.3.5 Синхронізатор потоку даних

Цей вузол будується з урахуванням примітивної схеми FIFO і має одне важливе і просте призначення – перетворення потоку даних із частоти дискретизації CLK\_ADC на частоту обробки чи частоту роботи внутрішньої шини пристрою SYS\_CLK (системна тактова частота). Для простоти можна створити вузол FIFO у Xilinx Core Generator, але в ідеалі краще написати свій модуль, який буде гнучким у конфігурації на довільну ширину шини (FIFO\_WIDTH) та глибину пам'яті (FIFO\_DEPTH). На наступному малюнку наведено часові діаграми роботи вузла FIFO, показані вхідні та вихідні дані, а також прапори порожнього (empty) та повного (full) FIFO. Слід зазначити, що для нормальної роботи FIFO АЦП прапор заповненості FULL повинен завжди бути рівним 0, що говорить про правильність прийняття даних від АЦП і відсутність переповнення. Якщо прапор FULL = 1, швидкість прийому даних віддаленим обробником недостатня, або швидкість роботи АЦП занадто велика для обраної системи [9].

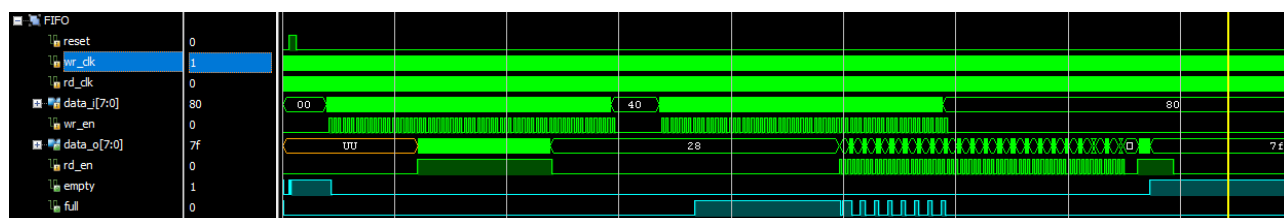


Рисунок 2.17 - Схематичне уявлення вузла FIFO серед Plan Ahead.

### 2.3.6 Внутрішня шина

Виникає питання — а куди далі йдуть зібрані та запаковані дані? Відповідь проста: за допомогою інтерфейсів PCI/PCIe, USB, Ethernet, SRIO, SATA дані досягають кінцевої мети, де проходять подальшу обробку і здійснюється збір всієї інформації. До цього ланки всередині ПЛІС може бути вузол цифрової обробки сигналів (DSP), наприклад — DDC, DUC, FFT / IFFT тощо. буд. Крім того, дані можуть буферизуватися на зовнішній пам'яті великого обсягу, типу DDR3 / DDR4. Це найчастіший прийом, який застосовується для обробки

великого потоку інформації на високих швидкостях. У разі зовнішня пам'ять імітує FIFO великого розміру. Для з'єднання ПЛІС із зовнішньою пам'яттю застосовуються спеціальні контролери, які, як правило, доступні та відкриті для використання. У випадку Altera вони знаходяться в IP Core Generator.

Крім іншого, розробнику необхідно створити власний кінцевий автомат управління всіма ресурсами приймача АЦП: це конфігурація затримок в IODELAY, програмування синтезатора частоти MMCM, передача команд за SPI на АЦП і ЦАП модулі, управління початком та кінцем збору даних з АЦП, керування кількістю активних каналів, використання зовнішньої пам'яті DDR та багато іншого. Усі це мають свою унікальну архітектуру, але базові принципи залишаються однаковими [9].

## **2.4 Розробка SDR на платі De0-nano**

### **2.4.1 Антена**

Гарна антена АМ-діапазону може бути простим довгим дротом, протягнутим між двома деревами або на даху. Навіть невелика довжина дроту забезпечить вашому приймачу значно кращий прийом із меншою статичністю, оскільки прийом сигналу відбувається на деякій відстані від побутових приладів, що створюють перешкоди. Ізолятор, встановлений високо на дереві, щоб провід мав великий вертикальний підйом, дасть чудові результати.

Від антени залежить дуже багато. Більшість найцікавіших сигналів на коротких хвилях не можна прийняти на просту антену (наприклад, шматок дроту). За містом особливих проблем немає – досить довгий провід може працювати гарною антеною (на прийом). У місті, особливо всередині великих залізобетонних будинків все значно гірше — довгу антену не розтягнути, при цьому шумів, що заважають дуже багато (побутові прилади здатні створювати в ефірі дуже великий рівень шуму), так що вибір антени стає непростою справою. Для прийому радіосигналів я користуюся активною рамковою антеною.

Основна перевага рамкової антени - за рахунок використання явища резонансу вона дозволяє придушувати шуми на частотах, що не використовуються, проте є і недолік - при переході з одного діапазону частот на іншу антену потрібно перебудовувати[15].

### 2.4.2 АЦП

АЦП повинен мати велику розрядність для підвищення динамічного діапазону, а для DDC приймача — ще й високою швидкодією. Зазвичай у хороші DDC приймачі ставлять АЦП з розрядністю 16-біт та швидкодією >50 MSPS.

Але можна вибрати дуже дешеві, але менш потужні АЦП наприклад AD9200 - 10-бітний 20 MSPS. Це дуже посередні характеристики для приймача DDC, проте, як показали його виробники в роботі, АЦП цілком придатний для прийому сигналів.

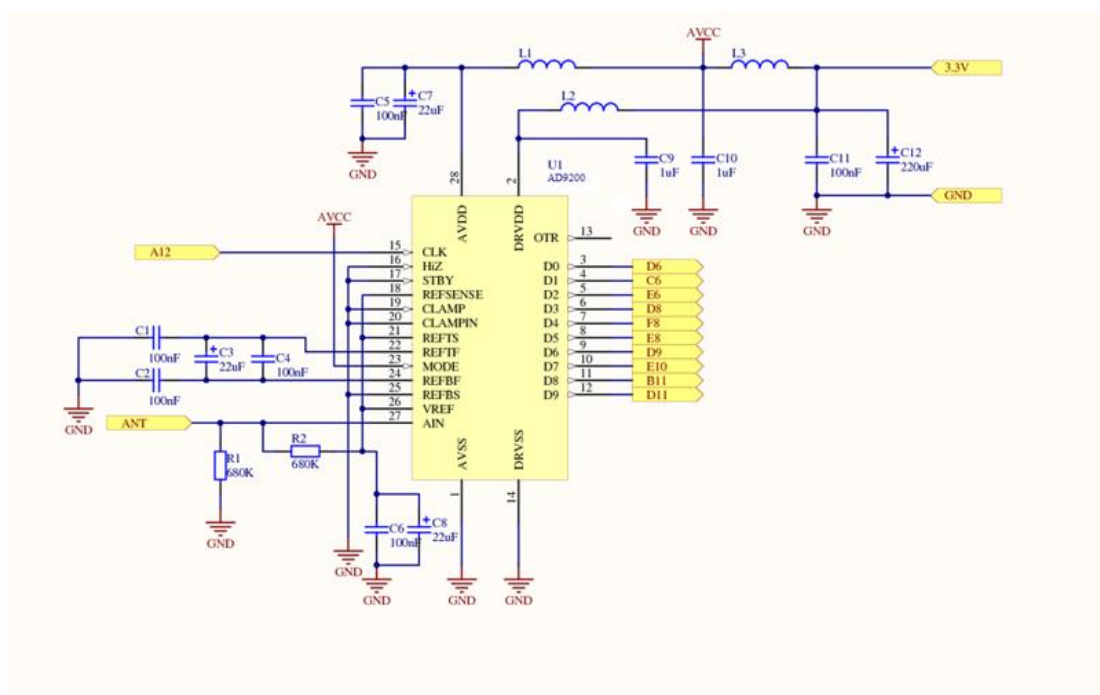


Рисунок 2.17 – Схема підключення АЦП

Так як АЦП оцифровує тільки сигнали позитивного рівня, а сигнал з антени біполярний, сигнал доводиться зміщувати на половину опорної напруги (для цього служать резистори R1 і R2). Штучно створена постійна складова потім

віднімається вже з цифрового сигналу ПЛІС. Вся подальша обробка сигналу після АЦП йде в ПЛІС, як показано на рисунку 2.17.

Потік даних із АЦП становить 200 Мбіт (10-bit x 20 MSPS). Передати такий потік безпосередньо в комп'ютер, а потім ще й обробити дуже складно, тому частоту сигналу потрібно спеціально знизити. При перенесенні більш низьку частоту виникає явище «дзеркального каналу», для боротьби з яким використовують квадратурне перетворення частоти — сигнал перетворюють на комплексну форму (відбувається поділ на два канали I/Q). Перенесення більш низьку частоту проводиться шляхом множення вихідного сигналу на сигнал генератора. У ПЛІС, що використовується, достатньо апаратних помножувачів, які допомагають вирішити цю проблему [20].

#### **2.4.3 NCO та CIC-фільтр**

Для того щоб переносити вхідний сигнал на потрібну частоту, її потрібно створити. Для цього використовується готовий компонент Quartus - NCO (numerically controlled oscillator). На генератор подається тактова частота, така ж як у АЦП (20 МГц), на його вхід управління подається значення, що визначає частоту, і на його виході формується цифровий синусоїдальний сигнал потрібної частоти, дискретизований з частотою 20 МГц. NCO здатний паралельно формувати косинусний сигнал, завдяки чому можна формувати квадратурний сигнал.

Після змішування з сигналом генератора з виходу помножувачів сигнал виходить вже перенесений більш низьку частоту, але все ще з високою частотою дискретизації (20 MSPS). Сигнал потрібно децимувати, тобто відкинути частину вибірок. Просто так відкинути зайві вибірки не можна, оскільки це призведе до викривлення вихідного сигналу. Тому сигнал слід пропустити через спеціальний фільтр (CIC-фільтр). У цьому випадку я хотів отримати на виході приймача частоту дискретизації 50 кГц сигналу. З цього випливає, що частота повинна бути знижена ( $20 \times 10^6 / 50 \times 10^3 = 400$ ) разів. Децимацію доведеться проводити у 2 етапи – спочатку у 200, потім у 2 рази.

Перший етап виконує саме CIC-фільтр. Я використав 5-каскадний фільтр.

В результаті роботи CIC-фільтр за рахунок зниження смуги сигналу збільшується розрядність вихідного сигналу. Оскільки каналів у приймачі два, те й фільтрів знадобиться також два.

CIC-фільтр має досить круту АЧХ, що прагне 0 при наближенні до вихідної частоти дискретизації (100 кГц). Для компенсації її кривизни слугує наступний компенсаційний FIR-фільтр [21].

#### 2.4.4 Компенсаційний FIR-фільтр

Цей фільтр потрібен для того, щоб компенсувати спад АЧХ CIC-фільтра та виконати ще один етап децимації (удвічі). Altera вже подбали про методику розрахунку цього фільтра — при створенні CIC-фільтра автоматично формується програма для Matlab, запустивши яку, можна сформувати коефіцієнти для компенсаційного фільтра.

Вигляд АЧХ CIC, FIR і результату (графіки будує та ж програма для Matlab):

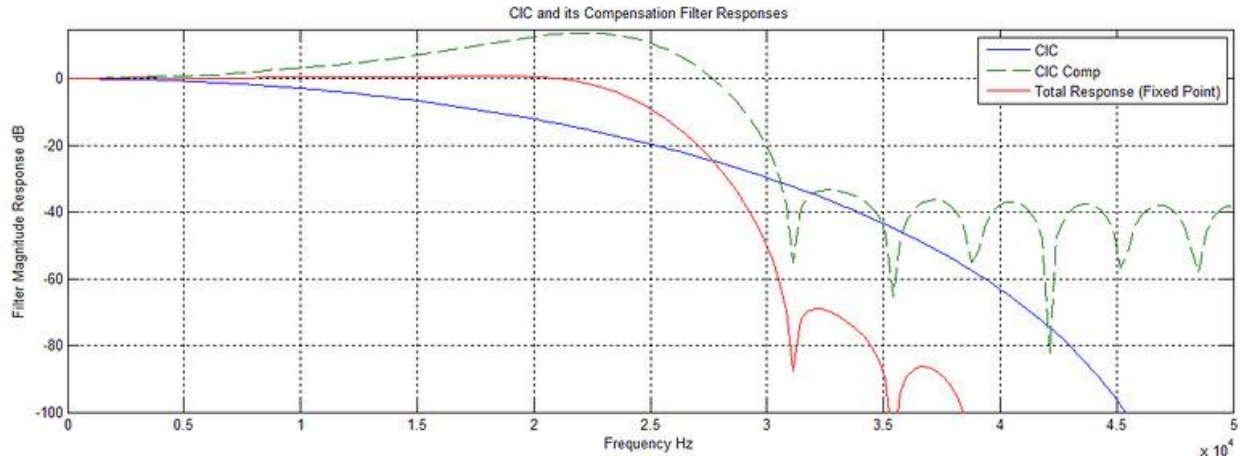


Рисунок 2.18 – графік вигляду АЧХ CIC, FIR

Видно, що на частоті 25 кГц CIC-фільтр послабить сигнал на 20 Дб, що дуже багато, проте з використанням FIR-фільтра ослаблення всього 10 Дб, а на нижчих частотах ослаблення практично відсутнє.

На виході FIR-фільтра з урахуванням децимації частота дискретизації сигналу буде 50 кГц.

Чому не можна відразу було зробити децимацію сигналу в 400 разів? Це з тим, що частота зрізу FIR-фільтра повинна становити  $1/4$  з його вихідний. У разі частота дискретизації на виході фільтра без децимації, як і його вході, становить 100 кГц. В результаті цього частота зрізу буде 25 кГц, що і видно на графіках вище [22].

Обидва фільтри є готовими компонентами Quartus.

#### **2.4.5 Передача даних на комп'ютер та від комп'ютера**

Отриманий потік даних  $((16+16)\text{bit} \times 50 \text{ KSPS} = 1.6 \text{ Mbit})$  потрібно передати на комп'ютер. Дані я вирішив надсилати через Ethernet. На налагоджувальній платі немає такого інтерфейсу. Найбільш правильно було б зробити окрему плату з контролером PHY, запустити soft-процесор Nios, та передавати дані через них. Однак це значно ускладнює конструкцію. Я пішов більш простим шляхом – Ethernet пакети можна формувати на самій ПЛІС, таким чином можна передавати дані зі швидкістю 10 Мбіт.

У проекті ПЛІС посилає на комп'ютер із заданою IP та MAC адресою певний UDP-пакет. Модуль Ethernet передавача може передавати 1024 байти, зчитуючи їх із RAM. В результаті на комп'ютер в одному пакеті відправляються 256 пар 16-бітових значень сигналу взятого з виходів фільтрів. Так як дані надходять від АЦП безперервно, а відправляти їх на комп'ютер потрібно пакетами, то довелося реалізувати подвійну буферизацію пам'яті - в той час поки одна RAM заповнюється, дані з іншої RAM передаються по Ethernet. Після того, як перша RAM заповниться, обидві RAM «міняються місцями», за що відповідає досить простий модуль, що управляє.

Так як на виході фільтрів дані передаються потоком з пари 16 бітних величин, а Ethernet передаються окремі байти, то для перетворення потоків в конструкцію введений модуль, який перетворює потік 32 біт 50 KSPS в 8 біт 200 KSPS.

Для того, щоб керувати приймачем (встановлювати частоту налаштування), на нього потрібно передавати певну величину, яка використовуватиметься для визначення частоти NCO.

Для прийому цієї величини також використовується компонент з вищевказаного сайту, модифікований для прийому даних, і видачі їх у вигляді 24 бітного числа. Оскільки модулі приймача і передавача не пов'язані друг з одним, то реалізувати ARP не можна, і це означає, що приймач нічого очікувати мати IP і MAC адреси. Надіслати інформацію на нього можна, якщо відправити в мережу ширококомовний пакет [23].

Ресурси, що будуть використовуватися програмою ПЛІС:

- 5006 LE;
- 68 9-bit помножувачів (64 з них використовуються у FIR-фільтрі);
- 16,826 bit пам'яті (8 блоків M9K).

Вигляд проекту у QUARTUS II показано на рисунку 2.19

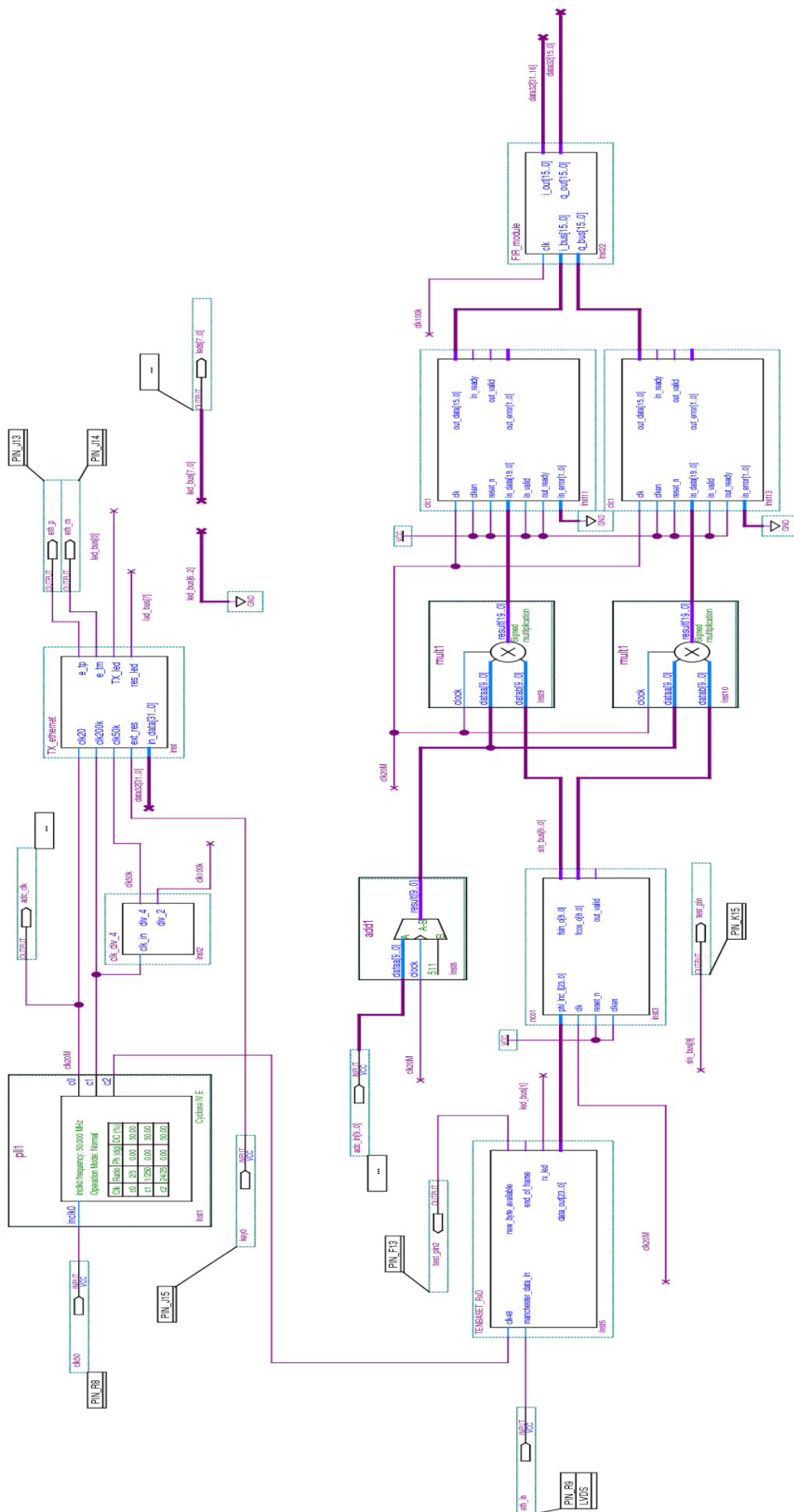


Рисунок 2.19 – проект у QUARTUS II

### 2.4.6 Обробка даних на комп'ютері

Після того, як комп'ютер отримав дані, їх потрібно обробити. Найкраще взяти готову програму. Зазвичай у SDR програмах реалізують необхідні цифрові фільтри, алгоритми, призначені на формування звуку та її фільтрацію, БПФ прийнятого сигналу, побудова його спектру і «водоспаду».

Я використовую програму HDSDR, вона підтримує введення даних за допомогою бібліотеки ExtIO (формат Winrad). Вимоги програм до бібліотеки добре документовані.

Оскільки частота дискретизації I/Q сигналів на виході фільтрів приймача становить 50 кГц, то програмі при прийомі для огляду буде доступна смуга 50 кГц. ( $\pm 25$  кГц від частоти, що формується NCO).

Блок живлення забезпечує напругу 12 В для живлення підсилювача антени.

Тепер про те, що вдається прийняти на КВ. Незважаючи на досить високий рівень шумів, вдається прийняти багато сигналів, добре приймаються мовні радіостанції, непогано приймаються радіоаматори.

Можна зробити висновок, що дана конфігуруєма радіосистема(SDR) на основі ПЛІС може приймати сигнали на заданій частоті, та повністю програмована, єдиним недоліком котрої є її залежність від комп'ютера(ноутбука), тобто неможливість її автономної роботи. Для вирішення цього моменту потрібно розробити демодулятор, для того щоб виділення сигналу та його модуляція відбувалася одразу на ПЛІС, а також розробити вивід звуку через ПЛІС, FFT, виведення спектру на екран та керування приймачем без використання потужностей комп'ютера [23]. Такий приймач мав би схему показано на плакаті Е1 .

## 3 КЕРІВНИЦТВО КОРИСТУВАЧА

### 3.1 Програмне забезпечення Quartus II(Quartus Prime)

Для створення проектів під ПЛІС Altera використовується фірмова IDE - Quartus. Версій Quartus існує незліченна безліч, і потрібно вибрати правильну під наявну мікросхему. Altera регулярно викидає підтримку старих чіпів з нових версій свого ПО. У свою чергу, в нових версіях Quartus змінюються вимоги до програмного і апаратного забезпечення комп'ютера. Наприклад, Quartus 13.1 є стабільною версією, що працює як 64-х так і 32-х розрядних системах. У 2015 році Altera була придбана компанією Intel. Бренд Altera припинив існування в липні 2018 року, адреса сайту altera.com тепер нікуди не веде. Нове ім'я Altera - Intel Programmable Solutions Group (PSG). Вся документація і ПО необхідні для роботи з ПЛІС Altera, а також детальна схема початку роботи з дистрибутивом тепер доступні на сайті intel.com [3].

Для роботи з мікросхемою De0 буде потрібно сам Quartus, і пакет для підтримки De0-Nano. Це файли «QuartusSetupWeb-13.1.0.162.exe» і «de0Nano\_web-13.1.0.162.qdz» відповідно.

Весь процес створення конфігурації ПЛІС в Quartus II можна розбити на кілька етапів:

- проектування цифрової схеми;
- аналіз створеної схеми на наявність помилок;
- прив'язка входів і виходів схеми до конкретних висновків ПЛІС;
- компіляція проекту і створення довічних файлів для конфігурації ПЛІС;
- конфігурація ПЛІС за допомогою USB Blaster.

USB Blaster - це USB-сумісний завантажувальний кабель для внутрисхемного конфігурації і програмування мікросхем ALTERA (FPGA, CPLD, Active Serial Configuration Devices і Enhanced Configuration Devices). Він сумісний з оригінальним PL-USB-BLASTER-RCN [3].

### 3.2 Налаштування програми HDSDR для приймача SDR

HDSDR — це безкоштовна програма для комп'ютерів Windows, написана та розповсюджена Маріо Теубелем. Це прикладна програма SDR, яка часто використовується для роботи програмно визначених радіостанцій на основі звукової карти.

Для більшості радіостанцій HDSDR контролює частоту гетеродина через USB-з'єднання через файл DLL ExtIO, написаний для конкретного SDR. Деякі файли DLL доступні на сайті виробника, а на деякі є посилання на веб-сайтах виробників радіоприймачів. DLL-файли, які використовуються HDSDR, часто називають «файлами Winrad ExtIO» на честь програмного коду Winrad SDR з відкритим кодом, на якому базується HDSDR [12].

Після розпакування zip-архіву ExtIO DLL-файлу для радіо, потрібно скопіюєте вміст у каталог програм Windows для HDSDR (зазвичай C:/Program Files (x86)/HDSDR/):

- натисніть кнопку ExtIO (зелена стрілка). Там можна ставити галочки, щоб вибрати оптимальне посилення приймача. Жовта стрілка - повзунок керування чутливістю як показано на рисунку 3.6;
- у цих опціях регулюється крок перебудови частоти, коли коліском мишки налаштовуєтеся на станції. Рекомендуємо поставити 50Гц, або можете вибрати свій крок, як показано на рисунку 3.7;
- в меню ви можете вибрати візуальне оформлення (кольорову гаму) водоспаду радіосигналів;

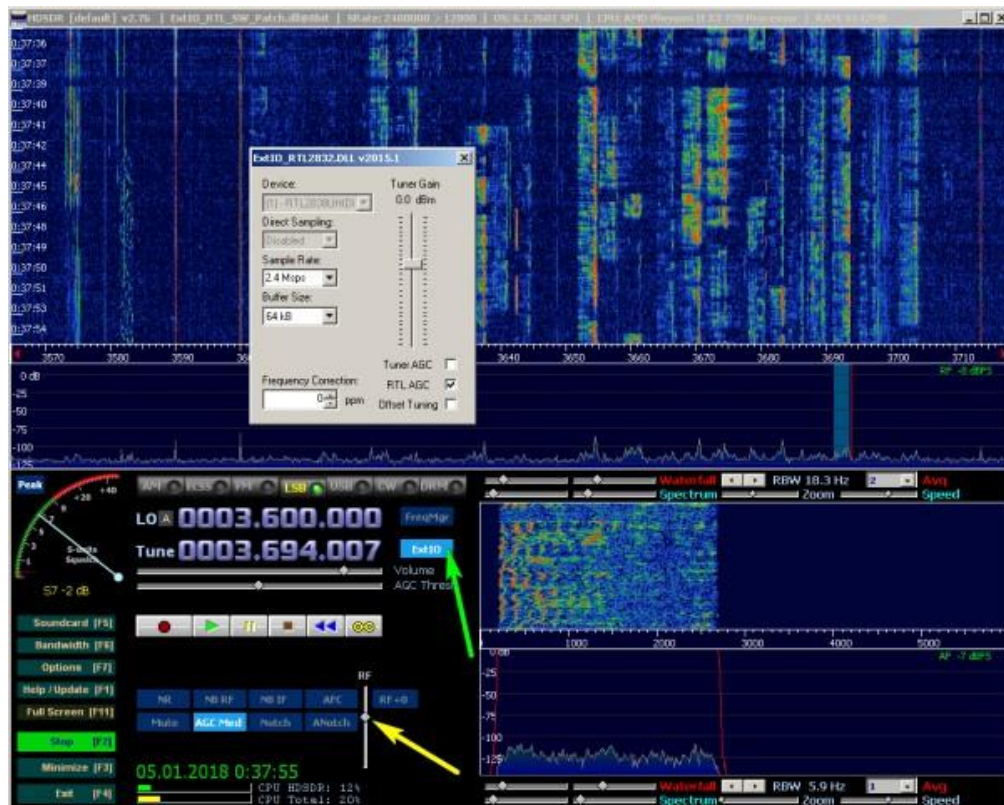


Рисунок 3.6 – Вікно налаштувань **HDSDR**



Рисунок 3.7 – Вікно налаштування кроку частоти

– як показано на рисунку 3.8. Зелена стрілка – це встановлення частоти. Можете навести мишку на шкалу, обернути коліщатко і вибирати частоту, або можете помітити курсор мишки на водоспад сигналів, обернути

коліщатко та налаштовуватися на станції. Жовті стрілки – це вибір виду модуляції;

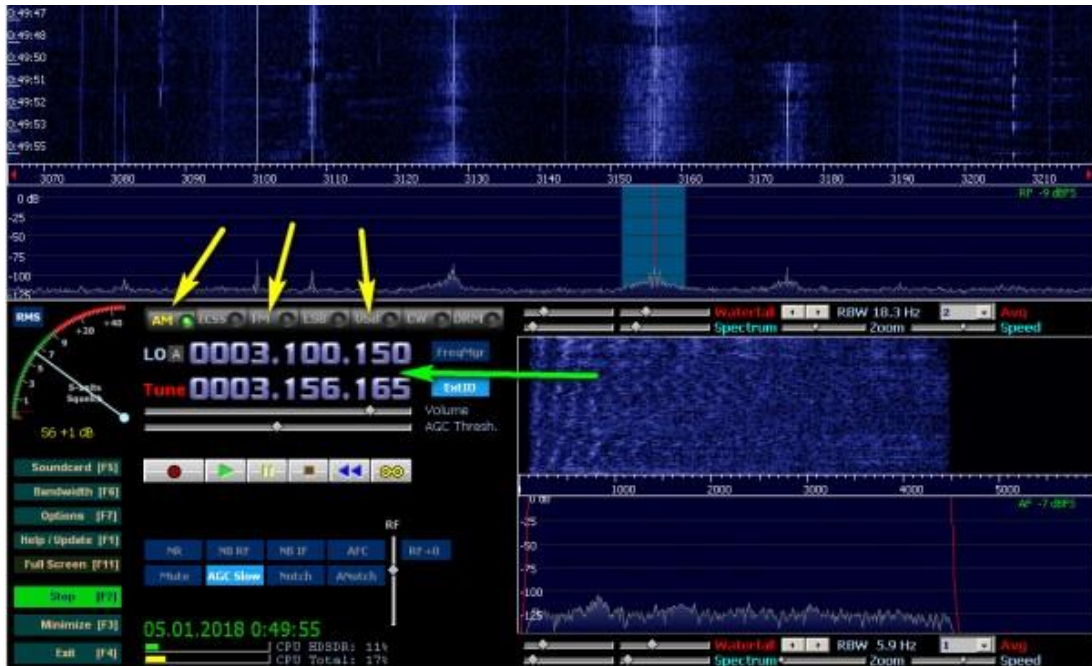


Рисунок 3.8 – Вікно встановлення частоти

- AM – це для прослуховування мовних радіостанцій та радіохуліганів на 3МГц;
- LSB - це односмугова (SSB) модуляція з нижньою бічною смугою, такою модуляцією працюють радіоаматори на діапазонах 80м (3.5-3.8МГц) та 40м (7.0-7.2МГц);
- USB – це односмугова модуляція (SSB) з верхньою бічною смугою, її використовують радіоаматори на діапазоні 20м (14.0-14.3МГц) та вище за частотою;
- CW – це для прийому азбуки Морзе.

Це найголовніші налаштування програми HDSDR, які необхідно знати для тестів SDR. В разі неправильних налаштувань, або зупинки роботи програми, повернутися до заводських налаштувань можна наступним чином:

- options;
- misc Options;
- reset to factory settings.

У підсумку можна сказати, що радіочастотний спектр високої видимості та водоспадний дисплей у HDSDR мають регульований розмір і вибір колірних схем. Окремий спектр АФ і водоспад показують налаштований сигнал більш детально. Комбінації клавіш можна використовувати замість миші для багатьох елементів керування програмою. HDSDR та її прямиї конкуренти SDRSharp - це корисні безкоштовні програми для налаштування SDR з комп'ютера [13].

## ВИСНОВКИ

Для розробки конфігуруємої радіосистеми на ПЛІС було проаналізовано структуру, мікросхеми, які використовують для проектування. Також ознайомлено з основними архітектурними особливостями ПЛІС(FPGA) мікросхем компанії Altera, а саме Cyclone III і основні їх характеристики. Розглянуті та проаналізовані дві сучасні стенди з FPGA, а саме De0-Nano від компанії Altera та Artix-7 FPGA AC701 від компанії Xilinx, зроблені відповідні висновки про перевагу саме De0-Nano для розробки SDR на її основі.

Було розглянуто SDR приймачі, їх види, проаналізована архітектура, розглянутий детально DDC SDR приймач, наведені його переваги, та зроблений вибір в його користь. Детально розглянуто АЦП по паралельній та послідовній шині, а також три режими прийому: SDR(дані приймаються по паралельній шині, для кожного біта даних використовується свій буфер ПЛІС), DDR(дані приймаються по паралельній шині, дані тактуються подвоєною частотою), SER-DES(дані приймаються по послідовній шині), метод та технологію отримання даних з ПЛІС за допомогою Ethernet кабелю та його переваги. Створено лістинги та розглянуті його головні частини, а також визначили варіанти покращення(апгрейду) радіосистеми, а саме розробити демодулятор, для того щоб виділення сигналу та його модуляція відбувалася одразу на ПЛІС, а також розробити вивід звуку через ПЛІС, FFT, виведення спектру на екран та керування приймачем без використання потужностей комп'ютера та розробили приблизну схему покращеного приймача.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Що таке VHDL. URL: <https://cutt.ly/ayRfGfH>
2. De0-Nano User Manual. URL: <https://www.ti.com/lit/ug/tidu737/tidu737.pdf>
3. Посібник з Quartus II. URL: <https://studfile.net/preview/3905247/>
4. Іванець С.А., Зубань В.В., Казимир В.В., Литвинов В.В. Проектування комп'ютерних систем на основі мікросхем програмованої логіки: Суми : Сумський державний університет, 2013. 313 с.
5. W. Kester. ADC Architectures I: The Flash Converter. Analog Devices, MT-020 Tutorial. URL: [www.analog.com/static/imported-files/tutorials/MT-020.pdf](http://www.analog.com/static/imported-files/tutorials/MT-020.pdf)
6. W. Kester. ADC Architectures II: Successive Approximation ADC. Analog Devices, MT-021 Tutorial. URL: [www.analog.com/static/imported-files/tutorials/MT-021.pdf](http://www.analog.com/static/imported-files/tutorials/MT-021.pdf)
7. W. Kester. ADC Architectures III: Sigma-Delta ADC Basics. Analog Devices, MT-022 Tutorial. URL: [www.analog.com/static/imported-files/tutorials/MT-022.pdf](http://www.analog.com/static/imported-files/tutorials/MT-022.pdf)
8. W. Kester. ADC Architectures IV: Sigma-Delta ADC Advanced Concepts and Applications. Analog Devices, MT-023 Tutorial. URL: [www.analog.com/static/imported-files/tutorials/MT-023.pdf](http://www.analog.com/static/imported-files/tutorials/MT-023.pdf)
9. What's the Difference Between VHDL, Verilog, and SystemVerilog? URL: <https://www.electronicdesign.com/resources/whats-the-difference-between/article/21800239/whats-the-difference-between-vhdl-verilog-and-systemverilog>
10. Software defined radio: architectures, systems, and functions. Dillinger, Madani, Alonistioti. Wiley, 2003. 454 с.
11. Software-Defined Radio for Engineers Illustrated Edition. Travis F Collins; Pu Di, 2018. 354 с.
12. HSDR Accessibility URL: <https://www.hdsdr.de/index.html>

13. Operating a Software-Defined Radio. URL: <https://www.hamradioandvision.com/hdsdr-accessibility>
14. UG952 - AC701 Evaluation Board for the Artix-7 FPGA User Guide (UG952) (v1.4). URL: <https://docs.xilinx.com/v/u/en-US/ug952-ac701-a7-eval-bd>
15. Home made antennas. URL: <http://techlib.com/electronics/antennas.html>
16. Design and asic implementation of duc/ddc for communication systems URL: <https://arxiv.org/ftp/arxiv/papers/1201/1201.2107.pdf>
17. VLSI Design - VHDL Introduction. URL: [https://www.tutorialspoint.com/vlsi\\_design/vlsi\\_design\\_vhdl\\_introduction.htm](https://www.tutorialspoint.com/vlsi_design/vlsi_design_vhdl_introduction.htm)
18. Hardware Description Languages: VHDL vs Verilog, and Their Functional Uses. URL: <https://resources.pcb.cadence.com/blog/2020-hardware-description-languages-vhdl-vs-verilog-and-their-functional-uses>
19. Verilog HDL and how it's used in programming. URL: <https://www.quora.com/What-is-Verilog-HDL-How-is-it-used-in-programming>
20. AD9200 Технічний опис та інформація про продукт. URL: <https://www.analog.com/ru/products/ad9200.html>
21. Digital Up-Down Converter Using Optimized CIC Filter. URL: [https://www.researchgate.net/publication/277898726\\_Digital\\_Up-Down\\_Converter\\_Using\\_Optimized\\_CIC\\_Filter](https://www.researchgate.net/publication/277898726_Digital_Up-Down_Converter_Using_Optimized_CIC_Filter)
22. FIR Filter Compensation Coefficients. URL: <https://www.intel.com/content/www/us/en/docs/programmable/683246/19-3-19-3/fir-filter-compensation-coefficients.html>
23. Software-Defined Radio Transceiver Design Using FPGA. URL: <https://ieeexplore.ieee.org/document/9274350>

## ДОДАТОК А

### ОСНОВНІ ФАЙЛИ ПРОГРАМ

#### Лістинг А.1 – Код передачі даних з ПЛІС через Ethernet на ПК

```

module TENBASET_TxD(clk20, Ethernet_TDp, Ethernet_TDm);

// a 20MHz clock
input clk20;

// the two differential 10BASE-T outputs
output Ethernet_TDp, Ethernet_TDm;

// "IP source" - вставте невикористану IP-адресу - якщо не впевнені,
// перегляньте коментар нижче після вихідного коду
parameter IPsource_1 = 192;
parameter IPsource_2 = 168;
parameter IPsource_3 = 0;
parameter IPsource_4 = 44;

// "IP destination" - введіть IP-адресу ПК, на який ви хочете
// надіслати
parameter IPdestination_1 = 192;
parameter IPdestination_2 = 168;
parameter IPdestination_3 = 0;
parameter IPdestination_4 = 2;

// "Physical Address" - введіть адресу ПК, на який ви хочете надіслати

parameter PhysicalAddress_1 = 8'h00;
parameter PhysicalAddress_2 = 8'h07;
parameter PhysicalAddress_3 = 8'h95;
parameter PhysicalAddress_4 = 8'h0B;
parameter PhysicalAddress_5 = 8'hFB;
parameter PhysicalAddress_6 = 8'hAF;

////////////////////////////////////
////
// надсилає пакет приблизно кожну секунду
reg [23:0] counter; always @(posedge clk20) counter<=counter+1;
reg StartSending; always @(posedge clk20) StartSending<=&counter;

////////////////////////////////////
////
// ми надсилаємо UDP-пакет, 18 байт навантаження

// calculate the IP checksum, big-endian style
parameter IPchecksum1 = 32'h0000C53F +
(IPsource_1<<8)+IPsource_2+(IPsource_3<<8)+IPsource_4+

```

```

(IPdestination_1<<8)+IPdestination_2+(IPdestination_3<<8)+(IPdesti
nation_4);
parameter                                IPchecksum2                =
((IPchecksum1&32'h0000FFFF)+(IPchecksum1>>16));
parameter                                IPchecksum3                =
~((IPchecksum2&32'h0000FFFF)+(IPchecksum2>>16));

reg [6:0] rdaddress;
reg [7:0] pkt_data;

always @(posedge clk20)
case(rdaddress)
// Ethernet preamble
7'h00: pkt_data <= 8'h55;
7'h01: pkt_data <= 8'h55;
7'h02: pkt_data <= 8'h55;
7'h03: pkt_data <= 8'h55;
7'h04: pkt_data <= 8'h55;
7'h05: pkt_data <= 8'h55;
7'h06: pkt_data <= 8'h55;
7'h07: pkt_data <= 8'hD5;
// Ethernet header
7'h08: pkt_data <= PhysicalAddress_1;
7'h09: pkt_data <= PhysicalAddress_2;
7'h0A: pkt_data <= PhysicalAddress_3;
7'h0B: pkt_data <= PhysicalAddress_4;
7'h0C: pkt_data <= PhysicalAddress_5;
7'h0D: pkt_data <= PhysicalAddress_6;
7'h0E: pkt_data <= 8'h00;
7'h0F: pkt_data <= 8'h12;
7'h10: pkt_data <= 8'h34;
7'h11: pkt_data <= 8'h56;
7'h12: pkt_data <= 8'h78;
7'h13: pkt_data <= 8'h90;
// IP header
7'h14: pkt_data <= 8'h08;
7'h15: pkt_data <= 8'h00;
7'h16: pkt_data <= 8'h45;
7'h17: pkt_data <= 8'h00;
7'h18: pkt_data <= 8'h00;
7'h19: pkt_data <= 8'h2E;
7'h1A: pkt_data <= 8'h00;
7'h1B: pkt_data <= 8'h00;
7'h1C: pkt_data <= 8'h00;
7'h1D: pkt_data <= 8'h00;
7'h1E: pkt_data <= 8'h80;
7'h1F: pkt_data <= 8'h11;
7'h20: pkt_data <= IPchecksum3[15:8];
7'h21: pkt_data <= IPchecksum3[ 7:0];
7'h22: pkt_data <= IPsource_1;
7'h23: pkt_data <= IPsource_2;
7'h24: pkt_data <= IPsource_3;

```

```

7'h25: pkt_data <= IPsource_4;
7'h26: pkt_data <= IPdestination_1;
7'h27: pkt_data <= IPdestination_2;
7'h28: pkt_data <= IPdestination_3;
7'h29: pkt_data <= IPdestination_4;
// UDP header
7'h2A: pkt_data <= 8'h04;
7'h2B: pkt_data <= 8'h00;
7'h2C: pkt_data <= 8'h04;
7'h2D: pkt_data <= 8'h00;
7'h2E: pkt_data <= 8'h00;
7'h2F: pkt_data <= 8'h1A;
7'h30: pkt_data <= 8'h00;
7'h31: pkt_data <= 8'h00;
// payload
7'h32: pkt_data <= 8'h00; // вставте сюда дані, які хочете
надіслати
7'h33: pkt_data <= 8'h01; // вставте сюда дані, які хочете
надіслати
7'h34: pkt_data <= 8'h02; // вставте сюда дані, які хочете
надіслати
7'h35: pkt_data <= 8'h03; // вставте сюда дані, які хочете
надіслати
7'h36: pkt_data <= 8'h04; // вставте сюда дані, які хочете
надіслати
7'h37: pkt_data <= 8'h05; // вставте сюда дані, які хочете
надіслати
7'h38: pkt_data <= 8'h06; // вставте сюда дані, які хочете
надіслати
7'h39: pkt_data <= 8'h07; // вставте сюда дані, які хочете
надіслати
7'h3A: pkt_data <= 8'h08; // вставте сюда дані, які хочете
надіслати
7'h3B: pkt_data <= 8'h09; // вставте сюда дані, які хочете
надіслати
7'h3C: pkt_data <= 8'h0A; // вставте сюда дані, які хочете
надіслати
7'h3D: pkt_data <= 8'h0B; // вставте сюда дані, які хочете
надіслати
7'h3E: pkt_data <= 8'h0C; // вставте сюда дані, які хочете
надіслати
7'h3F: pkt_data <= 8'h0D; // вставте сюда дані, які хочете
надіслати
7'h40: pkt_data <= 8'h0E; // вставте сюда дані, які хочете
надіслати
7'h41: pkt_data <= 8'h0F; // вставте сюда дані, які хочете
надіслати
7'h42: pkt_data <= 8'h10; // вставте сюда дані, які хочете
надіслати
7'h43: pkt_data <= 8'h11; // вставте сюда дані, які хочете
надіслати
default: pkt_data <= 8'h00;
endcase

```

```

reg [3:0] ShiftCount;
reg SendingPacket;
always @(posedge clk20) if(StartSending) SendingPacket<=1; else
if(ShiftCount==14 && rdaddress==7'h48) SendingPacket<=0;
always @(posedge clk20) ShiftCount <= SendingPacket ? ShiftCount+1
: 15;
wire readram = (ShiftCount==15);
always @(posedge clk20) if(ShiftCount==15) rdaddress <=
SendingPacket ? rdaddress+1 : 0;
reg [7:0] ShiftData; always @(posedge clk20) if(ShiftCount[0])
ShiftData <= readram ? pkt_data : {1'b0, ShiftData[7:1]};

// створення CRC32
reg [31:0] CRC;
reg CRCflush; always @(posedge clk20) if(CRCflush) CRCflush <=
SendingPacket; else if(readram) CRCflush <= (rdaddress==7'h44);
reg CRCinit; always @(posedge clk20) if(readram) CRCinit <=
(rdaddress==7);
wire CRCinput = CRCflush ? 0 : (ShiftData[0] ^ CRC[31]);
always @(posedge clk20) if(ShiftCount[0]) CRC <= CRCinit ? ~0 :
({CRC[30:0],1'b0} ^ ({32{CRCinput}} & 32'h04C11DB7));

// створення NLP
reg [17:0] LinkPulseCount; always @(posedge clk20) LinkPulseCount <=
SendingPacket ? 0 : LinkPulseCount+1;
reg LinkPulse; always @(posedge clk20) LinkPulse <=
&LinkPulseCount[17:1];

// TP_IDL, shift-perictp та manchester кодер
reg SendingPacketData; always @(posedge clk20) SendingPacketData <=
SendingPacket;
reg [2:0] idlecount; always @(posedge clk20) if(SendingPacketData)
idlecount<=0; else if(~&idlecount) idlecount<=idlecount+1;
wire dataout = CRCflush ? ~CRC[31] : ShiftData[0];
reg qo; always @(posedge clk20) qo <= SendingPacketData ?
~dataout^ShiftCount[0] : 1;
reg qoe; always @(posedge clk20) qoe <= SendingPacketData | LinkPulse
| (idlecount<6);
reg Ethernet_TDp; always @(posedge clk20) Ethernet_TDp <= (qoe ? qo
: 1'b0);
reg Ethernet_TDm; always @(posedge clk20) Ethernet_TDm <= (qoe ? ~qo
: 1'b0);

endmodule

```

## Лістинг А.2 – Конфігурація fifo

```

library ieee;
use ieee.std_logic_1164.all;

package ctrl_fifo_config_pkg is
    component ctrl_fifo_config is
        generic (
            DATA_WIDTH           : integer:= 8; --! Data width
            ADDR_WIDTH            : integer:= 4 --! Address width =
log2(FIFO_DEPTH)
        );
        port (
            ---- global clear ----
            reset                  : in  std_logic; --! Reset FIFO
            ---- clocks ----
            wr_clk                 : in  std_logic; --! Write clock

            rd_clk                 : in  std_logic; --! Read clock

            ---- data ----
            data_i                  :                               in
std_logic_vector(DATA_WIDTH-1 downto 0); --! Data input
            data_o                  :                               out
std_logic_vector(DATA_WIDTH-1 downto 0); --! Data output
            ---- wr/rd ----
            rd_en                  : in  std_logic; --! Read enable
            wr_en                  : in  std_logic; --! Write enable

            ---- flags ----
            empty                   : out std_logic; --! Empty fifo
            full                    : out std_logic --! Full fifo
        );
    end component;
end package;

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity ctrl_fifo_config is
    generic (
        DATA_WIDTH           : integer:= 8; --! Data width
        ADDR_WIDTH            : integer:= 4 --! Address width =
log2(FIFO_DEPTH)
    );
    port (
        ---- global clear ----
        reset                  : in  std_logic; --! Reset FIFO
        ---- clocks ----
        wr_clk                 : in  std_logic; --! Write clock
        rd_clk                 : in  std_logic; --! Read clock
        ---- data ----

```

```

        data_i          : in  std_logic_vector(DATA_WIDTH-1 downto
0); --! Data input
        data_o          : out std_logic_vector(DATA_WIDTH-1 downto
0); --! Data output
        ---- wr/rd ----
        rd_en           : in  std_logic; --! Read enable
        wr_en           : in  std_logic; --! Write enable
        ---- flags ----
        empty           : out std_logic; --! Empty fifo
        full            : out std_logic; --! Full fifo
    );
end entity;

architecture ctrl_fifo_config of ctrl_fifo_config is

    ---- Internal connections & variables ----
    constant FIFO_DEPTH      : integer:=2**ADDR_WIDTH;

    type RAM is array (integer range <>) of std_logic_vector(DATA_WIDTH-
1 downto 0);
    signal Mem : RAM (0 to FIFO_DEPTH-1);

    signal pNextWordToWrite    : std_logic_vector(ADDR_WIDTH-1 downto
0);
    signal pNextWordToRead    : std_logic_vector(ADDR_WIDTH-1 downto
0);
    signal EqualAddresses     : std_logic;
    signal NextWriteAddressEn : std_logic;
    signal NextReadAddressEn  : std_logic;
    signal Set_Status         : std_logic;
    signal Rst_Status         : std_logic;
    signal Status             : std_logic;
    signal PresetFull         : std_logic;
    signal PresetEmpty        : std_logic;

    signal xempty, xfull      : std_logic;

    ---- Gray counter ----
    component rtl_gray_count is
        generic (
            COUNTER_WIDTH : integer:=4
        );
        port (
            cnt          : out  std_logic_vector(COUNTER_WIDTH-1 downto
0);
            ena          : in   std_logic; --! Count enable
            rst          : in   std_logic; --! Count reset
            clk          : in   std_logic; --! Input clock
        );
    end component;

begin

```

```

---- Read data ----
pr_rd: process(rd_clk) begin
    if (rising_edge(rd_clk)) then
        if (rd_en = '1' and xempty = '0') then
            data_o <= Mem(conv_integer(pNextWordToRead));
        end if;
    end if;
end process;

---- Write data ----
pr_wr: process (wr_clk) begin
    if (rising_edge(wr_clk)) then
        if (wr_en = '1' and xfull = '0') then
            Mem(conv_integer(pNextWordToWrite)) <= data_i;
        end if;
    end if;
end process;

---- Gray counters ----
xGray_WR : rtl_gray_count
    generic map ( COUNTER_WIDTH => ADDR_WIDTH )
    port map (
        cnt      => pNextWordToWrite,
        ena      => NextWriteAddressEn,
        rst      => reset,
        clk      => wr_clk
    );

xGray_RD : rtl_gray_count
    generic map ( COUNTER_WIDTH => ADDR_WIDTH )
    port map (
        cnt      => pNextWordToRead,
        ena      => NextReadAddressEn,
        rst      => reset,
        clk      => rd_clk
    );

---- R/W Address ----
NextWriteAddressEn <= wr_en and (not xfull);
NextReadAddressEn  <= rd_en and (not xempty);
---- Equal Address ----
EqualAddresses <= '1' when (pNextWordToWrite = pNextWordToRead) else
'0';

---- Quadrant selectors ----
process (pNextWordToWrite, pNextWordToRead)
    variable set_status_bit0 :std_logic;
    variable set_status_bit1 :std_logic;
    variable rst_status_bit0 :std_logic;
    variable rst_status_bit1 :std_logic;
begin
    set_status_bit0      :=      pNextWordToWrite(ADDR_WIDTH-2)      xnor
pNextWordToRead(ADDR_WIDTH-1);

```

```

    set_status_bit1      :=    pNextWordToWrite(ADDR_WIDTH-1)    xor
pNextWordToRead(ADDR_WIDTH-2);
    Set_Status <= set_status_bit0 and set_status_bit1;

    rst_status_bit0      :=    pNextWordToWrite(ADDR_WIDTH-2)    xor
pNextWordToRead(ADDR_WIDTH-1);
    rst_status_bit1      :=    pNextWordToWrite(ADDR_WIDTH-1)    xnor
pNextWordToRead(ADDR_WIDTH-2);
    Rst_Status          <= rst_status_bit0 and rst_status_bit1;
end process;

---- Status latch ----
process (Set_Status, Rst_Status, reset)
begin
    if (Rst_Status = '1' or reset = '1') then
        Status <= '0'; -- Empty
    elsif (Set_Status = '1') then
        Status <= '1'; -- Full
    end if;
end process;

---- Full ----
PresetFull <= Status and EqualAddresses;

process (wr_clk, PresetFull) begin
    if (PresetFull = '1') then
        xfull <= '1';
    elsif (rising_edge(wr_clk)) then
        xfull <= '0';
    end if;
end process;
full <= xfull;

---- Empty ----
PresetEmpty <= not Status and EqualAddresses;

process (rd_clk, PresetEmpty) begin
    if (PresetEmpty = '1') then
        xempty <= '1';
    elsif (rising_edge(rd_clk)) then
        xempty <= '0';
    end if;
end process;
empty <= xempty;

end ctrl_fifo_config;

```

### Лістинг А.3 – Вузол перепакування даних для FIFO АЦП

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;

package ctrl_repackNxM_pkg is

```

```

component ctrl_repackNxM is
  generic (
    CHAN_NUM    : integer:=4;
    DATA_WIN   : integer:=16;
    DATA_WOUT  : integer:=256
  );
  port (
    reset       : in  std_logic:='0'; --! 0 - global
reset
    aclk        : in  std_logic:='0'; --! Clock

    chan        : in  std_logic_vector(CHAN_NUM-1 downto
0):=(others=>'0'); --! Selected chan
    din         : in  std_logic_vector(DATA_WIN*CHAN_NUM-1 downto 0):=(others=>'0'); --!
DataIn from ADC
    den         : in  std_logic:='0'; --! Data enable

    dout        : out std_logic_vector(DATA_WOUT-1 downto 0
); --! Data out (packed)
    dval        : out std_logic    --! Valid (enable for
FIFO ADC)
  );
end component;
end package;

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_UNSIGNED.all;

entity ctrl_repackNxM is
  generic (
    CHAN_NUM    : integer:=4;
    DATA_WIN   : integer:=16;
    DATA_WOUT  : integer:=256
  );
  port (
    reset       : in  std_logic:='0'; --! 0 - global reset
    aclk        : in  std_logic:='0'; --! Clock

    chan        : in  std_logic_vector(CHAN_NUM-1  downto
0):=(others=>'0'); --! Selected chan
    din         : in  std_logic_vector(DATA_WIN*CHAN_NUM-1
downto 0):=(others=>'0'); --! DataIn from ADC
    den         : in  std_logic:='0'; --! Data enable

    dout        : out std_logic_vector(DATA_WOUT-1 downto 0 );
--! Data out (packed)
    dval        : out std_logic    --! Valid (enable for FIFO
ADC)
  );
end entity;

```

architecture ctrl\_repackNxM of ctrl\_repackNxM is

```

-----
-- wi/wo 64      128      256      512

-- 008      8      16      32      64
-- 012      4      8      16      32
-- 014      4      8      16      32
-- 016      4      8      16      32
-- 032      2      4      8      16
-- 064      1      2      4      8
-- 128      0      1      2      4
-----

function pack_dat (wi, wo: in integer) return integer is
variable nwi   : integer;
variable nwo   : integer;
begin
    case wi is
        when 000008 => nwi := 64;--8;
        when 000012 => nwi := 32;--4;
        when 000014 => nwi := 32;--4;
        when 000016 => nwi := 32;--4;
        when 000032 => nwi := 16;--2;
        when 000064 => nwi := 08;--1;
        when 000128 => nwi := 04;--0;
        when others => nwi := 00;--0;
    end case;
    nwo := (nwi * wo)/512;
    return nwo;
end;

-- 1 chan:
-----
-- wi/wo 64      128      256      512

-- 008      8      16      32      64
-- 016      4      8      16      32
-- 032      2      4      8      16
-- 064      1      2      4      8
-- 128      0      1      2      4
-----

constant DXMUX : integer:=pack_dat(DATA_WIN, DATA_WOUT);
constant DCHAN : integer:=DXMUX/CHAN_NUM;
-- Cannot pack chan = 1 / width = 128 into 64 bit fifo.
-- Cannot pack some channels into N width fifo.

signal dena          : std_logic;
signal dpack         : std_logic_vector(DATA_WOUT-1 downto
0):=(others=>'0');

begin

dval <= dena when rising_edge(aclk);

```

```

pr_out: process(aclk, reset) is
begin
    if rising_edge(aclk) then
        if (reset = '0') then
            dout <= (others => '0');
        elsif (dena = '1') then
            dout <= dpack;
        end if;
    end if;
end process;

xFAIL: if (DCHAN = 0) generate
    assert (DCHAN /= 0) report "CANNOT PACK DATA INTO FIFO"
severity error;
end generate;

xPASS: if (DCHAN /= 0) generate
    signal ddat      : std_logic;
    signal dmux      : std_logic_vector(DCHAN-1 downto 0);
    signal dsel      : std_logic_vector(DCHAN-1 downto 0);
    signal ddis      : std_logic_vector(DATA_WIN*CHAN_NUM-1
downto 0):=(others=>'0');
begin

    pr_mux: process(aclk, reset) is
begin
        if rising_edge(aclk) then
            if (reset = '0') then
                dmux <= (0 => '1', others => '0');
            elsif (den = '1') then
                dmux <= dmux(DCHAN-2 downto 0) & dmux(DCHAN-1);
            end if;
        end if;
    end process;

    pr_sel: process(aclk, reset) is
begin
        if rising_edge(aclk) then
            if (reset = '0') then
                dsel <= (others => '0');
            else
                dsel <= dmux;
            end if;
        end if;
    end process;
    ddat <= dmux(DCHAN-1) when rising_edge(aclk);

    xPACK: for ii in 0 to DCHAN-1 generate
        xDATA: for jj in 0 to CHAN_NUM-1 generate
            constant DCUT : integer:=(CHAN_NUM*ii)+jj;
        begin
            pr_dat: process(aclk, reset) is
begin

```

```

        if rising_edge(aclk) then
            if (dsel(ii) = '1') then
                dpack(DATA_WIN*(DCUT+1)-1          downto
DATA_WIN*DCUT) <= ddis(DATA_WIN*(jj+1)-1 downto jj*DATA_WIN);
            end if;
        end if;
    end process;
end generate;

xSELECT: for ii in 0 to CHAN_NUM-1 generate
    pr_dat: process(aclk, reset) is
    begin
        if rising_edge(aclk) then
            if (chan(ii) = '1') then
                ddis(DATA_WIN*(ii+1)-1 downto DATA_WIN*ii)
<= din(DATA_WIN*(ii+1)-1 downto DATA_WIN*ii);
            else
                ddis(DATA_WIN*(ii+1)-1 downto DATA_WIN*ii)
<= (others=>'0');
            end if;
        end if;
    end process;
end generate;
dena <= ddat when rising_edge(aclk);

-- pr_chan: process(aclk, reset) is
-- begin
--     if rising_edge(aclk) then
--         if (reset = '0') then
--             nchan <= (0 => '1', others => '0');
--         else
--             nchan <= nchan(CHAN_NUM-2 downto 0) & nchan;
--         end if;
--     end if;
-- end process;

end generate;

end ctrl_repackNxM;

```

#### Лістинг А.4 – Вузол перепакування даних для FIFO АЦП

```

library ieee;
use ieee.std_logic_1164.all;

package ctrl_spi_adc_pkg is
    component ctrl_spi_adc is
        port(
            clk           : in  std_logic; --! Data clock
            -- sys_clk     : in  std_logic; --! System clock
            reset         : in  std_logic; --! Global reset

```

```

        mg_addr      : in  std_logic_vector(07 downto 0); -
--! Host address
        mg_data_i : in  std_logic_vector(15 downto 0); --!
Host data (in)
        mg_data_o : out std_logic_vector(15 downto 0); --!
Host data (out)
        mg_data_v : out std_logic; --! Host data valid to out
        mg_start  : in  std_logic; --! Host data start
transaction
        mg_ready  : out std_logic; --! Host ready to
send/receive data

        spi_do      : out std_logic; --! SPI: Data out
        spi_di      : in  std_logic; --! SPI: Data in
        spi_ck      : out std_logic; --! SPI: Clock data
        spi_cs      : out std_logic  --! SPI: Chip
select (enable)
    );
    end component;
end package;

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

use work.ctrl_clk_prescaller_pkg.all;

entity ctrl_spi_adc is
    port(
        clk          : in  std_logic; --! Data clock
        -- sys_clk    : in  std_logic; --! System clock
        reset        : in  std_logic; --! Global reset

        mg_addr      : in  std_logic_vector(07 downto 0); --!
Host address
        mg_data_i : in  std_logic_vector(15 downto 0); --! Host
data (in)
        mg_data_o : out std_logic_vector(15 downto 0); --! Host
data (out)
        mg_data_v : out std_logic; --! Host data valid to out
        mg_start  : in  std_logic; --! Host data start transaction
        mg_ready  : out std_logic; --! Host ready to send/receive
data

        spi_do      : out std_logic; --! SPI: Data out
        spi_di      : in  std_logic; --! SPI: Data in
        spi_ck      : out std_logic; --! SPI: Clock data
        spi_cs      : out std_logic  --! SPI: Chip select
(enable)
    );
end ctrl_spi_adc;

```

```

architecture ctrl_spi_adc of ctrl_spi_adc is

    signal    clkz0, clkz1, clkz2, clkz3          : std_logic;
    signal    z2z3, z3z2                          : std_logic;
    signal    rstp                                : std_logic;

    signal    ready                                : std_logic;

    signal    cnt                                  : std_logic_vector(05 downto 00);
    signal    reg_out                             : std_logic_vector(23 downto 00);
    signal    reg_int                             : std_logic_vector(23 downto 00);
    signal    data_i                              : std_logic;

    type      stp_type is ( s0, s1, s2, s3, s4, s5, s6, s7 );
    signal    stp                                  : stp_type;

    signal    sclk_s                              : std_logic;
    signal    sdata                              : std_logic;
    signal    csn                                : std_logic;
    signal    clk_c                              : std_logic;

    signal    data_o                              : std_logic_vector(15 downto 00);
    signal    data_v                              : std_logic;
    signal    data_z                              : std_logic;

begin

    xCLK_DIVIDE : ctrl_clk_prescaller
        generic map(
            CLKIN  => 200000,
            CLKOUT => 010000
        )
        port map(
            clk_in  => clk,
            clk_out => clk_c
        );

    clkz0 <= clk_c when rising_edge(clk);
    clkz1 <= clkz0 when rising_edge(clk);
    clkz2 <= clkz1 when rising_edge(clk);
    clkz3 <= clkz2 when rising_edge(clk);

    rstp <= not reset when rising_edge(clk);

    z2z3 <= (not clkz2) and clkz3 when rising_edge(clk);
    z3z2 <= (not clkz3) and clkz2 when rising_edge(clk);

    pr_stp: process(clk) is
    begin
        if (rising_edge(clk)) then
            case (stp) is
                when s0 =>

```

```

        sclk_s    <= '0';
        ready     <= '1';
        cs_n      <= '1';
        if (mg_start = '1') then
            stp <= s1 ;
        end if;
    when s1 =>
        ready <= '0' ;
        if (z2z3 = '1') then
            stp <= s2;
        end if;
    when s2 =>
        if (z2z3 = '1') then
            stp <= s3;
            cs_n <= '0';
        end if;
    when s3 =>
        if (z3z2 = '1') then
            sclk_s <= not sclk_s;
            stp <= s4;
        end if;
    when s4 =>
        if (z2z3 = '1') then
            sclk_s <= not sclk_s;
            stp <= s5;
        end if;
    when s5 =>
        if (cnt(5) = '1') then
            stp <= s6;
        else
            stp <= s3;
        end if;
    when s6 =>
        if (z3z2 = '1') then
            cs_n <= '1';
            stp <= s7;
        end if;
    when s7 =>
        if (z2z3 = '1') then
            stp <= s0;
        end if;
    end case;
    if (rstp = '1') then
        stp <= s0;
    end if;
end if;
end process;

pr_reg_out: process(clk) is
begin
    if (rising_edge(clk)) then
        if (stp = s1) then

```

```

                                reg_out(07 downto 00) <= mg_data_i(15 downto 08);--
DATA                                reg_out(15 downto 08) <= mg_data_i(07 downto 00);--
DATA                                reg_out(23 downto 16) <= mg_addr(07 downto 00);  --
ADDR

                                cnt <= "001001";
                                elsif (stp = s5) then
                                    reg_int  <= reg_int( 22 downto 0 ) & data_i;
                                    reg_out <= reg_out( 22 downto 0 ) & '0';
                                    cnt <= cnt + 1;
                                end if;
                                if (stp = s1) then
                                    data_o <= (others => '0');
                                    data_v <= '0';
                                elsif (stp = s6) then
                                    data_o <= reg_int(15 downto 0);
                                    data_v <= '1';
                                end if;
                                end if;
end process;

data_z <= data_v when rising_edge(clk);
mg_data_v <= (data_v and not data_z) when rising_edge(clk);
mg_data_o(15 downto 08) <= data_o(07 downto 00) when
rising_edge(clk);
mg_data_o(07 downto 00) <= data_o(15 downto 08) when
rising_edge(clk);
sdata <= reg_out(23);

spi_ck <= sclk_s    when rising_edge(clk);
spi_do <= sdata      when rising_edge(clk);
spi_cs <= csn        when rising_edge(clk);
data_i <= spi_di     when rising_edge(clk);

mg_ready <= ready when rising_edge(clk);

end ctrl_spi_adc;

```

### Лістинг А.5 – Скрипт для перевірки роботи БПФ в Matlab

```

set(0, 'DefaultAxesFontSize', 14, 'DefaultAxesFontName', 'Times New
Roman');
set(0, 'DefaultTextFontSize', 14, 'DefaultTextFontName', 'Times New
Roman');
NFFT = 2^10;
Asig = 32767;
Fsig = 1;
B = 0.85;
% For testing FORWARD and INVERSE FFT: FWT
STAGE = log2(NFFT);

```

```

F = 2;
for i = 0:NFFT
    Dre(i+1,1) = Asig * cos(Fsig*i* 2*pi/NFFT);
    Dim(i+1,1) = Asig * sin(Fsig*i* 2*pi/NFFT);
    Dre(i+1,1) = round(Asig * cos((Fsig*i + B*i*i/2) * 2*pi/NFFT) *
sin(i * pi /
NFFT));
    Dim(i+1,1) = round(Asig * sin((Fsig*i + B*i*i/2) * 2*pi/NFFT) *
sin(i * pi /
NFFT));
end
% Adding noise to real signal
SNR = -10;
DatRe = awgn(Dre, SNR, 0, 1);
86
DatIm = awgn(Dim, SNR, 0, 1);
Mre = max(abs(DatRe));
Mim = max(abs(DatIm));
Mdt = max(Mre, Mim);
DSVRe = round(((2^15 - 1)/Mdt)*DatRe);
DSVIm = round(((2^15 - 1)/Mdt)*DatIm);
DatIn(:,1) = DSVRe;
DatIn(:,2) = DSVIm;
% Save data to file
fid = fopen ("din_re.dat", "w");
for i = 1:NFFT
    fprintf(fid, "%d \n", DSVRe(i,1));
end
fclose(fid);
fid = fopen ("din_im.dat", "w");
for i = 1:NFFT
    fprintf(fid, "%d \n", DSVIm(i,1));
end
fclose(fid);
DatX = DSVRe + 1j*DatIm;
DatFFT = fft(DatX);
DtFFT(:,1) = real(DatFFT);
87
DtFFT(:,2) = imag(DatFFT);
figure(1) % Plot loaded data in Time Domain
for i = 1:2
    subplot(2,1,1)
    plot(DatIn(1:NFFT,i), '-', 'LineWidth', 1, 'Color',[2-i 0 i-1])
    grid on; hold on; axis tight;
    title(['Входной сигнал'])
    subplot(2,1,2)
    plot(DtFFT(1:NFFT,i), '-', 'LineWidth', 1, 'Color',[2-i 0 i-1])
    grid on; hold on; axis tight;
    title(['результат обработки БПФ'])
end

```

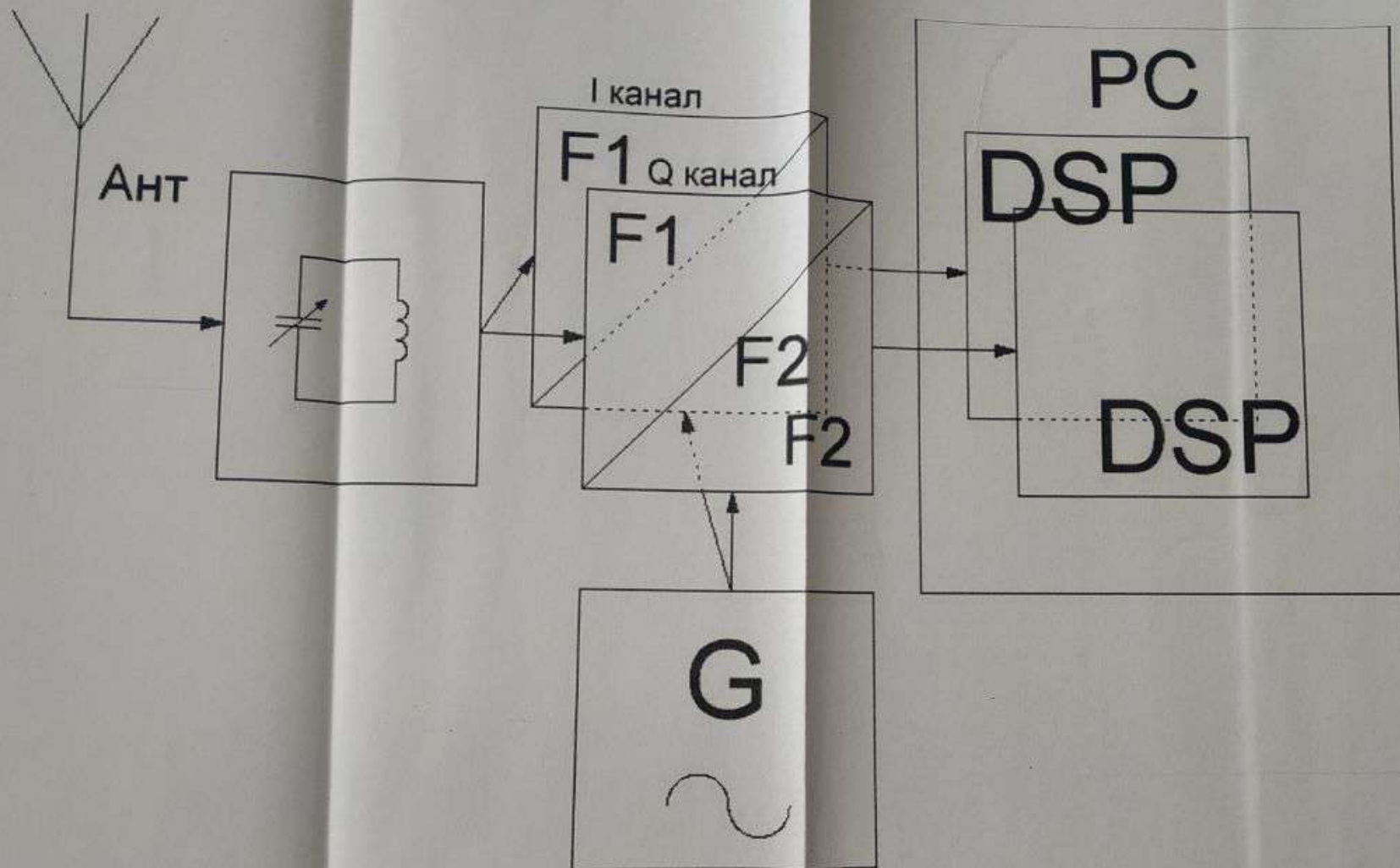




|          |       |                |               |      |                                                          |        |   |  |  |
|----------|-------|----------------|---------------|------|----------------------------------------------------------|--------|---|--|--|
|          |       |                |               |      | 13.02070849.00020 ПЛ1                                    |        |   |  |  |
|          |       |                |               |      | Конфігуруєма радіосистема на ПЛС                         |        |   |  |  |
|          |       |                |               |      | Функціональна схема SDR приймача супергетеродинного типу |        |   |  |  |
| Зам.     | Лист. | № докум.       | Підп.         | Дата | Лист                                                     | Маса   | М |  |  |
| Розроб.  |       | Неміський М.І. | М.І.Неміський |      |                                                          |        |   |  |  |
| Перевір. |       | Грушко С.С.    | С.С.Грушко    |      |                                                          |        |   |  |  |
| Т.контр. |       |                |               |      | Лист 1                                                   | Листів |   |  |  |
| Н-зам.   |       | Шербак Н.В.    | Н.В.Шербак    |      | НУ «Запорізька політехнічна школа»<br>КНТ-3-511М         |        |   |  |  |



|                                                        |       |                 |                    |      |          |
|--------------------------------------------------------|-------|-----------------|--------------------|------|----------|
| 13.02070849.00020 ПЛ2                                  |       |                 |                    |      |          |
| Конфігуруєма радіосистема на ПЛС                       |       |                 |                    |      |          |
| Функціональна схема SDR приймача прямого перетворення. |       |                 |                    |      |          |
| Зам.                                                   | Лист. | № докум.        | Підп.              | Дата |          |
| Розроб.                                                |       | Немітський М.І. | <i>[Signature]</i> |      |          |
| Перевір.                                               |       | Грушко С.С.     | <i>[Signature]</i> |      |          |
| Ухвал.                                                 |       |                 |                    |      |          |
| Н. контр.                                              |       | Шербак Н.В.     | <i>[Signature]</i> |      |          |
| Затверд.                                               |       | Кудерметов Р.К. | <i>[Signature]</i> |      |          |
|                                                        |       |                 |                    |      | Лист 1   |
|                                                        |       |                 |                    |      | Листів 1 |
| НУ «Запорізька політехніка» КНТУ-511М                  |       |                 |                    |      |          |



|          |                 |          |       |      |  |                                                                                                |        |          |        |
|----------|-----------------|----------|-------|------|--|------------------------------------------------------------------------------------------------|--------|----------|--------|
|          |                 |          |       |      |  | 13.02070849.00020 ПЛЗ                                                                          |        |          |        |
|          |                 |          |       |      |  | Конфігуруєма радіосистема на ПЛС<br><br>Структурна схема SDR приймача<br>прямого перетворення. | Лист.  | Маса     | Масшт. |
| Зам.     | Лист.           | № докум. | Шлях. | Дата |  |                                                                                                |        |          |        |
| Розроб.  | Неміський М.І.  |          |       |      |  |                                                                                                |        |          |        |
| Перевір. | Грушко С.С.     |          |       |      |  |                                                                                                |        |          |        |
| Т.контр. |                 |          |       |      |  |                                                                                                | Лист 1 | Листів 1 |        |
| Н.контр. | Шербак Н.В.     |          |       |      |  | НУ «Запорізька політехніка»<br>КНТЗ-511М                                                       |        |          |        |
| Затверд. | Кудеристов Р.К. |          |       |      |  |                                                                                                |        |          |        |