Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інфокомунікацій, радіоелектроніки та наносистем
(повне найменування інституту, назва факультету (відділення))

Кафедра радіотехніки
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Багатоканальна вимірювальна система на FPGA для радіовимірювальних частотних сенсорів»

Виконав: студент 2-го курсу, групи РТ-20м
спеціальності 172 – Телекомунікації та радіотехніка, ОП – Радіотехніка
(шифр і назва напряму підготовки, спеціальності)
Скощук В. К.
(прізвище та ініціали)

Керівник: д.т.н., професор каф. РТ
Осадчук О.В.
(прізвище та ініціали)
«____» _____ 2021 р.

Опонент: к.т.н., професор каф. ТКСТБ
Бортник Г.Г.
(прізвище та ініціали)
«____» _____ 2021 р.

Допущено до захисту
Завідувач кафедри РТ
д.т.н., професор О.В. Осадчук
(прізвище та ініціали)
«____» _____ 2021 р.

Вінниця ВНТУ - 2021 рік

Вінницький національний технічний університет

Факультет інфокомунікацій , радіоелектроніки та наносистем
Кафедра Радіотехніки
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 17 – Електроніка та телекомунікації
Спеціальність – 172 – Телекомунікації та радіотехніка
Освітньо-професійна програма – Радіотехніка

ЗАТВЕРДЖУЮ
Завідувач кафедри РТ
д.т.н., професор О. В. Осадчук
“___” _____2021 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Скощуку Валентину Костянтиновичу .
(прізвище, ім’я, по батькові)

1. Тема роботи. «Багатоканальна вимірювальна система на FPGA для радіовимірювальних_частотних_сенсорів».
керівник роботи Осадчук Олександр Володимирович, д.т.н., професор
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від _____ року №
2. Строк подання студентом роботи 20 грудня 2021 року.
3. Вихідні дані до роботи:
Багатоканальна вимірювальна система на основі FPGA Cyclone IV EP4CE10F17C8 компанії Altera, напруга живлення 5В, амплітуда вхідного сигналу від 2.5...5В, діапазон вимірюваних частот від 10Гц до 10МГц. Універсальний вимірювальний прилад, який має 12 вимірювальних каналів для сенсорів з частотним виходом і підтримує одночасну роботу з 127 цифровими сенсорами через I2C інтерфейс. У якості вихідного інтерфейсу використовується цифровий протокол UART.
4. Зміст текстової частини:
1) Аналіз сучасного стану розвитку багатоканальних частотомірів. 2)Розробка багатоканальної вимірювальної системи на FPGA. 3) Розширення функціональних можливостей приладу. 4) Розробка програмного комплексу для взаємодії з приладом. 5) Охорона праці та безпека в надзвичайних ситуаціях. 6) Економічна частина.
5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень):
1) Генератор імпульсів керування. 2) Лічильник імпульсів. 3) Багатоканальна вимірювальна система зчитування значень із сенсорів з частотним виходом. 4) Відладочна плата CoreEP4CE10. 5) Міжблочне з’єднання мікропроцесорної систем. 6) Узагальнена мікропроцесорна система.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Основна частина	д.т.н., професор Осадчук О.В.		
Охорона праці та безпека в надзвичайних ситуаціях	професор кафедри БЖД ПБ доцент, д.п.н., Дембіцька С. В.		
Економічна частина	доцент кафедри ЕПВМ к.е.н., Кавецький В. В.		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	**Примітка**
1.	Огляд літературних джерел. Вибір та узгодження МКР.	03.09.2021 - 14.09.2021	
2.	Аналіз літературних джерел. Попередня розробка основних розділів.	15.09.2021 - 21.09.2021	
3.	Затвердження теми. Розробка технічного завдання.	21.09.2021 - 25.09.2021	
4.	Аналіз вирішення поставленої задачі. Розробка структурної схеми.	26.09.2021 - 09.10.2021	
5.	Електричні розрахунки. Експериментальне дослідження.	10.10.2021 - 03.11.2021	
6.	Розділ моделювання	04.11.2021 - 12.11.2021	
7.	Розробка графічної частини МКР	13.11.2021 - 18.11.2021	
8.	Аналіз економічної ефективності розробки	19.11.2021 - 24.11.2021	
9.	Охорона праці (ОП)	25.11.2021 - 30.11.2021	
10.	Оформлення пояснювальної записки та графічної частини	01.12.2021 - 08.12.2021	
11.	Нормконтроль	09.12.2021 - 12.12.2021	
12.	Попередній захист МКР, доопрацювання рецензування МКР	13.12.2021 - 19.12.2021	
13.	Захист МКР ЕК	21.12.2021 - 23.12.2021	

Студент _____ Скощук В. К.
(підпис)

Керівник роботи _____ Осадчук О. В.
(підпис)

АНОТАЦІЇ

УДК 621.374.415

Скощук В. К. Багатоканальна радіовимірювальна система на FPGA для радіовимірювальних частотних сенсорів. Магістерська кваліфікаційна робота з спеціальності «172 – Телекомунікації та радіотехніка, ОП – Радіотехніка». – Вінниця: ВНТУ, 2021. – 180 с., 80 рис., 45 бібл., 18 табл. – українською мовою.

У даній МКР проведено розробку системи на основі FPGA з можливістю вимірювання частоти і підтримкою сучасних цифрових інтерфейсів сенсорів фізичних величин.

Розроблено багатоканальний вимірювальний прилад на основі FPGA фірми Altera Cyclone IV, який має 12 вимірювальних каналів для сенсорів з частотним виходом і підтримує одночасну роботу з 127 цифровими сенсорами через I2C інтерфейс. У якості вихідного інтерфейсу використовується широко розповсюджений цифровий протокол UART, який підтримується великою кількістю конверторів. Тому, теоретично, передачу даних з розробленого пристрою можна здійснювати і безпровідним шляхом. До ПК пристрій можна під’єднати через конвертор UART-USB.

Розроблено спеціалізоване програмне забезпечення для перевірки працездатності багатоканальної вимірювальної системи. Для зручності сприйняття, відбувається візуалізація отриманої інформацію від вимірювального приладу.

У п'ятому розділі описано рекомендації щодо охорони праці та безпеки при роботі з даним пристроєм.

У шостому розділі проведено розрахунок кошторису витрат на виробництво пристрою та ефективність вкладених інвестицій.

Ключові слова: *Багатоканальний радіовимірювальний прилад, сенсори фізичних величин з частотним виходом, сенсори фізичних величин з цифровим виходом, FPGA, частотомір, вимірювальні канали, Nios II, I2C.*

ABSTRACT

Skoshchuk V. K. Multichannel radio measuring system on FPGA for radio measuring frequency sensors: master's qualification work in the specialty "172 - Telecommunications and radio engineering, educational program - Radio engineering". - Vinnytsia: VNTU, 2021. - 180 pp., 80 figs., 45 bibl., 18 tables. - in Ukrainian.

In this master's degree the system based on FPGA with the ability to measure frequency and support modern digital interfaces of sensors of physical quantities.

An Altera Cyclone IV FPGA-based multi-channel measuring instrument has been developed, which has 12 measuring channels for frequency output sensors and supports simultaneous operation with 127 digital sensors via the I2C interface. The source interface is the widespread digital UART protocol, which is supported by a large number of converters. Therefore, theoretically, data transmission from the developed device can be carried out wirelessly. The device can be connected to a PC via a UART-USB converter.

Specialized software for checking the efficiency of a multi-channel measuring system has been developed. For convenience of perception, there is a visualization of the received information from the measuring device.

The fifth section describes the recommendations for health and safety when working with this device.

In the sixth section, the calculation of the cost of production of the device and the efficiency of the investment.

Key words: *Multichannel radio measuring device, sensors of physical quantities with frequency output, sensors of physical quantities with digital output, FPGA, frequency meter, measuring channels, Nios II, I2C.*

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ЕЛЧ – електронно-лічильні частотоміри
НВЧ – надвисокочастотний
ПМЧ – помножувач частоти
ГГ – генератора гармонік
ФВЧ – фільтр високих частот
ППЧ – підсилювач проміжної частоти
ФНЧ – фільтр нижніх частот
ППС – підсилювач постійного струму
FPGA – Програмована користувачем вентильна матриця
PLL – фазове автоналаштування частоти
RAM – random access memory
EEPROM – Electrically Erasable Programmable Read-Only Memory
CLB – настроюваний логічний блок
LE – логічний елемент
LUT – таблиця пошуку
CPLD – програмована логічна інтегральна схема
HBIC – надвеликі інтегральні схеми
ASIC – інтегральна схема для специфічного застосування
UART – універсальний асинхронний приймач/передавач
ALU– арифметико-логічний пристрій
GUI – графічний інтерфейс користувача
MMU – модуль керування пам'яттю
MPU – модуль захисту пам’яті
JTAG – Joint Test Action Group
IEEE – інститут інженерів з електротехніки та електроніки
I2C – послідовна шина даних для зв'язку інтегральних схем

ВСТУП

На даний час найбільш поширеними сенсорами фізичних величин являються сенсори з цифровими перетворювачами, які надають користувачу цифрові інтерфейси для отримання значень. Але паралельно з ними існують сенсори з радіовимірювальними перетворювачами, на основі реактивних властивостей напівпровідникових структур з від'ємним опором, які мають частотні виходи з повним збереженням у вихідному сигналі інформації про кількісне значення вимірюваної величини, через забезпечення функціональної залежності. Використання таких приладів виключає з їх конструкцій аналого-цифрові перетворювачі, що дозволяє знизити собівартість систем управління, а також створити «інтелектуальні» вимірювальні перетворювачі в результаті поєднання на одному кристалі схем обробки інформації та первинного перетворювача.

Розробками теорії та практичного застосування первинних перетворювачів в Україні займаються такі наукові центри як Вінницький національний технічний університет (м. Вінниця), Харківський національний технічний університет (м. Харків), Національний технічний університет України "КПІ" (м. Київ), Національний технічний університет "Львівська політехніка" (м. Львів), Інститут кібернетики НАН України (м. Київ), ВАТ "Український науково-дослідний інститут аналітичного приладобудування" України (м. Київ), Одеський національний університет імені І.І. Мечникова (м. Одеса), Київський національний університет імені Тараса Шевченко (м. Київ).

Найбільш відомими Українськими вченими які працюють над первинними перетворювачами є: Осадчука В.С., Осадчука О.В., Кухарчука В.В., Готри З.Ю., а також закордонних вчених Аша Ж., Віглеба Г., Новицького П.В., Стафеева В.І., Бутурліна А.І., Шаумбурга Г., Арутюняна В.М.та інші.

Актуальність теми

На даний час важко знайти широко доступні рішення які б задовільнили вимоги, щодо одночасного вимірювання значень сенсорів з цифровими і частотними виходами. Тому, існує необхідність розробки теоретичних підходів до створення такого роду систем, а також розробки схеми і конструкції.

Мета і задачі дослідження

Метою роботи є розробка багатоканальної вимірювальної системи яка одночасно зможе працювати із сенсорами принцип роботи яких базується на використанні функціональної залежності реактивних властивостей транзисторних структур з від'ємним опором і цифровими сенсорами фізичних величин.

Об'єктом дослідження є процес об'єднання вимірювань сенсорів з частотним і цифровим виходами.

Предметом дослідження – вимірювальна система на основі FPGA з можливістю вимірювання частоти і підтримкою сучасних цифрових інтерфейсів сенсорів фізичних величин.

Для досягнення поставленої мети у магістерській кваліфікаційній роботі розв'язуються такі *задачі*:

- проаналізувати існуючі методи вимірювання частоти і виділити найбільш перспективний для реалізації на FPGA;
- розробити багатоканальний радіовимірювальний прилад частоти на основі FPGA фірми Altera;
- розширити функціональні можливості розробленого приладу для підтримки сенсорів з цифровим перетворювачем;
- розробити спеціалізоване програмне забезпечення для тестування вимірювальної системи.
- виконати експериментальну перевірку розробленої багатоканальної вимірювальної системи використовуючи спеціалізоване програмне забезпечення.

Методи дослідження ґрунтуються на використанні:

- рівнянь математичної фізики під час аналізу математичних моделей частотомірів;

Наукова новизна одержаних результаті

Наукова новизна роботи полягає в отриманні наступних результатів:

1. Запропоновано використання FPGA для вимірювання величин сенсорів з частотним виходом.
2. Об`єднання різних підходів до вимірювання сенсорних величин в одному пристрої.

Практичне значення одержаних результатів

Практична цінність роботи полягає в тому, що:

1. Розроблено багатоканальний універсальний вимірювальний прилад на основі FPGA фірми Altera Cyclone IV, який має 12 вимірювальних каналів для сенсорів з частотним виходом і підтримує одночасну роботу з 127 цифровими сенсорами через I2C інтерфейс У якості вихідного інтерфейсу використовується широко розповсюджений цифровий протокол UART, який підтримується великою кількістю конверторів. Тому, теоретично, передачу даних з розробленого пристрою можна здійснювати і безпровідним шляхом. До ПК пристрій можна під'єднати через конвертор UART-USB.
2. Розроблено спеціалізоване програмне забезпечення для перевірки працездатності багатоканальної вимірювальної системи. Для зручності сприйняття, відбувається візуалізація отриманої інформацію від вимірювального приладу.

Особистий внесок здобувача

Основні положення і результати магістерської роботи отримані автором самостійно.

Апробація результатів

Результати досліджень, що викладені в дисертації, були апробовані на наукових конференціях, серед них:

1. L Науково-технічна конференція факультету інфокомунікацій, радіоелектроніки та наносистем (2021). Секція радіотехніка. Тема: Багатоканальна радіовимірювальна система на ПЛІС для частотних перетворювачів фізичних величин.
2. Сучасні проблеми інфокомунікацій, радіоелектроніки та наносистем (СПІРН-2021). Секція радіотехніка. Тема: Багатоканальний частотомір на FPGA для радіовимірювальної системи з частотними сенсорами.
3. Вісник Хмельницького національного університету (2021). Тема: Багатоканальний частотомір на програмованій логічній інтегральній схемі для радіовимірювальної системи з частотнимими сенсорами фізичних величин.

Структура і обсяг роботи

Магістерська кваліфікаційна робота складається зі вступу, 6 розділів, висновків, додатків та списку використаних джерел.

1 АНАЛІЗ СУЧАСНОГО СТАНУ РОЗВИТКУ БАГАТОКАНАЛЬНИХ ЧАСТОТОМІРІВ

1.1 Класифікація приладів для вимірювання частоти

Вихідна термінологія та одиниці вимірювання. Частотомір - радіовимірювальний прилад для визначення частоти періодичного процесу або частот гармонійних складових спектру сигналу.

Частотоміри можуть класифікуватися за декількома пунктами [1, 2]:

1. За методом вимірювання - прилади безпосередньої оцінки і прилади порівняння.
2. За фізичним змістом вимірюваної величини - вимірювання частоти синусоїдальних коливань, вимірювання частот гармонійних складових і вимірювання частоти дискретних подій.
3. За сферою застосування - електровимірювальні прилади і радіовимірювальні прилади.

Слід зауважити, що межа між сферами застосування приладів досить прозора:

1. До групи електровимірювальних приладів входять: аналогові стрілочні частотоміри різних систем, вібраційні, а також частково конденсаторні та електронно-лічильні частотоміри.
2. До групи радіовимірювальних приладів входять: резонансні, гетеродинні, конденсаторні та електронно-лічильні частотоміри.

За допомогою резонансних частотоміри, разом з перетворювачами механічних коливань в електричні, зазвичай вимірюється частота механічних коливань. За допомогою електромеханічних, електродинамічних, електронних, електромагнітних, магнітоелектричних частотомірів вимірюється частота електричних коливань. За допомогою електронних частотомірів (резонансні, гетеродинні, цифрові) вимірюється частота електромагнітних коливань в діапазоні радіочастот НВЧ.

1.1.1 Електронно-лічильний частотомір

Можливості вимірювання високих частот звичайними електронно-лічильними частотомірами (ЕЛЧ), що працюють за методом прямого рахунку,

обмежувались швидкодією існуючої елементної бази електроніки (подільників, формувачів, тригерів). Досягнення електроніки в створенні швидкодіючих елементів збільшили верхню межу частоти, вимірюваної методом прямого рахунку, до 10 ГГц. На більш високих частотах ЕЛЧ працюють за допомогою перетворювачів частоти, які бувають двох типів [3]: дискретні та перенесення частоти.

Принцип роботи ЕЛЧ з дискретним перетворювачем частоти показаний на рисунку 1.1. Перетворювач складається з помножувача частоти ПМЧ, генератора гармонік ГГ, фільтра ФВЧ, змішувача ЗМ, широкосмугового підсилювача проміжної частоти. Сигнал від кварцового генератора частотоміра ЕЛЧ подається на вхід помножувача ПМЧ, де формується частота $f_0 = 100$МГц. В ГГ з цієї частоти формується дискретний спектр гармонік $2f_0, 3f_0, \ldots nf_0$. За допомогою ФВЧ виділяється одна з гармонік nf_0, яка надходить на ЗМ, сюди ж надходить вимірюваний сигнал з частотою $f_г$. З спектра частот на виході змішувач ППЧ виділяє різницеву частоту $f_г - nf_0$. Ця частота вимірюється ЕЛЧ і індукується на його цифровому табло. Значення невідомої частоти знаходиться шляхом складання показань лічильникового пристрою ФВЧ і цифрового табло ЕЛЧ.

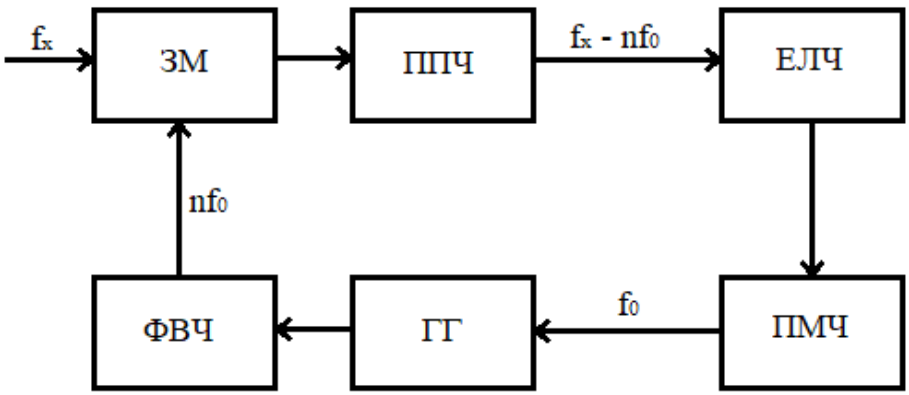


Рисунок 1.1–Схема частотоміра СВЧ з дискретним перетворювачем частоти

Подібні перетворювачі дозволяють вимірювати частоти до 12 ГГц. Похибка вимірювання частоти дорівнює похибці ЕЛЧ плюс похибка дискретності.

Принцип роботи ЕЛЧ з перенесенням частоти показаний на рисунку 1.2. На відміну від дискретного перетворювача даний тип має гетеродин, який перебудовується в широкому діапазоні частот, і систему фазового

автоналаштування частоти для синхронізації частот гетеродинна і вимірюваного сигналу.

В процесі вимірювання напруга від гетеродинна Гет подається на змішувач ЗМ, куди надходить сигнал невідомої частоти f. Підсилювач проміжної частоти ППЧ виділяє на виході змішувача частоту різниці $f = mf_x - nf_r$, яка надходить на фазовий детектор ФД, куди надходить частота f_0кварцового генератора ЕЛЧ Постійна напруга з виходу фазового детектора ФД, пропорційна різниці $f - f_0$, через фільтр ФНЧ і підсилювач ППС подається на елемент, керування частотою гетеродинна, завдяки чому частота гетеродинна змінюється таким чином, щоб різниця $f - f_0$ наближалася до нуля. Тим самим частота гетеродина синхронізується з частотою сигналу, при якому виконується рівність (1.1)

$$mf_x = nf_{r1} - f_{\text{пр}} \text{ або } mf_x = nf_{r2} - f_{\text{пр}}. \quad (1.1)$$

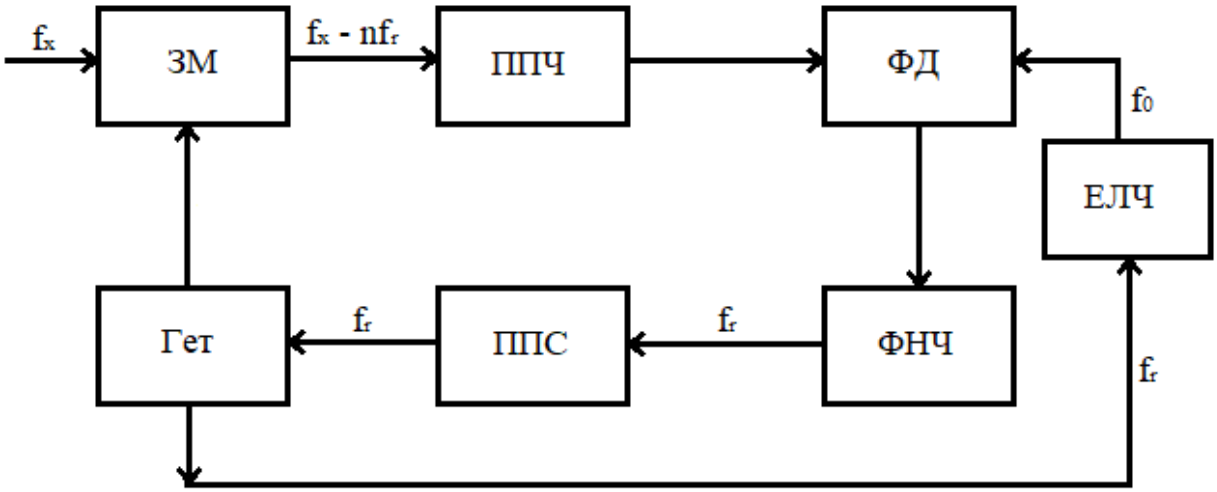


Рисунок 1.2 –Схема частотоміра СВЧ з переносником частоти

Частота гетеродинна вибирається такою, щоб вона вимірювалася частотоміром ЕЛЧ. Щоб визначити f_r, необхідно знайти номери гармонік n і m. Значення n визначають шляхом двох послідовних вимірювань частоти f_r. Номер гармоніки n визначають за формулою (1.2)

$$n = \frac{2f_{\text{пр}}}{f_{r2} - f_{r1}}. \quad (1.2)$$

Для знаходження m необхідно знати орієнтовне значення частоти f_x.

1.1.2 Резонансні частотоміри

Резонансні частотоміри працюють за методом порівняння вимірюваної частоти з резонансною частотою контуру [4]. Основним вузлом частотомірів НВЧ є резонатор. Вхід резонатора через спеціальний елемент зв'язку з'єднується з джерелом електромагнітних коливань, частоту яких необхідно виміряти, а через другий елемент зв'язку до резонатора підключається детектор, що перетворює коливання НВЧ в постійний струм, контрольований мікроамперметром. За допомогою механізму перебудови досягається збіг частоти резонатора з вимірюваною частотою. Момент збігу частот визначається по максимальному відхиленню стрілки мікроамперметра. Значення частоти знаходять за шкалою механізму перебудови резонатора.

На краю НВЧ-діапазону використовують в основному коаксіальні резонатори. Схематичне представлення резонансного частотоміра на основі півхвильового коаксіального резонатора показано на рисунку 1.3. Резонатор виконаний з відрізка коаксіальної лінії змінної довжини, закороченої з одного боку нерухомим *К*, а з іншого - рухомим замикачем *П*. Останній переміщається за допомогою мікрогвинта *М*. Для зв'язку резонатора з джерелом вимірюваної частоти передбачена петля зв'язку *С1*, детектор *Д* з мікроамперметром *А* підключені до резонатора через петлю *С2*.

При переміщенні замикача в такій лінії резонанс відбувається при виконанні умови (1.3)

$$l = \frac{n \times \lambda}{2}, \tag{1.3}$$

де l— відстань між короткозамикачами *Д* і *П*; λ— довжина хвилі в коаксіальної лінії, відповідна вимірюваній частоті f; n = 1, 2, 3...

Поряд з півхвильовимкоаксіальними резонаторами в частотомірах використовують чверть хвильові.

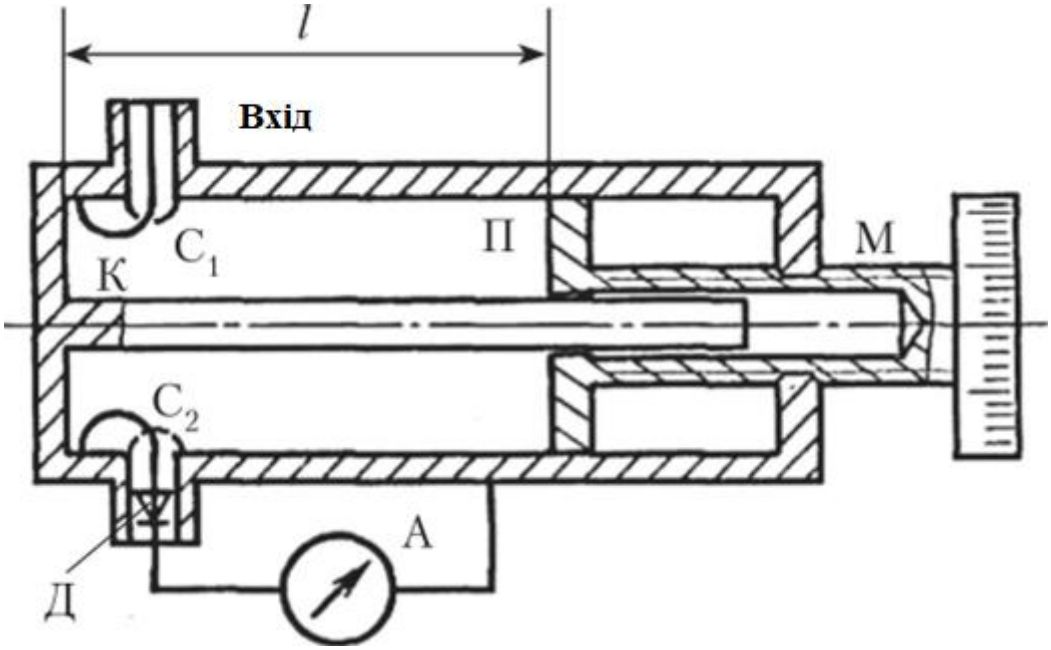


Рисунок 1.3–Схема коаксіального півхвильового резонатора

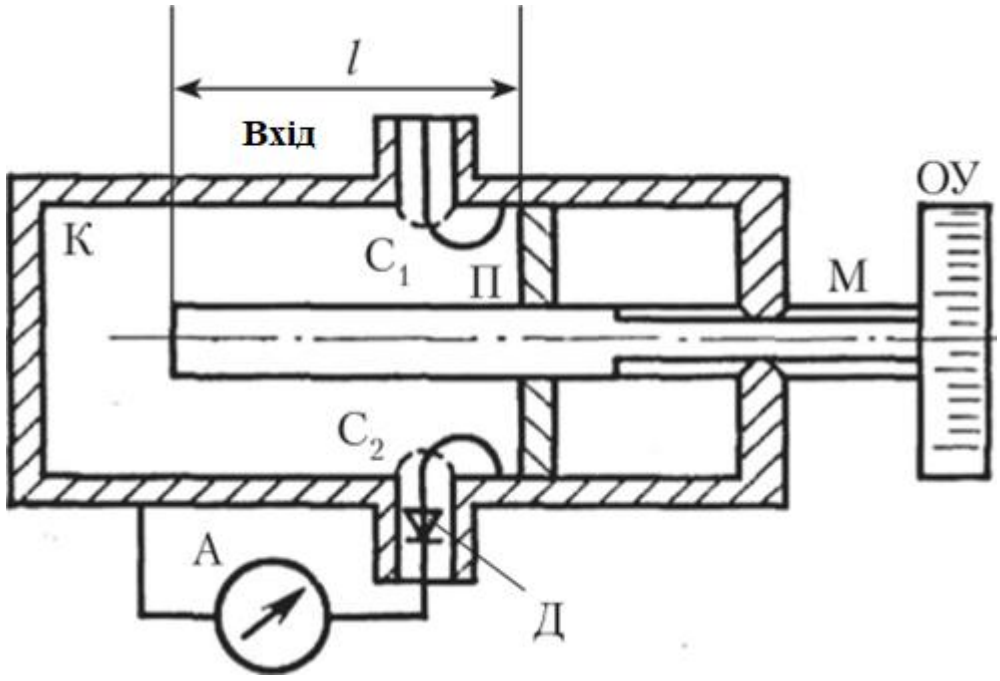


Рисунок 1.4 –Схема коаксіального четвертьхвильового резонатора

Основна відмінність їх від напівхвильових полягає в тому, що коаксіальна лінія з одного боку розімкнути, як показано на рисунку 1.4. Тут використовується той факт, що розімкнена лінія четвертьхвильової довжини еквівалентна

короткозамкненій напівхвильовій. Тому умова резонансу в такому коаксіальному резонаторі записується інакше (1.4)

$$l = \frac{(2n+1)\times\lambda}{4}, \tag{1.4}$$

де l— відстань від кінця центрального провідника коаксіальної лінії до замикача.

Частотоміри з четвертьхвильовими коаксіальними резонаторами застосовуються на частотах від 0,6 до 10 ГГц. На більш високих частотах в резонансних частотомірах застосовуються об'ємні резонатори. Вони являють собою відрізки круглого хвилеводу, закорочені з обох сторін. Один короткозамикачем зроблений рухомим, а механізм його переміщення (мікрогвинт) пов'язаний з відліковим пристроєм. Електромагнітні коливання підводяться до резонатора по прямокутному хвилеводу стандартного перетину і збуджують його через отвір зв'язку в нерухомому торці. Через другий подібний отвір енергія відводиться до хвилеводної детекторної секції. Струм детектора контролюється мікроамперметром. Залежно від розташування і форми отвору зв'язку в хвилеводному резонаторі можуть збуджуватися коливання вищого типу H_{11n} або H_{01n}. При збудженні коливань H_{11n} робочий діапазон частотоміра більше, ніж при коливаннях H_{01n}. Однак добротність резонатора з коливаннями H_{01n} вище і, отже, точніше визначається момент резонансу.

Джерелами похибок резонансних частотоміри є: невизначеність налаштування резонансу, що залежить від добротності резонатора і чутливості індикатора; люфт в механізмі переміщення; зміни геометричних розмірів резонатора при зміні температури; похибка градуювання лічильного пристрою.

Основні переваги резонансних частотомірів - простота, надійність і низька вартість, недолік - невисока точність.

1.1.3 Гетеродинні частотоміри

Принцип дії гетеродинних частотомірів заснований на порівнянні частоти вхідного сигналу з частотою допоміжного генератора змінної частоти (гетеродинна) за допомогою методу «нульового биття», порядок роботи аналогічний роботі з резонансними частотомірами [5]. Спрощена структурна схема гетеродинного частотоміра представлена на рисунку 1.5. Вона містить:

вхідний пристрій, кварцовий генератор, змішувач, гетеродин, підсилювач низької частоти і індикатор (нульового биття). Дія гетеродинного частотоміра зводиться до простого принципу: при встановленні ключа *К* в положення *1* виконують калібрування шкали гетеродинна; при положенні *2* - вимірювання частоти f_x, яка подається на вхідний пристрій [6].

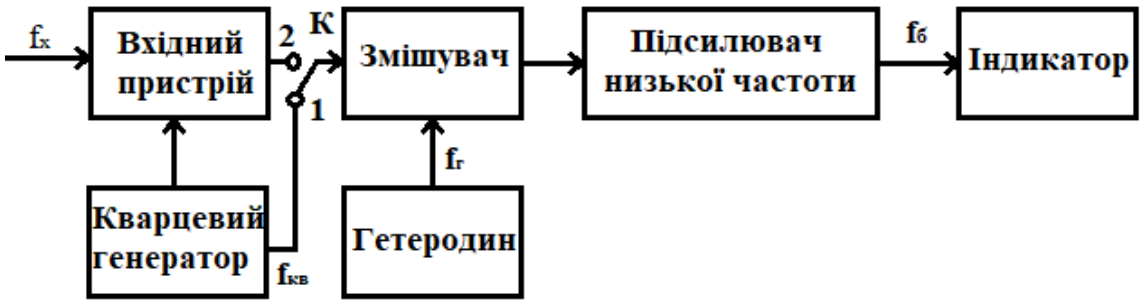


Рисунок 1.5 –Спрощена структурна схема гетеродинного частотоміра

Калібрування шкали гетеродинна здійснюють безпосередньо перед вимірюванням за допомогою додаткового кварцового генератора. Сигнал, що надходить з кварцового генератора, має складну форму і містить ряд гармонійних складових з кратними частотами: $f_{кв1}$, $f_{кв2}$, ..., $f_{кв n}$, де n - номер гармоніки. Ці частоти називають кварцовими точками. Відліковий лімб гетеродинна встановлюють в положення, відповідне найближчій до вимірюваної частоті f_xкварцової точці (приблизне значення вимірюваної частоти має бути відомо, інакше процес вимірювання дуже ускладнюється).

Сигнали з кварцового генератора $f_{кв}$ігетеродинна$f_г$ надходять на змішувач, тому на його виході виникають коливання з сумарними, різницевими і комбінаційними частотами. Індикатор фіксує наявність сигналу биття на мінімальній різницевій частоті $F_б = f_{кв} - f_г$ ∨, що проходить через підсилювач низької частоти (високочастотні складові, що виходять в результаті змішування частот кварцового генератора і гетеродинна, через підсилювач низької частоти не проходять). Змінюючи ємність конденсатора в контурі гетеродинна, отримують «нульові биття», отже, частота гетеродинна стає рівною частоті кварца$f_{кв} \approx f_г$. Після чого переходять до вимірювання невідомої частоти f_x, переводячи ключ *К* в положення *2*. Обертаючи відліковий лімб гетеродинна, домагаються «нульового

биття» і використовуючи відкориговану шкалугетеродинна визначають значення вимірюваної частоти $f_x \approx f_г$.

Гетеродинні частотоміри є досить точними вимірювальними приладами. Їх відносна похибка вимірювання лежить в межах 10^{-3}–10^{-5}. Однак в діапазоні середніх частот (до 300 МГц і нижче) їх витісняють електронно-лічильні частотоміри, які забезпечують ту ж високу точність, але значно простіше в експлуатації.

1.1.4 Конденсаторні частотоміри

Електронні конденсаторні частотоміри використовуються для вимірювання частоти періодичних напруг в діапазоні від 20 Гц до 500 кГц. Наведена похибка таких приладів зазвичай знаходиться в межах 1-2,5%. Принцип дії електронного конденсаторного частотоміра пояснюється схемою, наведеною на рисунку 1.6, і часовими діаграмами, наведеними на рисунку 1.7.

Напруга $u(t)$, частота якої змінюється (рисунки 1.6, 1.7), подається на вхід підсилювача-формувача *ПФ*, що підсилює вхідну напругу і формує з неї прямокутну напругу. Цією напругою керується схема електронного ключа *ЕК*. Передбачається, що при негативних сигналах *ЕК* розімкнений, а при позитивних замкнений. При розімкнутому стані ключа протягом половини періоду конденсатор *C* через R_Aзаряджається до значення *E*. Сила струму заряду $i_з$ протікає через діод D_1 При замиканні *ЕК* конденсатор *C* розряджається через замкнутий ключ, вимірювач *I* і діодD_2.

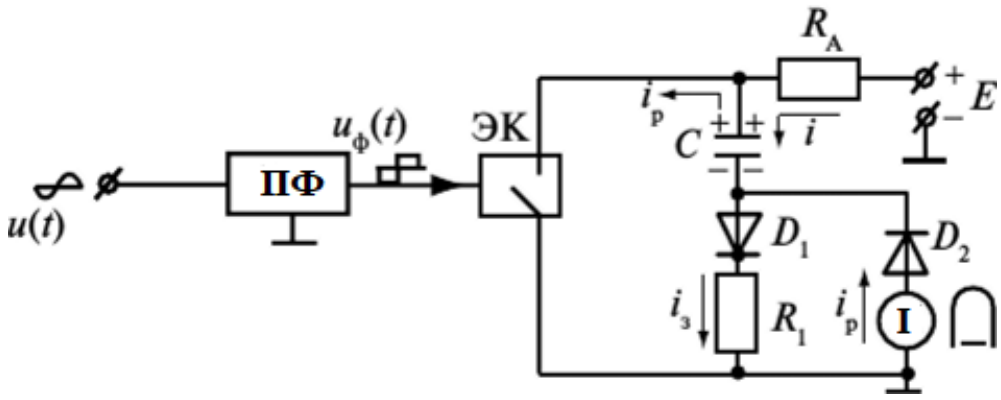


Рисунок 1.6 –Схема електронного конденсаторного частотоміра: *ПФ*– підсилювач-формувач; *ЕК*– електронний ключ; D_1, D_2–напівпровідникові діоди; I – вимірювач магнітоелектричної системи; *R*, R_1 - постійні опору.

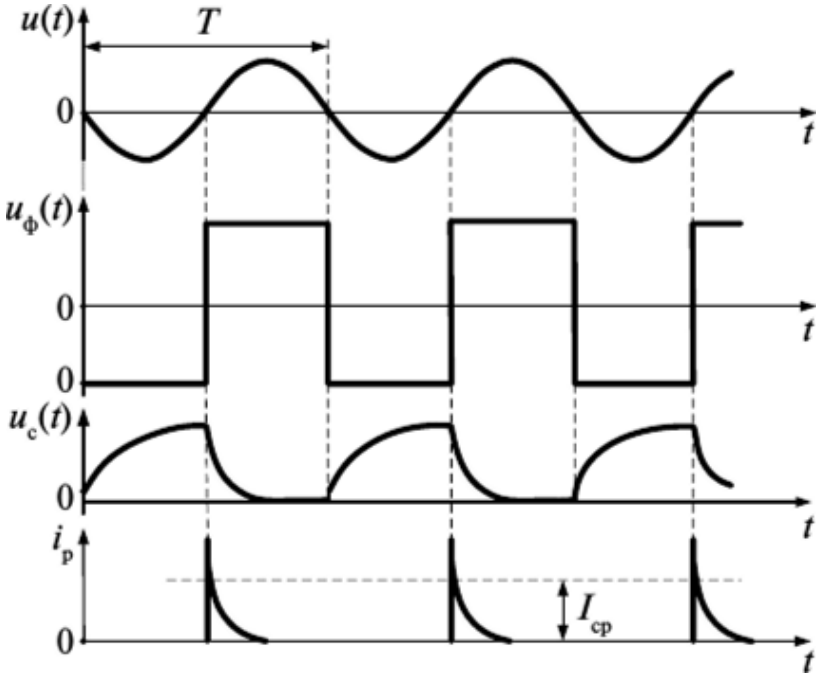


Рисунок 1.7 –Часова діаграма сигналів електронного конденсаторного частотоміра: $u(t)$– вхідний сигнал; $u_{\text{ф}}(t)$– напруга на виході *ПФ*; $u_{\text{с}}(t)$– напруга на конденсаторі *C*; $i_{\text{р}}$– сила струму розряду.

Заряд конденсатора *C* до напруги живлення станеться при виконанні умови (1.5)

$$t_{\text{з}} < 0.5T, \tag{1.5}$$

де $t_{\text{з}}$– тривалість заряду *C*; *T* – період вхідної напруги *C*.

Повний розряд конденсатора *C* відбудеться при виконанні умови (1.6)

$$t_{\text{р}} < 0.5T, \tag{1.6}$$

де $t_{\text{р}}$– тривалість розряду *C*.

При виконанні умов (1.5) і (1.6) конденсатор *C* за першу половину періоду запасає заряд q. За другу половину періоду ця кількість енергії проходить через вимірювач *I* (1.7)

$$q = EC. \tag{1.7}$$

Показання вимірювача магнітоелектричної системи пропорційні середній силі струму, що проходить через нього:

$$a = K_{\text{п}} I_{\text{ср}}, \tag{1.8}$$

$$I_{\text{ср}} = \frac{1}{T}\int_0^t i_p(t)dt, \tag{1.9}$$

$$q = \int_0^t i_p(t)dt, \tag{1.10}$$

$$f = \frac{1}{T}, \tag{1.11}$$

де a– показання приладу, Гц; $K_{\text{п}}$– коефіцієнт пропорційності;$I_{\text{ср}}$– середнє значення сили струму розрядження, А; i_p– миттєве значення струму розрядження конденсатора; t– поточний час; f– частота вхідної напруги.

На підставі співвідношень (1.7) - (1.11) можна записати:

$$a = K_{\text{п}} \frac{1}{T} q = K_{\text{п}} f E C, \tag{1.12}$$

$$K = K_{\text{п}} E C, \tag{1.13}$$

$$a = K f, \tag{1.14}$$

де K– коефіцієнт пропорційності по частоті.

Вираз (1.14) свідчить, що показання розглянутого приладу прямо пропорційні частоті вхідного сигналу.Додаткові відомості по електронним конденсаторним частотомірам наведені в літературних джерелах [8, 10].

1.1.5 Вібраційні частотоміри

Являються приладом з рухомою частиною у вигляді набору пружних елементів (пластинок, язичків), що приводяться в резонансні коливання при впливі змінного магнітного або електричного поля, рисунок 1.8. Найчастіше використовується електромагніт для збудження коливань і сталеві пластини в ролі елементів. Елемент, власна частота якого найближче до частоти струму, поточного по обмотці електромагніту, входить в резонанс і коливається з найбільшим розмахом, що відображається візуально [11].

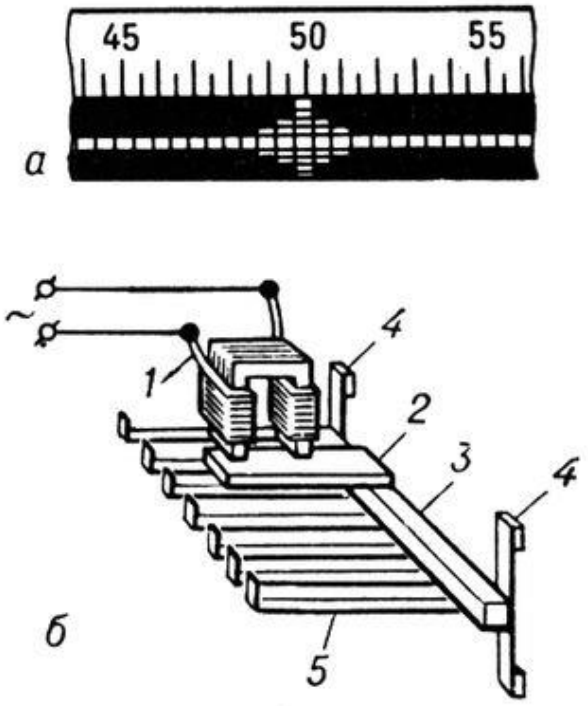


Рисунок 1.8 –Вібраційний частотомір: *а*–шкала; *б*– схема пристрою; *1*–обмотка електромагніта; *2*–якір електромагніта; *3*–основа частотоміра; *4*–кріплення які забезпечують пружність; *5*–пластини.

1.2 Аналіз існуючих багатоканальних систем вимірювання частоти на FPGA

Для одночасного вимірювання декількох частот доцільно застосовувати багатоканальні частотоміри. На відміну від кількох частотомірів, що працюють одночасно, багатоканальні частотоміри здійснюють вимірювання в єдиній базі часу, що забезпечується загальним зразковим генератором і лічильником часу.

Один з найбільш схожих приладів на систему яка представлена у цій роботі описаний у статті «Прецизійний частотомір для фундаментальної метрології» [27].

Пристрій містить три вимірювальних канали, принцип його дії можна розглянути на прикладі роботи одного каналу. В одному каналі є три лічильника, при цьому один з них є загальним для всіх каналів. Таким чином, в цьому пристрої один лічильник здійснює безперервний вимір часу, підраховуючи кількість імпульсів зразкової частоти, інший лічильник підраховує кількість імпульсів вимірюваної частоти, третій лічильник здійснює вимірювання

тривалості коректуючих імпульсів, сформованих спеціальною схемою і розтягнутих у часі в 1000 разів з високою точністю. У багатоканальному лічильнику час вимірюється загальним лічильником, а решта лічильників для кожного каналу індивідуальні. Лічильники часу і лічильники кількості імпульсів працюють без зупинки, читання їх показань здійснюється без зупинки лічби, що забезпечується схемою синхронізації імпульсів, а також відповідним вибором малої розрядності цих лічильників. У використаному технічному рішенні старші розряди лічильника відновлюються програмно. Від кожного імпульсу схема синхронізації формує новий імпульс, затриманий до часу приходу чергового переднього фронту імпульсів зразкової частоти, тому момент зміни значення лічильників завжди збігається з моментом надходження одного з цих фронтів. Читання значень лічильників в регістр, синхронізовано з задніми фронтами цих же імпульсів.

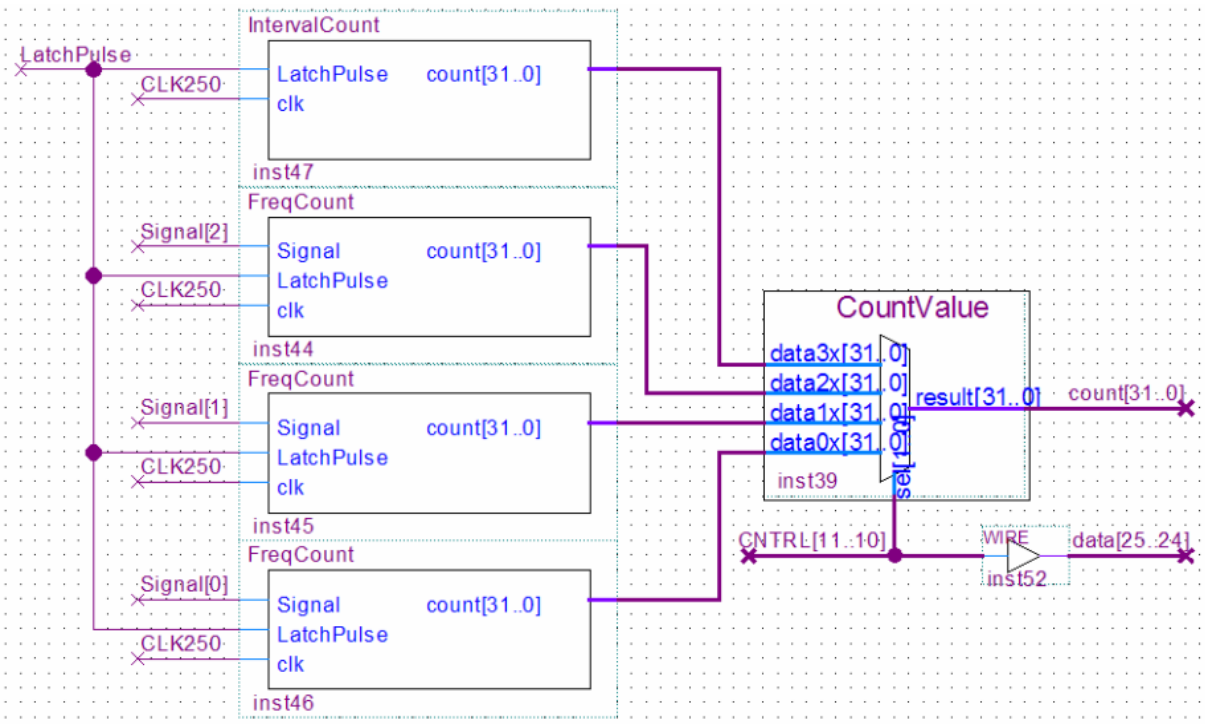


Рисунок 1.9 – Структура модуля лічильника

Хоч ціль у розглянутого прилада такаж сама – вимірювання частоти, але шляхи її досягнення різні. Реалізація системи, розробка якої представлена у даній МКР, краще пристосована до внесення змін і розширення функціональної

складової, пропонована система являється більш універсальною завдяки підтримці сучасних цифрових інтерфейсів.

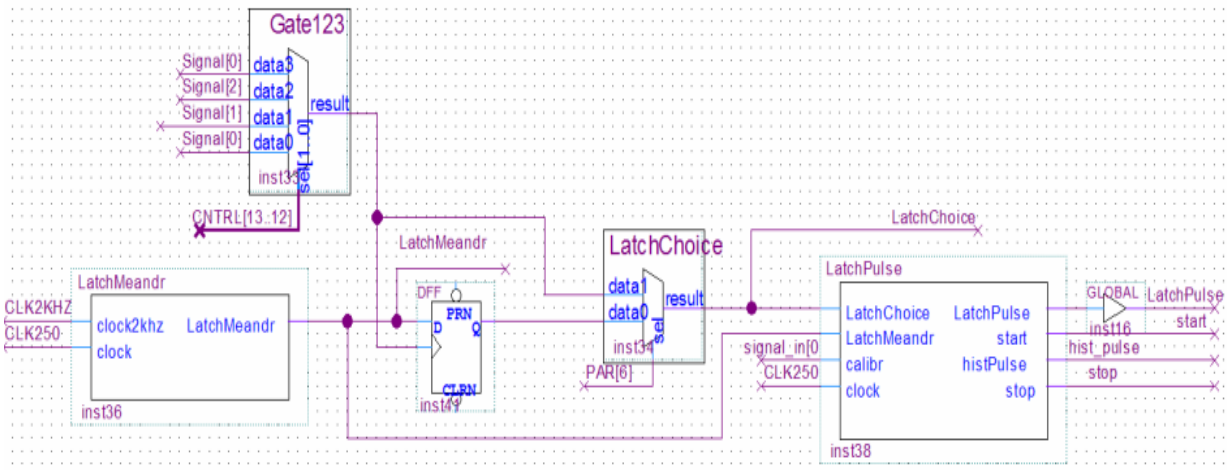


Рисунок 1.10 – Структура лічильника часу

1.3 Висновки до першого розділу

Проведено аналіз існуючих типів частотомірів і принципів які лежать в основі їх реалізації:

1. Електронно-лічильні, рахують кількість імпульсів вхідного сигналу за визначений проміжок часу, універсальні, великий діапазон частот, точність;
2. Резонансні, порівнюють частоту вхідного сигналу з особистою резонансною частотою;
3. Гетеродинні, порівнюють частоту вхідного сигналу з частотою гетеродинна використовуючи метод «нульового биття»;
4. Конденсаторні, зарядження конденсатора від батареї з наступним його розрядженням через електромагнітний механізм, частота відповідає вимірюваній частоті, похибка становить 2-3%;
5. Вібраційні, перетворюють електричні коливання у механічні.

На основі аналізу проведеного в даному розділі, було вирішено використовувати принципи вимірювання частоти які лежать в основі електронно-лічильних частотомірів, оскільки, вони: дозволяють вимірювати великий діапазон значень, не потребують додаткових дій для налаштування, можуть бути реалізовані на FPGA.

2 РОЗРОБКА БАГАТОКАНАЛЬНОЇ ВИМІРЮВАЛЬНОЇ СИСТЕМИ НА FPGA

2.1 Принципи роботи FPGA

FPGA - це електронний компонент, який використовується для створення цифрових інтегральних схем. На відміну від звичайних цифрових мікросхем, логіка роботи FPGA не визначається при виготовленні, а задається за допомогою програмування (проектування). Для цього використовуються спеціальні середовища розробки, що дозволяють задати бажану структуру цифрового пристрою у вигляді принципової електричної схеми або програми на спеціальних мовах опису апаратної структури Verilog, VHDL, AHDL. Що забезпечує бажану гнучкість як при створенні, так і при застосуванні.

Зазвичай, сама мікросхема FPGA складається з:

1. конфігуруючих логічних блоків, що реалізують необхідну логічну функцію;
2. програмованих електронних зв'язків між конфігуруючими логічними блоками;
3. програмованих блоків введення/виведення, що забезпечують зв'язок з внутрішньою логікою.

В сучасних FPGA часто бувають вбудовані додатково блоки пам'яті, блоки DSP або помножувачі, PLL.

2.1.1 Класифікація FPGA за типом зберігання конфігурації

У FPGA для конфігурації використовується оперативна пам'ять CRAM (Configuration RAM). Ця пам'ять розподілена по всьому кристалу, значення, записані в неї, управляють внутрішнім комутаційним полем, визначаючи структуру синтезованого цифрового пристрою. Як правило, в FPGA ця пам'ять енергозалежна і при подачі живлення на пристрій потрібно завантажити в неї значення з будь-якого зовнішнього (по відношенню до кристалу FPGA) носія, часто для цих цілей використовується мікросхема ПЗУ (Постійно запам'ятовуючий пристрій), або мікроконтролер.

Існує три найбільш поширених способи зберігання конфігурацій:

1. SRAM-Based. Це одна з найпоширеніших різновидів FPGA. Конфігурація FPGA зберігається у статичній пам'яті, виготовленій за стандартною технологією CMOS. Переваги цієї технології - можливість багаторазового перепрограмування. Недоліки - не найвища швидкодія, після включення живлення прошивку потрібно знову завантажувати. Отже на платі повинензнаходитись завантажувач, спеціальна мікросхема FLASH або мікроконтролер - все це збільшує вартість кінцевого виробу.

2. Flash-based. У таких мікросхемах зберігання конфігурації відбувається у внутрішній FLASH пам'яті або пам'яті типу EEPROM. Такі FPGA краще тим, що при виключенні живлення прошивка не зникає. Після подачі живлення мікросхема знову готова до роботи. Однак, у цього типу FPGA є і свої недоліки. Реалізація FLASH пам'яті всередині CMOS мікросхеми - це не дуже просто. Потрібно поєднати два різних техпроцеси для виробництва таких мікросхем. Отже вони виходять дорожче. Крім того, такі мікросхеми, як правило, мають обмежену кількість циклів перезапису конфігурацій.

3. Antifuse. Спеціальна технологія по якій виконуються одноразово програмовані FPGA. Програмування такої FPGA полягає в руйнуванні в потрібних місцях чіпа спеціальних перемичок для утворення потрібної схеми. Недолік - власне програмувати/прошивати чіп можна тільки один раз. Після цього виправити вже нічого не можна. Сам процес прошивки досить повільний. Зате є маса переваг у таких ПЛІС: вони швидкі, менше схильні до збоїв при радіації - все через те, що конфігурація виходить у вигляді перемичок, а не у вигляді додаткової логіки, як у SRAM-based .

2.1.2 Конфігуруючі логічні блоки FPGA

У документації компанії Alteraзустрічається абревіатураLAB(Logic Array Block) - масив логіки [14]. У компанії Xilinx в мікросхемах FPGA є приблизно такі ж блоки - Configurable Logic Block (CLB). CLB - це базовий елемент в FPGA, в ньому може бути виконана якась проста логічна функція або реалізовано зберігання результату обчислення в регістрах (тригерах).Складність і структура CLB визначається виробником.Теоретично, CLB може бути, наприклад, дуже простим, просто як окремий транзистор. Або він може бути дуже складним, як цілий процесор. Це крайні точки реалізації. У першому випадку буде потрібна величезна кількість програмованих зв'язків, щоб потім з окремих транзисторів

зібрати необхідну схему. У другому випадку зв'язків може потрібно і не так багато, але втрачається гнучкість проектування схеми. Саме тому конфігуруючий блок зазвичай представляє з себе щось середнє: він зазвичай досить складний(щоб можна було реалізувати базову операцію/функцію) і досить малий (щоб розмістити безліч таких блоків всередині FPGA).

З булевої алгебри відомо, що використовуючи якийсь елементний базис, наприклад елемент І-НЕ, АБО-НЕ, можна реалізувати будь-яку логічну функцію. Однак використання лише одного типу елемента не завжди виправдано технічно, при синтезі складних пристроїв велика кількість елементів збільшить час проходження сигналу і тим самим знизить швидкодію. Тому в FPGA у якості найпростішого логічного елемента (LE) використовується складніша структура, що представляє собою з'єднання програмованого комбінаційного пристрою і D-тригера, рисунок 2.1.

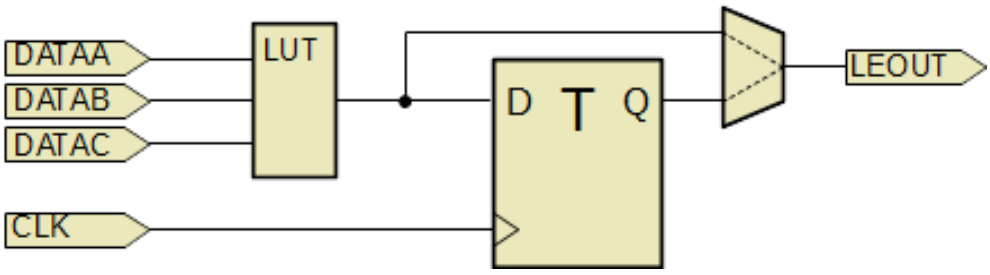


Рисунок 2.1 – Узагальнена структура логічного елемента FPGA

LE – має три логічних входи DATAA, DATAB і DATAC, вхід тактових імпульсів CLK і один вихід LEOUT. У разі якщо від LE потрібна робота в якості лише комбінаційного пристрою, то вихідний мультиплексор комутує вихід елемента LUT на вихід всього LE, якщо вихід повинен бути регістровим, то сигнал з LUT замикається по сигналу синхронізації в D-тригер, вихід якого через мультиплексор з'єднується з LEOUT. Керуючий вхід мультиплексора підключений до відповідного біту конфігураційної пам'яті CRAM.

Абревіатура LUT розшифровується як Look-Up Table або просто LookupTable, що дослівно можна перекласти як "таблиця пошуку". LUT - це більше, ніж таблиця, LUT - це скоріше метод реалізації функції, в якому безпосереднє обчислення замінюється пошуком по таблиці готових рішень. Стосовно FPGA це дозволяє реалізувати будь-яку логічну функцію у вигляді

пам'яті SRAM, де адреса - це аргумент, а вміст комірки - значення. Таким чином, для того, щоб описати логічну функцію трьох змінних достатньо пам'яті на 8 комірок. Необхідна таблиця істинності зберігається у вигляді маски (LUT-mask) в відповідній комірці CRAM. За допомогою мультиплексорів вибирається потрібне значення. Мультиплексорами керують сигнали вхідних портів для побудови k-вхідного LUT (k-LUT), яка реалізує будь-яку логічну функцію з k змінних, для реалізації потрібно 2^k біт SRAM і 2^{k-1} мультиплексорів [15], рисунок 2.2.

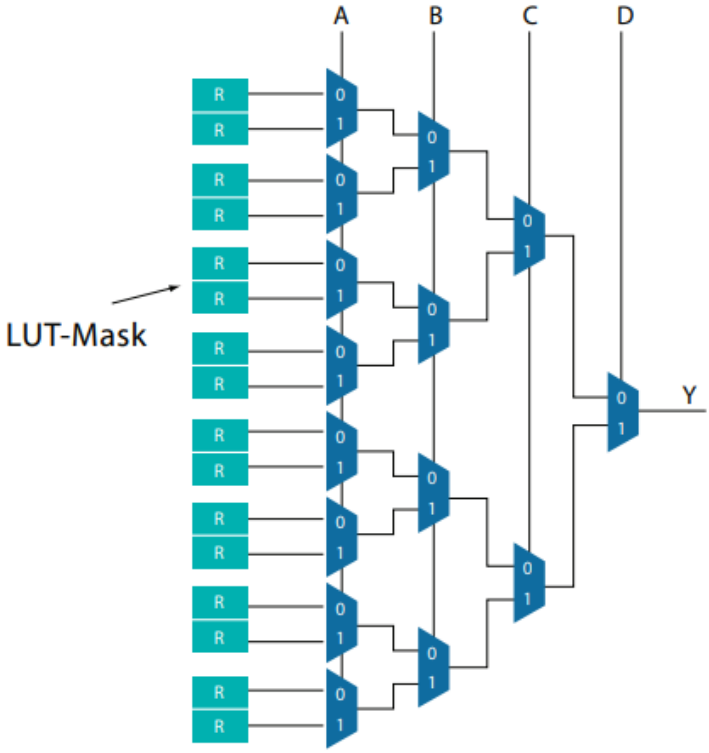


Рисунок 2.2 – Схема блоку LUT на чотири входи

При такому підході можна досить точно спрогнозувати час проходження сигналу і він не буде залежати від реалізованої логічної функції. Ця важлива особливість робить можливим часовий аналіз схеми.

Насправді базовий логічний елемент в різних FPGA виявляється набагато складніше, ніж показано на рисунку 2.1.

На рисунку 2.3 добре видно LUT і D-Тригер зберігання результату. У мікросхемах компанії Xilinx Virtex-6 базовий логічний елемент - це так званий Slice. В одному CLB всього два Slice. Зате один Slice - це досить складний пристрій рисунок 2.4

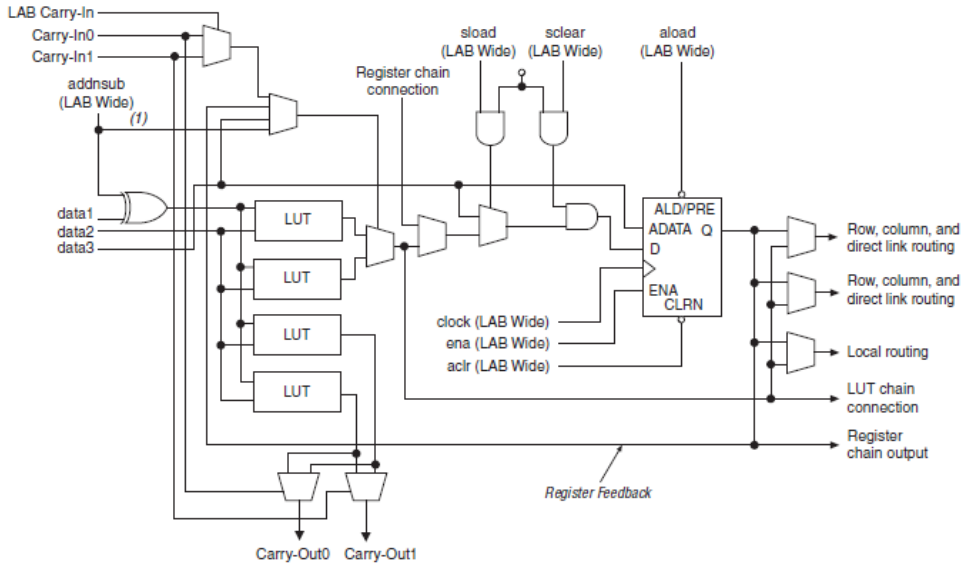


Рисунок 2.3– Базовий логічний елемент CPLD MAX II компанії Альтера.

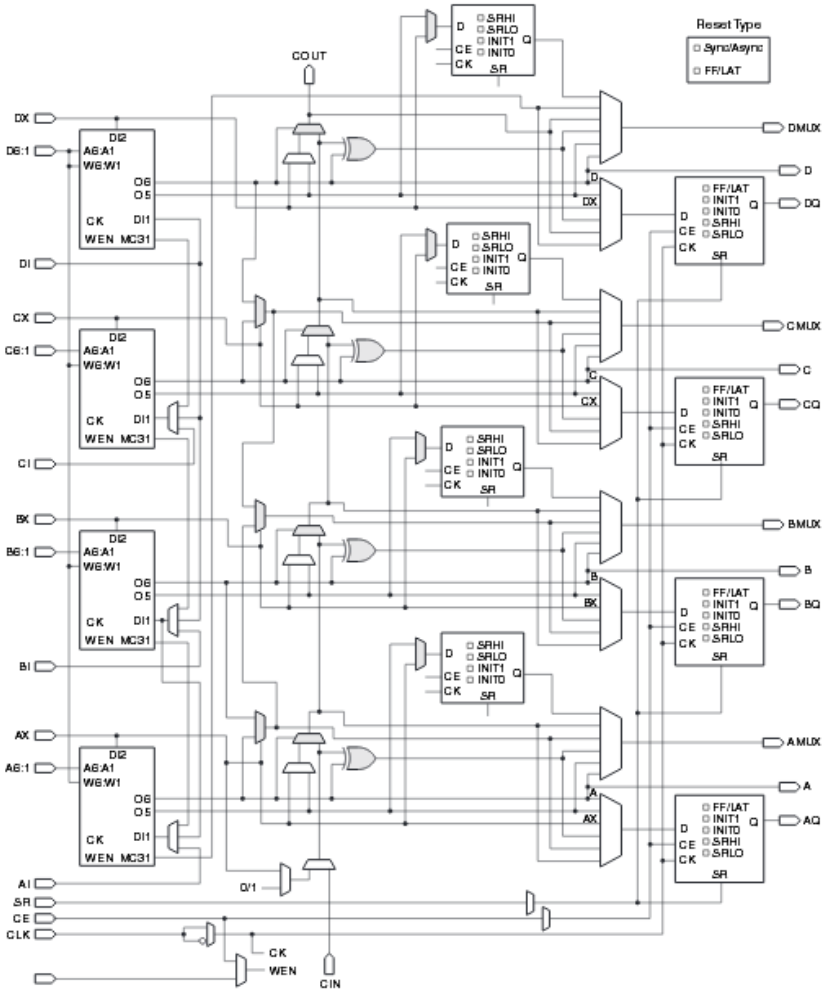


Рисунок 2.4– Базовий елемент Xilinx Virtex-6 Slice.

2.1.3 Програмовані зв'язки між логічними блоками FPGA

Щоб в FPGA запрацювала потрібна нам цифрова схема мало того, що потрібно конфігурувати наявні логічні блоки особливим чином, ще потрібно створити, запрограмувати зв'язки між логічними блоками. Для цього в FPGA є спеціальні конфігуруючі комутатори [16].

Відомо дві основні методики побудови FPGA за типом архітектури зв'язків: острівна і ієрархічна.

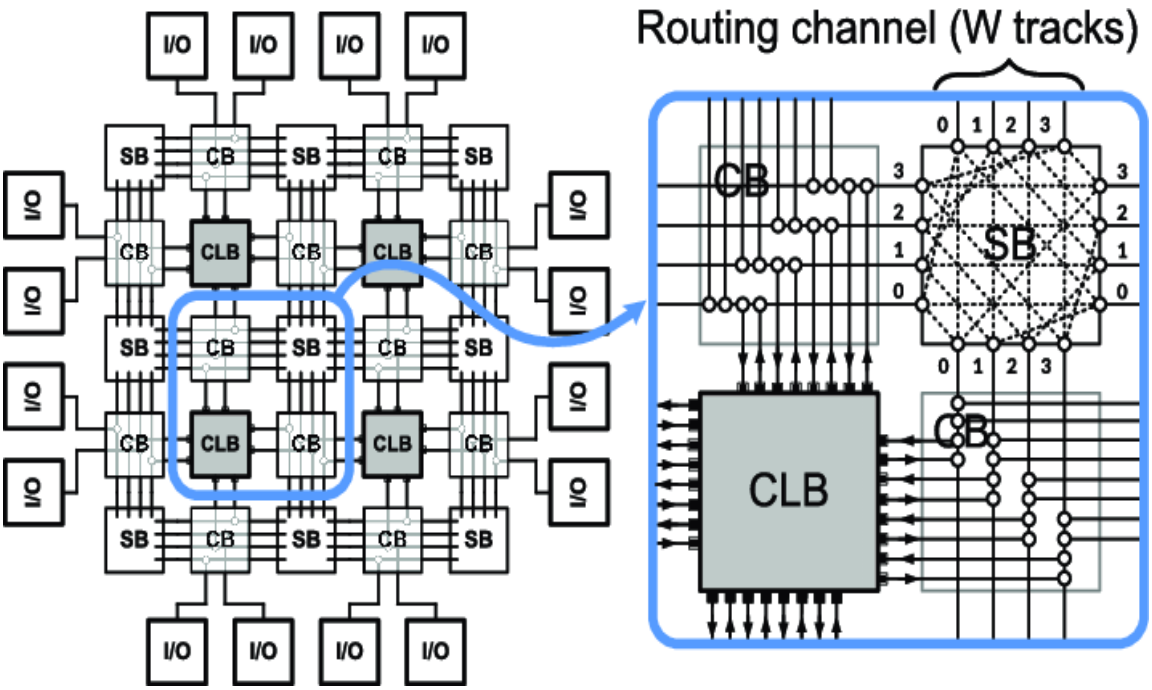


Рисунок 2.5– Глобальна архітектура FPGA в острівному стилі

Острівна FPGA називається так тому, що конфігуруючі блоки усі рівні між собою і знаходяться, як острова в океані, між вузлами комутації та лініями зв'язку. На рисунку 2.5 відзначені C - ConnectionBox і S - SwitchBox, по суті це програмовані мультиплексори, які підключають той чи інший CLB до іншого CLB через ланцюжки проводів в FPGA.

Другий відомий тип FPGA - це ієрархічні FPGA. Тут йде розрахунок на те, що в схемі завжди є ділянки які взаємодіють один з одним більш тісно, ніж з віддаленими модулями проекту.

На рисунок 2.6 зображена ієрархічна FPGA. Основні переваги: прилеглі CLB з'єднати досить просто, потрібна не велика кількість комутаторів через що зменшується затримка проходження сигналу. Але якщо потрібен більший блок

обчислювачів, то сигнал повинен вийти на більш високий рівень ієрархії і потім зайти всередину в сусідній CLB. Не можна сказати, що це істотно гірше, ніж острівний стиль. Кожен метод має своїпереваги.

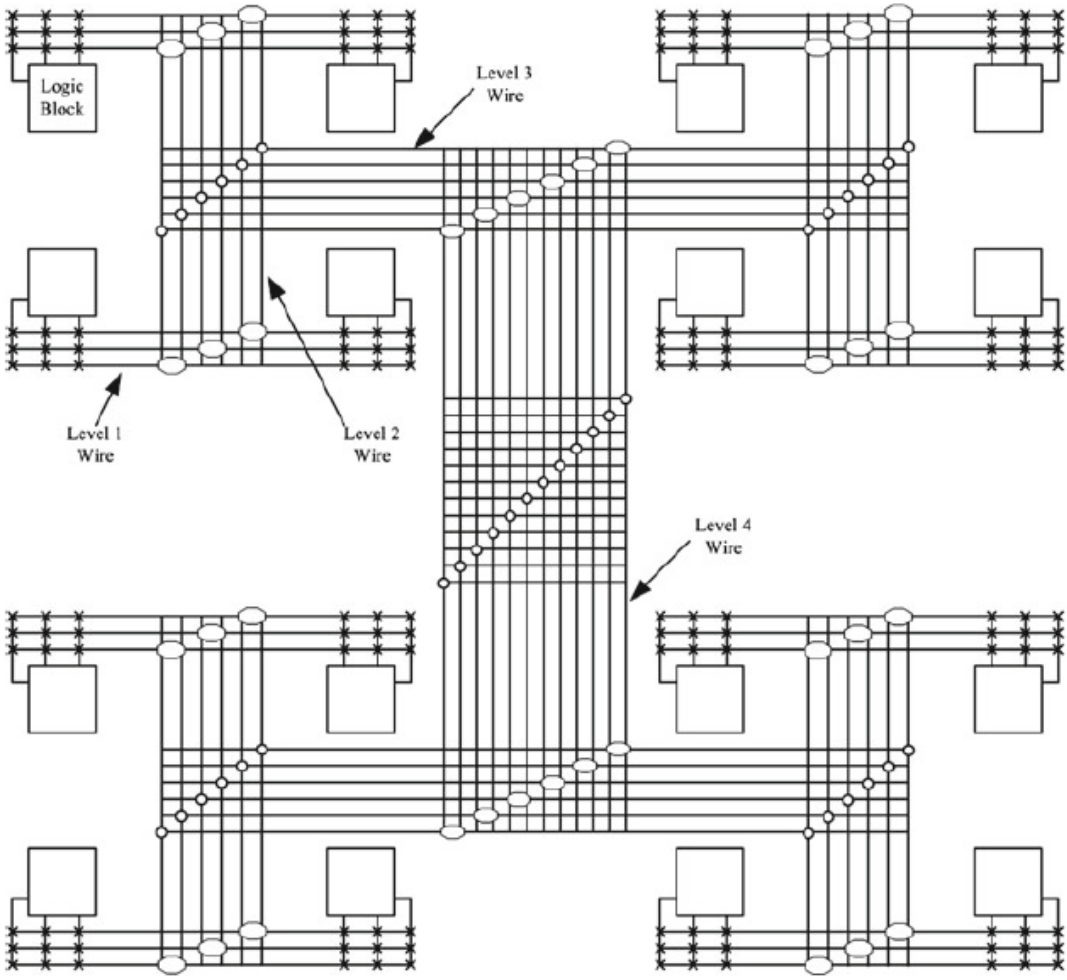


Рисунок 2.6– Глобальна архітектура FPGA в ієрархічному стилі

2.1.4 Програмне забезпечення для проектування FPGA

Програмне забезпечення для проектування FPGA, а саме компілятор (синтезатор логіки, фіттер, асемблер) – це можливо найскладніша частина всієї FPGA технології. Компілятор повинен проаналізувати проект користувача (схеми і текстові файли на VerilogHDL або VHDL) і згенерувати нетліст (netlist) - список всіх елементів схеми і зв'язки між ними. Netlist повинен бути оптимізований - логічні функції потрібно мінімізувати, можливі дубльовані регістри потрібно видалити.

Потім компілятор повинен вмістити всю логіку з netlist в наявну архітектуру FPGA. Це робить фіттер (fitter). Він розміщує логічні елементи і виконує трасування зв'язків між ними (процес placeandroute). Складність полягає в тому, що один і той же проект може бути розміщений в FPGA різними способами і цих способів мільйони. Деякі розміщення і трасування виявляються краще, інші гірше. Головний критерій якості отриманої системи - максимальна частота, на якій зможе працювати проект при даному розміщенні елементів і трасуванні зв'язків. Тут впливає довжина зв'язків між логічними блоками і кількість програмованих комутаторів між ними. Компілятор, знаючи архітектуру FPGA за результатами роботи додатково видає звіт про час проходження сигналів від регістру до регістру. Ця інформація часто буває корисною для розробника високопродуктивних систем. Розробник для FPGA має можливість давати деякі поради компілятору де, в якому місці кристала краще розмістити той чи інший модуль проекту. Вибираючи для свого проекту, для своєї плати конкретну мікросхеми FPGA розробник в деякій мірі потрапляє в залежність від виробника цієї FPGA, так як повинен в роботі користуватися програмним забезпеченням від цього ж виробника.

Програмне забезпечення компанії Альтера: QuartusII. Програмне забезпечення Xilinx для проектування для ПЛІС: ISESuite, VivaldoDesignSuite.

2.2 Передумови до розвитку FPGA

Загальною тенденцією розвитку елементної бази цифрової схемотехніки, починаючи з появи перших інтегральних мікросхем на початку 60-х років і до теперішнього часу, є безперервне підвищення числа логічних елементів, що розміщуються на кристалі, з одночасним зниженням питомої вартості одного елемента. Збільшення числа LE безперервно відкриває можливості створення все більш складних цифрових пристроїв, що розміщуються на одному кристалі. До основних позитивних результатів цієї тенденції можна віднести:

1. постійне розширення функціональних можливостей і поліпшення споживчих властивостей кінцевих виробів;
2. зменшення габаритів і споживаної потужності;
3. підвищення надійності.

Відображенням цієї тенденції в складі елементної бази цифрової схемотехніки являється перехід від інтегральних мікросхем малої і середньої ступені інтеграції до великих і надвеликих (НВІС) інтегральних мікросхем. Одним з найбільш революційних результатів розвитку мікроелектроніки з'явилася можливість створення перших мікропроцесорів (початок 70-х років), що дало потужний поштовх до впровадження цифрових технологій обробки інформації в усіх сферах людської діяльності.

Однак далеко не всі практичні завдання цифрової схемотехніки можуть бути вирішені тільки з використанням одних мікропроцесорів. Це обумовлено особливістю притаманною усім мікропроцесорам, вирішення будь-якої задачі мікропроцесором завжди складається з послідовності кроків кінцевої тривалості, в той час як для вирішення багатьох завдань (в тому числі пов'язаних і з забезпеченням роботи самих мікропроцесорів) потрібні пристрої з мінімальною затримкою виконання логічних функцій. Існує три основних способи задовольнити цю потребу:

1. використання наборів стандартної дискретної цифрової логіки загального застосування;
2. використання замовних НВІС;
3. використання програмованих логічних інтегральних схем (FPGA).

Набори дискретної цифрової логіки різних серій досить тривалий час були основною елементної базою для розробки цифрових пристроїв. До складу таких наборів входить велика кількість окремих мікросхем, призначених як для виконання базових логічних функцій (І, АБО, НЕ), так і для виконання функцій типових цифрових пристроїв, таких як тригери, регістри, лічильники, мультиплексори, дешифратори і т.д. , що дає можливість використовувати їх для розробки більш складних функціонально закінчених цифрових пристроїв.

Основний недолік дискретної логіки полягає в тому, що для розробки кінцевих виробів зазвичай потрібна велика кількість мікросхем. Наслідком цього є велика кількість зовнішніх з'єднань, складність конструкції і великі габарити друкованих плат, велика довжина сполучних провідників, складність побудови пристроїв з високою тактовою частотою, низька надійність. Для зменшення числа мікросхем при проектуванні мікропроцесорних систем був розроблений ряд периферійних ВІС, що представляють собою спеціалізовані цифрові пристрої, призначені для виконання деяких типових функцій в складі мікропроцесорних

систем, такі як контролери динамічних RAM, контролери переривань, контролери прямого доступу в пам'ять, контролери шин. Однак навіть застосування периферійних НВІС не дозволяє повністю подолати основні недоліки дискретної цифрової логіки.

Найбільш кардинально проблема габаритів, швидкодії, спрощення конструкцій друкованих плат і забезпечення надійності вирішується шляхом розробки і виготовлення замовних НВІС (класичний приклад - чіпсети материнських плат). На жаль, цей шлях економічно виправданий тільки при великосерійному виробництві однотипних кінцевих виробів, внаслідок високої вартості і тривалих термінів підготовки виробництва замовних НВІС. Крім того, при використанні замовних НВІС можлива модифікація виробів вимагає істотних додаткових матеріальних витрат.

У той же час на практиці досить часто виникають потреби в розробці оригінальних цифрових пристроїв і виробів, не розрахованих на багатосерійне виробництво, для яких розробка замовних НВІС не прийнятна або з економічних причин, або за термінами виконання. Протягом тривалого часу єдиним шляхом для вирішення таких завдань було використання інтегральних мікросхем дискретної логіки і периферійних НВІС, адже можливості перших поколінь простих FPGA були досить обмежені, а ціна складних FPGA дуже високою, крім того, були певні складнощі і з проектуванням цифрових пристроїв на FPGA.

За останнє десятиліття, однак, відбувся різкий прорив як в технології виготовлення FPGA, так і в розробці інструментальних засобів, призначених для проектування цифрових пристроїв на FPGA і випуску готових виробів. Технологічний прорив характеризується різким збільшенням числа еквівалентних логічних вентилів, що розміщуються на одному кристалі, підвищенням робочої частоти з різким одночасним зниженням як питомої, так і абсолютної вартості.

2.3 Порівняльний аналіз FPGA з іншими апаратними платформами

Перший чіп FPGA XC2064, створений Xilinx в 1985 році, містив всього 64 CLB рисунок 2.7. У той час інтеграція транзисторів на мікросхемах була набагато нижче, ніж зараз, і в цифрових пристроях часто використовувалися мікросхеми «розсипної логіки». Були окремо мікросхеми регістрів, лічильників,

мультиплексорів, помножувачів. Під конкретний пристрій створювалася своя друкована плата, на якій встановлювалися ці мікросхеми низької інтеграції [12].

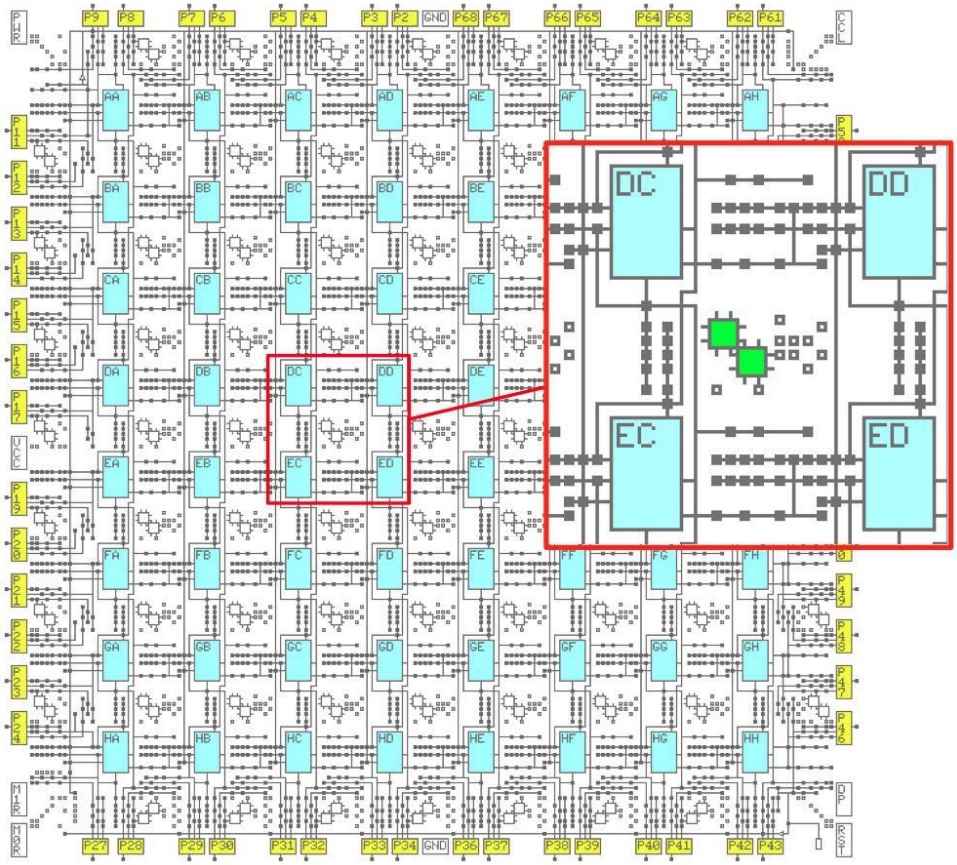
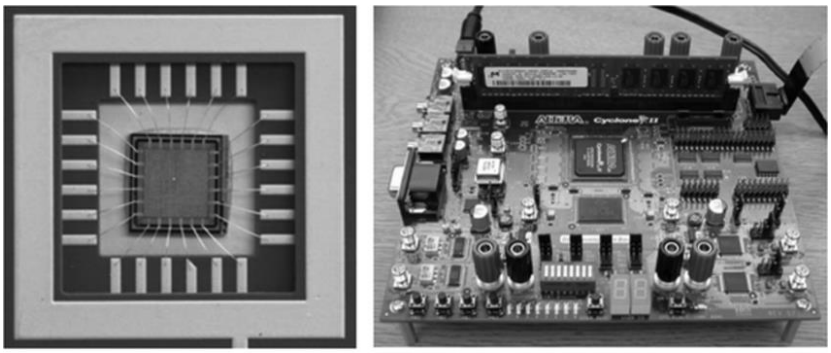


Рисунок 2.7 –Логічні блоки FPGAXC2064, з'єднані між собою.

Використання FPGA дозволило відмовитися від такого підходу. Навіть FPGA на 64 CLB значно економить місце на друкованій платі, а доступність реконфігурації додала можливість оновлювати функціональність пристроїв вже після виготовлення під час експлуатації. За рахунок того, що всередині FPGA можна створити будь-яку апаратну цифрову схему, одне з важливих застосувань FPGA - це прототипування мікросхем ASIC. Розробка ASIC дуже складна і витратна, ціна помилки дуже висока, і питання тестування логіки критичне. Тому одним з етапів розробки ще до початку роботи над фізичною топологією схеми стало її прототипування на одному або декількох кристалах FPGA.

Для розробки ASIC випускають спеціальні плати, що містять багато FPGA, з'єднаних між собою рисунок 2.8. Прототип мікросхеми працює на значно менших частотах, але дозволяє заощадити на виявленні проблем і логічних

помилок. Однак, існують більш цікаві застосування FPGA. Їхня гнучка структура дозволяє реалізовувати апаратні схеми для високошвидкісної і паралельної обробки даних з можливістю зміни алгоритму [13].



(*a*) (*b*)

Рисунок 2.8 –Мікросхема ASIC (*a*) та плата розробки FPGA (*b*).

Чим принципово відрізняються CPU, GPU, FPGA та ASIC рисунок 2.8. CPU – універсальний, можливо запустити будь-який алгоритм, найбільш зручний, і використовувати його легше всього за рахунок величезної кількості мов програмування та середовищ розробки.

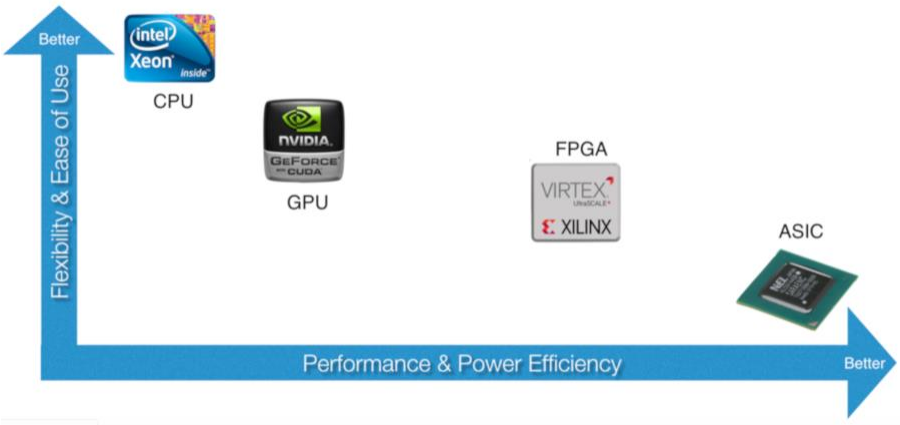


Рисунок 2.9 –Порівняння існуючих апаратних платформ.

При цьому через універсальності та послідовне виконання інструкцій зменшується продуктивність та підвищується енергоспоживання. Це відбувається через те, що на кожну корисну арифметичну операцію CPU виконує велику кількість додаткових операцій, пов'язаних із зчитуванням інструкцій,

переміщенням даних між регістрами та кешем. На іншій стороні знаходиться ASIC. На цій платформі необхідний алгоритм реалізується за рахунок прямого з'єднання транзисторів, всі операції зв'язуються тільки з виконанням алгоритму та немає жодних можливостей змінити його. Максимальна продуктивності та мінімальне енергоспоживання. Але перепрограмувати ASIC неможливо.

На рисунку 2.9 справа від CPU знаходиться GPU. Спочатку ці мікросхеми були розроблені для обробки графічних зображень, але на даний час сфера їх використання значно розширилась. Вони складаються з тисяч невеликих обчислювальних ядер і виконують паралельні операції над масовими даними. Якщо алгоритм можна розпаралелити, то на GPU буде отримано значне прискорення порівняно з CPU. З іншого боку, послідовні алгоритми будуть реалізовуватись гірше, тому платформа являється менш гнучкішою ніж CPU. Також для розробки під GPU потрібно мати спеціальні навики, знати OpenCL або CUDA.

FPGA. Ця платформа поєднує ефективність ASIC з можливістю зміни алгоритму. FPGA не являються універсальним, але є клас алгоритмів та завдань, які на них будуть показувати кращу продуктивність, ніж на CPUабо GPU. Важкість розробки під FPGA набагато вище ніж під CPU, але нові середовища розробки роблять цей розрив з кожним роком все менше. Найбільша перевага FPGA - це здатність обробляти дані в темпі їх надходження з мінімальною затримкою. Як приклад можна розглянути мережевий маршрутизатор з великою кількістю портів: при надходженні пакета Ethernet на один з його портів необхідно перевірити безліч правил, перш ніж вибрати вихідний порт. Можливо, буде потрібно зміна деяких полів пакету або додавання нових. Використання FPGA дозволяє вирішувати цю задачу миттєво: байти пакета ще тільки почали надходити в мікросхему з мережевого інтерфейсу, а його заголовок вже аналізується. Використання процесорів тут може істотно зменшити швидкість обробки мережевого трафіку. Звичайно, що для маршрутизаторів можна зробити замовну мікросхему ASIC, яка буде працювати найбільш ефективно, але до того часу поки алгоритми обробки залишаються незмінними. Досягти необхідної гнучкості в поєднанні з високою продуктивністю допоможе тільки FPGA.

Таким чином, FPGA використовується там, де необхідна висока продуктивність, мінімальна затримка, низьке енергосроживання.

2.4 Реалізація багатоканального частотоміра на FPGA

2.4.1 Вибір мікросхеми і середовища розробки

1 знаходяться у додатку Б. Параметрами, а саме значенням затримки поширення сигналу, визначається максимальна вимірювана частота. В даному випадку близько 3 нс (затримка проходження сигналу від входу до внутрішнього регістра) відповідає теоретично 333 МГц [17]. Вибір саме цієї FPGA аж ніяк не є обов'язковим в даному проекті можна застосувати будь-яку FPGA, в будь-якому корпусі, але з кількістю логічних елементів не менше 4000, відповідно доведеться перекомпілювати проект під застосування нової FPGA.

Для створення проектів під FPGAкомпанії Altera використовується середовище розробки – Quartus. Версій Quartus існує дуже велика кількість, тому потрібно вибрати найбільш стабільну і з підтримкою вибраної мікросхеми. Було вибрано версію Quartus 15.1 [18]. Для роботи з мікросхемою EP4CE10F17C8 потрібен сам Quartus і пакет для підтримки CycloneIV (cyclone-15.1.0.185.qdz).

2.4.2 Блок керування лічильниками імпульсів

Одним із основних блоків розроблюваного пристрою являється формувач сигналів керування зображений на рисунок 2.10 та наведений у додатку В. Код на мовіопису обладнання – Verilog, для даного компонента наведена у додатку Д.

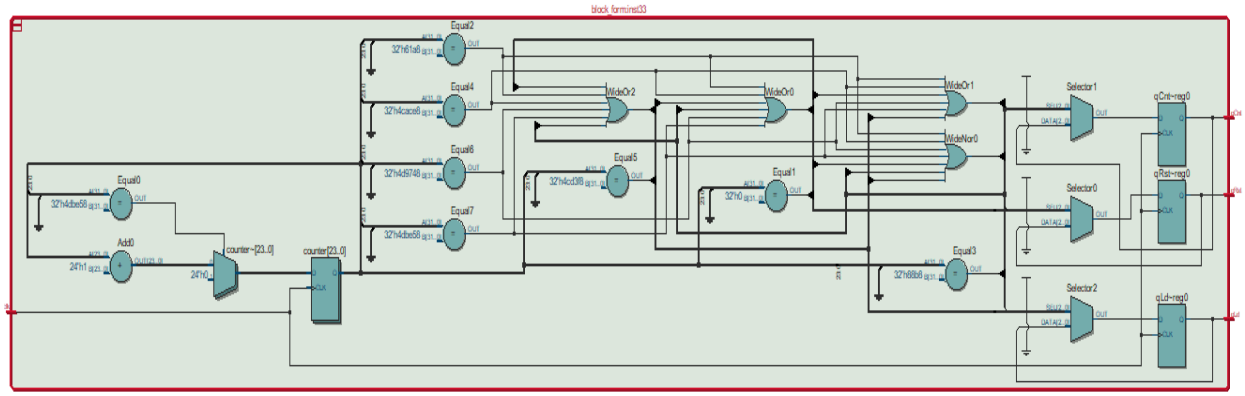


Рисунок 2.10 – Схематичне зображення генератора сигналів керування

У якості вхідного параметра він приймає сигнал від основного генератора на 50МГц. На виході послідовно створюються три сигнали керування:

1) «qRst» - Скидає значення лічильників, тривалість 0.5мс, рисунок 2.11;

2) «qCnt» - Запускає підрахунок імпульсів на вхідних каналах, тривалість 100мс, рисунок 2.12;
3) «qLd» - Запускає процес обробки і інтерпретації даних з лічильників, тривалість 1мс, рисунок 2.13.

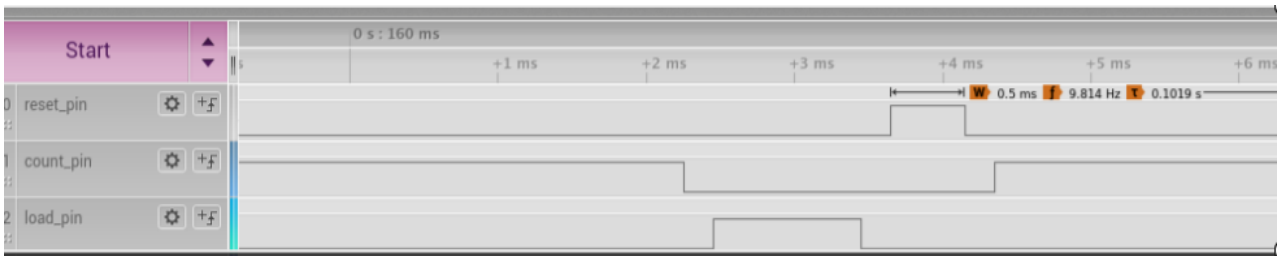


Рисунок 2.11 – Сигнал керування «qRst»

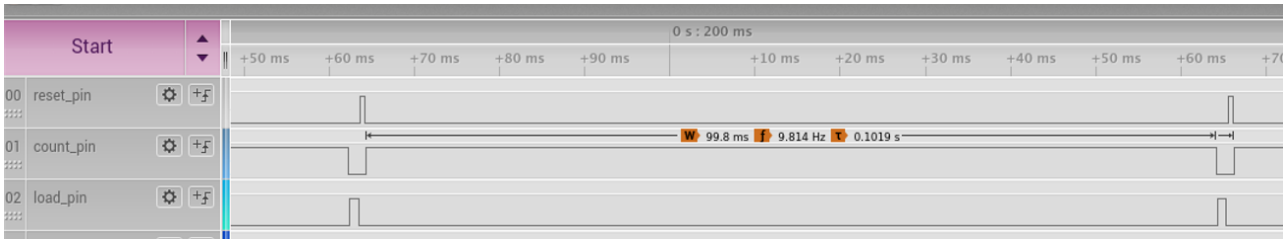


Рисунок 2.12 – Сигнал керування «qCnt»

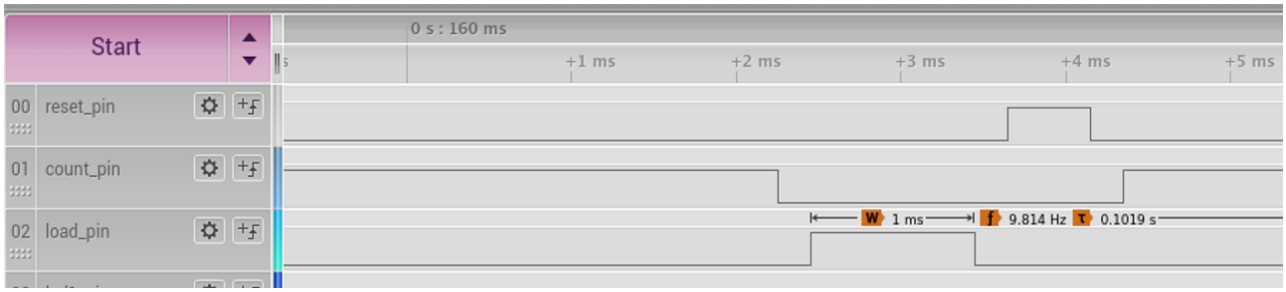


Рисунок 2.13 – Сигнал керування «qLd»

2.4.3 Блок лічильника імпульсів

Наступним не менш важливим компонентом являється лічильник імпульсів зображений на рисуноку 2.14 та наведений в додатку Е.

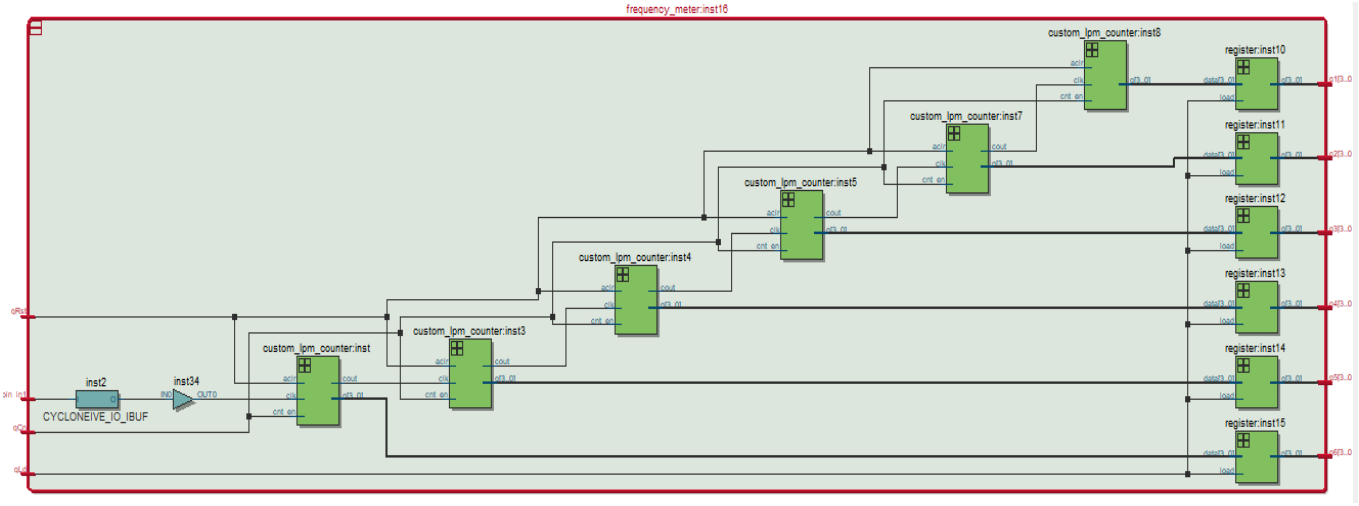


Рисунок 2.14 – Схематичне зображення лічильника імпульсів

Лічильник складається з трьох основних компонентів :

1. Тригер Шмітта, що забезпечує завадостійкість на вході;
2. Блоків «custom_lpm_counter» які виступають у якості десятичних лічильників з можливістю перенесення старшого розряду, вони рахують у діапазоні від 0 до 9 (включно), рисунок 2.15, опис блоку на мові Verilog знаходиться у додатку Ж;
3. Чотирирозрядних регістрів тимчасового зберігання значень лічильників, рисунок 2.16.

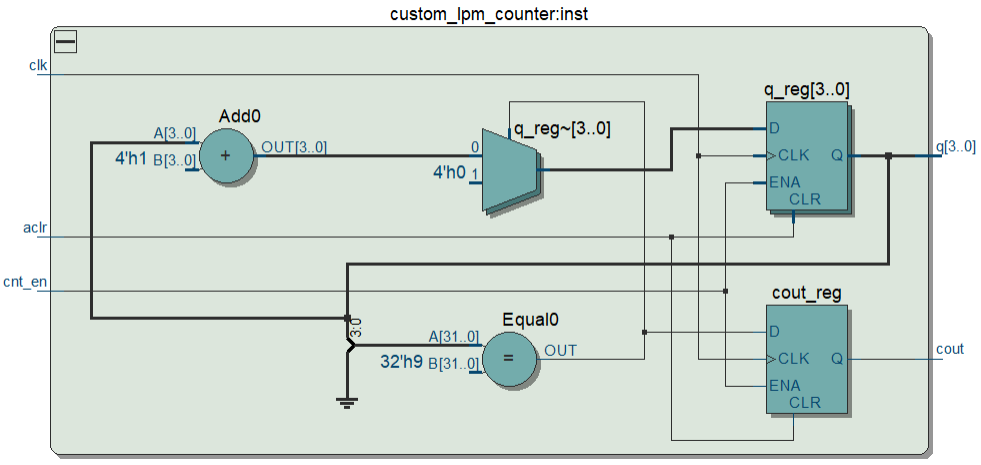


Рисунок 2.15 – Схематичне зображення десятичного лічильника

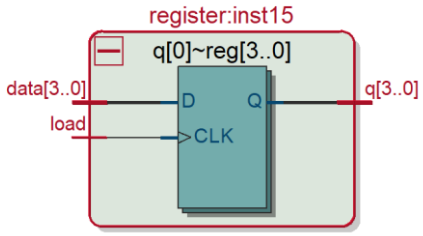


Рисунок 2.16– Схематичне зображення регістра тимчасових значень

На вхід блока лічильника подається сигнал від генератора частоту якого необхідно виміряти, після чого сигнал керування «qCnt» приймає високий рівень і компоненти «custom_lpm_counter» починають рахувати, переносячи кожний 10 імпульс на сусідній «custom_lpm_counter» який під'єднаний до виходу «cout», таким чином після завершення роботи кожний з компонентів містить у собі один із розрядів десятичного числа. Після того як «qCnt» приймає низький рівень усі «custom_lpm_counter» одночасно припиняють підрахунок імпульсів і очікують на наступні команди. Наступним командним сигналом являється «qLd» який змушує компоненти «custom_lpm_counter» вивантажити свої значення у регістри «register», з яких дані будуть зчитані наступним ключовим блоком системи. Останнім з'являється сигнал «qRst» який очищає значення компонентів «custom_lpm_counter».

2.4.4 Блок обробки даних з лічильників

Дані після лічильника розбиті на шість чотирирозрядних значень, даний формат не являється зручним для швидкого аналізу людиною, тому необхідний блок для конвертування даних від лічильників у числа в десятковій системі і запису їх у вигляді ASCII символів [20]. Таким блоком являється «txtgen». Код на мовіопису обладнання – Verilog, для даного компонента наведена у додатку К.

Після підняття командного сигналу «qLd», блок у внутрішній регістр зберігає значення яке вказує на перший розряд першого лічильника:

```
***
reg [96:0]shift;
…
shift<={shift[96:0],qLd};
***
```

До блоку «txtgen» також під'єднаний особистий генератор з частотою 1кГц. Кожний імпульс від генератора зміщує вказівник на один розряд, таким чином за шість імпульсів буде оброблено дані від одного лічильника:

```
***
reg [96:0]shift;
always @(posedge led1)
        shift<={shift[96:0],qLd};
assign wr = (shift!=0);
***
```

Коли вказівник оновлено відбуваєтьсяконвертація чотирирозрядного двійкового значення у ASCII символ:

```
***
if(val<10) ascii = val+8'h30;
elseascii = val + 8'h41 - 10;
***
```

Після конвертації ASCII символ відправляється у чергу де очікує на обробку наступним блоком системи:

```
***
assign byte_wr = shift[0] ? ascii( f1[3:0] ) :
…
shift[93] ? ascii( fh6[3:0] ) :
shift[94] ? 8'h30:
[95] ? 8'h0D :
8'h0A;
***
```

2.4.5 Блок UART передавача

ASCII символи послідовно зчитуються із черги, куди їх поклав «txtgen», і синхронно передаються по UART протоколу [19], рисунок 2.17. Код на мовіопису обладнання – Verilog, для даного компонента наведена у додатку Л.

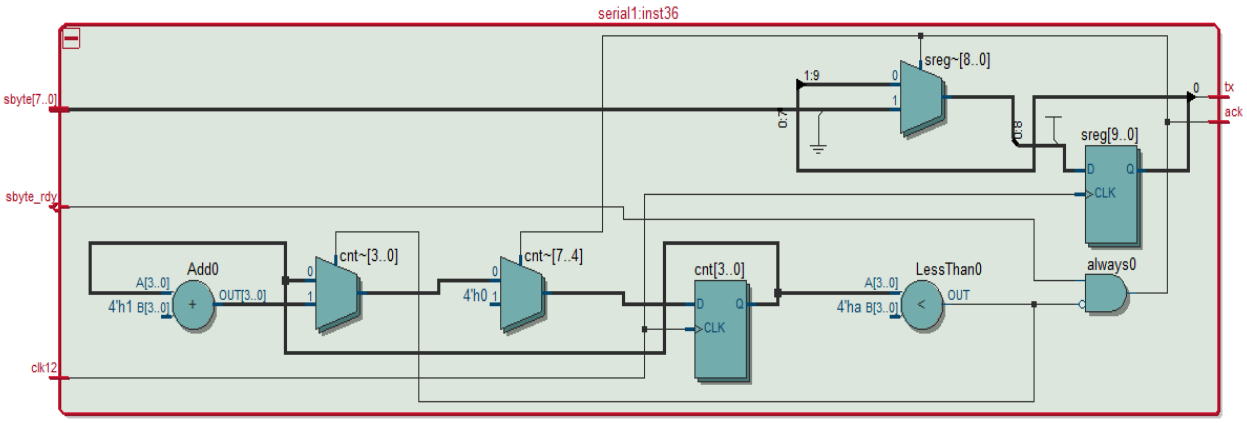


Рисунок 2.17– Схематичне зображення UART передавача

Для реалізації UARTпередавача було створено окремий генератор, з частотою 115 кГц, який дозволяє передавати дані на швидкості 115200 бод, рисунок 2.18.

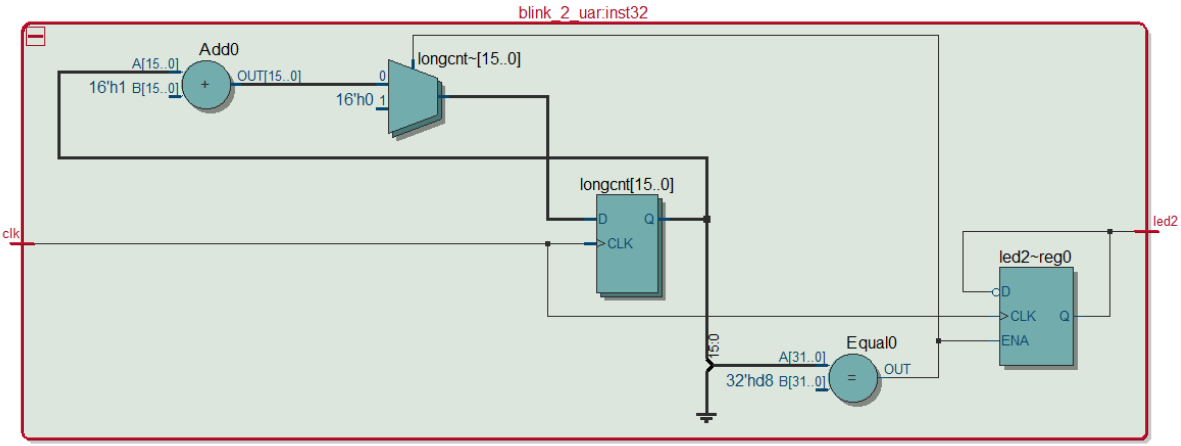


Рисунок 2.18– Схематичне зображення генератора UART передавача

2.4.6 Синтез схеми багатоканального частотоміра

Після реалізації усіх необхідних компонентів, необхідно поєднати їх в одній схемі. На рисунку 2.19 зображене поєднання компонентів які використовуються для передачі даних:

1) «myfifi»- черга для зберігання оброблених даних з лічильників;
2) «blink_2»- генератор UART передавача;
3) «serial1» - UART передавач.

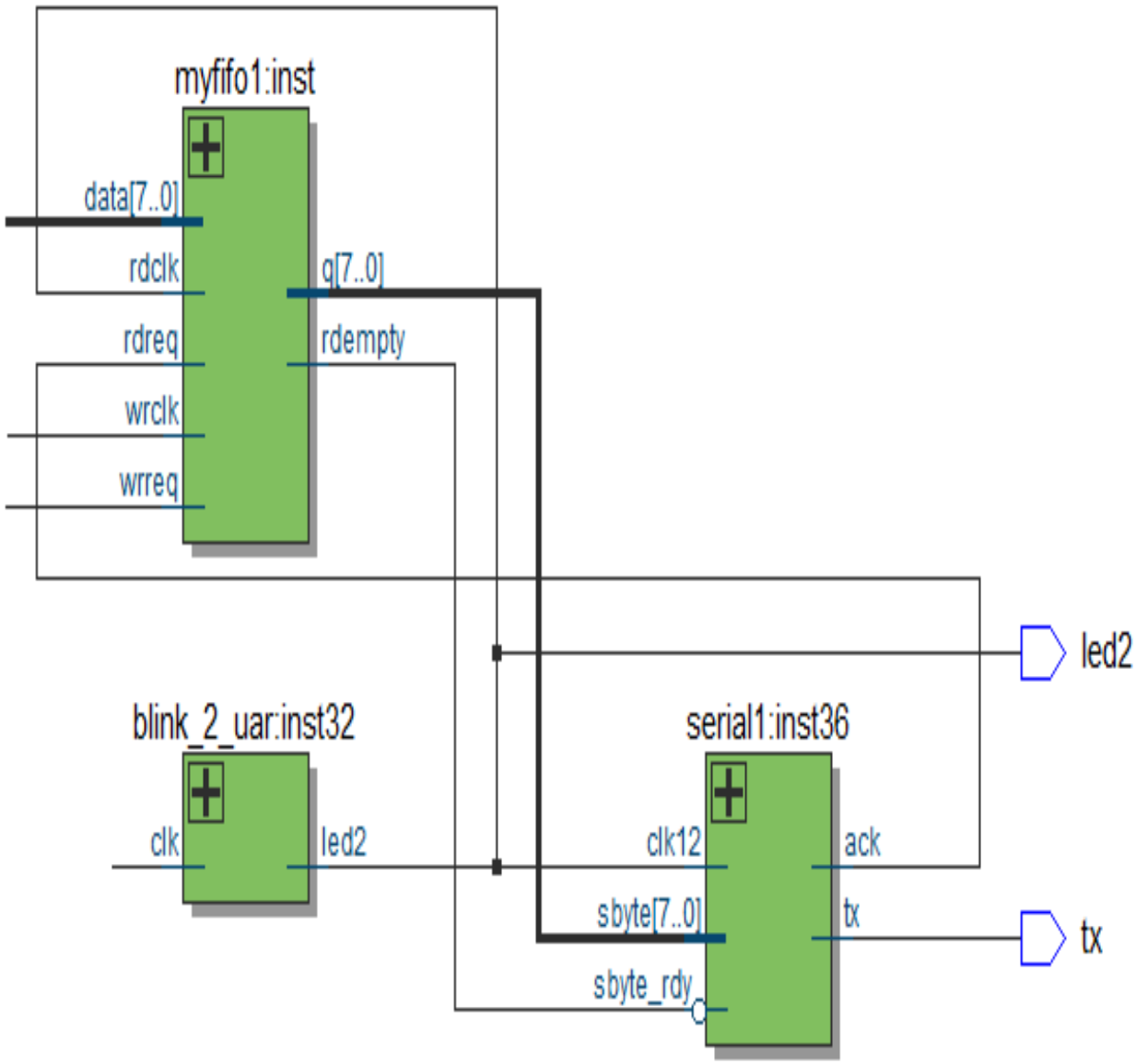


Рисунок 2.19 – Схематичне зображення підсистеми передачі даних

На рисунку 2.20 зображенопоєднання компонентів які використовуються для отримання і обробки даних:

1. «frequency_meter» - лічильник імпульсів (у розробленій схемі використовується 12 лічильників але для відображення було використано лише два);
2. «block_form» – блок керування ключовими вузлами системи;
3. «blink_1» - генератор обробника даних;
4. «txtgen» - блок обробки даних від лічильників.

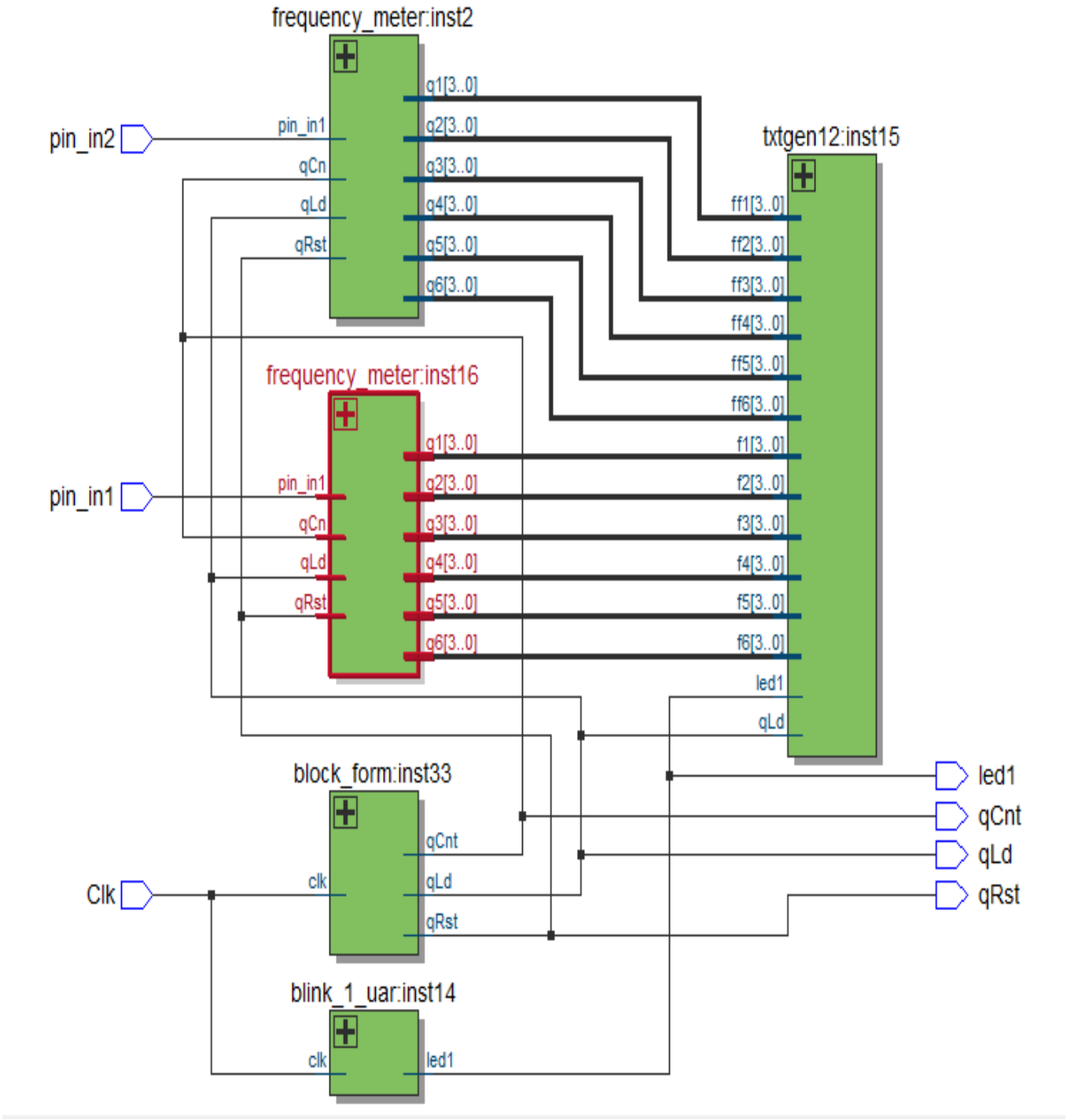


Рисунок 2.20 – Схематичне зображення підсистем підрахунку і обробки даних

На рисунку 2.21 зображенопоєднання компонентів які використовуються для отримання і передачі даних. У додатку М зображена повна схема розробленого пристрою.

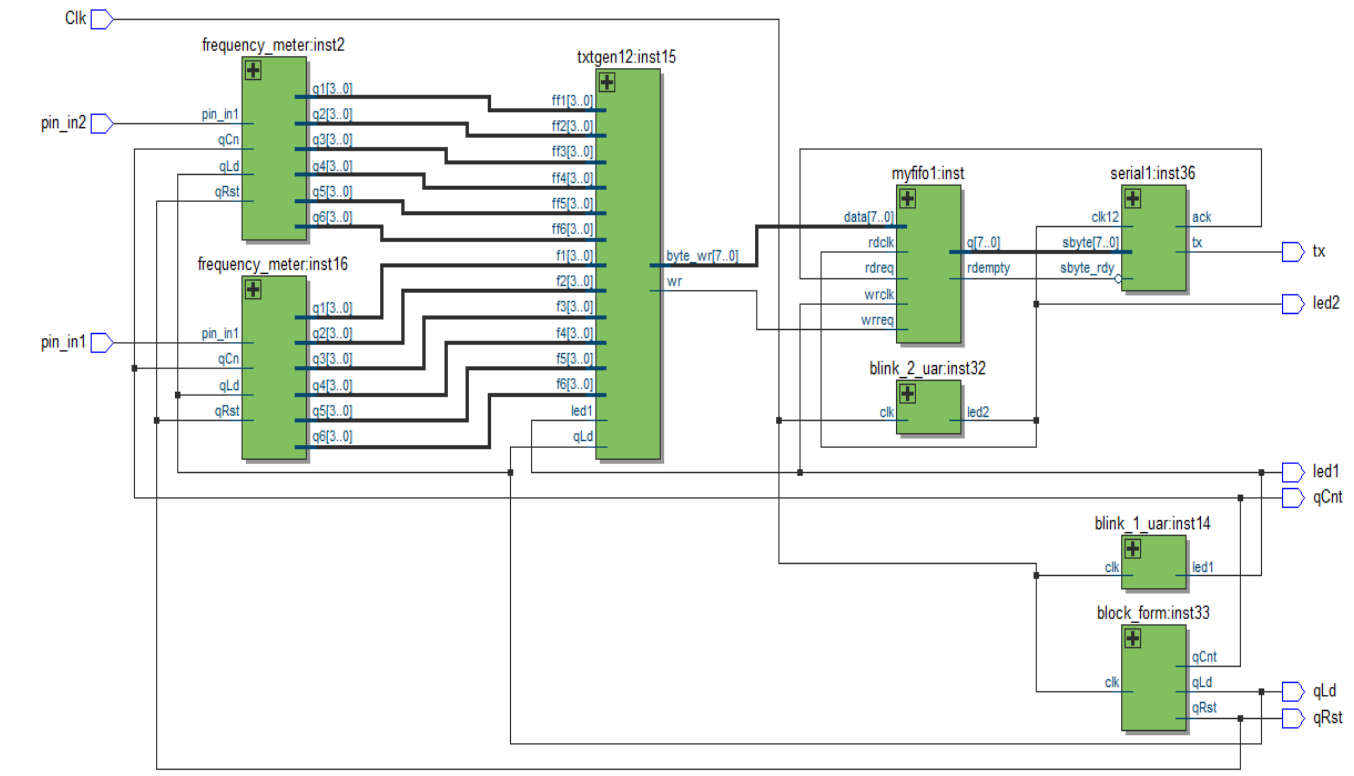


Рисунок 2.21 – Схема багатоканального частотоміра

Основним призначенням розробленої системи являється, одночасне вимірювання значень у сенсорів із частотним виходом [21,26]. На рисунку 2.22 зображено результат роботи багатоканальної вимірювальної системи на FPGA для радіовимірювальних частотних сенсорів при під'єднанні генератора із частотою 85кГц, на рисунку 2.23 при роботі із сенсором тиску.

Рисунок 2.22 – Вимірювання частоти під'єднаного генератора

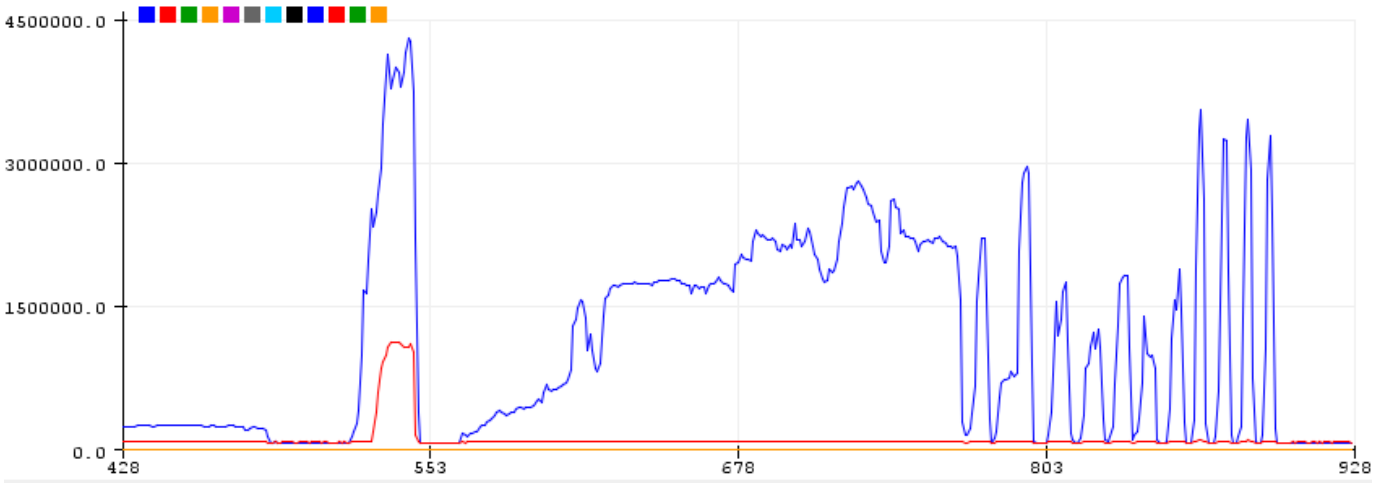


Рисунок 2.23 – Вимірювання значень датчика тиску з частотним виходом

2.5 Висновки до другого розділу

1. Розглянуто принципи роботи які лежать в основі реалізації FPGA.
2. Проаналізовано різні апаратні платформи і випадки їх застосування: CPU, GPU, MCU, FPGA.
3. Розроблено багатоканальну систему вимірювання частоти, основною задачею якої являється вимірювання значень сенсорів з частотним виходом.

3 РОЗШИРЕННЯ ФУНКЦІОНАЛЬНИХ МОЖЛИВОСТЕЙ ПРИЛАДУ

3.1 Ядро мікропроцесора Nios II

Система з процесором Nios II - це еквівалент мікроконтролеру, який містить процесор, комбінацію додаткових підсистем і пам'яті в одному чіпі [28]. Така система складається з ядра процесора Nios II і набору допоміжних підсистем на чіпі, рисунок 3.1. Аналогічно сімейству мікроконтролерів, система з процесором Nios II використовує постійний набір інструкцій і модель програмування.

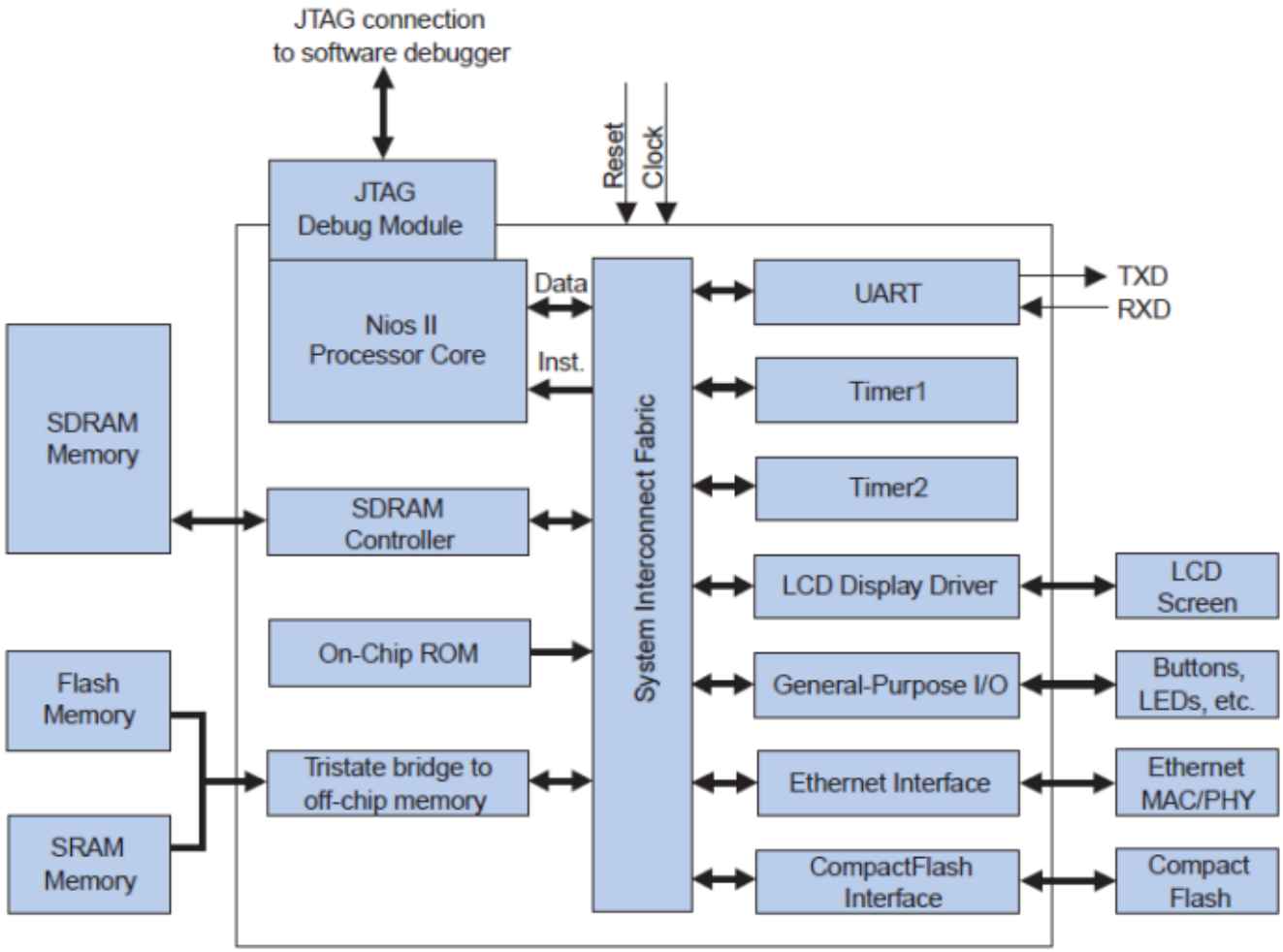


Рисунок 3.1 – Приклад мікропроцесорної системи Nios II

Процесор Nios II - це процесор з конфігурованим програмним ядром, на противагу фіксованому ядру готового мікроконтролера. У цьому контексті, конфігурованість дає можливість додавати або видаляти функціональні блоки,

щоб вирішувати завдання продуктивності або вартості системи. Програмне ядро дозволяє не обмежуватися певною апаратною частиною, а розміщуватися в будь-яких чіпах сімейств Altera FPGA. Гнучкий набір допоміжних систем - одна з найголовніших відмінностей між процесорною системою Nios II і фіксованими мікроконтролерами. Оскільки процесор Nios II реалізується програмованою логікою, є можливість створити нестандартну процесорну систему з точним набором допоміжних підсистем, необхідних для вирішення задачі. Altera надає досить великий набір такого роду підсистем, наприклад: таймери, послідовні інтерфейси, стандартні I/O, контролери SDRAM.

Також є можливість створення особистих допоміжних підсистем і їх інтеграції у вже існуючі процесорні системи. Для критичних за характеристиками систем, які витрачають велику кількість внутрішніх циклів для виконання частини коду, існує можливість створення власного функціонального блоку, який реалізує ту ж функцію в апаратному вигляді. Перевагами такого підходу є: апаратна частина швидше програмної; процесор звільняється для виконання інших функцій. Процесор Nios II підтримує створення нових інструкцій, користувачем, що дозволяє краще налаштувати систему під вирішення необхідної задачі. Власна логіка інтегрується в арифметико-логічний пристрій (ALU) процесора Nios II. Як і звичайні інструкції Nios II, логіка власних інструкцій бере значення з двох регістрів і опціонально повертає результат в призначений регістр. Створені інструкції застосовуються як згенерований асемблерний макрос або C функції.

Інструмент розробки Altera's SOPC Builder повністю автоматизує процес конфігурації засобів процесора і генерування апаратної частини проекту. Графічна оболонка (GUI) SOPC Builder дозволяєте конфігурувати процесорні системи Nios II з будь-якою кількістю допоміжних систем. Є можливість створювати процесорні системи у вигляді блок-схеми або HDL проекту.

3.1.1 Архітектура ядра

Архітектура Nios II представлена структурою системних команд (ISA). ISA оточена необхідним набором функціональних модулів, які виконують інструкції. Ядро процесора Nios II - це апаратний проект, який реалізує набір інструкцій і підтримує функціональні модулі. Ядро процесора не має допоміжних підсистем або логіки підключення до зовнішнього світу. Воно містить тільки схеми реалізації архітектури Nios II, рисунок 3.2.

Архітектура NiosII визначається наступними функціональними модулями:

1. Регістровий файл;
2. Арифметико-логічний пристрій (ALU) ;
3. Інтерфейс з логікою користувальницьких інструкцій;
4. Внутрішній або зовнішній контролер переривань;
5. Шина даних;
6. Диспетчер пам'яті (MMU) ;
7. Елемент захисту пам'яті (MPU) ;
8. Кеш пам'ять під інструкції і дані;
9. Інтерфейс пам'яті для інструкцій і даних.

Функціональні модулі архітектури Nios II формують основний набір інструкцій [29]. Однак, вони не відображаються при апаратній реалізації. Архітектура описується набором інструкцій, без будь-якої апаратної реалізації. Функціональні модулі можуть бути апаратно реалізовані, імітовані програмно або повністю пропущені. Реалізація - це вибір з набору проектів об'єднаних в специфічному ядрі Nios II. Кожна реалізація отримує на виході специфічні об'єкти, наприклад, ядра маленькі за розміром або високопродуктивні ядра. Це дозволяє адаптувати архітектуру Nios II відповідно до потреб поставлених задач. В кінцевій реалізації вибирається один з трьох компромісних варіантів: більша чи менша продуктивність, включення або виключення допоміжних функцій, апаратна реалізація або програмна емуляція.

Проілюструємо прикладами кожен варіант:

1. Більша або менша продуктивність - наприклад, для точного виконання, можна збільшувати або зменшувати кількість інструкцій в кеш пам'яті. Великий кеш збільшує швидкість виконання великих програм, тоді як малий кеш зберігає ресурси пам’яті всередині FPGA.

2. Включення або виключення допоміжних функцій - наприклад, для заощадження LE, можна не включати налагоджуючи модуль JTAG. У такому випадку зберігаються логічні блоки і ресурси пам’яті, але з’являється необхідність використання програмного налагоджуючого модуля.

3. Апаратна реалізація або програмна емуляція - наприклад, для додатків які рідко виконують складні арифметичні операції, є можливість вибрати

програмну емуляцію інструкцій ділення. Видалення апаратного ділення зберігає ресурси FPGA, але збільшує час виконання операції.

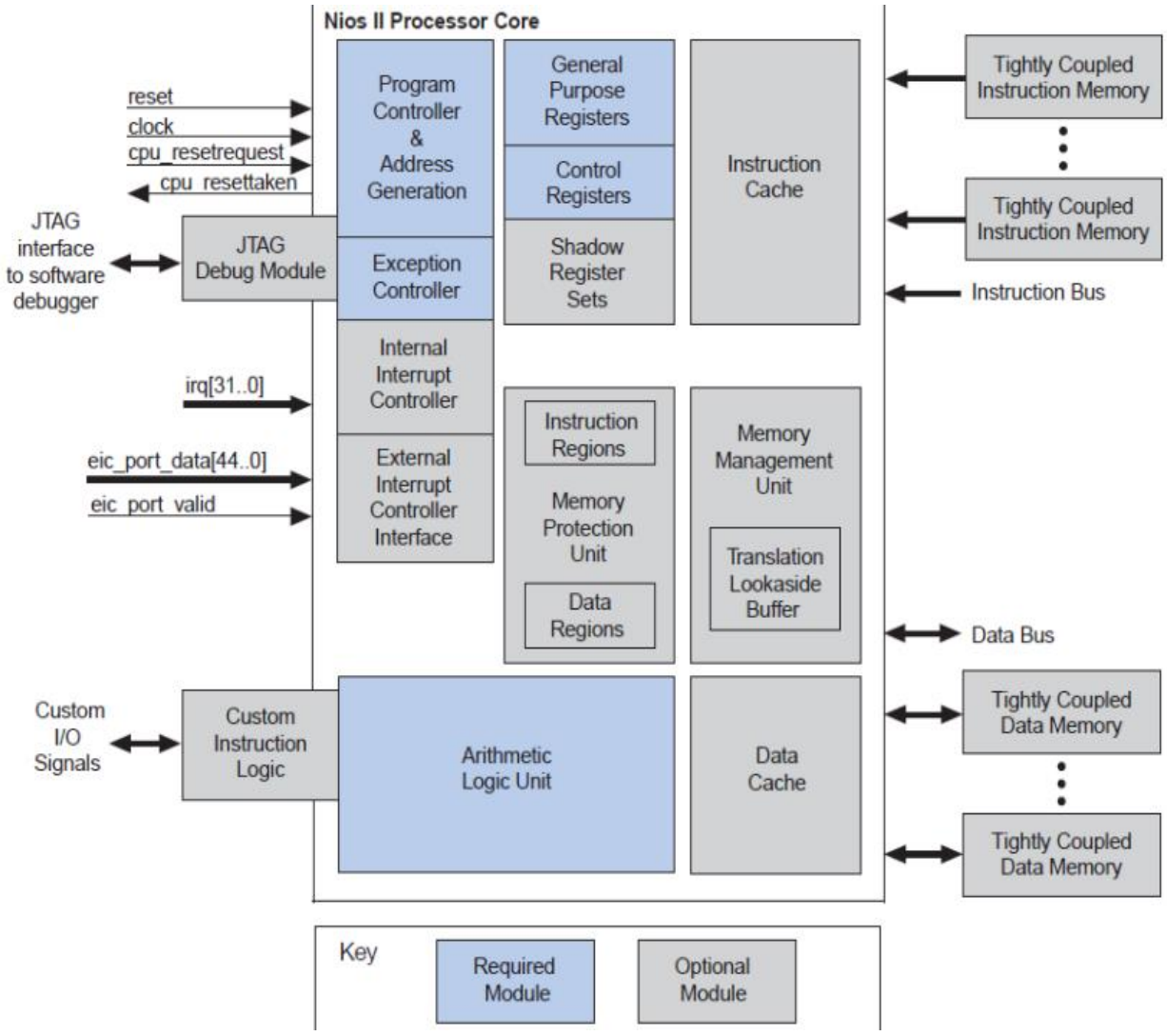


Рисунок 3.2 – Блок-схема ядра процесора Nios II

Архітектура Nios II підтримує однорідний регістровий файл, що складається із здвоєних 32-бітних цілочисельних регістрів загального призначення і додаткових здвоєних 32-бітних контрольних регістрів. Архітектура підтримує привілейований і призначений для користувача режими, які дозволяють системному коду захищати контрольні регістри. Процесор може опціонально мати один або кілька наборів тіньових регістрів. Набір тіньових регістрів це повний набір регістрів загального призначення. Коли реалізується набір тіньових регістрів, поле CRS статусного регістра показує поточний набір використовуваних регістрів. Доступ інструкцій до регістру загального

призначення використовує будь-який набір активних регістрів. Типове використання набору тіньових регістрів - прискорення контекстного перемикання. Процесор має дві спеціальні інструкції, «rdprs» і «wrprs», для переміщення даних між наборами регістрів. Набір тіньових регістрів зазвичай керується ядром операційної системи, тому не помітний в коді додатків. Процесор може мати до 63 наборів таких регістрів.

Nios II ALU оперує з даними, що лежать в регістрах загального призначення [30]. ALU оперує з одним або двома входами регістра і повертає результат у визначений регістр. Операції над даними які підтримуються ALU, описані в таблиці 3.1.

Таблиця 3.1 – Операції які підтримуються ALU

Категорія	Опис
Арифметичні	Додавання, віднімання, множення і ділення знакових і без знакових операндів.
Відношення	Рівність, нерівність, більше або дорівнює, менше або дорівнює (==,! => =, <) знакових і без знакових операндів.
Логічні	AND, OR, NOR і XOR.
Зсув і циклічність	Зсувні і циклічні операції, підтримка зсуву даних від 0 до 31 позиції в інструкції, зсув вправо і логічний зсув вправо/вліво, циклічний зсув вправо /вліво.

Деякі реалізації ядра процесора не мають апаратної підтримки певного набору інструкцій. Таким чином, інструкції ядра без апаратної підтримки відомі як одиночні інструкції. Процесор генерує виключення, коли потрапляє на одиночну інструкцію і викликає обробник виключень.

Архітектура ядра підтримує інструкції з плаваючою точкою одинарної точності, як це визначено в IEEE Std 754 – 1985 [31]. У базовий набір власних інструкцій з плаваючою точкою входять: додавання, віднімання і множення. Ці інструкції з плаваючою точкою розглядаються як власні інструкції. У таблиці 3.2 наводиться докладний опис відповідності з IEEE 754 - 1985.

Таблиця 3.2 – Інструкції які підтримує ядро

Характеристики		Реалізація
Операції	Додавання	Реалізовано
	Віднімання	Реалізовано
	Множення	Реалізовано
	Ділення	Опціонально
Точність	Одинарна	Реалізовано
	Подвійна	Не реалізовано. Подвійна точність операцій реалізується програмно.
Виключна ситуація	Неправильна операція	Результат - немає номера (NaN)
	Ділення на нуль	Результат - ±∞
	Переповнення	Результат - ±∞
	Не точність	Результат – нормальний номер
	Зникнення даних	Результат - ±0
Округлення	До найближчого значення	Реалізовано
	До нуля	Не реалізовано
	До +∞	Не реалізовано
	До -∞	Не реалізовано
NaN	Виправлення	Реалізовано
	Сигнальний	Не реалізовано
Денормалізовані числа		Денормалізовані оператори розглядаються як нульові. Власні інструкції не генерують денормалізовані числа

Найкращим вибором для апаратного проекту буде баланс між використанням плаваючої точки, використанням апаратних ресурсів і робочими характеристиками. Власні інструкції з плаваючою точкою прискорюють арифметику, але це призводить до збільшення розмірів апаратного проекту. Якщо використання ресурсів обмежене, доведеться переробляти алгоритми для мінімізації арифметики із плаваючою точкою.

3.2 Послідовна шина даних I2C

Кожен, хто займався розробкою радіоелектронної техніки, стикався з ситуацією, коли для співставлення рівнів сигналів, вибірки і адресації функціонально-закінчених вузлів, доводиться використовувати величезну

кількість проміжних інтегральних схем (ІС). Для збільшення ефективності, спрощення схемотехніки, Philips [32] розробила просту двосторонню двопровідну шину для між мікросхемного (inter-IC) управління. Шина отримала назву - InterIC, або IIC (I2C) шина. В даний час тільки Philips виробляє більше 150 найменувань I2C-сумісних пристроїв, функціонально призначених для роботи в електронному обладнанні різного призначення. У їх числі ІС пам'яті, відеопроцесорів і модулів обробки аудіо та відео-сигналів, АЦП і ЦАП, процесори з вбудованим контролером I2C шини.

I2C шина є однією з модифікацій послідовних протоколів обміну даних. У стандартному режимі забезпечується передача послідовних 8-бітних даних зі швидкістю до 100 кбіт/с, і до 400 кбіт/с в "швидкому" режимі. Для здійснення процесу обміну інформацією по I2C шині, використовується всього два сигнали лінія даних SDA лінія синхронізації SCL. Для забезпечення реалізації двобічної шини без застосування складних арбітрів шини, вихідні каскади пристроїв, підключені до шини, мають відкритий стік або відкритий колектор для забезпечення функції монтажного "I" . Проста двопровідна послідовна шина I2C мінімізує кількість з’єднань між ІС, що призводить до зменшення об’єму комунікаційних сполучень. Як результат - друковані плати стають простішими і технологічними при виготовленні. Інтегрований I2C - протокол усуває необхідність в дешифраторах адрес та інших зовнішніх логічних узгодженнях. Максимальна допустима кількість мікросхем, приєднаних до однієї шини, обмежується максимальною ємністю шини яка становить 400 пФ.

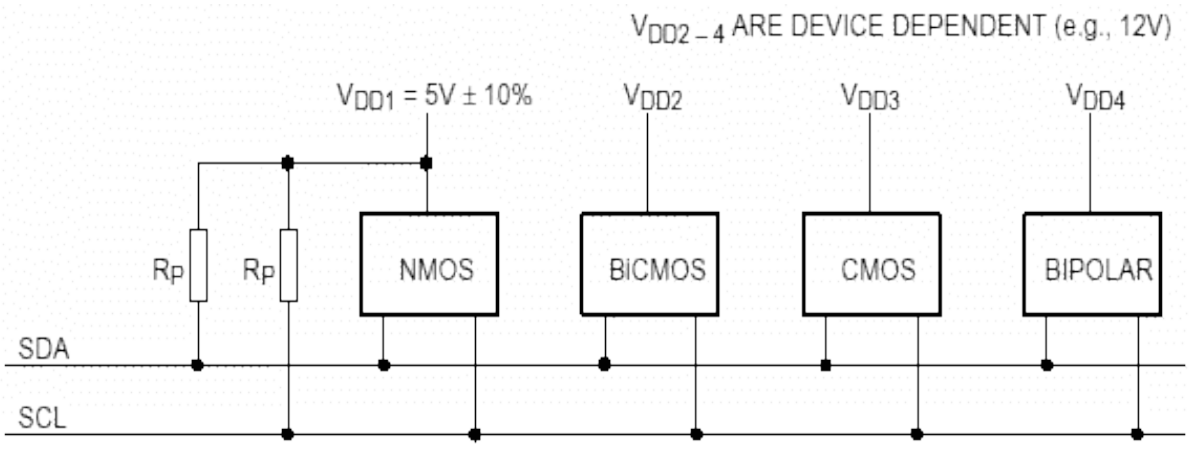


Рисунок 3.3 – Підключення декількохпристроїв до спільної шини

Вбудований в мікросхеми апаратний алгоритм завадостійкості забезпечує цілісність даних в умовах перешкод значної величини. Всі I2C-сумісні пристрої мають інтерфейс, який дозволяє їм зв'язуватися один з одним по шині навіть у тому випадку, якщо їх напруга живлення істотно відрізняється. На рисунку 3.3 представлений принцип підключення декількох ІС з різними напругами живлення до однієї шини обміну.

Кожен пристрій розпізнається за унікальним адресом і може працювати як передавач або приймач, в залежності від призначення пристрою. Крім того, пристрої можуть бути класифіковані як ведучі і ведені при передачі даних. Ведучий - це пристрій, який ініціює передачу даних і виробляє сигнали синхронізації. При цьому будь-який пристрій, що адресується вважається веденим по відношенню до ведучого Виходячи з специфікації роботи шини, в кожен окремий момент в шині може бути тільки один ведучий, а саме той пристрій, що забезпечує формування сигналу SCL шини. Ведучий може виступати як в ролі ведучого-передавача, так і ведучого-приймача. Проте - шина дозволяє мати кілька ведучих, накладаючи певні особливості їхньої поведінки в формуванні сигналів управління і контролю стану шини. Можливість підключення більше одного ведучого до шини означає, що більш ніж один ведучий може спробувати почати пересилання в один і той же момент часу. Для усунення колізій, які можуть виникнути в даному випадку, розроблена процедура арбітражу - поведінки ведучого при виявленні колізії на шині. Ця процедура заснована на тому, що всі I2C - пристрої підключаються до шини за правилом монтажного І. В початковому стані обидва сигнали SDA і SCL знаходяться у високому стані.

Процедура обміну починається з того, що ведучий формує стан СТАРТ - ведучий генерує перехід сигналу лінії SDA з високого стану в низький при високому рівні на лінії SCL. Цей перехід сприймається усіма пристроями, підключеними до шини як ознака початку процедури обміну, рисунок 3.4.

Генерація синхросигналу - це завжди обов'язок ведучого; кожен ведений генерує свій власний сигнал синхронізації при пересиланні даних по шині. Процедура обміну завершується тим, що ведучий формує сигнал СТОП - перехід стану лінії SDA з низького у високий при високому стані лінії SCL

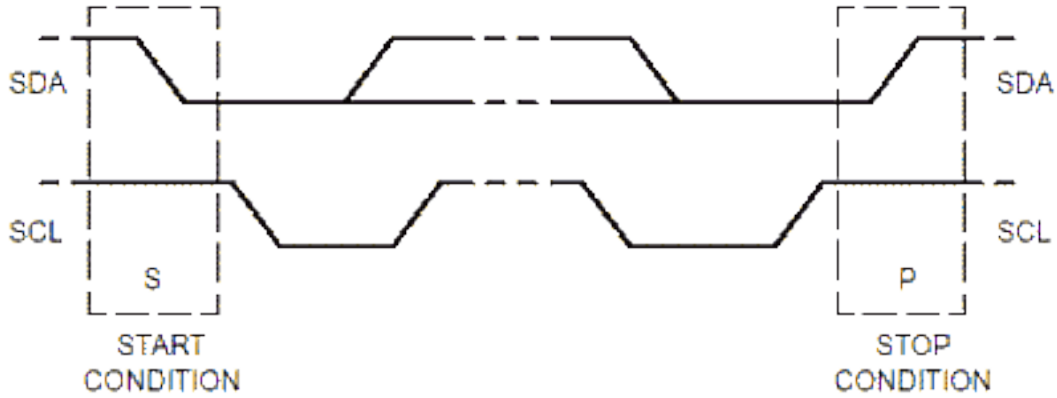


Рисунок 3.4 – Приклад інформаційних сигналів СТАРТ і СТОП

При передачі пакетів по шині I2C кожен ведучий генерує свій синхросигнал на лінії SCL. Після формування стану СТАРТ, ведучий опускає стан лінії SCL в низький стан і виставляє на лінію SDA старший біт першого байта повідомлення. Кількість байт в повідомленні не обмежена. Специфікація шини I2C допускає зміни на лінії SDA тільки при низькому рівні сигналу на лінії SCL, рисунок 3.5.

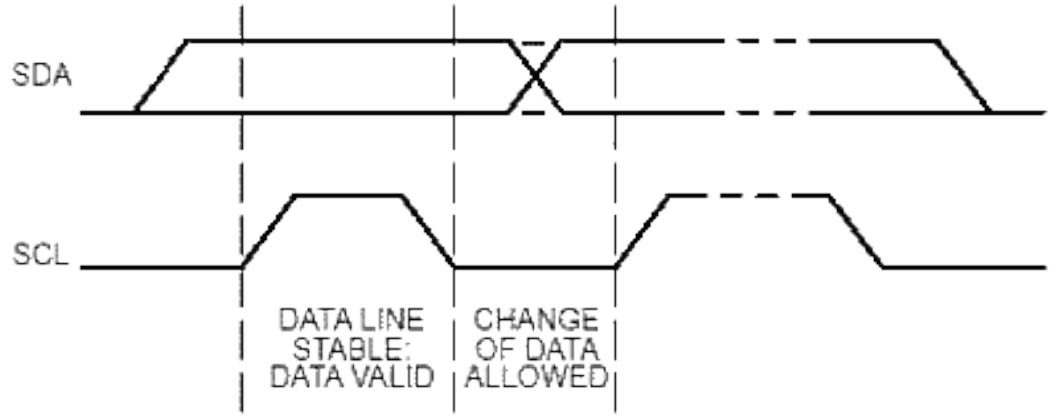


Рисунок 3.5 – Ілюстрація зміни сигналу на шині

3.2.1 Підтвердження

Дані дійсні і повинні залишатися стабільними тільки під час високого стану синхроімпульса. Для підтвердження прийому байта від ведучого передавача, веденим приймачем в специфікації протоколу обміну по шині I2C вводиться спеціальний біт підтвердження, який виставляється на шину SDA після прийому 8 біта даних. Таким чином передача 8 біт даних від передавача до приймача завершуються додатковим циклом (формуванням 9-го тактового імпульсу лінії

SCL), при якому приймач виставляє низький рівень сигналу на лінії SDA, як ознака успішного прийому байта, рисунок 3.6.

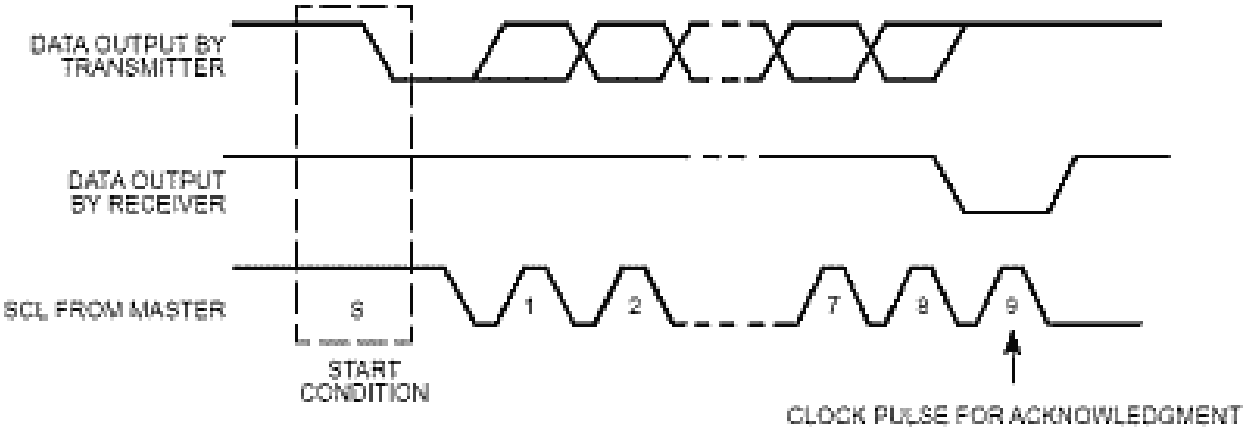


Рисунок 3.5 – Підтвердження прийому пакета веденим пристроєм

Підтвердження при передачі даних обов'язкове. Відповідний імпульс синхронізації генерується веденим. Передавач відпускає лінію SDA на час синхроімпульса підтвердження. Приймач повинен утримувати лінію SDA протягом високого стану синхроіпульса підтвердження в стабільному низькому стані. У тому випадку, коли ведений-приймач не може підтвердити свою адресу (наприклад, коли він виконує в даний момент будь-які функції реального часу), лінія даних повинна бути залишена у високому стані. Після цього ведучий може видати сигнал СТОП для переривання, повторного пересилання даних. Якщо в передачі бере участь ведучий-приймач, то він повинен повідомити про закінчення передачі веденому-передавачу шляхом непідтвердження останнього байта. Ведений-передавач повинен звільнити лінію даних для того, щоб дозволити ведучому видати сигнал СТОП або повторити сигнал СТАРТ. Через особливості правил синхронізації на шині, ведучий не має монопольного права на управління переходом лінії SCL з низької стану у високого. У тому випадку, коли введеному необхідний додатковий час на обробку прийнятого біта, він має можливість утримувати лінію SCL в низькому стані до моменту готовності до прийому наступного біта. Таким чином, лінія SCL буде перебувати в низькому стані протягом найдовшого низького періоду синхросигналу.

Пристрої з більш коротким низькими періодом будуть входити в стан очікування на час, поки не скінчиться довгий період. Коли у всіх задіяних

пристроїв скінчиться низький період синхросигналу, лінія SCL перейде в високий стан. Всі пристрої почнуть проходити високий період своїх синхросигналів. Перший пристрій, у якого скінчиться цей період, знову встановить лінію SCL в низький стан. Таким чином, низький період синхролінії SCL визначається найтривалішим періодом синхронізації з усіх задіяних пристроїв, а високий період визначається найкоротшим періодом синхронізації пристроїв. Механізм синхронізації може бути використаний приймачами як засіб управління пересиланням даних на байтовому і бітовому рівнях. На рівні байта, якщо пристрій може приймати байти даних з великою швидкістю, але вимагає певний час для збереження прийнятого байта або підготовки до прийому наступного, то він може утримувати лінію SCL в низькому стані після прийому і підтвердження байта, переводячи таким чином передавач в стан очікування. На рівні бітів, пристрій такий як мікроконтролер без вбудованих апаратних ланцюгів I2C або з обмеженими ланцюгами може уповільнити частоту синхроімпульсів шляхом продовження їх низького періоду. Таким чином швидкість передачі будь-якого ведучого адаптується до швидкості повільного пристрою.

3.2.2 Адресація

Кожен пристрій, підключений до шини, може бути програмно адресованим по унікальному адресу. Для вибору приймача повідомлення ведучий використовує адресний компонент в форматі посилки. При використанні однотипних пристроїв, ІС часто мають додатковий селектор адреси, який може бути реалізований як у вигляді додаткових цифрових входів селектора адреси, так і у вигляді аналогового входу. При цьому адреси таких однотипних пристроїв виявляються рознесені в адресному просторі пристроїв, підключених до шини. У звичайному режимі використовується 7 -бітова адресація. Процедура адресації на шині I2C полягає в тому, що перший байт після сигналу СТАРТ визначає, який ведений адресується провідним для проведення циклу обміну. Виняток становить широкомовна адреса, яка адресує всі пристрої на шині. Коли використовується ця адреса, всі пристрої в теорії повинні послати сигнал підтвердження. Перші сім бітів першого байта утворюють адрес веденого. Восьмий, молодший біт, визначає напрямок пересилки даних. Нуль означає, що ведучий буде записувати інформацію в обраного веденого. "Одиниця" означає, що ведучий буде зчитувати інформацію з веденого.

Після того, як адресу надіслано, кожен пристрій в системі порівнює перші сім біт після сигналу СТАРТ зі своєю адресою. При збігу пристрій вважає себе обраним як ведений-приймач або як ведений-передавач, в залежності від біта напрямку. Адреса веденого може складатися з фіксованої і програмованої частини. Часто трапляється, що в системі буде кілька однотипних, тому за допомогою програмованої частини адреси, стає можливим підключити до шини максимально можливу кількість таких пристроїв. Всі ІС, що підтримують роботу в стандарті шини I2C, мають набір фіксованих адрес, перелік яких зазначений виробником в документації. У загальному вигляді процес обміну пакетами по шині, від моменту формування стану СТАРТ до стану СТОП проілюстрований на рисунку 3.6.

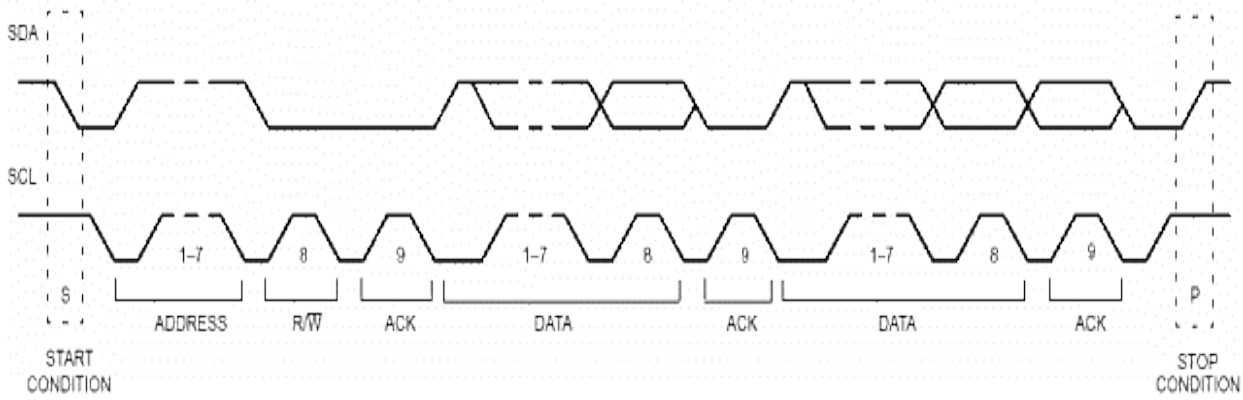


Рисунок 3.6 – Сеанс передачі одного пакета

Як випливає із специфікації шини, допускаються як прості формати обміну, так і комбіновані, коли в проміжку від стану СТАРТ до стану СТОП ведучий і ведений можуть виступати як приймачем так і передавачем даних. Комбіновані формати можуть бути використані, наприклад, для керування послідовною пам'яттю. Під час першого байта даних можна передавати адресу в пам'яті, яка записується у внутрішній тимчасовий регістр. Після повторення сигналу старту і адреси веденого видаються дані з пам'яті. Всі рішення про авто-інкремент або декремент адреси, до якої стався попередній доступ, приймаються розробником конкретного пристрою. Тому, в будь-якому випадку кращий спосіб уникнути неконтрольованої ситуації на шині, перед використанням нової ІС слід ретельно вивчити її документацію.

3.3 Інтеграція Nios II в систему багатоканального частотоміра

Для інтеграція мікропроцесорної системи в існуючу схему частотоміра потрібно частково переробити блоки: лічильника імпульсів, обробки даних, UART передавача і повторно синтезувати схему. Першим етапом являється генерування і налаштування мікропроцесорної системи Nios II.

3.3.1 Створення мікропроцесорної системи Nios II

Генерація і налаштування мікропроцесорної системи відбувається за допомогою утиліти Qsys. Першим елементом який потрібно налаштувати являється «Clock Source», він визначає робочу частоту системи, рисунок 3.7.

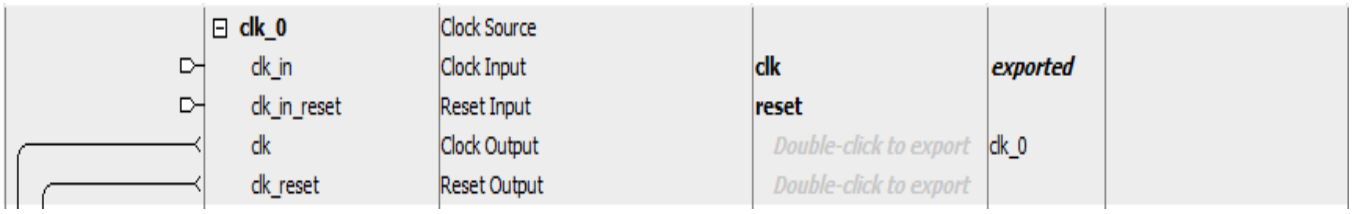


Рисунок 3.7 – Графічне відображення елемента «Clock Source»

На платі разом з FPGA знаходиться кварцовий резонатор на 50 МГц , він буде використаний у якості вхідного генератора для розроблюваної мікропроцесорної системи, у додатку Н зображена плата на базі якої розроблюється процесорна система. В елементі «Clock Source» виставляємо у якості робочої частоти 50 МГц, рисунок 3.8

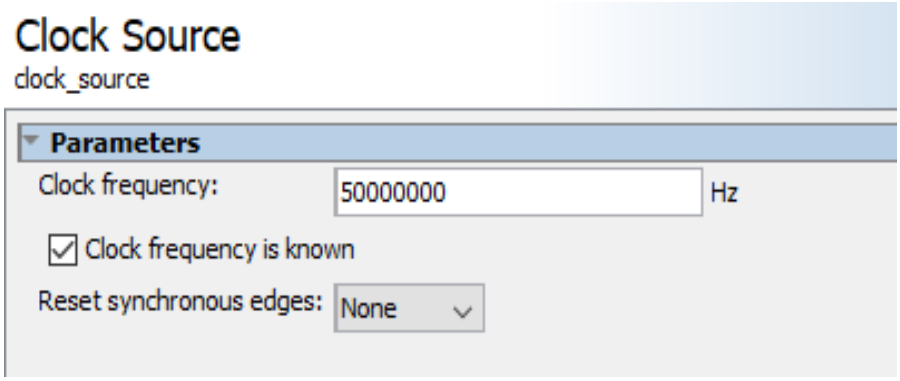


Рисунок 3.8– Параметри налаштування елемента «Clock Source»

«On-Chip Memory» - використовується у якості блоку пам’яті RAM для мікропроцесорної системи, рисунок 3.9. Розмір слова відповідає 32 бітам. Загальний розмір RAM пам’яті дорівнює 20 кБ, рисунок 3.10.

Рисунок 3.9– Графічне відображення елемента «On-Chip Memory»

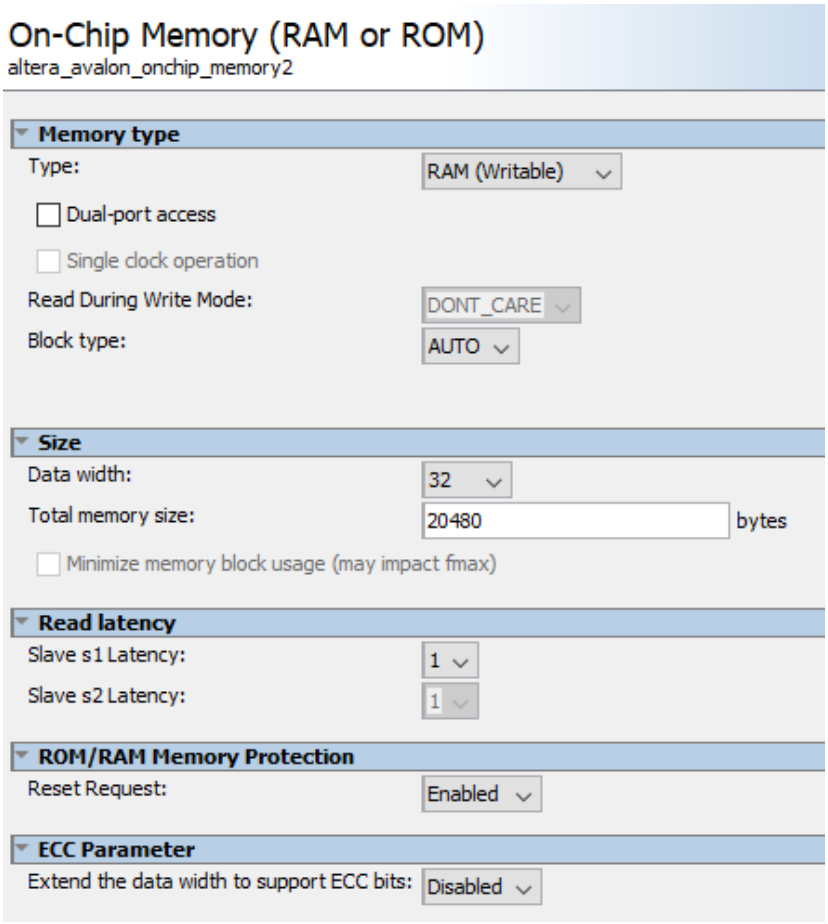


Рисунок 3.10– Параметри налаштування елемента «On-Chip Memory»

«UART» - використовується для взаємодії із зовнішніми пристроями черезпослідовний порт, рисунок 3.11. Основні налаштування: швидкість передачі 115200 Бод, автоматичний контроль цілісності пакета вимкнений, розмір даних в одному пакеті становить 8 біт, рисунок 3.12.

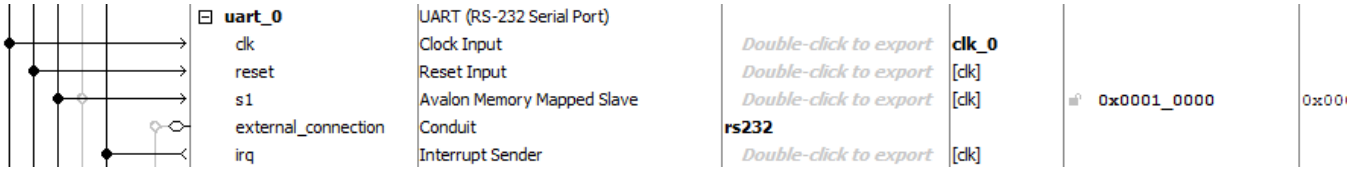


Рисунок 3.11– Графічне відображення інтерфейсу «UART»

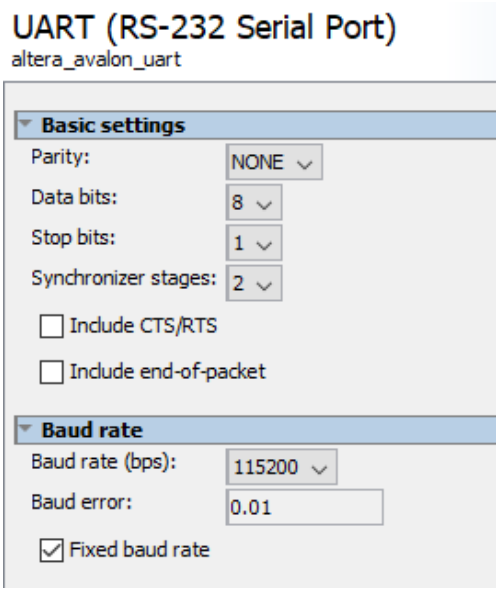


Рисунок 3.12– Параметри налаштування інтерфейсу «UART»

«I2C» - інтерфейс для підключення блоку який реалізує I2C протокол, рисунок 3.13, інтерфейс розділений на дві частини: 10 розрядна вихідна шина з підтримкою переривань (на стороні процесора); 17 розрядна вхідна шина, рисунок 3.14.

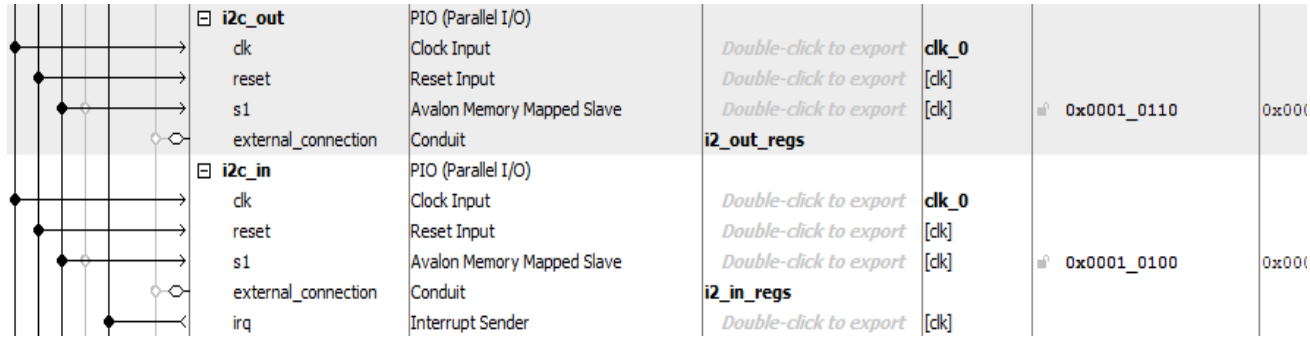


Рисунок 3.13– Графічне відображення інтерфейсу для блоку «I2C»

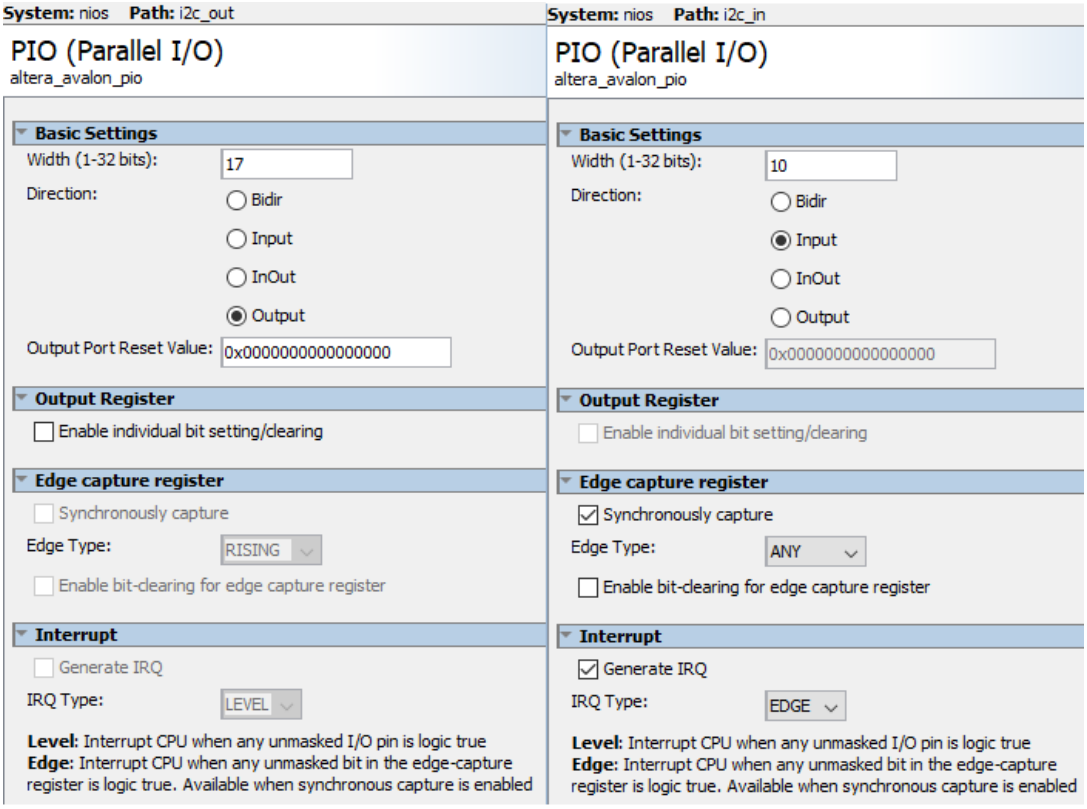


Рисунок 3.14– Параметри налаштування інтерфейсу для блоку «I2C»

«Counter» - використовується для підрахунку часу роботи системи і синхронізації операцій пов’язаних з вимірюваннями, рисунок 3.15. Реалізація даного блоку відсутня у стандартній бібліотеці, тому, він був реалізований самостійно, у якості інтерфейсу використовується вихідна 32-х розрядна шина, рисунок 3.16. «Counter» розроблений у вигляді звичайного регістра, який змінює своє значення з кожним вхідним імпульсом від генератора. Для вимірювання часу, частота від генератор пропускається через подільник який визначає інтервал часу якому буде відповідати одне значення регістра, рисунок 3.17. Для вимірювання 1мс необхідно виставити подільник у значення 1000. Оскільки частота основного резонатора 50 МГц, то одне значення регістра відповідає 50000 тактів генератора.

Рисунок 3.15– Графічне відображення інтерфейсу для блока «Counter»

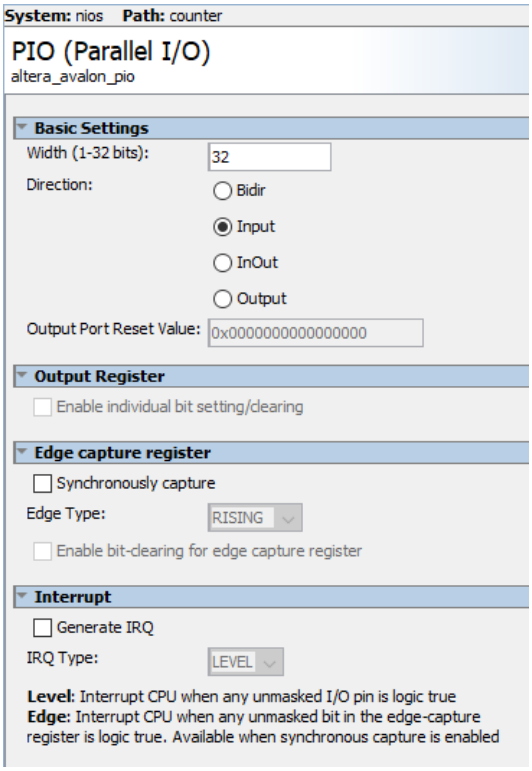


Рисунок 3.16– Параметри налаштування інтерфейсу для блока «Counter»

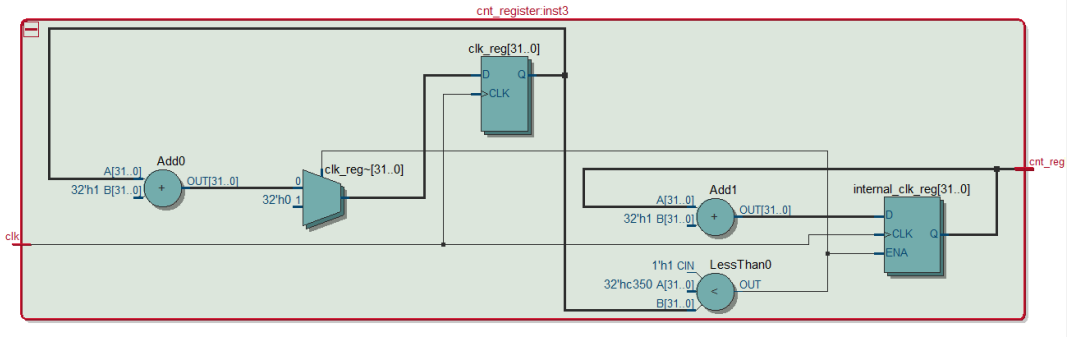


Рисунок 3.17– Схематичне зображення лічильника часу

«Sys IRQ» - інтерфейс для під’єднання зовнішніх джерел переривання, рисунок 3.18. Підтримується до 8 зовнішніх джерел переривання, рисунок 3.19.

Рисунок 3.18– Графічне відображення інтерфейсу «Sys IRQ»

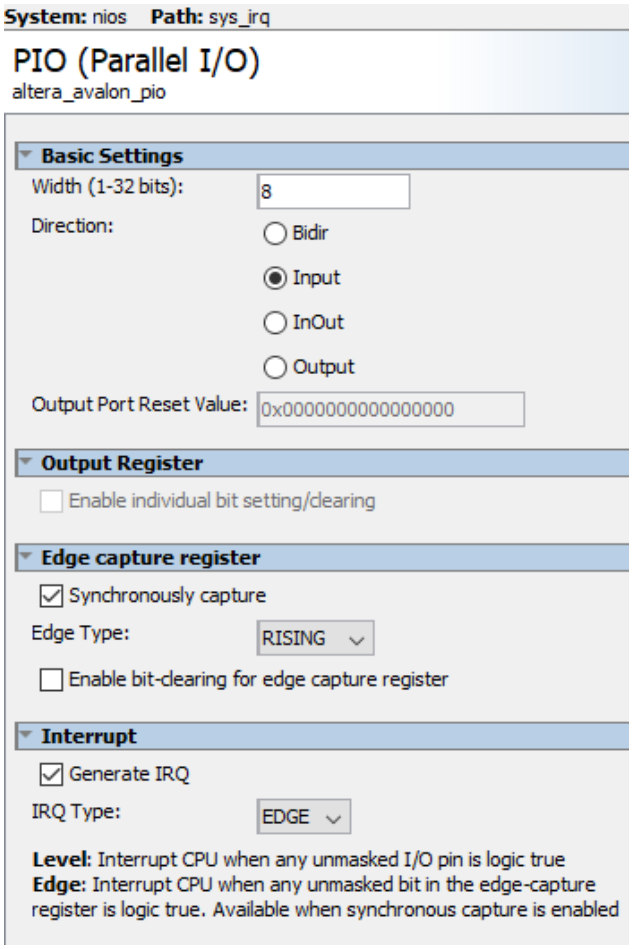


Рисунок 3.19– Параметри налаштування інтерфейсу «Sys IRQ»

«FM» - інтерфейс для під’єднання блоку який реалізує вимірювання частоти, загалом підтримується до 12 частотомірів, рисунок 3.20. Один блок під’єднується за допомогою 32-х розрядної вхідної шини, рисунок 3.21.

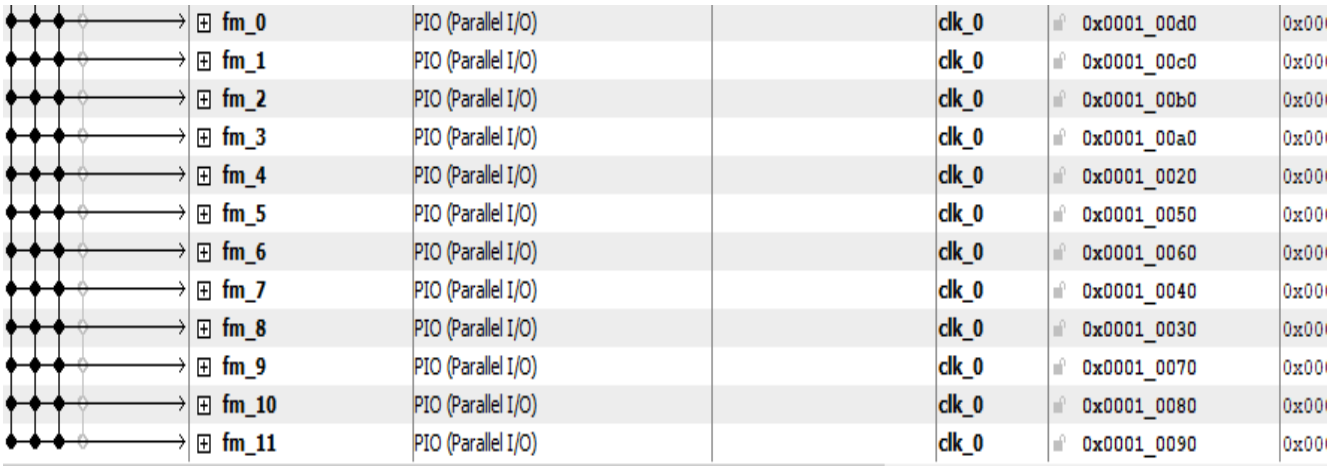


Рисунок 3.20– Графічне відображення інтерфейсу для блоків «FM»

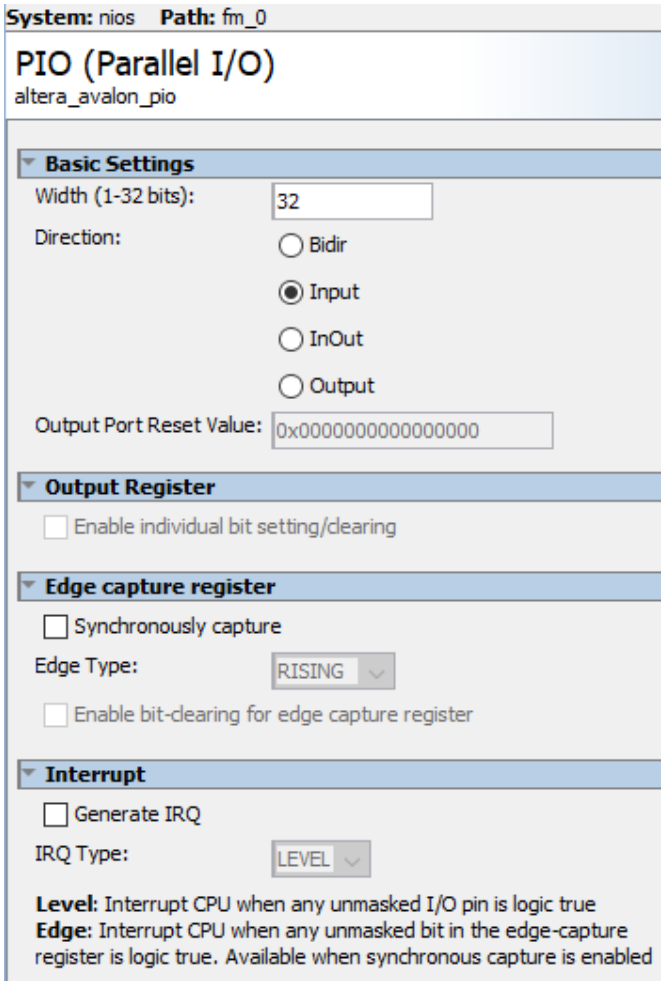


Рисунок 3.21– Параметри налаштування інтерфейсу для блоків «FM»

«Nios2» - найважливіший компонент, ядро мікропроцесорної системи, рисунок 3.22. Основними налаштуванням являється: мало-продуктивна версія ядра, для збільшення кількості вільних логічних елементів на FPGA, рисунок 3.23; відключення компонентів для налагодження, рисунок 3.24. Усі інші налаштування генеруються автоматично.

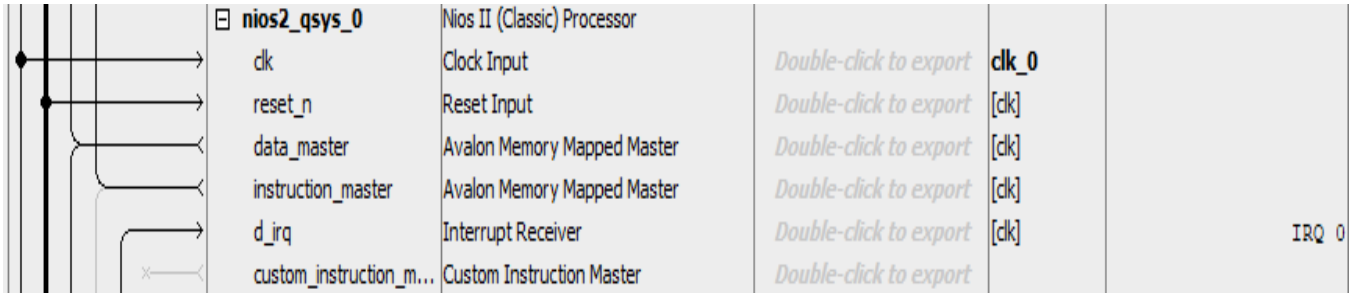


Рисунок 3.22– Графічне відображення інтерфейсу ядра «Nios2»

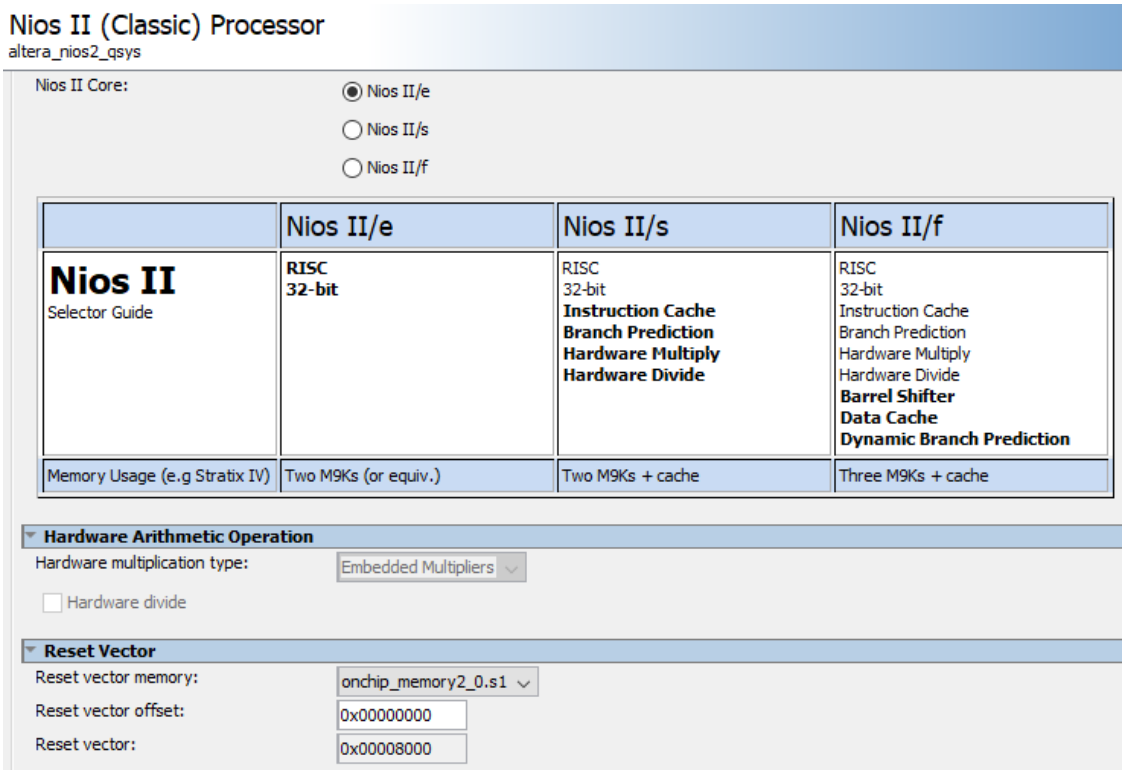


Рисунок 3.23– Вибір продуктивності ядра мікропроцесорної системи

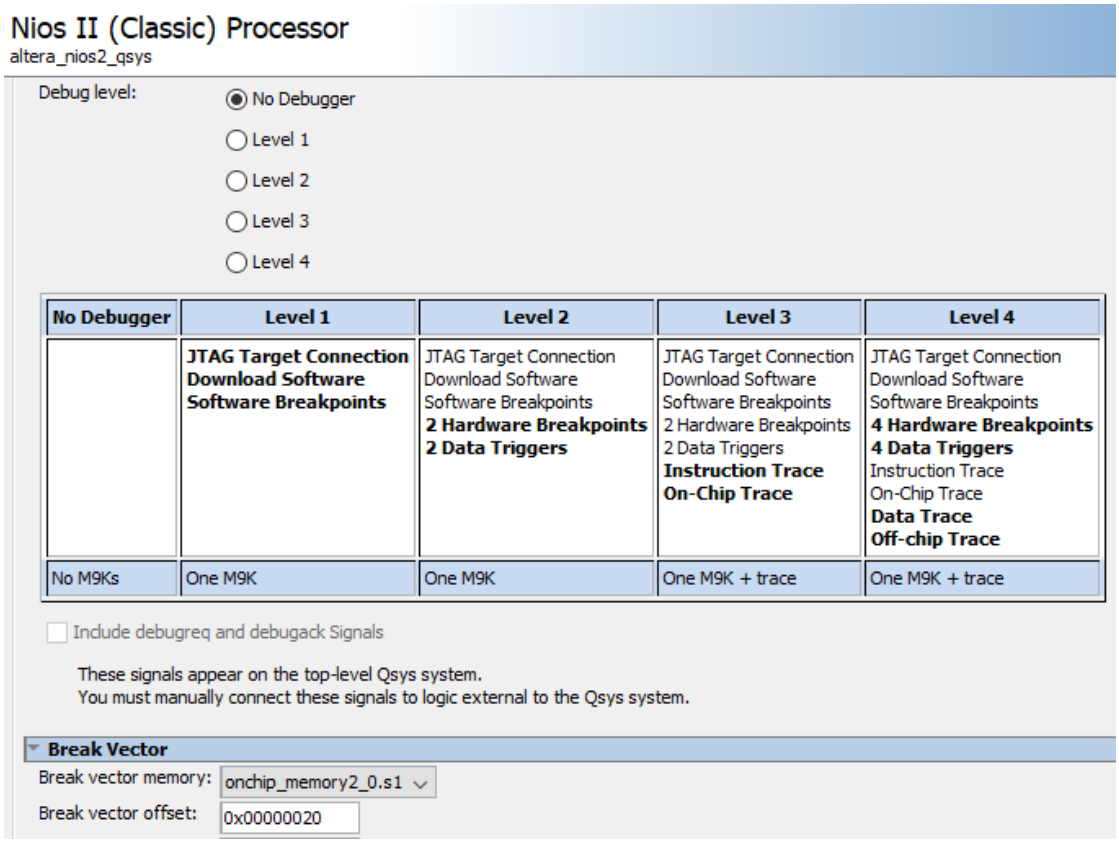


Рисунок 3.24– Налаштування налагоджувального компонента

Повна структурна схема з’єднань мікропроцесорної системи наведена у додатку О. На рисунку 3.25 зображений блок який містить всі компоненти згенерованої мікропроцесорної системи.

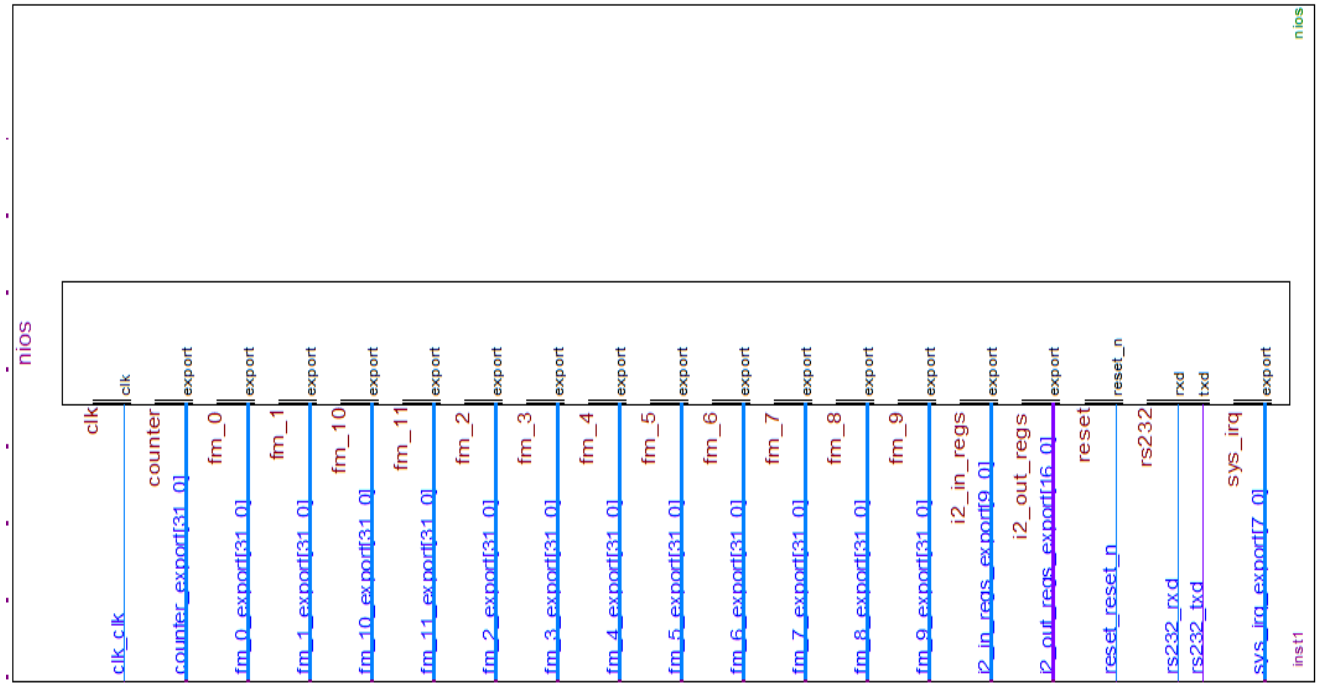


Рисунок 3.25– Блок згенерований утилітою Qsys

3.3.2 Підтримка I2C шини даних

I2C шина використовується для взаємодії із цифровими сенсорами. Реалізація даного протоколу відсутня у стандартній бібліотеці, тому, він був реалізований апаратно у вигляді окремого системного блоку, рисунок 3.26.

Призначення входів/виходів:

1. clk – сигнал від зовнішнього тактового генератора;
2. reset_n – очищення внутрішнього стану до початкових значень;
3. ena – керує дозволом на виконання транзакцій;
4. addr – 7-ми розрядна шина значення якої відповідають адресу веденого пристрою;
5. rw – вказує напрямок передачі даних;
6. data_wr – 8-ми розрядна шина значення якої відповідає байту даних призначеного для відправки веденому;
7. busy – вказує на процес виконання транзакції;

8. data_rd – 8-ми розрядна шина,яка зберігає зчитаний байт з веденого;
9. ack_error – вказує на виникнення помилки під час транзакції;
10. sda – послідовна лінія даних;
11. scl – послідовна лінія тактування.

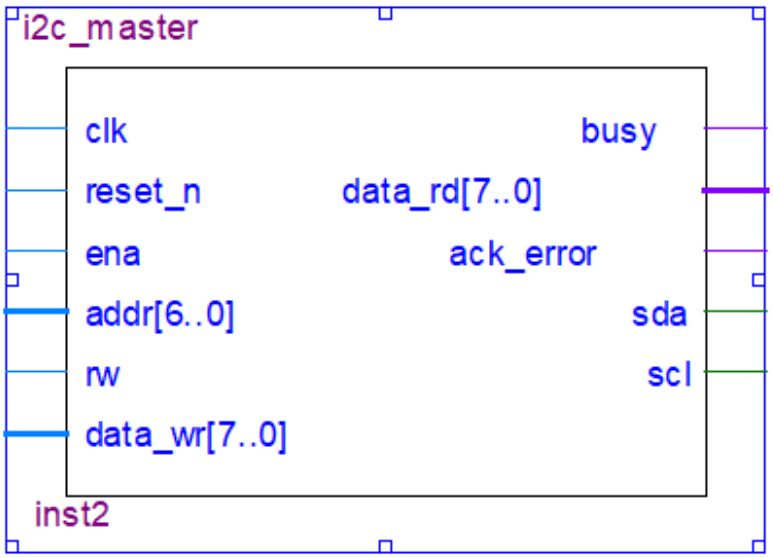


Рисунок 3.26– Блок реалізації I2C приймача/передавач

Код реалізації I2C приймача/передавача на мові Verilog наведений у додатку П. На рисунку 3.27 зображено процес передачі даних I2C компонентом.

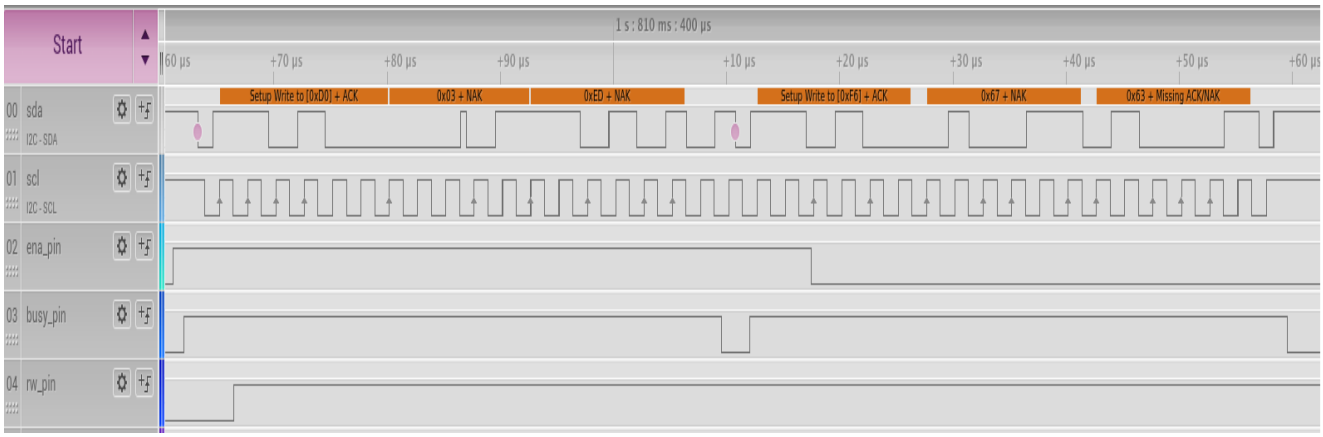


Рисунок 3.27– Відображення сигналів на усіх лініях інтерфейсу I2C компонента при спробі передачі даних

3.3.3 Модифікація системи з вимірювання частоти

При внесенні у схему частотоміра, мікропроцесорної системи, частина задач з існуючих блоків переноситься на більш гнучкий компонент – мікропроцесор. Такі блоки як: обробка даних з лічильників, UART передавач, повністю втрачають свою актуальність, вони будуть реалізовані засобами процесора. Блок лічильника імпульсів буде частково перероблений, для зручності використання з мікропроцесорною системою.

Оновлений лічильник імпульсів представляє з себе простий 32-х розрядний регістр який збільшує своє значення на одиницю при кожному вхідному імпульсі, у якості вихідного інтерфейсу використовується 32-х розрядна шина даних, рисунок 3.28. У додатку Р знаходиться оновлений код на мові Verilog.

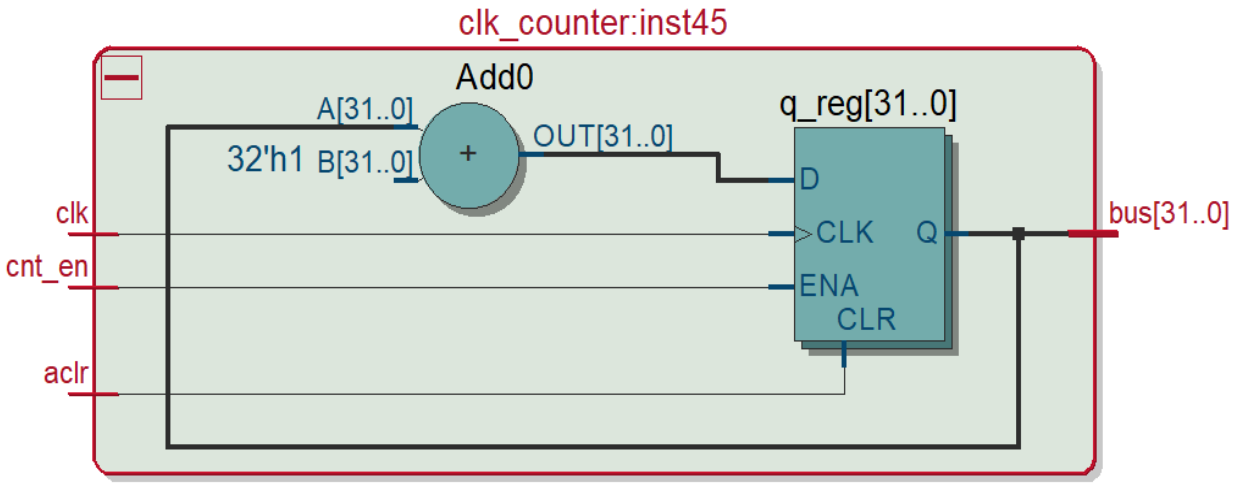


Рисунок 3.28– Лічильник імпульсів

Блок керування і 12 лічильників були поміщені у окремий компонент для зручності використання, рисунок 3.29. Інтерфейс отриманого компонента містить: вхід для зовнішнього тактового генератора, 12 вимірювальних входів, 12-ть вихідних шин для передачі результату роботи, інформаційні сигнали для визначення внутрішнього стану компонента, рисунок 3.30. На рисунку 3.31 зображена спрощена схема розробленої мікропроцесорної системи разом з блоком частотоміра і I2Cпередавача/приймача.

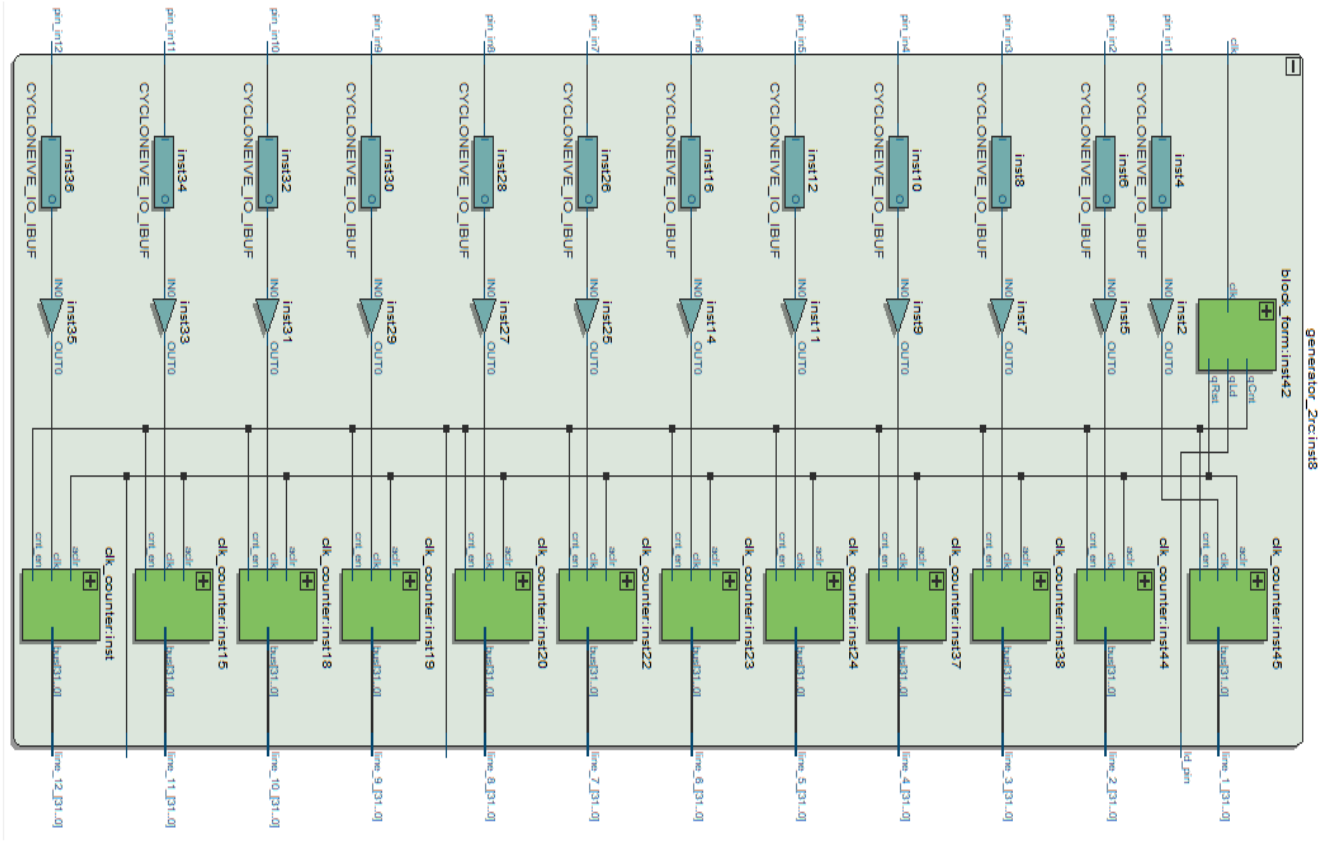


Рисунок 3.29– Внутрішня структура компонента із частотомірами

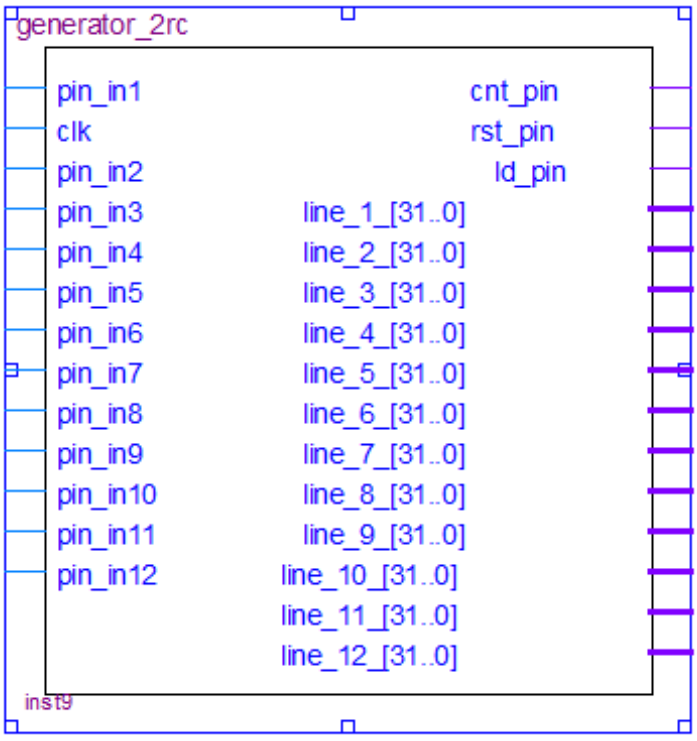


Рисунок 3.30– Інтерфейс компонента із частотомірами

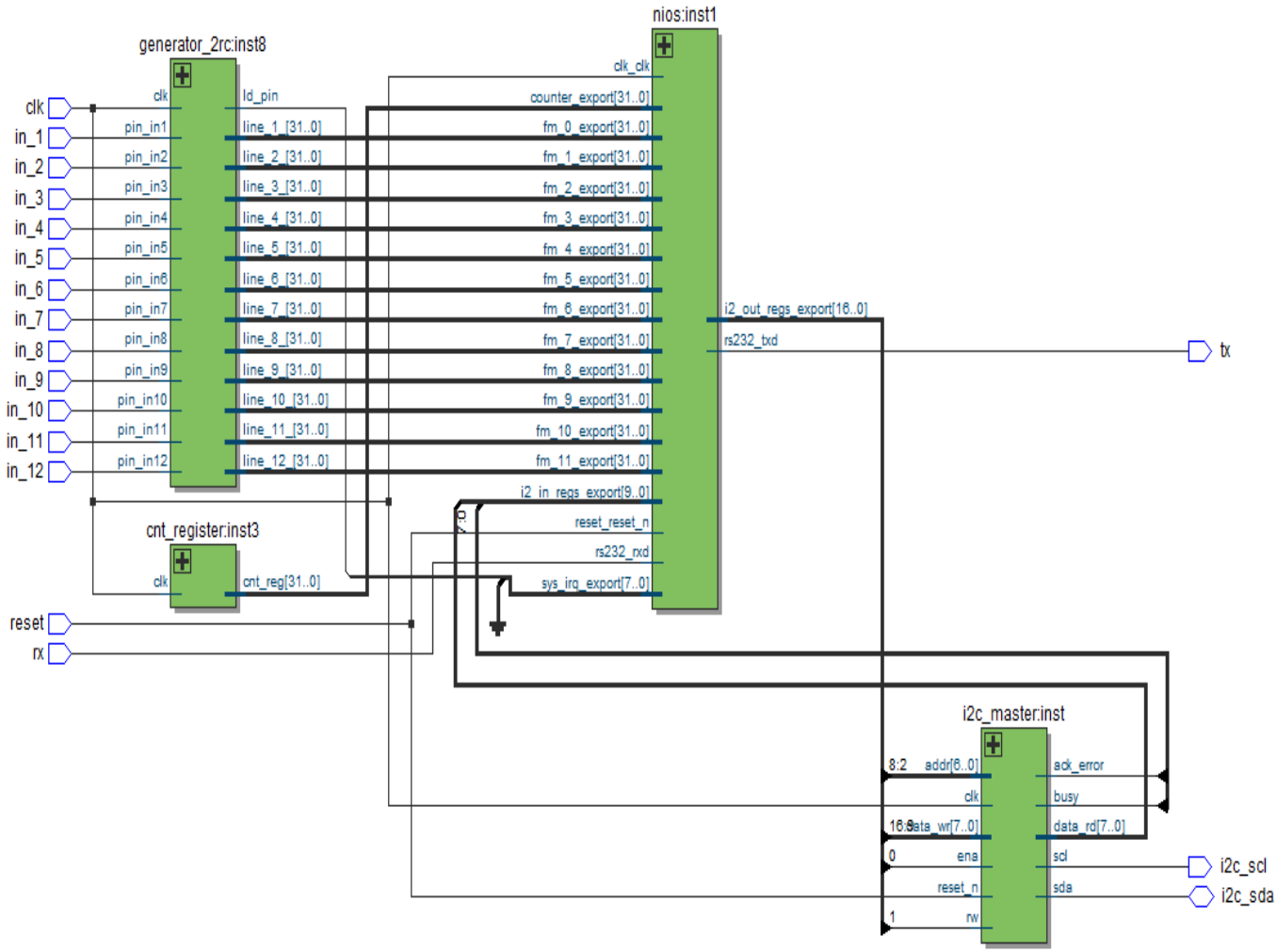


Рисунок 3.31– Спрощена схема розробленої системи

Повна структурна схема мікропроцесорної системи разом з частотомірами, I2C приймачем/передавачем і таймером знаходиться у додатку С.

3.4 Програмне забезпечення для мікропроцесорної системи

3.3.4 I2C драйвер

Для взаємодії з I2C компонентом було написано драйвер який розділений на дві частини:

1) Нижній рівень – наближений до апаратного рівня, інкапсулює роботу з апаратними інтерфейсами;

2) Верхній рівень - наближений до програмного забезпечення користувача, використовує незмінний інтерфейс який реалізується апаратним рівнем, для побудови високорівневих політик протоколу I2C.

Основними програмними компонентами нижнього рівня являються дві функції які використовують регістри процесорної системи створені у розділі 3.3.2. Використовуючи їх відбувається переключення між різними станами I2C компонента, запис/зчитування даних, отримання інформації про стан I2C системи, рисунки 3.32 - 3.33.

```
alt_8 i2c_hal_reg_write(alt_u8 reg_mask, alt_u8 reg_offset, alt_u8 value) {
    // Read last reg value and clear bits for set new value
    alt_u32 reg_value = IORD_ALTERA_AVALON_PIO_DATA(I2C_OUT_BASE) & (~((alt_u32)reg_mask << reg_offset));
    // Update value for reg
    reg_value |= ((alt_u32)value & reg_mask) << reg_offset;
    // Write new value into register
    IOWR_ALTERA_AVALON_PIO_DATA(I2C_OUT_BASE, reg_value);
    return RES_OK;
}
```

Рисунок 3.32– Реалізація функції запису у регістри I2C компонента

```
alt_8 i2c_hal_reg_read(alt_u8 reg_mask, alt_u8 reg_offset, alt_u8 *value) {
    // check if pointer is valid
    if (!value)
        return RES_ERR;
    // cut part of common register value
    *value = (IORD_ALTERA_AVALON_PIO_DATA(I2C_IN_BASE) & ((alt_u32)reg_mask << reg_offset)) >> reg_offset;

    return RES_OK;
}
```

Рисунок 3.33– Реалізація функції зчитування з регістрів I2C компонента

У верхньому рівні знаходяться реалізації функцій які використовуючи реалізації нижнього рівня, описують алгоритми запису і зчитування байтів даних з ведених пристроїв, рисунки 3.34 – 3.35.

```
alt_8 i2c_write_bytes(alt_u8 dev_addr, alt_u8 reg_addr, alt_u8 *w_data_arr, alt_u8 len) {
    alt_u8 i = 0x00;
    alt_u32 irq_state = 0x00;

    if (i2c_is_busy() || !w_data_arr || !len)
        return RES_ERR;

    // Disable all interrupts
    irq_state = sys_irq_critical_section_begin();

    i2c_hal_set_slave_addr(dev_addr);
    i2c_hal_set_rw(I2C_WRITE);
    i2c_hal_set_write_data(reg_addr);
    i2c_hal_set_ena(I2C_ENABLE_TRANSMISSION);

    // Wait stage of set register value
    _i2c_wait_for_busy(TRUE);

    for (i = 0x00; i < len; i++) {
        i2c_hal_set_write_data(w_data_arr[i]);
        if (i2c_is_err()) {
            i2c_hal_set_ena(I2C_DISABLE_TRANSMISSION);
            sys_irq_critical_section_end(irq_state);
            sys_printf("\r\n%s() -> Err at set register value stage, dev: %d, reg: %d\r\n",
                       __func__, dev_addr, reg_addr + i);
            return RES_ERR;
        }
        // Wait for write register value stage
        _i2c_wait_for_busy(FALSE);
        _i2c_wait_for_busy(TRUE);
    }

    i2c_hal_set_ena(I2C_DISABLE_TRANSMISSION);
    // End of transmission
    _i2c_wait_for_busy(FALSE);
    if (i2c_is_err()) {
        i2c_hal_set_ena(I2C_DISABLE_TRANSMISSION);
        sys_irq_critical_section_end(irq_state);
        sys_printf("\r\n%s() -> Err at end of read process, dev: %d, reg: %d\r\n",
                   __func__, dev_addr, reg_addr + i);
        return RES_ERR;
    }

    // Enable interrupts
    sys_irq_critical_section_end(irq_state);

    return RES_OK;
```

Рисунок 3.34– Реалізація алгоритму запису байтів у ведений пристрій

```
alt_8 i2c_read_bytes(alt_u8 dev_addr, alt_u8 reg_addr, alt_u8 *r_data_arr, alt_u8 len) {
    alt_u8 i = 0x00;
    alt_u32 irq_state = 0x00;

    if (i2c_is_busy() || !r_data_arr || !len)
        return RES_ERR;

    // Disable all interrupts
    irq_state = sys_irq_critical_section_begin();

    // Set up I2C registers for read
    i2c_hal_set_slave_addr(dev_addr);
    i2c_hal_set_rw(I2C_WRITE);
    i2c_hal_set_write_data(reg_addr);
    i2c_hal_set_ena(I2C_ENABLE_TRANSMISSION);

    // Wait for start transmission
    _i2c_wait_for_busy(TRUE);
    i2c_hal_set_rw(I2C_READ);
    if (i2c_is_err()) {
        i2c_hal_set_ena(I2C_DISABLE_TRANSMISSION);
        sys_irq_critical_section_end(irq_state);
        sys_printf("\r\n%s() -> Err at start transmission, dev: %d, reg: %d\r\n",
                __func__, dev_addr, reg_addr);
        return RES_ERR;
    }

    // Wait read stage
    _i2c_wait_for_busy(FALSE);
    for (i = 0x00; i < len; i++) {
        _i2c_wait_for_busy(TRUE);
        // Disable transmission before end of current
        // transaction for avoid starting new transaction
        if (i == len - 1)
            i2c_hal_set_ena(I2C_DISABLE_TRANSMISSION);

        _i2c_wait_for_busy(FALSE);
        i2c_hal_get_read_data(r_data_arr + i);
        if (i2c_is_err()) {
            i2c_hal_set_ena(I2C_DISABLE_TRANSMISSION);
            sys_irq_critical_section_end(irq_state);
            sys_printf("\r\n%s() -> Err during read register, dev: %d, reg: %d\r\n",
                    __func__, dev_addr, reg_addr + i);
            return RES_ERR;
        }
    }

    // Enable interrupts
    sys_irq_critical_section_end(irq_state);

    return RES_OK;
}
```

Рисунок 3.35– Реалізація алгоритму зчитування байтів з веденого пристрою

Загальна реалізація нижнього рівня знаходиться у додатку Т, а верхнього у додатку У.

3.4.2 Драйвер частотоміра

Для отримання значень з частотомірів використовується інтерфейс який був розроблений у розділі 3.3.3. Коли частотоміри закінчують підрахунок, генерується переривання яке викликає у якості обробника функцію «fm_irq_handler», рисунок 3.36.

```
void fm_irq_handler(void* context) {
    sys_irq_clear_interrupt_register(IRQ_MASK);
    alt_u8 i = 0x00;
    for (i = 0x00; i < CHANNEL_MAX; i++) {
        channels_data[i] = IORD_ALTERA_AVALON_PIO_DATA(channel_id_to_addr(i));
    }
}
```

Рисунок 3.36– Обробник переривань для частотомірів

Функція «fm_irq_handler» зчитує значення частотомірів, які зберігаються в регістрах, у масив на 12 елементів, звідки можуть бути доступні для користувача у будь-який час, рисунок 3.37. Кожний елемент масива представляє значення з окремого частотоміра.

```
alt_32 fm_get_freq(alt_u8 channel_id) {
    if (channel_id >= CHANNEL_MAX)
        return RES_ERR;

    return channels_data[channel_id];
}
```

Рисунок 3.37– Функція отримання значення з певного частотоміра

Загальна реалізація драйвера частотомірів знаходиться у додатку Ф.

3.4.3 Перевірка працездатності програмного забезпечення

Для перевірки I2C драйвера і драйвера частотомірів до системи було під’єднано генератор частоти на 85 кГц до першого каналу і модуль MPU6050 який містить три цифрових датчика (гіроскоп, акселерометр, датчик температури), взаємодія з ними відбувається за допомогою I2C-шини.

Для отримання даних з трьох датчиків було реалізовано драйвер MPU6050, він використовує I2C драйвер і містить три основних функції які використовують реалізацію I2C протоколу для зв'язку з датчиками, рисунки 3.38 – 3.40.

```
alt_8 mpu6050_accel(mpu6050_t *mpu_dev, alt_16 *x, alt_16 *y, alt_16 *z) {
    alt_u8 accel_data[MPU6050_ACCEL_REG_NUM] = {0x00};

    if (!mpu_dev || !_mpu6050_get_dev_by(mpu_dev->addr) || !x || !y || !z) {
        sys_printf("\r\n%s() -> Error incorrect parameters\r\n", __func__);
        return RES_ERR;
    }

    if (i2c_read_bytes(mpu_dev->addr, MPU6050_RA_ACCEL_XOUT_H, accel_data, MPU6050_ACCEL_REG_NUM)) {
        sys_printf("\r\n%s() -> Error read from dev: %d, reg: %d\r\n", __func__, mpu_dev->addr, MPU6050_RA_ACCEL_XOUT_H);
        return RES_ERR;
    }

    // Converting received data into accel x, y, z
    *x = (((alt_16)accel_data[0]) << 8) | accel_data[1];
    *y = (((alt_16)accel_data[2]) << 8) | accel_data[3];
    *z = (((alt_16)accel_data[4]) << 8) | accel_data[5];

    *x = *x < 0? (alt_16)(mpu_dev->acc_coeff * (*x * -1)) * -1 : (alt_16)(mpu_dev->acc_coeff * *x);
    *y = *y < 0? (alt_16)(mpu_dev->acc_coeff * (*y * -1)) * -1 : (alt_16)(mpu_dev->acc_coeff * *y);
    *z = *z < 0? (alt_16)(mpu_dev->acc_coeff * (*z * -1)) * -1 : (alt_16)(mpu_dev->acc_coeff * *z);

    return RES_OK;
}
```

Рисунок 3.38– Зчитування значень з акселерометра

```
alt_8 mpu6050_gyro(mpu6050_t *mpu_dev, alt_16 *x, alt_16 *y, alt_16 *z) {
    alt_u8 gyro_data[MPU6050_GYRO_REG_NUM] = {0x00};

    if (!mpu_dev || !_mpu6050_get_dev_by(mpu_dev->addr) || !x || !y || !z) {
        sys_printf("\r\n%s() -> Error incorrect parameters\r\n", __func__);
        return RES_ERR;
    }

    if (i2c_read_bytes(mpu_dev->addr, MPU6050_RA_GYRO_XOUT_H, gyro_data, MPU6050_GYRO_REG_NUM)) {
        sys_printf("\r\n%s() -> Error read from dev: %d, reg: %d\r\n", __func__, mpu_dev->addr, MPU6050_RA_GYRO_XOUT_H);
        return RES_ERR;
    }

    // Converting received data into gyro x, y, z
    *x = (((alt_16)gyro_data[0]) << 8) | gyro_data[1];
    *y = (((alt_16)gyro_data[2]) << 8) | gyro_data[3];
    *z = (((alt_16)gyro_data[4]) << 8) | gyro_data[5];

    *x = *x < 0? (alt_16)(mpu_dev->gyro_coeff * (*x * -1)) * -1 : (alt_16)(mpu_dev->gyro_coeff * *x);
    *y = *y < 0? (alt_16)(mpu_dev->gyro_coeff * (*y * -1)) * -1 : (alt_16)(mpu_dev->gyro_coeff * *y);
    *z = *z < 0? (alt_16)(mpu_dev->gyro_coeff * (*z * -1)) * -1 : (alt_16)(mpu_dev->gyro_coeff * *z);

    return RES_OK;
}
```

Рисунок 3.39– Зчитування значень з гіроскопа

```
alt_8 mpu6050_temp(mpu6050_t *mpu_dev ,alt_ul6 *temp) {
    alt_u8 temp_data[MPU6050_TEMP_REG_NUM] = {0x00}, guard_cnt = MPU6050_TRYS_TO_READ;

    if (!mpu_dev || !_mpu6050_get_dev_by(mpu_dev->addr) || !temp) {
        sys_printf("\r\n%s() -> Error incorrect parameters\r\n", __func__);
        return RES_ERR;
    }

    read_temp:
    if (i2c_read_bytes(mpu_dev->addr, MPU6050_RA_TEMP_OUT_H, temp_data, MPU6050_TEMP_REG_NUM)) {
        sys_printf("\r\n%s() -> Error read from dev: %d, reg: %d\r\n", __func__, mpu_dev->addr, MPU6050_RA_TEMP_OUT_H);
        return RES_ERR;
    }

    // Check if data reads correctly
    if (temp_data[0] == 0 && temp_data[1] == 0) {
        // Read allowed only guard_cnt numbers
        if (guard_cnt--) {
            goto read_temp;
        } else {
            //sys_printf("\r\n%s() -> Error max count of read temperature\r\n", __func__);
            return RES_ERR;
        }
    }

    // Converting received data into correct temperature value
    *temp = (((alt_ul6)temp_data[0]) << 8) | temp_data[1];
    *temp = ((float)*temp / 340) + 36.53;

    return RES_OK;
}
```

Рисунок 3.40– Зчитування значень з температурного датчика

Основна функція яка використовує усі реалізовані драйвера спершу ініціалізує кожний з них, після чого послідовно викликає зчитування значень з усіх каналів частотомірів і зчитування даних з акселерометра, гіроскопа і температурного датчика, реалізація функції зображена на рисунку 3.41. Після зчитування отриманих значень відбувається формування повідомлення з ними, у послідовності їх зчитування, кожне значення розділене символом «\t». Сформоване повідомлення відправляється для передачі на UART, до якого під’єднаний конвертор PL2303 який у свою чергу під’єднується до USB порту ПК. Перші 12 значень відповідають значенням частотомірів, наступне– значення температури, наступні три – значення акселерометра і останні три – значення гіроскопа, рисунок 3.42.

```c
int main() {
    sys_printf("Frequency meter initializing\r\n");
    if (fm_init())
        sys_printf("Error during frequency meter initializing\r\n");

    sys_printf("I2C initializing\r\n");
    if (i2c_init())
        sys_printf("Error during I2C initializing\r\n");

    sys_printf("MPU6050 initializing\r\n");
    if (mpu6050_init(&mpu_dev))
        sys_printf("Error during MPU6050 initializing\r\n");

    /* Event loop never exits. */
    while (1) {
        alt_16 x = 0x00, y = 0x00, z = 0x00;
        alt_u16 temp = 0x00;
        alt_u8 i = 0x00;

        for (; i < CHANNEL_MAX; i++)
            data_printf(i == 0? "%d" : "\t%d", fm_get_freq(i));

        mpu6050_temp(&mpu_dev, &temp);
        data_printf("\t%d", temp);

        mpu6050_accel(&mpu_dev, &x, &y, &z);
        data_printf("\t%d\t%d\t%d", x, y, z);

        x = y = z = 0x00;
        mpu6050_gyro(&mpu_dev, &x, &y, &z);
        data_printf("\t%d\t%d\t%d\r\n", x, y, z);

        data_printf("\r\n");
        sys_delay_ms(100);
    }

    return 0;
}
```

Рисунок 3.41– Зчитування значень з усіх драйверів і вивід отриманих значень на UART.

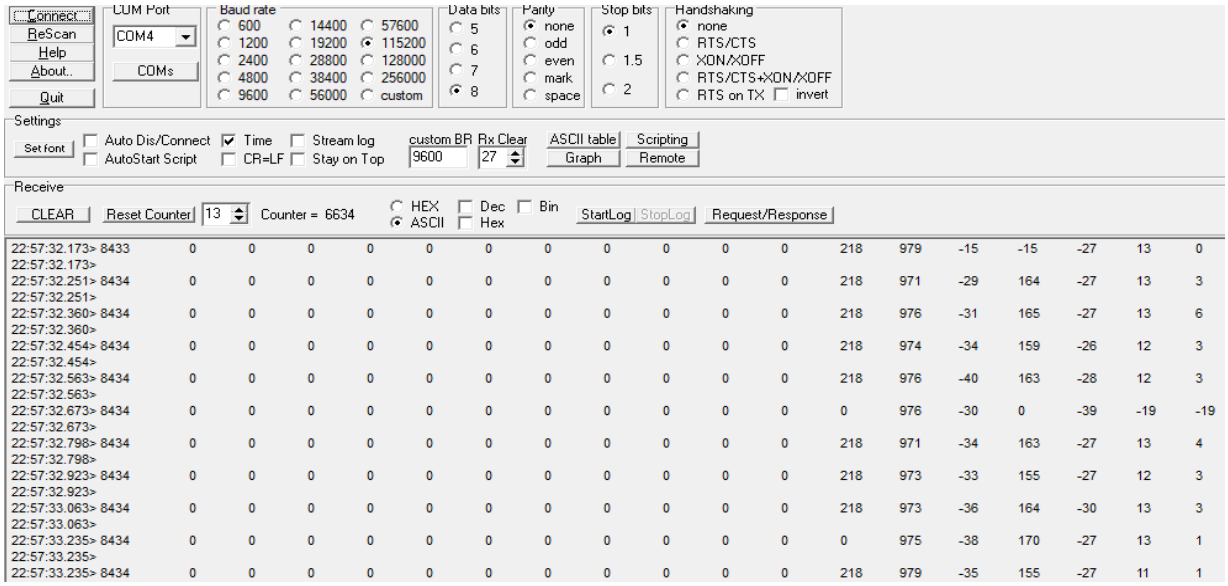


Рисунок 3.42– Результат роботи процесорної системи з під’єднаними до неї датчиками.

3.5 Висновки до третього розділу

У даному розділі було:

1. Розглянуто складові архітектури процесорних систем на основі ядра «Nios II» і спроектовано свою процесорну систему яка складається, частково, з блоків присутніх у стандартній бібліотеці, і частково, з особисто реалізованих блоків;
2. Розглянуто протокол передачі даних I2C і реалізовано його у вигляді апаратного блоку на FPGA.
3. Перероблено схему частотомірів для можливості інтеграції процесорної системи з додатковим набором інтерфейсів і підтримкою датчиків з цифровим входом.
4. Розроблено і протестовано програмне забезпечення під створену процесорну систему для підтримки усіх реалізованих інтерфейсів.

4 РОЗРОБКА ПРОГРАМНОГО КОМПЛЕКСУ ДЛЯ ВЗАЄМОДІЇ З ПРИЛАДОМ

Для отримання виміряних і зчитаних приладом значення, необхідно створити програмну систему яка зможе використовуючи USB-порт, налаштувати лінію зв'язку і необхідним чином інтерпретувати отримані дані. Розробка програмного забезпечення, для спрощення процесу, буде відбуватись на мові python [33] у середовищі розробки PyCharm [34] з використанням фреймворку PyQt5 [35].

Спершу створимо проект у середовищі розробки PyCharm і назвемо його «SerialMonitor», рисунок 4.1.

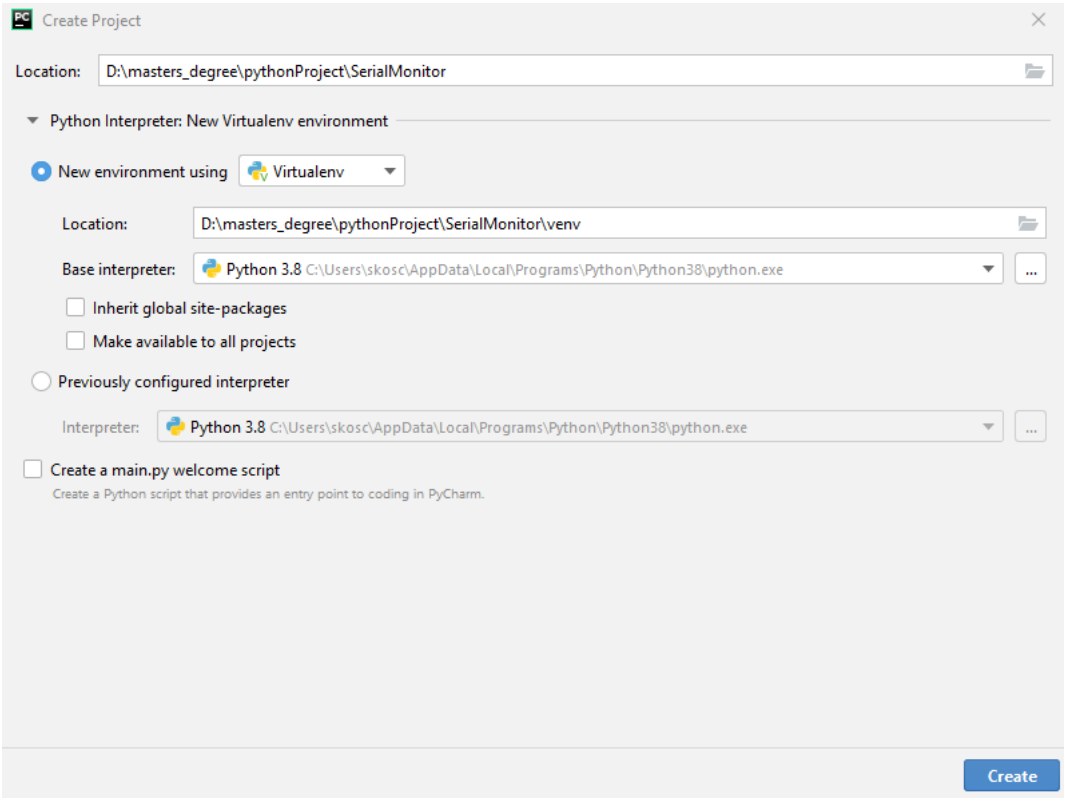


Рисунок 4.1 – Створення нового проекту у середовищі розробки PyCharm.

Після того як проект був успішно створений можна приступати до написання програми, першим етапом буде створення класу для налаштування USB порту і отримання даних з нього. На рисунку 4.2 зображено процес ініціалізації вибраного порту, для ініціалізації використовуються параметри які

були задані при створенні інтерфейсу UART на пристрої. На рисунку 4.3 зображено алгоритм зчитування і обробки отриманих даних. Після того як було отримано символ ‘\n’ з пристрою, який свідчить про завершення передачі одного пакету відбувається десеріалізація отриманих даних.

```
def __init__(self, context, port: str, read_callback: Callable[[List[int]], None]):
    self.__s_data: str = ""
    self.read_callback = read_callback
    self.__serial: Optional[QSerialPort] = QSerialPort(context)
    self.__serial.setPortName(port)
    self.__serial.setBaudRate(115200)
    self.__serial.readyRead.connect(self.__on_serial_read)
    if self.__serial.open(QIODevice.ReadOnly) is False:
        raise Exception(f"Incorrect serial port: {port}")
```

Рисунок 4.2 – Ініціалізація USB порту

```
def __on_serial_read(self):
    # Read character
    self.__s_data += str(self.__serial.readAll(), "utf-8")
    if '\n' not in self.__s_data:
        return

    # Parse read line
    try:
        last_index = self.__s_data.find("\n")
        data_list = self.__s_data[:last_index+1].split("\r\n")
        self.__s_data = self.__s_data[last_index+1:] if len(self.__s_data) > last_index+1 else ""
        for sub_data in data_list:
            if len(sub_data) == 0:
                continue
            data = sub_data.split('\t')
            data = list(map(int, data))
            # Send processed line to user
            self.read_callback(data)
    except Exception as ex:
        print(f"__on_serial_read() -> Unable to process data: {data_list}, ex: {ex}")
```

Рисунок 4.3 – Зчитування і обробка даних з пристрою

Наступним етапом являється розробка графічного інтерфейсу який дозволить користувачу вибрати порт з під'єднаним пристроєм, відобразити виміряні значення частоти і зчитані значення із сенсорів з цифровим виходом, рисунок 4.4. Розроблений інтерфейс буде складатись з трьох основних частин:

1) Вибір USB порту до якого під'єднано пристрій;
2) Відображення даних з 12 каналів частотоміра у вигляді графіку;
3) Відображення даних з сенсорів із цифровим виходом у вигляді графіку.

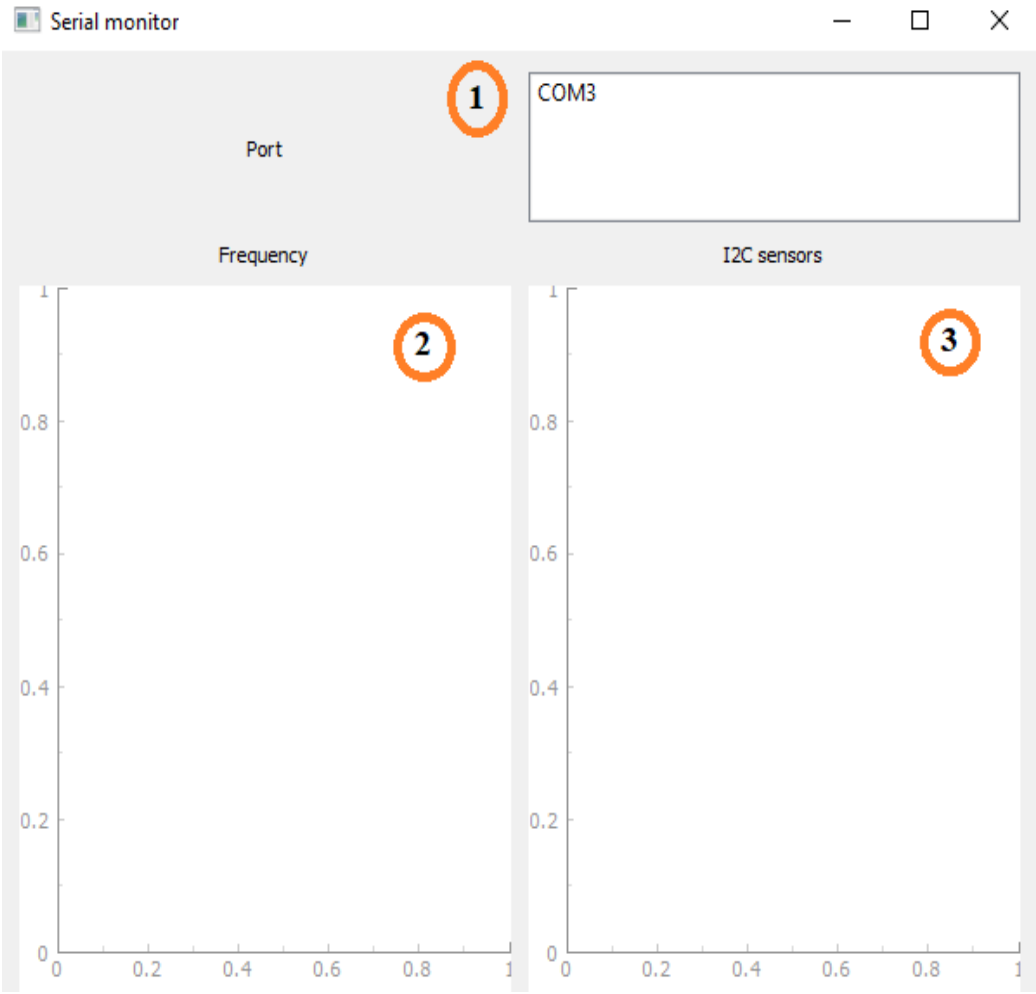


Рисунок 4.4 – Графічний інтерфейс з пронумерованими компонентами

На рисунку4.5 відображено роботу розробленого програмного забезпечення. До частотоміра під'єднано два канали від генератора з частотою 85 кГц і 79 кГц, до I2C шини приєднано акселерометр, гіроскоп і датчик температури.

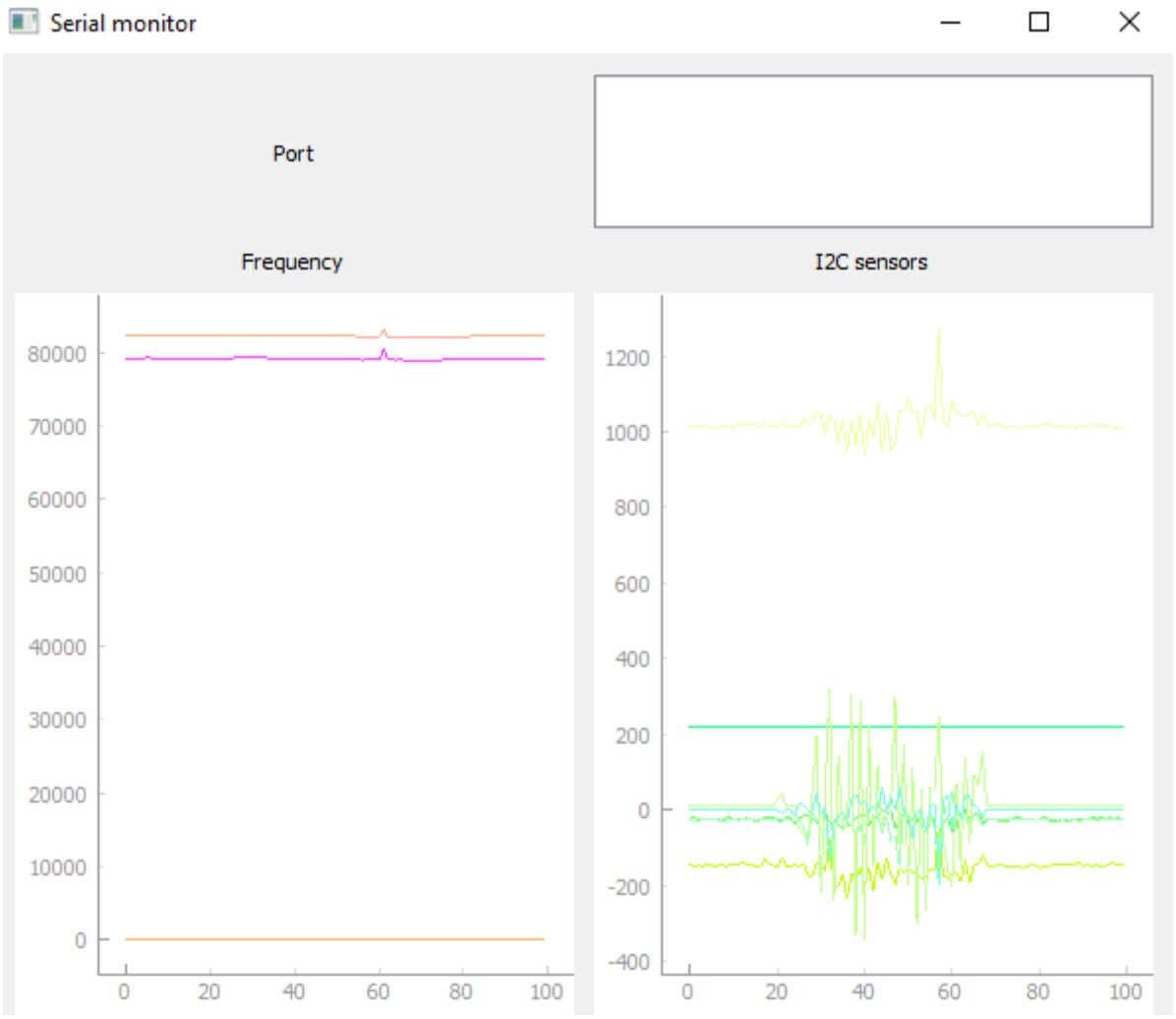


Рисунок 4.5 – Результат роботи розробленого програмного забезпечення

Загальний код реалізації «SerialMonitor» знаходиться у додатку X.

4.1 Висновки до четвертого розділу

У даному розділі був розроблений останній компонент системи вимірювання. Який дозволяє відображати отримані від приладу дані у вигляді графіків, для спрощення аналізу і оцінки. Для розробки програмного забезпечення використовувалась мова програмування python і середовище розробки PyCharm.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Однією із характерних особливостей сучасного розвитку суспільства є зростання сфер діяльності людини, в яких використовуються інформаційні технології. Широке розповсюдження отримали персональні комп'ютери. Однак їх використання загострило проблеми збереження власного та суспільного здоров'я, вимагає удосконалення існуючих та розробки нових підходів до організації робочих місць, проведення профілактичних заходів для запобігання розвитку негативних наслідків впливу ПК на здоров'я користувачів.

У магістерській кваліфікаційній роботі досліджується багатоканальна вимірювальна система на FPGA для радіовимірювальних частотних сенсорів. Дослідження і експерименти відбуваються з участю автоматизованого робочого місця. Основним завданням охорони праці є виявлення шкідливих факторів в процесі трудової діяльності і зменшення їх шкідливого впливу на працівника.

При розробленні заходів щодо підвищення інформаційної безпеки стільникового зв’язку, дослідження проводяться на ПК тому на працівника можуть впливати такі небезпечні та шкідливі фактори, у відповідності з прийнятою класифікацією за ГОСТ 12.0003.-74 [36].:

Фізичні:

- підвищена та понижена температура повітря робочої зони;
- підвищена та понижена рухливість повітря робочої зони;
- недостатня освітленість робочої зони;
- недостатність природного освітлення;
- небезпечний рівень напруги в електричному колі, замикання якого може відбутися через тіло людини;
- підвищена та понижена вологість повітря;
- підвищений рівень електромагнітного опромінення.

Психофізіологічні: нервово – психічні перевантаження (монотонність праці, емоційні перевантаження, перенапруга аналізаторів).

5.1 Технічні рішення з безпечного виконання робіт

5.1.1 Технічні рішення з організації робочого місця під час проектування

Для організації робочого місця керуємось основними вимогами до організації роботи з екранними пристроями.

Основні вимоги до організації роботи з екранними пристроями [39]:

– площа на одне робоче місце має становити не менше ніж 6,0 м,а об'єм не менше ніж 20,0м3;

– природне освітлювання має забезпечувати коефіцієнт природної освітленості не нижче 1,5%. Розраховується КПО за методикою, викладеною в ДБН В.2.5–28–2006;

– віконні прорізи приміщень для роботи з ВДТ мають бути обладнані регульованими пристроями (жалюзі, завіски, зовнішні козирки);

– покриття підлоги повинне бути матовим з коефіцієнтом відбиття 0,3–0,5;

– забороняється для оздоблення інтер'єру приміщень ВДТ застосовувати полімерні матеріали (деревинно –стружкові плити, шпалери, що миються, рулонні синтетичні матеріали, шаруватий паперовий пластик тощо), що виділяють у повітря шкідливі хімічні речовини .

– у приміщеннях з ВДТ слід щоденно робити вологе прибирання;

– приміщенням з ВДТ мають бути обладнані побутові приміщення для відпочинку під час роботи, кімната психологічного розвантаження. В кімнаті психологічного розвантаження слід передбачити встановлення пристроїв для приготування й роздачі тонізуючих напоїв, а також місця для занять фізичною культурою.

5.1.2. Електробезпека виробничого приміщення

Класифікація приміщень за категоріями електробезпеки залежно від мікроклімату виробничих будівель. Допустимими визнаються умови праці у

будівлях, де відносна вологість повітря не перевищує 60%, температура повітря не перевищує 35 °С, а пил та хімічно агресивне середовище – відсутні. За таких умов праці мікроклімат вважається сухим. Вологими називаються умови роботи, де відносна вологість повітря становить від 60% до 75%. Вологі – це такі умови, які характеризуються відносною вологістю повітря в будівлі більшою за 75%. Особливо сирі умови – це умови, із майже стовідсотковою відносною вологістю повітря. Гарячими умовами праці є становище, при якому температура повітря перевищує 35 °С. Запиленими визнаються умови, при яких виділяється велика кількість виробничого пилу, внаслідок чого він може залишатись на зовнішніх поверхнях або навіть проникати у середину обладнання чи апаратів. До умов праці з хімічно активним середовищем відносять умови, при яких у повітрі протягом тривалого часу залишаються гази або краплі рідин, які негативно впливають на ізолюючі властивості і струмопровідні елементи електричних інструментів.

Класифікація приміщень за рівнем електробезпеки Відповідно до ПУЕ, усі промислові приміщення (цехи, майстерні, склади) за ступенем ймовірності ураження електричним струмом можна розділити на три категорії: Будівлі з підвищеною небезпекою До цього типу будівель належать споруди, в яких наявний принаймні один із таких факторів: сирі приміщення, відносна вологість в яких протягом тривалого часу перевищує 75%; приміщення, в яких пил покриває провідники, забивається всередину машин та обладнання; приміщення зі струмопровідними підлогами (металевими, земляними, цегляними, залізобетонними); приміщення, в яких середня температура повітря зазвичай перевищує +30 °С; приміщення, в яких існує ймовірність одночасного торкання співробітника до зовнішніх елементів електричного устаткування і заземлених металевих конструкцій будівель, технологічного обладнання тощо.

Будівлі з особливо небезпечними умовами - ці будівлі характеризуються як дуже сирі приміщення зі стовідсотковою відносною вологістю. Стеля, стіни, підлога, устаткування у таких приміщеннях постійно вкриті тонким шаром крапель чи пліснявою. Слід бути вкрай обережними, оскільки при виконанні робіт

з використанням електричної напруги на відкритому повітрі, всередині посудин, всередині непросохлих приміщень ймовірність ураження співробітників чи сторонніх осіб електричним струмом дуже висока. До будівель з особливо небезпечними умовами належать також споруди з хімічно активним середовищем, яке завдяки своїм властивостям завдає шкоду ізоляції та електричним матеріалам. Окрім того, будівлі, які мають одночасно дві або більше ознаки приміщень з підвищеною небезпекою так само належать до будівель із особливо небезпечними умовами. Будівлі без факторів збільшеної небезпеки До цієї категорії належать будівлі, в яких немає жодної з ознак, властивих приміщенням з підвищеною або особливою небезпекою[37].

Для створення оптимальних умов електробезпеки в виробничому приміщенні потрібно підтримувати режим вологості повітря на рівні 60%, для того щоб приміщення було сухим, та температуру повітря не вище 35ºС. Постійний моніторинг середовища, на предмет пилу та хімічно агресивних речовин у складі повітря, для забезпечення електробезпеки приміщення. Крім цього забезпечити ізоляцію всіх електрод ротів та заземлення всіх елекетропристроїв.

5.2. Технічні рішення з гігієни праці та виробничої санітарії

5.2.1 Мікроклімат

Метеорологічні умови виробничих приміщень (санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042−99) можна оцінювати за сукупністю таких факторів, як температура (t, °С), відносна вологість (φ, %), швидкість руху повітря (V, м/с) та величина інтенсивності теплового опромінення (Е, Вт/м2). За ступенем впливу на тепловий стан людини мікрокліматичної умови поділяють на оптимальні та допустимі. Оптимальні мікрокліматичні умови − поєднання параметрів мікроклімату, які при тривалому та систематичному впливі на людину забезпечують зберігання нормального теплового стану організму без активізації механізмів терморегуляції. Вони забезпечують відчуття теплового

комфорту та створюють передумови для високого рівня працездатності [38]. Допустимі мікрокліматичні умови − поєднання параметрів мікроклімату,які при тривалому та систематичному впливі на людину можуть викликати зміни теплового стану організму, що швидко минають і нормалізуються та супроводжуються напруженням механізмів терморегуляції в межах фізіологічної адаптації. При цьому не виникає ушкоджень або порушень стану здоров'я, але можуть спостерігатися дискомфортні тепловідчуття, погіршення самопочуття та зниження працездатності[38].

Категорія робіт − розмежування робіт за важкістю на основі загальних енерговитрат організму[38].Легкі фізичні роботи (категорія I) охоплюють види діяльності, при яких витрата енергії дорівнює 105−140 Вт (90−120 ккал/год.) − категорія Іа та 141−175 Вт (121−150 ккал/год.) − категорія Іб. До категорії Іа належать роботи, що виконуються сидячи і не потребують фізичного напруження. До категорії Іб належать роботи, що виконуються сидячи, стоячи або пов'язані з ходінням та супроводжуються деяким фізичним напруженням. Визначаємо наявну категорію робіт, як Іа. В кабінах, на пультах та місцях керування технологічними процесами, в залах ЕОМ при виконанні робіт операторського типу повинні забезпечуватися такі оптимальні величини температури, відносної вологості та швидкості руху повітря, що зазначені в нормативному акті НПАОП 0.00-7.15-18 Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями[39]. Оптимальні параметри мікроклімату наведені в таблиці 1.

Таблиця 5.1 – Оптимальні параметри мікроклімату при виконанні робіт операторського типу на ЕОМ для робіт категорії Іа

Період року	Температура повітря, ºС	Відносна вологість, %	Швидкість руху, м\с
Холодний	22-24	60-40	0,1
Теплий	23-25	60-40	0,1

При плануванні умов робочого місця потрібно дотримуватись саме цих параметрів мікроклімату для досягнення оптимальних умов праці, що виконує важливу роль в якості виконаної роботи працівником та в безпеці його трудової діяльності. Приміщення повинно бути забезпечене системою опалення та вентиляції.

5.2.2. Склад повітря робочої зони

В приміщенні, де здійснюється розробка багатоканальної вимірювальної система на FPGA для радіовимірювальних частотних сенсорів, можливими шкідливими речовинами у повітрі є фенол, пил, озон та вуглекислий газ. Джерелами цих речовин є офісна техніка. Пил потрапляє у приміщення ззовні. ГДК шкідливих речовин, згідно ДСН 3.3.6.042-99 [43] які знаходяться в досліджуваному приміщені, наведені нижче:

Назва речовини	Клас небезпечності	Максимально разова ГДК, мг/м 3	Середньо добова ГДК, мг/м 3
Фенол	3	0,01	0,01
Пил нетоксичний	4	0,5	0,15
Озон	4	0,16	0,03
Вуглекислий газ	4	3	1

Потрібен постійний контроль за показниками рівнів концентрації наведених вище речовин, дотримання гігієни приміщення (режиму прибирання, провітрювання). Постійний контроль роботи систем вентиляції приміщення, якщо є кондиціонер, регулярне очищення його фільтрів. Дотримання режиму праці відпочинку також відіграє значну роль як в перевантаженні техніки, так і в кількості шкідливих речовин, що виділяються і які діють на організм на протязі робочого дня .

5.2.3 Виробниче освітлення

Відповідно до ДБН В.2.5-28:2018 [41] Система природного освітлення відноситься до бокової. Характеристика зорових робіт – середньої точності. Норми освітленості при штучному освітленні та КПО (для III пояса світлового клімату) при природному та сумісному освітленні зазначені у таблиці 2:

Таблиця 5.2 - Норми освітленості в приміщенні

Характеристика зорової роботи	Найменший розмір об'єкта розрізнювання	Розряд зорової роботи	Підрозряд зорової роботи	Контраст об'єкта розрізнення з фоном	Характеристика фона	Освітленість, лк		КПО,%			
						Штучне освітлення		Природнє освітлення		Сумісне освітлення	
						Комбіноване	Загальне	Верхнє або верхнє і бокове	Бокове	Верхнє або верхнє і бокове	Бокове
Середньої точності	Від 0,5 до 1,0	IV	б	середній	середній	200	500	4	1,5	2,4	0,9

При експлуатації штучного освітлення здійснюється контроль за рівнем напруги освітлювальної мережі, своєчасна заміна перегорілих ламп, забезпечується чистота повітря у приміщенні.

Для забезпечення достатнього освітлення слід максимально використовувати бічне природного освітлення, систематично очищувати скло від бруду та систематично замінювати перегорілі лампи (оптимальним варіантом

буде встановлення світодіодних ламп відповідної потужності для забезпечення економічності).

5.2.4 Виробничий шум

Джерелами шуму під час виконання робіт є обладнання, машини, механізми – механічний шум. Шум – це хаотична сукупність різних за силою і частотою звуків, що заважають сприйняттю корисних сигналів і негативно впливають на людину. Постійна дія сильного шуму може не лише негативно вплинути на слух, але й викликати інші шкідливі наслідки - дзвін у вухах, запаморочення, головний біль, підвищення втоми, зниження працездатності.

Нормативним документом, який регламентує рівні шуму для різних категорій робочих місць службових приміщень, є ДСН 3.3.6.037-99 [42].

Таблиця 5.3 – Допустимі рівні звукового тиску в октанових смугах частот, рівні шуму та еквівалентні рівні шуму

Робоче місце	Рівні звукового тиску (дБ) в октавних смугах з середньо герметичними частотами, Гц									Рівні шуму та еквівалентні рівні шуму, (дБ)
	1,5	3	25	50	00	000	000	000	000	
Для програмістів операторів ЕОМ (ПЕОМ)	6	1	1	4	9	5	2	0	8	50

Для зменшення рівня шуму до допустимого в цеху двигуни виконуються в металевому кожусі, а також виконують змащення, застосовують пластмасові деталі, використовують протишумні навушники, які закривають вушну раковину.

5.2.5. Електромагнітні випромінювання

У теперішній час рядом країн розроблено документи, які регламентують правила користування дисплеями. Найбільш відомі шведські документи MPR II 1990:8 (Шведського національного комітету з захисту від випромінювань) та більш жорсткий стандарт TCO 95 (Шведської конференції професійних союзів). Ці норми застосовуються у всіх країнах Скандинавії і рекомендовані до розповсюдження в країнах ЕС. Вимоги норм MPR до рівня електромагнітних випромінювань у 20 разів жорсткіші, ніж вимоги ГОСТ, що обмежують рівень випромінювання радіочастот, вимоги TCO 95 жорсткіші у 50 разів. Нижче приводяться для порівняння з ГОСТ 12.1.006-84 «Електромагнітні поля радіочастот» дані шведського стандарту MPR II1990:8. В діапазоні частот 5 Гц−2 кГц напруженість електричного поля Е не повинна перевищувати 25 В/м, а магнітна індукція − 250 нТл. Це рівнозначно напруженості магнітного поля Н = 0,2 А/м. В діапазоні частот 2−400 кГц − Е ≤ 2,2 В/м, а Н ≤ 0,02 А/м. Цими нормами рекомендується користуватися і в Україні.

У всіх випадках для захисту від випромінювань очі повинні бути розташовані на відстані витягнутої руки до монітора (не ближче 70 см). Більш пізні монітори з маркуванням Low Radiation практично задовольняють вимоги шведських стандартів. Комп’ютери з рідкокристалічним екраном не наводять статичної електрики і не мають джерел відносно потужного електромагнітного випромінювання. При використанні блока живлення виникає деяке перевищення рівня на промисловій частоті, тому рекомендується працювати від акумулятора. Найбільш ефективна система захисту від випромінювань реалізується через створення додаткового металічного внутрішнього корпусу, що замикається на вбудований закритий екран. За такої конструкції вдається зменшити електричне та електростатичне поле до фонових значень вже на відстані 5–7 см від корпуса, а за умови компенсації магнітного поля така конструкція забезпечує максимально можливу у наш час безпеку. Такі монітори коштують на 200−400 доларів дорожче звичайних.

5.2.6 Психофізіологічні фактори

Напруженість праці–це характеристика трудового процесу, яка обумовлює навантаження на центральну нервову систему, органи почуттів, емоційну сферу людини. Напруженість характеризується:

- інтелектуальними здібностями;
- роботою сенсорних механізмів;
- емоційними навантаженнями;-степеню монотонності;
- режимом роботи.

Важкість праці–це характеристика трудового процесу, які обумовлюють навантаження на опорно-рухливий апарат і функціональні системи організму людини (серцево-судинна, дихальна та інші). Важкість характеризується:

- фізичними динамічними навантаженнями;
- масою вантажу, що підіймається та переміщується;
- загальним числом стереотипних робочих рухів;
- розміром статичного навантаження;
- робочою позою (водій, кранівник та інші);
- ступенем нахилу тулуба.

Важкість та напруженість праці призводить до втомленості. Як наслідок –до механічних травм, пошкодження кістко-м’язового апарату людини.

Психофізіологічні фактори вибираються відповідно з Гігієнічною класифікацією праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу [43].

Психофізіологічні фактории при роботі з ПК:

- перенапруження зорових аналізаторів;
- монотонність трудового процесу;
- розумове перенапруження;
- нервово-емоційні перевантаження.

Класи умов праці за показниками напруженості праці:

1. Інтелектуальні навантаження:
   - Зміст роботи – рішення складних завдань з вибором за алгоритмом;

- Сприймання інформації та їх оцінка – сприймання інформації з наступною корекцією дій та операцій;
- Розподіл функцій за ступенем складності завдання - обробка, контроль, перевірка завдання.

2. Сенсорні навантаження:
   - Зосередження (%за зміну) – до 50%;
   - Щільність сигналів (звукові за1 год) – до 150;
   - Навантаження на слуховий аналізатор (%) – розбірливість слів та сигналів від 50 до 80 %;
   - Навантаження на голосовий апарат (протягом тижня) – від 20 до 25%.
3. Емоційне навантаження:
   - Ступінь відповідальності за результат своєї діяльності – є відповідальним за функціональну якість основної роботи; Ступінь ризику для власного життя – вірогідний;
   - Ступінь відповідальності за безпеку інших осіб – є відповідальним за безпеку інших.

Режим праці:
- Тривалість робочого дня – більше 8 год;
- Змінність роботи – однозмінна (без нічної зміни).

5.3 Безпека у надзвичайних ситуаціях. Визначення області працездатності багатоканальної вимірювальної системи на FPGA в умовах дії загрозливих чинників надзвичайних ситуацій

Система багатоканальних вимірювань спрямована на забезпечення вивіреною інформацією складних ситем не тільки цивільних споживачів інформації, але і військових. У зв’язку з тим, що багатоканальна вимірювальна система на FPGA має важливе значення для обороноздатності, а на них можуть справляти значний вплив загрозливі чинники надзвичайні ситуації різного типу, то необхідно провести оцінку безпеки роботи пристрою. До таких НС можна

віднести: стихійні лиха (землетруси, блискавка, зливи), а особливо впливовими на РЕА мають іонізуючі випромінювання та EMI. Тому при забезпеченні даних пристроїв слід забезпечити найвищий рівень захисту від тої чи іншої НС, оскільки кожна НС справляє свій вплив на дану систему.

Тож, в даній частині розділу необхідно виконати дослідження оцінки безпеки роботи та розробку заходів по підвищенню стійкості роботи багатоканальної вимірювальної системи на FPGA в умовах дії іонізуючих випромінювань та електромагнітного імпульсу.

В РЕА застосовуються елементи, до складу яких входять такі матеріали: метали, неорганічні матеріали (в основному діелектрики), провідники і різноманітні органічні сполуки (діелектрики, смоли і т.д.). Серед цих матеріалів метали найбільш чутливі до впливу іонізуючих випромінювань, оскільки їм властива висока концентрація вільних носіїв.

В радіоелектронній апаратурі іонізуючі випромінювання, викликають зворотні і незворотні процеси, внаслідок яких можуть відбуватися порушення роботи електричних елементів схеми, що призводять до виходу з ладу апаратури. Так, проходячи через елементи РЕА, потік гамма-випромінювань створює в них вільні носії електричних зарядів, в результаті переміщення яких виникає помилковий імпульс, який призводить до спрацьовування пристрою. При великих дозах випромінювання втрачають працездатність комплектуючі елементи систем радіоелектроніки і електроавтоматики. В результаті опромінення у транзисторах змінюється обернений струм і коефіцієнт підсилення, у конденсаторах знижуються напруги пробою та опір стікання, змінюється провідність і внутрішній нагрів; руйнується електрична ізоляція дротів з полімерних матеріалів. В органічних ізоляційних і діелектричних матеріалах змінюються такі параметри, як: електрична провідність, діелектрична проникність і тангенс кута втрат. Неорганічні матеріали менш чутливі до впливу іонізуючих випромінювань.

Для інженерної практики найбільший інтерес представляє перший випадок, тобто оцінка стійкості роботи РЕА при перебуванні її в зараженій

радіоактивними речовинами місцевості протягом певного часу після випадання радіоактивних речовин у даній місцевості.

EMI ушкоджує напівпровідникові прилади, резистори, конденсатори. Це являє велику небезпеку для апаратури, добре захищеної від впливу інших загрозливих чинників. Тому слід пам'ятати про те, що захист апаратури від механічних ушкоджень не захищає від впливу EMI і апаратура може втратити працездатність, знаходячись у надійних захисних спорудженнях.

5.3.1 Визначення області працездатності багатоканальної вимірювальної системи на FPGA в умовах дії іонізуючих випромінювань

За критерій безпеки роботи системи в цих умовах приймається таке граничне значення рівня ($Р_{зв}$, Р/год), при якому можуть виникнути тимчасові зміни, але системи буде працювати з потрібною якістю.

Приймаючи до уваги елементну базу, що використовується для реалізації розроблювальної системи, складається таблиця потужностей експозиційної дози опромінення для кожного елемента $Р_{зв.і}$, що викликають початок зворотних змін. Отримані значення занесемо до таблиці 5.1.

Таблиця 5.4 – Потужність експозиційної дози для кожного елемента, що викликають початок зворотних змін

№	Елементи вимірювальної системи на FPGA	$Р_{зв.і}$, Р/год	$Р_{зв.}$, Р/год
1	Процесори, інтегральні мікросхеми	10^3	10^3
2	Діоди кремнієві загального призначення	10^5	
3	Транзистори загального призначення	10^4	
4	Мікросхеми ТТЛ логіки	10^4	
5	Конденсатори керамічні	10^7	
6	Резистори МЛТ	10^8	

Визначається елемент, який найбільшою мірою піддається впливу випромінюванням, тобто елемент із мінімальним значенням $Р_{зв}$.

$$Р_{зв} = 10^3 \text{ (Р/год)}.$$

В якості критерію стійкості роботи системи використовується граничне значення рівня іонізуючих випромінювань:

$$Р_{гр} = К_{над*} Р_{зв*} К_{посл}, \quad (5.1)$$

де $Р_{зв}$ - рівень радіації незворотних змін пристрою в цілому;

$К_{над}$ - коефіцієнт надійності ($К_{над}$ =0,9÷0,95);

$К_{посл}$ - коефіцієнт послаблення.

$$Р_{гр} = 0,93* 10^3 *2 = 1860 \text{ (Р/год)}.$$

З наведеної таблиці слідує, що мінімальні значення граничних рівнів радіації елементів, при яких в елементній базі можливі незворотні зміни мають інтегральні мікросхеми великої ступені інтеграції та мікропроцесори – $Р_{зв}=10^3$, $k_{посл}=2$.

Визначаємо допустимий час роботи системи:

$$t_{доп} = \left(\frac{Д_{гр} \cdot К_{осл} + 2 \cdot P_1 \cdot \sqrt{1}}{2 \cdot P_1} \right)^2, \quad (5.2)$$

$$t_{доп} = \left(\frac{10^3 \cdot 2 + 2 \cdot 4,26 \cdot \sqrt{1}}{2 \cdot 4,26} \right)^2 = 15,354 (год).$$

Таким чином, допустимий час роботи системи складатиме 15,354 годин при максимальному рівні радіації 4,26 Р/с.

5.3.2 Визначення області працездатності багатоканальна вимірювальна система на FPGA в умовах дії електромагнітного імпульсу

За критерієм безпеки роботи багатоканальна вимірювальна система на FPGA в умовах дії електромагнітного імпульсу можна прийняти коефіцієнт безпеки:

$$K_б = 20\lg\frac{U_д}{U_г} \geq 40 \text{ (дБ)},$$

де $U_д$ – допустиме коливання напруги живлення (для мікросхем 5 В);

$U_г$ – напруга наведена за рахунок електромагнітних випромінювань у вертикальних (горизонтальних) струмопровідних частинах, В.

Допустимі коливання напруги живлення:

$$U_д = U_ж + \frac{Uж}{100} * N = 5 + \frac{5}{100} = 5{,}25(B).$$

В зв'язку з тим, що окремі елементи приладу можуть мати різні значення коефіцієнтів безпеки, то безпека роботи системи в цілому визначається мінімальним значенням коефіцієнта безпеки.

З рівняння (5.1) визначаємо:

$$U_г = \frac{Uд}{10^{\frac{40}{20}}} = \frac{5{,}25}{100} = 0{,}05(B).$$

Приймемо максимальну довжину горизонтальних струмопровідних частин $l_г$=0,086 м. Тоді горизонтальна складова напруженості електричного поля визначається за формулою:

$$E_г = U_в / l_в = 0{,}05/0{,}086 = 0{,}581 \text{ (В/м)}.$$

Звідси вертикальна складова напруженості буде $E_в$=581 В/м.

Таким чином, робота багатоканальна вимірювальна система на FPGA можлива у випадку, якщо не перевищується значення вертикальної складової напруженості електричного поля 581 В/м.

5.4 Розробка заходів по підвищенню безпеки роботи вимірювальної системи на FPGA в умовах надзвичайних ситуацій

З метою зменшення негативного впливу на багатоканальну вимірювальну система на FPGA можна використати наступні методи.

Для захисту розробки, як і будь-яких радіоелектронних пристроїв від дії іонізуючих випромінювань можна використати алюмінієві сплави, леговані елементами з високим атомним номером (лантаноїдами і рідкоземельними елементами), сплави на основі тугоплавких і рідкоземельних елементів і багатошарові матеріали. Також для боротьби з впливом іонізуючого випромінювання можна використати новітній вітчизняний метод, що полягає в захисному покритті радіоелектронної апаратури, що розміщується на поверхнях даних елементів, які піддаються впливу іонізуючого випромінювання, відмінним тим, що захисне покриття виконане у вигляді наноструктури, яка включає сукупність атомів рідкоземельних елементів, введених в структуру армованої атомно-молекулярної металічної матриці, або утворює її захисний шар.

Найкращим для захисту від електромагнітного імпульсу є захищене металічним екраном приміщення, в якому розміщена радіоелектронна апаратура. Оскільки такий захист в ряді випадків неможливо виконати, то використовуються менш надійні засоби захисту, такі як струмопровідні сітки та плівкові покриття вікон, стільникові металеві конструкції для повітрозбірників та вентиляційних отворів і контактні пружинні прокладки, що розміщуються по периметру дверей і люків. Для захисту від проникнення електромагнітного імпульсу в апаратуру через різні кабельні вводи використовується перехід від електричних мереж

зв’язку до практично не залежних від впливів EMI волоконно-оптичних. Також для захисту кабельних вводів використовують в їх конструкції фільтрів та встановлення вбудованих зенерівських діодів.

5.5 Висновки до п’ятого розділу

В ході виконання було розглянуто вплив іонізуючого випромінювання та EMI на компоненти схеми, виконано розрахунки з яких видно, що ні один з класів елементів схеми не зазнає більшого впливу за граничне значення, також розраховано термін безпечної роботи системи, який складає 15,354год. Що стосується впливу електромагнітного імпульсу, то з урахуванням необхідного рівня коефіцієнта безпеки було розраховано значення напруженості електричного поля. Для підвищення безпеки роботи багатоканальної вимірювальної системи на FPGA наведено основні заходи боротьби з впливом загрозливих чинників НС.

Отже основною метою даної частини розділу було дослідження безпеки роботи вимірювальної системи на FPGA та розробка заходів по підвищенню безпеки роботи багатоканальної вимірювальної системи в умовах надзвичайних ситуацій.

6 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота з розробки та дослідження на тему «Багатоканальна вимірювальна система на FPGA для радіовимірювальних частотних сенсорів» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

6.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Багатоканальна вимірювальна система на FPGA для радіовимірювальних частотних сенсорів» є оцінювання науково-технічного рівня та рівня

комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 6.1 [44].

Таблиця 6.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів

Продовження таблиці 6.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження таблиці 6.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці.

Таблиця 6.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	4	4	4
2. Ринкові переваги (наявність аналогів)	2	3	2
3. Ринкові переваги (ціна продукту)	3	3	3
4. Ринкові переваги (технічні властивості)	2	3	3
5. Ринкові переваги (експлуатаційні витрати)	2	2	3

Продовження таблиці 6.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	2	2	3
8. Практична здійсненність (наявність фахівців)	4	4	4
9. Практична здійсненність (наявність фінансів)	3	3	2
10. Практична здійсненність (необхідність нових матеріалів)	2	2	2
11. Практична здійсненність (термін реалізації)	4	3	4
12. Практична здійсненність (розробка документів)	3	3	3
Сума балів	34	35	36
Середньо арифметична сума балів $СБ_с$	35,0		

За результатами розрахунків, наведених в таблиці 6.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 6.3 [44].

Таблиця 6.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньо арифметична сума балів СБ , розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Багатоканальна вимірювальна система на FPGA для радіовимірювальних частотних сенсорів» становить 35,0 бала, що, відповідно до таблиці 6.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього).

6.2 Визначення рівня конкурентоспроможності розробки

В процесі визначення економічної ефективності науково-технічної розробки також доцільно провести прогноз рівня її конкурентоспроможності за сукупністю параметрів, що підлягають оцінюванню.

Одиничний параметричний індекс розраховуємо за формулою[44]:

$$q_i = \frac{P_i}{P_{базі}}. \quad (6.1)$$

де q_i– одиничний параметричний індекс, розрахований за i-м параметром;

P_i–значенняi-го параметра виробу;

$P_{базі}$– аналогічний параметр базового виробу-аналога, з яким проводиться порівняння.

Загальні технічні та економічні характеристики розробки представлено в таблиці 6.4.

Таблиця 6.4 – Основні техніко-економічні показники аналога та розробки, що проектується

Показники (параметри)	Одиниця вимірювання	Аналог	Проектований пристрій	Відношення параметрів нової розробки до аналога	Питома вага показника
Напруга живлення	В	5	5	1	0,15
Кількість виконуваних функцій	шт.	4	13	3,25	0,2
Максимально вимірювана частота	МГц	15,6	10	0,64	0,25
Термін безвідмовної роботи	год.	5000	6000	1,2	0,2
Швидкість та легкість налаштування	Бали (0 - 10)	7	6	0,86	0,2
Експлуатаційні витрати	грн	400	200	0,5	0,5
Ціна пристрою	грн	3050	2350	0,77	0,5

Нормативні параметри оцінюємо показником, який отримує одне з двох значень: 1 – пристрій відповідає нормам і стандартам; 0 – не відповідає.

Груповий показник конкурентоспроможності за нормативними параметрами розраховуємо як добуток частинних показників за кожним параметром за формулою[44]

$$I_{НП} = \prod_{i=1}^{n} q_i \,, \tag{6.2}$$

де $I_{нп}$ – загальний показник конкурентоспроможності за нормативними параметрами;

q_i – одиничний (частинний) показник за i-м нормативним параметром;

n – кількість нормативних параметрів, які підлягають оцінюванню.

За нормативними параметрами розроблюваний пристрій відповідає вимогам ДСТУ, тому $I_{нп}$= 1.

Значення групового параметричного індексу за технічними параметрами визначаємо з урахуванням вагомості (частки) кожного параметра[44]

$$I_{ТП} = \sum_{i=1}^{n} q_i \cdot \alpha_i \,, \tag{6.3}$$

де $I_{тп}$ – груповий параметричний індекс за технічними показниками (порівняно з виробом-аналогом);

q_i– одиничний параметричний показник i-го параметра;

α_i– вагомість i-гопараметричного показника, $\sum_{i=1}^{n} \alpha_i = 1$;

n – кількість технічних параметрів, за якими оцінюється конкурентоспроможність.

Проведемо аналіз параметрів згідно даних таблиці 6.4.

$$I_{mn} = 1 \cdot 0{,}15 + 3{,}25 \cdot 0{,}2 + 0{,}64 \cdot 0{,}25 + 1{,}2 \cdot 0{,}2 + 0{,}86 \cdot 0{,}2 = 1{,}37.$$

Груповий параметричний індекс за економічними параметрами розраховуємо за формулою[44]

$$I_{ЕП} = \sum_{i=1}^{m} q_i \cdot \beta_i \,, \tag{6.4}$$

де $I_{ЕП}$ – груповий параметричний індекс за економічними показниками;

q_i– економічний параметр i-го виду;

β_i – частка i-го економічного параметра, $\sum_{i=1}^{m} \beta_i = 1$;

m – кількість економічних параметрів, за якими здійснюється оцінювання.

Проведемо аналіз параметрів згідно даних таблиці .

$$I_{ЕП}=0{,}5\cdot 0{,}5 + 0{,}77\cdot 0{,}5 = 0{,}64.$$

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розрахуємо інтегральний показник конкурентоспроможності за формулою[44]

$$K_{ІНТ} = I_{НП} \cdot \frac{I_{ТП}}{I_{ЕП}}, \tag{6.5}$$

$$K_{ІНТ} = 1 \cdot 1{,}37 / 0{,}64 = 2{,}16.$$

Інтегральний показник конкурентоспроможності $К_{ІНТ} > 1$, отже розробка переважає відомі аналоги за своїми техніко-економічними показниками.

6.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Багатоканальна вимірювальна система на FPGA для радіовимірювальних частотних сенсорів», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

6.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [44]

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \quad (6.6)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=22 дні.

$$З_о = 12470{,}00 \cdot 22 / 22 = 12470{,}00 \ (грн.).$$

Проведені розрахунки зведемо до таблиці.

Таблиця 6.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	12470,00	566,82	22	12470,00
Ст. науковийспівробітник	11630,00	528,64	12	6343,64
Інженер-метролог	7200,00	327,27	5	1636,36
Інженер-радіотехнік	11500,00	522,73	18	9409,09
Лаборант	6500,00	295,45	10	2954,55
Всього				32813,64

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Багатоканальнавимірювальна система на FPGA для радіовимірювальних частотних сенсорів» розраховуємо за формулою

$$З_р = \sum_{i=1}^{n} C_i \cdot t_i , \qquad (6.7)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} , \qquad (6.8)$$

де M_M – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо *Мм*=2379,00 (грн.);

$K_і$– коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду;

K_c – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об’єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

T_p – середнє число робочих днів в місяці, приблизно T_p = 22дн;

$t_{зм}$ – тривалість зміни, год.

C_1 = 2379,00 · 1,10 · 1,65 / (22 · 8) = 24,53 *(грн.).*
$З_{р1}$ = 24,53 · 7,50 = 184,00 *(грн.).*

Таблиця 6.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Установка обладнання	7,50	2	1,10	24,53	184,00
Підготовка робочого місця розробника багатоканальної вимірювальної системи для радіовимірювальнихчастотних сенсорів	12,00	3	1,35	30,11	361,31
Встановлення програмного забезпечення розробки електронних схем	6,20	4	1,50	33,45	207,42
Підготовкабази даних	10,00	3	1,35	30,11	301,09
Монтаж компонентів багатоканальної вимірювальної системи на FPGA для радіовимірювальних частотних сенсорів	16,00	5	1,70	37,92	606,65
Випробування дослідних блоків	10,00	5	1,70	37,92	379,15
Налагодженняс истеми	4,00	6	2,00	44,61	178,43
Технічна підтримка експериментів	16,00	3	1,35	30,11	481,75
Всього					2699,79

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10-12% від суми основної заробітної плати дослідників та робітників за формулою

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (6.9)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 11%.

$З_{дод} = (32813,64 + 2699,79) \cdot 11 / 100\% = 3906,48$ *(грн.)*.

6.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (6.10)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (32813,64 + 2699,79 + 3906,48) · 22 / 100% = 8672,38(грн.).

6.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Багатоканальна вимірювальна система на FPGA для радіовимірювальних частотних сенсорів».

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою

$$M = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j - \sum_{j=1}^{n} В_j \cdot Ц_{в\,j}\,, \quad (6.11)$$

де $Н_j$– норма витрат матеріалу *j*-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу *j*-го найменування, грн/кг;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ *= 1,1 ... 1,15*);

$В_j$ – маса відходів *j*-го найменування, кг;

$Ц_{вj}$ – вартість відходів *j*-го найменування, грн/кг.

$М_1$ = 5,00 · 94,00 · 1,1 - 0,000 · 0,00 = 517,00 (грн.).

Проведені розрахунки зведемо до таблиці.

Таблиця 6.7– Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Папір	94,00	5,00	0,000	0,00	517,00
Канцелярське приладдя	190,00	5,00	0,000	0,00	1045,00
Тека паперова	30,00	6,00	0,000	0,00	198,00
Дріт монтажний	80,00	0,05	0,000	0,00	3,96
Спирт етиловий	92,00	0,24	0,000	0,00	24,29
Припій ПОС-61	590,00	0,04	0,000	0,00	22,72
Флюс БС-2	120,00	0,01	0,000	0,00	1,32
Кабель силовий	65,00	0,20	0,000	0,00	14,30
Всього					1826,58

6.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Багатоканальна вимірювальна система на FPGA для радіовимірювальних частотних сенсорів», розраховуємо, згідно з їхньою номенклатурою, за формулою

$$К_в = \sum_{j=1}^{n} Н_j \cdot Ц_j \cdot К_j \qquad (6.12)$$

де $Н_j$ – кількість комплектуючих j-го виду, шт.;

$Ц_j$ – покупна ціна комплектуючих j-го виду, грн;

$К_j$ – коефіцієнт транспортних витрат, ($К_j$ *= 1,1 … 1,15*).

$К_в = 1 \cdot 1303{,}00 \cdot 1{,}1 = 1433{,}30$ *(грн.).*

Проведені розрахунки зведемо до таблиці.

Таблиця 6.8– Витрати на комплектуючі

Найменування комплектуючих	Кількість, шт.	Ціна за штуку, грн	Сума, грн
WaveShare плата розробки ALTERA CYCLONE IV EP4CE10	1	1303,00	1433,30
Датчик MPU-6050	1	63,00	69,30
Резистивний датчик тиску FSR402	1	227,00	249,70
Резистори	4	0,53	2,33
Імпульсний адаптер живлення Merlion MLPSP5-1mini, 5В 1А	1	61,00	67,10
Конектори Dupont	24	1,00	26,40
Макетна плата MB-102 400 отворів	1	51,00	56,10
Всього			1904,23

6.3.5 Спец устаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (6.13)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань устаткування.

$B_{спец} = 16200,00 \cdot 1 \cdot 1,1 = 17820,00$ *(грн.).*

Отримані результати зведемо до таблиці

Таблиця 6.9 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Метрологічна станція МС-ХЧК з комплектом інтерфейсів частотного вимірювання	1	16200,00	17820,00
Всього			17820,00

6.3.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (6.14)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань програмних засобів.

$B_{прг} = 5630,00 \cdot 1 \cdot 1,1 = 6193,00$ *(грн.).*

Отримані результати зведемо до таблиці:

Таблиця 4.10 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
ОС Windows	1	5630,00	6193,00
Прикладний пакет MicrosoftOffice	1	5140,00	5654,00
Прикладний пакет моделювання процесів MatLab	1	7640,00	8404,00
Всього			20251,00

6.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (6.15)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл}$ = (24600,00 · 1) / (2 · 12) = 1025,00 (грн.).

Проведені розрахунки зведемо до таблиці.

Таблиця 6.11 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональнийкомп'ютер ПЕОМ	24600,00	2	1	1025,00

Продовження таблиця 6.11 – Амортизаційні відрахування по кожному виду обладнання

Обчислювальний комплекс обробкиданих	25400,00	2	1	1058,33
Робоче місце розробника	10300,00	5	1	171,67
Пристрій графічного виводу інформації	9400,00	4	1	195,83
Оргтехніка	9600,00	4	1	200,00
Приміщення лабораторії	275000,00	20	1	1145,83
Частотомір цифровий ЧМ-СЦ12	5600,00	4	1	116,67
Генератор еталонної частоти ГЧ-1880	10500,00	4	1	218,75
Всього				4132,08

6.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \qquad (6.16)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт;

t_i– тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 4,50 (грн.);

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni}<1$;

η_i– коефіцієнт корисної дії обладнання, $\eta_i<1$.

$B_e = 0,20 \cdot 164,0 \cdot 4,50 \cdot 0,95 / 0,97 = 147,60$ *(грн.).*

Проведені розрахунки зведемо до таблиці.

Таблиця 6.12 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер ПЕОМ	0,20	164,0	147,60
Обчислювальний комплекс обробкиданих	0,25	164,0	184,50
Робоче місце розробника	0,15	100,0	67,50
Пристрій графічного виводу інформації	0,40	8,5	15,30
Оргтехніка	0,56	6,0	15,12
Частотомір цифровий ЧМ-СЦ12	0,10	86,0	38,70
Генератор еталонної частоти ГЧ-1880	0,26	86,0	100,62
Метрологічна станція МС-ХЧК з комплектом інтерфейсів частотного вимірювання	0,20	20,0	18,00
Всього			587,34

4.3.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Багатоканальна вимірювальна система на FPGA для радіовимірювальних частотних сенсорів» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20-25% від суми основної заробітної плати дослідників та робітників за формулою

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \quad (6.17)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 22%.

$В_{св}$ = (32813,64 + 2699,79) · 22 / 100% = 7812,95 (грн.).

6.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30-45% від суми основної заробітної плати дослідників та робітників за формулою

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \quad (6.18)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 30%.

$В_{сп}$ = (32813,64 + 2699,79) · 30 / 100% = 10654,03 (грн.).

6.3.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50-100% від суми основної заробітної плати дослідників та робітників за формулою

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (4.19)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 60%.

$І_в$ = (32813,64 + 2699,79) · 60 / 100% = 21308,06 (грн.).

6.3.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100-150% від суми основної заробітної плати дослідників та робітників за формулою

$$B_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (4.20)$$

де $Н_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 110%.

$В_{нзв}$ = (32813,64 + 2699,79) · 110 / 100% = 39064,77 (грн.).

Витрати на проведення науково-дослідної роботи на тему «Багатоканальнавимірювальна система на FPGA для радіовимірювальнихчастотнихсенсорів» розраховуємо як суму всіх попередніх статей витрат за формулою

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} + В_{сп} + І_в + В_{нзв}. \quad (4.21)$$

$В_{заг}$= 32813,64 +2699,79 +3906,48 +8672,379519 +1826,58 +1904,23 +17820,00 + 20251,00 + 4132,08 +587,34 +7812,95 +10654,03 +21308,06 +39064,77 = =173453,34 (грн.).

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою

$$ЗВ = \frac{В_{заг}}{\eta}, \quad (4.22)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,8.

$$ЗВ = 173453,34 / 0,8 = 216816,67 (грн.).$$

6.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Багатоканальна вимірювальна система на FPGA для радіовимірювальних частотних сенсорів» передбачають комерціалізацію протягом 4-х років реалізації на ринку.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів пристрою, у періоди часу, що аналізуються, від покращення його певних характеристик;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	500	750	1000	500

N – кількість споживачів які використовували аналогічний пристрійу році до впровадження результатів нової науково-технічної розробки, приймемо 7000 осіб;

$Ц_б$ – вартість пристрою у році до впровадження результатів розробки, приймемо 2300,00 (грн.);

$\pm\Delta Ц_о$ – зміна вартості пристрою від впровадження результатів науково-технічної розробки, приймемо 55,60 (грн.).

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [45]:

$$\Delta П_i = (\pm \Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (6.23)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2021 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо $\rho = 27\%$;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2021 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (55{,}60 \cdot 7000{,}00 + 2355{,}60 \cdot 500) \cdot 0{,}83 \cdot 0{,}27 \cdot (1\text{-}0{,}18/100\%) = 287955{,}05$ (грн.).

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (55{,}60 \cdot 7000{,}00 + 2355{,}60 \cdot 1250) \cdot 0{,}83 \cdot 0{,}27 \cdot (1\text{-}0{,}18/100\%) = 612607{,}38$ (грн.).

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (55{,}60 \cdot 7000{,}00 + 2355{,}60 \cdot 2250) \cdot 0{,}83 \cdot 0{,}27 \cdot (1\text{-}0{,}18/100\%) = 1045477{,}15$ (грн.).

Збільшення чистого прибутку 4-го року:

$\Delta П_4 = (55{,}60 \cdot 7000{,}00 + 2355{,}60 \cdot 2750) \cdot 0{,}83 \cdot 0{,}27 \cdot (1\text{-}0{,}18/100\%) = 1261912{,}03$ (грн.).

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \quad (6.24)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,08;

t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 287955,05/(1+0,08)^1+612607,38/(1+0,08)^2+1045477,15/(1+0,08)^3+$$
$$+1261912,03/(1+0,08)^4=266625,05+525212,09+829933,47+$$
$$+927543,01=2549313,62 \text{ (грн.)}.$$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки

$$PV = k_{інв} \cdot ЗВ, \quad (6.25)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 216816,67 (грн.).

$PV = k_{інв} \cdot ЗВ = 2 \cdot 216816,67 = 433633,35$ (грн.).

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме

$$E_{абс} = ПП - PV \quad (6.26)$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 2549313,62 (грн.);

PV – теперішня вартість початкових інвестицій, 433633,35 (грн.).

$E_{абс} = ПП - PV$ = 2549313,62 - 433633,35 = 2115680,27 (грн.).

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (6.27)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 2115680,27 (грн.).;

PV – теперішня вартість початкових інвестицій, 433633,35 (грн.).;

$T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1$ = (1+2115680,27/433633,35)$^{1/4}$= 0,56.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$

$$\tau_{мін} = d + f, \quad (6.28)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2021 році в Україні d =0,1;

f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,06.

$\tau_{мін}$= 0,1+0,06 = 0,16<0,56 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної

внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Багатоканальна вимірювальна система на FPGA для радіовимірювальних частотних сенсорів» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки

$$T_{ок} = \frac{1}{E_в}, \tag{6.29}$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій.

$T_{ок}$ = 1 / 0,56 = 1,79 р.

$T_{ок} <$ 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

6.5 Висновки до шостого розділу

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Багатоканальна вимірювальна система на FPGA для радіовимірювальних частотних сенсорів» становить 35,0 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього). При оцінюванні рівня конкуренто спроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 2,16 рази. Також термін окупності становить 1,79 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Багатоканальна вимірювальна система на FPGA для радіовимірювальних частотних сенсорів».

ВИСНОВКИ

1. Проведено аналіз існуючих типів частотомірів і принципів які лежать в основі їх реалізації: електронно-лічильні, рахують кількість імпульсів вхідного сигналу за визначений проміжок часу, універсальні, великий діапазон частот, точність; резонансні, порівнюють частоту вхідного сигналу з особистою резонансною частотою; гетеродинні, порівнюють частоту вхідного сигналу з частотою гетеродинна використовуючи метод «нульового биття»; конденсаторні, заряд конденсатора від батареї з наступним його розрядом через електромагнітний механізм, частота відповідає вимірюваній частоті, похибка становить 2-3%; вібраційні, перетворюють електричні коливання у механічні. На основі проведеного аналізу було вирішено використовувати принципи вимірювання частоти які лежать в основі електронно-лічильних частотомірів, оскільки, вони: дозволяють вимірювати великий діапазон значень, не потребують додаткових дій для налаштування, можуть бути реалізовані на FPGA.

2. Розглянуто принципи роботи які лежать в основі реалізації FPGA. Проаналізовано різні апаратні платформи і випадки їх застосування: CPU, GPU, MCU, FPGA. Розроблено багатоканальну систему вимірювання частоти, основною задачею якої являється вимірювання значень сенсорів з частотним виходом.

3. Розглянуто складові архітектури процесорних систем на основі ядра «Nios II» і спроектовано свою процесорну систему яка складається з блоків присутніх у стандартній бібліотеці і з особисто реалізованих блоків. Розглянуто протокол передачі даних I2C і реалізовано його у вигляді апаратного блоку на FPGA. Перероблено схему частотомірів для можливості інтеграції процесорної системи з додатковим набором інтерфейсів і підтримкою датчиків з цифровим входом. Розроблено і протестовано програмне забезпечення під створену процесорну систему для підтримки усіх реалізованих інтерфейсів.

4. Розроблено програму «SerialMonitor», яка дозволяє відображати отримані від приладу дані у вигляді графіків, для спрощення аналізу і оцінки. Для розробки програмного забезпечення використовувалась мова програмування python і середовище розробки PyCharm.

5. В ході виконання було розглянуто вплив іонізуючого випромінювання та EMI на компоненти схеми, виконано розрахунки з яких видно,

що ні один з класів елементів схеми не зазнає більшого впливу за граничне значення, також розраховано термін безпечної роботи системи, який складає 15,354год. Що стосується впливу електромагнітного імпульсу, то з урахуванням необхідного рівня коефіцієнта безпеки було розраховано значення напруженості електричного поля. Для підвищення безпеки роботи багатоканальної вимірювальної системи на FPGA наведено основні заходи боротьби з впливом загрозливих чинників НС.

6. Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Багатоканальна вимірювальна система на FPGA для радіовимірювальних частотних сенсорів» становить 35,0 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки вище середнього). При оцінюванні рівня конкуренто спроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 2,16 рази. Також термін окупності становить 1,79 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

7. В кінцевому результаті отримано:багатоканальну вимірювальну систему яка одночасно може працювати із сенсорами принцип роботи яких базується на використанні функціональної залежності реактивних властивостей транзисторних структур з від’ємним опором і цифровими сенсорами фізичних величин; програмне забезпечення для відображення виміряних значень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Види і класифікація частотомірів. [Електронний ресурс]: - Режим доступу: https://radio-detaly.com/uk/chastotomir. Дата доступу: 15.09.2021.
2. Класифікація частотомірів. [Електронний ресурс]: - Режим доступу: https://radio-detaly.com/uk/chastotomir. Дата доступу: 15.09.2021.
3. Хамадулин Э.Ф. Методы и средства измерений в телекоммуникационных системах. –Москва: «Юрайт», 2018. –226 с.
4. Хамадулин Э.Ф. Методы и средства измерений в телекоммуникационных системах. –Москва: «Юрайт», 2018. –223 с.
5. Heterodyne. [Електронний ресурс]: - Режим доступу: https://en.wikipedia.org/wiki/Heterodyne. Дата доступу: 20.09.2021.
6. Резонансні і гетеродинні методи вимірювання частоти. [Електронний ресурс]: - Режим доступу: http://univer64.ru/rezonansnye-i-geterodinnye-metody-izmerenija.html. Дата доступу: 20.09.2021.
7. Миронов Э.Г., Бессонов Н.П. Метрология и технические измерения. – Москва: «КноРус», 2016. –296 с.
8. Евтихаева Н.Н. Измерения электрических и неэлектрических величин. – Москва: «Энергоатомиздат», 1990.
9. Классен К.Б. Основы измерений. Электронные методы и приборы в измерительной технике. –Москва: «Постмаркет», 2000.
10. Корн Г., Корн. Т. Справочник по математике для научных работников и инженеров. –Москва: «Наука», 1970.
11. Электромеханический вибрационный частотомер. [Електронний ресурс]: - Режим доступу: https://www.booksite.ru/fulltext/1/001/008/121/581.htm. Дата доступу: 22.09.2021.
12. Reverse-engineering the First FPGA Chip Xilinx XC2064. [Електронний ресурс]: - Режим доступу: https://semiwiki.com/fpga/290990-reverse-engineering-the-first-fpga-chip-xilinx-xc2064/. Дата доступу: 22.09.2021.
13. ASIC Design and Verification in an FPGA Environment. [Електронний ресурс]: - Режим доступу: https://people.eecs.berkeley.edu/~bora/publications/CICC07.pdf. Дата доступу: 22.09.2021.

14. Cyclone IV Device Handbook. [Електронний ресурс]: - Режим доступу: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-iv/cyclone4-handbook.pdf . Дата доступу: 23.09.2021.
15. FPGA Architecture. [Електронний ресурс]: - Режим доступу: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/wp/wp-01003.pdf . Дата доступу: 23.09.2021.
16. Строгонов А.В. СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ ИНТЕГРАЛЬНЫХ СХЕМ. –Воронеж, 2012.
17. Cyclone IV Device Datasheet. [Електронний ресурс]: - Режим доступу: https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/cyclone-iv/cyiv-53001.pdf. Дата доступу: 24.09.2021.
18. Quartus Prime Standard Edition. [Електронний ресурс]: - Режим доступу: https://fpgasoftware.intel.com/15.1/?edition=standard&platform=windows. Дата доступу: 24.09.2021.
19. American Standard Code for Information Interchange. [Електронний ресурс]: - Режим доступу: https://www.asciitable.com/ . Дата доступу: 27.09.2021.
20. Hardware Communication Protocol. [Електронний ресурс]: - Режим доступу: https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html. Дата доступу: 27.09.2021.
21. Осадчук В.С., Осадчук О.В., Вербицький В.Г. Температурні та оптичні мікроелектронні частотні перетворювачі. – Вінниця: «УНІВЕРСУМ – Вінниця», 2001. – 195 с.
22. Осадчук В.С., Осадчук О.В., Крилик Л.В. Сенсори вологості – Вінниця: «УНІВЕРСУМ – Вінниця», 2003. – 208 с.
23. Осадчук О.В. Мікроелектронні частотні перетворювачі на основі транзисторних структур з від'ємним опором. – Вінниця: «УНІВЕРСУМ – Вінниця», 2000. – 303 с.
24. Осадчук А.В. Фоточувствительные преобразователи на основе структур с отрицательным сопротивлением. – Винница: Континент, 1998. – 130 с.
25. Осадчук В.С., Осадчук О.В. Напівпровідникові прилади з від'ємним опором. Навчальний посібник. – Вінниця: ВНТУ, 2006.– 162 с.

26. Осадчук В.С., Осадчук О.В., Ющенко Ю.А. Радіовимірювальні мікроелектронні перетворювачі витрат газу з частотним виходом. – Вінниця: ВНТУ, 2012. – 140 с.
27. Жмудь В.А., Гончаренко А.М. Прецизионный частотомер для фундаментальной метрологии. – Новосибирск: «Новосибирский государственный технический университет», 2014.
28. Nios II Processor Reference Handbook. – San Jose: Altera, 2016. – 2 с.
29. Nios II Processor Reference Handbook. – San Jose: Altera, 2016. – 9 с.
30. Nios II Processor Reference Handbook. – San Jose: Altera, 2016. – 11 с.
31. IEEE Standard for Binary Floating-Point Arithmetic. [Електронний ресурс]: - Режим доступу: https://ieeexplore.ieee.org/servlet/opac?punumber=2355. Дата доступу: 06.10.2021.
32. THE I2C-BUS SPECIFICATION. – Amsterdam: Philips Semiconductors, 1998.
33. Python. [Електронний ресурс]: - Режим доступу: https://www.python.org/. Дата доступу: 24.10.2021.
34. The Python IDE for Professional Developers by JetBrains. [Електронний ресурс]: - Режим доступу: https://www.python.org/. Дата доступу: 24.10.2021.
35. Qt for Python. [Електронний ресурс]: - Режим доступу: https://doc.qt.io/qtforpython. Дата доступу: 24.10.2021.
36. ГОСТ 12.0.003-74 ССБТ. Опасные и вредные производственные факторы. Классификация.
37. Правила улаштування електроустановок - [Електронний ресурс] - Режим доступу: http://www.energiy.com.ua/PUE.html
38. Санітарні норми мікроклімату виробничих приміщень. ДСН 3.3.6.042−99 [Електронний ресурс]. –Режим доступу: http//www.dnaop.com.
39. НПАОП 0.00-7.15-18 Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями. - [Електронний ресурс] - Режим доступу: http://sop.zp.ua/norm_npaop_0_00-7_15-18_01_ua.php
40. СНиП 2.04.05-91*У. Отопление, вентиляция и кондиционирование[Електронний ресурс] - Режим доступу: https://dnaop.com/html/1671/doc-%D0%A1%D0%9D%D0%B8%D0%9F_2.04.05-91_%D0%A3

41. ДБН В.2.5-28:2018 Природне і штучне освітлення - [Електронний ресурс] - Режим доступу: http://online.budstandart.com/ua/catalog/doc-page.html?id_doc=79885
42. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку. - [Електронний ресурс] - Режим доступу: http://document.ua/sanitarni-normi-virobnichogo-shumu-ultrazvuku-ta-infrazvuku-nor4878.html
43. Наказ від 08.04.2014 № 248 Про затвердження Державних санітарних норм та правил Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу - [Електронний ресурс] - Режим доступу: http://online.budstandart.com/ua/catalog/topiccatalogua/laborprotection/14._nakazy_ta_rozpor_183575/248+58074-detail.html
44. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.
45. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

Додаток А
(обов’язковий)

ЗАТВЕРДЖУЮ
Зав.кафедри РТ ВНТУ,
докт. техн. наук, професор
О.В. Осадчук
“__” _____ 2021 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання магістерської кваліфікаційної роботи
Багатоканальна вимірювальна система на FPGA для радіовимірювальних частотних сенсорів
08-36.МКР.003.00.000 ТЗ

Керівник МКР
д.т.н., проф. зав. кафедри РТ ВНТУ
Осадчук О. В.

Розробив студент гр. РТ-20м
Скощук В. К.

Вінниця-2021

1 ПІДСТАВА ДЛЯ ВИКОНАННЯ РОБОТИ

Робота проводиться на підставі наказу ректора по Вінницькому національному технічному університету № від р. та індивідуального завдання на магістерську кваліфікаційну роботу.

Дата початку роботи: 03.09.2021 р.

Дата закінчення: 20.12.2021 р.

2 МЕТА І ПРИЗНАЧЕННЯ МКР

Метою роботи є розробка багатоканальної вимірювальної системи яка одночасно зможе працювати із сенсорами принцип роботи яких базується на використанні функціональної залежності реактивних властивостей транзисторних структур з від'ємним опором і цифровими сенсорами фізичних величин.

Об'єктом дослідження є процес об'єднання вимірювань сенсорів з частотним і цифровим виходами.

Предметом дослідження – вимірювальна система на основі FPGA з можливістю вимірювання частоти і підтримкою сучасних цифрових інтерфейсів сенсорів фізичних величин.

Для досягнення поставленої мети у магістерській кваліфікаційній роботі розв'язуються такі *задачі*:

- проаналізувати існуючі методи вимірювання частоти і виділити найбільш перспективний для реалізації на FPGA;
- розробити багатоканальний радіовимірювальний прилад частоти на основі FPGA фірми Altera;
- розширити функціональні можливості розробленого приладу для підтримки сенсорів з цифровим перетворювачем;
- розробити спеціалізоване програмне забезпечення для тестування вимірювальної системи.
- виконати експериментальну перевірку розробленої багатоканальної вимірювальної системи використовуючи спеціалізоване програмне забезпечення.

3 ДЖЕРЕЛА РОЗРОБКИ

1. Осадчук В.С., Осадчук О.В., Вербицький В.Г. Температурні та оптичні мікроелектронні частотні перетворювачі. – Вінниця: «УНІВЕРСУМ – Вінниця», 2001. – 195 с.
2. Осадчук В.С., Осадчук О.В., Крилик Л.В. Сенсори вологості – Вінниця: «УНІВЕРСУМ – Вінниця», 2003. – 208 с.
3. Осадчук О.В. Мікроелектронні частотні перетворювачі на основі транзисторних структур з від'ємним опором. – Вінниця: «УНІВЕРСУМ – Вінниця», 2000. – 303 с.
4. Осадчук А.В. Фоточувствительные преобразователи на основе структур с отрицательным сопротивлением. – Винница: Континент, 1998. – 130 с.

5. ГОСТ 12.0.003-74 ССБТ. Опасные и вредные производственные факторы. Классификация.
6. Правила улаштування електроустановок - [Електронний ресурс] - Режим доступу: http://www.energiy.com.ua/PUE.html
7. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.
8. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

4 ВИКОНАВЕЦЬ

Вінницький національний технічний університет, кафедра радіотехніки, студент групи РТ-20м Скощук В. К.

5 ВИМОГИ ДО ВИКОНАННЯ МКР

Багатоканальна вимірювальна система на основі FPGA Cyclone IV EP4CE10F17C8 компанії Altera, напруга живлення 5В, амплітуда вхідного сигналу від 2.5...5В, діапазон вимірюваних частот від 10Гц до 10МГц.

Універсальний вимірювальний прилад, який має 12 вимірювальних каналів для сенсорів з частотним виходом і підтримує одночасну роботу з 127 цифровими сенсорами через I2C інтерфейс. У якості вихідного інтерфейсу використовується цифровий протокол UART.

Основними вимогами є:

- вимірювання значень сенсорів з частотним виходом на FPGA;
- вимірювання значень сенсорів з цифровим виходом.

6 ЕТАПИ МКР І ТЕРМІНИ ЇХ ВИКОНАННЯ

№	Назва та зміст етапу	Термін виконання		Очікувані результати	Звітна документація
		Початок	Закінчення		
1.	Огляд літературних джерел. Вибір та узгодження МКР.	03.09.2021	14.09.2021	Проведено огляд літературних джерел. Вибрана тема	Узгодження теми МКР по кафедрі
2.	Аналіз літературних джерел. Попередня розробка основних розділів.	15.09.2021	21.09.2021	Аналіз літературних джерел. Підготовлений матеріал основних розділів.	Вступ

3.	Затвердження теми. Розробка технічного завдання.	21.09.2021	25.09.2021	Розроблене ТЗ	Наказ ВНТУ про затвердження теми. Додаток А
4.	Аналіз вирішення поставленої задачі. Розробка структурної схеми.	26.09.2021	09.10.2021	Проведений аналіз. Розроблені схеми пристрою	Вступ. Розділ 1-2. Звіт по переддипломній практиці.
5.	Електричні розрахунки. Експериментальне дослідження.	10.10.2021	03.11.2021	Проведені розрахунки та дослідження.	Розділ 3-4.
6.	Розділ моделювання	04.11.2021	12.11.2021	Проведено моделювання	Результати моделювання
7.	Розробка графічної частини МКР	13.11.2021	18.11.2021	Структурні та електричні схеми	Графічна частина
8.	Аналіз економічної ефективності розробки	19.11.2021	24.11.2021	Економічна частина	Розділ 5
9.	Охорона праці (ОП)	25.11.2021	30.11.2021	Частина БЖД	Розділ 6
10.	Оформлення пояснювальної записки та графічної частини	01.12.2021	08.12.2021	Оформлена документація	ПЗ та графічна частина
11.	Нормконтроль	09.12.2021	12.12.2021	Підпис нормконтроля	Оформлена ПЗ та графічна частина
12.	Попередній захист МКР, доопрацювання рецензування МКР	13.12.2021	19.12.2021	Позитивні відзиви	Відзив. Рецензія
13.	Захист МКР ЕК	21.12.2021	23.12.2021	Позитивний захист	Протокол ЕК

7 ОЧІКУВАНІ РЕЗУЛЬТАТИ ТА ПОРЯДОК РЕАЛІЗАЦІЇ МКР

У результаті виконання роботи будуть розроблені:

- теоретичний підхід до реалізації системи на FPGA яка одночасно взаємодіє з частотними і цифровими сенсорами;

- практична реалізація багатоканального вимірювального приладу і його схема;
- розділ безпеки життєдіяльності і ЦЗ;
- економічна частина МКР.

Результати, отримані в процесі виконання даної роботи, можуть бути впроваджені в різних галузях науки і техніки.

8 МАТЕРІАЛИ, ЯКІ ПОДАЮТЬ ПІСЛЯ ЗАКІНЧЕННЯ РОБОТИ ТА ПІД ЧАС ЕТАПІВ

За результатами виконання МКР до ЕК подаються пояснювальна записка, графічна частина МКР, відзив і рецензія.

9 ПОРЯДОК ПРИЙМАННЯ МКР ТА ЇЇ ЕТАПІВ

Поетапно результати виконання МКР розглядаються керівником роботи та обговорюються на засіданні кафедри.

Захист магістерської кваліфікаційної роботи відбувається на відкритому засіданні ЕК.

10 ВИМОГИ ДО РОЗРОБЛЮВАНОЇ ДОКУМЕНТАЦІЇ

Документація, що розробляється в процесі виконання досліджень повинна містити:

- техніко-економічне обґрунтування розробки;
- дослідження поставленого питання;
- багатоканальний вимірювальний пристрій на основі FPGA Cyclone IV EP4CE10F17C8 компанії Altera який має 12 вимірювальних каналів для сенсорів з частотним виходом і підтримує одночасну роботу з 127 цифровими сенсорами через I2C інтерфейс. У якості вихідного інтерфейсу використовується цифровий протокол UART;
- схематичне зображення розробленого пристрою;
- економічну частину та розділ ОП та ЦЗ.

11 ВИМОГИ ЩОДО ТЕХНІЧНОГО ЗАХИСТУ ІНФОРМАЦІЇ З ОБМЕЖЕНИМ ДОСТУПОМ

У зв’язку з тим, що інформація не є конфіденційною, заходи з її технічного захисту не передбачаються.

Додаток Б
(довідниковий)

БАГАТОКАНАЛЬНА ВИМІРЮВАЛЬНА СИСТЕМА НА FPGA ДЛЯ РАДІОВИМІРЮВАЛЬНИХ ЧАСТОТНИХ СЕНСОРІВ

Характеристики сімейства чіпів CycloneIV

Таблиця Б.1 – Характеристики вибраного сімейства чіпів Cyclone IV

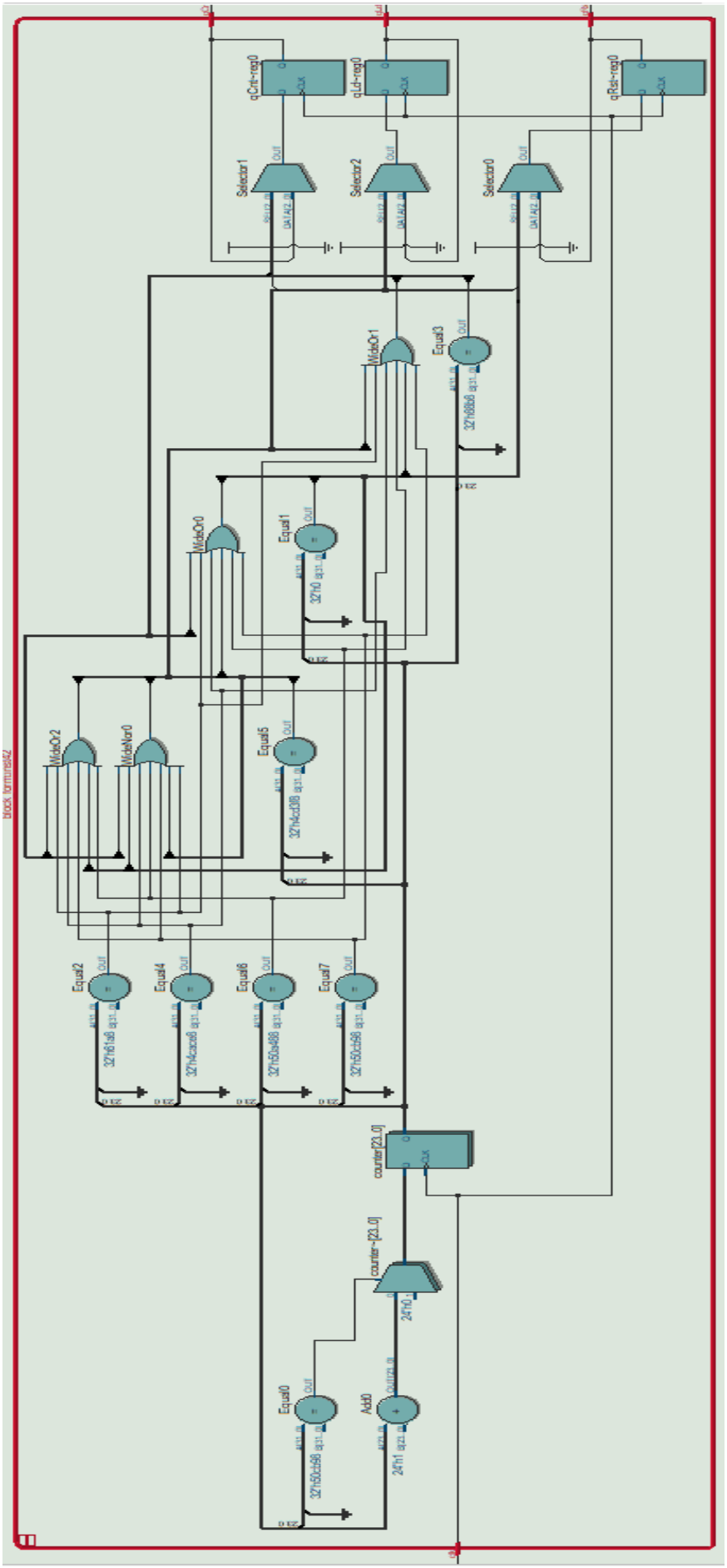
Ресурси	**P4CE6**	**P4CE10**	**EP4CE15**	**EP4CE22**	**P4CE30**	**EP4CE40**	**P4CE55**	**EP4CE75**	**EP4CE115**
Логічні елементи (LE)	6,272	10,320	15,408	22,320	28,848	39,600	55,856	75,408	114,480
Вбудована пам’ять (Kbits)	270	414	504	594	594	1,134	2340	2,745	3888
Вбудовані помножувачі 18*18	15	23	56	66	66	116	154	200	266
PLL загального призначення	2	2	4	4	4	4	4	4	4
Global Clock Networks	10	10	20	20	20	20	20	20	20
I/O Блоки	8	8	8	8	8	8	8	8	8
Максималтна кількість I/O	179	179	343	153	532	532	374	426	528

Додаток В
(обов’язковий)

БАГАТОКАНАЛЬНА ВИМІРЮВАЛЬНА СИСТЕМА НА FPGA ДЛЯ РАДІОВИМІРЮВАЛЬНИХ ЧАСТОТНИХ СЕНСОРІВ

Генератор імпульсів керування

Структурна схема


qCnt~reg0
qLd~reg0
qRst~reg0
Selector1
Selector2
Selector0
WideOr1
Equal3
32'h6bb6
WideOr0
Equal1
32'h0
WideOr2
WideNor0
Equal5
32'h4cd98
Equal2
32'h61d8
Equal4
32'h4cace8
Equal6
32'h50a488
Equal7
32'h50cb98
counter[23..0]
counter~[23..0]
24'h0
Equal0
32'h50cb98
Add0
24'h1
block formula42

Додаток Д
(довідниковий)

БАГАТОКАНАЛЬНА ВИМІРЮВАЛЬНА СИСТЕМА НА FPGA ДЛЯ РАДІОВИМІРЮВАЛЬНИХ ЧАСТОТНИХ СЕНСОРІВ

Опис блоку формувача сигналів керування на мові Verilog

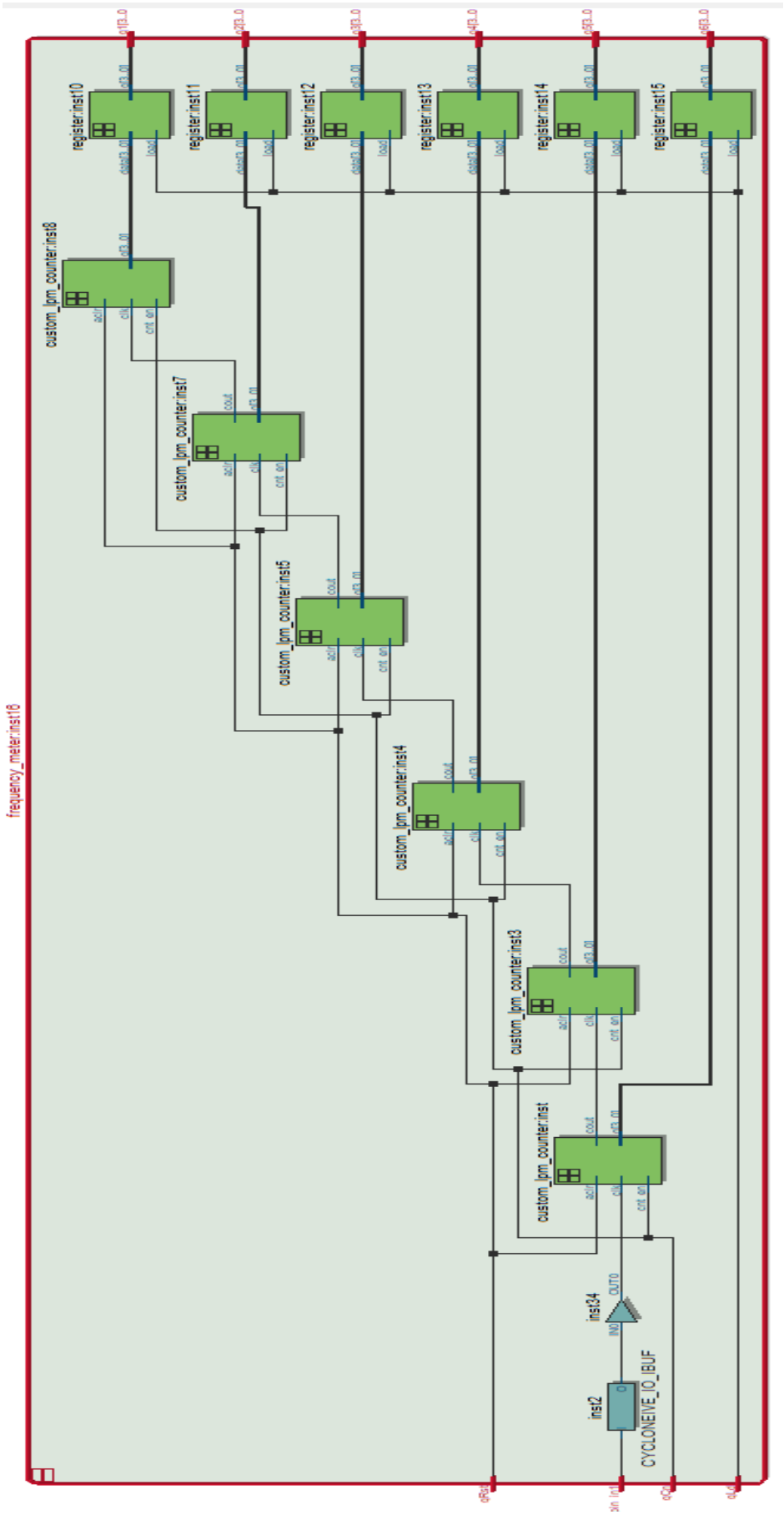
```
module block_form (
input wire clk,
output reg qCnt,
output reg qLd,
output reg qRst );

reg [23:0]counter;
localparam F = 50;

localparam      t1 = 500,
                t2 = 200,
                t3 = 99800,
                t4 = 200,
                t5 = 1000,
                t6 = 200;

localparam      n1 = 0,
                n2 = n1 + t1 * F,

                n3 = n2 + t2 * F,
                n4 = n3 + t3 * F,

                n5 = n4 + t4 * F,
                n6 = n5 + t5 * F,

                n7 = n6 + t6 * F;

always @(posedge clk)
begin
        if(counter == n7)
        counter <= 0;
else
        counter <= counter + 1;

case (counter)
        n1: begin qRst <= 1; qCnt <= 0; qLd <= 0; end
        n2: begin qRst <= 0; qCnt <= 0; qLd <= 0; end

        n3: begin qRst <= 0; qCnt <= 1; qLd <= 0; end
        n4: begin qRst <= 0; qCnt <= 0; qLd <= 0; end

        n5: begin qRst <= 0; qCnt <= 0; qLd <= 1; end
        n6: begin qRst <= 0; qCnt <= 0; qLd <= 0; end

        n7: begin qRst <= 0; qCnt <= 0; qLd <= 0; end
        default;
endcase
end
endmodule
```

Додаток Е
(обов’язковий)

БАГАТОКАНАЛЬНА ВИМІРЮВАЛЬНА СИСТЕМА НА FPGA ДЛЯ РАДІОВИМІРЮВАЛЬНИХ ЧАСТОТНИХ СЕНСОРІВ

Лічильник імпульсів

Структурна схема


frequency_meter:inst18
register:inst10
register:inst11
register:inst12
register:inst13
register:inst14
register:inst15
custom_lpm_counter:inst8
custom_lpm_counter:inst7
custom_lpm_counter:inst5
custom_lpm_counter:inst4
custom_lpm_counter:inst3
custom_lpm_counter:inst
inst34
inst2
CYCLONEIVE_IO_IBUF

Додаток Ж
(довідниковий)

БАГАТОКАНАЛЬНА ВИМІРЮВАЛЬНА СИСТЕМА НА FPGA ДЛЯ РАДІОВИМІРЮВАЛЬНИХ ЧАСТОТНИХ СЕНСОРІВ

Опис блоку LPM лічильника на мові Verilog

```
// BUS_WITH - Count of bus lines
// CNT_MAX_VALUE - Max value that can be stored in counter,
// at  next iteration after max value the caunter will reset and set cout to high
module custom_lpm_counter
#(parameter BUS_WITH=2, parameter CNT_MAX_VALUE=2)
(
input clk,
input cnt_en,
input aclr,
output [BUS_WITH-1:0]q,
output cout
);

`define CLEAR_Q_BUS 0

reg [BUS_WITH-1:0]q_reg = `CLEAR_Q_BUS;
reg cout_reg = 1'b0;

assign cout = cout_reg;
assign q[BUS_WITH-1:0] = q_reg[BUS_WITH-1:0];


always @(posedge clk, posedge aclr) begin
	// When aclr is set counter is blocked and reset
	if (aclr) begin
		// Clear register and carry out pin information
		q_reg <= `CLEAR_Q_BUS;
		cout_reg <= 1'b0;
	end
	else if (cnt_en) begin
		if (q_reg == CNT_MAX_VALUE) begin
			// Clear register
			q_reg <= `CLEAR_Q_BUS;
			// Set carry out pin
			cout_reg <= 1'b1;
		end
		else begin
			// Clear carry out pin
			cout_reg <= 1'b0;
			// Change counter value
			q_reg <= q_reg + 1;
		end
	end

end

endmodule
```

Додаток К
(довідниковий)

БАГАТОКАНАЛЬНА ВИМІРЮВАЛЬНА СИСТЕМА НА FPGA ДЛЯ РАДІОВИМІРЮВАЛЬНИХ ЧАСТОТНИХ СЕНСОРІВ

Опис блоку обробника даних з лічильників на мові Verilog

```
module txtgen12(
        input wire led1,
        input wire qLd,
        input wire [3:0]f6,input wire [3:0]f5,input wire [3:0]f4,input wire [3:0]f3,input wire [3:0]f2,
        input wire [3:0]f1,input wire [3:0]ff6,input wire [3:0]ff5,wire [3:0]ff4, wire [3:0]ff3,
        input wire [3:0]ff2,input wire [3:0]ff1,input wire [3:0]fq6,input wire [3:0]fq5, wire [3:0]fq4,
        input wire [3:0]fq3,input wire [3:0]fq2,input wire [3:0]fq1,input wire [3:0]fw6,
        input wire [3:0]fw5,input wire [3:0]fw4,input wire [3:0]fw3,input wire [3:0]fw2,
input wire [3:0]fw1,input wire [3:0]fe6,input wire [3:0]fe5,input wire [3:0]fe4,
input wire [3:0]fe3,input wire [3:0]fe2,input wire [3:0]fe1,input wire [3:0]fr6,
        input wire [3:0]fr5,input wire [3:0]fr4,input wire [3:0]fr3,input wire [3:0]fr2,
input wire [3:0]fr1,input wire [3:0]fl6,input wire [3:0]fl5,input wire [3:0]fl4,input wire [3:0]fl3,
        input wire [3:0]fl2,input wire [3:0]fl1,input wire [3:0]fx6,   input wire [3:0]fx5,
input wire [3:0]fx4,input wire [3:0]fx3,input wire [3:0]fx2,input wire [3:0]fx1,
        input wire [3:0]fy6,input wire [3:0]fy5,input wire [3:0]fy4,input wire [3:0]fy3,
        input wire [3:0]fy2,input wire [3:0]fy1,input wire [3:0]fd6,input wire [3:0]fd5,
        input wire [3:0]fd4,input wire [3:0]fd3,input wire [3:0]fd2,input wire [3:0]fd1,
        input wire [3:0]fg6,input wire [3:0]fg5,input wire [3:0]fg4,input wire [3:0]fg3,
        input wire [3:0]fg2,input wire [3:0]fg1,input wire [3:0]fh6,input wire [3:0]fh5,
        input wire [3:0]fh4,input wire [3:0]fh3,input wire [3:0]fh2,input wire [3:0]fh1,
        output wire [7:0]byte_wr,output wire wr);

reg [96:0]shift;
always @(posedge led1)
        shift<={shift[96:0],qLd};

assign wr = (shift!=0);

function [7:0] ascii;
input [3:0] val;
begin
         if(val<10) ascii = val+8'h30;  //8'h30 mean symbol "0"
         else ascii = val + 8'h41 - 10; //8'h41 mean symbol "A"
end
endfunction


assign byte_wr =
shift[0] ? ascii( f1[3:0] ) : shift[1] ? ascii( f2[3:0] ) : shift[2] ? ascii( f3[3:0] ) :
        shift[3] ? ascii( f4[3:0] ) : shift[4] ? ascii( f5[3:0] ) :shift[5] ? ascii( f6[3:0] ) :
        shift[6] ? 8'h30: shift[7] ? 8'h20: shift[8] ? ascii( ff1[3:0] ) : shift[9] ? ascii( ff2[3:0] ) :
        shift[10] ? ascii( ff3[3:0] ) : shift[11] ? ascii( ff4[3:0] ) : shift[12] ? ascii( ff5[3:0] ) :
        shift[13] ? ascii( ff6[3:0] ) : shift[14] ? 8'h30: shift[15] ? 8'h20:
        shift[16] ? ascii( fq1[3:0] ) : shift[17] ? ascii( fq2[3:0] ) : shift[18] ? ascii( fq3[3:0] ) :
        shift[19] ? ascii( fq4[3:0] ) : shift[20] ? ascii( fq5[3:0] ) : [21] ? ascii( fq6[3:0] ) :
        shift[22] ? 8'h30: shift[23] ? 8'h20: shift[24] ? ascii( fw1[3:0] ) :shift[25] ? ascii( fw2[3:0] ) :
        shift[26] ? ascii( fw3[3:0] ) : shift[27] ? ascii( fw4[3:0] ) : shift[28] ? ascii( fw5[3:0] ) :
        shift[29] ? ascii( fw6[3:0] ) : shift[30] ? 8'h30: shift[31] ? 8'h20: shift[32] ? ascii( fe1[3:0] ) :
        shift[33] ? ascii( fe2[3:0] ) :[34] ? ascii( fe3[3:0] ) : shift[35] ? ascii( fe4[3:0] ) :
        shift[36] ? ascii( fe5[3:0] ) : shift[37] ? ascii( fe6[3:0] ) : shift[38] ? 8'h30:
        shift[39] ? 8'h20: shift[40] ? ascii( fr1[3:0] ) : [41] ? ascii( fr2[3:0] ) :
        shift[42] ? ascii( fr3[3:0] ) : shift[43] ? ascii( fr4[3:0] ) : shift[44] ? ascii( fr5[3:0] ) :
```

```
        shift[45] ? ascii( fr6[3:0] ) : shift[46] ? 8'h30: shift[47] ? 8'h20: shift[48] ? ascii( fl1[3:0] ) :
        shift[49] ? ascii( fl2[3:0] ) : shift[50] ? ascii( fl3[3:0] ) : [51] ? ascii( fl4[3:0] ) :
        shift[52] ? ascii( fl5[3:0] ) : shift[53] ? ascii( fl6[3:0] ) : shift[54] ? 8'h30:
        shift[55] ? 8'h20: shift[56] ? ascii( fx1[3:0] ) : shift[57] ? ascii( fx2[3:0] ) :
        shift[58] ? ascii( fx3[3:0] ) : shift[59] ? ascii( fx4[3:0] ) : shift[60] ? ascii( fx5[3:0] ) :
        shift[61] ? ascii( fx6[3:0] ) : shift[62] ? 8'h30: shift[63] ? 8'h20: shift[64] ? ascii( fy1[3:0] ) :
        shift[65] ? ascii( fy2[3:0] ) : shift[66] ? ascii( fy3[3:0] ) : shift[67] ? ascii( fy4[3:0] ) :
        shift[68] ? ascii( fy5[3:0] ) : shift[69] ? ascii( fy6[3:0] ) : shift[70] ? 8'h30:
        shift[71] ? 8'h20: shift[72] ? ascii( fd1[3:0] ) : shift[73] ? ascii( fd2[3:0] ) :
        shift[74] ? ascii( fd3[3:0] ) : shift[75] ? ascii( fd4[3:0] ) : shift[76] ? ascii( fd5[3:0] ) :
        shift[77] ? ascii( fd6[3:0] ) : shift[78] ? 8'h30: shift[79] ? 8'h20: shift[80] ? ascii( fg1[3:0] ) :
        shift[81] ? ascii( fg2[3:0] ) : shift[82] ? ascii( fg3[3:0] ) : shift[83] ? ascii( fg4[3:0] ) :
        shift[84] ? ascii( fg5[3:0] ) : shift[85] ? ascii( fg6[3:0] ) : shift[86] ? 8'h30:
        shift[87] ? 8'h20: shift[88] ? ascii( fh1[3:0] ) : shift[89] ? ascii( fh2[3:0] ) :
        shift[90] ? ascii( fh3[3:0] ) : shift[91] ? ascii( fh4[3:0] ) : [92] ? ascii( fh5[3:0] ) :
        shift[93] ? ascii( fh6[3:0] ) : shift[94] ? 8'h30: shift[95] ? 8'h0D : 8'h0A;
endmodule
```

Додаток Л
(довідниковий)

БАГАТОКАНАЛЬНА ВИМІРЮВАЛЬНА СИСТЕМА НА FPGA ДЛЯ РАДІОВИМІРЮВАЛЬНИХ ЧАСТОТНИХ СЕНСОРІВ

Опис блоку UART передавача на мові Verilog

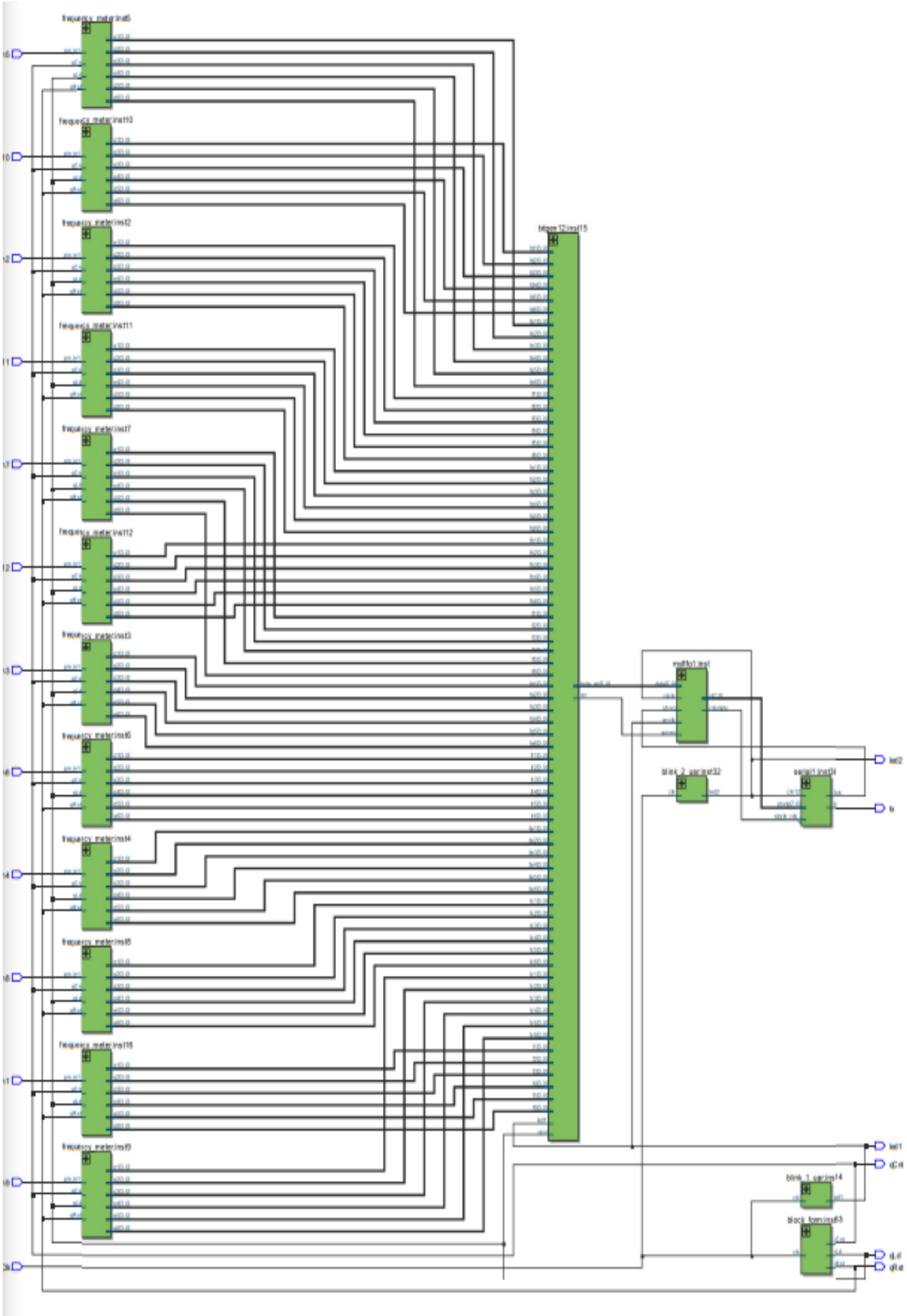
```
module serial1(
	input wire clk12,
	input wire [7:0]sbyte,
	input wire sbyte_rdy,
	output wire tx,
	output wire ack
	);

reg [9:0]sreg;
assign tx = sreg[0];
reg [3:0]cnt = 0;
wire busy; assign busy = (cnt<10);
assign ack = sbyte_rdy & ~busy;

always @(posedge clk12)
begin
	if(sbyte_rdy & ~busy)
		sreg <= { 1'b1, sbyte, 1'b0 }; //load
	else
		sreg <= {1'b1, sreg[9:1] }; //shift

	if(sbyte_rdy & ~busy)
		cnt <= 0;
	else
	if(busy)
		cnt <= cnt + 1'b1;

end

endmodule
```

Додаток М
(обов’язковий)

БАГАТОКАНАЛЬНА ВИМІРЮВАЛЬНА СИСТЕМА НА FPGA ДЛЯ РАДІОВИМІРЮВАЛЬНИХ ЧАСТОТНИХ СЕНСОРІВ

Багатоканальна вимірювальна система зчитування значень із сенсорів з частотним виходом

Структурна схема

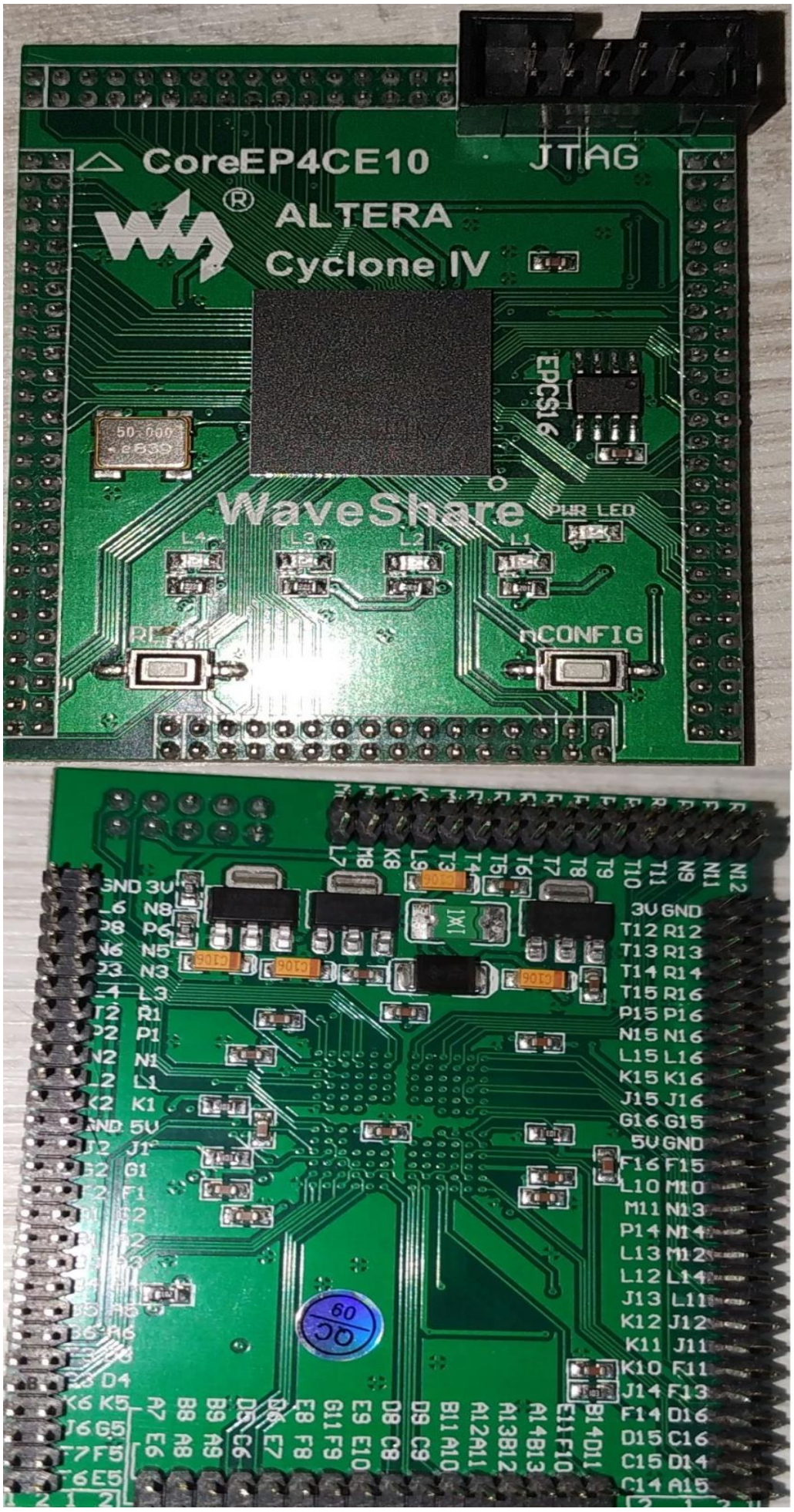


Додаток Н
(обов’язковий)

БАГАТОКАНАЛЬНА ВИМІРЮВАЛЬНА СИСТЕМА НА FPGA ДЛЯ РАДІОВИМІРЮВАЛЬНИХ ЧАСТОТНИХ СЕНСОРІВ

Відладочна плата CoreEP4CE10

Плакат

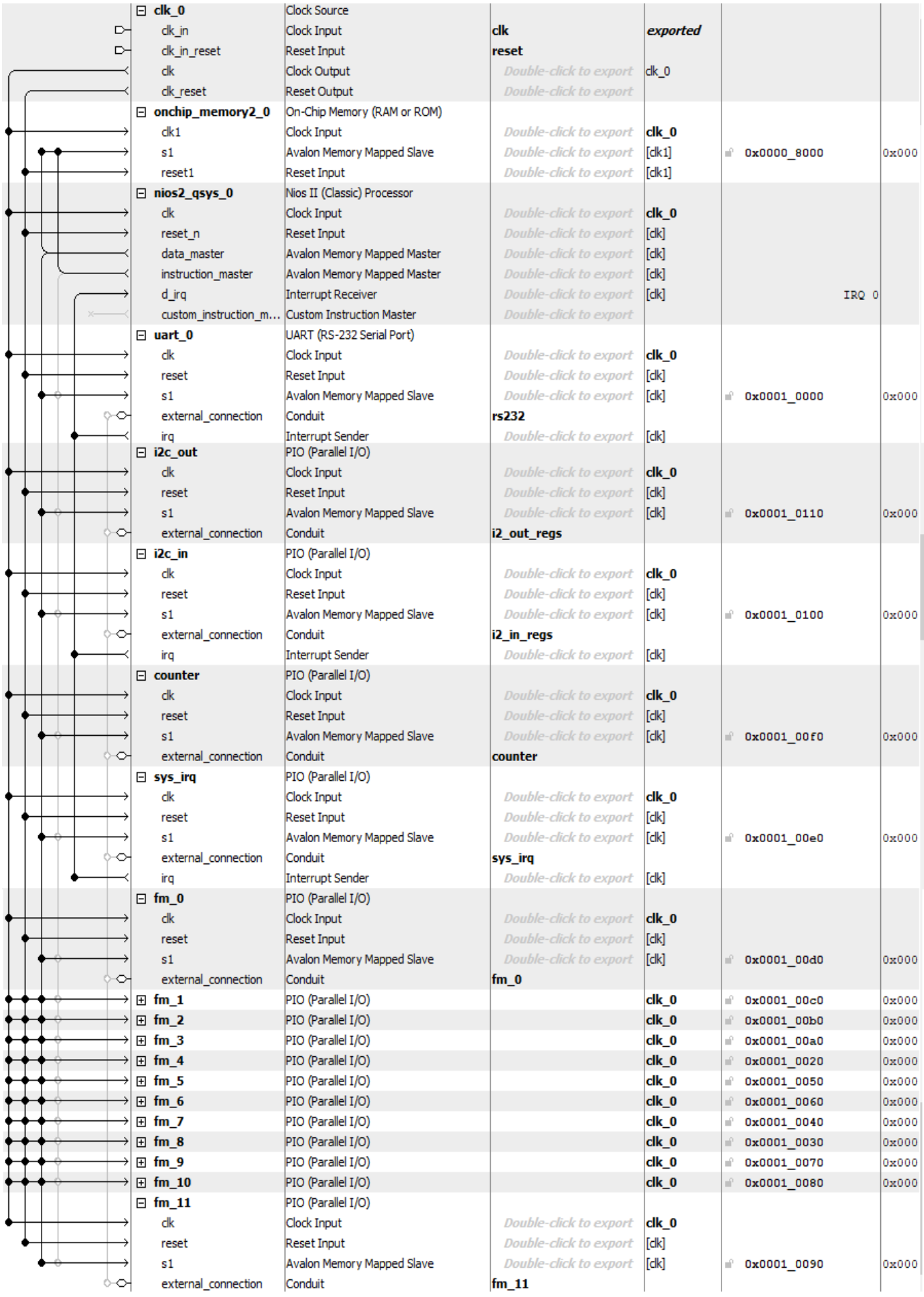

CoreEP4CE10
JTAG
ALTERA
Cyclone IV
EPCS16
WaveShare
PWR LED
nCONFIG

Додаток О
(обов’язковий)

БАГАТОКАНАЛЬНА ВИМІРЮВАЛЬНА СИСТЕМА НА FPGA ДЛЯ РАДІОВИМІРЮВАЛЬНИХ ЧАСТОТНИХ СЕНСОРІВ

Міжблочне з’єднання мікропроцесорної систем

Структурна схема



clk_0	Clock Source					
clk_in	Clock Input	**clk**	*exported*			
clk_in_reset	Reset Input	**reset**				
clk	Clock Output	*Double-click to export*	clk_0			
clk_reset	Reset Output	*Double-click to export*				
onchip_memory2_0	On-Chip Memory (RAM or ROM)					
clk1	Clock Input	*Double-click to export*	**clk_0**			
s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk1]	0x0000_8000		0x000
reset1	Reset Input	*Double-click to export*	[clk1]			
nios2_qsys_0	Nios II (Classic) Processor					
clk	Clock Input	*Double-click to export*	**clk_0**			
reset_n	Reset Input	*Double-click to export*	[clk]			
data_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]			
instruction_master	Avalon Memory Mapped Master	*Double-click to export*	[clk]			
d_irq	Interrupt Receiver	*Double-click to export*	[clk]		IRQ 0	
custom_instruction_m...	Custom Instruction Master	*Double-click to export*				
uart_0	UART (RS-232 Serial Port)					
clk	Clock Input	*Double-click to export*	**clk_0**			
reset	Reset Input	*Double-click to export*	[clk]			
s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0001_0000		0x000
external_connection	Conduit	**rs232**				
irq	Interrupt Sender	*Double-click to export*	[clk]			
i2c_out	PIO (Parallel I/O)					
clk	Clock Input	*Double-click to export*	**clk_0**			
reset	Reset Input	*Double-click to export*	[clk]			
s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0001_0110		0x000
external_connection	Conduit	**i2_out_regs**				
i2c_in	PIO (Parallel I/O)					
clk	Clock Input	*Double-click to export*	**clk_0**			
reset	Reset Input	*Double-click to export*	[clk]			
s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0001_0100		0x000
external_connection	Conduit	**i2_in_regs**				
irq	Interrupt Sender	*Double-click to export*	[clk]			
counter	PIO (Parallel I/O)					
clk	Clock Input	*Double-click to export*	**clk_0**			
reset	Reset Input	*Double-click to export*	[clk]			
s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0001_00f0		0x000
external_connection	Conduit	**counter**				
sys_irq	PIO (Parallel I/O)					
clk	Clock Input	*Double-click to export*	**clk_0**			
reset	Reset Input	*Double-click to export*	[clk]			
s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0001_00e0		0x000
external_connection	Conduit	**sys_irq**				
irq	Interrupt Sender	*Double-click to export*	[clk]			
fm_0	PIO (Parallel I/O)					
clk	Clock Input	*Double-click to export*	**clk_0**			
reset	Reset Input	*Double-click to export*	[clk]			
s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0001_00d0		0x000
external_connection	Conduit	**fm_0**				
fm_1	PIO (Parallel I/O)		**clk_0**	0x0001_00c0		0x000
fm_2	PIO (Parallel I/O)		**clk_0**	0x0001_00b0		0x000
fm_3	PIO (Parallel I/O)		**clk_0**	0x0001_00a0		0x000
fm_4	PIO (Parallel I/O)		**clk_0**	0x0001_0020		0x000
fm_5	PIO (Parallel I/O)		**clk_0**	0x0001_0050		0x000
fm_6	PIO (Parallel I/O)		**clk_0**	0x0001_0060		0x000
fm_7	PIO (Parallel I/O)		**clk_0**	0x0001_0040		0x000
fm_8	PIO (Parallel I/O)		**clk_0**	0x0001_0030		0x000
fm_9	PIO (Parallel I/O)		**clk_0**	0x0001_0070		0x000
fm_10	PIO (Parallel I/O)		**clk_0**	0x0001_0080		0x000
fm_11	PIO (Parallel I/O)					
clk	Clock Input	*Double-click to export*	**clk_0**			
reset	Reset Input	*Double-click to export*	[clk]			
s1	Avalon Memory Mapped Slave	*Double-click to export*	[clk]	0x0001_0090		0x000
external_connection	Conduit	**fm_11**				

Додаток П
(довідниковий)

БАГАТОКАНАЛЬНА ВИМІРЮВАЛЬНА СИСТЕМА НА FPGA ДЛЯ РАДІОВИМІРЮВАЛЬНИХ ЧАСТОТНИХ СЕНСОРІВ

Опис блоку I2C передавача/приймача на мові Verilog

```
LIBRARY ieee;
USE ieee.std_logic_1164.all;
USE ieee.std_logic_unsigned.all;
ENTITY i2c_master IS
  GENERIC(input_clk : INTEGER := 50_000_000; bus_clk   : INTEGER := 400_000);
  PORT(clk       : IN     STD_LOGIC; reset_n   : IN     STD_LOGIC; ena       : IN     STD_LOGIC;
addr      : IN     STD_LOGIC_VECTOR(6 DOWNTO 0); rw        : IN     STD_LOGIC; data_wr   : IN
STD_LOGIC_VECTOR(7 DOWNTO 0); busy        : OUT    STD_LOGIC; data_rd    : OUT
STD_LOGIC_VECTOR(7 DOWNTO 0); ack_error : BUFFER STD_LOGIC; sda         : INOUT
STD_LOGIC; scl       : INOUT  STD_LOGIC);
END i2c_master;
ARCHITECTURE logic OF i2c_master IS
  CONSTANT divider :  INTEGER := (input_clk/bus_clk)/4;
  TYPE machine IS(ready, start, command, slv_ack1, wr, rd, slv_ack2, mstr_ack, stop);
  SIGNAL state          : machine; SIGNAL data_clk       : STD_LOGIC; SIGNAL data_clk_prev :
STD_LOGIC; SIGNAL scl_clk        : STD_LOGIC; SIGNAL scl_ena        : STD_LOGIC := '0';
SIGNAL sda_int        : STD_LOGIC := '1'; sda_ena_n      : STD_LOGIC; SIGNAL addr_rw        :
STD_LOGIC_VECTOR(7 DOWNTO 0); SIGNAL data_tx            : STD_LOGIC_VECTOR(7
DOWNTO 0);  data_rx         : STD_LOGIC_VECTOR(7 DOWNTO 0); SIGNAL bit_cnt        :
INTEGER RANGE 0 TO 7 := 7; SIGNAL stretch      : STD_LOGIC := '0';
BEGIN
  --generate the timing for the bus clock (scl_clk) and the data clock (data_clk)
  PROCESS(clk, reset_n)
    VARIABLE count  :  INTEGER RANGE 0 TO divider*4;  --timing for clock generation
  BEGIN
    IF(reset_n = '0') THEN                --reset asserted
      stretch <= '0';
      count := 0;
    ELSIF(clk'EVENT AND clk = '1') THEN
      data_clk_prev <= data_clk;          --store previous value of data clock
      IF(count = divider*4-1) THEN        --end of timing cycle
        count := 0;                       --reset timer
      ELSIF(stretch = '0') THEN           --clock stretching from slave not detected
        count := count + 1;               --continue clock generation timing
      END IF;
      CASE count IS
        WHEN 0 TO divider-1 =>            --first 1/4 cycle of clocking
          scl_clk <= '0';
          data_clk <= '0';
        WHEN divider TO divider*2-1 =>    --second 1/4 cycle of clocking
          scl_clk <= '0';
          data_clk <= '1';
        WHEN divider*2 TO divider*3-1 =>  --third 1/4 cycle of clocking
          scl_clk <= '1';                 --release scl
          IF(scl = '0') THEN              --detect if slave is stretching clock
            stretch <= '1';
          ELSE
            stretch <= '0';
          END IF;
          data_clk <= '1';
        WHEN OTHERS =>                    --last 1/4 cycle of clocking
          scl_clk <= '1';
```

```
      data_clk <= '0';
    END CASE;
  END IF;
END PROCESS;
--state machine and writing to sda during scl low (data_clk rising edge)
PROCESS(clk, reset_n)
BEGIN
  IF(reset_n = '0') THEN               --reset asserted
    state <= ready;                    --return to initial state
    busy <= '1';                       --indicate not available
    scl_ena <= '0';                    --sets scl high impedance
    sda_int <= '1';                    --sets sda high impedance
    ack_error <= '0';                  --clear acknowledge error flag
    bit_cnt <= 7;                      --restarts data bit counter
    data_rd <= "00000000";             --clear data read port
  ELSIF(clk'EVENT AND clk = '1') THEN
    IF(data_clk = '1' AND data_clk_prev = '0') THEN  --data clock rising edge
      CASE state IS
        WHEN ready =>                  --idle state
          IF(ena = '1') THEN           --transaction requested
            busy <= '1';               --flag busy
            addr_rw <= addr & rw;      --collect requested slave address and command
            data_tx <= data_wr;        --collect requested data to write
            state <= start;            --go to start bit
          ELSE                         --remain idle
            busy <= '0';               --unflag busy
            state <= ready;            --remain idle
          END IF;
        WHEN start =>                  --start bit of transaction
          busy <= '1';                 --resume busy if continuous mode
          sda_int <= addr_rw(bit_cnt);     --set first address bit to bus
          state <= command;            --go to command
        WHEN command =>                --address and command byte of transaction
          IF(bit_cnt = 0) THEN         --command transmit finished
            sda_int <= '1';            --release sda for slave acknowledge
            bit_cnt <= 7;              --reset bit counter for "byte" states
            state <= slv_ack1;         --go to slave acknowledge (command)
          ELSE                         --next clock cycle of command state
            bit_cnt <= bit_cnt - 1;    --keep track of transaction bits
            sda_int <= addr_rw(bit_cnt-1); --write address/command bit to bus
            state <= command;          --continue with command
          END IF;
        WHEN slv_ack1 =>               --slave acknowledge bit (command)
          IF(addr_rw(0) = '0') THEN    --write command
            sda_int <= data_tx(bit_cnt);   --write first bit of data
            state <= wr;               --go to write byte
          ELSE                         --read command
            sda_int <= '1';            --release sda from incoming data
            state <= rd;               --go to read byte
          END IF;
        WHEN wr =>                     --write byte of transaction
          busy <= '1';                 --resume busy if continuous mode
```

```
          IF(bit_cnt = 0) THEN           --write byte transmit finished
            sda_int <= '1';              --release sda for slave acknowledge
            bit_cnt <= 7;                --reset bit counter for "byte" states
            state <= slv_ack2;           --go to slave acknowledge (write)
          ELSE                           --next clock cycle of write state
            bit_cnt <= bit_cnt - 1;      --keep track of transaction bits
            sda_int <= data_tx(bit_cnt-1); --write next bit to bus
            state <= wr;                 --continue writing
          END IF;
        WHEN rd =>                       --read byte of transaction
          busy <= '1';                   --resume busy if continuous mode
          IF(bit_cnt = 0) THEN           --read byte receive finished
            IF(ena = '1' AND addr_rw = addr & rw) THEN  --continuing with another read at same address
              sda_int <= '0';            --acknowledge the byte has been received
            ELSE                         --stopping or continuing with a write
              sda_int <= '1';            --send a no-acknowledge (before stop or repeated start)
            END IF;
            bit_cnt <= 7;                --reset bit counter for "byte" states
            data_rd <= data_rx;          --output received data
            state <= mstr_ack;           --go to master acknowledge
          ELSE                           --next clock cycle of read state
            bit_cnt <= bit_cnt - 1;      --keep track of transaction bits
            state <= rd;                 --continue reading
          END IF;
        WHEN slv_ack2 =>                 --slave acknowledge bit (write)
          IF(ena = '1') THEN             --continue transaction
            busy <= '0';                 --continue is accepted
            addr_rw <= addr & rw;        --collect requested slave address and command
            data_tx <= data_wr;          --collect requested data to write
            IF(addr_rw = addr & rw) THEN   --continue transaction with another write
              sda_int <= data_wr(bit_cnt); --write first bit of data
              state <= wr;               --go to write byte
            ELSE                         --continue transaction with a read or new slave
              state <= start;            --go to repeated start
            END IF;
          ELSE                           --complete transaction
            state <= stop;               --go to stop bit
          END IF;
        WHEN mstr_ack =>                 --master acknowledge bit after a read
          IF(ena = '1') THEN             --continue transaction
            busy <= '0';                 --continue is accepted and data received is available on bus
            addr_rw <= addr & rw;        --collect requested slave address and command
            data_tx <= data_wr;          --collect requested data to write
            IF(addr_rw = addr & rw) THEN   --continue transaction with another read
              sda_int <= '1';            --release sda from incoming data
              state <= rd;               --go to read byte
            ELSE                         --continue transaction with a write or new slave
              state <= start;            --repeated start
            END IF;
          ELSE                           --complete transaction
            state <= stop;               --go to stop bit
```

```
          END IF;
        WHEN stop =>                    --stop bit of transaction
          busy <= '0';                  --unflag busy
          state <= ready;               --go to idle state
      END CASE;
    ELSIF(data_clk = '0' AND data_clk_prev = '1') THEN  --data clock falling edge
      CASE state IS
        WHEN start =>
          IF(scl_ena = '0') THEN                --starting new transaction
            scl_ena <= '1';                     --enable scl output
            ack_error <= '0';                   --reset acknowledge error output
          END IF;
        WHEN slv_ack1 =>                        --receiving slave acknowledge (command)
          IF(sda /= '0' OR ack_error = '1') THEN  --no-acknowledge or previous no-acknowledge
            ack_error <= '1';                   --set error output if no-acknowledge
          END IF;
        WHEN rd =>                              --receiving slave data
          data_rx(bit_cnt) <= sda;              --receive current slave data bit
        WHEN slv_ack2 =>                        --receiving slave acknowledge (write)
          IF(sda /= '0' OR ack_error = '1') THEN  --no-acknowledge or previous no-acknowledge
            ack_error <= '1';                   --set error output if no-acknowledge
          END IF;
        WHEN stop =>
          scl_ena <= '0';                       --disable scl
        WHEN OTHERS =>
          NULL;
      END CASE;
    END IF;
  END IF;
END PROCESS;
--set sda output
WITH state SELECT
  sda_ena_n <= data_clk_prev WHEN start,     --generate start condition
               NOT data_clk_prev WHEN stop,  --generate stop condition
               sda_int WHEN OTHERS;          --set to internal sda signal
--set scl and sda outputs
scl <= '0' WHEN (scl_ena = '1' AND scl_clk = '0') ELSE 'Z';
sda <= '0' WHEN sda_ena_n = '0' ELSE 'Z';
END logic;
```

Додаток Р
(довідниковий)

БАГАТОКАНАЛЬНА ВИМІРЮВАЛЬНА СИСТЕМА НА FPGA ДЛЯ РАДІОВИМІРЮВАЛЬНИХ ЧАСТОТНИХ СЕНСОРІВ

Опис оновленого лічильника імпульсів на мові Verilog

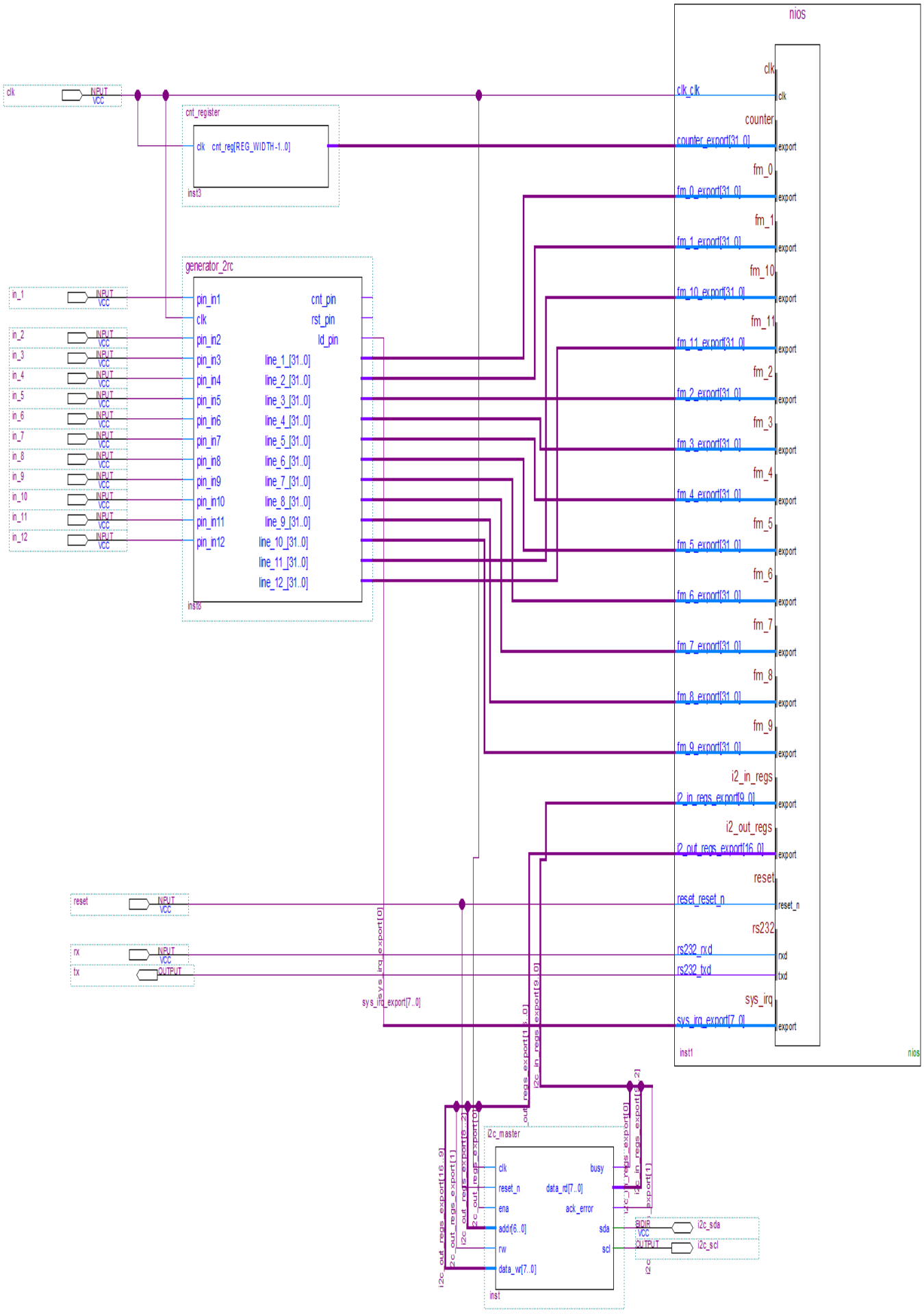
```
// BUS_WIDTH - Count of bus lines
module clk_counter
#(parameter BUS_WIDTH=32)
(
 input clk,
 input cnt_en,
 input aclr,
 output [BUS_WIDTH-1:0]bus
);

`define CLEAR_Q_BUS 0

reg [BUS_WIDTH-1:0]q_reg = `CLEAR_Q_BUS;
assign bus[BUS_WIDTH-1:0] = q_reg[BUS_WIDTH-1:0];


always @(posedge clk, posedge aclr) begin
	// When aclr is set counter is blocked and reset
	if (aclr) begin
		// Clear register
		q_reg <= `CLEAR_Q_BUS;
	end
	else if (cnt_en) begin
		// Change counter value
		q_reg <= q_reg + 1;
	end

end
endmodule
```

Додаток С
(обов’язковий)

БАГАТОКАНАЛЬНА ВИМІРЮВАЛЬНА СИСТЕМА НА FPGA ДЛЯ РАДІОВИМІРЮВАЛЬНИХ ЧАСТОТНИХ СЕНСОРІВ

Узагальнена мікропроцесорна система

Структурна схема


nios
clk
clk_clk
counter
counter_export[31..0]
fm_0
fm_0_export[31..0]
fm_1
fm_1_export[31..0]
fm_10
fm_10_export[31..0]
fm_11
fm_11_export[31..0]
fm_2
fm_2_export[31..0]
fm_3
fm_3_export[31..0]
fm_4
fm_4_export[31..0]
fm_5
fm_5_export[31..0]
fm_6
fm_6_export[31..0]
fm_7
fm_7_export[31..0]
fm_8
fm_8_export[31..0]
fm_9
fm_9_export[31..0]
i2_in_regs
i2_in_regs_export[9..0]
i2_out_regs
i2_out_regs_export[16..0]
reset
reset_reset_n
rs232
rs232_rxd
rs232_txd
sys_irq
sys_irq_export[7..0]
inst1
clk
INPUT
VCC
cnt_register
clk cnt_reg[REG_WIDTH-1..0]
inst3
generator_2rc
pin_in1
clk
pin_in2
pin_in3
pin_in4
pin_in5
pin_in6
pin_in7
pin_in8
pin_in9
pin_in10
pin_in11
pin_in12
cnt_pin
rst_pin
ld_pin
line_1_[31..0]
line_2_[31..0]
line_3_[31..0]
line_4_[31..0]
line_5_[31..0]
line_6_[31..0]
line_7_[31..0]
line_8_[31..0]
line_9_[31..0]
line_10_[31..0]
line_11_[31..0]
line_12_[31..0]
inst8
in_1
in_2
in_3
in_4
in_5
in_6
in_7
in_8
in_9
in_10
in_11
in_12
reset
rx
tx
OUTPUT
sys_irq_export[7..0]
i2c_out_regs_export[16..9]
i2c_out_regs_export[1]
i2c_out_regs_export[8..2]
i2c_out_regs_export[0]
i2c_in_regs_export[9..0]
i2c_in_regs_export[0]
i2c_in_regs_export[1]
i2c_in_regs_export[9..2]
i2c_master
clk
reset_n
ena
addr[6..0]
rw
data_w[7..0]
busy
data_rd[7..0]
ack_error
sda
scl
inst
BIDIR
i2c_sda
i2c_scl
i2c

Додаток Т
(довідниковий)

БАГАТОКАНАЛЬНА ВИМІРЮВАЛЬНА СИСТЕМА НА FPGA ДЛЯ РАДІОВИМІРЮВАЛЬНИХ ЧАСТОТНИХ СЕНСОРІВ

Реалізація нижнього рівня I2C драйвера

```
#include "i2c/hal_driver/i2c_hal.h"
#include "general_defs.h"

#define ENA_OFFSET                  (0x00)
#define ENA_MASK                    (0x01)

#define RW_OFFSET                   (0x01)
#define RW_MASK                     (0x01)

#define SLAVE_ADDR_OFFSET           (0x02)
#define SLAVE_ADDR_MASK             (0x7F)

#define W_DATA_OFFSET               (0x09)
#define W_DATA_MASK                 (0xFF)

#define BUSY_OFFSET                 (0x00)
#define BUSY_MASK                   (0x01)

#define ACK_ERR_OFFSET              (0x01)
#define ACK_ERR_MASK                (0x01)

#define R_DATA_OFFSET               (0x02)
#define R_DATA_MASK                 (0xFF)

static alt_8 i2c_hal_reg_write(alt_u8 reg_mask, alt_u8 reg_offset, alt_u8 value);
static alt_8 i2c_hal_reg_read(alt_u8 reg_mask, alt_u8 reg_offset, alt_u8 *value);
staticvoidi2c_hal_irq_handler(void* context);

typedefstruct {
        i2c_hall_irq_func_t func;
        void *context;
} i2c_data_t;

i2c_data_t _i2c_data = {
                NULL,
                NULL
};
/*------------Internal functions------------*/
alt_8 i2c_hal_reg_write(alt_u8 reg_mask, alt_u8 reg_offset, alt_u8 value) {
        // Read last reg value and clear bits for set new value
        alt_u32 reg_value = IORD_ALTERA_AVALON_PIO_DATA(I2C_OUT_BASE) &
(~((alt_u32)reg_mask << reg_offset));
        // Update value for reg
        reg_value |= ((alt_u32)value & reg_mask) << reg_offset;
        // Write new value into register
        IOWR_ALTERA_AVALON_PIO_DATA(I2C_OUT_BASE, reg_value);
        return RES_OK;
}

alt_8 i2c_hal_reg_read(alt_u8 reg_mask, alt_u8 reg_offset, alt_u8 *value) {
        // check if pointer is valid
        if (!value)
```

```
                return RES_ERR;
        // cut part of common register value
        *value = (IORD_ALTERA_AVALON_PIO_DATA(I2C_IN_BASE) & ((alt_u32)reg_mask
<< reg_offset)) >> reg_offset;

        return RES_OK;
}
/*------------Functions for user------------*/
inline alt_8 i2c_hal_set_ena(alt_u8 state){
        return i2c_hal_reg_write(ENA_MASK, ENA_OFFSET, state);
}

inline alt_8 i2c_hal_get_ena(alt_u8 *state) {
        return i2c_hal_reg_read(ENA_MASK, ENA_OFFSET, state);
}

inline alt_8 i2c_hal_set_rw(alt_u8 rw){
        return i2c_hal_reg_write(RW_MASK, RW_OFFSET, rw);
}

inline alt_8 i2c_hal_set_slave_addr(alt_u8 addr){
        return i2c_hal_reg_write(SLAVE_ADDR_MASK, SLAVE_ADDR_OFFSET, addr);
}

inline alt_8 i2c_hal_set_write_data(alt_u8 w_data){
        return i2c_hal_reg_write(W_DATA_MASK, W_DATA_OFFSET, w_data);
}

inline alt_8 i2c_hal_get_busy(alt_u8 *busy){
        return i2c_hal_reg_read(BUSY_MASK, BUSY_OFFSET, busy);
}

inline alt_8 i2c_hal_get_ack_error(alt_u8 *ack_err){
        return i2c_hal_reg_read(ACK_ERR_MASK, ACK_ERR_OFFSET, ack_err);
}

inline alt_8 i2c_hal_get_read_data(alt_u8 *r_data){
        return i2c_hal_reg_read(R_DATA_MASK, R_DATA_OFFSET, r_data);
}
```

Додаток У
(довідниковий)

БАГАТОКАНАЛЬНА ВИМІРЮВАЛЬНА СИСТЕМА НА FPGA ДЛЯ РАДІОВИМІРЮВАЛЬНИХ ЧАСТОТНИХ СЕНСОРІВ

Реалізація верхнього рівня I2C драйвера

```
#include "i2c/hal_driver/i2c_hal.h"
#include "i2c/i2c.h"
#include "system/sys_irq.h"
#include "system/bsp_printf.h"

void_i2c_irq_handler(void *context, alt_u8 state) {
	_busy_state = state;
}

void_i2c_wait_for_busy(alt_u8 state) {
	alt_u8 curr_busy = FALSE;
	do {
		i2c_hal_get_busy(&curr_busy);
	} while (curr_busy != state);
}

alt_8 i2c_init(void) {
	i2c_hal_init(_i2c_irq_handler, NULL);
	return RES_OK;
}

alt_u8 i2c_is_busy(void) {
	alt_u8 state = I2C_ENABLE_TRANSMISSION, busy = TRUE;
	return i2c_hal_get_ena(&state) || i2c_hal_get_busy(&busy) || state || busy;
}

alt_u8 i2c_is_err(void) {
	alt_u8 err = TRUE;
	return i2c_hal_get_ack_error(&err) || err;
}

alt_8 i2c_write_byte(alt_u8 dev_addr, alt_u8 reg_addr, alt_u8 w_data) {
	return i2c_write_bytes(dev_addr, reg_addr, &w_data, 0x01);
}

alt_8 i2c_read_byte(alt_u8 dev_addr, alt_u8 reg_addr, alt_u8 *r_data) {
	// Reading data from register
	return i2c_read_bytes(dev_addr, reg_addr, r_data, 0x01);
}

alt_8 i2c_read_bytes(alt_u8 dev_addr, alt_u8 reg_addr, alt_u8 *r_data_arr, alt_u8 len) {
	alt_u8 i = 0x00;
	alt_u32 irq_state = 0x00;
	if (i2c_is_busy() || !r_data_arr || !len)
		return RES_ERR;
	// Disable all interrupts
	irq_state = sys_irq_critical_section_begin();
	// Set up I2C registers for read
	i2c_hal_set_slave_addr(dev_addr);
	i2c_hal_set_rw(I2C_WRITE);
	i2c_hal_set_write_data(reg_addr);
	i2c_hal_set_ena(I2C_ENABLE_TRANSMISSION);
```

```
	// Wait for start transmission
	_i2c_wait_for_busy(TRUE);
	i2c_hal_set_rw(I2C_READ);
	if (i2c_is_err()) {
		i2c_hal_set_ena(I2C_DISABLE_TRANSMISSION);
		sys_irq_critical_section_end(irq_state);
		sys_printf("\r\n%s() -> Err at start transmission, dev: %d, reg: %d\r\n",
				__func__, dev_addr, reg_addr);
		return RES_ERR;
	}
	// Wait read stage
	_i2c_wait_for_busy(FALSE);
	for (i = 0x00; i < len; i++) {
		_i2c_wait_for_busy(TRUE);
		// Disable transmission before end of current
		// transaction for avoid starting new transaction
		if (i == len - 1)
			i2c_hal_set_ena(I2C_DISABLE_TRANSMISSION);

		_i2c_wait_for_busy(FALSE);
		i2c_hal_get_read_data(r_data_arr + i);
		if (i2c_is_err()) {
			i2c_hal_set_ena(I2C_DISABLE_TRANSMISSION);
			sys_irq_critical_section_end(irq_state);
			sys_printf("\r\n%s() -> Err during read register, dev: %d, reg: %d\r\n",
					__func__, dev_addr, reg_addr + i);
			return RES_ERR;
		}
	}
	// Enable interrupts
	sys_irq_critical_section_end(irq_state);
	return RES_OK;
}

alt_8 i2c_write_bytes(alt_u8 dev_addr, alt_u8 reg_addr, alt_u8 *w_data_arr, alt_u8 len) {
	alt_u8 i = 0x00;
	alt_u32 irq_state = 0x00;
	if (i2c_is_busy() || !w_data_arr || !len)
		return RES_ERR;
	// Disable all interrupts
	irq_state = sys_irq_critical_section_begin();
	i2c_hal_set_slave_addr(dev_addr);
	i2c_hal_set_rw(I2C_WRITE);
	i2c_hal_set_write_data(reg_addr);
	i2c_hal_set_ena(I2C_ENABLE_TRANSMISSION);
	// Wait stage of set register value
	_i2c_wait_for_busy(TRUE);
	for (i = 0x00; i < len; i++) {
		i2c_hal_set_write_data(w_data_arr[i]);
		if (i2c_is_err()) {
			i2c_hal_set_ena(I2C_DISABLE_TRANSMISSION);
			sys_irq_critical_section_end(irq_state);
```

```
                        sys_printf("\r\n%s() -> Err at set register value stage, dev: %d, reg: %d\r\n",
                                        __func__, dev_addr, reg_addr + i);
                        return RES_ERR;
                }
                // Wait for write register value stage
                _i2c_wait_for_busy(FALSE);
                _i2c_wait_for_busy(TRUE);
        }
        i2c_hal_set_ena(I2C_DISABLE_TRANSMISSION);
        // End of transmission
        _i2c_wait_for_busy(FALSE);
        if (i2c_is_err()) {
                i2c_hal_set_ena(I2C_DISABLE_TRANSMISSION);
                sys_irq_critical_section_end(irq_state);
                sys_printf("\r\n%s() -> Err at end of read process, dev: %d, reg: %d\r\n",
                                __func__, dev_addr, reg_addr + i);
                return RES_ERR;
        }

        // Enable interrupts
        sys_irq_critical_section_end(irq_state);
        return RES_OK;
}
```

Додаток Ф
(довідниковий)

БАГАТОКАНАЛЬНА ВИМІРЮВАЛЬНА СИСТЕМА НА FPGA ДЛЯ РАДІОВИМІРЮВАЛЬНИХ ЧАСТОТНИХ СЕНСОРІВ

Реалізація драйвера частотомірів

```
#include "frequency_meter/fm.h"
#include "system/sys_irq.h"
#include "general_defs.h"

#define IRQ_MASK                    (0x01)

alt_u32 channel_id_to_addr(alt_u8 channel_id) {
        alt_u32 addr = (alt_u32)NULL;
        switch(channel_id) {
        caseCHANNEL_0: addr = FM_0_BASE; break;
        caseCHANNEL_1: addr = FM_1_BASE; break;
        caseCHANNEL_2: addr = FM_2_BASE; break;
        caseCHANNEL_3: addr = FM_3_BASE; break;
        caseCHANNEL_4: addr = FM_4_BASE; break;
        caseCHANNEL_5: addr = FM_5_BASE; break;
        caseCHANNEL_6: addr = FM_6_BASE; break;
        caseCHANNEL_7: addr = FM_7_BASE; break;
        caseCHANNEL_8: addr = FM_8_BASE; break;
        caseCHANNEL_9: addr = FM_9_BASE; break;
        caseCHANNEL_10: addr = FM_10_BASE; break;
        caseCHANNEL_11: addr = FM_11_BASE; break;
        default: addr = (alt_u32)NULL; break;
        }
        return addr;
}

voidfm_irq_handler(void* context) {
        sys_irq_clear_interrupt_register(IRQ_MASK);
        alt_u8 i = 0x00;
        for (i = 0x00; i <CHANNEL_MAX; i++) {
                channels_data[i] = IORD_ALTERA_AVALON_PIO_DATA(channel_id_to_addr(i));
        }
}

alt_8 fm_init() {
        sys_irq_enable(IRQ_MASK);
        // Register ISR
        alt_ic_isr_register(SYS_IRQ_IRQ_INTERRUPT_CONTROLLER_ID, SYS_IRQ_IRQ,
fm_irq_handler, NULL, 0x00);
        return RES_OK;
}

alt_32 fm_get_freq(alt_u8 channel_id) {
        if (channel_id >= CHANNEL_MAX)
                return RES_ERR;

        return channels_data[channel_id];
}
```

Додаток Х
(довідниковий)

БАГАТОКАНАЛЬНА ВИМІРЮВАЛЬНА СИСТЕМА НА FPGA ДЛЯ РАДІОВИМІРЮВАЛЬНИХ ЧАСТОТНИХ СЕНСОРІВ

Код реалізації програми «SerialMonitor»

```
from typing import Optional, List
from PyQt5 import QtWidgets, QtCore
from PyQt5.QtCore import Qt
from PyQt5.QtWidgets import QWidget, QListWidgetItem, QMessageBox, QLabel, QGridLayout
import pyqtgraph as pg
import sys
from random import randrange
from serial_port import SerialMonitor
from typing import Optional, List, Callable
import serial
from PyQt5.QtCore import QIODevice
from PyQt5.QtSerialPort import QSerialPort

DATA_VALUE_CNT = 19
DATA_MAX_SAMPLES_AT_PLOT = 100
DATA_FM_POS = 0
DATA_FM_CNT = 12
DATA_SEN_POS = 12
DATA_SEN_CNT = 7

class SerialMonitor:
def __init__(self, context, port: str, read_callback: Callable[[List[int]], None]):
        self.__s_data: str = ""
self.read_callback = read_callback
        self.__serial: Optional[QSerialPort] = QSerialPort(context)
        self.__serial.setPortName(port)
        self.__serial.setBaudRate(115200)
        self.__serial.readyRead.connect(self.__on_serial_read)
if self.__serial.open(QIODevice.ReadOnly) is False:
raise Exception(f"Incorrect serial port: {port}")

def __del__(self):
if hasattr(self, "__serial") and self.__serial.isOpen():
            self.__serial.close()

def __on_serial_read(self):
# Read character
self.__s_data += str(self.__serial.readAll(), "utf-8")
if '\n' not in self.__s_data:
return
# Parse read line
try:
            last_index = self.__s_data.find("\n")
            data_list = self.__s_data[:last_index+1].split("\r\n")
            self.__s_data = self.__s_data[last_index+1:] if len(self.__s_data) > last_index+1 else ""
            for sub_data in data_list:
if len(sub_data) == 0:
continue
data = sub_data.split('\t')
                data = list(map(int, data))
# Send processed line to user
self.read_callback(data)
```

```
except Exception as ex:
        print(f"__on_serial_read() -> Unable to process data: {data_list}, ex: {ex}")

   @classmethod
def available_ports(cls) -> List[str]:
      ports = ['COM%s' % (i + 1) for i in range(256)]
      result = []
for port in ports:
try:
            s = serial.Serial(port)
            s.close()
            result.append(port)
except (OSError, serial.SerialException):
pass
      return result

class MainWindow(QWidget):

def __init__(self):
      super().__init__()

      self.serial: Optional[SerialMonitor] = None
self.data_line = []
      self.data = {}
for i in range(DATA_VALUE_CNT):
         self.data[i] = []
# Set up list widget
port_lable = QLabel('Port')
      port_lable.setAlignment(Qt.AlignCenter)
      self.listWidget = QtWidgets.QListWidget()
      self.listWidget.resize(50, 20)
      self.listWidget.itemClicked.connect(self.__on_list_clicked)
# Set up plot widget for frequency
fm_lable = QLabel("Frequency")
      fm_lable.setAlignment(Qt.AlignCenter)
      self.fm_graph_widget = pg.PlotWidget()
      self.fm_graph_widget.setBackground('w')
for i in range(DATA_FM_CNT):
         self.data_line.append(self.fm_graph_widget.plot([], [], pen=pg.mkPen(color=(255,
randrange(256), randrange(256)))))
# Set up plot widget for I2C sensors
sen_lable = QLabel("I2C sensors")
      sen_lable.setAlignment(Qt.AlignCenter)
      self.sen_graph_widget = pg.PlotWidget()
      self.sen_graph_widget.setBackground('w')
for i in range(DATA_SEN_CNT):
         self.data_line.append(self.sen_graph_widget.plot([], [],
pen=pg.mkPen(color=(randrange(256), 255, randrange(256)))))
# Update layout
grid = QGridLayout()
      grid.setSpacing(10)
      grid.addWidget(port_lable, 1, 0)
```

```
        grid.addWidget(self.listWidget, 1, 1)
        grid.addWidget(fm_lable, 2, 0)
        grid.addWidget(sen_lable, 2, 1)
        grid.addWidget(self.fm_graph_widget, 3, 0)
        grid.addWidget(self.sen_graph_widget, 3, 1)
        self.setLayout(grid)
        self.setGeometry(400, 100, 600, 500)
        self.setWindowTitle('Serial monitor')
        QtCore.QTimer.singleShot(10, self.__process)

def __on_list_clicked(self, item: QListWidgetItem):
try:
            self.serial = SerialMonitor(self, str(item.data(0)), self.__on_data_read)
except Exception as ex:
            self.__show_msg(QMessageBox.Warning, "Warning", f"__on_list_clicked() -> Warning
message: {ex}")

def __on_data_read(self, data: List[int]):
if len(data) < DATA_VALUE_CNT:
             print(f"Received data with incorrect value count: {data}")
return
self.__update_plot(data)

def __process(self):
        self.__update_port_info()
        QtCore.QTimer.singleShot(200, self.__process)

def __update_port_info(self):
        ports = SerialMonitor.available_ports()
        update = False
        for i in range(self.listWidget.count()):
if self.listWidget.item(i).data(0) not in ports:
                update = True
                break
        if self.listWidget.count() == len(ports) and update is False:
return
self.listWidget.clear()
        self.listWidget.addItems(SerialMonitor.available_ports())

def __update_plot(self,  data: List[int]):
for i in range(DATA_VALUE_CNT):
# Remove first value
if len(self.data[i]) >= DATA_MAX_SAMPLES_AT_PLOT:
                self.data[i] = self.data[i][1:]
# Add new value
self.data[i].append(data[i])
             self.data_line[i].setData(list(range(len(self.data[i]))), self.data[i])

def __show_msg(self, icon: int, title: str, text: str):
        msg = QMessageBox()
        msg.setIcon(icon)
        msg.setWindowTitle(title)
```

```
        msg.setText(text)
        msg.exec_()


app = QtWidgets.QApplication(sys.argv)
w = MainWindow()
w.show()
sys.exit(app.exec_())
```