УДК 681.3
© Т. Микитів[1], Л. Мороз[9], 2013

ОБЧИСЛЕННЯ ФУНКЦІЇ ГІПЕРБОЛІЧНОГО ТАНГЕНСА З ДОПОМОГОЮ МЕТОДУ CORDIC

Розглянуто різноманітні методи обчислення функції гіперболічного тангенса, описано переваги та недоліки кожного з методів. Запропоновано новий метод обчислення на основі модифікованих ітераційних формул методу CORDIC. Проаналізовано шляхи вдосконалення алгоритму, зокрема, застосування пропуску частини ітерацій, використання пам'яті на початковому етапі роботи та кусково-лінійну апроксимацію. Наведено та проаналізовано результати моделювання запропонованого та найвідоміших методів обчислення гіперболічного тангенса на мікроконтролері.

This paper presents various methods for calculating the hyperbolic tangent function, describes the advantages and disadvantages of each method. A new method of calculation, based on the modified CORDIC. The ways of improving the algorithm, including using a linear approximation of the pass iterations and memory usage in the initial phase of work. Presented and analyzed the results of simulation on the microcontroller, and the most famous of the proposed methods calculate the hyperbolic tangent

1. ФУНКЦІЇ АКТИВАЦІЇ В НЕЙРОННИХ МЕРЕЖАХ.

Гіперболічні функції широко застосовуються у різних обчислювальних процесах. Зокрема, вони використовуються у нейронних мережах, де однією з найпоширеніших сигмоїдних функцій активації [4, 5] є залежність:

$$f(Y) = \tanh\left(\frac{Y}{\alpha}\right), \tag{1}$$

де Y – вхідний сигнал, α – параметр що впливає на кривизну сигмоїдної функції. Крім тангенса, є також функції активації типу (2) та їх модифікації.

$$f(Y) = \frac{1}{1+e^{-\alpha Y}}, \; f(Y) = e^{-\alpha Y}. \tag{2}$$

Огляд методів обчислення функції гіперболічного тангенса

Серед усього різноманіття методів обчислення гіперболічного тангенса [2, 4, 5, 6, 7, 8] виділимо наступні. Одним з них є розклад функції

[9] Національний університет «Львівська політехніка»

в ряд Тейлора. Наприклад, функцію гіперболічного тангенса можна наблизити з допомогою ряду [4]:

$$\tanh(x) \approx x - \frac{x^3}{3} + \frac{2x^5}{15} - \frac{17x^7}{315} + ... \quad (3)$$

Недолік такого методу у надзвичайно великій похибці, яка стрімко зростає при віддаленні аргументу x від початку координат. На практиці цей метод, як правило, не використовується.

Також, дуже часто при обчисленнях функцію тангенса виражають через експоненти, які легше апроксимувати [2]:

$$\tanh(x) = \frac{e^{2x} - 1}{e^{2x} + 1} \quad (4)$$

або [12]:

$$\tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \quad (5)$$

Ряд Тейлора для функції e^x наступний:

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!} + \cdots + \frac{x^n}{n!} \quad (6)$$

аналогічно, для e^{2x}:

$$e^{2x} = 1 + 2x + \frac{4x^2}{2!} + \frac{8x^3}{3!} + \frac{16x^4}{4!} + \frac{32x^5}{5!} + \frac{64x^6}{6!} + \cdots + \frac{2^n x^n}{n!} \quad (7)$$

Реалізація формул (5)-(7) апаратним методом є достатньо складною і затратною, навіть застосовуючи схему Горнера для обчислення поліномів.

Цікавий метод обчислення показникової функції на ПЛІС запропоновано у роботі [3]. Автори пропонують таблично-алгоритмічний метод обчислення зі спеціальною конвеєрною системою помножувачів, що дозволяє максимально швидко отримати результат, при цьому використовуючи незначну кількість блоків множення. Аргумент x подається як сума константних значень X_k, які є розкладом x на суму двійкових розрядів, поділених на групи:

$$x = X_1 + X_2 + \cdots + X_k = \sum_{i=1}^{k} X_i \,. \quad (8)$$

Потім використавши властивість показникової функції, необхідно перемножити значення e^{X_i}, завантажені з таблиці:

$$e^x = e^{X_1} \cdot e^{X_2} \cdot \cdots \cdot e^{X_k} \,. \quad (9)$$

Одними з найкращих є методи наближення з допомогою поліномів. Наприклад, використавши поліном першого степеня [7] та поділивши відрізок $x \in [0;\quad 1]$ на два підінтервали, можна добитися хорошого результату за точністю обчислень:

$$e^x \approx \begin{cases} 1{,}29744 \cdot \mathrm{x} + 0{,}97980; & \mathrm{x} \in [0;\quad 0{,}5], \quad \Delta = 0{,}02020 \\ 2{,}13912\mathrm{x} + 0.54585; & \mathrm{x} \in [0{,}5;\quad 1], \quad \Delta = 0{,}03331 \end{cases}, \qquad (10)$$

тут Δ – абсолютна похибка наближення.

Для забезпечення 16 точних біт результату ($\Delta \le 1.53 \cdot 10^{-5}$) на відрізку $x \in [0;\quad \ln(2)]$, достатньо використати поліном третього степеня та поділити інтервал вхідних значень на два відрізки:

$$e^x \approx \begin{cases} 0{,}9999944 + 1{,}0005078 \cdot x + 0{,}49282376 \cdot x^2 + 0{,}19864761 \cdot x^3, & x < \ln(2)/2 \\ 0{,}9958470 + 1{,}0330672 \cdot x + 0{,}40486968 \cdot x^2 + 0{,}28092986 \cdot x^3, & x \ge \ln(2)/2 \end{cases} \qquad (11)$$

Для наближення функції тангенса гіперболічного використовують метод CORDIC, що працює у гіперболічній системі кординат [8]. Одним із варіантів є знаходження частки від ділення функцій $\sinh(x)$ та $\cosh(x)$:

$$\tanh(x) = \frac{\sinh(x)}{\cosh(x)} . \qquad (12)$$

Метод CORDIC дозволяє також обчислити значення функцій e^{φ} та $e^{-\varphi}$, зробивши заміну $\varphi = 2x$, можна застосувати цей метод для обчислення функції тангенса гіперболічного, використавши формули (4) або (5).

2. МЕТОД ОБЧИСЛЕННЯ ФУНКЦІЙ e^x ТА e^{-x} БЕЗ ДЕФОРМАЦІЇ МОДУЛЯ ВЕКТОРА.

Як відомо, ітераційні формули класичного методу CORDIC для обчислення функції e^{φ} є наступними [9]:

$$\begin{aligned} q_{i+1} &= q_i + d_i \cdot q_i \cdot 2^{-i} \\ z_{i+1} &= z_i - d_i \cdot \alpha_i \end{aligned}, \qquad (13)$$

тут $d_i = sign(q_i)$ – напрямок повороту вектора; $\alpha_i = \operatorname{arctanh}(2^{-i})$ – таблиця кутів поворотів.

При початкових умовах для (13):

$$q_1 = 1; \quad z_1 = \varphi; \quad i = 1, \tag{14}$$

після виконання m ітерацій, отримаємо:

$$\begin{cases} q_m \approx K_m \cdot e^x \\ z_m \approx 0 \end{cases}, \tag{15}$$

тут K_m – коефіцієнт деформації вектора. Слід зауважити, що деякі ітерації повторюються з метою забезпечення умови збіжності ітераційного процесу [8].

Є декілька методів для усунення коефіцієнта деформації вектора, наприклад, корекція початкового значення $q_1 = 1/K_m$. В такому випадку $q_m \approx e^x$.

Альтернативним методом, який також усуває коефіцієнт деформації, є використання таблиці кутів елементарних поворотів α'_i такого виду [10]:

$$\alpha'_i = \ln(1 + d_i 2^{-i}). \tag{16}$$

У роботі [13] було запропоновано використати (17), яка є простішою у порівнянні з (16):

$$\alpha'_i = \ln(1 + 2^{-i}). \tag{17}$$

У випадку (17) обчислення виконуються без деформації вектора. Це також дозволяє прискорити роботу алгоритму за рахунок пропуску частини ітерацій. Додатково прискорити алгоритм можна і за рахунок використання пам'яті (переглядових таблиць) та кусково-лінійної апроксимації.

Проте модифікація таблиці кутів α'_i згідно формули (17) порушує умову збіжності ітераційного процесу. Можливим рішенням є виконання подвійних ітерацій, однак це ускладнює реалізацію алгоритму та сповільнює його швидкодію. Автори пропонують свій варіант реалізації, який подано нижче.

3. ЗАПРОПОНОВАНИЙ МЕТОД ОБЧИСЛЕННЯ ЕКСПОНЕНТИ

Нехай необхідно обчислити функцію e^{φ} ($\varphi \in [0; \ln(2)]$) з точністю m двійкових розрядів. Тоді рекурентні рівняння будуть наступними:

$$\begin{aligned} q_{i+1} &= q_i + q_i \cdot 2^{-i} \\ z_{i+1} &= z_i - \alpha'_i \end{aligned} \tag{18}$$

тут $\alpha_i' = \ln(1 + 2^{-i})$. При цьому, початкові умови (14). Для обчислення функції $e^{-\varphi}$ ітераційний метод (18) набуде вигляду:

$$\begin{aligned} q_{i+1} &= q_i - q_i \cdot 2^{-i} \\ z_{i+1} &= z_i - \alpha_i' \end{aligned}, \tag{19}$$

де $\alpha_i' = -\ln(1 - 2^{-i})$. Позначимо максимальну абсолютну похибку обчислень як:

$$\mathrm{Ref} = 2^{-m}. \tag{20}$$

Як відомо з кожною наступною ітерацією $z_i \to 0$, при цьому $q_i \to e^{\varphi}$. Причому на кожному кроці $i > m/2$ точність значень z_i та q_i є приблизно однаковою. Тому для оцінки абсолютної похибки результату q_i можна скористатися абсолютним значенням залишкового кута z_i. Тобто, для забезпечення заданої точності обчислень повинна виконуватися умова:

$$z_i < \mathrm{Ref}\ . \tag{21}$$

Окрім цього, застосуємо метод одностороннього повороту, де ітерації проводяться тільки за умови:

$$z_i \geq \alpha_i'\ . \tag{22}$$

Блок-схема запропонованого алгоритму, для обчислення $e^{-\varphi}$, наведена на рис.1.

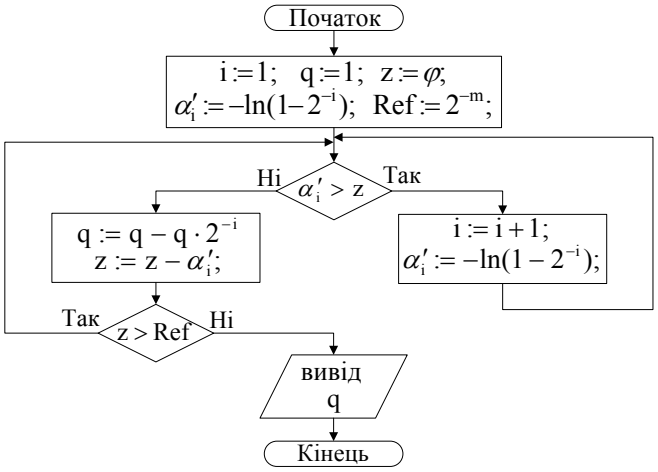

Рис. 1. Блок-схема запропонованого алгоритму обчислення $e^{-\varphi}$

4. ПІДВИЩЕННЯ ШВИДКОДІЇ МЕТОДУ

Як вже згадувалось, підвищити швидкодію запропонованого методу можна за рахунок:

Використання пам'яті;

Застосування методу залишкового множення (кусково-лінійна апроксимація).

При поєднанні першого та другого підходів алгоритм складається з наступних кроків:

1) Шляхом аналізу старших k розрядів вхідного кута φ завантажуємо початкове значення q_k з пам'яті;

2) Виконуємо ітерації запропонованим методом до моменту досягання половини встановленої точності. Для цього визначимо $\mathrm{Ref} = 2^{-m/2}$.

3) Застосуємо метод залишкового множення, модифікувавши отриманий результат за формулою:

$$q_m = q_{m/2} + p \cdot q_{m/2} \cdot z_{m/2} \tag{23}$$

тут $p = 1$ для обчислення функції e^{φ} та $p = -1$ – для $e^{-\varphi}$ відповідно.

У результаті отримаємо $q_m \approx e^{\varphi}$ або $q_m \approx e^{-\varphi}$ з точністю m двійкових розрядів.

При цьому, слід звернути увагу на обмеження вхідного значення φ [11]:

$$\varphi \in [0; \ln(2)]. \tag{24}$$

Обійти це обмеження можна, застосувавши формулу:

$$e^{-T} = 2^n \cdot e^{-\varphi} \tag{25}$$

тут числа T, φ – дійсні, а n – ціле. Відповідно:

$$n = \lfloor -T / \ln(2) \rfloor \text{ та } \varphi = T + n \cdot \ln(2). \tag{26}$$

Таким чином, перед початком обчислень, вхідне значення T необхідно перетворити у відповідне значення кута φ, що подається на вхід алгоритму та значення n, яке показує, на скільки двійкових розрядів необхідно зсунути отриманий результат.

5. РЕЗУЛЬТАТИ МОДЕЛЮВАННЯ

Для визначення найкращих методів обчислення за швидкодією, було проведено їх детальний аналіз. Оцінка здійснювалася за такими критеріями як приблизна кількість операції (додавання/віднімання, множення/ділення) та з урахуванням особливостей апаратної платфо-

рми (мікроконтролера). Найкращі результати, отримані шляхом аналізу алгоритмів, виявились у мінімаксного наближення, тому саме його обрано для порівняння швидкодії та точності із запропонованим методом.

Було змодельовано наступні варіанти обчислення гіперболічного тангенса:

1. Запропонований метод (на основі обчислення e^{-x} з допомогою алгоритму (див. рис.1));

2. Метод обчислення поліномами (на основі наближення e^{-x} поліномами третього степеня за формулами (11)).

Моделювання проводилось на 32-х бітному мікроконтролері ST_F100xxx побудованому на архітектурі ARM. Для отримання максимально достовірних результатів, кожен алгоритм було поставлено у рівні умови. Програмна реалізація методів написана на С без використання асемблерних вставок. При цьому для усього проекту задано максимальний рівень оптимізації за швидкодією. Моделювання функції tanh(x) проводилось для проміжку $x \in [0;\ \ 7]$. Оскільки діапазон вхідних значень є достатньо великим, табуляція функцій виконувалася з кроком $h = 2^{-10} \approx 0{,}001$.

Встановлена точність – 16 біт результату. Перетворення аргументів та відновлення результатів здійснювалося за формулами (25), (26). При значенні аргументу $x \geq 5{,}9$ приймалося, що $\tanh(x) \approx 1$, при цьому забезпечувалася необхідна точність результату.

Результати моделювання наведено на рис. 2-3.

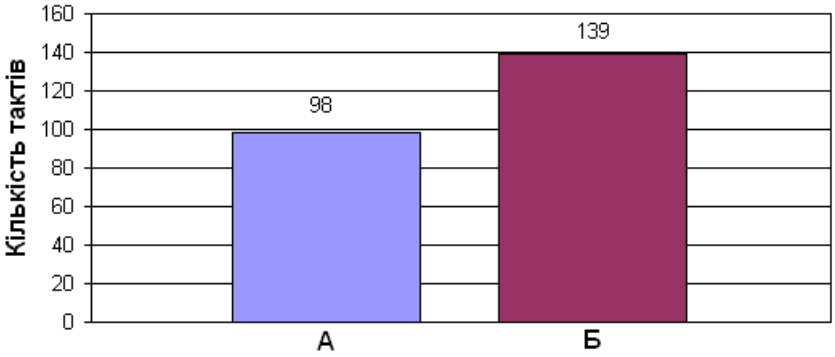

Рис. 2. Середня кількість тактів (А – запропонований метод; Б – метод обчислення поліномами)

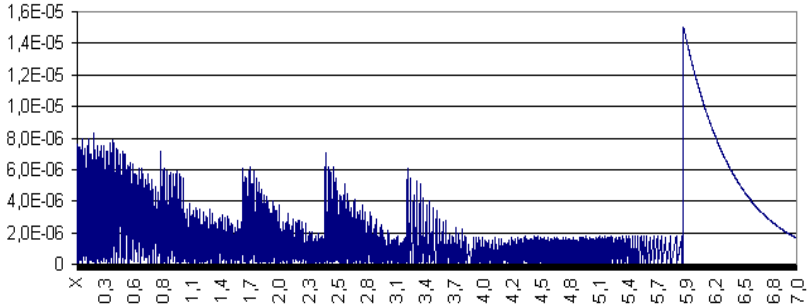

Рис. 3. Модуль абсолютної похибки. Запропонований метод обчислення функції tanh(x)

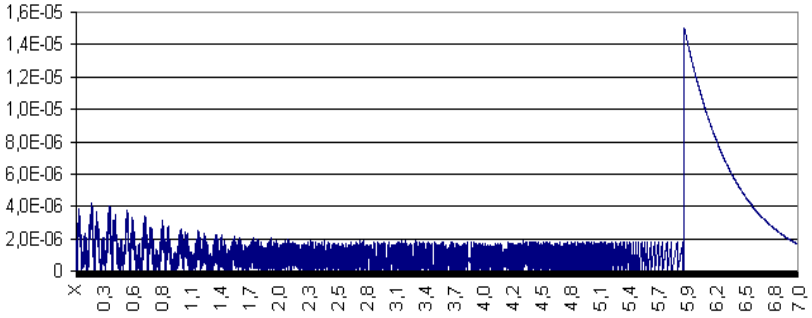

Рис. 4. Модуль абсолютної похибки. Наближення поліномами функції tanh(x)

6. ВИСНОВОК

Запропонований метод обчислення функції тангенса гіперболічного є швидшим (приблизно у 1,4 рази) у порівнянні з мінімаксним наближенням поліномами, що дозволяє прискорювати обчислення без заміни апаратної платформи. Окрім цього, отримані результати дають змогу прискорити обчислення експоненціальної функції та деяких найпоширеніших функцій активації у нейронних мережах.

1. Huajun Z., Jin Z. Direct Exponential Function Computer in Neural Network Application Based on Evolvable Hardware. Information Technology Journal, 7, 2008: 1156-1162pp. 2. Avcı M., Yıldırım T. Generation of tangent hyperbolic sigmoid function for microcontroller based digital implementations of neural networks. International XII. Turkish Symposium on Artificial Intelligence and Neural Networks – TAINN 2003 3. Попов С.Д., Опадчий Ю.Ф. Вычисление показательной функции на ПЛИС. Современные проблемы науки и образования. – 2013. – №5 4. Motcha P., Mary Rathi Pushpa, Manimala K. Implementation of Hyperbolic Tan-

gent Activation Function in VLSI. International Journal of Advanced Research in Computer Science & Technology (IJARCST 2014), Vol. 2, Jan-March 2014 5. Namin A.H., Leboeuf K., Muscedere R., Wu H., Ahmadi M. Efficient hardware implementation of the hyperbolic tangent sigmoid function. Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS'09), Taiwan, 2009, 2117-2120pp. 6. Maicon A. Sartin, Alexandre C. R. da Silva. Approximation of Hyperbolic Tangent Activation Function Using Hybrid Methods. Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC), 2013 8th International Workshop, July 2013. 7. Frank. J. Testa. Floating Point Math Functions // Microchip Technology Inc, 1997. 8. Andraka R. A survey of CORDIC algorithms for FPGA based computers. ACM/SIGDA 6th International Symposium on FPGAs, 1998, 1981–2000pp. 9. Pottathuparambil R., Sass, R. Implementation of a CORDIC-based Double-Precision Exponential Core on an FPGA. Proceedings of RSSI 2008, Urbana, Illinois, USA, 2008. 10. Байков В.Д., Смолов В.Б. Аппаратурная реализация элементарных функций в ЦВМ. Ленінград, 1975 г., 96 с. 11. Jean-Michel Muller. Elementary Functions: Algorithms and Implementation. Second Edition, Birkhaeuser, 2006. 12. Richard W. Hamming. Introduction to applied numerical analysis. Bell Telephone Laboratories, Incorporated and City College of New York, 1971, 331 pp. 13. Guyot, Alain. Technique de l'informatique et de la microélectronique pour l'architecture. Université Joseph Fourier, Grenoble, France, 2005.