УДК 681.317
©Л. Мороз[4], 2015

ПЕРЕКОДУВАННЯ КУТА ДЛЯ CORDIC – МЕТОДУ

Подано математичний опис нового методу перекодування вхідного кута, який дає змогу гнучко змінювати об'єм пам'яті таблиці та число ітерацій методу CORDIC.
Ключові слова: ітерація, CORDIC–метод, обчислення, перекодування коду кута.

The paper presents the mathematical description of a new method of recoding of input angle which enables to change flexibly the size of the table memory and the number of iterations method of CORDIC.
Keywords: iteration, CORDIC-method, calculation, angle recoding.

1. ВСТУП

У даній роботі запропоновано новий метод перекодування вхідного кутапри обчисленнісинуса та косинуса.

2. АНАЛІЗ ВІДОМИХ АПАРАТНИХ РІШЕНЬ МЕТОДУ CORDIC

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:

$$x_i = x_{i-1} - \sigma_i 2^{-i} y_{i-1};$$
$$y_i = y_{i-1} + \sigma_i 2^{-i} x_{i-1}; (1)$$
$$z_i = z_{i-1} - \sigma_i \arctan(2^{-i});$$
$$\sigma_i = sign(z_{i-1}), i = 1 \dots m,$$

де m- число двійкових розрядів обчислювача.

З метою спрощення апаратної реалізації обчислювача запропоновано метод CORDIC з перекодуванням вхідного кута [4,5,6,7], 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

---

[4] Національний університет "Львівська політехніка"

використовують послідовно три методи: табличний + CORDIC + кусково - лінійна апроксимація [3, 4, 8, 9]. Причому CORDIC виконаний у різних варіантах – класичному [3], з використанням ітераційних формул вищих порядків [8,9,10], без перекодування та з перекодуванням вхідного кута [4]. Найпростішим з точки зору апаратного втілення є CORDICз перекодуванням вхідного кута [4]. Однак він має суттєвий недолік – великий об'єм пам'яті (типу LUT) при великих значеннях m (необхідна таблиця розміром, не меншим, ніж $2^{m/3} \cdot m$ бітів). Крім того, вихідні помножувачі тут реалізовані у базисі $\{-1,1\}$, що унеможливлює використання помножувачів, які є у складі блоків DSP сучасних ПЛІС. Метою даної роботи є опис нового методу перекодування вхідного кута, який дає змогу гнучко змінювати об'єм пам'яті таблиці та число ітерацій CORDICа.

3. ОПИС ПРОПОНОВАНОГО МЕТОДУ

Нехай маємо двійкове $m-$ розрядне представлення кута φ:

$$\varphi = \sum_{i=1}^{m} a_i 2^{-i} \quad , \qquad (2)$$

причому

$$a_i = \{0,1\}, \varphi \in [0,\pi/4].$$

Необхідно знайти функції $\sin(\varphi)$ та $\cos(\varphi)$, тобто повернути одиничний вектор на кут φ.

Подамо φу вигляді:

$$\varphi = \varphi_{var} + \Delta. \qquad (3)$$

Тут φ_{var} - кут повороту, отриманий у результаті перекодування кута φ, а Δ - кут відставання, яке з'явилось саме через перекодування. Перекодування кута φ можна описати такими рівняннями:

$$\varphi_{var} = \varphi_r + \varphi_{const}, \qquad (4)$$

де

$$\varphi_r = \sum_{i=2}^{m+1} b_i \cdot \arctan(2^{-i}), b_i = 2a_{i-1} \quad (5)$$

Після очевидних перетворень отримаємо, що

$$\varphi_{var} = \sum_{i=2}^{m+1} a_{i-1} \cdot \arctan(2^{-i}) = \sum_{i=1}^{m} 2a_i \cdot \arctan(2^{-i-1}), \quad (6)$$

$$\varphi_{const} = \sum_{i=2}^{m+1} \arctan(2^{-i}) = \sum_{i=1}^{m} acrtan(2^{-i-1}) \ . \qquad (7)$$

Для повороту одиничного вектора на кут φ_{var} застосуємо метод CORDIC з такими ітераційними рівняннями:

$$x_i = x_{i-1} - b_i y_{i-1} \cdot 2^{-i-1};$$
$$y_i = y_{i-1} - b_i x_{i-1} \cdot 2^{-i-1}; \qquad (8)$$
$$b_i = 2a_i - 1;$$
$$i = \overline{1,m};$$
$$x_0 = P_c \cos(\varphi_{const});$$
$$y_0 = P_c \sin(\varphi_{const});$$

$$P_c = \frac{1}{\prod_{i=1}^{m}\sqrt{1+2^{-2i-2}}}.$$

У результаті отримаємо синус -y_mта косинус - x_mкутаφ_{var}.

Кут відставання Δзнаходиться за такими формулами:

$\Delta = \varphi - \varphi_{var} = \sum_{i=1}^{m} a_i d_i,$

$$d_i = 2^{-i} - 2\arctan(2^{-i-1}).$$

Особливістю такого представлення кута відставання є те, що він завжди додатній або рівний нулю. Зауважимо, що $\Delta -$ це кут, на який потрібно додатково повернути вектор $\{x_{m+1}, y_{m+1}\}$, що маємо після завершення ітерацій за методом CORDIC. Отримані в результаті такого додаткового обертання значення x та y і будуть шуканими функціями косинуса і синуса кута φ:

$$x = x_m\cos(\Delta) - y_m\sin(\Delta), \qquad (9)$$

$$y = y_m\cos(\Delta) + x_m\sin(\Delta).$$

Оцінимо число ітерацій, в межах яких слід брати до уваги значення Δ. Для цього поставимо умову:

$$d_i \le 2^{-m},$$

тоді

$$d_i = 2^{-i} - 2\arctan(2^{-i-1}) \approx 2^{-i} - 2(2^{-i-1} - 2^{-3i-3}/3) = 2^{-m};$$

звідки знайдемо, що

$$i_a = \left\lceil \frac{m-2-log_2 3}{3} \right\rceil. (10)$$

Саме при такому $i = i_a$найбільша складова кута відставання завжди буде меншою від 2^{-m}, тобто $d_{i_a} \le 2^{-m}$.

Нижче наведена таблиця значень i_a для різних m.

Таблиця 1

m=16	$i_a = 5$
m=24	$i_a = 7$
m=32	$i_a = 10$
m=48	$i_a = 15$
m=54	$i_a = 17$
m=64	$i_a = 21$

Звідси випливає, що Δ слід обчислювати лише умежах значень i:

$$\Delta = \sum_{i=1}^{i_{a-1}} a_i d_i.$$

Якщо ж в обчислювачі використовуються вихідні помножувачі для кусково - лінійної апроксимації, то необхідно, щоб найбільша складова d_i кута відставання Δ не перевищувала значення

$d_i \le 2^{-\frac{m}{2}}$,(11)

або

$$2^{-i} - 2\arctan(2^{-i-1}) \leq 2^{-\frac{m}{2}}.$$

Тоді неважко знайти, що

$i_b = \left\lceil \frac{m-4-2\log_2 3}{6} \right\rceil$.(12)

У таблиці 2 наведені значення i_b для різних m.

Таблиця 2

m=16	$i_b = 2$
m=24	$i_b = 3$
m=32	$i_b = 5$
m=48	$i_b = 7$
m=54	$i_b = 8$
m=64	$i_b = 10$

Отже, біти вхідного кута обробляються тоді таким чином: старші з номерами $i1 = \overline{1, \iota_{LUT}}$, де $i_{LUT} = \overline{\iota_b - 1, \ldots, \iota_a - 1}$- за рахунок використання пам'яті, під CORDIC відводяться біти з номерами $i2 = \overline{\iota_{LUT} + 1, m/2}$, а під кусково - лінійну апроксимацію – наймолодші бітиз номерами $i3 = \overline{\frac{m}{2} + 1, m}$.

4. ВИСНОВКИ

Описаний новий метод перекодування вхідного кута, який дає змогу гнучко змінювати обсяг пам'яті таблиці та число ітерацій CORDICа.

1.J. E. Volder, "The CORDIC Trigonometric Computing Technique," IEEE Transactions on Electronic Computers, vol. EC-8, no. 3, pp. 330–334, Sep. 1959.2.J.S. Walther, "A unified algorithm for elementary functions", in Proc. AFIPS Conf., vol. 38, 1971, pp. 385-389. 3.Л.В.Мороз, Я.І.Грабовський, Т.М.Микитів, Т.Р.Борецький, Ю.М.Костів, С.С.Войтусік. Швидкодіючий гібридний CORDIC-обчислювач тригонометричних функцій. Науковий вісник НЛТУ України. – 2014. – Вип. 24.8, с.352-357. 4.A. Madisetti, A.Y.Kwentus, A.N.Willson. A 100 MHz, 16-b, directdigitalfrequencysynthesierwith 100-dBcsupurious-freedynamicrange. IEEE Journal of Solid-State Circuits. –vol.34,–№8,– 1999, – pp.1034–1043. 5.M. Kuhlmann and K. K. Parhi, "P-CORDIC : A Precomputation Based Rotation," EURASIP Journal on Applied Signal Processing, vol. 2002, no. 1, pp. 936–943, 2002. Available:http://dx.doi.org/10.1155/S1110865702205028.6. T.-B. Juang, S.-F. Hsiao, and M.-Y. Tsai, "Para-CORDIC: Parallel CORDIC Rotation Algorithm," IEEE Transactions on Circuits and Systems I: Regular Papers, vol. 51, no. 8, pp. 1515–1524, Aug. 2004. 7. T. Juang, "Low Latency Angle Recoding Methods for the Higher Bit-Width Parallel CORDIC Rotator Implementations," *IEEE Transactions on Circuits and Systems II: Express Briefs, vol. 55, no. 11, pp. 1139–1143, Nov. 2008. 8.Leonid Moroz, Shinobu Nagayama, Taras Mykytiv, Ihor Kirenko, Taras Boretskyy / "Simple Hybrid Scaling-Free CORDIC Solution for FPGAs " // International Journal of Reconfigurable Computing, vol. 2014, Article ID 615472, 4 pages, 2014. http:// dx.doi.org/10.1155/2014/615472.9. Л.В.Мороз. Теорія та швидкодіючі апаратно-програмні засоби ітераційних методів обчислення функцій. Автореф. дис. д.т.н.,Львів,2013. 10. Мороз Л.В. Ітераційніформулидля CORDIC-методу. Збірник наукових праць Української академії друкарства "Комп'ютерні технології друкарства". – 2012. – № 28. – с. 111-120.*