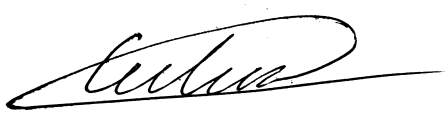
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ “ЛЬВІВСЬКА ПОЛІТЕХНІКА”

Борецький Тарас Романович



УДК 004.272:004.315.7:004.42

РОЗРОБКА ТА РЕАЛІЗАЦІЯ МЕТОДІВ ОБЧИСЛЕННЯ ЕЛЕМЕНТАРНИХ ФУНКЦІЙ НА ОСНОВІ ПРОГРАМНИХ ТА АПАРАТНИХ ЗАСОБІВ

05.13.05 – комп’ютерні системи та компоненти

Автореферат
дисертації на здобуття наукового ступеня
кандидата технічних наук

Львів - 2019

Дисертацією є рукопис.

Робота виконана у Національному університеті “Львівська політехніка” Міністерства освіти і науки України.

Науковий керівник: доктор технічних наук, доцент
Мороз Леонід Васильович,
професор кафедри безпеки інформаційних технологій
Національного університету “Львівська політехніка”.

Офіційні опоненти: доктор технічних наук, доцент
Яцків Василь Васильович,
завідувач кафедри кібербезпеки
Тернопільського національного
економічного університету, м. Тернопіль;

кандидат технічних наук
Грига Володимир Михайлович,
доцент кафедри комп’ютерної інженерії та електроніки
Прикарпатського національного університету
імені Василя Стефаника, м. Івано-Франківськ.

Захист відбудеться 29 березня 2019 р. о 14 годині на засіданні спеціалізованої вченої ради Д 35.052.08 у Національному університеті “Львівська політехніка” (79013, Львів-13, вул. С.Бандери, 28а, ауд. 711, V навчального корпусу).

З дисертацією можна ознайомитися у бібліотеці Національного університету “Львівська політехніка” (79013, Львів, вул. Професорська, 1).

Автореферат розісланий 28 лютого 2019 р.

Учений секретар
спеціалізованої вченої ради,
доктор технічних наук, професор  Луцик Я.Т.

ЗАГАЛЬНА ХАРАКТЕРИСТИКА РОБОТИ

Актуальність теми. Дисертаційна робота присвячена удосконаленню методів обчислення елементарних функцій та їх подальшому впровадженню у складі високотехнологічних програмно-апаратних засобів. Незважаючи на суттєві напрацювання стосовно проектування та реалізації методів обчислення елементарних функцій, в даному напрямі залишається багато невирішених питань — таких, як пошук оптимальних алгоритмів обчислення функцій та нових підходів до їх проектування, оптимізації структур існуючих моделей та покращення їх характеристик. Високий рівень зацікавлення до галузі підвищення ефективності обчислень функцій та послідовностей привів до появи різноманітних підходів до проектування та синтезу алгоритмів і схем, що постійно вдосконалюються.

При проектуванні та побудові більшості цифрових пристроїв широкого а також спеціального призначення для використання в різних галузях науки і техніки часто необхідно досягти оптимального співвідношення показників ефективності та ресурсозатратності. Частковим випадком даної проблеми служить задача максимально можливого покращення певної характеристики методу без обмежень на підвищення складності реалізації схеми в цілому. Причому тенденція щодо покращення методів обчислень прослідковується як на програмному так і на апаратному рівнях, що дозволяє говорити про неоптимальність існуючих підходів для розв’язання даної проблеми.

Необхідним завданням на сьогоднішній день є проведення аналізу переваг та недоліків особливостей функціонування сучасних засобів обчислень та їх компонентів, вивчення особливостей архітектури та структури а також пропонованих методів їх удосконалення в той час, як розвиток потужностей обчислювальної техніки надає додаткові можливості для застосування доволі складних та ресурсовитратних методів. Це призводить до збільшення складності обчислювальної системи і відповідно зростання енергозатрат, тоді як в загальному випадку застосування удосконалених методів обчислень забезпечує високу продуктивність функціонування системи в цілому. Тому актуальним завданням є розроблення і розвиток нових методів, які забезпечували б якісні результати на шляху підвищення ефективності роботи сучасної техніки, що є одним із перспективних напрямків її подальшого розвитку. Таким чином, науково-прикладне завдання підвищення продуктивності та ефективності функціонування програмних та апаратних комплексів є актуальним, що зумовлює необхідність розробки комплексу нових методів та засобів обчислень елементарних функцій. Крім того, важливо дослідити та розробити варіанти імплементації цих методів на різних обчислювальних платформах, забезпечити високу швидкодію їх функціонування, а також можливість вбудовування розроблених обчислювальних компонентів у широкий спектр електронних засобів.

Зв’язок роботи з науковими програмами, планами, темами. Тема дисертаційної роботи відповідає одному з наукових напрямів кафедри безпеки інформаційних технологій Національного університету «Львівська політехніка» “Розробка та реалізація методів обчислення елементарних функцій на основі програмних та апаратних засобів”. Результати досліджень, відображені у дисертаційній роботі, отримані у межах виконання науково-дослідних робіт кафедри безпеки інформаційних технологій Національного університету «Львівська

політехніка» за темою «Розробка та вдосконалення ітераційних методів обчислення елементарних функцій для систем захисту інформації» № державної реєстрації 0110U004687 та “Використання ітераційного методу CORDIC у системах розпізнавання відбитків пальців” № державної реєстрації 0114U001234.

Мета і завдання дослідження. Метою дисертаційної роботи є підвищення ефективності функціонування цифрових пристроїв як шляхом оптимізації вже існуючої програмної та апаратної бази, так і розробкою нових способів підвищення якості обчислювального процесу. Задля досягнення поставленої мети завдання дослідження були наступні:

- Проаналізувати сучасний стан та шляхи удосконалення відомих методів та засобів обчислень, аналіз їх особливостей, недоліків та переваг у порівнянні з існуючими аналогами.
- Здійснити комп’ютерне імітаційне моделювання функціонування розроблених методів обчислення елементарних, тригонометричних та трансцендентних функцій.
- Розробити структурно-функціональну модель представлених методів.
- Здійснити програмну реалізацію розроблених і вдосконалених методів. Визначити особливості її функціонування на платформах із різною архітектурою та системою команд.
- Дослідити способи ефективного синтезу та імплементації розроблених моделей та алгоритмів. Вивчити можливість розпаралелення обчислень та її вплив на функціонування схеми в цілому.
- Розробити апаратну реалізацію пропонованих пристроїв для платформи ПЛІС з різною функціональністю та архітектурою залежно від виробника.
- Порівняти з точки зору ефективності запропоновані рішення та відомі підходи щодо реалізації пропонованих функцій.

Об’єкт дослідження — дискретні процеси у цифрових схемах та пристроях.

Предмет дослідження — способи та методи удосконалення обчислень математичних функцій у засобах обробки інформації.

Методи дослідження – методи і апаратно-програмні засоби комп’ютерних систем, методи наближення функцій, теорія похибок і непевності результатів вимірювань, теорія ймовірності, математичної статистики, лінійної алгебри, методи цифрового опрацювання сигналів, методи імітаційного моделювання, чисельні методи, методи обробки даних.

Наукова новизна отриманих результатів. За результатами проведених теоретичних та експериментальних досліджень а також їх практичної реалізації розв’язано ряд важливих науково-технічних проблем, щодо підвищення ефективності функціонування програмних та апаратних засобів, а також компонентів комп’ютерних систем. При цьому отримано такі наукові та практичні результати:

вперше:

- запропоновано метод інверсного повороту (вектора), який функціонує паралельно з алгоритмом одностороннього повороту і за рахунок повороту вектора лише в сторону зменшення кута дає змогу вдвічі скоротити кількість ітерацій методу при будь-якому вхідному значенні аргумента.
- запропоновано спосіб паралельної поліноміальної інтерполяції, який за рахунок оптимізації вибору коефіцієнтів та паралельного використання помножувачів

дає змогу скоротити кількість ітерацій та час обчислення широкого спектру функцій.

удосконалено:

- метод перекодування кута, який дає змогу скоротити кількість необхідних для обчислень ресурсів шляхом зміни вхідного значення аргумента з наступним представленням та оперуванням лише додатними величинами, що забезпечує коректну роботу алгоритму для беззнакової логіки та спрощує проектування та апаратні затрати при описі схем мовами вищого рівня.
- метод кусково-нелінійної апроксимації, в якому збільшення розрядності та відповідно точності обчислень відбувається за рахунок використання квадраторів, де розрахунок вищих порядків залишкового кута відбувається без використання додаткових коефіцієнтів та зміни розрядності аргументів у діапазоні обчислень.
- метод паралельної поліноміальної інтерполяції із використанням таблиці попередньої вибірки (ТПВ), що дає змогу зменшувати кількість поліномів та множень за рахунок використання ПЗП без обмежень на мінімальний об'єм таблиці, а при невеликих розрядностях аргументів використовувати лише поліноми першого порядку.

Практичне значення одержаних результатів.

- вперше на базі x86 платформи на мовах C та асемблера реалізовано метод перекодування кута із спрощеним конвеєром обчислень, що дає змогу реалізації метода CORDIC із меншою кількістю апаратних ресурсів. Виявлено сильні та слабкі сторони методу.
- адаптовано та реалізовано метод перекодування кута — беззнакове перекодування для платформи ПЛІС "Cyclone" мовою System Verilog, в якому всі аргументи приймають лише додатні величини, чим спрощують реалізацію алгоритму та необхідні апаратні ресурси із збереженням переваг підходу перекодування.
- за допомогою способу представлення розрядів при програмній реалізації методу одностороннього та інверсного повороту використана можливість наперед визначити майбутній хід ітераційного процесу, чим зробити залежності в середині алгоритму менш критичними та більш прогнозованими, що дозволило покращити швидкодію.
- вперше, використовуючи широкосмугові помножувачі ПЛІС реалізовано метод квадратичної апроксимації, який дає можливість з мінімальною латентністю обчислювати тригонометричні функції та спрощувати ітерації методу CORDIC.
- на базі мікроконтролера AVR мовою асемблера розроблено генератор псевдовипадкових чисел у вигляді зовнішнього інтерфейсу для ПК, із можливістю перекладення частини ресурсів процесора на даний периферійний пристрій.
- отримані результати по енергоспоживанню та ресурсоємності пропонованих методів. Практично перевірено коректність їхнього функціонування та визначено перспективи їхнього подальшого впровадження.
- окремі положення дисертаційного дослідження використовувались під час виконання науково-дослідних робіт кафедри безпеки інформаційних технологій Національного університету «Львівська політехніка», а також впроваджені у навчальний процес кафедри у таких дисциплінах як "Комп'ютерні методи високорівневого проектування систем захисту" та "Комп'ютерні методи аналізу та проектування електронних засобів".

Особистий внесок здобувача. Усі наукові результати дисертаційної роботи отримані автором самостійно. У працях, опублікованих у співавторстві, автору

належать: розроблення та опрацювання теоретичних основ, отримання математичних співвідношень [1,2,5]; формулювання ідей удосконалення алгоритму та визначення оптимальних шляхів їх реалізації [1,2,4]; організація методів досліджень, розрахунків та проведення експериментів [1,3,4,5,6]; дослідження характеристик пропонованих алгоритмів [1,4,5]; розроблення структури, проектування блок-схем та алгоритмів функціонування програмно-апаратних засобів [1,5,6]; написання програмного забезпечення, та його оптимізація [1-6]; замір швидкодії, точності обчислень, [1,3-6], а також латентності та споживаної потужності [4-6]; запропоновано метод інверсного повороту та перекодування кута [4,5]; вдосконалення методу одностороннього повороту [2,6]; удосконалення класичного методу CORDIC із покращеною латентністю обчислень [1,4,6]; аналіз впливу різної розрядності імплементованих функцій на продуктивність роботи ПЛІС [4,5]; аналіз роботи гібридного методу із застосуванням запропонованого адаптивного алгоритму [3,5]; групування, аналіз та представлення результатів дослідження [1,3,4,5].

Апробація результатів дисертації. Про основні результати наукових досліджень автор доповідав на таких конференціях: V Міжнародна науково-технічна конференція "Захист інформації і безпека інформаційних систем" (Львів, 2-3 червня 2016р.) I Міжнародна науково-технічна конференція "Інформаційна безпека в сучасному суспільстві" (Львів, 21-22 листопада 2014р.) 69-та студентська науково-технічна конференція секції кафедр «Захист інформації» та «Безпека інформаційних технологій» (Львів, 17-18 жовтня 2011р.) Матеріали дисертації неодноразово були оприлюднені та обговорені на наукових семінарах кафедри безпеки інформаційних технологій та кафедри захисту інформації Національного університету «Львівська політехніка».

Достовірність отриманих результатів. Для перевірки достовірності результатів представлених у дисертації було проведено ряд експериментальних досліджень під час яких одержано:

- показники швидкодії пропонованих алгоритмів при використанні декількох видів апаратних платформ різних виробників з відмінною розрядністю, системою команд та функціональністю.
- дані щодо точності обчислення реалізованих методів представлених функцій у всьому діапазоні вхідних значень та показники відхилень від очікуваних величин. Результати випробувань свідчать про коректність як програмної реалізації розроблених методів, так і їхніх апаратних аналогів. Результати моделювання у достатній мірі корелюють і з результатами теоретичних досліджень, що відображено у дисертаційній роботі.

Публікації. Основні положення та результати дисертаційного дослідження викладено в шести друкованих працях, серед них одна стаття у науковому періодичному виданні іноземної держави, який включено до міжнародної наукометричної бази Scopus; п'ять статей у наукових фахових виданнях України з технічних наук та три публікації за матеріалами наукових конференцій.

Структура та обсяг дисертації. Дисертаційна робота складається із переліку умовних позначень, вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. Загальний обсяг роботи складає 232 сторінки, з яких 216 сторінок основного тексту, що містять 36 рисунків та 42 таблиці. Список використаних джерел налічує 136 найменувань.

ОСНОВНИЙ ЗМІСТ РОБОТИ

У вступі обґрунтовано актуальність вибраного напряму досліджень, показано зв'язок роботи з науковими програмами, планами, темами, сформульовано мету та основні задачі досліджень, подано наукову новизну і практичне значення отриманих результатів, визначено особистий внесок здобувача, наведено дані про апробацію, публікації за темою роботи та використання результатів дослідження.

У першому розділі розглянуто відомі на сьогоднішній день способи обчислення елементарних функцій. Серед них обчислення експоненти, логарифму, квадратного кореня, синуса, косинуса та їхніх гіперболічних аналогів. Враховуючи, що наближення даних функцій можна здійснювати різними способами, далі приводиться стислий огляд поширених методів їх обчислення в залежно від складності заданої функції. Розглядаються способи поліноміального наближення функцій за допомогою схеми Горнера, поліномом Тейлора, поліномом нормального рівномірного наближення та ін. Оскільки апроксимація поліномом не завжди дає хороші результати, розглядаються також способи дробово-раціональної апроксимації, наближення ланцюговим дробом та ітераційні методи. Значну увагу приділено методу CORDIC, який дозволяє наближувати більшість вищезгаданих функцій. Розділ завершується описом середовищ та засобів розробки, використовуваних у дослідженні.

У **другому розділі** показано, що для здійснення обчислень елементарних функцій, таких як синус, косинус, тангенс, квадратний корінь чи експонента на апаратному та частково на програмному рівні значна частина виробників напівпровідникової техніки використовує ітераційні методи, такі як CORDIC.

При використанні класичних методів, точність обчислень залежить від розрядності операндів та кількості здійснених ітерацій. При невеликих кількостях ітерацій всі вхідні та вихідні значення можна задавати у вигляді таблиці, розміщеної в пам'яті. Враховуючи, що в переважній більшості пристроїв доступним є деякий об'єм пам'яті, хоч і незначний, його можна використати для прискорення методу CORDIC. Причому ефект буде відчутним і у випадку, якщо об'єм виділеної пам'яті становить сотні чи навіть десятки байт (оперативної чи флеш пам'яті у випадку МК). Автором пропонується новий метод знакозмінного перекодування вхідного аргументу, який дає змогу гнучко змінювати об'єм пам'яті таблиці та число ітерацій. Для класичного методу CORDIC основним недоліком є низька швидкодія через лінійну збіжність методу (не більше одного опрацьованого розряду вхідного аргумента за одну ітерацію) та відносна апаратна складність, пов'язана з необхідністю реалізації одночасно трьох ітераційних рівнянь (для x_i, y_i, z_i) у випадку застосування конвеєрної структури обчислювача:

$$x_i = x_{i-1} - \sigma_i 2^{-i} y_{i-1} \quad y_i = y_{i-1} + \sigma_i 2^{-i} x_{i-1} \quad z_i = z_{i-1} - \sigma_i \arctan\left(2^{-i}\right) \quad \sigma_i = sign\left(z_{i-1}\right), i = 1 \ldots m, \qquad (1)$$

де m - число двійкових розрядів обчислювача.

З метою спрощення апаратної реалізації обчислювача пропонується використання методу CORDIC із знакозмінним перекодуванням, що дає змогу звести систему (1) лише до двох ітераційних рівнянь для x_i, та y_i.

$$x_i = x_{i-1} - b_i y_{i-1} \cdot 2^{-i-1}; \quad y_i = y_{i-1} - b_i x_{i-1} \cdot 2^{-i-1}; , \; b_i = 2a_i - 1 , \; i = \overline{1,m} \qquad (2)$$

Одночасно з цим для підвищення швидкодії методу, шляхом зменшення числа ітерацій, розроблено гібридні структури, що використовують послідовно три методи:

табличний + CORDIC + залишкове множення. Найпростішим з точки зору апаратного втілення є CORDIC з класичним перекодуванням вхідного аргументу. Однак він має суттєвий недолік – великий об'єм пам'яті переглядових таблиць, і при великих значеннях m - необхідна таблиця розміром, не меншим, ніж $2^{m/3} \cdot m$ розрядів. Крім того, вихідні помножувачі в класичному варіанті реалізовані у базисі {-1,1}, що ускладнює використання помножувачів для беззнакової логіки. Тоді як пропонований метод перекодування дає змогу при необхідності здійснювати представлення величин так, що вони завжди додатні або рівні нулю.

Таблиця 1

Співставлення значень ТПВ для класичного способу і пропонованого методу знакозмінного перекодування

m=16	m=24	m=32	m=40	m=48	m=54	m=64
i_a=5	i_a=7	i_a=10	i_a=13	i_a=15	i_a=17	i_a=21
i_b=2	i_b=3	i_b=5	i_b=6	i_b=7	i_b=8	i_b=10

де i_a - мінімальне значення кількості біт, при якому можливе класичне перекодування

i_b - номери вхідних розрядів пропонованого методу, для яких похибка при обчисленнях кусково-лінійною апроксимацією, не вийде за межі розрядної сітки.

Як випливає з поданої таблиці, запропонований метод значно розширює межі значень $m_{ТПВ}$ у порівнянні з відомими результатами робіт. Причому, якщо для класичного перекодування кількість розрядів таблиці є жорстко фіксованою, то вказані в таблиці 1 діапазони можна додатково зменшити за рахунок помножувачів. Наприклад для ПЛІС, де блок помножувача здатний функціонувати в режимі 18x18, було скорочено розмір таблиці з 2^4 до 2^3 при m рівним 32-м розрядам за рахунок розширення діапазону корекції залишкового множення, яке за замовчуванням становить 17 розрядів.

Описаний метод знакозмінного перекодування підходить також і для обчислення гіперболічних синуса, косинуса, тангенса та експоненти. Також варто відзначити можливість прогнозування ітерацій, яка відсутня у класичному методі та його модифікаціях. Завдяки цьому частину операцій можна визначити відразу після одержання вхідного аргументу, що при апаратній реалізації дозволяє краще розмістити логічні елементи у схемі. Маючи інформацію про напрям повороту вектора, який є наперед відомим, не виникає потреби у здійсненні зміни шляху подальшого функціонування алгоритму, що позитивно позначається на процесі обчислень.

Далі запропоновано метод інверсного повороту, який функціонує паралельно з алгоритмом одностороннього повороту та дає змогу вдвічі зменшити кількість ітерацій методу при будь-якому вхідному значенні аргумента. Інверсний поворот вектора базується на використанні модифікованого метода Рунге-Кутта третього порядку. Оригінальний метод Рунге-Кутта має вигляд:

$$\begin{bmatrix} x_{n+1} \\ y_{n+1} \end{bmatrix} = \begin{bmatrix} 1-\frac{\varepsilon^2}{2} & \frac{\varepsilon^3}{6}-\varepsilon \\ \varepsilon-\frac{\varepsilon^3}{6} & 1-\frac{\varepsilon^2}{2} \end{bmatrix} \begin{bmatrix} x_n \\ y_n \end{bmatrix}, \text{де } \varepsilon = 2^{-i} \qquad (3)$$

де для операції ділення на шість використовується бінарний зсув операнда на три розряди в сторону молодших бітів (ділення на вісім). Отримане таким чином спрощення апаратної реалізації має незначний вплив на точність результатів, і, як виявила практика, збільшення відхилення все ж лежить в тих самих межах розрядної

сітки, для якої необхідно забезпечити точний результат після округлення. В цьому методі ротація вектора здійснюється тільки в одну сторону — в сторону зменшення кута, пропускаючи в процесі обчислень одиничні розряди вхідного кута. Це означає, що ТПВ повинна містити значення синусів та косинусів без врахування коефіцієнта деформації, а їхнє значення буде коригуватись в процесі повороту вектора шляхом здійснення додаткових операцій згідно з матрицею перетворень (3). В кінцевому варіанті рівняння для обчислень методом інверсного повороту мають вигляд:

$$x_{i+1}=x_i\cdot\left(1-\frac{\varepsilon^2}{2}\right)-y_i\cdot\left(\frac{\varepsilon-\varepsilon^3}{8}\right)\quad y_{i+1}=y_i\cdot\left(1-\frac{\varepsilon^2}{2}\right)-x_i\cdot\left(\frac{\varepsilon-\varepsilon^3}{8}\right)\quad \varepsilon=2^{-i} \tag{4}$$

У випадку, коли $2^{-3(i-1)}<2^{-n}$, розрядність операції виходить за межі обчислень частину коефіцієнтів можна упустити. Отримана спрощена формула для обчислень:

$$x_{i+1}=x_i\cdot\left(1-\frac{\varepsilon^2}{2}\right)-y_i\cdot\varepsilon\quad y_{i+1}=y_i\cdot\left(1-\frac{\varepsilon^2}{2}\right)-x_i\cdot\varepsilon \tag{5}$$

Незважаючи на вищу складність формул у порівнянні з класичним методом, основний виграш в швидкодії забезпечується за рахунок меншої кількості ітерацій.

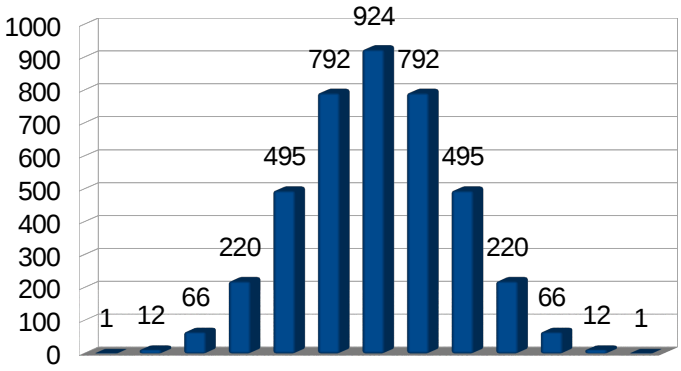


Рис. 1 Розподіл кількості вхідних комбінацій методу інверсного повороту вектора

При реалізації формул методом інверсного повороту для 32 розрядної платформи можна відвести під зберігання значень таблиці лише 4 розряди, для чого знадобиться 128 байт пам’яті. Враховуючи, що молодші шістнадцять розрядів обчислюються методом кусково-лінійної апроксимації, через що обчислення методом інверсного повороту відбувається для 32 - 4 - 16 = 12 розрядів. При цьому кількість випадків, при яких n — біт розрядів повороту дорівнює нулю (одиниці), подана на рис. 1. Якщо весь діапазон значень однотипних розрядів поділити на три частини, то 3,5 тисячі аргументів з 4 тисяч припадають на центральну третину, тобто в середньому сім з восьми аргументів. Таким чином ймовірність попадання значення в проміжок 4 - 8 кроків зі всього діапазону 0 - 12 кроків буде становити 85.4%, що і дає основний виграш алгоритму.

Метод інверсного повороту здатний забезпечити швидкодію на рівні метода одностороннього повороту, проте основною його перевагою є можливість сумісного з ним функціонування. Операції в такому випадку здійснюються лише якщо значення розрядів вхідного кута дорівнюють нулю/одиниці. Звичайно, що для кожного вхідного значення потрібно використовувати тільки один із методів: односторонній поворот або інверсний поворот. Вибір метода, яким буде виконуватись поворот, пропонується здійснювати паралельно, або зразу після приведення аргумента в перший квадрант у випадку послідовної програмної реалізації. При апаратній реалізації метод легко модифікується для можливості використання в обох варіантах. Так, напрям повороту (заміна знаку при обчисленнях) реалізується за допомогою інверторів на входах та виходах суматорів. Звичайно, поява інверторів та мультиплексорів для ввімкнення та вимкнення інверторів у схемі здатна дещо знизити швидкодію, проте загальний результат все ж

покращується через сумарне зменшення числа ітерацій, а при необхідності досягнення максимальної швидкодії можлива реалізація двох паралельних конвеєрів для обох методів. Можна помітити, що значення кутів φ_{LUT} , для яких шукається значення функції, практично не відрізняється від:

$$\phi_{lut} \approx \phi_{lut-1} + a_i 2^{-m(lut)} - a_i 2^{-m(lut+rot)}, \quad (6)$$

а різниця між сусідніми значеннями буде становити лише $2^{-i(lut+rot)}$. Таким чином, за рахунок незначної втрати точності можна використовувати одну таблицю для обох методів. Значення кута для методу інверсного повороту буде відповідати наступному значенню в ТПВ відносно того, яке б використовувалось для одностороннього повороту. А досягнення максимальної точності методу можливе шляхом використання двох таблиць, що є співрозмірним по затратах до здійснення однієї ітерації методом одностороннього повороту.

Подальше підвищення швидкості та точності обчислення функцій можливе за допомогою використання додаткових дій множення. Розвиток цього способу, відомого як — метод кусково-нелінійної апроксимації, дає змогу збільшити розрядність обчислень на 50%. Досягається це за рахунок квадратичної апроксимації при обчисленнях вищих порядків залишкового кута, і відбувається без використання додаткових коефіцієнтів та зміни розрядності аргументів у діапазоні обчислень. Операції методу квадратичної апроксимації здійснюються на основі залишкового кута $\varphi_{зал}$, який утворюється незалежно від способу, використаного для одержання цього значення. В методі квадратичної апроксимації одночасно із дією додавання, яка утворилась при залишковому множенні, відбувається повторне множення результату на кут $\varphi_{зал}$, та зсув результату на один розряд згідно з формулами:

$$x_m = x_{\frac{m}{3}} \cdot \left(1 - \frac{\phi_{зал}^2}{2}\right) - y_{\frac{m}{3}} \cdot \phi_{зал} \quad y_m = y_{\frac{m}{3}} \cdot \left(1 - \frac{\phi_{зал}^2}{2}\right) + x_{\frac{m}{3}} \cdot \phi_{зал} \quad (7)$$

Розрядність в цьому випадку збільшиться на третину у порівнянні з залишковим множенням, яке дає можливість подвоїти кількість вірних розрядів. Також можна підкреслити, що розрядність обчислень можна збільшувати і далі, додавши до формули множник у вигляді залишкового кута третього порядку:

$$x_m = x_{\frac{m}{4}} \cdot \left(1 - \frac{\phi_{зал}^2}{2}\right) - y_{\frac{m}{4}} \cdot \left(\phi_{зал} + \frac{\phi_{зал}^3}{8}\right) \quad y_m = y_{\frac{m}{4}} \cdot \left(1 - \frac{\phi_{зал}^2}{2}\right) + x_{\frac{m}{4}} \cdot \left(\phi_{зал} + \frac{\phi_{зал}^3}{8}\right) \quad (8)$$

Автором також запропонований метод паралельної поліноміально-квадратичної інтерполяції (ППКІ), який базується на факторизації поліномів найкращого рівномірного наближення. При апаратній реалізації алгоритмів, в тому числі на платформі ПЛІС, доступна можливість задіювання ресурсів значної кількості помножувачів для досягнення мінімальних затримок шляхом розпаралелювання операцій. Для досягнення високої ефективності методу пропонується обчислювати добутки поліноми найкращого рівномірного наближення за допомогою добутків підфункцій загального виду $x^2+k_1x+k_2$. Це дасть можливість шукати добуток kx одночасно із операцією піднесення до степеня, а операція додавання після операції множення є вбудованою у DSP блоки сучасних ПЛІС (множення з накопиченням). Незважаючи на появу коефіцієнта k_2 загальна кількість коефіцієнтів є меншою, оскільки всі вони є сталими для всього діапазону обчислень. В методі застосовано групування підфункцій таким чином, щоб забезпечити

мінімальну латентність методу шляхом застосування деревоподібної ієрархічної структури помножувачів (зв'язний неорієнтований граф), та з використанням поліномів лише другого порядку. Метод може легко бути як звужений, так і розширений залежно від необхідної точності обчислень, шляхом зміни кількості добутків підфункцій, кількість яких прямо пропорційна порядку полінома. Оптимальна ж кількість помножувачів для знаходження добутку всіх підфункцій є степенем двійки, адже в цьому випадку множення доцільно здійснювати за допомогою деревоподібної топології. Серед функцій які були реалізовані методом ППКІ є синус, косинус, тангенс, арктангенс, гіперболічні функції, а також експонента, квадратний корінь, ділення, та інверсний корінь. Для побудови оптимальної схеми методом ППКІ бажано наперед визначити необхідну точність обчислень, яку потрібно досягнути. Далі базуючись на даних таблиці вибирається поліном відповідного степеня, виходячи з кількості точних бінарних розрядів, які він здатний забезпечити. Якщо ж задача полягає у забезпеченні максимальної точності на деякій платформі, то основним показником для оптимізації швидкодії стає кількість помножувачів, які можуть працювати паралельно. Далі наведені у таблиці 2 дані точності та ресурсомісткості обчислення тригонометричних функцій для першої чверті декартової системи координат, адже точність обчислень суттєво залежить від величини вибраного діапазону.

Таблиця 2

Точність обчислень функцій методом ППКІ та необхідна кількість коефіцієнтів полінома

степінь полінома	синус			косинус			тангенс			арктангенс		
	x//kx//x^2	±ΔX	точн. біт	x//kx//x^2	±ΔX	точн. біт	x//kx//x^2	±ΔX	точн. біт	x//kx//x^2	±ΔX	точн. біт
5	1//2//2	$6{,}0\text{x}10^{-8}$	23,0	0//5//0	$1{,}4\text{x}10^{-7}$	21,7	1//2//2	$4{,}6\text{x}10^{-5}$	13,4	1//3//1	$2{,}1\text{x}10^{-5}$	14,5
8	1//5//2	$2{,}2\text{x}10^{-12}$	37,7	0//6//2	$9{,}1\text{x}10^{-13}$	39,0	1//4//3	$2{,}4\text{x}10^{-7}$	21,0	1//4//3	$1{,}9\text{x}10^{-7}$	21,3
13	-	-	-	-	-	-	1//6//6	$3{,}5\text{x}10^{-11}$	33,7	1//6//6	$2{,}9\text{x}10^{-11}$	34,0

При наближенні поліномами високих порядків найкращою точністю володіють функції синуса-косинуса, а також гіперболічний синус-косинус. Так, функція косинуса буде мати тридцять дев’ять точних розрядів при використанні полінома восьмої степені. Дещо меншу точність мають функції (у порядку спадання) експоненти, логарифму, квадратного кореня, інверсного кореня та ділення. Функції тангенса та арктангенса є одні з найменш ефективних функцій, які ще можна послідовно наближувати за допомогою степеневих поліномів. Так, з допомогою полінома тринадцятого порядку обчислюється будь-яка з перелічених вище функцій з точністю понад 32 двійкові розряди. Крім того, існує ряд функцій, які не вдається наближати за допомогою поліномів. До них, наприклад, належать арксинус та арккосинус. Порядок полінома практично не впливає на точність їх обчислення, а також не представляється можливим обчислити коефіцієнти поліномів 9-го та вищих порядків. Максимальну точність, яку вдалось досягнути для даних функцій є 4,5 двійкові розряди, що не представляє практичної цінності. Таким чином запропонований спосіб ППКІ дає змогу скоротити кількість ітерацій та час обчислення широкого спектру функцій за рахунок паралельного використання помножувачів. Цей спосіб найкраще підходить для систем, у яких доступно декілька незалежних помножувачів.

В **третьому розділі** розглянуті способи апаратної реалізації пропонованих методів обчислення для ПЛІС. Першим кроком при оптимізації алгоритмів тут

виступає можливість ідентифікації причин обмеження, які виникають при імплементації алгоритмів. Для цього потрібно мати змогу оперувати характеристиками базових елементів, іноді опускаючись до рівня вентильного рівня архітектури ПЛІС, хоча в більшості випадків достатньо оперувати більш високими примітивами, такими, як регістр, мультиплексор, суматор. Так, наприклад, для класичного алгоритму CORDIC, саме суматор якраз і є тим елементом, швидкодія якого обмежує загальну швидкодію алгоритму. В більш складних алгоритмах такими елементами можуть виявитись помножувачі, блоки пам'яті чи просто з'єднання між деякими апаратними блоками ПЛІС. Тому на початку розділу подані дослідження щодо швидкодії та ресурсоємності базових елементів ПЛІС у різних конфігураціях. Далі, оперуючи отриманими даними, можна достатньо об'єктивно дати відповідь на питання про фактори, які накладають обмеження на функціонування досліджуваних алгоритмів, оцінити їх потенціал та ресурсозатратність. Також якщо в кристалі відсутні або зайняті необхідні для реалізації алгоритму блоки, вони можуть бути реалізовані за допомогою функцій комбінаційної логіки кристалу, хоча і не оптимальним чином. Так, реалізація апаратного та комбінаційного блоку помножувача залежно від розрядності, а також кількість необхідних елементів для їх імплементації відображена в таблиці 3. Саме частота функціонування даного блоку виступає основним бар'єром на шляху підняття тактової частоти більшості алгоритмів, оскільки, як блоки пам'яті, так і логічні елементи кристалу здатні функціонувати на вищих тактових частотах. Щодо падіння швидкодії в програмних помножувачах, то її можна компенсувати за рахунок збільшення латентності.

Таблиця 3

Тактові частоти груп помножувачів при множенні знакових та беззнакових чисел

кількість помножув.	1x 9x9	1x 18x18	2x 18x18	3x 18x18	5x 18x18	10x 18x18	16x 18x18	56x 18x18	1x9x9 програм.	1x18x18 програм.	36x18x18 програм.
0°C (мгц)	353,11	327,23	250,06	236,46	233,37	235,57	222,72	211,69	170,10	123,56	112,89
85°C (мгц)	343,29	288,85	223,06	211,28	208,72	210,35	199,96	188,25	190,26	110,31	100,92

Реалізацію функцій додавання та віднімання в архітектурі ПЛІС виконують логічні елементи у спеціально передбаченому для такого випадку арифметичному режимі. Розрядність суматора в ПЛІС обмежена кількістю логічних елементів, розміщених в одному стовпці, і для вибраного кристалу становить 448 розрядів. Часто при збільшенні розрядності змінних в алгоритмі виникає падіння швидкодії саме на каскадах суматорів. Щоб оцінити степінь оптимізованості проекту, швидкодія якого впирається в швидкість роботи деякого суматора, рекомендовано використовувати таблицю з максимальними частотами суматорів для різних розрядностей (таблиця 4). За допомогою цієї таблиці можна визначити верхню межу швидкодії, до якої можливо покращувати алгоритм.

Таблиця 4

Швидкодія суматорів для аргументів різних розрядностей

розрядність	9	18	36	72	114	128	256	448
част. (мгц) 0°C	508,13	400	279,56	176,15	125,49	117,34	62,53	36,89
част. (мгц) 85°C	575,04	455,58	320,1	202,35	144,3	135,74	72,43	42,7
макс. част. (мгц)	817,0	702,74	487,33	303,67	216,4	201,2	106,98	63,06

Далі представлені результати реалізації пропонованих методів у зіставленні із класичними варіантами та бібліотечними мегафункціями від Intel та IPCore від Xilinx. Так, для запропонованої виробником класичної реалізації методу CORDIC в

середовищі Quartus, необхідно виділити близько третини ресурсів кристалу (таблиця 5). Розрядність вхідних та вихідних даних фіксована і становить тридцять два біти.

Таблиця 5

Реалізація синуса та косинуса за допомогою інтегрованої мегафункції

шина (біт)	к-ть тактів	пропуск. здатність (мбіт/с)	латентність (нс)	к-ть блоків	к-ть логіч. елем.	к-ть тригерів	розмір пам'яті (біт)	частота при 85°C (мгц)	частота при 0°C (мгц)	макс. частота (мгц)
32	36	4372.16	263.49	5248	4996	2350	1362	136.63	152.91	243.7
32	36	4562.88	252.47	5056	4768	2247	320	142.59	158.15	246.6

Для розширення спектру порівняння різних версій пропонованих функцій була розроблена власна реалізація класичного алгоритму CORDIC з можливістю оптимізувати його структуру та характеристики, виходячи з специфіки поставленої задачі. Розрядність та кількість ітерацій алгоритму можна змінювати з кроком в один розряд, забезпечуючи необхідну точність обчислень.

При реалізації класичного методу CORDIC, у кристалі ПЛІС використовувались лише логічні елементи, тоді як для інших методів необхідним буде використання додаткових блоків пам'яті та помножувачів. Табличний метод застосовується для старших розрядів операнду, кількість яких визначається доступною кількістю пам'яті. Для вибраного кристалу максимальна кількість ітерацій, які можна помістити в пам'ять, дорівнює дванадцяти (таблиця 6).

У свою чергу, метод кусково-лінійної апроксимації застосується лише для молодшої половини розрядів вхідного операнду, і при наявності в кристалі блоків із функцією множення стає можливим вдвічі зменшити кількість необхідних ітерацій, замінивши їх двома операціями множення та додавання. Швидкодія в цьому випадку буде обмежена частотою функціонування блоку помножувача та його розрядністю.

Таблиця 6

Табличний метод та кусково-лінійна апроксимація (мінімальна латентність).

розрядність (біт)	к-ть тактів	пропуск. здатність (мбіт/с)	латентність (нс)	к-ть блоків	к-ть логіч. елем.	к-ть тригерів	к-ть помнож.	розмір пам'яті (біт)	частота при 85°C (мгц)	частота при 0°C (мгц)	макс. частота (мгц)
14	4	2029	27.60	46	24	46	2	6144	144.91	160.98	269.8
14	56 x 4	110544	28.37	5317	3987	3620	112	334848	141.0	156.23	262.3
14	5	3126	22.39	62	26	68	2	6144	223.31	248.45	388.5
14	56 x 5	167164	23.45	5437	3735	3968	112	337920	213.22	236.46	370.0
24	4	3144	30.53	114	46	114	4	360448	131.01	146.13	241.0
24	5	4369	27.46	125	46	125	4	360448	182.05	203.62	314.4
24	6	6932	20.77	180	46	180	4	360448	288.85	327.12	493.1

Для розрядності в 14 біт існує можливість задіювання усіх помножувачів кристалу в конфігурації 9x9 біт. Всього таким чином можна створити 56 незалежних конвеєрів. Подальше збільшення розрядності призведе до зменшення кількості конвеєрів та використовуваних блоків помножувачів, що при значенні в 24 розряди досягне свого мінімуму з використанням лише двох помножувачів з конфігурацією 18x18. В цьому випадку критичним значенням виступає об'єм вбудованої пам'яті, для доступу до якої на максимальній швидкості доводиться збільшувати кількість тактів. Проте максимальна тактова частота може бути досягнута лише при попаданні зчитаних даних з пам'яті безпосередньо у регістри помножувача, тоді як у всіх

сімействах ПЛІС блоки пам’яті та помножувачі розміщені на певній відстані один від одного, що не дозволяє здійснити множення відразу після операції зчитування з пам’яті. Тому для підвищення тактової частоти необхідно розбити шлях від елементів пам’яті до блоків помножувачів на декілька розділених тригерами частин, при проходженні сигналу між якими відсутні будь-які логічні чи арифметичні операції. В отриманій моделі при зростанні латентності відсутні покращення у характеристиках алгоритму. Тому для ефективнішого використання ітерацій, здійснених на шляху до блоків помножувачів, запропоновано метод знакозмінного перекодування, який реалізований в двох варіантах: 16 та 38 розрядному. В першому випадку можна задіяти всі доступні помножувачі, при цьому використання блоків пам’яті становить менше 10% ресурсів кристалу. Варто також відзначити, що між операціями зчитування з пам’яті та блоками помножувачів здійснюються ітерації методу перекодування, що дає змогу використати “холості” такти для арифметичних ітерацій алгоритму, ефективність яких найкраще проявляється при великих об’ємах пам’яті. У випадку 16-ти розрядного алгоритму з 56-ма конвеєрами критичним параметром виступає кількість логічних елементів кристалу. Тоді як збільшення шини до 38 біт буде обмежуватись розрядністю блоку помножувача. Незважаючи на повну завантаженість блоків пам’яті, вони в даному випадку не є обмежуючим фактором, а їхня кількість може бути зменшена за рахунок збільшення латентності.

Таблиця 7

Результати знакового методу перекодування кута

шина (біт)	пам’ять/ поворот/ апкрокс.	кіль-кість тактів	пропускна здатність (мб/с)	латент-ність (нс)	к-ть викор. блоків	к-ть логіч. елем.	к-ть триг-герів	викор. пам’яті (біт)	частота при 85°С (мгц)	частота при 0°С (мгц)	макс. частота (мгц)
16	6/2/8	6	3868.5	28.95	228	138	217	832	241.78	266.31	435.73
16	56 x 6/2/8	56 x 6	208132.8	34.44	14952	7728	14560	46592	232.29	260.28	411.35
38	13/6/18	10	7069.9	53.75	1311	1073	793	488984	186.05	207.30	321.85

Модифікований метод знакозмінного перекодування — беззнакове перекодування, в якому всі аргументи приймають лише додатні величини, дозволяють спростити реалізацію алгоритму, та необхідні для цього апаратні ресурси із збереженням переваг підходу перекодування. Основним параметром, що демонструє переваги пропонованого алгоритму, є латентність, яка обернено-пропорційна тактовій частоті, та залежить від кількості тактів необхідних для обчислення функції. Кількість тактів, в свою чергу, може змінюватись за рахунок зміни розміру ТПВ. Порівняння результатів методу беззнакового перекодування для платформи Xilinx здійснено із пропонованою виробником імплементацією IP бібліотеки CORIDC. При порівнянні класичного алгоритму CORDIC із пропонованим алгоритмом слід зауважити, що метод кусково-лінійної апроксимації для платформи Artix-7 так само як і на платформі CycloneIII обмежується частотою блоку помножувача. Причому, враховуючи ефект збільшення частоти на невеликих розрядностях частота алгоритму буде значно перевищувати частоту блоку помножувача. Так, максимальна частота помножувача для Artix-7 становить 350±1МГц, за умови використання вбудованих в блок помножувача тригерів. Оскільки структура пропонованого алгоритму не передбачає використання даних тригерів, частота буде меншою від максимальної на декілька відсотків. Тоді як застосування кусково-лінійної апроксимації дозволяє замінити 12 класичних

ітерацій лише двома ітераціями з використанням ресурсів блоку помножувача, хоча і з меншою тактовою частотою. При збільшенні розрядності до 32 тактова частота дещо падає - до величини 334-340МГц, залежно від величини ТПВ. Прямо пропорційно до збільшення ТПВ зростає і об'єм необхідної для реалізації алгоритму пам'яті, проте знижується кількість логічних елементів через зменшення кількості ітерацій методу. Так, кількість тригерів та логічних елементів для 24-розрядного пропонованого методу знизилась від 4.5 до 5.5 раз, залежно від значень ТПВ.

Таблиця 8

Характеристики швидкодії, латентності, ресурсоємності та енергоспоживання алгоритму перекодування кута, у порівнянні із вбудованим CORDIC IP CORE для плаформи Artix-7

шина біт	ТПВ біт	к-ть тактів	латент-ність нс	чачтота мгц	найращий період(нс)	триг-гери	логіч. ел-ти	пам'ять ЛУТ	вводи-виводи	к-ть ДСП	потужн. (мвт)	т-ра (С°)
32	4	15	44.100	340.14	2.940	955	827	14	97	2	378	26.1
32	6	13	38.181	340.48	2.937	826	710	16	97	2	364	26.0
32	8	11	32.879	334.56	2.989	698	584	18	97	2	345	26.0
24	5	10	28.630	349.28	2.863	477	381	13	73	2	278	25.8
16	-	18	41.364	435.16	2.298	976	947	2	51	0	327	25.9
24	-	26	67.600	384.62	2.600	2161	2113	4	75	0	499	26.4
32	-	34	99.280	342.47	2.920	3588	3529	2	99	0	673	26.9
48	-	50	177.55	281.61	3.551	7834	7746	4	147	0	1052	28.0

Як видно з таблиці 9, для 32 розрядного пропонованого алгоритму латентність може становити від 40 до 64 нс залежно від розміру ТПВ. При значеннях таблиці 3 чи 4 розряди внутрішня пам'ять не задіюється, а пам'ять синтезується за допомогою логіки кристалу. При досягненні розміру таблиці в 10 розрядів необхідно мати доступними 64 кбіт пам'яті, що дає виграш як у латентності, так і в ресурсоємності алгоритму. Коливання частоти для різних конфігурацій становить менше ±4%. При 32 розрядній реалізації пропонованого алгоритму частоти лежать в межах 247.65 МГц (ТПВ = 3) — 276.4 МГц (ТПВ = 9).

Таблиця 9

Алгоритм 32-розрядного перекодування для плаформи Cyclone з різним розміром ТПВ

ТПВ	3	4	5	6	7	8	9	10
кількість тактів	17	16	15	15	14	13	12	11
латентність	64.38	60.19	57.16	55.23	54.71	50.43	47.68	40.48
використано блоків	1974	1852	1568	1441	1359	1214	1068	945
к-ть логічн. елементів	1609	1494	1298	1165	1034	905	778	670
кількість триггерів	1520	1444	1224	1131	1039	948	851	763
викор.пам'яті (біт)	0	0	2048	4096	8192	16384	32768	65536
частота при 85°(мгц)	233.05	237.47	233.92	241.6	225.58	229.25	225.43	240.21
частота при 0°(мгц)	264.06	265.82	262.4	271.59	255.89	257.8	251.7	271.74
макс. частота (мгц)	410.34	397.30	393.55	405.19	382.12	395.41	394.79	407.66

Для 32 розрядного методу залежно від значення ТПВ кількість елементів знизилась від 3,5 до 6 разів. Виграш було досягнуто за допомогою використання двох блоків помножувача та логічних елементів у режимі імітації пам'яті, оскільки для досягнення максимальної швидкодії компілятор не використовував апаратні блоки пам'яті через її незначний об'єм.Споживана потужність при використанні пропонованого методу знизилась майже вдвічі у порівнянні з класичним варіантом алгоритму. Основний виграш у зменшенні потужності відіграло динамічне споживання кристалу, яке прямо пропорційне кількості задіяних елементів логіки та

кількості сигнальних ліній. Енергоспоживання блоків помножувача, які використовуються у пропонованому методі, виявилось незначним і становить 2-3% від загальної потужності кристалу.

Таблиця 10

Зіставлення методів перекодування та власної реалізації класичного алгоритму

	шина (біт)	к-ть такти	латент. (нс)	к-ть блоків	к-ть лог. елемент.	к-ть триггерів	частота 85°C мгц	частота 0°C мгц	частота макс.мгц	пропуск. здат.мб/с
1 перекод. 3T	32	17	64,38	1974	1609	1520	233,05	264,06	410,34	7457,6
2 перекод.10T	32	11	40,48	945	670	763	240,21	271,74	407,66	7686,72
3 перекод. 13T	38	10	53,75	1311	1073	793	186,05	207,3	321,85	7069,9
4 класич.власн.	32	32	150,82	2889	2775	2011	212,18	237,08	364,3	6790
5 класич.уточн.	40	32	161,09	3889	3749	2525	198,65	224,01	337,9	7946
порівняння 1-4	17 - 32		57,31%	31,67%	42,02%	24,42%	9,84%	11,38%	12,64%	9,83%
порівняння 2-4	11 - 32		73,16%	67,29%	75,86%	62,06%	13,21%	14,62%	11,90%	13,21%
порівняння 3-5	10 - 32		66,63%	66,29%	71,38%	68,59%	-6,34%	-7,46%	-4,75%	11,03%

Пропоновані алгоритми знакозмінного та беззнакового перекодування дають змогу спростити реалізацію класичного методу CORDIC у його апаратній реалізації, а також отримати виграш у порівнянні із вбудованими бібліотечними функціями. У випадку використання платформи Xilinx, виграш в латентності пропонованих алгоритмів збільшується більш як у 2 рази, при втраті тактової частоти у декілька мегагерц. Шляхом ширшого використання пам’яті та помножувачів вдається отримати економію як ресурсів кристалу, так і споживаної потужності від 2 до 7 раз.

У **четвертому розділі** здійснено програмну реалізацію пропонованих методів на базі x86 персонального комп’ютера та мікроконтролера AVR. Враховуючи, що у

Таблиця 11

Величина відхилень пропонованих методів на прикладі 32-розрядного синуса та косинуса представлені в одиницях молодшого розряду

	класичний cordic (sin)	класичний cordic (cos)	куск-лінійн. апрокс.(sin)	куск-лінійн. апрокс(cos)	односторон. поворот(sin)	односторон. поворот(cos)	інверсний поворот(cos)	інверсний поворот(sin)	квадратичн. апрокс.(sin)	квадратичн. апрокс.(cos)
0	348262335	345989658	468810087	469470145	476881962	478950496	224917494	382037038	289684222	265926222
1	668701693	663974630	872120658	874918637	886614949	887806538	536165351	754037524	617109497	519126222
2	591546456	586612563	706086237	709235061	711122255	714884303	668969219	709182754	657240250	483061702
3	481862292	477364994	493051809	493513578	492763118	493080431	693470464	597069812	624426794	428358361
4	360937838	358064030	286912558	285872934	278362244	276759124	544306255	429769649	502166306	361977451
5	248276282	247932751	136237335	134518654	127427206	126046667	349997434	260581422	335270922	291510802
6	156786113	158862985	52398202	50327696	46309076	44189214	193587272	137371305	188904175	223703576
7	90916432	94503306	16179228	14915924	13296618	12029541	94670419	61816699	93097254	163578449
8	48404133	52396977	4054859	3352051	3447890	2614302	41334412	25416401	39838027	113949249
9	23642344	27213624	754292	507599	427924	310628	17633147	9038761	15183517	75646697
10	10581938	13296955	70792	43817	22854	4852	4594179	3650721	5663341	47756654
11	4341496	6126487	39	-	-	-	1916988	1324902	2167150	28691963
12	1630107	2653988	-	-	-	-	488646	596354	930239	16345741
13	556146	1076361	-	-	-	-	169136	209747	332619	8813259
14	170942	405174	-	-	-	-	-	66698	138382	4482485
15	46080	141039	-	-	-	-	-	16362	27343	2145344
16	10951	44348	-	-	-	-	-	1498	7609	959612
17	2134	12575	-	-	-	-	-	-	-	401906
18	345	2952	-	-	-	-	-	-	-	157071
19	37	604	-	-	-	-	-	-	-	56455
20	2	88	-	-	-	-	-	-	-	-
21	-	7	-	-	-	-	-	-	-	-

сучасних процесорах підтримується апаратне обчислення тригонометричних функцій з точністю в 64 бінарні розряди, стає можливим проведення програмного порівняння і аналізу результатів роботи створених алгоритмів, яке недоступне для більшості платформ. Таким чином з'являється можливість давати об'єктивну оцінку кожній із кінцевих та альтернативних підверсій алгоритмів і відповідного їм програмного коду швидко та точно. Порівняння відхилень алгоритму CORDIC, та його модифікацій зведено в таблицю 11. Відхилення подано по модулю різниці між точним та одержаним значенням в одиницях молодшого розряду.Деякі методи не мають свого відображення в таблиці, оскільки напряму не впливають на процес обчислень (для прикладу табличний метод). Таким чином відсікається їхній вплив на точність обчислень, що дає змогу оптимізовувати параметри алгоритму, не боячись збільшити похибку кінцевих результатів.

Переважна більшість систем, для яких призначаються розроблені алгоритми, є цілочисельними. Така тенденція пояснюється кращою швидкодією цілочисельної логіки, її простішою апаратною реалізацією та більшою універсальністю. Для більшості функцій, при використанні метода CORDIC, вхідні та вихідні значення лежать в межах першого квадранту, чи не перевищують одиниці. Таким чином цілочисельна мантиса для зберігання вхідних значень функції має наступний вигляд:

x_0 .	x_1	x_2	x_3	.....	x_{n-4}	x_{n-3}	x_{n-2}	x_{n-1}

Наступна складність полягає у точному практичному визначенні кількості тактів деякої частини коду для x86-ї платформи. Проте існує ряд напрацювань в цьому напрямі, які дали змогу визначати достатньо довгі періоди часу (до 1 години $\approx 10^{13}$ тактів) з точністю до одного машинного такту. Щоправда, даний спосіб підходить далеко не для всіх архітектур ЦП. Тому проводити заміри усіх досліджуваних алгоритмів слід лише на одній, наперед вибраній платформі, яка забезпечує мінімальні показники флуктуації результатів. Як видно з рисунку 2 кількість тактів суттєво відрізняється в залежності від платформи, що насамперед можна пояснити складністю процесорної архітектури. Тому для порівняння була вибрана платформа AMD K7, при тестуванні на якій спостерігалась найкраща кучність отримуваних результатів.

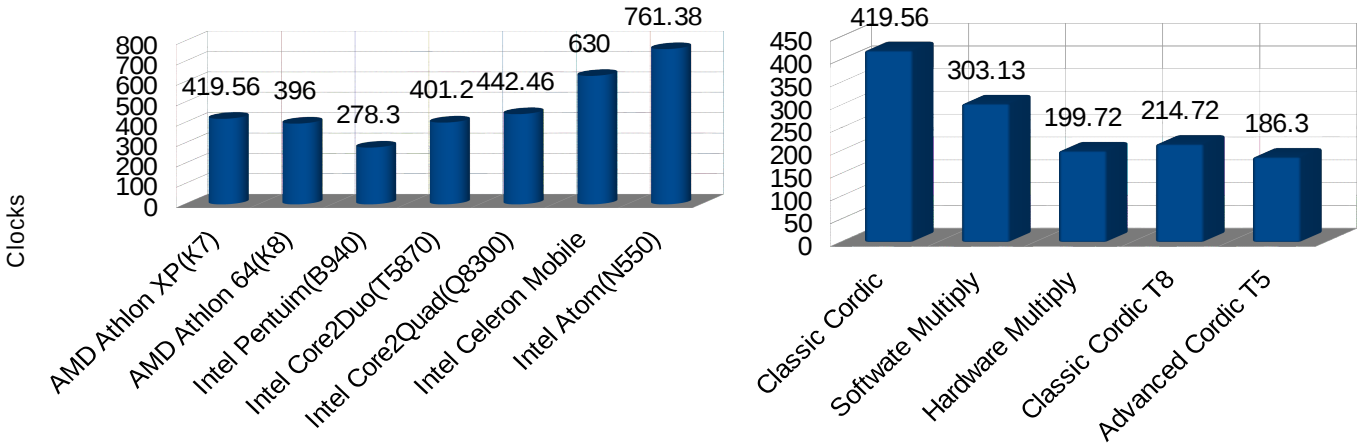


Рис. 2 Порівняння швидкодії різних модифікацій алгоритму CORDIC

Метод Add-Shift (Softwate multiply) розроблений як альтернативна версія кусково-лінійної апроксимації (Hardware Multiply), без використання апаратних помножувачів, і призначається для мікроконтролерів, які не підтримують операцій

множення. Застосування методу інверсного повороту (Advanced Cordic) проходить у два етапи. Другий етап можна розглядати як частковий випадок першого, коли розрядність операндів вищих порядків стають меншими за розрядність обчислювальної мантиси, що дало змогу пропустити частину ітерацій і прискорити виконання алгоритму не вплинувши на точність обчислень. Для 32-розрядного методу перший етап буде тривати п’ять ітерацій, другий – шість. Оскільки вхідними даними алгоритму виступає двійкове представлення деякого числа, значення розрядів якого буде поетапно проаналізоване, то можна зообразити етапи виконуваних операцій на прикладі розрядів артумента:

табличний метод					перший етап інверс-ного повороту вектора					другий етап інверсного повороту вектора						кусково-лінійна апроксимація					
x_0	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_8	x_9	x_{10}	x_{11}	x_{12}	x_{13}	x_{14}	x_{15}	x_{16}	x_{17}	x_{18}	…	x_{30}	x_{31}

Рис. 3 Схема поділу розрядів вхідного кута між застосовуваними методами

Пропонований метод також реалізовано для платформи AVR, арсенал команд якої містить лише логічні та арифметичні інструкції. Виконання мультиплікативних операцій множення та ділення вибраний МК (ATTiny2313) не підтримує. До переваг МК слід віднести можливість виконання більшості операцій за один такт, роботу на частоті 20MHz, а також низьку ціну МК.

Мікроконтролер функціонує у ролі генератора ПВП, виконаного у вигляді зовнішнього периферійного пристрою, комунікація з яким відбувається по інтерфейсу IEEE1284. Згідно з протоколом, МК повинен самостійно проводити усі розрахунки алгоритму, а в момент одержання результатів викликати за допомогою переривання ПК. Завдяки такому підходу досягається розвантаженість процесора комп’ютера організацією комунікації, а дані зчитуються лише в момент виникнення переривання, ініційований контролером. Особливістю реалізованого алгоритму генерації ПВП є те, що він не використовує оперативну пам’ять МК та стеку, який розміщується в ОП, та EEPROM пам’ять (використання якої не рекомендована виробником на максимальних частотах). При цьому всі регістри є зайняті динамічними параметрами програми під час виконання основного циклу, що дає 100% використання регістрового файлу МК. Метод інверсного повороту, який лежить в основі обчислень, реалізованих у МК, був випробуваний у двох різних комбінаціях із п’ятикроковою та шестикроковою ТПВ. Усі попередні тести для даного алгоритму, в тому числі для x86 платформи, здійснювались при п’ятикроковій таблиці виходячи з міркувань, що для вбудованих систем з обмеженими ресурсами доступно в середньому 2^8 байт постійної пам’яті. У випадку використання мікроконтолера, після написання та відлагодження коду алгоритму виявилось можливим використати шестикрокову таблицю, тобто запас вільної статичної пам’яті становив 2^9 байта. Відповідно за основу для порівнянь взято шестикроковий варіант коду, із зазначенням його відмінностей від п’ятикрокового варіанту.

Таблиця 12

Кількість тактів необхідних алгоритму залежно від значення біт поточного аргументу

Номер біту кута	0...4(5)	5	6	7	8	9	10	11	12	13	14	15	16...23	24...31
Одиничне значення	26	3	3	3	3	3	3	3	3	3	3	3	10 x 8	9 x 8
Нульове значення	(28)	74	62	46	41	49	45	49	45	51	41	31	7 x 8	6 x 8

У МК відсутні засоби моніторингу, наприклад, для визначення точної кількості виконаних тактів, на відміну від x86 платформи, на якій досліджувались характери

різних методів CORDICa. Це саме стосується і програмної симуляції роботи МК, яка у кілька разів повільніша за апаратну реалізацію. Тому аналіз швидкодії проведено виходячи з умови, що ймовірність одержання всіх можливих комбінацій вхідних розрядів є однакова. Основний параметр, який оптимізовувався в першу чергу при написанні алгоритмів – досягнення максимальної швидкодії. Тому алгоритм для МК написаний у вигляді “розгорнутого” коду, де всі альтернативи циклів та процедур замінені аналогами макрокоманд, які здатні максимально точно передати дії методу. Такий підхід вимагає наявності більшої кількості пам’яті для коду програми, але, без сумніву, має вищу швидкодію. Крім того кожна така макрооперація може бути оптимізована для обчислення тих статичних параметрів функції, для яких вона буде виконуватись. В цьому випадку кожна така макрооперація містить унікальну частину коду і є оптимзованою за часом її виконання. Також хорошим способом збільшити швидкодію та зменшити довжину коду є використання ТПВ більшого розміру. Проте максимальний розмір не може перевищувати половини пам’яті МК (одного кілобайту з двох наявних), що в представленому випадку може прискорити лише один крок методу, а саме, для розряду номер шість.

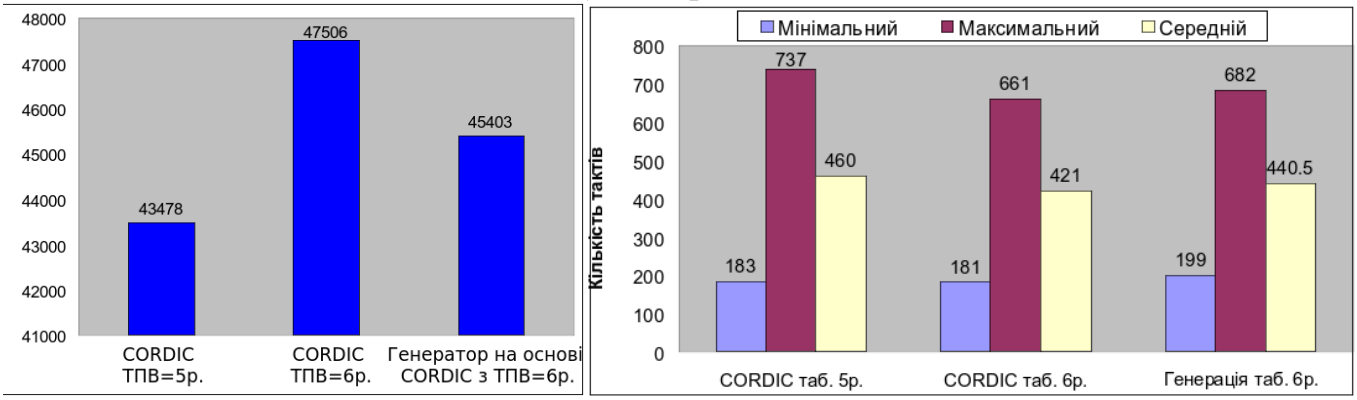


Рис. 4 Середня кількість обчислених значень за секунду, та порівняння швидкодії модифікацій алгоритму залежно від розміру ТПВ

Розроблений алгоритм оптимізовано для досягнення максимальної швидкодії та найбільш повного використання ресурсів мікроконтролера. Організовано комунікацію між МК та ПК використовуючи IEEE 1284 та режим “Byte Mode”, для двосторонньої передачі даних, що дає змогу використовувати МК як додатковий периферійний пристрій комп’ютера та перекласти на нього частину задач. Отриманий у підсумку генератор ПВП функціонує на частотах, що відповідають стандарту IEC 60908, і, при потребі, можна підключити його до ЦАП зі шиною у 32 розряди. Таким чином реалізовані алгоритми обчислення більшості тригонометричних функцій як для платформи з обмеженими можливостями (AVR), так і для x86 мікропроцесорів із використанням цілочисельної логіки та доступного широкого набору команд. Даний факт дозволяє стверджувати, що запропоновані методи будуть працювати на більшості сучасних платформ.

ВИСНОВКИ

У дисертаційній роботі на основі здійснених теоретичних та експериментальних досліджень розв’язано актуальну науково-прикладну задачу в галузі створення нових ефективних способів обчислення елементарних функцій. Основні теоретичні та експериментальні дослідження, представлені в роботі, можна узагальнити такими висновками:

1. Розроблено нові методи, технології та апаратно-програмні засоби, які в ході експериментальних досліджень підтвердили коректність постановки задач і математичних методів, які були використані при їх розв'язанні. Серед таких методів можна виділити метод ППКІ, який підходить для платформ з великою кількістю незалежних помножувачів. Якщо ж їх кількість обмежена чи емулюється програмно, можна використовувати метод квадратичної апроксимації чи одностороннього та інверсного повороту. Для економії ресурсів кристалу також корисним буде метод перекодування кута.

2. Здійснено комп'ютерне симуляційне моделювання функціонування розроблених методів обчислення елементарних функцій. Результати моделювання показали, що запропоновані методи дають змогу знизити латентність, ресурсоємкість та збільшити швидкодію обчислень. Також вдалось покращити точність розрахунків, знизивши прохибки методу CORDIC з 20 до 10 (чи 11, залежно від наявності корекції) одиниць молодшого розряду.

3. Розроблено пристрій генерації псевдовипадкових чисел на основі тригонометричних функцій в кристалах з обмеженими можливостями. Практично продемонстрована можливість використання такого підходу для бюджетних платформ без втрати якості в процесі їх експлуатації. Робочий прототип пристрою здатний генерувати до 2.9 мбіт/с корисної інформації при частоті в 20МГц.

4. Здійснено програмну реалізацію пропонованих методів як на процесорних архітектурах з високою розрядністю та широким набором команд, так і для бюджетних мікроконтролерів, з урахуванням специфіки функціонування даних алгоритмів на кожній із них. Враховуючи, що апаратне забезпечення відіграє суттєву роль у функціонуванні програмної частини, найкраща за стабільністю платформа AMD K7 продемонструвала приріст швидкодії у 2.25 рази в порівнянні з класичними методами обчислення функцій.

5. Проведене оцінювання апаратних затрат при використанні різних сімейств ПЛІС для реалізації пропонованих методів. Також одержані параметри енергоефективності використання того чи іншого методу обчислень залежно від сфери його подальшого впровадження, основним параметером значення якого було – латентність, де виграш становив до 55% - 65% від початкового значення (залежно від розрядності). У той же час падіння тактової частоти не перевищувало 10% - 20%.

6. У результаті проведених досліджень, присвячених розв'язанню важливої науково-технічної проблеми в галузі створення нових ефективних алгоритмів обчислення тригонометричних, гіперболічних, експоненціальних, та степеневих функцій, запропоновано удосконалення методу CORDIC, як ефективного способу прискорення обчислень чи зменшення необхідних для цього апаратних затрат.

СПИСОК ОСНОВНИХ ОПУБЛІКОВАНИХ ПРАЦЬ ЗА ТЕМОЮ ДИСЕРТАЦІЇ

1. Борецький Т. Реалізація класичного та адаптивного CORDIC-методу на платформі IA32 / Т. Борецький, Л. Мороз // Комп'ютерні технології друкарства : зб. наук. пр. / Укр. акад. друкарства. Львів, 2012. № 28. С. 130–140.

2. Борецький Т. Модифікований CORDIC-метод обчислення синуса-косинуса / Л. Мороз, Т. Борецький, Т. Луковський, С. Войтусік // Комп'ютерні технології друкарства : зб. наук. пр. / Укр. акад. друкарства. Львів, 2015. № 33. С. 56–63.

3. Борецький Т. Швидкодіючий гібридний CORDIC-обчислювач

тригонометричних функцій / Л. В. Мороз, Я. І. Грабовський, Т. М. Микитів, Т. Р. Борецький, Ю. М. Костів, С. С. Войтусік // Науковий вісник НЛТУ України : зб. наук.-техн. пр. Львів, 2014. Вип. 24.8. С. 352–358.

4. Борецький Т. Удосконалення методу CORDIC для обчислення тригонометричних функцій засобами програмованої логічної інтегральної схеми / Л. В. Мороз, Т. Р. Борецький, М. М. Сколоздра // Науковий вісник НЛТУ України : зб. наук.-техн. пр. Львів, 2015. Вип. 25.5. С. 292–301.

5. Борецький Т. Синус-косинусний FPGA-обчислювач на основі CORDIC-методу з перекодуванням кута / Л. В. Мороз, Т. Р. Борецький, Ю. М. Костів // Науковий вісник НЛТУ України : зб. наук.-техн. пр. Львів,2015. Вип. 25.6. С.288-297.

6. Moroz L. Simple hybrid scaling-free CORDIC solution for FPGAs / L. Moroz, S. Nagayama, T. Mykytiv, I. Kirenko, T. Boretskyy // International Journal of Reconfigurable Computing. - 2014. Vol. - 2014. - pp. 1–4.

Наукові праці, які засвідчують апробацію матеріалів дисертації:

7. Борецький Т. Генерація випадкових чисел засобами ПЛІС / Л. Мороз, Т. Борецький // V Міжнародна науково-технічна конференція “Захист інформації і безпека інформаційних систем”. - Львів. - 2016.

8. Борецький Т. Покращення характеристик генератора псевдовипадкових послідовностей на основі тригонометричних функцій / Т. Борецький, Т. Микитів, Л. Мороз //І Міжнародна науково-технічна конференція “Інформаційна безпека в сучасному суспільстві”. - Львів. - 2014. - С. 17-19.

9. Борецький Т. Альтернативні методи обчислення тригонометричних функцій / Т. Борецький, Л. Мороз // 69-та студентська науково-технічна конференція секції кафедри «Безпека інформаційних технологій». - Львів. - 2011. - С. 179-180.

АНОТАЦІЯ

Борецький Т. Р. Розробка та реалізація методів обчислення елементарних функцій на основі програмних та апаратних засобів. - На правах рукопису.

Дисертація на здобуття наукового ступеня кандидата технічних наук за спеціальністю 05.13.05 «Комп’ютерні системи та компоненти». Національний університет “Львівська політехніка” Міністерства освіти і науки України, Львів, 2018.

Дисертаційна робота присвячена удосконаленню методів обчислення тригонометричних функцій, які на сьогоднішній день використовуютьтся переважно у засобах напівпровідникової техніки. Основним завданням у дослідженні було збільшення швидкодії алгоритмів шляхом оптимізації використовуваних структур для досягнення нижчих значень латентності (і відповідно кількості тактів), а також покращення показників ресурсоємності при реалізації алгоритмів за допомогою апаратних засобів. Запропоновані методи були реалізовані як на програмному рівні для процесорних архітектур різного рівня складності, так і з допомогою програмованих логічних інтегральних схем, що, в свою чергу, дає змогу здійснити портування розроблених методів у систему на кристалі.

Ключові слова: ітераційні методи, тригонометричні функції, мікроконтролер, програмовані логічні інтегральні схеми, цифровий обчислювач повороту системи координат.

АННОТАЦИЯ

Борецкий Т. Р. Разработка и реализация методов вычисления элементарных функций на основе программных и аппаратных средств. - На правах рукописи.

Диссертация на соискание ученой степени кандидата технических наук по специальности 05.13.05 «Компьютерные системы и компоненты». Национальный университет "Львівська політехніка" Министерства образования и науки Украины, Львов, 2018.

Диссертационная работа посвящена совершенствованию методов вычисления

тригонометрических функций, которые на сегодня используются преимущественно в средствах полупроводниковой техники. Основной задачей исследования было увеличение быстродействия алгоритмов путем оптимизации используемых структур для достижения более низких значений латентности (и соответственно количества тактов), а также улучшение показателей ресурсоемкости при реализации алгоритмов с помощью аппаратных средств. Предложенные методы были реализованы как на программном уровне для процессорных архитектур разного уровня сложности, так и с помощью программируемых логических интегральных схем, что позволяет осуществить портирование разработанных методов в систему на кристалле.

Ключевые слова: итерационные методы, тригонометрические функции, микроконтроллер, программируемые логические интегральные схемы, цифровой вычислитель поворота системы координат.

SUMMARY

Boretskyy T. R. Development and implementation of methods for elementary functions calculation on the basis of software and hardware. - On the rights of the manuscript.

A thesis submitted in fulfilment of the candidate of sciences degree in technical sciences on specialty 05.13.05 «Information technologies». Lviv Polytechnic National University, Ministry of Education and Science of Ukraine, Lviv, Ukraine, 2018.

The dissertation is devoted to the improvement of the methods of calculating trigonometric functions, which today are used mainly in semiconductor equipment. The main tasks of the study were to increase the speed of algorithms by optimizing the structures used to achieve lower latency values (and the number of cycles), as well as improving the resource-intensity indices when implementing algorithms using hardware. The proposed methods were implemented at the software level for processor architectures of different complexity levels and with the help of programmable logic integrated circuits, which, in turn, allows the porting of the developed methods to the system on the crystal. Considerable attention is paid to the CORDIC method by which one can compute functions such as sinus, cosine, tangent, exponent, square root, hyperbolic and inverse trigonometric functions. The hardware implementation of the considered methods on the FPGA platforms are realized. The features of selected FPGAs, the specifics of the hardware implementation of algorithms, in particular, the influence of architecture and integrated FPGA blocks on the functionality of the implemented methods are considered. The schemes of the offered algorithms on the level of register gears and the level of the stage in their placement (fitting) in the crystal are given. The methods of optimization of algorithms in terms of using a specific platform and depending on the version and settings of the development environment are considered. The results of impetitioning methods directly in the FPGA with the estimation of their initial characteristics, such as maximum clock speed, resource intensity and energy consumption, are presented. Verification of the correctness of algorithms functioning is carried out both by means of simulation modeling, and after measurement of physical indices and data obtained during the testing of programmable seamed FPGA.

Much attention is paid to determining the effectiveness of the algorithms, as well as the accuracy of the abstracts and the errors they make. The specificity of the operation of methods depending on the arsenal of teams and the possibilities of the target architecture, which helped to improve the range of characteristics of both the algorithms themselves and their hardware implementations, was explored. The obtained experimental results are compared with the classical methods of realization of the reduced functions. The integration of developed algorithms in the form of a separate device, which acts as a microcontroller with its own data transfer protocol and implementation.

Keywords: iteration methods, trigonometric functions, microcontrollers, field programmable gate array, coordinate rotation digital computer.