5. ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ГАЛУЗІ

УДК 681.3 *Доц. Л.В. Мороз, д-р техн. наук; аспір. Т.Р. Борецький; асист. М.М. Сколоздра, канд. техн. наук – НУ "Львівська політехніка"*

УДОСКОНАЛЕННЯ МЕТОДУ CORDIC ДЛЯ ОБЧИСЛЕННЯ ТРИГОНОМЕТРИЧНИХ ФУНКЦІЙ ЗАСОБАМИ ПРОГРАМОВАНОЇ ЛОГІЧНОЇ ІНТЕГРАЛЬНОЇ СХЕМИ

Наведено оптимізовані алгоритми обчислення функцій синуса-косинуса засобами програмованої логічної інтегральної схеми (ПЛІС), виявлено їх переваги та недоліки порівняно із класичними реалізаціями та отримано основні характеристики реалізованих методів. Використання методів оптимізації обчислень синуса та косинуса у засобах ПЛІС дають змогу покращити основні характеристики алгоритму у їхній апаратній реалізації порівняно з класичним методом у вигляді мегафункції, за допомогою якої істотно зменшується кількість тактів, латентність, кількість необхідних блоків та збільшується мінімальна тактова частота.

Ключові слова: CORDIC, ПЛІС, алгоритм, латентність, мегафункція.

Опис методу. Метод CORDIC широко використовують для обчислення тригонометричних, гіперболічних та обернених тригонометричних функцій у сучасній цифровій техніці **[**1-12**]**. Найбільш вживаним є обчислення синуса та косинуса вхідного кута, яке можна здійснювати як програмними, так і апаратними засобами. У програмній реалізації метод часто використовується у мікроконтролерах, у яких доступний лише обмежений спектр арифметично-логічних команд, невеликий об'єм пам'яті та цілочисельні регістри. В апаратній реалізації саме цим методом відбувається обчислення тригонометричних функцій у деяких сучасних процесорах з плаваючою комою. Основним недоліком методу є його висока латентність, що зумовлена складністю обчислення зазначених вище функцій, їхнім ітераційним характером. Наприклад, у сучасних персональних комп'ютерах обчислення функцій синуса/косинуса – це одна з кількох найдовших арифметичних операцій, яка триває більш як 100 тактів та у кілька разів перевищує час обрахунку квадратного кореня, ділення чи розрахунку логарифму з основою два **[**20-22**]**. У цій роботі розглянуто практичний аспект реалізації методів зменшення латентності та підвищення тактової частоти, таких як залишкове множення, табличний метод, перекодування кута за допомогою засобів FPGA. Саме при використанні такого програмно-апаратного рішення доступна можливість якісної оцінки методів оптимізації, внесених у структуру алгоритму, та дослідження вихідних характеристик методу в програмній та апаратній реалізації.

Мета роботи: практична реалізація оптимізованих алгоритмів обчислення функцій синуса-косинуса, описаних у **[**18**]**, засобами ПЛІС; виявлення їх переваг і недоліків порівняно із класичними реалізаціями **[**13-15**]**; отримання та зіставлення основних характеристик реалізованих методів.

Програмно-апаратна база, формати даних. Алгоритми методу CORDIC реалізовувались на базі програмних засобів від Altera: Quartus 13.1, та ModelSim – Altera Edition 10.4 с. Апаратну реалізацію було здійснено на платформі з кристалом Cyclone III EP3C16F484C6N, розміщеного на платі Terasic de0. Алгоритми та тестбенчі реалізовані мовою SystemVerilog, де частина, що відповідає за тестування та взаємодію з користувачем і ресурсами плати, винесена в окремий модуль, який не використовувався при синтезі та імплементації представлених результатів.

Вхідні та вихідні значення представлені в форматі з фіксованою комою **[**19**]**. Застосування плаваючої коми при реалізації функцій класичного синуса/косинуса в цьому випадку є недоцільним, оскільки як вхідні, так і вихідні значення функцій лежать в чітко визначених межах. Для вхідного значення кута це діапазон **[**0…π/2), а для вихідних значень синуса та косинуса **[**0…1).

Переваги формату з фіксованою комою:

- такий формат легше реалізувати апаратно;
- нема необхідності зберігання знаку мантиси та порядку;
- операції зсуву здійснюються монтажним чином, що дає змогу підвищити швидкодію та економити логічні елементи, необхідні для операцій зміни порядку числа;
- у форматі відсутні надлишкові розряди знаку та порядку числа, які в цьому конкретному випадку не використовуються;
- вихідні значення функцій можна без додаткових перетворень подавати на вхід цифро-аналогового перетворювача (ЦАП);
- за потреби, легше перетворити формат вихідних даних на число з плаваючою комою, ніж здійснювати операції в цьому форматі всередині алгоритму.

Перевагами формату з плаваючою комою можна назвати:

- ширший діапазон значень операндів;
- стандартизація формату.

Щодо діапазону значень, які приймає функція синуса/косинуса, то він є обмежений зверху значеннями **[**-1…1**]**, відповідно значення порядку числа є константою. Збільшення точності обчислень (нижня межа) досягається збільшенням розрядності операндів.

Стандартизація формату представлення може бути спричинена необхідністю узгодити вихідні формати різних функцій, діапазони яких істотно відрізняються. Наприклад, діапазон значень гіперболічного синуса лежить в діапазоні (-∞…+∞) і також може бути обчислений алгоритмами CORDICa. Тим не менше першочерговою задачею під час проведення досліджень є досягнення оптимальних характеристик роботи алгоритму, а узгодження форматів може істотно відрізнятись залежно від поставлених технічних умов. Зважаючи на наведені вище аргументи, формат з фіксованою комою найкраще підходить для поставленої задачі зіставлення алгоритмів, за якого виконання надлишкових операцій не може повною мірою розкрити всі якісні характеристики алгоритмів.

Реалізація методів. Реалізовувати обчислення синуса та косинуса класичним методом CORDIC за допомогою засобів ПЛІС можна як з використанням вбудованих мегафункцій, так і самостійно, згідно з формулою (1), описаною в **[**1, 2**]**

$$x_{i+1} = x_i - \sigma_i \cdot y_i \cdot 2^{-i};$$
$$y_{i+1} = y_i - \sigma_i \cdot x_i \cdot 2^{-i}; \quad (1)$$
$$z_{i+1} = z_i - \sigma_i \cdot \alpha_i,$$

де: $\alpha_i = \arctan\left(2^{-i}\right)$; $\sigma_i = \begin{cases} -1 if z_i < 0 \\ +1 if z_i \geq 0 \end{cases}$ $i = 0,1,2,...,m$; $x_0 = P = \frac{1}{K_m}$; $y_0 = 0; z_0 = \varphi$;

$K_m = \prod_{i=0}^{m-1} \frac{1}{\sqrt{1+2^{-2i}}}$, m – кількість двійкових розрядів.

Із запропонованої виробником реалізації CORDIC Megafunction для обчислення синуса та косинуса необхідно виділити близько третини ресурсів кристалу (табл. 1). Причому розрядність мантиси фіксована та становить 32 розряди, а на виході алгоритму одержуємо лише одну із двох функцій – синус чи косинус.

Табл. 1. Реалізація синуса та косинуса за допомогою інтегрованих мегафункцій

Bus, bits	Clock	Bandwith, Mbit/s	Latency, ns	Blocks	Logic	Flip-flops	Mem, bits	Freq. 85°C	Freq. 0°C	Freq. Max
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]

Під час здійснення обчислень тригонометричних функцій за допомогою власної реалізації класичного алгоритму CORDIC є можливість оптимізувати його структуру та характеристики, виходячи із специфіки поставленої задачі. Розрядність та кількість ітерацій алгоритму можна змінювати з кроком в один розряд, забезпечуючи необхідну точність обчислень. Тим не менше, для уніфікації результатів наведено значення, кратні восьми бітам (табл. 2).

Табл. 2. Реалізація класичного CORDICa високої розрядності

Bus, bits	Clock	Bandwith, Mbit/s	Latency, ns	Blocks	Logic	Flip-flops	Mem, bits	Freq. 85°C	Freq. 0°C	Freq. Max
32	24	7031	109,22	3306	3142	2132	0	219,73	248,39	[376,8]
32	24	7030	109,25	2285	2175	1491	558	219,68	246,06	[369,9]
32	32	6963	147,07	4326	4173	2900	0	217,58	246,79	[378,2]
32	32	6790	150,82	2889	2775	2011	608	212,18	237,08	[364,3]
40	32	7946	161,09	3889	3749	2525	1014	198,65	224,01	[337,9]
48	32	9006	170,56	4732	4664	2919	1316	187,62	212,99	[314,0]
48	48	8624	267,17	6577	6505	4471	2068	179,66	203,96	[306,8]
64	48	9912	309,94	9428	9360	5928	3474	154,87	174,89	[268,0]
64	64	9568	428,09	12158	12086	8245	3904	149,5	169,49	[252,3]

Під час порівняння результатів потрібно врахувати, що вбудовані мегафункції розраховані, передусім, на універсальність як з погляду незалежності від платформи, так і даних, що представлені у форматі з плаваючою комою. Відповідно, з усіх основних параметрів (частота, пропускна здатність, латентність) власна реалізація алгоритму програє лише за латентністю, за розрядності, більшій за 48.

Саме висока латентність є слабким місцем класичного методу CORDIC. Кількість тактів, необхідних для обчислення функцій, переважно дорівнює

кількості розрядів вхідного аргументу. Як результат, вихідні значення алгоритму отримуються із значною затримкою, і при цьому з використанням великої кількості ресурсів кристалу. Також варто зазначити, що зі збільшенням розрядності в n разів кількість необхідних логічних елементів збільшується в n^2 разів.

Для зменшення кількості тактів, логічних елементів і латентності алгоритму можна використати табличний метод та залишкове множення [16, 17] згідно з такою формулою:

$$m_{LUT} = \frac{m}{2}; \varphi_{input} = \varphi_1 + \varphi_2;\ \varphi_1 = \sum_{i=1}^{m_{LUT}} a_i 2^{-i};\ \varphi_2 = \sum_{i=m_{LUT}+1}^{m} a_i 2^{-i} \quad (2)$$

$$x_{LUT} = \cos(\varphi_1); y_{LUT} = \sin(\varphi_1);\ x_m = x_{LUT} - \varphi_2 \cdot y_{LUT};\ y_m = y_{LUT} + \varphi_2 \cdot x_{LUT}.$$

Табличний метод застосовується для старших розрядів операнду, кількість яких визначається доступним об'ємом пам'яті. Розмір таблиці збільшується більш як удвічі з кожною наступною ітерацією. Для вибраного кристалу максимальна кількість ітерацій, які можна помістити в пам'ять, дорівнює дванадцяти, що дає змогу виконати одну ітерацію зчитування з пам'яті замість перших дванадцяти ітерацій алгоритму CORDIC (табл. 3).

Своєю чергою, метод залишкового множення дає змогу замінити другу половину ітерацій методу CORDIC однією операцією множення та додавання для кожної з функцій синуса та косинуса. Метод залишкового множення можна застосувати лише для половини молодших розрядів вхідного кута. За наявності в кристалі блоків з функцією множення з'являється можливість 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, яка в представленому випадку буде становити близько трьохсот мегагерц.

Табл. 3. Табличний метод та залишкове множення (мінімальна латентність)

Bus, bits	Clock	Bandwith, Mbit/s	Latency, ns	Blocks	Logic	Flip-flops	Mul-tipliers	Mem, bits	Freq 85°C	Freq 0°C	Freq Max
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	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]

У табл. 3 наведено результати імплементації для двох варіантів розрядності, які є критичними для вибраного кристалу. За розрядності 14 біт існує можливість задіяти всі помножувачі кристалу в конфігурації 9×9 біт, чим досягти високих показників пропускної здатності та мінімальної латентності. Всього можливо створити 56 незалежних конвеєрів. Подальше збільшення розрядності призведе до зменшення кількості конвеєрів та використовуваних блоків помножувачів і за значення 24 досягне свого максимуму з використанням лише одного помножувача з конфігурацією 18×18. У цьому випадку критичним значенням

буде об'єм доступної пам'яті, для доступу до якої на максимальній швидкості доводиться збільшувати кількість тактів.

За невеликих розрядностей аргументів можна використовувати тільки наведені вище два методи для отримання результату обчислюваної функції. У нашому випадку верхня межа становить 24 розряди, з яких старші 12 розрядів обчислюються табличним методом, а молодші 12 – залишковим множенням. Теоретично, в цьому випадку максимальна частота дорівнює частоті блоку помножувача, а латентність становить чотири такти, два з яких йде на зчитування адреси та вибірку з пам'яті, а наступні два – на множення та додавання.

Під час практичної реалізації такого підходу відразу отримуємо ряд "підводних каменів":

- по-перше, блоки пам'яті за великої їх кількості розміщені в різних кінцях кристалу (рис. 1), і затримка сигналу між ними та будь-яким блоком помножувача буде істотна навіть на невеликих кристалах. У розглянутому випадку максимальна частота зменшилась більш як вдвічі, хоча об'єм пам'яті у кристалі становить менше 1 % від об'єму пам'яті в топових кристалах ПЛІС;
- по-друге, навіть у разі використання одного чи кількох блоків пам'яті максимальна тактова частота може бути досягнута лише при попаданні зчитаних даних з пам'яті безпосередньо у регістри помножувача, а у всіх сімействах ПЛІС блоки пам'яті та помножувачі розміщені на певній відстані один від одного, що не дає змогу здійснити множення відразу після операції зчитування з пам'яті;
- по-третє, для підвищення тактової частоти необхідно розбити шлях від елементів пам'яті до блоків помножувачів на кілька розділених тригерами частин, під час проходження сигналу між якими ніяких логічних чи арифметичних операцій не відбувається. Внаслідок отримаємо модель, в якій зі зростанням латентності методу не відбувається ніяких покращень у характеристиках алгоритму.

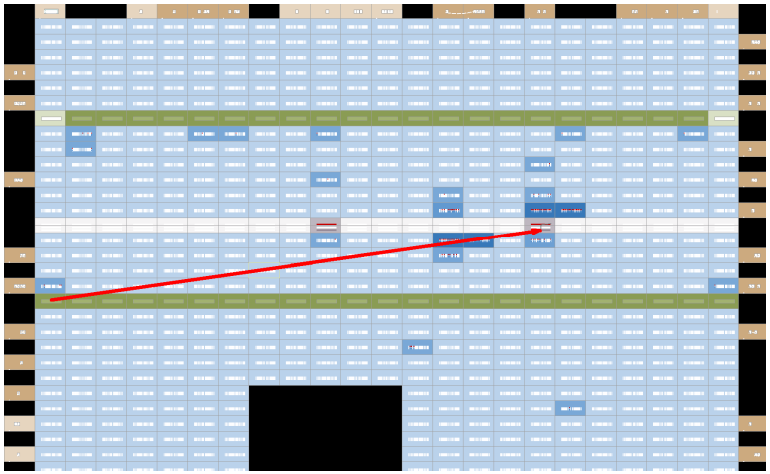

Рис. 1. Найдовший шлях сигналу (червоним) між блоками пам'яті та помножувачем

З цим самим успіхом можна здійснювати ітерації методу CORDIC між блоками пам'яті та помножувачами, не змінивши латентність та тактову частоту алгоритму. При цьому кожна ітерація CORDIC дасть змогу отримати на виході

функції два додаткові розряди, враховуючи розряди, отримані за допомогою методу залишкового множення. При цьому недоліком є лише збільшення кількості необхідних логічних елементів для реалізації ітерацій алгоритму. Також не варто забувати, що в разі збільшення розрядності між табличним методом та залишковим множення необхідно буде проводити ітерації класичного CORDIC методу, з кожною ітерацією якого кількість логічних елементів буде зростати в арифметично-геометричній прогресії [24].

Тому для економії логічних елементів можна застосовувати запропонований метод перекодування кута [18], в якому частина логіки класичного методу CORDIC, що відповідає за визначення напряму повороту, не використовується. Для цього методу вхідний кут φ розбивається на три кути

$$\varphi = \varphi_1 + \varphi_2 + \varphi_3,\ \varphi_1 = \sum_{i=1}^{m_{LUT}} a_i 2^{-i},\ \varphi_2 = \sum_{i=m_{LUT}+1}^{m_{CORDIC}} a_i 2^{-i},\ \varphi_3 = \sum_{i=m_{CORDIC}+1}^{m} a_i 2^{-i}. \qquad (3)$$

Тут m_{LUT} – число старших бітів кута φ, що подаються на переглядову таблицю LUT, яка виконує функцію

$$x_{m_{LUT}+1} = P_c \cdot \cos\left(\varphi_1 + \varphi_{2const}\right),\ y_{m_{LUT}+1} = P_c \cdot \sin\left(\varphi_1 + \varphi_{2const}\right),$$

$$\varphi_{2const} = 2^{-(m_{LUT}+1)} - 2^{-(m_{CORDIC}+1)}, \qquad (4)$$

де m_{CORDIC} – кількість середніх бітів $\left(i_{CORDIC} = m_{LUT}+2 \ldots m_{CORDIC}+1\right)$, які обробляються за методом CORDIC (біти кута φ_2). Причому саме в CORDICу кут φ_2 перекодовується в кут φ_{2r}

$$\varphi_{2r} = \sum_{i=m_{LUT}+2}^{m_{CORDIC}} b_i 2^{-i} = \sum_{i=m_{LUT}+2}^{m_{CORDIC}} (2a_{i-1} - 1) \cdot 2^{-i}, \qquad (5)$$

де φ_{2const} визначається за формулою (4), так що $\varphi_2 = \varphi_{2r} - \varphi_{2const}$. Після завершення ітерацій за формулам:

$$x_i = x_{i-1} - b_i y_{i-1} 2^{-i},\ y_i = y_{i-1} + b_i x_{i-1} 2^{-i},\ i = m_{LUT}+2, \ldots, m_{CORDIC}+1 \qquad (6)$$

отримаємо вектор з координатами $\left\{x_{m_{CORDIC}+1}; y_{m_{CORDIC}+1}\right\}$. Цей вектор здійснює додатковий поворот на залишковий кут $\varphi_{зал}$, який дорівнює скоригованому куту φ_3 на кут Δ $\varphi_{зал} = \varphi_3 + \Delta$, де Δ визначається за формулою

$$\Delta = \sum_{m_{LUT}+2}^{m_3} d_i = \sum_{m_{LUT}+2}^{m_3} b_i \left[2^{-i} - \arctan\left(2^{-i}\right)\right],\ m_3 = \frac{m}{3} - 1. \qquad (7\text{-}8)$$

Далі виконуються дві дії, подібно до алгоритму, описаному у (2) – множення на кут $\varphi_{зал}$:

$$x_{m_{CORDIC}+2} = x_{m_{CORDIC}+1} - \varphi_{зал} \cdot y_{m_{CORDIC}+1};\ x_{m_{CORDIC}+2} = \cos\left(\varphi_{input}\right)$$

$$y_{m_{CORDIC}+2} = y_{m_{CORDIC}+1} + \varphi_{зал} \cdot y_{m_{CORDIC}+1};\ y_{m_{CORDIC}+2} = \sin\left(\varphi_{input}\right). \qquad (9)$$

Цей спосіб дає змогу усунути конвеєр із каскадованих суматорів та мультиплексорів класичного методу CORDIC. Окрім цього, таблиця арктанген-

сів, яка часто реалізується компілятором у вигляді монтажних з'єднань логічних елементів, стає непотрібною.

Табл. 4. Результати модифікованого методу перекодування кута

Bus, bits	Mem/Rot /Mult, bits	Clock	Bandwith, Mbit/s	Latency, ns	Blocks	Logic	Flip-flops	Mem, bits	Freq. 85°C	Freq. 0°C	Freq. Max
16	6/2/8	6	3868,5	28,95	228	138	217	832	241,78	266,31	[435,73]
16	56 × 6/2/8	56 × 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	207,3	[321,85]

Запропонований метод перекодування кута реалізований у двох варіантах: 16 та 38 розрядному. У першому випадку задіюються всі помножувачі, при цьому використання блоків пам'яті становить менше 10 % від ресурсів кристалу. Варто також зазначити, що між операціями зчитування з пам'яті та множенням здійснюється метод перекодування кута, що дає змогу використати холості такти для арифметичних ітерацій алгоритму, користь яких найкраще проявляється за великих об'ємів пам'яті. У випадку 16-розрядного CORDICa з 56 конвеєрами критичним параметром виступає кількість логічних елементів кристалу, кількість яких становить 15,4 тис. (табл. 4). Подальше збільшення розрядності до 38 біт впирається в розрядність блоку помножувача. Незважаючи на повну завантаженість блоків пам'яті, вони в цьому випадку не виступають гальмівним фактором, і їх кількість може бути зменшена за рахунок збільшення латентності.

Під час практичної реалізації запропонованого методу перекодування кута через спрощення основного алгоритму та зменшення необхідної кількості логічних елементів для його реалізації фіттеру вдається краще розмістити комбінаційні функції всередині кристалу, що дасть змогу збільшити тактову частоту алгоритму. Цей ефект також спостерігається за збільшення розрядності класичного CORDICa, коли із збільшенням кількості логічних елементів відбувається падіння тактової частоти, хоча в теорії повинна змінюватись лише латентність методу. Таким чином, використовуючи перекодування кута, можна добитись не лише економії логічних елементів, але й підвищення швидкодії та пропускної здатності.

Вплив додаткових параметрів компіляції на результати роботи алгоритмів. Якщо в кристалі відсутні або зайняті блоки пам'яті чи помножувачі, вони можуть бути реалізовані за допомогою функцій комбінаційної логіки кристалу. Компілятор Quartus у цьому випадку здатний самостійно згенерувати таку комбінаційну схему, хоча вона не завжди буде оптимальною. Також відбудеться падіння швидкодії, яку можна скомпенсувати внаслідок збільшення латентності. Падіння частоти для апаратного та комбінаційного блоку помножувача залежно від розрядності, а також кількість необхідних елементів для їх реалізації відображено в табл. 5. Саме частота функціонування блоку помножувача є основним бар'єром на шляху підняття тактової частоти алгоритму загалом, оскільки як блоки пам'яті, так і логічні елементи кристалу функціонують на вищій частоті, ніж помножувачі.

Табл. 5. Швидкодія програмних і апаратних помножувачів

Multipliers width	Multiply Blocks	Freq. 85°C	Freq. 0°C	Freq. Max
Hard Multipliers 9×9	1 Block	353,11	343,29	[559,6]
Hard Multipliers 18×18	2 Blocks	327,23	288,85	[510,7]
Soft Multipliers 9×9	115 LE	190,26	170,10	[301,1]
Soft Multipliers 18×18	424 LE	123,56	110,31	[192,8]

Своєю чергою, блоки пам'яті, так само як і помножувачі, можуть бути реалізовані за допомогою логічних елементів чи безпосередньо монтажних з'єднань, якщо дані є статичними. Структура таких блоків залежить від розрядності та типу логічних елементів, використовуваних у кристалі, які можуть мати різну архітектуру, зокрема в межах однієї серії. Як правило, компілятор самостійно здатний згенерувати елементи пам'яті, враховуючи архітектуру, наявні вільні ресурси та налаштування оптимізації.

Споживана потужність змінюється у вузьких межах, та здебільшого залежить від кількості використовуваних виводів кристалу. Існує взаємозв'язок між кількістю задіяних логічних елементів та енергоспоживанням ядра алгоритму, але вона дуже незначна, і переважно не змінюється за різних конфігурацій проекту. Загалом статична споживана потужність ядра без врахування енергоспоживання ніжок ПЛІС для всіх апробованих алгоритмів лежить у межах 51,75-52,25 mW, та через низьку інформативність не подана в табл. 5. Також на споживану потужність впливає температура середовища, швидкість руху повітря, тип (активне чи пасивне) охолодження та його параметри. Окрім цього, проекти не були оптимізовані для зменшення енергоспоживання, і наведені вище параметри не мінімізовані.

Оптимізація всіх представлених алгоритмів проводилась задля досягнення максимальної швидкодії, за рахунок збільшення кількості зайнятих ресурсів та потужності. Щоправда, іноді через зменшення загальної площі проекту вдавалось покращити швидкодію, хоча компілятор не помічав такого варіанта компонування. Також у деяких випадках можна зменшити кількість логічних елементів за рахунок блоків пам'яті, що практично не впливало на швидкодію алгоритму. В таких випадках результати наводились для обох варіантів.

Розрахунок пропускної здатності та латентності здійснювався на основі найгірших експлуатаційних умов, при яких робота пристрою є гарантована виробником. Для пропускної здатності наведене мінімально-можливе значення, а для латентності – максимально-можливе. На практиці ці показники в 1,5-2 рази кращі, залежності від умов експлуатації.

Висновки. Використання описаних методів оптимізації обчислень синуса та косинуса у засобах ПЛІС дають змогу покращити основні характеристики алгоритму у їхній апаратній реалізації порівняно з класичним методом у вигляді мегафункції [15]. Найкраще це проявляється у зменшенні латентності алгоритму, а також у збільшенні пропускної здатності та економії ресурсів кристалу.

Для 32 вірних розрядів функцій синуса та косинуса, за допомогою методу перекодування кута, порівняно з класичним алгоритмом CORDIC із вбудованої бібліотеки середовища, покращено:

- кількість тактів зменшилась від 36 до 10;

- латентність знизилась у 5 разів – від 263,5 нс до 53,75 нс;
- кількість необхідних елементів зменшилась в чотири рази – від 5248 до 1311 блоків;
- мінімальна тактова частота зросла на 36 % – від 136,63 до 186,05 МГц.

Значна частина показників покращена за рахунок ширшого використання пам'яті, об'єм якої становить 489 Кбіт. Це значення може змінюватись залежно від платформи і впливати на латентність. Враховуючи низьку ресурсоємність алгоритму в представленому кристалі можливо реалізувати до 56 паралельних конвеєрів алгоритму на основі методу перекодування кута, кількість яких обмежується кількістю помножувачів, а розрядність – кількістю логічних елементів. Внутрішня сумарна пропускна здатність такої конфігурації перевищує 200 Гбіт/с, а всі ітерації містять логічне навантаження, що збільшує корисну роботу алгоритму.

Література

1. Volder, J. Binary computation algorithms for coordinate rotation and function generation / J. Volder // Convair Report IAR-1 148 Aeroelectrics Group, June 1956.
2. Volder, J. The CORDIC Trigonometric Computing Technique / J. Volder // IRE Trans. Electronic Computing. – Vol. EC-8. – Pp. 330-334 Sept 1959.
3. F. de Dinechin "Fixed-point trigonometric functions on FPGAs", in Proceedings of the 4th International Symposium on Highly-Efficient Accelerators and Reconfigurable Technologies / F. de Dinechin, M. Istoan, and G. Sergent // Royaume-Uni, Edimburgh, UK, March 2013.
4. Francisco AGUIRRE-RAMOS, Alicia MORALES-REYES, Rene CUMPLIDO, Claudia FEREGRINO-URIBE. An Area Efficient Composed CORDIC Architecture. Advances in Electrical and Computer Engineering. – Vol. 14, Number 2, 2014. – Pp. 113-116.
5. Shoab, A.K. Digital design of signal processing systems: A practical approach (1st ed). New York, NY: John Wiley, 2011.
6. P.K. Meher, J. Valls, T.-B. Juang, K. Sridharan, andK. Maharatna. 50 years of cordic: Algorithms, architectures, and applications. IEEE Transactions on Circuits and Systems I: Regular papers, Sept. 2009. – Vol. 56(9) . – Pp. 1893-1907.
7. D. Timmermann, H. Hahn, and B. Hostika. Modified CORDIC algorithm with reduced iterations.Electronics Letters, 1989. – Vol. 25(15). – Pp. 950-951.
8. Jean-Michel Muller. Elementary Functions: Algorithms and Implementation. Second Edition, Birkhaeuser, 2006.
9. Pramod K. Meher and S.Y. Park. CORDIC Designs for Fixed Angle of Rotation. IEEE transactions on very large scale integration (VLSI) systems. Issue: 99. – Pp. 1-12.7 March, 2012.
10. Lakshmi B. and Dhar A.S. CORDIC Architectures: A Survey. VLSI Design. – Vol. 2010, Article ID 794891, 19 pages, January 8, 2010.
11. Beuler M. CORDIC-Algorithmus zur Auswertung elementarer Funktionen in Hardware. FH-Report. Juni 2008. **[**Electronic resource**]**. – Mode of access http://dok.bib.fhgiessen. de/opus/volltexte/2009/4148/.
12. Pramod K. Meher, Javier Valls, Tso-Bing Juang, K. Sridharan, Koushik Maharatna. 50 Years of CORDIC: Algorithms, Architectures, and Applications. IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS – I: REGULAR PAPERS. – Vol. 56, No. 9, SEPTEMBER 2009. – Pp. 1893-1907.
13. **[**Electronic resource**]**. – Mode of access http://www.altera.com.cn/content/dam/altera-www/global/zh_CN/pdfs/literature/ug/ug_ altfp_mfug.pdf
14. **[**Electronic resource**]**. – Mode of access http://www.altera.com/content/dam/altera-www/global/en_US/pdfs/literature/ug/ug_ altfp_mfug.pdf.
15. Altera ALTFP_SINCOS quartus megafunction.
16. Мороз Л.В. 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. праць. – Львів : РВВ НЛТУ України. – 2014. – Вип. 24.8. – С. 352-357.
17. Байков В.Д., Смолов В.Б. Специализированные процессоры: итерационные алгоритмы и структуры. – М. : Изд-во "Радио и связь", 1985. – 288 с.

18. Л. Мороз, Т. Борецький, Т. Луковський, С. Войтусік. Модифікований CORDIC-метод обчислення синуса – косинуса // Компютерні технології друкарства. – 2015. – № 33. – С. 72 –80.
19. **[**Electronic resource**]**. – Mode of access http://en.wikipedia.org/wiki/Fixed-point_arithmetic
20. **[**Electronic resource**]**. – Mode of access http://www.intel.com/content/dam/www/public/us/en/documents/manuals/64-ia-32-architectures-optimization-manual.pdf
21. **[**Electronic resource**]**. – Mode of access http://developer.amd.com/wordpress/media/2012/10/47414_15 h_sw_opt_guide.pdf
22. **[**Electronic resource**]**. – Mode of access http://www.codemachine.com/downloads/AMD_SoftwareOptimizationGuide.pdf
23. **[**Electronic resource**]**. – Mode of access http://www.kit-e.ru/articles/plis/2011_12_36.php
24. **[**Electronic resource**]**. – Mode of access http://en.wikipedia.org/wiki/Arithmetico-geometric_sequence.

Мороз Л.В., Борецкий Т.Р., Сколоздра М.М. **Совершенствование метода CORDIC для вычисления тригонометрических функций средствами программируемой логической интегральной схемы**

Приведены оптимизированные алгоритмы вычисления функций синуса-косинуса средствами программируемой логической интегральной схемы (ПЛИС), выявлены их преимущества и недостатки по сравнению с классическими реализациями и получены основные характеристики реализованных методов. Использование методов оптимизации вычислений синуса и косинуса в средствах ПЛИС дают возможность улучшить основные характеристики алгоритма в их аппаратной реализации по сравнению с классическим методом в виде мегафункции, с помощью которой существенно уменьшается количество тактов, латентность, количество необходимых блоков и увеличивается минимальная тактовая частота.

Ключевые слова: CORDIC, ПЛІС, алгоритм, латентность, мегафункция.

Moroz L.D., Boretskyy T.R., Skolozdra M.M. **Improvement of the CORDIC Method for Calculating of Trigonometric Functions by Using FPGA Tools**

The optimized algorithms for calculating the sine-cosine functions by using FPGA tools are presented; their advantages and disadvantages compared with classical implementations are shown; main characteristics of the implemented methods are obtained. Usage of optimization methods in calculating sine and cosine in FPGA tools makes enable improving the main characteristics of the algorithm in their hardware implementation compared to the classical method as mega functions, where substantially reduces the number of clock cycles, latency, number of required blocks and increases the minimum clock frequency.

Keywords: CORDIC, FPGA, algorithm, latency, mega functions.

УДК 004.[942+772]

Доц. І.М. Дронюк, канд. фіз.-мат. наук;
аспір. О.Ю. Федевич – НУ "Львівська політехніка"

ПРОГНОЗУВАННЯ ТРАФІКУ КОМП'ЮТЕРНОЇ МЕРЕЖІ ДЛЯ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ВИКОРИСТАННЯ МЕРЕЖЕВОГО ОБЛАДНАННЯ

На основі розробленої інформаційної технології реалізовано моніторинг трафіку комп'ютерної мережі. Описано розроблене програмне забезпечення для моніторингу трафіку. На основі теорії Ateb-функцій реалізовано прогнозування тренду трафіку. Спираючись на результати прогнозування трафіку та максимальне допустиме завантаження вузла мережі для забезпечення якості обслуговування, реалізується перерозподіл навантаження у мережі. Здійснений перерозподіл забезпечує збільшення коефіцієнта завантаження обладнання, що підвищує ефективність використання мережевого обладнання.

Ключові слова: трафік, комп'ютерна мережа, моніторинг мережі, швидкість передачі даних, прогнозування трафіку, Ateb-функції.