ПОТОЧНЕ ОБЧИСЛЕННЯ ДВІЙКОВОГО ЛОГАРИФМУ ДЛЯ ПЛІС ARTIX-7 ТА СИНТЕЗАТОРА VIVADO

доц., к.т.н. Свид І.В., асистент Чумак В.С.,
завідувач лабораторії Бойко Н.В.
Харківський національний університет радіоелектроніки,
кафедра мікропроцесорних технологій і систем
e-mail: valeriia.chumak@nure.ua

Abstract. Variants of the implementation of stream calculation of the binary logarithm are considered, the results of measuring the accuracy and resource intensity for the FPGA Artix-7 are presented. A couple of calculation algorithms have been tested. Selected the most efficient in terms of resources and speed

Вступ. Обробка даних в складних системах вимагає рішення задач цифрової обробки сигналів (ЦОС) і великої кількості каналів, з чим справляються Spartan-7, Artix-7, Kintex-7, Virtex-7. ПЛІС Xilinx 7 серії, які мають високошвидкісну смугу пропускання, велику кількість логічних елементів, низьке енергоспоживання і високу продуктивність за низькою ціною [1-3].

У математичних обчисленнях та цифровій обробці сигналів на ПЛІС часто доводиться вдаватися до обчислення логарифму. Наприклад, для перетворення потужності з мВт в дБм потрібно обчислення логарифму за основою 10:

$$\mathrm{PdBm} = 10 * \log10(\text{РмВт} \ / \ 1\text{мВт}).$$

Обчислення логарифму є досить актуальним завданням, розробники застосовують різні методи обчислення [4]:

- табличний метод;
- ітераційний метод;
- CORDIC метод;
- метод на основі рядів Тейлора.

Усі алгоритми ґрунтуються на тому, що:

$$\log2(x * 2^n) = \log2(x) + n.$$

Тобто завдання обчислення двійкового логарифму зводиться до приведення аргументу до інтервалу [1..2) (нормалізація) множенням/розподілом на 2 або двійковим зсувом, та обчислення дробової частини. Цілою частиною логарифму буде кількість множень/поділів (зсувів), необхідні нормалізації. Алгоритми, що показані, виконують обчислення дробової частини.

Основна частина.

1. Табличний метод. У табличному способі дробова частина логарифму береться із заздалегідь створеної таблиці значень log2(x) в інтервалі [1..2). У ПЛІС таблиця реалізується у вигляді ROM з адресацією по дробовій частині нормалізованого числа. Розрядність ROM (довжина таблиці) вибирається з необхідної точності обчислень.

Поліпшити точність табличного алгоритму можна за допомогою лінійної інтерполяції між точками табличних значень. Для цього беруться біти дробової частини нормалізованого числа, що залишилися після адреси ROM, і множаться на коефіцієнт, що дорівнює різниці наступного і поточного значень з таблиці. Так як в потокової реалізації ми можемо двічі прочитати значення з ROM, доцільно створити ще одну ROM з обчисленими значеннями коефіцієнтів кожної точки. Помилка обчислення становить 0,00001534.

2. Алгоритм з лінійною інтерполяцією дробової частини. Для обчислення логарифму береться дробова частина нормалізованого числа та конкатенується з цілою частиною логарифму.

Наприклад:

Візьмемо 16-бітне число x = 16'b0000.0000_0101_0001, Його десяткове представлення та значення функції log2(x):

$$x = 16'b0000.0000_{0101_{0001}} \approx 0.019775390625,$$
$$log2(0.019775390625) \approx -5.660149997.$$

Після нормалізації з'єднуємо e та дробову частину нормалізованого x. При такому обчислені помилка становить 0.0742, а в значенні децибел (20 * log10 (x)) помилка становитиме приблизно 0,45 dB. Якщо додати суматор – точність покращиться удвічі.

3. Алгоритм CORDIC. Спочатку CORDIC-алгоритм був придуманий для повороту вектора на площині за допомогою операцій "зсув регістру вправо" і "складання регістрів". Іншими словами - для реалізації повороту вектора апаратно (за допомогою цифрової схемотехніки) [4]. Щодо рішення завдання з логарифмом, то алгоритм заснований на послідовному наближенні аргументу функції log2(x) до одиниці з одночасним перерахуванням результату функції шляхом додавання або віднімання заздалегідь обчислених значень log2(x), що відповідають множнику аргументу. Щоб не використовувати множення, у методі CORDIC застосовується бітовий зсув, який еквівалентний множенню/поділу на 2.

В основі алгоритму наступна формула [5]:

$$log(a * (1 \pm k)) = log(a) + log(1 \pm k) = y + log(1 \pm k).$$

При деяких початкових значеннях x недостатньо наближається до одиниці, через що падає точність обчислень. Поліпшити збіжність допомагає повторне обчислення кроку з тим самим значенням k. Це можна

робити на кожному кроці, але найкращий результат дає вибіркове повторення кроків.

Висновки.

Алгоритм лінійної інтерполяції, незважаючи на його простоту, виявився не найкращим з погляду ресурсів. Однак, можливо, показану реалізацію можна оптимізувати.

Найефективнішим з погляду ресурсів та швидкості виявився табличний алгоритм з інтерполяцією. А його варіант без інтерполяції та з розміром таблиці до 512 не потребує використання BRAM або помножувача та показує прийнятні результати точності, при цьому вмістивши все в 100-150 LUT. Крім того, кількість тригерів помітно менша, ніж в алгоритмі з лінійною інтерполяцією.

Через повільну збіжність наближення в алгоритмі CORDIC необхідно збільшувати кількість кроків наближення, що з урахуванням потокової реалізації вимагає великої кількості ресурсів. Незважаючи на це, ітераційна реалізація може показати хороші результати як за точністю, так і за ресурсами. Крім того, цей метод має саму рівномірну помилку обчислення, що може бути важливим у деяких випадках. Ще одна перевага алгоритму в тому, що він не використовує помножувачів.

Список використаних джерел.

1. В.С. Чумак, И.В. Свид. Перспектива использования продукта FPGA в медицинских системах. // XIII Міжнародна науково-практична конференція магістрантів та аспірантів «Теоретичні та практичні дослідження молодих науковців» (19–22 листопада 2019 року): матеріали конференції. – Харків : НТУ «ХПІ», 2019. – С. 288-289.

2. Oleg Zubkov, Iryna Svyd, Oleksandr Maltsev, Liliia Saikivska. In-circuit Signal Analysis in the Development of Digital Devices in Vivado 2018. // First International Scientific and Practical Conference «Theoretical and Applied Aspects of Device Development on Microcontrollers and FPGAs» MC&FPGA-2019, Kharkiv, Ukraine, July 26-27, 2019. – Kharkiv: 2019. – P. 12-13. DOI: 10.35598/mcfpga.2019.003/.

3. Чумак В.С. Особливості реалізації вузла швидкого перетворення фур'є на ПЛІС архітектури FPGA / В. С. Чумак, І. В. Свид // Радиоэлектроника и молодежь в XXI веке : материалы 25-го Междунар. молодеж. форума, 20–22 апр. 2021 г. – Харьков : ХНУРЭ, 2021. – Т. 3. – С. 187–188.

4. A.M. Mansour. A New Hardware Implementation of Base 2 Logarithm for FPGA, 2015.

5. Дайнеко Д. Реализация CORDIC-алгоритма на ПЛИС // Компоненты и Технологии. 2011. №12 (125).