Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет __Комп'ютерної інженерії та управління_____
(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти ____другий (магістерський)_____
(рівень вищої освіти)

Методи імітаційного моделювання
при автоматизованому проектуванні СІС-фільтрів_
(тема)

Виконав: здобувач 2 року навчання,
групи СКСм-23-2
Корольков І. О._____
(прізвище, ініціали)

Спеціальність *123 _Комп'ютерна інженерія*

Тип програми ____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма_____

_____Комп'ютерна інженерія _____
(повна назва освітньої програми)

Керівник _доц. Філіппенко І.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ _Чумаченко С.В
(підпис) (прізвище, ініціали)

2025 р.

Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерної інженерії та управління_____

Кафедра ____Автоматизації проектування обчислювальної техніки_____

Рівень вищої освіти _____другий (магістерський)_____

Спеціальність _____123 Комп'ютерна інженерія _____
(шифр і назва)

Тип програми _____Освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма _____Спеціалізовані комп'ютерні системи_____
(повна назва

ЗАТВЕРДЖУЮ:

Зав. кафедри ___ _____
(підпис)
«____»_____ 20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві___Королькову Івану Олександровичу_____
(прізвище, ім'я, по батькові)

Тема роботи (проєкту)___Методи імітаційного моделювання при автоматизованому проектуванні CIC фільтрів_____
_____

затверджена наказом по університету від «_08_»__11__2024р. № _1189 Ст_____
2. Термін подання студентом роботи _____20.01.2024_____
3. Вихідні дані до роботи (проєкту)_____
Алгоритми цифрової фільтрації: CIC._____
Мови програмування C/C++_____
Мови опису апаратури VHDL, Verilog_____
САПР Xilinx ISE, Quartus, ModelSim_____
Інструменти моделювання: MATLAB, Simulink_____
_____
Перелік питань, що потрібно опрацювати у роботі _____
Аналіз структури CIC-фільтрів_____
Розробка моделей цифрової фільтрації в MATLAB та Simulink_____
Реалізація алгоритмів децимації та інтерполяції_____
Оптимізація продуктивності фільтрів для платформ FPGA_____
_____

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) _____
_____
_____
_____
_____
_____

6. Консультанти розділів роботи (проєкту)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи (проєкту)	Термін виконання етапів проєкту (роботи)	Примітка
1	Аналіз літератури та постановка задачі	02.09.2024-08.09.2024	
2	Огляд існуючих алгоритмів	09.09. 2024-15.09. 2024	
3	Імітаційне моделювання у MATLAB та Simulink	16.09.2024-10.10.2024	
4	Тестування та верифікація розробок	11.10. 2024-19.11.2024	
5	Оптимізація моделей для FPGA у Vivado	20.11.2024-09.12.2024	
6	Оформлення пояснювальної записки	10.12.2024-30.12.2024	
7	Захист роботи	13.01. 2025-25.01. 2025	

Дата видачі завдання 02.09.2024

Здобувач _____ ___ Корольков І.О.
(підпис)

Керівник роботи (проєкту) __ __ доц. Філіппенкл І.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка містить 66 сторінок, 30 рисунків та 8 джерел за переліком посилань.

ЦИФРОВА ФІЛЬТРАЦІЯ, CIC ФІЛЬТР, ДЕЦИМАЦІЯ, ІНТЕРПОЛЯЦІЯ, ПЛІС, КАСКАДНИЙ ІНТЕГРАЛЬНО-ГРЕБЕНЧАТИЙ ФІЛЬТР, ЧАСТОТНА ХАРАКТЕРИСТИКА,

Метою роботи є вивчення принципів роботи CIC фільтрів, їхнього застосування в цифровій обробці сигналів, моделювання цих фільтрів, особливостей реалізації на програмованих логічних інтегральних схемах (ПЛІС).

Проведений аналіз структур цифрових CIC фільтрів, включаючи каскадну реалізацію інтеграторів та гребінчастих фільтрів для виконання децимації та інтерполяції. Розглянуто переваги застосування CIC фільтрів для зниження частоти дискретизації в системах з високою продуктивністю, таких як бездротовий зв'язок та обробка радіолокаційних сигналів.

ABSTRACT

The explanatory note contains 66 pages, 30 figures and 8 sources according to the list of links.

DIGITAL FILTRATION, CIC FILTER, DECIMATION, INTERPOLATION, PLIS, CASCADE INTEGRAL COMB FILTER, FREQUENCY RESPONSE,

The aim of this work is to study the principles of operation of CIC filters, their application in digital signal processing, modeling of these filters, and the specifics of their implementation on field-programmable gate arrays (FPGAs).

The analysis of CIC filter structures was conducted, including the cascaded implementation of integrators and comb filters for performing decimation and interpolation. The advantages of using CIC filters for reducing the sampling rate in high-performance systems, such as wireless communication and radar signal processing, were examined.

ЗМІСТ

_Toc188785047
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ 7
ВСТУП 8
1 ЦИФРОВА ФІЛЬТРАЦІЯ 10
1.1 Класифікація цифрових фільтрів 10
1.2 FIR-фільтри 13
1.3 IIR-фільтри 14
1.4 Адаптивні фільтри 15
1.5 CIC-фільтри 17
1.6 Рекурсивний фільтр із сумуванням 19
2 АПАРАТНА ПЛАТФОРМА FPGA DE0-NANO 25
2.1 Модель архітектури DE0-Nano 25
2.2 Структура вбудованої платформи FPGA DE0-Nano 26
2.3 Програмована логіка 27
2.4 Інтерфейси вводу/виводу 29
2.5 Переваги FPGA DE0-Nano 31
3 ТЕХНОЛОГІЇ СИМУЛЯЦІЇ ЦИФОВИХ ФІЛЬТРІВ 33
3.1 Симуляція та використання середовиша MATLAB 33
3.2 Моделювання CIC-фільтра в MATLAB 35
4 Реалізація CIC фільтра 56
4.1 Quartus II 56
4.2 Моделювання CIC фільтра 57
ВИСНОВКИ 65
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 66

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ADC (Analog-to-Digital Converter) – аналого-цифровий перетворювач.

CIC (Cascaded Integrator-Comb) – каскадний інтегрально-гребінчастий фільтр.

DAC (Digital-to-Analog Converter) – цифро-аналоговий перетворювач.

DSP (Digital Signal Processing) – цифрова обробка сигналів.

FPGA (Field-Programmable Gate Array) – програмована логічна інтегральна схема.

FIR (Finite Impulse Response) – фільтр з кінцевою імпульсною характеристикою.

IIR (Infinite Impulse Response) – фільтр з нескінченною імпульсною характеристикою.

HLS (High-Level Synthesis) – високорівневий синтез апаратних схем.

MATLAB – середовище для числового моделювання та обчислень.

ModelSim – програмне середовище для моделювання та верифікації
фрових схем.

RTL (Register-Transfer Level) – рівень передачі регістрів, який описує функціональність цифрових схем.

SNR (Signal-to-Noise Ratio) – відношення сигнал/шум.

SoC (System-on-Chip) – система на кристалі.

VHDL (VHSIC Hardware Description Language) – мова опису апаратури.

Verilog – мова опису апаратури для моделювання і синтезу цифрових схем.

Quartus – програмне середовище для проєктування цифрових схем на FPGA (розроблене компанією Intel, раніше Altera)..

ВСТУП

Сучасні технології стрімко розвиваються, забезпечуючи все більшу інтеграцію цифрових систем в різні галузі науки, промисловості та побуту. Особливу увагу привертають системи цифрової обробки сигналів (ЦОС), які лежать в основі телекомунікаційних мереж, медичних приладів, аудіо- та відеосистем, радіолокаційних пристроїв та багатьох інших високотехнологічних рішень. Зі зростанням вимог до продуктивності та точності таких систем виникає потреба у створенні ефективних методів проєктування цифрових фільтрів, що відіграють ключову роль у ЦОС.

Каскадні інтегрально-гребінчасті (CIC) фільтри використовуються для децимації (зменшення частоти дискретизації) або інтерполяції (збільшення частоти дискретизації) сигналів. Їхня конструкція, яка не вимагає операцій множення, знижує обчислювальні витрати, що робить їх оптимальними для застосувань у цифровому зв’язку, SDR (програмно визначеному радіо) та інших високопродуктивних системах.

Програмовані логічні інтегральні схеми (FPGA) є перспективними платформами для реалізації цифрових фільтрів, оскільки забезпечують високу швидкість обробки сигналів і гнучкість налаштувань. Завдяки можливостям паралельної обробки FPGA дозволяють реалізовувати складні алгоритми, включаючи CIC фільтри, що відкриває нові можливості для високопродуктивних систем обробки сигналів.

Метою роботи є дослідження, моделювання та вибір оптимальних параметрів CIC фільтрів із використанням сучасних інструментів проєктування для платформ FPGA. Для досягнення поставленої мети сформульовано такі завдання:

- провести аналіз структур CIC фільтрів та їх характеристик;
- дослідити методи децимації та інтерполяції сигналів із використанням CIC фільтрів;

- розробити математичну модель CIC фільтра та протестувати її у MATLAB;
- реалізувати CIC фільтри на FPGA за допомогою Quartus A;
- оцінити ефективність оптимізованих методів у реальних задачах ЦОС.

Використання MATLAB, Simulink та FPGA-платформ спрощує процес проєктування цифрових фільтрів. Ціллю роботи є синтез CIC фільтра для задач децимації сигналів та аналіз результатів моделювання.

1 ЦИФРОВА ФІЛЬТРАЦІЯ

1.1 Класифікація цифрових фільтрів

Цифрові фільтри є ключовими компонентами в цифровій обробці сигналів, оскільки вони дозволяють виконувати різноманітні функції обробки сигналів, такі як видалення шуму, виділення корисних сигналів та інше. Цифрові фільтри можуть бути класифіковані на кілька типів залежно від їх структури та функцій. Розглянемо основні типи цифрових фільтрів, які використовуються, та особливості їхньої класифікації у цифровій обробці сигналів.

У цифровій обробці сигналів (ЦОС) цифрові фільтри класифікуються за кількома критеріями, включаючи тип імпульсної характеристики, призначення, частотні властивості, методи реалізації та здатність до адаптації. Такий підхід дозволяє систематизувати різноманітні фільтри для конкретних задач, спрощуючи вибір оптимального типу фільтра залежно від вимог до обчислювальної ефективності, надійності та частотних характеристик. Наведемо основні категорії цифрових фільтрів, які широко використовуються в сучасних системах ЦОС.

Класифікація за типом імпульсної характеристики. Цей критерій є одним із основних, оскільки імпульсна характеристика визначає, як фільтр реагує на вхідний сигнал у часі. Бувають наступні типи фільтрів.

Фільтри з кінцевою імпульсною характеристикою (КІХ, FIR) мають скінченний відгук на одиничний імпульс, тобто їхня імпульсна характеристика закінчується через визначену кількість відліків. Така характеристика забезпечує лінійну фазу (за відповідних налаштувань) і повну стабільність фільтра, що особливо важливо для запобігання фазовим спотворенням. КІХ-фільтри також гнучкі в проектуванні та часто використовуються у задачах, де критична точність передачі сигналу. До цього класу належать звичайні КІХ-фільтри та гребінчасті фільтри.

Фільтри з нескінченною імпульсною характеристикою (БІХ, IIR) мають нескінченний відгук на вхідний імпульс, оскільки їхня структура передбачає залежність виходу не тільки від вхідних, але й від попередніх вихідних значень. Це дозволяє досягати значної вибірковості на частотах з меншою кількістю коефіцієнтів, проте фільтри можуть не мати лінійної фази, що призводить до можливих фазових спотворень. Прикладами БІХ-фільтрів є фільтри Баттерворта, Чебишева та еліптичні фільтри, які знаходять застосування в задачах, що вимагають високої вибірковості.

Класифікація за призначенням. Цифрові фільтри також можна класифікувати за їхнім основним призначенням, тобто за здатністю пригнічувати чи посилювати певні частотні компоненти сигналу:

1) фільтри низьких частот пропускають низькочастотні компоненти та пригнічують високочастотні, що дозволяє зменшувати шум та виділяти сигнали низької частоти;

2) фільтри високих частот пропускають високочастотні компоненти та пригнічують низькочастотні, що є корисним для виділення високочастотних сигналів;

3) смугові фільтри пропускають сигнали лише у певному частотному діапазоні, пригнічуючи інші частоти, що робить їх незамінними у комунікаційних системах для виділення окремих смуг частот;

4) режекторні (notch) фільтри пригнічують вузьку смугу частот, залишаючи інші частоти недоторканими, що дозволяє вибірково усувати шум або небажані частоти.

Класифікація за частотною характеристикою. Частотна характеристика визначає, як фільтр впливає на фазу та амплітуду різних частотних компонентів сигналу:

1) лінійно-фазові фільтри мають лінійну фазу, що запобігає фазовим спотворенням, проте для їх реалізації може знадобитися більше обчислювальних ресурсів. Вони найчастіше реалізуються як КІХ-фільтри;

2) мінімально-фазові фільтри характеризуються мінімальною затримкою, однак можуть мати нелінійну фазу, що іноді спричиняє фазові спотворення. Це корисно, коли затримка має вирішальне значення;

3) максимально-фазові фільтри мають максимальну фазову затримку, що інколи є доцільним для специфічних завдань.

Та класифікація за методами реалізації. Метод реалізації визначає, як побудований фільтр:

1) прямий (каскадний) метод передбачає послідовне з'єднання кількох каскадів для досягнення бажаного рівня придушення частот;

2) рекурсивні структури передбачають зворотний зв'язок, при якому вихідний сигнал залежить не лише від поточних вхідних значень, але й від попередніх виходів, що дозволяє створювати компактні фільтри з необхідними характеристиками; ця структура характерна для БІХ-фільтрів;

3) полісинтетичні структури (polyphase) застосовуються для децимації та інтерполяції, підвищуючи ефективність обчислень і зменшуючи кількість операцій;

4) схеми із затримкою та накопиченням використовуються у CIC-фільтрах, які не потребують множення, що знижує вимоги до обчислювальних ресурсів, роблячи їх ідеальними для задач з високою пропускною здатністю.

Класифікація за адаптивністю. Адаптивність фільтра визначає, чи можуть його коефіцієнти змінюватись відповідно до зміни сигналу. Можна виділити два типи: постійні (стаціонарні) фільтри, які мають фіксовані коефіцієнти і призначені для роботи з незмінним сигналом та адаптивні фільтри, котрі можуть змінювати свої коефіцієнти відповідно до характеристик вхідного сигналу, що робить їх незамінними для задач, де характеристики сигналу змінюються з часом, як-от придушення шуму або компенсування ехо.

Такий поділ надає широкий вибір методів та підходів для розробки та застосування фільтрів у різних областях, від обробки аудіосигналів до телекомунікацій та радіолокації.

1.2 FIR-фільтри

FIR-фільтри (фільтри з кінцевою імпульсною характеристикою) мають скінченну імпульсну характеристику, що означає, що їхня імпульсна характеристика закінчується після скінченної кількості вибірок. Основні переваги FIR-фільтрів включають лінійну фазову характеристику, стійкість та гнучкість у проектуванні. FIR-фільтри широко використовуються, наприклад, у цифрових аудіоплеєрах для видалення небажаних частотних компонентів, у системах обробки зображень для згладжування та виділення країв. Основна структура FIR-фільтра описується рівнянням 2.1:

$$y[n] = \sum_{i=0}^{k} x(n-i)h(i), \tag{2.1}$$

де y[n] – вихідний сигнал у момент часу n, x(n-i) – вхідний сигнал із затримкою на і вибірок, h(i) – коефіцієнти фільтра, k – порядок фільтра.[5]

Блок схема Fir фільтра наведена на рисунку 2.1, з якої можна побачити сутність побудови цього фільтра.

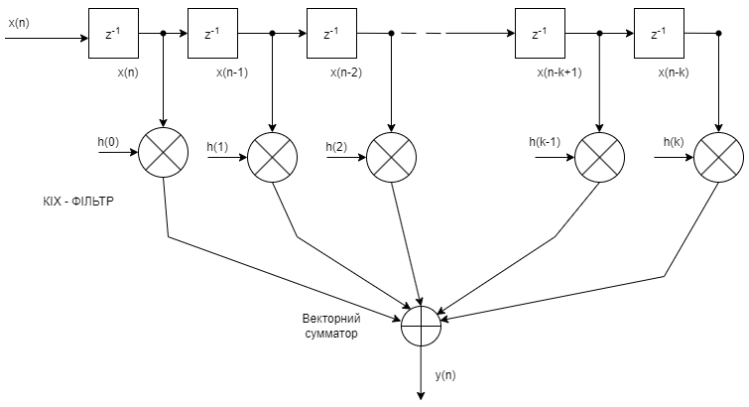

Рисунок 2.1 – Загальна блок схема КІХ фільтра

Формула 2.1 означає, що вихідний сигнал є зваженою сумою попередніх значень вхідного сигналу, де ваги визначаються коефіцієнтами фільтра. Основні переваги FIR-фільтрів включають лінійну фазову характеристику, що забезпечує відсутність фазових спотворень, та стійкість, оскільки немає зворотного зв'язку, який може викликати нестійкість.

1.3 IIR-фільтри

IIR-фільтри (фільтри з безкінечною імпульсною характеристикою) мають безкінечну імпульсну характеристику завдяки наявності зворотного зв'язку. Вони часто вимагають меншої кількості коефіцієнтів порівняно з FIR-фільтрами для досягнення подібної амплітудно-частотної характеристики. IIR-фільтри широко використовуються в таких галузях, як цифровий зв'язок, радіо обробка та системи управління. Ці фільтри використовують зворотний зв'язок, що дозволяє їм мати безкінечну імпульсну характеристику. Це робить їх більш ефективними у порівнянні з FIR-фільтрами для досягнення бажаних амплітудно-частотних характеристик при меншій кількості коефіцієнтів. У загальному фільтри IIR можуть бути описані узагальненими рівняннями 2.2.

З (2.2) можна побачити, що вихідний сигнал залежить не тільки від попередніх значень вхідного сигналу, але й від попередніх значень вихідного сигналу, що забезпечує зворотний зв'язок. Основні переваги IIR-фільтрів включають високу ефективність і компактність реалізації та можливість досягнення бажаних характеристик при меншій кількості коефіцієнтів.

$$y[n] = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k), \qquad (2.2)$$

де y[n] – вихідний сигнал фільтра на момент часу n, x[n] – вхідний сигнал фільтра на момент часу n, a_k – коефіцієнти зворотного зв'язку

(відповідають полюсам системи), b_k – коефіцієнти прямого зв'язку (відповідають нулям системи), N – порядок фільтра (кількість попередніх вихідних значень, які використовуються), M – порядок чисельника (кількість попередніх вхідних значень, які використовуються). [5]

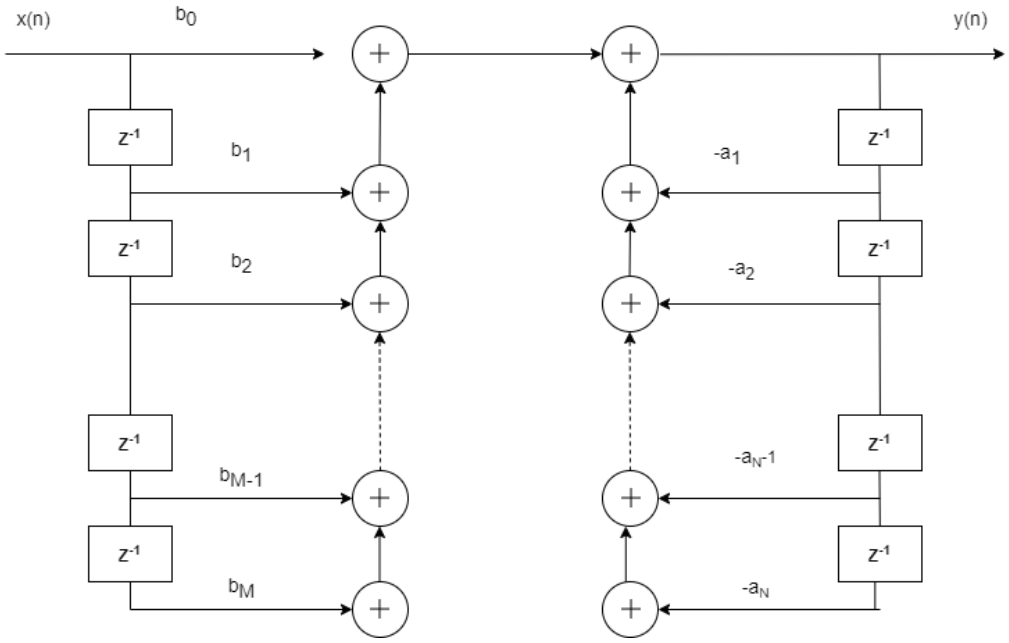

Рисунок 2.2 – Загальна блок схема БІХ фільтра

1.4 Адаптивні фільтри

Адаптивні фільтри можуть змінювати свої характеристики в реальному часі для оптимізації продуктивності в умовах змінного середовища. Вони знаходять застосування в системах активного шум поглинання, адаптивних антенах та інших передових технологіях. Ці фільтри використовують алгоритми, які автоматично налаштовують параметри фільтра на основі вхідних даних. Найбільш поширенні алгоритми адаптивних фільтрів включають алгоритм найменших середньоквадратичних (LMS**)** та рекурсивний алгоритм найменших квадратів

Сучасні цифрові фільтри є надзвичайно важливими інструментами в різних сферах науки і техніки. Вони забезпечують високоякісну обробку сигналів і відкривають нові можливості для інновацій. Завдяки постійним

розробкам і вдосконаленням, цифрові фільтри продовжують грати ключову роль у розвитку сучасних технологій, забезпечуючи більш ефективну і точну обробку сигналів.

Алгоритм найменших квадратів (LMS) є одним з найпоширеніших методів адаптивної фільтрації. Він базується на обчисленні вихідного сигналу фільтра, оцінці похибки (різниця між бажаним і фактичним вихідним сигналом) та оновленні коефіцієнтів фільтра пропорційно до похибки. Формула для оновлення коефіцієнтів виглядає так:

$$w(n+1) = w(n) + \mu \cdot e(n) \cdot x(n), \tag{2.4}$$

де w w(n) – вектор коефіцієнтів фільтра на n-му кроці, μ – швидкість навчання, e(n) – похибка, x(n) – вектор вхідних даних.

Алгоритм рекурсивних найменших квадратів (RLS) забезпечує швидше зближення порівняно з LMS, проте він є складнішим з обчислювальної точки зору. RLS мінімізує суму квадратів похибок з урахуванням всіх попередніх вхідних даних. Основні кроки алгоритму включають обчислення похибки прогнозу та оновлення вектора коефіцієнтів фільтра за допомогою зважених минулих даних. Формула для оновлення коефіцієнтів виглядає так:

$$w(n+1) = w(n-1) + k(n) \cdot e(n), \tag{2.5}$$

де k(n) – вектор посилення, який визначає внесок кожного вхідного зразка в оновлення. Адаптивні фільтри широко використовуються у системах активного шум поглинання, адаптивних антенах та інших передових технологіях, де необхідно динамічно реагувати на зміни вхідного сигналу.

1.5 CIC-фільтри

CIC-фільтри (каскадні інтегратор-гребінчасті фільтри) стали незамінними в тих випадках, коли потрібно знизити частоту дискретизації сигналу, і водночас обійтися без складних операцій множення. Це значно скорочує обчислювальні витрати та економить енергію. Наприклад, у системах із сигма-дельта модуляцією CIC-фільтри дозволяють точно відтворити сигнал, навіть після суттєвого зниження частоти дискретизації. Це робить їх ключовими компонентами в таких галузях, як цифровий зв'язок і радіолокація. Завдяки простій конструкції, що поєднує інтегратори та гребінчасті фільтри, вони знижують витрати на апаратну реалізацію.

Унікальність CIC-фільтрів у тому, що їм не потрібні складні коефіцієнти чи операції множення, лише додавання і віднімання. Це не тільки спрощує архітектуру, а й робить фільтри чудовим вибором для зниження шуму та зміни частоти, що особливо важливо в багатоканальних комунікаційних системах. CIC-фільтри часто використовуються для інтерполяції сигналів у цифро-аналогових перетворювачах і обробки аудіосигналів у пристроях з обмеженими ресурсами.

Ці фільтри також ідеально підходять для VLSI-систем завдяки своїй регулярній структурі, що легко адаптується до FPGA та інших апаратних платформ. CIC-фільтри є оптимальним рішенням для цифрових приймально-передавальних пристроїв і SDR (програмно визначеного радіо), де вони забезпечують узгодження частоти між різними етапами обробки сигналу.

Завдяки своїй економічності та ефективності, CIC-фільтри стали важливим компонентом сучасної цифрової обробки сигналів, що дозволяє значно скоротити апаратні витрати при високій продуктивності.

CIC-фільтри використовуються для зниження частоти дискретизації (decimation) або підвищення частоти дискретизації (interpolation) у цифрових системах. Вони складаються з каскаду інтеграторів і гребінчатих фільтрів. Імпульсна характеристика CIC-фільтра:

$$H(z) = \frac{\left(1-z^{-RM}\right)^N}{(1-z^{-1})^N}, \qquad (2.6)$$

де z – вхідний сигнал, N - кількість каскадів, M - диференціальна затримка і R - коефіцієнт децимації. Тепер розглянемо блок схемну структуру цього фільтру.

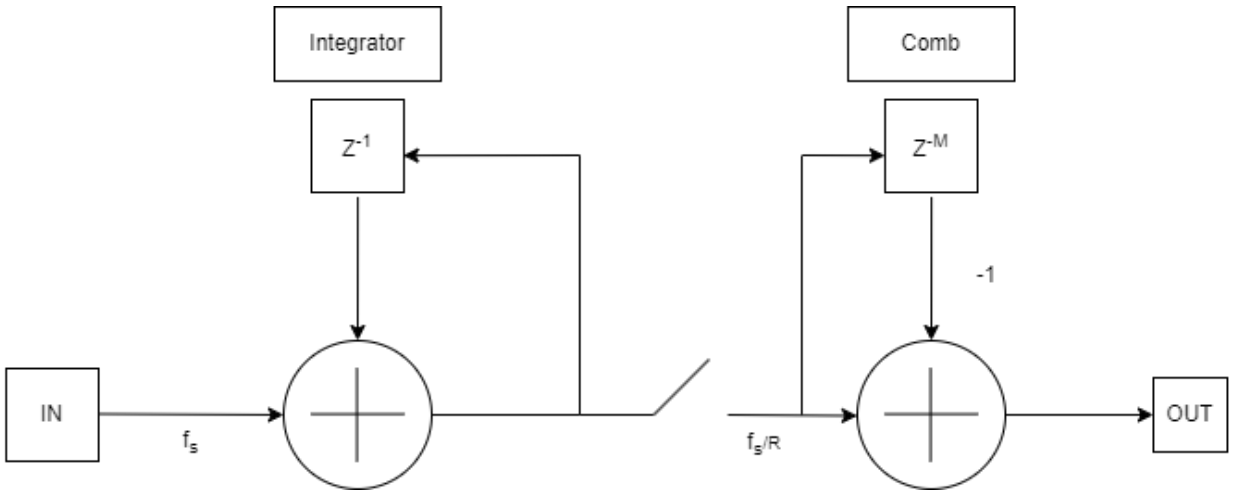

Рисунок 2.4 – Структурна схема гребінчастого фільтра та БІХ фільтра інтегратора [1]

Фільтри CIC чудово підходять для антиаліайзінгової фільтрації перед децимацією (зниженням частоти дискретизації), як показано на рисунку 1(a); і для антиміджингової фільтрації для інтерпольованих сигналів (збільшення частоти дискретизації), як показано на рисунку 1(b). Обидва застосування пов'язані з фільтрацією даних, таких як апаратне модуляція і демодуляція в сучасних бездротових системах і дельта-сигма перетворювачах A/D та D/A.

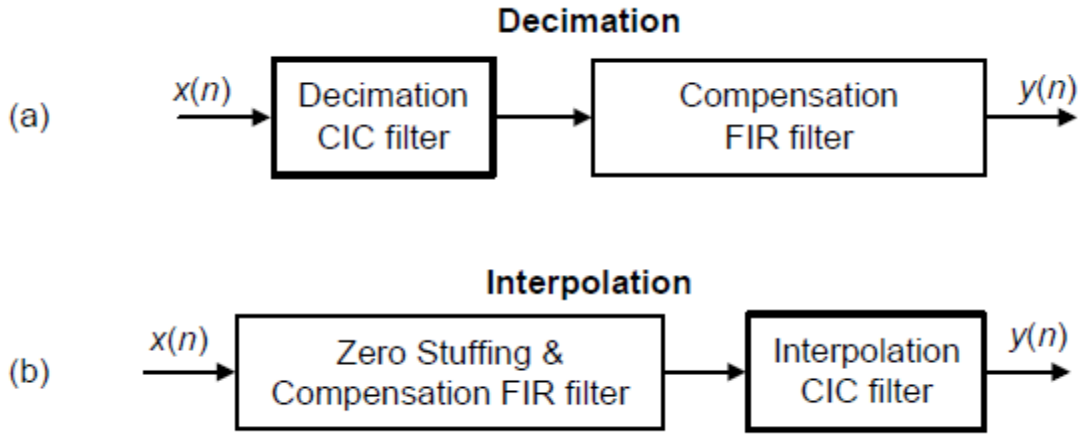

Рисунок 2.5 – Застосування фільтрів CIC: (a) для децимації; (b) для інтерполяції.

Через те, що їх частотні характеристики схожі на sin(x)/x, CIC фільтри зазвичай слідують або передують лінійно-фазним низькочастотним фільтрам з кінцевою імпульсною характеристикою (FIR), завдання яких – компенсувати неплоску смугу пропускання фільтра CIC.

1.6 Рекурсивний фільтр із сумуванням

Фільтри CIC походять від концепції рекурсивного фільтра із сумуванням, який сам по собі є ефективною формою нерекурсивного середньоарифметичного фільтра. Розглянемо стандартний процес середньоарифметичного фільтра на D точок на рисунку 2(a). Тут ми бачимо, що для обчислення вихідної послідовності y(n) необхідні D–1 додавань (плюс одне множення на 1/D). [5]

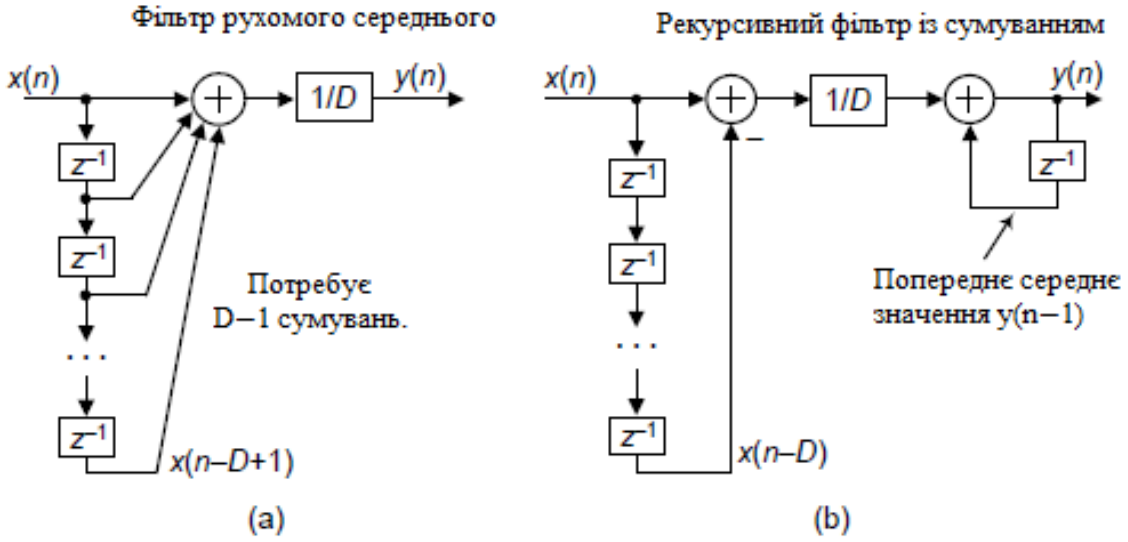

Рисунок 2.6 – Середньоарифметичний фільтр на D точок і рекурсивний фільтр із сумуванням.

Часова область вихідного сигналу середньоарифметичного фільтра виражається як:

$$y(n) = \frac{1}{D[x(n)+x(n-1)+x(n-2)+\cdots+x(n-D+1)]} \tag{2.7}$$

де n - наш часовий індекс. Вираз у z-перетворенні для цього фільтра:

$$Y(z) = \frac{1}{D[X(z)+X(z)z^{-1}+X(z)z^{-2}+\cdots+X(z)z^{-D+1}]} \tag{2.8}$$

а передавальна функція у z- перетворенні є:

$$H_{MA}(z) = \frac{1}{D}\sum_{n=0}^{D-1} z^{-n} \tag{2.9}$$

1.7 Децимація

У попередніх розділах розповідалось про структуру CIC фільтра. Вона складається з двох основних частин які виконують функції децимації та інтерполяції. Розглянемо децимацію більше детально.

Децимація (проріджування) – це процес зниження частоти дискретизації сигнала. Зниження частоти дискретизації досягається шляхом видалення кожного R-го відліку вхідної послідовності, де R – коефіцієнт децімаії, що виражається такою формулою:

$$y[n] = x[nR] \tag{2.10}$$

Проста компресія цифрових послідовностей призводить до накладання періодичних складових спектру прорідженого сигналу через ефект відображення, що погіршує спектральні характеристики вихідного сигналу. Отже, для зниження частоти дискретизації необхідно підключати низькочастотний фільтр (ФНЧ). У такому разі структурна схема фільтра-дециматора отримується шляхом каскадного з'єднання ФНЧ та компресора. Формула, яка описує зв'язок між вхідним та вихідним сигналами, має вигляд:

$$y[n] = \sum_{k=0}^{N-1} x[nR - k] \cdot h[k], \tag{2.11}$$

де h[k] – відліки фільтра з кінцевою імпульсною характеристикою, x[nR-K] – прорідженний вхдіний сигнал. Вихідний сигнал фільтра-дециматора зберігає лише кожний R-й відлік. Таким чином, для КІХ-фільтрів із кінцевими межами суми обчислювальні витрати потенційно можна зменшити в R разів. Це призводить до зменшення швидкості обробки або кількості використаних логічних ресурсів при реалізації на ПЛІС.

1.8 Інтерполяція

Інтерполяція це процес підвищення частоти дискретизації. Операція збільшення частоти дискретизації виконується через введення R-1 нульових відліків у вхідну послідовність і описується формулою

$$y[n] = \begin{cases} x[n/R], n = 0, L, 2L, \dots \\ 0, n \neq 0, L, 2L, \dots \end{cases} \quad (2.12)$$

Вставка нулів еквівалентна стисненню спектра в R разів та його періодичному продовженню. Тому при підвищенні частоти дискретизації також потрібен фільтр низьких частот. Структурна схема фільтра-інтерполятора – це каскадне з'єднання експандера та фільтра низьких частот. Формула для вихідного сигналу перетворюється на:

$$y[n] = \sum_{k=0}^{N-1} x[nR - k] \cdot h[k], \quad (2.13)$$

де h(k) – відліки фільтра за кінцевою імпульсною характеристикою, [[nR-k] – вхідний сигнал із вставленими нулями між відліками Вихідний сигнал фільтра-інтерполятора використовує проріджену в R рази вхідну послідовність. Це знижує обчислювальні витрати та обсяг пам’яті, потрібної для КІХ-фільтрів.

CIC-фільтри формуються шляхом каскадного з’єднання інтегратора та диференціюючого елемента. Така структура отримала назву інтегрально-гребінчастих фільтрів (Cascaded Integrator-Comb). Основними елементами CIC-фільтра є інтегратор, який виконує функцію простого акумулятора, та гребінчастий фільтр, що реалізує віднімання сусідніх відліків сигналу. Завдяки цій простій і ефективній конструкції CIC-фільтри набули широкого застосування в сучасних цифрових системах обробки сигналів.

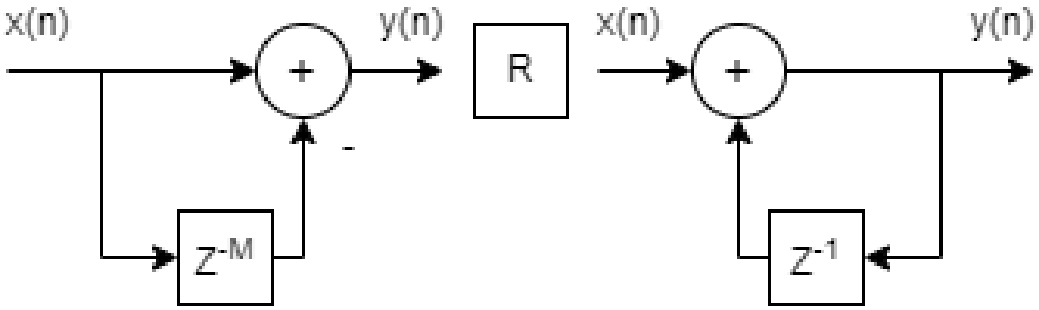

Рисунок 2.7 – Структура CIC фільтра

Між інтегратором і гребінчастим фільтром зазвичай розташовується вузол, що забезпечує підвищення або зниження частоти дискретизації на ціле число разів R. У випадку зниження частоти дискретизації кожен R-й відлік вхідної послідовності зберігається, формуючи проріджену вихідну послідовність. У випадку підвищення частоти дискретизації між відліками вхідної послідовності вставляються нулі. Ці нулі потім згладжуються інтегруючою секцією, що дозволяє отримати послідовність із вищою частотою дискретизації. Формули для опису передавальної та амплітудно-частотної характеристик CIC-фільтра. [3]

$$\|H(f)\| = \left[\frac{\sin(\pi RMf)}{\sin(\pi f)}\right]^N, \tag{2.14}$$

Якщо CIC-фільтр використовується для зниження частоти дискретизації, то він називається дециматором. У такому разі першою ланкою йде інтегратор, потім відбувається зниження частоти дискретизації і, нарешті, йде ланка фільтра, що диференціює.

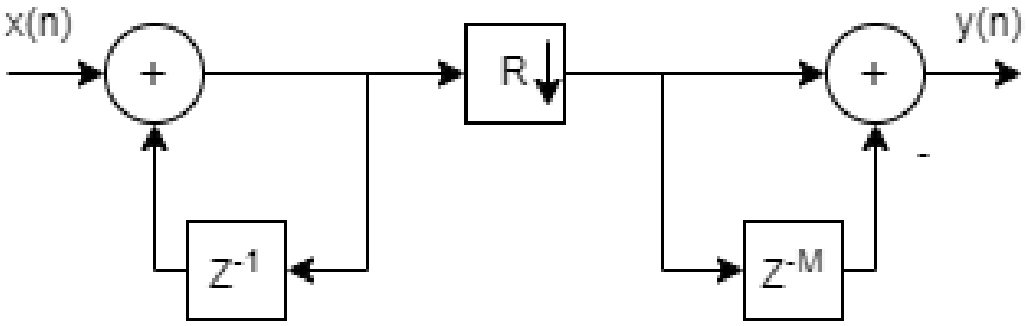

Рисунок 2.8 – Структура CIC фільтра дециматора

Якщо CIC-фільтр використовується для підвищення частоти дискретизації, то він називається інтерполятором. У такому разі диференціювальна ланка стоїть на першому місці, потім відбувається підвищення частоти дискретизації і, нарешті, йде ланка інтегрувального фільтра.

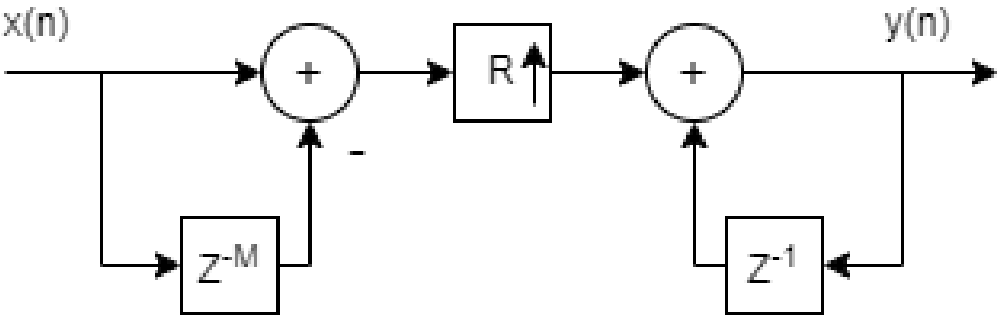

Рисунок 2.9 – Структура CIC фільтра інтерполятора

2 АПАРАТНА ПЛАТФОРМА FPGA DE0-NANO

2.1 Модель архітектури DE0-Nano

Основна концепція FPGA (Field Programmable Gate Array) полягає в тому, що програмована логіка дозволяє користувачам створювати власні апаратні рішення без необхідності виготовлення спеціалізованих інтегральних схем. На відміну від ASIC (прикладної інтегральної схеми), FPGA є універсальними та повторно програмованими, що дозволяє використовувати один пристрій для різноманітних застосувань.

FPGA-платформи, такі як DE0-Nano, дозволяють поєднувати всі аспекти цифрової системи: обробку сигналів, управління інтерфейсами, пам’ять та інші функції. Всі ці функції можуть бути реалізовані програмним шляхом за допомогою мови опису апаратури, наприклад, Verilog або VHDL. Рішення на основі FPGA мають такі переваги, як швидка адаптація до змін, нижча вартість розробки в порівнянні з ASIC, менші розміри системи та підвищена гнучкість.

Основними недоліками FPGA є вища енергоспоживання в порівнянні з ASIC та нижча швидкість роботи в окремих випадках. Проте завдяки повторній програмованості та можливості швидкої реалізації прототипів, FPGA широко використовуються в навчанні, дослідженнях і розробці цифрових систем.

DE0-Nano є популярною платформою для вивчення FPGA та реалізації цифрових систем. Високорівнева модель архітектури DE0-Nano складається з FPGA-чіпа Cyclone IV EP4CE22F17C6N, а також периферійних компонентів і високошвидкісних інтерфейсів, інтегрованих на друкованій платі. DE0-Nano включає такі основні компоненти:

- FPGA-чіп Altera Cyclone IV, що забезпечує 22 320 логічних елементів, 594 Кб пам’яті та підтримку мультиплікаторів для роботи з цифровими сигналами.

- Периферійні інтерфейси, зокрема GPIO (64 лінії), SPI, I²C та аналогово-цифрові перетворювачі.
- Інтегровані компоненти: 32 Мб SDRAM, 2 Мб флеш-пам’яті для зберігання прошивок та 3-вісний акселерометр для роботи з орієнтацією пристрою.
- Комунікація та живлення: USB-інтерфейс використовується для програмування та живлення плати. Підтримується також зовнішнє живлення через спеціальний конектор. Це забезпечує зручність під час розробки та тестування систем.

DE0-Nano є компактним і універсальним рішенням для проектів, що вимагають реалізації цифрової логіки, обробки сигналів або експериментів із вбудованими системами. Завдяки FPGA Cyclone IV, платформа дозволяє реалізовувати складні обчислювальні алгоритми та інтегрувати їх із зовнішніми сенсорами чи пристроями.

DE0-Nano також ідеально підходить для навчальних і дослідницьких цілей, завдяки своїй гнучкості, простоті використання та підтримці широкого спектра програмного забезпечення, такого як Intel Quartus Prime. Це робить платформу популярним вибором серед студентів, інженерів і розробників для прототипування та вивчення сучасних цифрових технологій.

2.2 Структура вбудованої платформи FPGA DE0-Nano

Базова структура FPGA DE0-Nano включає цифрову логіку, яка реалізована на основі FPGA Cyclone IV. Ця складова забезпечує високу гнучкість та масштабованість, дозволяючи реалізовувати будь-які цифрові схеми. Завдяки використанню LUT (таблиць пошуку) та тригерів, розробники можуть створювати спеціалізовані апаратні рішення для різних застосувань, таких як обробка сигналів чи керування потоками даних. Плата оснащена оперативною пам’яттю SDRAM об’ємом 32 МБ, що дозволяє зберігати значні обсяги даних і програмного коду. SDRAM підключається через 16-

бітну шину даних, забезпечуючи високу пропускну здатність і швидкий доступ до інформації, що особливо корисно для обчислювально-інтенсивних завдань, таких як алгоритми машинного навчання чи обробка зображень. DE0-Nano має вбудований цифровий акселерометр ADXL345 для точного вимірювання руху, а також ADC128S022 – аналогово-цифровий перетворювач із 8 каналами, який дозволяє обробляти аналогові сигнали. Світлодіоди, DIP-перемикачі та кнопки додають зручності в тестуванні та налагодженні. Розширювальні GPIO забезпечують 72 лінії вводу/виводу для підключення до зовнішніх пристроїв. Додатково доступні інтерфейси SPI та I2C для обміну даними з периферійними модулями, а також вбудований USB Blaster для програмування FPGA.

Ця структура дозволяє створювати інтегровані системи з високим рівнем продуктивності. Завдяки багатофункціональним компонентам, DE0-Nano підходить для широкого спектра задач – від навчальних проектів до розробки складних промислових систем. Різноманітні можливості конфігурації роблять платформу ідеальною для досліджень у галузі цифрової обробки сигналів, автоматизації та вбудованих систем.

2.3 Програмована логіка

Розробницька плата DE0-Nano, побудована на основі ПЛІС Cyclone IV E, демонструє виняткові можливості програмованості технології ПЛІС. На відміну від традиційних пристроїв із фіксованими функціями, ПЛІС дозволяє користувачам визначати власну апаратну архітектуру через програмування, що забезпечує неперевершену гнучкість для різноманітних застосувань.

Основою програмованої логіки в Cyclone IV є комбінаційні логічні блоки, тригери, блоки пам’яті (Block RAM) та блоки цифрової обробки сигналів (DSP). Логічні блоки містять LUT (таблиці станів) і тригери, які дозволяють виконувати складні цифрові обчислення. Блоки пам’яті забезпечують швидке зберігання даних із низькою затримкою і високою

пропускною здатністю, а DSP-блоки оптимізовані для арифметичної обробки, наприклад множення чи додавання. Ця структура надає розробникам можливість створювати рішення для задач, що вимагають високої продуктивності та паралельності.

Програмування ПЛІС здійснюється за допомогою мов опису апаратури, таких як Verilog або VHDL, у середовищі Altera Quartus. Ці мови дозволяють описувати поведінку логіки й архітектуру пристрою, після чого генерується конфігураційний файл. Цей файл завантажується в FPGA, визначаючи з’єднання та функціональність логічних елементів.

DE0-Nano підтримує JTAG програмування для прямої конфігурації, а також містить пристрій серійної конфігурації (EPCS64), що дозволяє зберігати дизайн у пам’яті навіть після вимкнення живлення. Це забезпечує автоматичну конфігурацію пристрою під час увімкнення, без потреби в ручному перепрограмуванні.

Багатий набір периферійних пристроїв, таких як кнопки, DIP-перемикачі, світлодіоди та розширювальні роз’єми, додатково демонструє гнучкість плати. Ці елементи можна інтегрувати у проєкти для створення прототипів систем — від простих механізмів управління до складних блоків обробки даних.

Наприклад, розробник може створити модуль обробки сигналів у реальному часі на DE0-Nano, де сирі аналогові дані перетворюються на цифрові за допомогою вбудованого АЦП і обробляються користувацькою логікою, запрограмованою в ПЛІС. Ця функція демонструє можливість перепрограмування системи для абсолютно нових завдань без зміни самого обладнання.

Загалом, програмованість плати DE0-Nano яскраво ілюструє потужність ПЛІС, дозволяючи реалізовувати динамічний підхід на апаратному рівні для вирішення обчислювальних та інженерних завдань. Незалежно від того, йдеться про створення прототипів, дослідження чи остаточне впровадження, ця гнучкість є основою технології ПЛІС.

2.4 Інтерфейси вводу/виводу

Рисунок 2.1 – Плата DE0-Nano Board та схема компонентів (вигляд зверху)

Основним способом взаємодії між FPGA та зовнішніми пристроями на платі DE0-Nano є використання широкого набору інтерфейсів, які забезпечують різні типи комунікації. Вони включають універсальні GPIO-роз'єми, послідовні інтерфейси I2C і SPI, а також спеціалізовані периферійні елементи, як-от АЦП та акселерометр. [1]

GPIO (General Purpose Input/Output) є універсальними контактами, що дозволяють підключати до FPGA різноманітні периферійні пристрої, такі як датчики, дисплеї чи мотори. Плата має два 40-контактні GPIO-роз'єми, кожен із яких забезпечує до 36 I/O-ліній. GPIO підтримують сигнали стандарту 3.3V, а також надають контакти живлення (5V і 3.3V). Це робить DE0-Nano зручним для інтеграції з іншими пристроями в системах автоматизації чи робототехніки.

SPI (Serial Peripheral Interface) використовується для підключення високошвидкісних периферійних пристроїв, таких як цифровий акселерометр

ADXL345. Цей інтерфейс забезпечує синхронний обмін даними між FPGA та периферією, що дозволяє досягати високої швидкості передачі інформації. Це ідеальний вибір для систем, які вимагають швидкої обробки даних, таких як обробка показників з датчиків.

I2C (Inter-Integrated Circuit) є двопровідним послідовним інтерфейсом, який забезпечує підключення до низькошвидкісних пристроїв, таких як EEPROM. Цей інтерфейс дозволяє організувати передачу даних між FPGA і периферійними пристроями з мінімальними витратами на апаратну реалізацію.

Аналого-цифровий перетворювач (АЦП) є одним із ключових компонентів DE0-Nano. Вбудований 8-канальний 12-бітний АЦП дозволяє перетворювати аналогові сигнали в цифрові значення для подальшої обробки в FPGA. Цей компонент є незамінним для роботи з аналоговими сенсорами, такими як температурні або світлові датчики. Він підтримує швидкість перетворення до 200 тис. вибірок на секунду, що дозволяє використовувати його для задач у реальному часі.

Кнопки та світлодіоди забезпечують простий спосіб тестування та налагодження проектів. Плата має дві кнопки, чотири DIP-перемикачі та вісім світлодіодів. Кнопки можна використовувати для подачі керуючих сигналів, а світлодіоди — для індикації стану системи.

Тактовий генератор є ще одним важливим компонентом DE0-Nano. Він забезпечує базову тактову частоту 50 МГц для роботи FPGA. Внутрішні PLL-блоки (Phase Locked Loop) дозволяють генерувати нестандартні частоти, необхідні для певних задач.

Завдяки використанню цих інтерфейсів, DE0-Nano може легко підключатися до зовнішніх пристроїв та забезпечувати ефективну обробку даних. Такий набір функціональних можливостей дозволяє використовувати плату в системах збору даних, обробки сигналів та управління пристроями, роблячи її універсальним інструментом для розробників.

2.5 Переваги FPGA DE0-Nano

Існує кілька унікальних переваг, які пропонує технологія FPGA, що робить її важливою в безлічі областей застосування. Основними перевагами є гнучкість і адаптація апаратної архітектури під конкретне завдання, що дає можливість розробляти індивідуальні рішення, які точно відповідають вимогам конкретного проекту. Можна навести приклад із радіотехніки, де FPGA дозволяють реалізовувати складні алгоритми обробки цифрового сигналу з мінімальною затримкою, що є дуже бажаною функцією в програмах реального часу.

Ще однією важливою перевагою FPGA є її здатність обробляти дані паралельно. Це означає, що він може виконувати купу завдань одночасно, що значно підвищує продуктивність у порівнянні з іншими технологіями. Ця функція особливо корисна для таких речей, як аналіз великих даних, криптографія або розпізнавання зображень. Наприклад, FPGA дуже добре працюють із нейронними мережами в машинному навчанні, пропонуючи швидкі обчислення з низьким енергоспоживанням.

FPGA також характеризуються енергоефективністю. Операції на рівні апаратної логіки зменшують витрати програмного забезпечення та можуть призвести до значно меншого споживання енергії такими системами. Це важливий момент для пристроїв у віддалених або автономних середовищах, наприклад, супутників або портативних пристроїв обробки медичних даних.

Ще однією перевагою є гнучкість FPGA протягом життєвого циклу продукту. Можливість перепрограмування платформи дозволяє вносити зміни або оновлення навіть після запуску продукту, тим самим істотно скорочуючи витрати, пов'язані зі створенням нових пристроїв. Ця можливість застосування помітна в контексті створення прототипів нових рішень для промисловості чи інноваційних досліджень. Проте використання FPGA складно. Розробка такої платформи вимагає досвіду в мовах опису обладнання, таких як Verilog або VHDL, що ускладнює доступ новачкам.

Крім того, вартість FPGA може стати значною, особливо у великих проектах після ліцензії на спеціалізоване програмне забезпечення для розробки.

Великі або складні системи можуть бути дещо обмежені обмеженнями внутрішньо кристальних ресурсів, таких як логічні елементи та пам’ять, навіть якщо FPGA є надзвичайно гнучким рішенням. Такі випадки часто вимагають від розробників значних витрат часу на оптимізацію проектів. Крім того, базові FPGA зазвичай не містять вбудованих процесорів, а зовнішні мікроконтролери повинні бути інтегровані для виконання програмних завдань.

Незважаючи на всі ці проблеми, FPGA доводять свою цінність у багатьох реальних ситуаціях. Наприклад, у сфері телекомунікацій вони потрібні для обробки швидких мережевих потоків, де має значення низька затримка. В автомобільній техніці FPGA допомагають створювати самокеровані автомобілі, надаючи швидку перевірку інформації датчиків одразу. Крім того, в авіації та космосі FPGA добре використовуються для виготовлення радіаційно-захищених систем.

Таким чином, навіть незважаючи на те, що FPGA має деякі обмеження, її особливості дозволяють платформі бути значущою у проєктах, де необхідна висока продуктивність, енергоефективність і адаптивність.

3 ТЕХНОЛОГІЇ СИМУЛЯЦІЇ ЦИФОВИХ ФІЛЬТРІВ

3.1 Симуляція та використання середовиша MATLAB

MATLAB, скорочено від "MATrix LABoratory", є потужним середовищем програмування, яке розроблене для виконання різноманітних технічних та наукових обчислень. Створене компанією MathWorks, воно стало універсальним інструментом для інженерів, науковців та дослідників у всьому світі, завдяки своїм інтуїтивно зрозумілим інтерфейсам, багатим функціональним бібліотекам та широким можливостям роботи з даними. MATLAB дозволяє швидко переходити від ідей до практичної реалізації, забезпечуючи ефективність у чисельних розрахунках, обробці даних, моделюванні та створенні графічних візуалізацій.

Основна особливість MATLAB – це матричний підхід до обчислень, що дозволяє легко працювати з великими обсягами даних та здійснювати над ними складні операції. Ця особливість особливо важлива в інженерних, наукових та фінансових обчисленнях, де потрібна висока точність та ефективність обробки чисел. MATLAB є універсальним середовищем за зручністю та можливостями. MATLAB дозволяє виконувати математичні розрахунки, включаючи операції з лінійною алгеброю, статистичні розрахунки, оптимізацію, обробку сигналів та багатовимірний аналіз. Завдяки цьому інженери та дослідники можуть вирішувати складні математичні задачі, не вдаючись до складної реалізації алгоритмів "з нуля". MATLAB має широкі можливості для графічної візуалізації даних, включаючи двовимірні та тривимірні графіки, контурні графіки, поверхні, гістограми та інші інструменти. Це особливо корисно для інженерного аналізу, адже візуалізація допомагає краще зрозуміти тенденції та закономірності, а також полегшує комунікацію результатів досліджень з іншими спеціалістами. MATLAB також широко використовується в області

цифрової обробки сигналів і зображень, пропонуючи набір інструментів для фільтрації, спектрального аналізу, часово-частотного аналізу та інших операцій. Це робить MATLAB зручним вибором для задач, які потребують обробки і покращення якості сигналів або зображень. MATLAB надає потужні засоби для створення та тестування моделей машинного навчання, глибокого навчання та штучного інтелекту, включаючи широкий набір алгоритмів для класифікації, регресії, кластеризації та побудови нейронних мереж. Це дозволяє швидко створювати й тренувати моделі для аналізу великих обсягів даних. MATLAB також дозволяє моделювати динамічні системи в реальному часі, а середовище Simulink надає додаткові можливості для графічного моделювання та симуляції складних систем. Це особливо корисно для задач управління та автоматизації, де потрібно створити модель складної системи і перевірити її в умовах, максимально наближених до реальних.

У даній роботі MATLAB використовується як основний інструмент для симуляції та аналізу цифрових фільтрів, зокрема кінцевих імпульсних фільтрів та каскадних інтегрально-гребінчастих фільтрів. MATLAB забезпечує можливості для швидкого створення та тестування фільтрів, що дозволяє детально досліджувати їх характеристики, ефективність у фільтрації шуму та покращенні сигналу.

MATLAB пропонує широкий набір вбудованих функцій, таких як filter, filtfilt, butter, impz тощо, що значно спрощує процес проектування та тестування фільтрів для різних застосувань. MATLAB дозволяє точно налаштовувати різні параметри фільтра, як-от частоту зрізу, порядок фільтра, тип фільтра, що полегшує процес оптимізації під конкретні вимоги системи. MATLAB дозволяє візуалізувати результати симуляцій у вигляді графіків, що допомагає побачити вплив фільтрації на зашумлений сигнал і порівняти результати до та після обробки. MATLAB дає змогу швидко перевірити теоретичні знання щодо обробки сигналів та фільтрації. Зміна параметрів фільтра і перегляд результатів у реальному часі дозволяє отримати цінні

інсайти для практичного налаштування систем. MATLAB дозволяє побудувати симуляції низькочастотного фільтра, який зменшує шум у цифрових сигналах. MATLAB дозволяє налаштовувати такі параметри, як порядок фільтра, частота зрізу, тип фільтра, що є важливим для досягнення бажаного результату. Це корисно для точного налаштування фільтра під конкретні умови використання і підвищення якості обробки сигналів.

MATLAB є надзвичайно цінним інструментом у дослідженнях з цифрової обробки сигналів завдяки широким можливостям для моделювання, симуляції, візуалізації результатів та налаштування параметрів фільтрів. MATLAB дозволяє побудувати симуляції та CIC фільтрів, аналізувати їх продуктивність, досліджувати особливості фільтрації та вивчати поведінку зашумлених сигналів після обробки. Цей інструмент є незамінним для дослідників, які прагнуть розширити розуміння та оптимізувати процеси обробки сигналів для подальших інженерних застосувань.

3.2 Моделювання CIC-фільтра в MATLAB

Для дослідження CIC-фільтрів спершу було написано програму в середовищі MATLAB, яка дозволяє окремо будувати амплітудно-частотні характеристики (АЧХ) для різних коефіцієнтів децимації, від 2 до 10, CIC фільтра першого порядку. Ця програма реалізує побудову індивідуальних графіків для кожного коефіцієнта. Результати представлені у вигляді дев’яти окремих графіків, що ілюструють зміну АЧХ залежно від коефіцієнта децимації (рис. 3.1)

Лістинг 3.1

```
% Початкові коефіцієнти фільтра
b = [1 1];
% Для кожного графіка задаємо комбінацію "LineStyle" та "Marker"
styleDefs = {
    struct('ls','-','mk','+')
```

```
        struct('ls','--','mk','+')
        struct('ls',':','mk','+')
        struct('ls','-.','mk','+')
        struct('ls','-','mk','*')
        struct('ls','--','mk','*')
        struct('ls',':','mk','*')
        struct('ls','-.','mk','*')
        struct('ls','-','mk','o')};
    figure('Position', [100 100 1600 800], 'Color', 'w');
    for i = 1:9
        subplot(3, 3, i); hold on; grid on;
        [H, ~] = freqz(b, 1, N);
        H_mag_dB = 20*log10(abs(H)/max(abs(H)) + 1e-12);
        % Зчитуємо стиль i-го рядка
        ls = styleDefs{i}.ls;  % лінія (наприклад, '-')
        mk = styleDefs{i}.mk;  % маркер (наприклад, '+')
        % Побудова в чорному кольорі з вибраними LineStyle та
Marker
        plot(f, H_mag_dB, ...
            'LineWidth', 2.5, ...
            'LineStyle', ls, ...
            'Marker', mk, ...
            'Color', 'k', ...
            'MarkerIndices',  1:1500:length(f),  ...     %  щоб
маркери не були на кожній точці
            'DisplayName', ['R = ' num2str(length(b))]);
        xlim([0 0.5]);
        ylim([-30 0]);
        % Додаємо 1 у кінець вектора b (для наступного каскаду)
        b = [b 1];
        legend('Location','northeast','FontSize',12);
    end
```

На наступному етапі дослідження було створено програму для побудови АЧХ на єдиному графіку, але вже лише для коефіцієнтів децимації від 2 до 4. Це дало змогу зосередитися на порівнянні змін в АЧХ у більш стислому діапазоні децимації. Графік, що демонструє ці результати, наведено на рисунку 3.2.

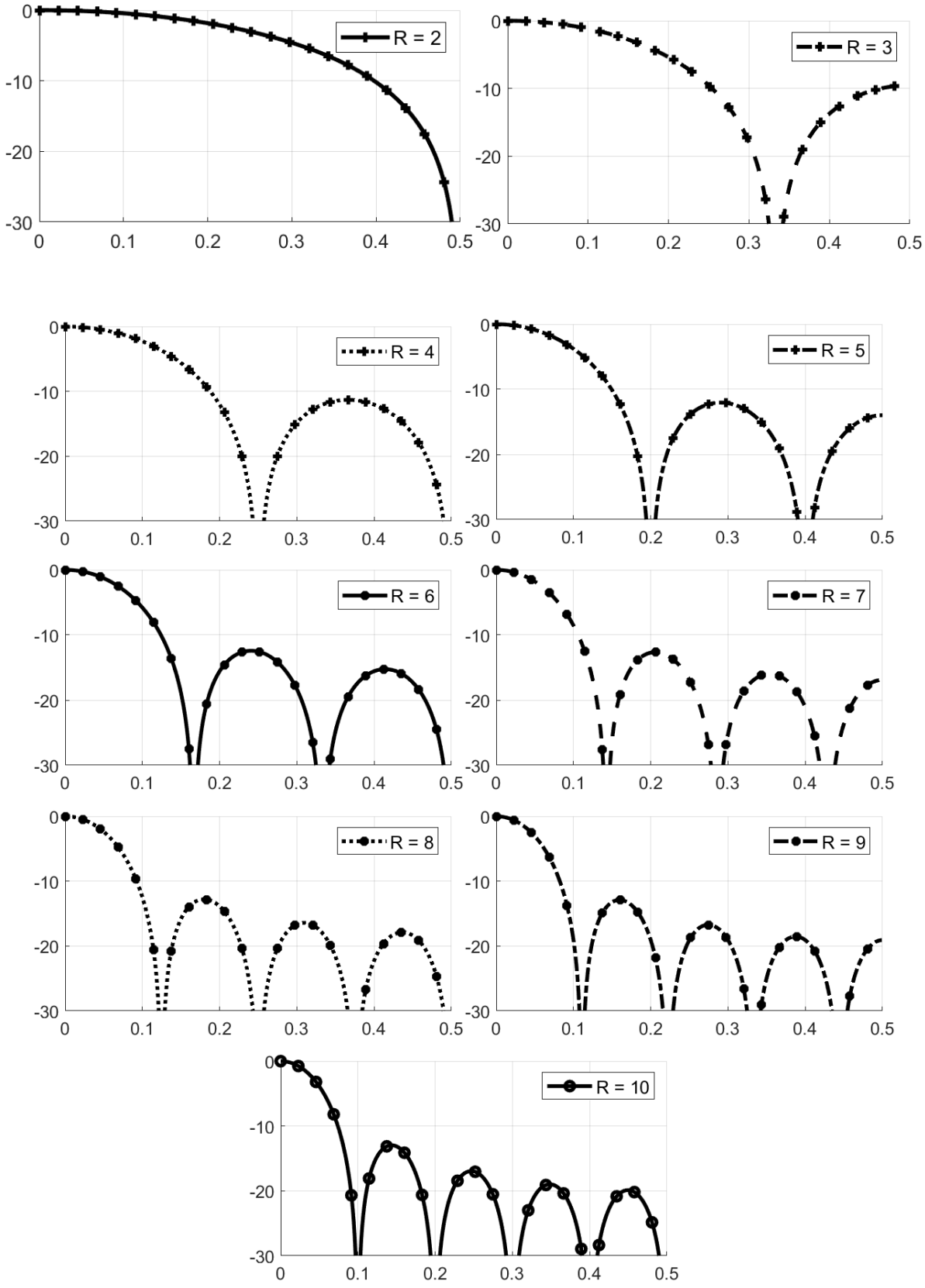

Рисунок 3.1 – Графіки частотної характеристики CIC-фільтра

Лістинг 3.2

```
clear; clc; close all;
N = 2^15;
% Вектор нормованих частот [0 .. 0.5]
f = linspace(0, 0.5, N);
% Початкові коефіцієнти CIC-фільтра
b = [1];
% Масив стилів ліній для 4 графіків
lineStyles4 = {'-', '--', ':', '-.'};
figure('Position',[100 100 1200 400], 'Color','w');
hold on; grid on;
for i = 1:4
    [H, ~] = freqz(b, 1, N);
    H_mag = abs(H);
    H_mag_dB = 20*log10(H_mag / max(H_mag) + 1e-12);

    plot(f, H_mag_dB, ...
         'LineWidth', 2.5, ...
         'LineStyle', lineStyles4{i}, ...
         'Color', 'k', ...
         'DisplayName', ['R = ' num2str(length(b))]);
    b = [b 1];
end

xlim([0 0.5]);
ylim([-30 0]);
xlabel('Нормована частота');
ylabel('Амплітуда (дБ)');
title('Частотна характеристика CIC-фільтра (4 графіки)');
legend('Location','northeast');
```

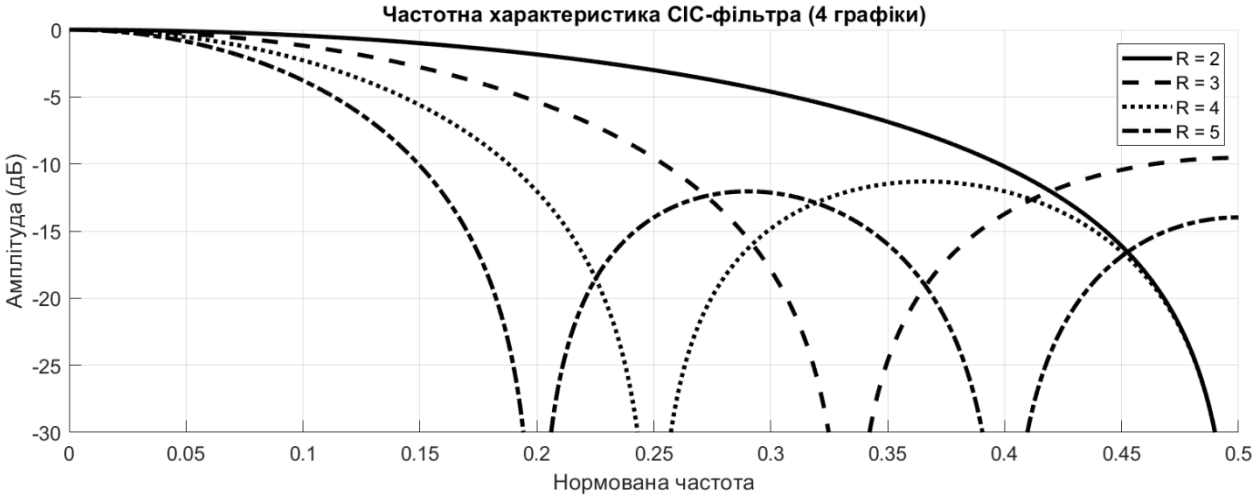

Рисунок 3.2 – Графіки АЧХ CIC-фільтру коефіцієнту децимації 2-5 [8]

Можна побачити (рис. 3.2), що зі збільшенням коефіцієнта смуга пропускання звужується. Частота першого нуля (f=1/R) зменшується, дозволяючи фільтру ефективніше придушувати високочастотні компоненти. Однак бокові пелюстки стають вужчими без суттєвого зменшення їхньої амплітуди, що робить фільтр більш чутливим до низькочастотних сигналів і менш ефективним у придушенні високочастотних компонентів поза смугою пропускання.

При збільшенні коефіцієнта децимації R, який визначає розмір вікна, смуга пропускання стає ще вужчою. Ширина цієї смуги може бути приблизно оцінена як 2/R. Лепестки АЧХ при цьому стають більш вузькими, однак їхня амплітуда практично не змінюється. Фільтр стає більш чутливим до низьких частот, але водночас менш ефективно пригнічує високі частоти, що знаходяться за межами смуги пропускання.

Фільтр без гребінчастого каскаду та інтегратора не має каскадної структури і не виконує додаткових перетворень. Він фактично еквівалентний фільтру ковзного середнього із кутовим відгуком у частотній області. У реальних умовах децимація з коефіцієнтом із таким фільтром може призводити до значного накладання спектрів (aliasing), оскільки гребінчаста структура є необхідною для придушення високочастотних компонентів перед децимацією. [4]

Далі було проведено експеримент, при якому R оставалось незмінним, а кількість каскадів збільшувалась поступово. Було написано відповідно програму, яка наведена в лістингу 3.3.

З результатів моделювання можна побачити що, АЧХ CIC-фільтра залежить від кількості каскадів N. Збільшення кількості каскадів призводить до зміни форми АЧХ та впливає на наступні параметри.

1. Збільшення крутості фільтрації (селективності):

– при збільшенні числа каскадів N фільтр стає більш "вузькосмуговим", це означає, що пригнічення частот поза пропускною смугою значно покращується;

– крутизна скатів фільтра зростає: поблизу частоти зрізу, придушення небажаних частот стає ефективнішим.

– глибина загасання на частотах за межами смуги пропускання зростає.

2. Звуження смуги пропускання:

– зі зростанням N ширина головної пелюстки (смуги пропускання) зменшується, що робить фільтр більш вузькосмуговим.

– призводить до збільшення спотворень у корисній смузі сигналу (наприклад, завали АЧХ поблизу краю смуги пропускання).

3. Збільшення рівня загасання бічних пелюсток:

– збільшення кількості каскадів знижує амплітуду бічних пелюсток.

– рівень придушення бічних пелюсток збільшується приблизно пропорційно 20lg(N) у децибелах.

Для CIC-фільтра з коефіцієнтом децимації R=4 можна побачити наступне. При N=1 АЧХ нагадує один ступінь низькочастотного фільтра з широкою смугою пропускання та невисоким загасанням бічних пелюсток. При N=2 фільтр стає більш вузькосмуговим, з кращим придушенням поза смужкою. При N=3 згасання поза смугою пропускання збільшується ще сильніше, але смуга пропускання стає вже.

Лістинг 3.3

```
%% Plot 2: Variable N (M=2, R=2)
R = 2; % Decimation factor
N_values = [1, 2, 3, 4, 5]; % Number of stages

figure;
hold on;
for i = 1:length(N_values)
    N = N_values(i);
    H = cic_response(f, R, M, N);
    H(f == 0) = 1; % Handle division by zero
```

```
        plot(f, 20 * log10(H), 'LineStyle',
lineStyles{i},'Color', 'k', 'DisplayName', sprintf('N = %d',
N));
    end
    title('Frequency Response: Variable N (M=2, R=2)',
'FontSize', 14);
    xlabel('Normalized Frequency (f/fs)', 'FontSize', 12);
    ylabel('Magnitude (dB)', 'FontSize', 12);
    grid on;
    legend show;
    ylim([-100, 5]);
```

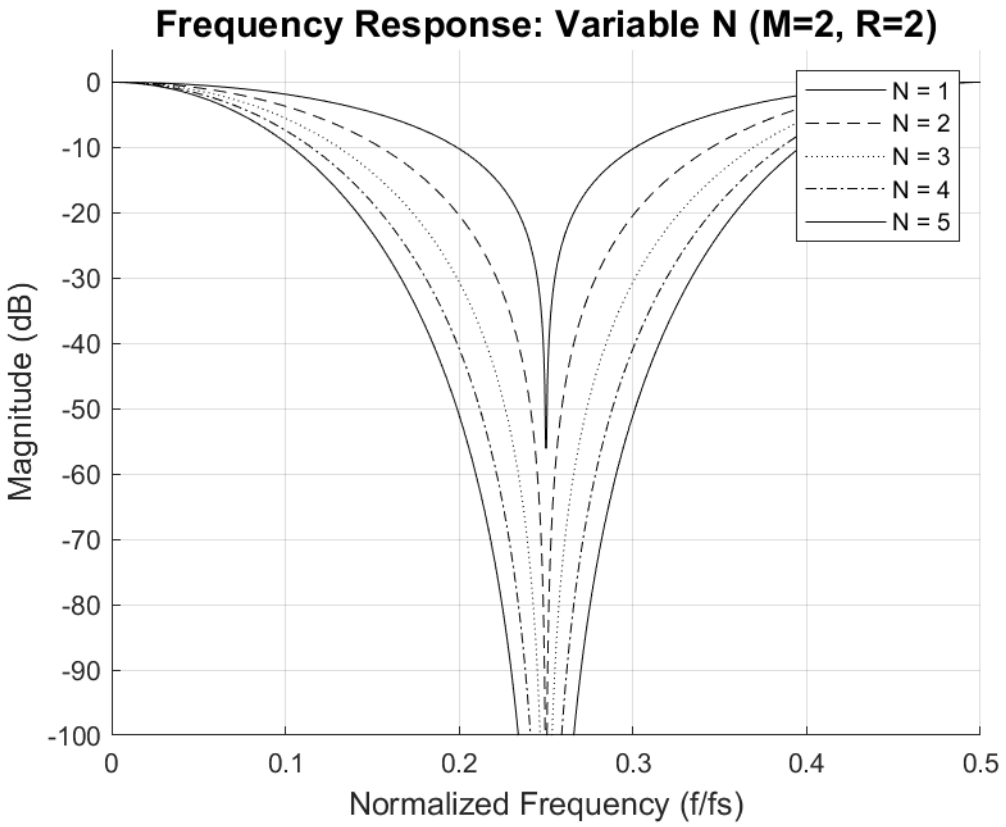

Рисунок 3.3 – АЧХ для різної кількості каскадів (N= 1-5)

Для подальшого дослідження було реалізовано модель фільтра-дециматора. Код, який описує дану модель наведено у лістингу 3.4.

Лістинг 3.4 – Реалізація CIC-дециматора

```
clear; clc; close all;
N = 500;
flt_r = [3, 6, 4, 6, 8, 6, 15];
```

```
    flt_n = [4, 1, 4, 2, 4, 8, 4];
    plot_filter_local(flt_r, flt_n, N, 'Decimator');
    function plot_filter_local(r, n, samples, modeVal)
        tt = linspace(0, 1, samples);
        rng(1, 'twister');
        if strcmp(modeVal, 'Decimator')
            x = 1.5*sin(4*pi*tt) + 1.7*sin(8.3*pi*tt);
            x = x + 0.9*randn(1, samples);
        end
        clfObj = CicFilter(x);
        zz = cell(1, length(r));
        if strcmp(modeVal, 'Decimator')
            for k = 1:length(r)
                zz{k} = clfObj.decimator(r(k), n(k));
            end
        else
            for k = 1:length(r)
                zz{k} = clfObj.interpolator(r(k), n(k), true);
            end
        end
        figure('Position', [100 100 1600 800], 'Color','w');
        subplot(4,2,1); hold on; grid on;
        title('Вхідний сигнал');
        plot(x, '-', 'Color', [0 0.4470 0.7410],
'DisplayName','Сигнал');
        xlim([1 samples]);
        legend('Location','northeast','FontSize',14);
        for j = 1:length(r)
            subplot(4,2,1 + j); hold on; grid on;
            if j == 1, title('Результат CIC'); end
                    hSt = stem(zz{j}, 'Color', [0.4660 0.6740
0.1880], ...
                       'LineWidth', 1.5, ...
                       'DisplayName', ['R=', num2str(r(j)), ',
N=', num2str(n(j))]);
            hSt.Marker = 'o';
            hSt.MarkerSize = 5;
            hSt.MarkerFaceColor = [0.2 0.4 0.8];
            hSt.MarkerEdgeColor = 'none';
            hSt.BaseValue = 0;

legend('Location','northeast','FontSize',14);
        end
    end
```

Код (лістинг 3.4) демонструє застосування класу CicFilter для роботи з CIC-фільтром, який використовується для зменшення частоти дискретизації сигналу (децимації). У прикладі створюється функція для візуалізації роботи дециматора. Вхідний сигнал генерується як сума синусоїд з додаванням

шуму. Конфігурація фільтра задається двома масивами параметрів: коефіцієнтами децимації flt_r і кількістю каскадів flt_n.

Функція plot_filter_local приймає ці параметри, генерує вхідний сигнал, створює об’єкт CIC-фільтра та обробляє сигнал методом decimator для кожного набору параметрів. Результати роботи фільтра візуалізуються у вигляді графіків, що показують вплив децимації на сигнал.

На графіках (рис 3.4, рис 3.5) зображений вихідний сигнал після обробки для кожної пари параметрів фільтра. Налаштовані маркери та кольори дозволяють наочно побачити, як змінюється структура сигналу після застосування фільтра.

Після виконання лістингу 3.4 реалізується робота CIC-фільтра в режимі децимації. Цей приклад демонструє, як змінюється частота дискретизації сигналу та його частотні характеристики при різних значеннях коефіцієнта децимації R і порядку фільтра N. Збільшення R зменшує кількість відліків, ефективно усуваючи високочастотний шум, але водночас може призводити до втрати високочастотної інформації. Зростання N дозволяє досягти більш ефективного згладжування сигналу за рахунок зниження високочастотних компонентів.

Однак децимація впливає не лише на частоту дискретизації, але й на розрядність вихідних даних. Вихідна розрядність для CIC-дециматора обчислюється за формулою:

$$B_{out} = \left\lceil \log_2\left(\frac{(RM)^N}{R}\right) + B_{in} \right\rceil, \qquad (3.1)$$

де B_{in} – розрядність вхідних даних, M – параметр затримки, R – коефіцієнт децимації, N – порядок фільтра. Це означає, що зі збільшенням параметрів R, M і N може значно зрости кількість необхідних бітів для представлення вихідного сигналу. У результатах децимації можна спостерігати, як ці зміни впливають на сигнал.

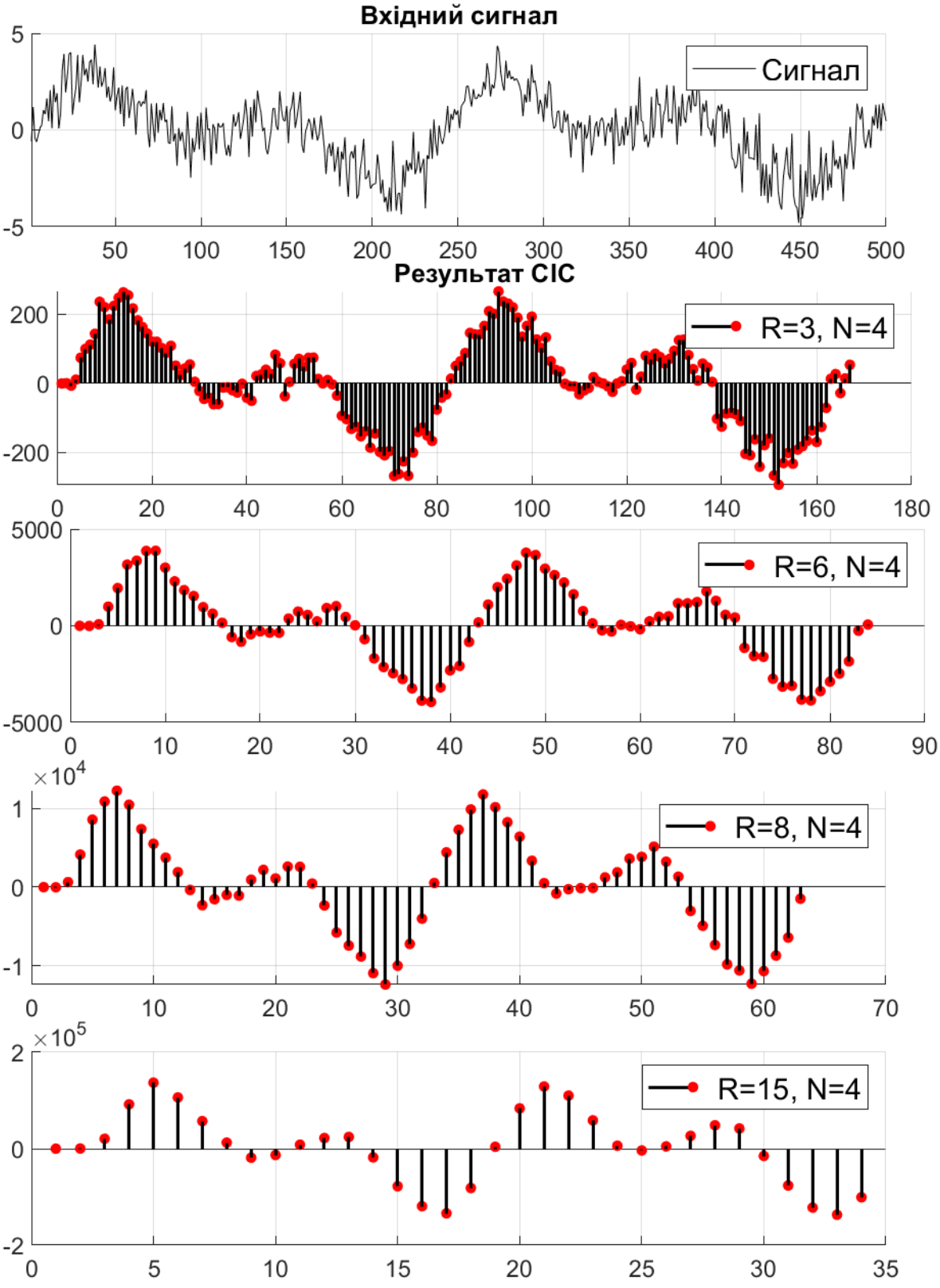

Рисунок 3.4 – Вплив коефіцієнта децимації R на вихідний сигнал

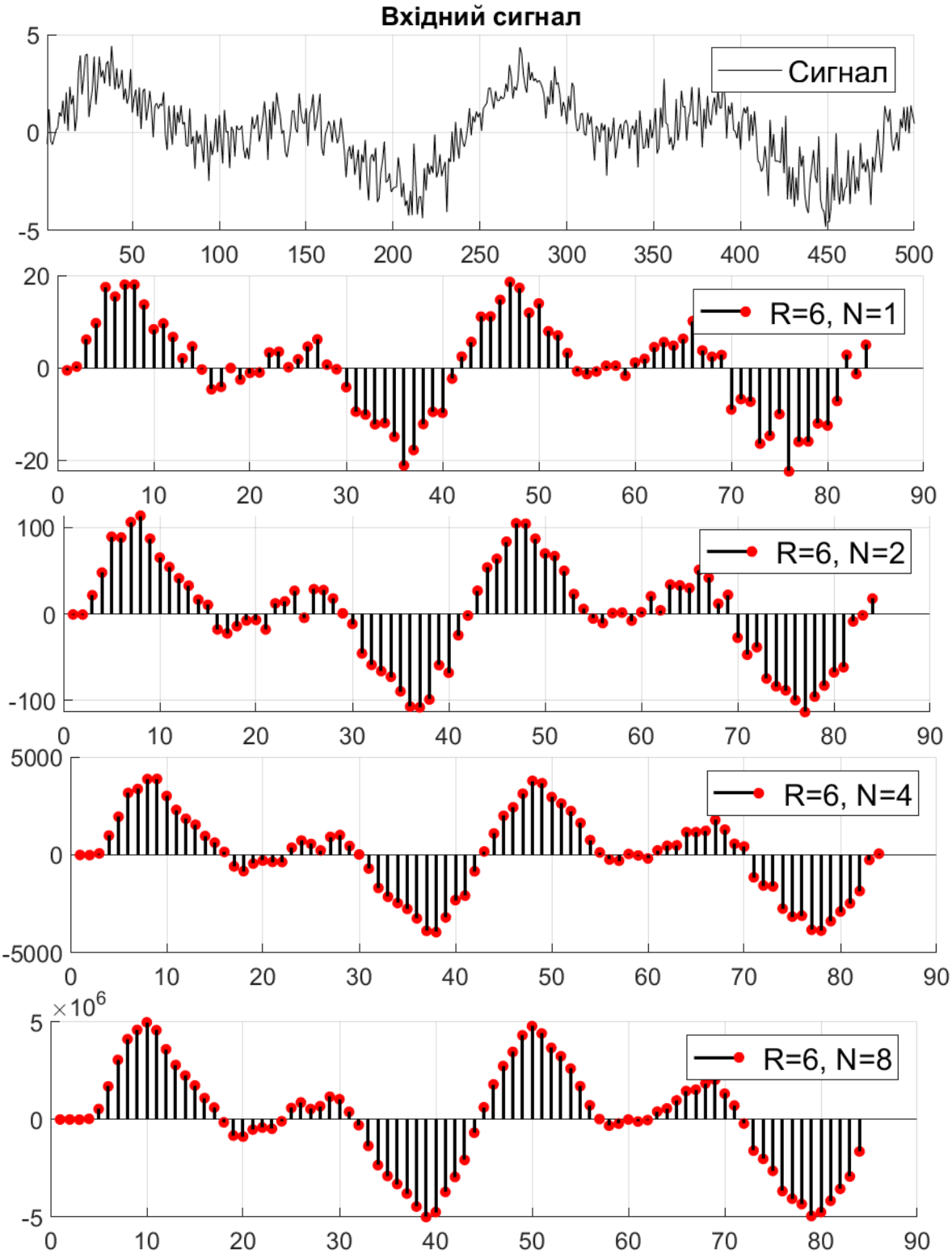

Рисунок 3.5 – Вплив порядку фільтра N на вихідний сигнал

На графіках, де порядок фільтра N залишається постійним, а змінюється коефіцієнт децимації R, спостерігається значний вплив цього параметра на роботу CIC-дециматора. Зі збільшенням R знижується частота

дискретизації вихідного сигналу, що проявляється як зменшення кількості відліків. Це пояснюється тим, що дециматор зберігає лише кожен R-й відлік із вхідного сигналу. Можна побачити, що збільшення R веде до зниження щільності точок у вихідному сигналі, оскільки частота дискретизації зменшується через вибір кожного R-го відліку.

Водночас зі збільшенням R відбувається значне зростання амплітуди вихідного сигналу, що є наслідком збільшення коефіцієнта передачі фільтра. Це зростання вимагає врахування додаткових обчислювальних ресурсів для забезпечення точності представлення сигналу.

На графіках, де коефіцієнт децимації R залишається постійним, а змінюється порядок фільтра N, видно, як це впливає на згладжування та амплітуду вихідного сигналу. Зі збільшенням N фільтр стає більш ефективним у видаленні високочастотних компонентів, що забезпечує плавніший і згладжений вихідний сигнал. Це обумовлено тим, що збільшення порядку фільтра означає більшу кількість каскадів інтеграції та комбі, через які проходить сигнал.

Проте зі зростанням N також відбувається значне підвищення амплітуди сигналу. Це є наслідком збільшення коефіцієнта передачі фільтра, що потребує врахування при реалізації системи.

Таким чином, збільшення порядку фільтра дозволяє досягти більш ефективного згладжування та видалення шуму, але це також призводить до підвищення вимог до амплітуди та розрядності сигналу. Оптимальне значення N необхідно підбирати залежно від вимог технічного завдання та ресурсних обмежень системи.

Лістинг 3.5 – Реалізація CIC-інтерполятора

```
%% Приклад 2: CIC-Інтерполятор
clear; clc; close all;
N = 30;
flt_r = [2, 3, 3, 3, 5, 3, 7];
flt_n = [3, 1, 3, 2, 3, 6, 3];
plot_filter_local(flt_r, flt_n, N, 'Interpolator');
```

```
%% Локальна функція для побудови графіків CIC-фільтра
function plot_filter_local(r, n, samples, modeVal)
    tt = linspace(0, 1, samples);
    rng(1, 'twister');
    if strcmp(modeVal, 'Decimator')
        x = 1.5*sin(4*pi*tt) + 1.7*sin(8.3*pi*tt);
        x = x + 0.9*randn(1, samples);
    elseif strcmp(modeVal, 'Interpolator')
        x = sin(1.7*pi*tt) + 1.7*sin(5.3*pi*tt);
        x = x + 0.3*randn(1, samples);
    else
        error('modeVal має бути Decimator або Interpolator.');
    end
    clfObj = CicFilter(x);
    zz = cell(1, length(r));
    if strcmp(modeVal, 'Decimator')
        for k = 1:length(r)
            zz{k} = clfObj.decimator(r(k), n(k));
        end
    else
        for k = 1:length(r)
            zz{k} = clfObj.interpolator(r(k), n(k), true);
        end
    end
    figure('Position', [100 100 1600 800], 'Color','w');
    subplot(4,2,1); hold on; grid on;
    title('Вхідний сигнал');
    plot(x, '-', 'Color', [0 0.4470 0.7410], 'DisplayName','Сигнал');
    xlim([1 samples]);
    legend('Location','northeast','FontSize',14);
    for j = 1:length(r)
        subplot(4,2,1 + j); hold on; grid on;
        if j == 1, title('Результат CIC'); end
              hSt = stem(zz{j}, 'Color', [0.4660 0.6740 0.1880], ...
                'LineWidth', 1.5, ...
                'DisplayName', ['R=', num2str(r(j)), ', N=', num2str(n(j))]);
        hSt.Marker = 'o';
        hSt.MarkerSize = 5;
        hSt.MarkerFaceColor = [0.2 0.4 0.8];
        hSt.MarkerEdgeColor = 'none';
        hSt.BaseValue = 0;
              legend('Location','northeast','FontSize',14);
    end
end
```

Результати моделювання роботи CIC-фільтра в режимі інтерполяції наведені на рисунку 3.6 – 3.7. У цьому прикладі демонструється, як коефіцієнт інтерполяції R і порядок фільтра N впливають на якість сигналу. Збільшення R додає більше відліків між початковими точками сигналу, що дозволяє масштабувати його для систем із вищою частотою дискретизації.

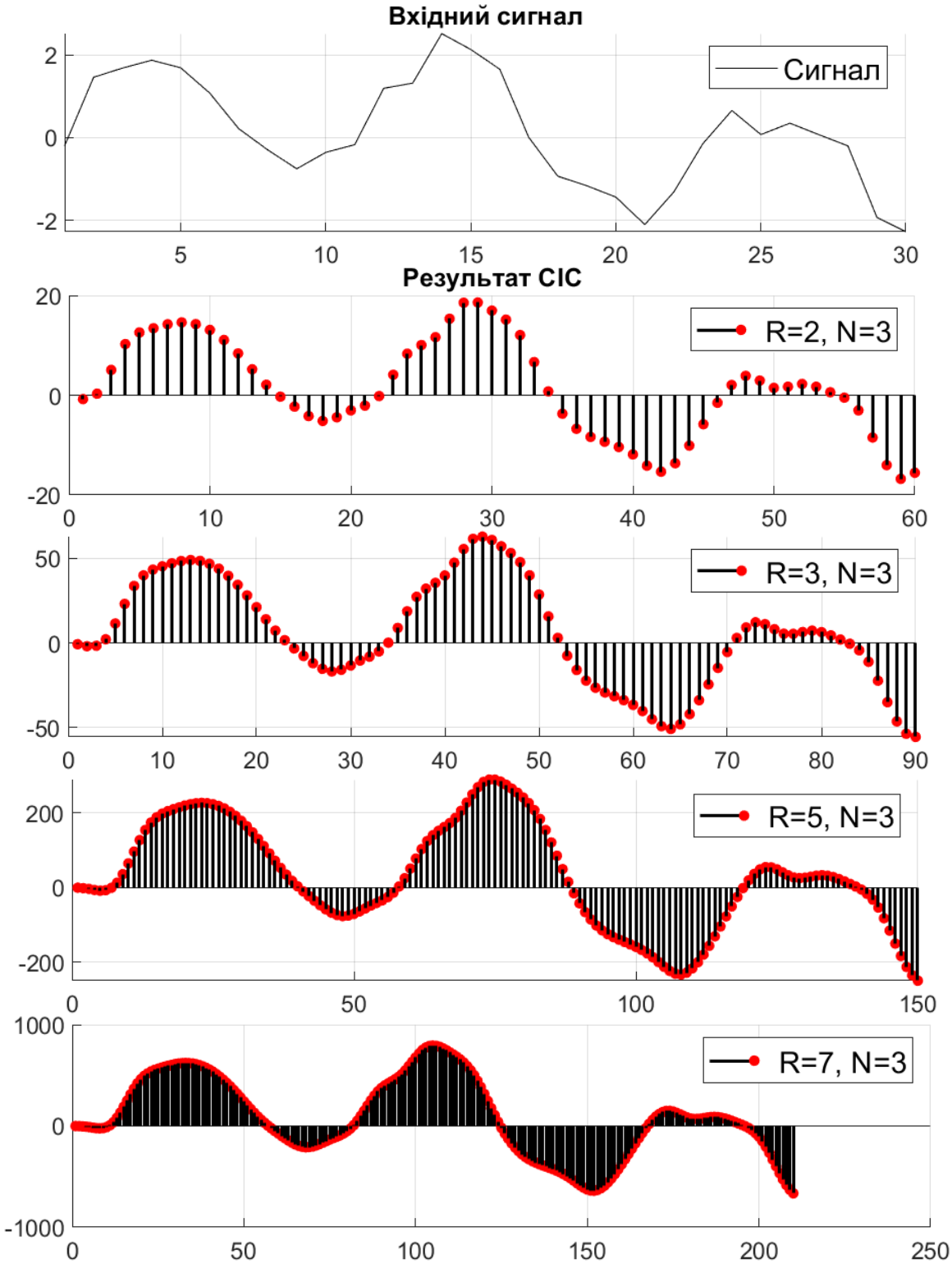

Рисунок 3.6 – Результати обробки сигналу CIC фільтром, порядок фільтра N=3, коефіцієнт інтерполяції R змінний

Параметр N визначає рівень згладжування сигналу: зі зростанням N фільтр ефективніше видаляє високочастотні компоненти, роблячи сигнал більш гладким, але також додає затримку. Важливо враховувати, що інтерполяція впливає на розрядність вихідних даних. Для CIC-інтерполятора розрядність визначається за формулою:

$$B_{out} = \lceil N \log_2(RM) + B_{in} \rceil, \quad (3.2)$$

де B_{in} – розрядність вхідних даних, M – параметр затримки, R – коефіцієнт інтерполяції, N – порядок фільтра.

Зі збільшенням параметрів R, M і N відбувається значне зростання вихідної розрядності, що критично для систем із обмеженими обчислювальними ресурсами.

На основі графіків (рис 3.5), де порядок фільтра N залишається постійним, а змінюється коефіцієнт інтерполяції R, можна зробити кілька важливих спостережень. Зі збільшенням R у вихідному сигналі додається більше точок між початковими відліками, що підвищує частоту дискретизації. Водночас стає помітним посилення високочастотних коливань, оскільки зі зростанням R більше енергії високих частот потрапляє у спектр сигналу. Це проявляється як збільшення інтенсивності коливань у вихідному сигналі.

Порядок фільтра N, який залишається постійним, забезпечує фіксований рівень згладжування, завдяки чому сигнал виглядає відносно стабільно на низьких частотах. Проте зі зростанням R зростає амплітуда вихідного сигналу, що є наслідком збільшення коефіцієнта передачі фільтра.

Загалом, збільшення R дозволяє досягти більш високої частоти дискретизації, але це потребує контролю за амплітудою сигналу та його розрядністю. Вибір оптимального значення R має враховувати компроміс між частотою дискретизації, збереженням низькочастотної інформації та можливими високочастотними спотвореннями.

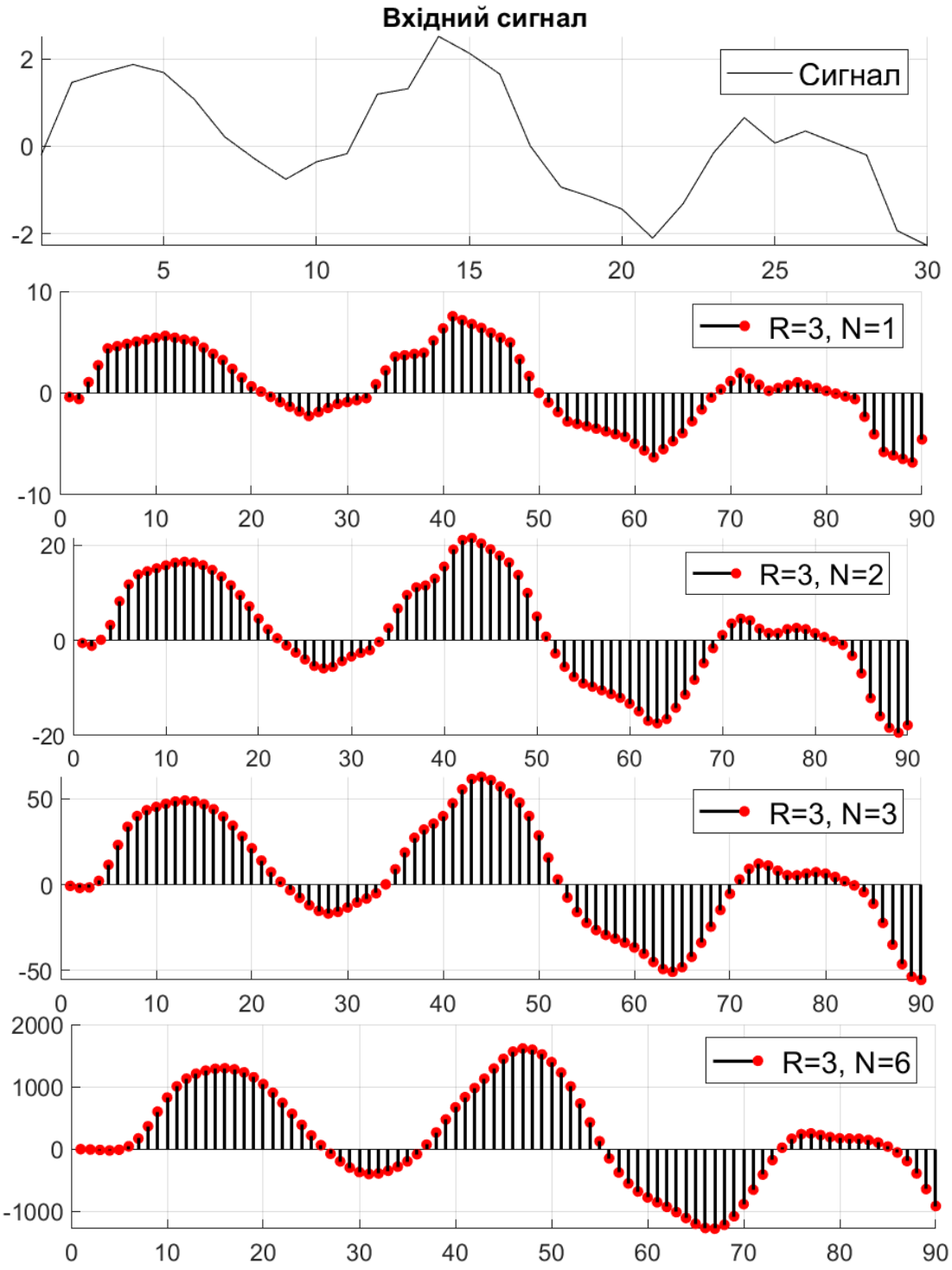

Рисунок 3.7 – Результати обробки сигналу CIC фільтром, коефіцієнта інтерполяції R=3, порядок фільтра N змінний

Результати інтерполяції демонструють, як CIC-фільтр справляється із завданням підвищення частоти дискретизації, забезпечуючи збереження низькочастотної інформації та ефективне згладжування шуму.

Далі було розроблено модель CIC фільтра в Simulink MATLAB, яка наведена на рисунку 3.8.

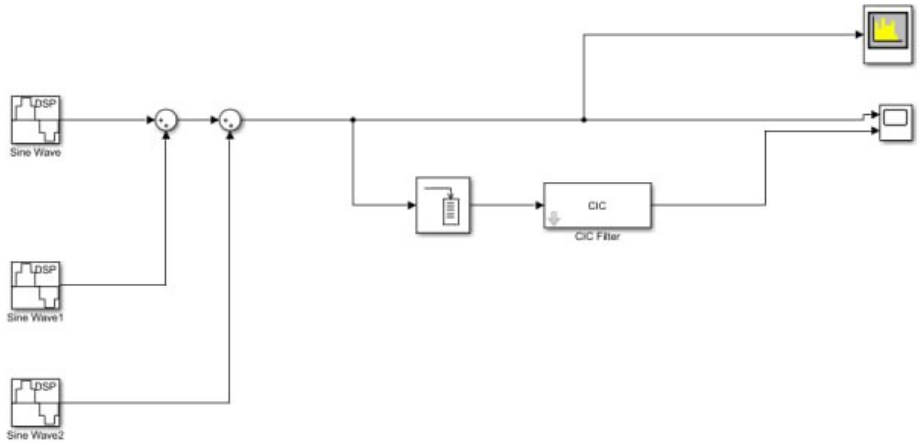

Рисунок 3.8 – Блок-схема структура CIC фільтру дециатора.

Налаштування вхідних сигналів наведено на рисунку 3.9. Для цього використовується три генератори синусоїдальних сигналів, кожен із яких має унікальні параметри, такі як амплітуда, частота і початкова фаза.

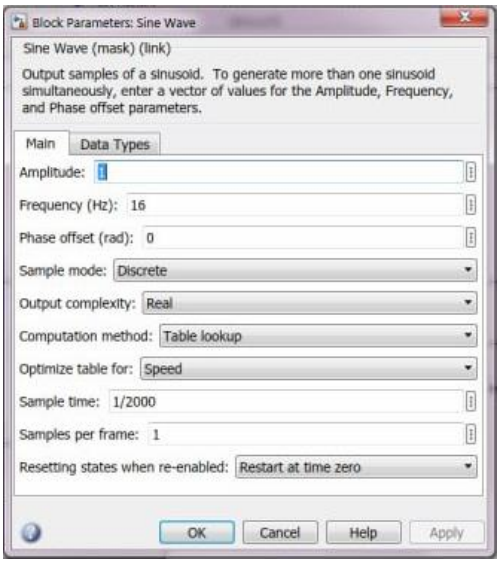

Рисунок 3.9 – Налаштування першого вхідного сигналу

Перший сигнал задається через блок Sine Wave. Його амплітуда – 1, частота 16 Гц (вибрана для створення базового низькочастотного компоненту). Початкова фаза 0 рад.

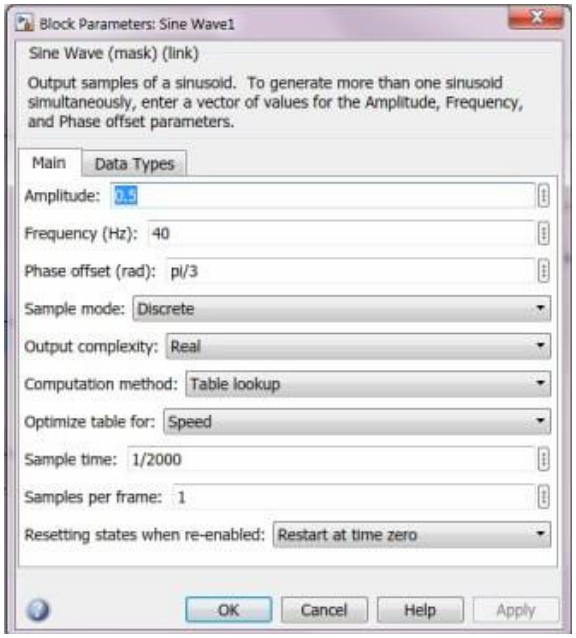

Рисунок 3.10 – Налаштування другого вхідного сигналу

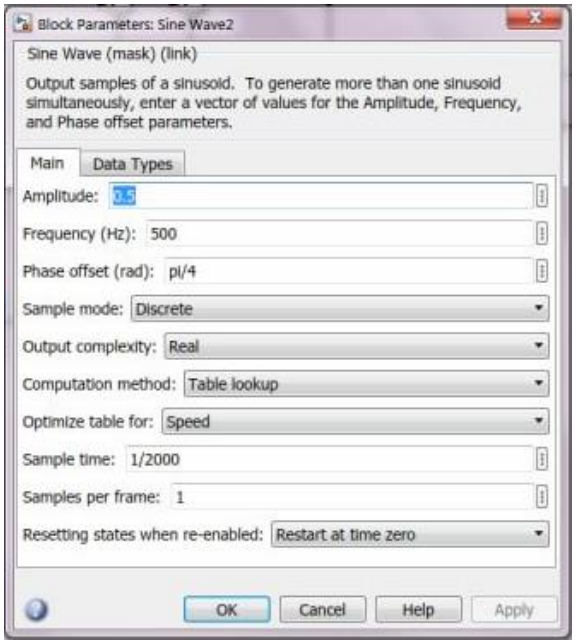

Рисунок 3.11 – Налаштування третього вхідного сигналу

Другий сигнал налаштовується через блок Sine Wave1. Амплітуда цього сигналу зменшена до 0.5, частота 40 Гц. Початкова фаза π/3 рад, щщо дозволяє змістити сигнал у часі, додаючи фазовий зсув.

Третій сигнал створюється через блок Sine Wave2. Його амплітуда також 0.5, але частота значно вища – 500 Гц, що додає високочастотний компонент до загального сигналу. Початкова фаза π/4 рад використовується для додаткового зсуву в часі, забезпечуючи унікальну поведінку сигналу.

Ці три сигнали разом складаються та формують комплексний вхідний сигнал для CIC фільтра, забезпечуючи різноманітність амплітуд, частот і фаз.

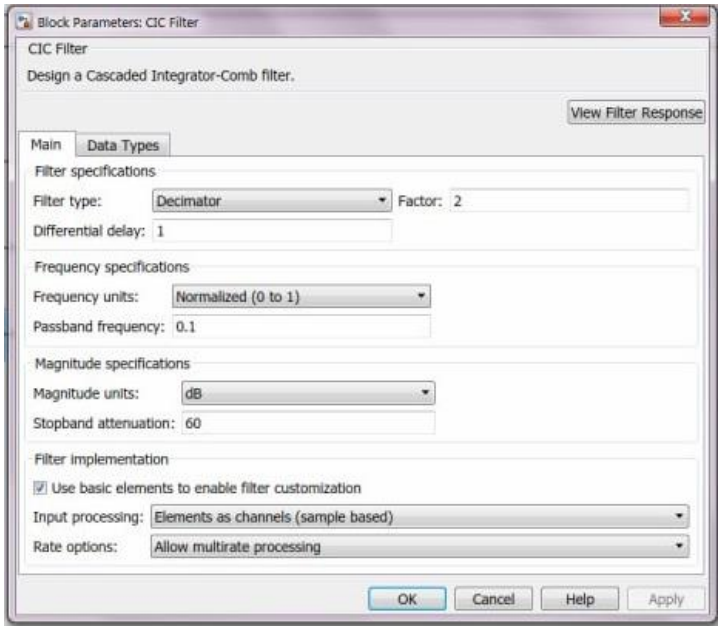

Рисунок 3.12 – Вікно параметрів CIC фільтра"

На рисунку 3.12 задаються параметри CIC фільтра, що дозволяє налаштувати його роботу для обробки сигналу. Фільтр налаштований у режимі Decimator, що зменшує частоту дискретизації сигналу, з коефіцієнтом зменшення Factor = 2. Значення диференціальної затримки встановлено на 1, що визначає мінімальний інтервал між відліками для обчислення різниці.

Частотні параметри вказані у нормалізованих одиницях (0–1), з частотою пропускання 0.1, яка визначає смугу частот, що проходитиме через фільтр. У розділі амплітудних характеристик задається ослаблення у смузі затримки на рівні 60 дБ.

Фільтр реалізований з можливістю використання базових елементів для кастомізації. Сигнали обробляються у форматі Elements as channels (sample based), де кожен елемент сигналу розглядається як окремий канал. Також увімкнено опцію Allow multirate processing, яка дає змогу обробляти сигнали з різною частотою дискретизації. Ці параметри забезпечують високу гнучкість і точність обробки сигналів.

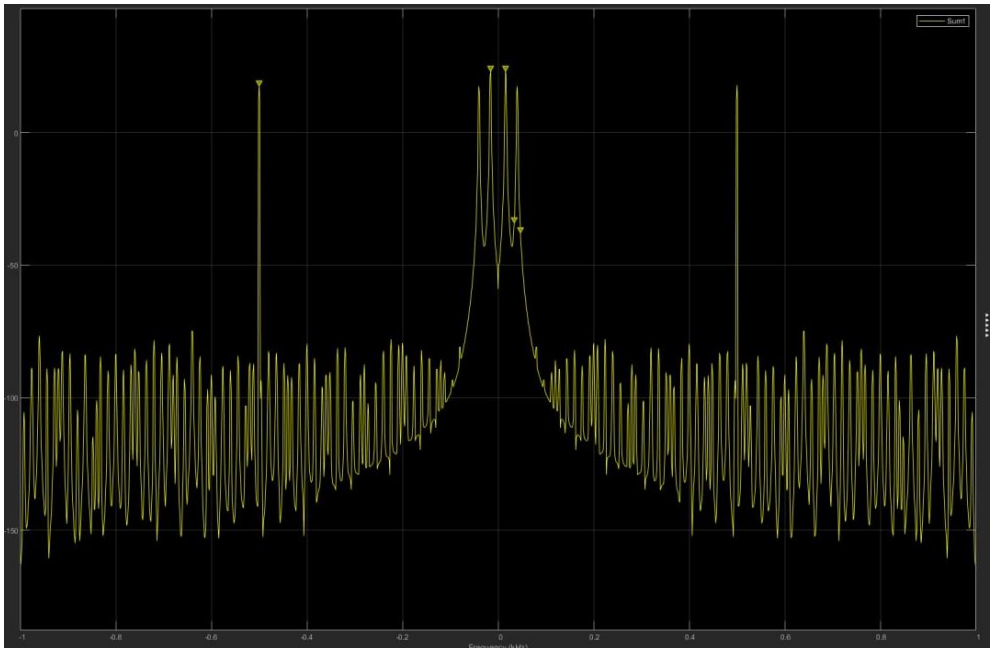

Рисунок 3.13 – Спектр вхідного сигналу

Після запуску моделі були отримані 2 графіки. В вікні Spectrum analyzer, що наведено на рисунку 3.13, можна побачити 3 основні піки вхідного сигналу, що подається на фільтрацію.

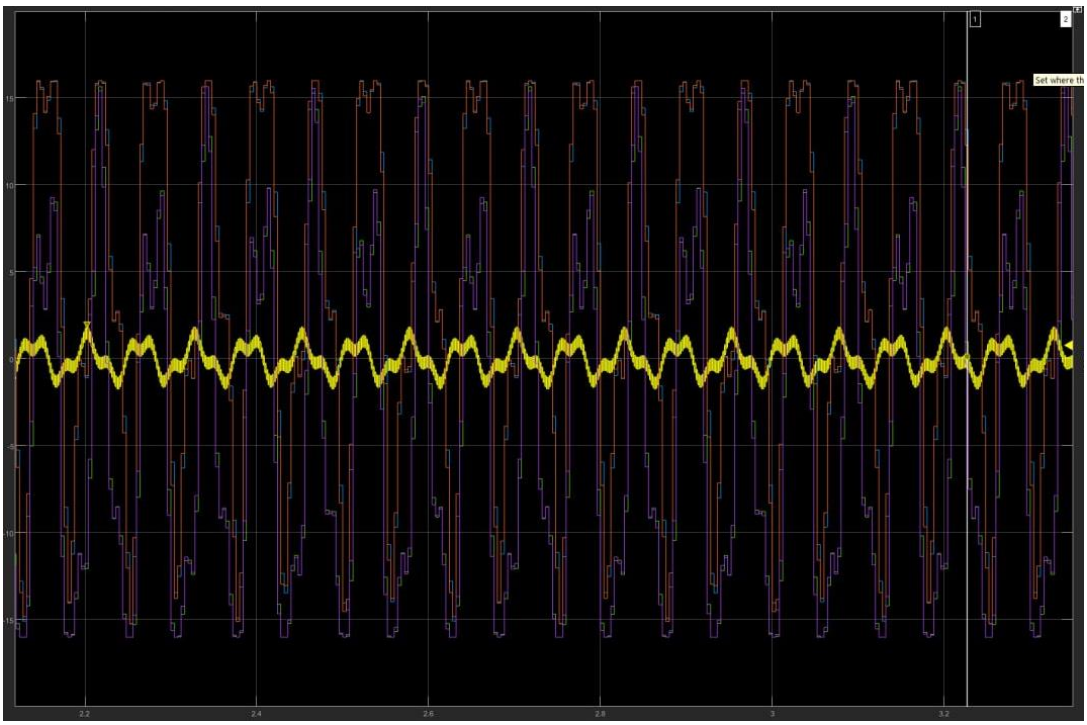

Рисунок 3.14 – Waveform сигналів до та після CIC фільтру

На графіку 3.14 наведені вхідний та вихідний сигнали до та після проходження CIC фільтру. Жовтим позначено вихідний сигнал, що містить три синусоїди з різними частотами. Результат фільтрації (інші лінії). Після децимації (див. рис.3.14) сигнал стає менш деталізованим, оскільки його частота дискретизації знижена. Тим не менш, основний спектр сигналу (низькі частоти, що визначають форму синусоїд), зберігається, а високочастотні шуми придушуються.

На графіках видно, що лінії після фільтрації стають більш згладженими та очищеними порівняно з вихідним сигналом. Після децимації вдвічі зменшується кількість точок даних у сигналі, що відповідає поставленій задачі. CIC-фільтр компенсує можливе порушення спектра, викликане перевідображення частот (аліасингом), за рахунок вбудованого згладжування.

4 Реалізація CIC фільтра

4.1 Quartus II

Quartus II – це професійне середовище розробки цифрових схем, створене компанією Altera (зараз Intel), яке широко використовується для проектування, моделювання, компіляції та програмування FPGA та CPLD, зокрема таких платформ, як DE0-Nano. Це комплексний інструмент, який дозволяє реалізовувати складні логічні конструкції, починаючи від написання коду і закінчуючи програмуванням апаратури.[2]

Quartus II надає можливість працювати з мовами опису апаратури, такими як Verilog і VHDL, а також створювати логіку у вигляді блок-схем, що робить його гнучким для розробників з різним рівнем підготовки. Після написання коду чи побудови схеми програма виконує синтез, під час якого HDL-код перетворюється в апаратну реалізацію, використовуючи логічні блоки FPGA, такі як LUT та тригери. Результатом синтезу стає конфігураційний файл, який можна завантажити в DE0-Nano.

Для налагодження проекту Quartus II підтримує функції моделювання, що дозволяють перевірити правильність роботи схеми ще до її завантаження в ПЛІС. Наприклад, якщо ви реалізували CIC-фільтр, ви могли б змоделювати, як він обробляє вхідні сигнали, і переконатися, що результати відповідають очікуваним. Це значно знижує ризик помилок на етапі програмування апаратури.

Після успішного моделювання середовище дозволяє призначити фізичні контакти FPGA для сигналів проекту, наприклад, для підключення вхідного сигналу до фільтра або передачі оброблених даних на зовнішній пристрій. Завантаження готового конфігураційного файлу в FPGA здійснюється через JTAG-інтерфейс за допомогою вбудованого програматора USB Blaster.

Для реалізації CIC-фільтра Quartus II надає все необхідне: від моделювання сигналів до створення апаратної конфігурації, що дозволяє ефективно обробляти сигнали в реальному часі. Завдяки такому програмному середовищу ви змогли успішно реалізувати фільтр на DE0-Nano, забезпечивши високу точність і продуктивність його роботи.

4.2 Моделювання CIC фільтра

Практичне моделювання є ключовим етапом у створенні цифрових систем, оскільки дозволяє перевірити роботу розробленого алгоритму або архітектури до її фізичної реалізації. У рамках цієї роботи було реалізовано та протестовано CIC фільтр відповідно до технічного завдання.

Для створення моделі було обрано мову опису апаратури Verilog, яка дозволяє реалізовувати цифрові системи на рівні регістрів і логіки. Це рішення обумовлене її поширеністю та можливістю синтезу на FPGA чи ASIC. Для перевірки правильності роботи розробленого модуля використано симуляцію в середовищі ModelSim, яке надає гнучкі інструменти для верифікації проєктів. Verilog описання фільтру наведено в лістингу 4.1.

Лістинг 4.1 – Реалізація CIC фільтра на Verilog

```
module signal_decimator #(parameter WIDTH = 16, DECIMATION_FACTOR = 4)(
    input clk,
    input reset,
    input [WIDTH-1:0] in_data,
    input valid_in,
    output reg [WIDTH-1:0] out_data,
    output reg valid_out);
    reg [31:0] counter; // Counter for decimation
    reg [WIDTH-1:0] integrator; // Single-stage integrator
    reg [WIDTH-1:0] comb; // Single-stage comb filter
    reg [WIDTH-1:0] comb_delay; // Delay register for comb filter
    always @(posedge clk or posedge reset) begin
        if (reset) begin
            counter <= 0;
            integrator <= 0;
            comb <= 0;
            comb_delay <= 0;
```

```
            out_data <= 0;
            valid_out <= 0;
        end else if (valid_in) begin
            // Integrator stage
            integrator <= integrator + in_data;
            if (counter == DECIMATION_FACTOR - 1) begin
                counter <= 0;
                // Comb stage
                comb <= integrator - comb_delay;
                comb_delay <= integrator;
                out_data <= comb;
                valid_out <= 1;
            end else begin
                counter <= counter + 1;
                valid_out <= 0;
            end
        end
    end
endmodule
```

У цьому модулі реалізується однокаскадний CIC-дециматор, який знижує частоту дискретизації вхідного сигналу за допомогою інтегратора, гребеневого фільтра (comb-фільтра) та механізму децимації. Інтегратор накопичує вхідний сигнал із кроком, визначеним тактовими імпульсами, що дозволяє усереднити сигнал і зменшити високочастотні компоненти. Гребеневий фільтр обчислює різницю між поточним значенням інтегратора та його попереднім значенням, видаляючи низькочастотні компоненти, які не є корисними. Механізм децимації зберігає тільки кожен заданий відлік інтегрованого сигналу, знижуючи частоту дискретизації вхідного сигналу.

Модуль параметризований двома змінними. Параметр WIDTH визначає розрядність сигналу, що дозволяє адаптувати модуль до різних форматів даних. Параметр DECIMATION_FACTOR задає коефіцієнт децимації, який визначає, наскільки знижується частота дискретизації сигналу. Регістри модуля виконують основні функції фільтрації. Лічильник counter рахує тактові імпульси до досягнення коефіцієнта децимації, після чого дозволяється передача відліку через comb-фільтр. Регістри integrator і comb виконують функції інтегратора та гребеневого фільтра відповідно. Регістри comb_delay зберігають попередні значення інтегратора для обчислення різниці.

Коли надходить сигнал reset, усі регістри скидаються до початкових значень, що гарантує коректну ініціалізацію модуля. У робочому режимі, якщо вхідні дані valid_in є дійсними, значення вхідного сигналу додається до регістра інтегратора. Лічильник counter слідкує за кількістю тактових імпульсів. Коли його значення досягає DECIMATION_FACTOR - 1, відбувається передача відліку через comb-фільтр на вихід. У протилежному випадку вихідний сигнал valid_out стає недійсним.

Цей модуль забезпечує ефективну децимацію вхідного сигналу із збереженням його корисної низькочастотної інформації. Його реалізація підходить для цифрових систем обробки сигналів, де потрібно знизити частоту дискретизації без втрати важливих характеристик сигналу. Завдяки параметризації, модуль може легко адаптуватися до різних форматів вхідних даних і частот дискретизації, що робить його універсальним рішенням у задачах обробки сигналів.

Flow Status	Successful - Sun Jan 26 11:18:11 2025
Quartus Prime Version	18.0.0 Build 614 04/24/2018 SJ Standard Edition
Revision Name	signal_decimator
Top-level Entity Name	signal_decimator
Family	Cyclone IV E
Device	EP4CE15F17C6
Timing Models	Final
Total logic elements	109
Total registers	97
Total pins	36
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0
Total PLLs	0

Рисунок 4.1 – Результат аналізу і синтезу

В результаті було отримано успішний синтез проекту під назвою signal_decimator, розробленого для FPGA Cyclone IV E (модель пристрою EP4CE15F17C6).

У процесі синтезу було використано 109 логічних елементів, які є основними компонентами для реалізації цифрових схем. Було задіяно 97 тригерів для реалізації регістрів і 36 виводів (пінів) пристрою для забезпечення взаємодії з зовнішніми сигналами. При цьому жодного біта внутрішньої пам’яті FPGA не було витрачено, як і вбудованих 9-бітових множників та фазових автопідстроювачів частоти (PLLs).

Такий результат свідчить про компактність і ефективність проекту, що не потребує значних апаратних ресурсів і підходить для реалізації в умовах обмежених можливостей FPGA. Крім цього були отримані додаткові результати синтезу

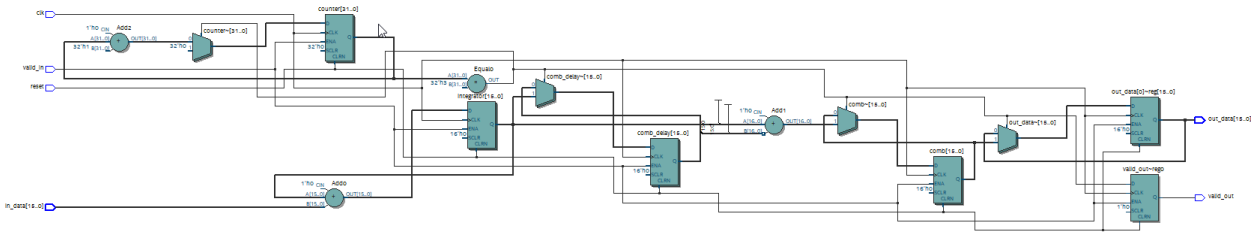

Рисунок 4.2 – Схема апаратної реалізації CIC фільтра

На рисунку 4.2 показано апаратну реалізацію модуля signal_decimator після синтезу в Quartus Prime. Вхідні сигнали проходять через інтегратор, який накопичує дані, і децимаційний блок, керований лічильником. Результат інтеграції потрапляє до комбінаторної ланки, яка виконує операцію віднімання з попередньо збереженим значенням (затримка). На виході дані зберігаються у регістрі, який синхронізується тактовим сигналом, і передаються у сигнал REGOUT. Схема оптимізована для виконання функцій CIC-фільтра із мінімальним використанням ресурсів.

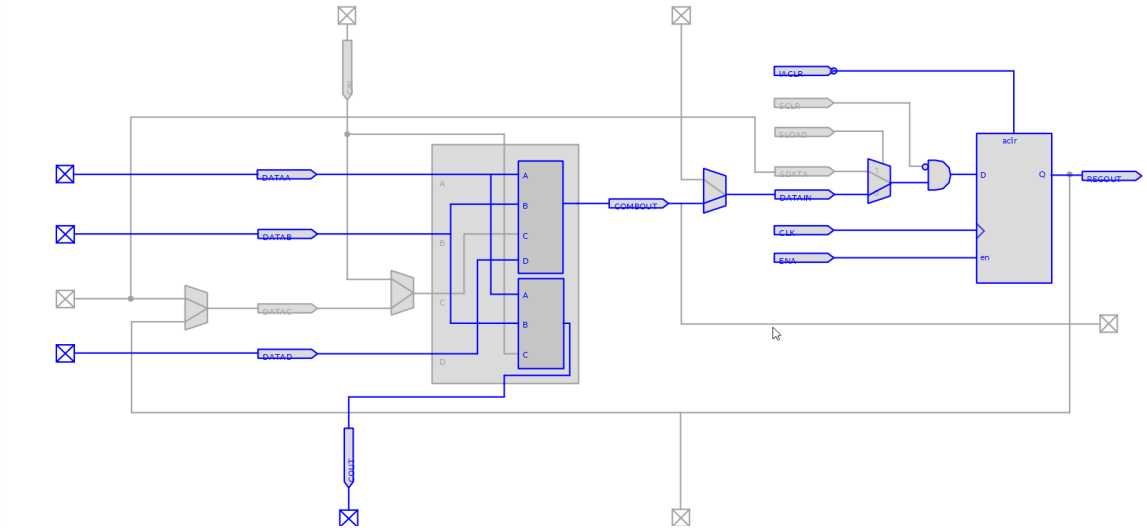

Рисунок 4.3 – діаграма ресурсів FPGA

На представленій діаграмі (рис 4.3) відображено апаратну реалізацію одного з компонентів лістингу 4.1. Вхідні сигнали (DATAA, DATAB, DATAC, DATAD) обробляються через мультиплексори та комбінаторну логіку, формуючи вихід COMBOUT, який надходить на вхід D-регістра. Регістр синхронізується сигналом CLK, а його робота керується сигналом ENA. Передбачено асинхронний скидаючий сигнал IACLR, який дозволяє встановлювати початковий стан. На виході компонента формується сигнал REGOUT, що є результатом обробки даних. Уся схема компактно інтегрована та оптимізована для роботи на FPGA

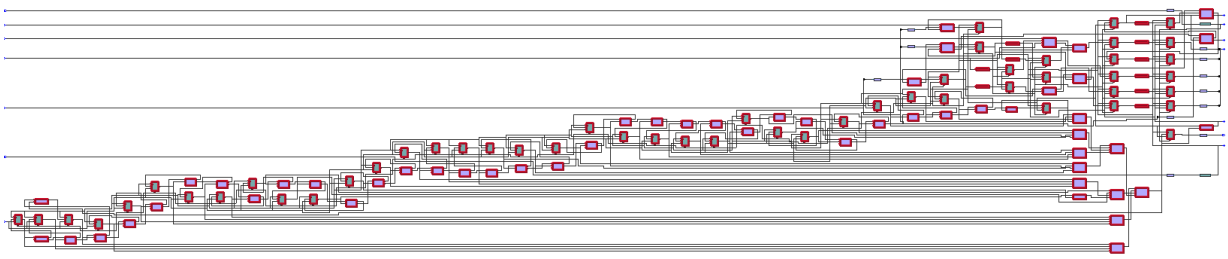

Рисунок 4.4 – Розподіл логіки, регістрів і з'єднань після синтезу.

Було отримано технологічну мапу реалізації компонента signal_decimator. Це детальне відображення того, як логічні елементи (комбінаційна логіка та регістри) і їх з'єднання розміщуються та оптимізуються для роботи на конкретній FPGA. Червоні прямокутники представляють регістри та комбінаційні логічні блоки, а чорні лінії – це з'єднання між ними.

Схема демонструє розподіл логічних елементів, які реалізують декілька рівнів обробки сигналів: інтеграцію, комбінаційну обробку та затримки (comb delays). Вхідні дані (in_data[0..15]) проходять через каскад логіки з використанням інтеграторів та затримок для формування вихідного сигналу. Також видно, що між блоками існують складні зв’язки, які забезпечують правильний порядок обробки

Лістинг 4.2 – Тестбенч для перевірки модуля CIC-дециматора

```
module tb_signal_decimator;
    parameter WIDTH = 16;
    parameter DECIMATION_FACTOR = 2;
    reg clk;
    reg reset;
    reg [WIDTH-1:0] in_data;
    reg valid_in;
    wire [WIDTH-1:0] out_data;
    wire valid_out;
    signal_decimator #(WIDTH, DECIMATION_FACTOR) dut (
        .clk(clk),
        .reset(reset),
        .in_data(in_data),
        .valid_in(valid_in),
        .out_data(out_data),
        .valid_out(valid_out));
    // Clock generation
    always #5 clk = ~clk;
    initial begin
        $dumpfile("signal_decimator.vcd");
        $dumpvars(0, tb_signal_decimator);
        // Initialize inputs
        clk = 0;
        reset = 1;
        in_data = 0;
        valid_in = 0;
        // Release reset
        #20 reset = 0;
```

```
        // Generate input signal
        repeat (2000) begin // Increased iterations to scale runtime
            valid_in = 1;
            in_data = $random % 256; // Random 8-bit signal
            #10;
        end
        valid_in = 0;
        #100;
        $finish;
    end
    // Monitor output
    always @(posedge clk) begin
        if (valid_out) begin
            $display("Time: %0t | Output Data: %d", $time, out_data);
        end
    end
endmodule
```

У тестбенчі (лістинг 4.2) реалізується перевірка роботи модуля signal_decimator, який знижує частоту дискретизації вхідного сигналу. Генерується випадковий сигнал із заданою шириною, що імітує реальні умови роботи модуля. Для кожного вхідного відліку відстежується відповідний вихідний сигнал, який формується після децимації. Завдяки використанню параметрів WIDTH та DECIMATION_FACTOR, тестбенч дозволяє легко адаптувати тест до різних конфігурацій і потреб, забезпечуючи універсальність тестування.

Основною метою тесту є перевірка роботи інтегратора, гребеневого фільтра та лічильника для виконання децимації. Важливим є те, що вивід сигналу здійснюється тільки після виконання повного циклу децимації, тобто коли лічильник досягає значення DECIMATION_FACTOR - 1. Це гарантує, що вихідний сигнал зберігає правильну структуру і відповідність до вхідного сигналу з меншою частотою дискретизації.

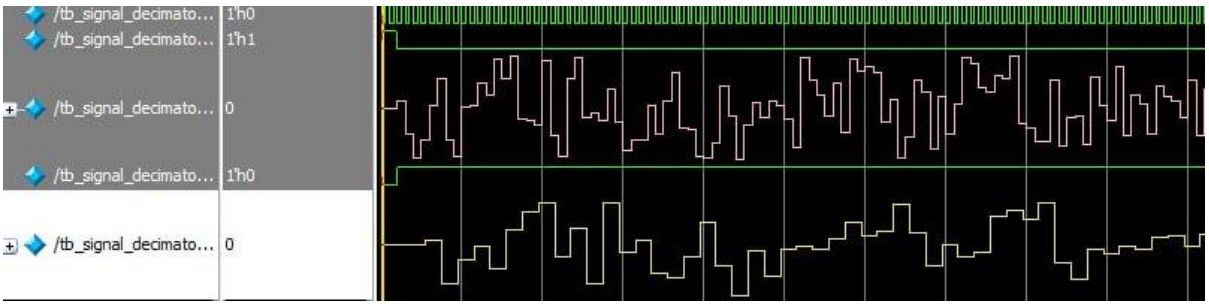

Рисунок 4.5 – Результати роботи тестбенча CIC-дециматора (waveform)

На часовій діаграмі представлено результати роботи модуля signal_decimator у тестових умовах. Верхній графік відображає вхідний сигнал signal in, який має високу частоту дискретизації. Нижній графік демонструє вихідний сигнал Decimated signal out, отриманий після зниження частоти дискретизації. Модуль зберігає кожен DECIMATION_FACTOR-й відлік вхідного сигналу, що видно на рис. 4.5. Логічний сигнал valid out синхронізується з моментами, коли вихідні дані є дійсними, що підтверджує коректну роботу модуля.

Цей результат ілюструє, як децимація дозволяє зберігати основну структуру сигналу з меншою кількістю відліків, забезпечуючи ефективне використання ресурсів. Видно, що вихідний сигнал залишається чітким і зберігає форму вхідного сигналу, при цьому видаляючи зайві високочастотні компоненти, що відповідає очікуванням

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було проаналізовано CIC-фільтри, їх структури та особливості роботи. Проведено порівняльний аналіз CIC-фільтрів із іншими типами цифрових фільтрів, визначено їх переваги, недоліки та сфери ефективного застосування. Наведено теоретичні основи децимації та інтерполяції сигналів, а також вплив параметрів фільтра на його характеристики.

В якості технологічної платформи обрано програмовану логіку FPGA на основі мікросхеми Cyclone IV E, реалізованої на платі DE0-Nano. Моделювання фільтрів було виконано за допомогою MATLAB/Simulink та Quartus II. Верифікація проводилася у середовищі ModelSim.

Сформульована послідовність етапів проектування CIC-фільтрів для FPGA, включаючи моделювання, оптимізацію параметрів і синтез апаратури. Практична реалізація виконана з використанням Verilog та інструментів Quartus II. Реалізовано однокаскадний CIC-дециматор із варіативним налаштуванням коефіцієнтів децимації та порядку фільтра.

Для перевірки ефективності роботи CIC-фільтра було проведено тестування на основі випадкових сигналів і синусоїд. Вхідні дані зберігали низькочастотну інформацію з мінімальними спотвореннями. Синтезований фільтр відзначився компактністю реалізації, використанням мінімальної кількості логічних елементів і високою швидкістю обробки сигналів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) Методи згладжування сигналів за допомогою однорідних нерекурсивних фільтрів. *Current directions of development of information and communication technologies and control tools*, 25–26 April 2024. 2024. P. 76. URL: https://doi.org/10.32620/ict.24.t1

2) Introduction to intel® quartus® prime standard edition. *Intel*. URL: https://www.intel.com/content/www/us/en/docs/programmable/683475/19-4/introduction-to.html

3) Cascaded integrator comb (CIC) filters - A staircase of DSP | wireless pi. *Wireless Pi*. URL: https://wirelesspi.com/cascaded-integrator-comb-cic-filters-a-staircase-of-dsp/

4) Hogenauer E. An economical class of digital filters for decimation and interpolation. *IEEE transactions on acoustics, speech, and signal processing*. 1981. Vol. 29, no. 2. P. 155–162. URL: https://doi.org/10.1109/tassp.1981.1163535

5) Lyons R. G. Understanding digital signal processing. Pearson Education, Limited, 2004. 688 p.

6) Smith S. Digital signal processing: a practical guide for engineers and scientists. Elsevier Science & Technology Books, 2013. 672 p.

7) Smith S. W. The scientist and engineer's guide to digital signal processing. 2nd ed. San Diego, Calif : California Technical Pub., 1999. 650 p.

8) Teymourzadeh R., Othman M. VLSI Implementation of Cascaded Integrator comb filters for DSP applications. *arXiv.org*. URL: https://arxiv.org/abs/1808.09369