Koval O.M., Hulida Ye.M. **Mathematical Model of Optimal Choice of Fire Fighting Tactics at Millyard Applications Woodworking Companies**

The mathematical model, flowchart and a program for optimizing the choice of tactics of containment and extinguishing fires on the open storage timber wood processing enterprises are designed. The model includes all of the dependencies for establishing the optimal time localization and extinguishing effect based on the velocity and direction of the wind, causing changes in the speed of flame propagation of the fire. In addition, the mathematical model can reasonably determine the optimal variant of tactics and duration localization and fire extinguishing with the necessary amount with the forces and resources, and the need for fire suppression protection from gas and smoke equipment and heat protection clothing. Scientific novelty lies in the fact that for the first time with the help of optimization mathematical model considered and justified basic provisions of tactical action fire suppression systems on open storage timber wood processing enterprises.

Keywords: mathematical model, the tactics of containment and extinguishing the fire barrel and thermal radiation.

УДК 681.3

Доц. Л.В. Мороз, д-р техн. наук; аспір. Т.Р. Борецький; доц. Ю.М. Костів, канд. техн. наук – НУ "Львівська політехніка"

СИНУС-КОСИНУСНИЙ FPGA-ОБЧИСЛЮВАЧ НА ОСНОВІ CORDIC-МЕТОДУ З ПЕРЕКОДУВАННЯМ КУТА

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 в їхній апаратній реалізації порівняно з класичним методом, зокрема у вигляді мегафункції, за допомогою якої істотно зменшується кількість тактів, латентність, кількість необхідних блоків та збільшується мінімальна тактова частота.

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

Вступ. В цій роботі розглянуто практичну реалізацію алгоритмів обчислення синуса-косинуса та його синтез на платформі ПЛІС (FPGA). Імплементацію проведено для двох типів ПЛІС – від фірм Altera та Xillix.

Опис відомих методів. Основним недоліком класичного методу CORDIC **[**1, 2**]** є низька швидкодія через лінійну збіжність методу (один правильний біт результату 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:

$$\begin{aligned} x_i &= x_{i-1} - \sigma_i \cdot y_{i-1} \cdot 2^{-i}; \\ y_i &= y_{i-1} - \sigma_i \cdot x_{i-1} \cdot 2^{-i}; \\ z_i &= z_{i-1} - \sigma_i \cdot \arctan(2^{-i}); \\ \sigma_i &= sign(z_{i-1}), i = 1...m, \end{aligned} \quad (1)$$

де m – кількість двійкових розрядів обчислювача.

З метою спрощення апаратної реалізації обчислювача запропоновано метод CORDIC з перекодуванням вхідного кута **[**4-7**]**, що дає змогу звести систему (1) лише до двох ітераційних рівнянь (для x_i, y_i). Одночасно з цим для підвищення швидкодії методу (зменшення числа ітерацій) розроблено гібридні структури, що використовують послідовно три методи: табличний + CORDIC + кусково-лінійна апроксимація **[**3, 4, 8, 9**]**. Причому CORDIC виконано у різних варіантах – класичному **[**3**]**, з використанням ітераційних формул вищих порядків **[**8-10**]**, без перекодування та з перекодуванням вхідного кута **[**4**]**. Найпростішим з погляду апаратного втілення є CORDIC з перекодуванням вхідного кута **[**4**]**. Однак він має істотний недолік – великий об'єм пам'яті таблиці попередньої вибірки (ТПВ типу LUT) при великих значеннях m (необхідна таблиця розміром, не меншим, ніж $2^{m/3} \cdot m$ бітів). Окрім цього, вихідні помножувачі тут реалізовані у базисі {-1;1}, що унеможливлює використання помножувачів, які є у складі блоків DSP сучасних ПЛІС.

Слабким місцем методу класичного перекодування кута **[**4**]** була необхідність використання ТПВ великого об'єму. У статтях **[**12, 13**]** цей недолік було зведено до мінімуму, коли при розрядності 24-32 бітів мінімально можлива за об'ємом ТПВ вимагала аналізу лише 2-3 розрядів вхідного аргументу. Удосконалений алгоритм перекодування кута був реалізований на платформі FPGA CycloneIII мовою програмування SystemVerilog **[**11**]**.

Основними складностями у процесі реалізації алгоритмів цього методу були:

- реалізація арифметичних і логічних операцій зі змінними та масивами різного типу;
- необхідна наявність блоків помножувача з підтримкою знакового представлення аргументів;
- неможливість коректно виконувати дію множення знакових і беззнакових величин засобами мови Verilog **[**14**]**;
- необхідність зведення всіх представлених величин до одного формату;
- зменшення розрядності та точності обчислень через надлишковість формату.

Під час проектування та налаштування алгоритму виникали труднощі у зв'язку з наведеними вище обмеженнями. Усі ці труднощі можна усунути, використавши метод перекодування, описаний у роботі **[**12**]**.

Однак у **[**12**]** описано лише основи теорії методу, які потребують формалізації у вигляді алгоритмів, придатних для апаратного виконання на платформі FPGA. Саме це і є метою цієї роботи.

Опис пропонованого алгоритму для тригонометричних функцій синус-косинус. Пропонований алгоритм методу **[**12**]** здатний коректно функціонувати на проміжку

$$\varphi \in [0, \pi/2] \quad (2)$$

та використовувати при арифметичних операціях лише додатні величини. Розділимо вхідний кут ϕ на три кути

$$\varphi = \varphi_1 + \varphi_2 + \varphi_3. \quad (3)$$

Кут φ_1 опрацьовуємо табличним методом (m_{LUT} – число старших бітів кута ϕ, що подаються на переглядову таблицю ТПВ), кут φ_2 – пропонованим Cordic-методом, кут φ_3 – за допомогою методу залишкового множення (кусково-лінійної апроксимації), причому:

$$\varphi_1 = \sum_{i=1}^{m_{LUT}} a_i \cdot 2^{-i}; \qquad \varphi_2 = \sum_{i=m_{LUT}+1}^{m_{CORDIC}} a_i \cdot 2^{-i}; \quad \varphi_3 = \sum_{i=m_{CORDIC}+1}^{m} a_i \cdot 2^{-i}; a_i \in \{0,1\}. \tag{4}$$

Застосування табличного методу передбачає зчитування наперед обчислених значень синусів та косинусів кута φ_1 згідно з вибраною розрядністю ТПВ. Дані значення отримуємо за формулою:

$$x_{LUT} = P \cdot \cos(\varphi_1 + \varphi_{const}); \quad y_{LUT} = P \cdot \sin(\varphi_1 + \varphi_{const}); \tag{5}$$

$$P = \prod_{m_{LUT}}^{m_{CORDIC}} \frac{1}{\sqrt{1 + 2^{-2i-2}}}; \tag{6}$$

$$\varphi_{const} = \sum_{m_{LUT}}^{m_{CORDIC}} \arctan(2^{i-2}). \tag{7}$$

Для кута φ_2 використовуємо ітераційні рівняння пропонованого методу:

$$x_{i+1} = x_i - b_{i+1} y_i \cdot 2^{-i}; \; y_{i+1} = y_i - b_{i+1} x_i \cdot 2^{-i}; \tag{8}$$

$$b_i = 2a_i - 1. \tag{9}$$

При кусково-лінійній апроксимації застосуємо формули:

$$x = x_{CORDIC+1} - z \cdot y_{CORDIC+1}; \qquad y = y_{CORDIC+1} + z \cdot x_{CORDIC+1}, \tag{10}$$

значення z при обрахунку (10) набуває лише додатних значень і може обчислюватись незалежно від основного ітераційного процесу (8):

$$z = \varphi 3 + \Delta; \quad \Delta = \sum_{i=m_{LUT}}^{m/3} 2^{-i} - 2 \cdot \arctan 2^{-i-1}. \tag{11}$$

Для реалізації формул (10) бажаною є наявність апаратного помножувача, а у випадку відсутності такого – його програмна емуляція (наприклад, за методом add-shift **[**11**]**). Порівняння швидкодії програмної та апаратної реалізації блоку помножувача було здійснено в **[**11**]**. Далі в статті розкрито практичну частину реалізації пропонованого методу.

Практична реалізація. Важливою перевагою пропонованого алгоритму є оперування лише додатними величинами, що досягається поворотом кута лише в одному напрямі після операції зчитування з ТПВ. Цей спосіб дає змогу отримувати лише значення, більші від нуля, що знаходить своє відображення в тому, що усі змінні в методі можна використовувати як беззнакові, якими вони є по замовчуванню у мові Verilog. Також можна говорити про хоч і незначне, але збільшення швидкодії – точності при застосуванні методу шляхом економії знакових розрядів після усунення надлишкових біт. Характеристики реалізованого алгоритму було досліджено за допомогою програмного забезпечення Vivado 2015.2 від фірми Xillinx. Реалізація мегафункції обчислення синуса/косинуса від цього виробника є більш гнучкою та продуманою порівняно з версією від фірми Altera. За її допомогою можна обчислювати також гіперболічні та обернені функції синуса/косинуса/тангенса, квадратний корінь та поворот вектора. Окрім цього, доступне порозрядне редагування розрядності обчислень та кількість ітерацій у широких межах (для синуса/косинуса вона становить від 8 до 48 розрядів), а також зміна ітераційної структуру алгоритму. Детальніша інформація про можливості Xilinx IP Core є у **[**16**]**.

Першим етапом пропонованого алгоритму є табличний метод, в якому значення результату вибирається із пам'яті, а вхідний аргумент функції відіграє роль адреси. Для оцінки апаратної ресурсоємності такого підходу можна взяти кристал ПЛІС, класу CycloneIII, який використовувався для дослідів у цій роботі. Один логічний елемент кристалу здатний зберігати 16 двійкових розрядів, замінюючи собою невелику ТПВ. Для реалізації класичного CORDIC методу необхідно використати 2-5 тис. логічних елементів. Так, наприклад, при використанні Xillix IP Core 32-розрядний CORDIC вимагає 3 тис. логічних елементів, що відповідає об'єму пам'яті 3 тис.*16 = 50 Кбіт. У разі використання такого об'єму пам'яті як ТПВ можна реалізувати функцію в межах дванадцяти розрядів, причому кожен наступний розряд буде збільшувати розмір таблиці більш як вдвічі, враховуючи збільшення розрядності даних, значення яких необхідно зберегти. Також зі збільшенням розмірів таблиці, з'являється необхідність у додатковій логіці, зазвичай, у вигляді мультиплексорів для вибору даних за заданою адресою, причому кількість таких елементів може бути співрозмірна з об'ємом самої таблиці.

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, своєю чергою, може змінюватись за рахунок розміру таблиці попередньої вибірки. Чим більший об'єм пам'яті кристалу, який ми можемо виділити для алгоритму, тим менша кількість тактів і логічних елементів, необхідних для функціонування алгоритму. Другим параметром, який змінювався при реалізації алгоритму, була розрядність помножувача. CycloneIII містить 56 блоків помножувачів, кожен з яких здатен здійснювати множення розрядністю 18×18, або два множення 9×9. Оптимальна розрядність помножувача становить

$$MultiplerBusSize = \frac{m}{2} + 1, \tag{12}$$

де m – кількість розрядів вхідного аргумента.

Для розрядності алгоритму 16 бітів однозначним рішенням буде розрядність помножувача 9. Для розрядності 32 біти вона становить 17, але при таблиці попередньої вибірки, що дорівнює трьом – розрядність потрібно збільшити до 18, згідно з формулою (11). При розрядності 24 біти, питання режиму роботи помножувачів є неоднозначним. Звичайно, що з погляду кількості зайнятих ресурсів необхідно використати помножувачі 18×18, але при таблиці попередньої вибірки, меншій за 5 розрядів, формула (11) дає близький до нуля результат, через що можна використовувати помножувачі в режимі 12×12, і параметри частоти в цьому випадку будуть відрізнятись. Тому для розрядності 24 розряди наведено три таблиці із конфігурацією помножувачів у режимі 12×12, 13×13, та 18×18. При експериментах з розрядністю помножувача було виявлено, що компілятор краще оптимізує проекти, в яких задіяні всі розряди помножувача, що однозначно визначило використання помножувачів 9×9 при 16 розрядах, у режимах, коли згідно з формулою (11) можна використати 8×8. Те саме стосується помножувачів 18×18 для 32 розрядів, у випадках коли можна було бра-

ти 17×17, чи навіть 16×16. Але для 24-х розрядів ефект виявився неоднозначний, хоча, наприклад, при ТПВ, що дорівнює п'ятьом, цей режим 18×18 продемонстрував кращі результати швидкодії, ніж у режимах помножувача 12×12 чи 13×13. Тому для об'єктивнішого зіставлення результатів у всіх випадках вибирався помножувач 18×18 або 9×9, залежно від розрядності. Щодо таблиць з розрядністю 12×12 чи 13×13, то вони присутні для демонстрації залежності швидкодії ресурсів та латентності залежно від зміни розрядності помножувачів.

При реалізації функції синуса-косинуса інтегрованими засобами Altera CORDIC magafunction не дає можливості вибрати параметри компіляції. Для CycloneIII можливою є лише 32-розрядна конвеєрна функція синуса чи косинуса. Обчислення здійснюються у форматі з плаваючою комою, де розмір мантиси становить 24 розряди, а значення порядку дорівнює восьми розрядам. Пропонований алгоритм був реалізований у середовищі Quartus 13.1.4 з конфігурацією 16, 24 та 32 розряди із різною розрядністю ТПВ. Для 32 розрядів кількість розрядів таблиці лежить у межах від 3 до 10 (рис. 1), для 16 розрядів – від 2 до 8 (рис. 2).

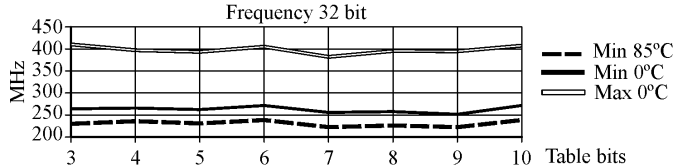

Рис. 1. Представлення частот у вигляді графіка для 32-х розрядів, залежно від розміру таблиці

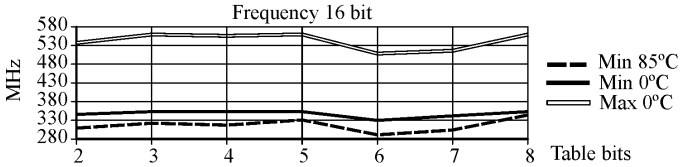

Рис. 2. Частота пропонованого алгоритму для 16 розрядів при ТПВ 2-8 бітів

Як видно з табл. 2, для 32-розрядного пропонованого алгоритму латентність може становити від 40 до 64 нс залежно від розміру таблиці попередньої вибірки. При значеннях таблиці 3 чи 4 розряди внутрішня пам'ять не задіюється, а пам'ять синтезується за допомогою логіки кристалу. У разі досягнення розміру таблиці 10 розрядів необхідно мати доступними 64 Кбіт пам'яті, що дає виграш як у латентності, так і ресурсоємності алгоритму. Коливання частоти для різних конфігурацій становить менше ±4 %. Для 16-розрядної реалізації алгоритму максимальна частота в більшості випадків обмежується швидкістю помножувача у режимі 9×9, хоча при значеннях таблиці попередньої вибірки 6, 7 з'являється відчутний провал швидкодії, зв'язаний швидше за все з недосконалістю компілятора. Для прикладу, при зміні кристалу з CycloneIII на CycloneIV з аналогічними параметрами цей провал відсутній за умови використання середовища 13.1.4. Якщо ж взяти цей же кристал CycloneIV у новішому середовищі 15.0.1, провал швидкодії тільки посилиться. Причому будь-які спроби оптимізації тільки погіршать становище. Так, найкращих результатів для середовища 15.0.1 було досягнуто при конфігурації компілятора по замовчуванню, причому дані результати були нижчі за результати провалу компілятора 13.1.4. Щодо 24-розрядного алгоритму (табл. 1), то максимальну частоту необхідно визначати для кожного випадку окремо, виходячи з розрядності таблиці та розрядності блоку помножувача. При 32-розрядній реалізації пропонованого алгоритму частоти лежать в межах 247,65 МГц (ТПВ = 3) – 276,4 МГц (ТПВ = 9), в обох випадках при конфігурації помножувача 18×18, що становить $^{\pm 5}$ %.

Табл. 1. Характеристики пропонованого алгоритму на платформі Artix-7 середовища Vivado 2015.2

Bus bit		Table bit	Latency clk	Total Time ns	Freq Best MHz	Period Best ns	Flip-flops	LUT	Mem LUT	DSP	POWER mW
Пропонований алгоритм	32	4	15	44,100	340,14	2,940	955	827	14	2	378
	32	6	13	38,181	340,48	2,937	826	710	16	2	364
	32	8	11	32,879	334,56	2,989	698	584	18	2	345
	24	5	10	28,630	349,28	2,863	477	381	13	2	278
Xilinx IP core	32	-	34	99,280	342,47	2,920	3588	3529	2	0	673
	24	-	26	67,600	384,62	2,600	2161	2113	4	0	499

Табл. 2. Характеристики пропонованого алгоритму на платформі CycloneIII середовища Quartus13.1.4

Bus 32 bit	Table bit	Latency clk	Total latency ns	Blocks used	Logic elements	Flip-flops	Memory bits	Multipliers	Freq. 85°Mhz	Freq. 0°Mhz	Freq. max	Power mW
Пропонований алгоритм	3	17	64,38	1,974	1,609	1,520	0	4	233,05	264,06	410,34	73,14
	4	16	60,19	1,852	1,494	1,444	0	4	237,47	265,82	397,3	73,13
	5	15	57,16	1,568	1,298	1,224	2,048	4	233,92	262,40	393,55	73,13
	6	15	55,23	1,441	1,165	1,131	4,096	4	241,60	271,59	405,19	73,12
	7	14	54,71	1,359	1,034	1,039	8,192	4	225,58	255,89	382,12	73,12
	8	13	50,43	1,214	905	948	16,384	4	229,25	257,80	395,41	73,11
	9	12	47,68	1,068	778	851	32,768	4	225,43	251,70	394,79	73,11
	10	11	40,48	945	670	763	65,536	4	240,21	271,74	407,66	73,1
Класичний Cordic	Speed	32	147,07	4326	4173	2900	0	0	217,58	246,79	378,20	
	Optimal	32	150,82	2889	2775	2011	608	0	212,18	237,08	364,30	
Altera Megafunction	Sin	36	263,49	5248	4996	2350	1362	31	136,63	152,91	243,66	70,27
	Cos	36	252,47	5056	4768	2247	320	31	142,59	158,15	246,12	70,26

Класична реалізація CORDIC засобами Vivado без масштабування кута вимагає здійснення $m+2$ тактів, де m – кількість розрядів вхідного кута. Розрядність алгоритму прямо пропорційна латентності та обернено пропорційна максимальній частоті, на якій здатний коректно працювати алгоритм. Так, для 16 розрядів класичного методу латентність становить 41,4 нс, а максимальна частота – 435 МГц. У разі досягнення максимальної розрядності 48 розрядів, латентність збільшується до 177,5 нс, а частота падає до 282 МГц. Бібліотека IP Core реалізована отже, що кількість вхідних і вихідних розрядів округлюється до величини, кратної вісім. Тому в нашому випадку доцільним буде зіставлення результатів для 24- та 32-розрядних версій алгоритму з пропонованим методом.

Латентність цих розрядностей становить 67,6 нс та 99,3 нс, а частота 385 МГц та 342 МГц відповідно. При порівнянні класичного алгоритму CORDIC з пропонованим алгоритмом варто зауважити, що метод залишкового множення на платформі Artix-7, так само як і на платформі CycloneIII буде обмежуватись частотою блоку помножувача, причому, враховуючи ефект збільшення частоти при зменшенні розрядності, на невеликих розрядностях частота алгоритму буде значно перевищувати частоту блоку помножувача. Так, максимальна частота помножувача для Artix-7 становить $350^{\pm1}$ МГц, за умови використання вбудованих у блок помножувача тригерів. Оскільки структура пропонованого алгоритму не передбачає використання даних тригерів, частота буде меншою від максимальної на декілька відсотків. Так, при розрядності 24 частота пропонованого алгоритму буде становити 349,3 МГц, що використовує практично всі ресурси помножувача, а це на 35 МГц менше за частоту класичного алгоритму. Проте застосування методу залишкового множення дає змогу замінити 12 класичних ітерацій лише 2-а ітераціями з використанням ресурсів блоку помножувача, хоча і з меншою тактовою частотою.

У разі збільшення розрядності до 32, тактова частота дещо падає – до величини 334-340 МГц, залежно від величини таблиці попередньої вибірки. Максимальна частота і надалі залежить від характеристик помножувача, але через збільшення кількості логічних елементів стає важчим для компілятора пошук оптимального способу їх розташування. Так, при ТПВ у 4-6 розрядів, тактова частота залишається стабільною в межах 340 МГц. При подальшому збільшенні таблиці до 8 розрядів тактова частота падає до 334,5 МГц. Як вже зазначено вище, падіння частоти з надлишком компенсується покращенням параметрів латентності, і при значенні 99,3 нс для класичного 32-розрядного методу її можна знизити до 44,1 нс для пропонованого методу з ТПВ у 4, чи навіть 32,9 нс при ТПВ, що дорівнює 8.

Прямо пропорційно до збільшення ТПВ зростає і об'єм необхідної для реалізації алгоритму пам'яті, але знижується кількість логічних елементів через зменшення кількості ітерацій методу. Так, кількість тригерів і логічних елементів для 24-розрядного пропонованого методу знизилась у 4,5 та 5,5 разів відповідно. 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 маже вдвічі, порівняно 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 з'явились у пропонованому методі, виявилось незначним і становить лише 2-3 % від загальної потужності кристалу.

Оптимізація алгоритму можлива лише за допомогою зміни настройок синтезу та імплементації, запропонованих виробником. Для синтезу схеми, ок-

рім профілю по замовчуванню, пропонується ще три: оптимізації швидкодії, площі та часу компіляції. Кожен профіль містить набір із 13 параметрів, значення яких подано в табл. 2. При пошуку оптимальної комбінації даних параметрів були апробовані різні комбінації зазначених параметрів. За результатами дослідження видно, що найкращих результатів можна досягти, при значенні параметрів оптимізації flatten=rebuild та directive=runtime (табл. 3).

Табл. 3. Оптимізація параметрів синтезу середовища Vivado

Parameter name	value					
flatten_hierarchy	off	full	rebuilt*			
gated_clock_conversion	off	on	auto			
bufg	0	12	24	100	1000	100 000
fanout_limit	0	100	400	1000	10 000	100 000
directive	default	area optimize	runtime optimize*			
fsm_extraction	off	one hot	sequencual	johnson	gray	auto
keep_equivalent_registers	off	on				
recource_sharing	off	on	auto			
control_set_opt_threshold	0	1	4	8	16	auto
no_lc	off	on				
shering_min_size	0	1	3	5	10	100
max_bram	-10	-1	0	1	10	100
max_dsp	-10	-1	0	1	10	100
cascade_dsp	tree	force	auto			

Таблиця попередньої вибірки у середовищі Vivado завжди формується монтажним чином. Відповідно, швидкодія і кількість логічних елементів залежить від значень вибраних констант у таблиці. Взявши до уваги, що всі значення таблиці є округленим двійковим представленням ірраціональних чисел, стає зрозумілим, що залежно від настройок заокруглення та варіантів їхніх комбінацій, можливі конфігурації таблиці можуть відрізнятись залежно від пріоритетів точності чи швидкодії обчислень. Експериментально встановлено, що зміна способів представлення даних таблиці можуть відчутно впливати на характеристики вихідного проекту. Отже, всі основні параметри алгоритму, зокрема спосіб розміщення його на кристалі, можуть залежати від значень параметрів ТПВ.

Висновки. Пропонований алгоритм дає змогу спростити реалізацію методу перекодування кута у його програмній та апаратній реалізації, а також отримати виграш порівняно із вбудованими бібліотечними функціями, пропонованими розробниками середовищ. У випадку використання платформи Xilinx, виграш у латентності пропонованого алгоритму збільшується більш як у 2 рази, при втраті тактової частоти всього на декілька мегагерц. Шляхом ширшого використання пам'яті та помножувачів вдається отримати економію як ресурсів кристалу, так і споживаної потужності від 2 до 7 раз. Приблизно таку ж економію ресурсів можна отримати і на платформі від Altera, причому як для класичної реалізації алгоритму CORDIC, так і для вбудованої мегафункції.

Література

1. Volder J.E. The CORDIC Trigonometric Computing Technique / J.E. Volder // IEEE Transactions on Electronic Computers, Sep. – 1959. – Vol. EC-8. – No. 3. – Pp. 330-334.

2. Walther J.S. A unified algorithm for elementary functions / J.S. Walther // in Proc. AFIPS Conf. – 1971. – Vol. 38. – Pp. 385-389.

3. 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.

4. Madisetti A. A 100 MHz, 16-b, direct digital frequency synthesier with 100-dBc supurious-free dynamic range / A. Madisetti, A.Y. Kwentus, A.N. Willson // IEEE Journal of Solid-State Circuits. – 1999. – Vol. 34, № 8. – Pp. 1034-1043.

5. Kuhlmann M. P-CORDIC : A Precomputation Based Rotation / M. Kuhlmann and K.K. Parhi // EURASIP Journal on Applied Signal Processing. – 2002. – Vol 50. – No. 1. – Pp. 936-943, 2002. [Electronic resource]. – Mode of access http://dx.doi.org/10.1155/S1110865702205028.

6. Juang T.-B. Para-CORDIC: Parallel CORDIC Rotation Algorithm / T.-B. Juang, S.-F. Hsiao, and M.-Y. Tsai // IEEE Transactions on Circuits and Systems I: Regular Papers. – 2004. – Vol. 51, No. 8. – Pp. 1515-1524, Aug.

7. Juang T. Low Latency Angle Recoding Methods for the Higher Bit-Width Parallel CORDIC Rotator Implementations / T. Juang // IEEE Transactions on Circuits and Systems II: Express Briefs. – 2008. – Vol. 55. – No. 11. – Pp. 1139-1143, Nov.

8. Moroz Leonid. Simple Hybrid Scaling-Free CORDIC Solution for FPGAs " / Leonid Moroz, Shinobu Nagayama, Taras Mykytiv, Ihor Kirenko, Taras Boretskyy // International Journal of Reconfigurable Computing. – 2014. – Vol. Article ID 615472, 4 pages, 2014. [Electronic resource]. – Mode of access http://dx.doi.org/10.1155/2014/615472.

9. Мороз Л.В. Теорія та швидкодіючі апаратно-програмні засоби ітераційних методів обчислення функцій : автореф. дис. на здобуття наук. ступеня д-р техн. наук / Л.В. Мороз; НУ "Львівська політехніка". – Львів, 2013. – 40 с.

10. Мороз Л.В. Ітераційні формули для CORDIC-методу / Л.В. Мороз // Комп'ютерні технології друкарства : зб. наук. праць Української академії друкарства. – Львів : Вид-во УкрАД. – 2012. – № 28. – С. 111-120.

11. Мороз Л.В. 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. праць. – Львів : РВВ НЛТУ України. – 2015. – № 5. – С. 292-301.

12. Мороз Л.В. Перекодування кута CORDIC-методу / Л.В. Мороз // Комп'ютерні технології друкарства. : зб. наук. праць Української академії друкарства. – Львів : Вид-во УкрАД. – 2015. – № 33. – С. 51-55.

13. Мороз Л.В. Модифікований CORDIC-методу обчислення синуса-косинуса / Л.В. Мороз, Т.Р. Борецький, Т.Л. Луковський // Комп'ютерні технології друкарства : зб. наук. праць Української академії друкарства. – Львів : Вид-во УкрАД. – 2015. – № 33. – С. 56-63.

14. Greg Dr. Tumbush, Starkey Labs, Colorado Springs, CO / Dr. Greg // Signed Arithmetic in Verilog 2001 – Opportunities and Hazards 5 p. [Electronic resource]. – Mode of access http://www.uccs.edu/~gtumbush/published_papers/Tumbush%20DVCon%2005.pdf

15. [Electronic resource]. – Mode of access http://www.xilinx.com/products/silicon-devices/fpga/artix-7.html

16. [Electronic resource]. – Mode of access http://www.xilinx.com/support/documentation / ip_documentation/cordic/v6_0/pg105-cordic.pdf

Мороз Л.В., Борецкий Т.Р., Костив Ю.М. **Синусо-косинусный FPGA-вычислитель на основе CORDIC-метода перекодировки угла**

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 том числе в виде мегафункции, с помощью которой существенно уменьшается количество тактов, латентность, количество необходимых блоков и увеличивается минимальная тактовая частота.

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

Moroz L.D., Boretskyy T.R., Kostiv U.M. **Sine-cosine CORDIC-based Method of Angle Transcoding for FPGA**

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

Keywords: CORDIC, IP Core, FPGA, algorithm, latency, megafunction.

УДК 004.9

Проф. В.В. Пасічник[1], д-р техн. наук;
доц. О.І. Артеменко[2], канд. техн. наук

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ МОДЕЛЮВАННЯ ПРОСТОРОВОГО РОЗВИТКУ ТУРИСТИЧНОЇ ІНФРАСТРУКТУРИ

Проаналізовано дослідження та розробки в галузі інформаційних технологій для галузі туризму та моделювання процесів розвитку урбаністичної інфраструктури. Сформовано класифікацію інформаційних технологій для різних ланок туристичної галузі. Обґрунтовано доцільність поєднання інтелектуальних технологій аналізу даних з функціональними можливостями геоінформаційних систем для розв'язання задач підтримки прийняття рішень та прогнозування розвитку туристичної інфраструктури регіону. Показано, що актуальною є потреба в розробленні інформаційної технології, яка дасть змогу аналізувати просторовий розподіл рекреаційних ресурсів регіону та сприяти прийняттю науково обґрунтованих рішень щодо інвестування в туристичну інфраструктуру регіону.

Ключові слова: туризм, інформаційні технології, туристична інфраструктура, моделювання просторового розвитку об'єктів, урбанізація.

Вступ. Туристичний бізнес – це один з небагатьох видів господарської діяльності, який має високу рентабельність та акумулює значні трудові ресурси **[1]**. За прогнозами експертів, незважаючи на глобальну економічну кризу, відбуватиметься щорічне зростання туристичної галузі на 2-5 % **[2]**. Річні надходження від світової туристичної галузі становлять понад трлн дол. США **[3]**. У цих умовах Україна повинна використовувати наявні можливості для збільшення власних доходів від туризму.

Для реалізації програми розвитку туризму в Україні **[4]** передусім необхідна матеріально-технічна база з розвиненою туристичною інфраструктурою, особливо це стосується регіонів. Розвиток туристичної інфраструктури неможливий без впровадження технологічних інновацій.

Мета і задачі дослідження. Актуальність роботи полягає у вивченні стану та можливостей використання інформаційних технологій моделювання для прогнозування просторового розвитку туристичної інфраструктури регіону.

Метою дослідження є аналіз стану та виявлення можливих напрямів розвитку інформаційних технологій моделювання просторового розвитку в ту-

[1] НУ "Львівська політехніка";
[2] ПВНЗ "Буковинський університет", м. Чернівці