Аппаратный делитель на основе алгоритма Гольдшмидта для реализации на ПЛИС

О. И. Буренева

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)

OIBureneva@etu.ru

А. П. Павлов

Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)

appavlov@stud.etu.ru

Аннотация. **Статья посвящена разработке устройства для деления целых чисел с представлением результата в формате с фиксированной точкой. Работа устройства основана на алгоритме Гольдшмидта, но в отличие от известного алгоритма в предлагаемом делителе не требуется масштабирование аргументов. Разработанный делитель представлен синтезируемым описанием на языке VerilogHDL в базисе логических элементов и ориентирован на реализацию в базисе ПЛИС. Анализ характеристик устройства проведен на основе результатов его имплементации в микросхему Cyclone 10LP: в статье показаны временные параметры и аппаратные затраты, проведено сравнение с известными решениями. Разработанный аппаратный делитель может быть использован при создании систем на кристалле, а также при проектировании макроблоков заказных интегральных схем.**

Ключевые слова: быстрое деление; алгоритм Гольдшмидта; вычислительные алгоритмы; итеративный процесс; аппаратная реализация арифметических операций

I. Введение

С повышением вычислительной сложности задач, решаемых в точках получения информации в рамках парадигмы граничных вычислений, актуальным становится создание аппаратных ускорителей. Быстрые арифметические вычислители необходимы там, где требуется выполнять большое количество арифметических операций в реальном времени: в системах рендеринга, при реализации средств искусственного интеллекта, в ML алгоритмах, при реализации процессорных систем на кристалле на базе soft-ядер, где они используются как быстрые, внешние относительно ядра преобразователи [1–3].

Наиболее затратной операцией с точки зрения времени выполнения и аппаратных затрат является деление. Известно большое число алгоритмов деления, авторы [4] выполнили их классификацию, выделив пять основных классов: повторение цифр, функциональная итерация, высокое основание, табличные методы и варианты с переменной задержкой. Часто алгоритмы совмещают, получая решения, соединяющие в себе достоинства разных вариантов. Выбор конкретного алгоритма зависит от аппаратной базы. При реализации в базисе программируемых логических интегральных схем (ПЛИС) необходимо учитывать структуру ячеек и наличие встроенных аппаратных умножителей [3, 5]. При проектировании вычислителей в виде макроблоков заказных интегральных схем требуется сконцентрировать внимание на занимаемой площади, потребляемой энергии и удобстве масштабирования топологии [6].

В базовом варианте деление реализуется как последовательность вычитаний делителя сначала из делимого, а затем из образующихся в процессе деления частичных остатков. Этот способ основан на операциях вычитания/сложения и сдвига и позволяет определить частное Q и остаток R от деления N на D в соответствии с формулами:

$$Q = \left\lfloor \frac{N}{D} \right\rfloor \text{ и } R = N - Q \cdot D \text{ при } R < D.$$

Известны различные модификации этого алгоритма, повышающие быстродействие за счет ускорения операций суммирования (вычитания) [7].

Для ускорения деления также разработаны различные оригинальные алгоритмы [8]:

- SRT алгоритм (Sweeney, Robertson, Tocher) [9], являющийся модификацией деления без восстановления остатка, в которой сложение или вычитание, в зависимости от получающегося частного остатка, на некоторых шагах не выполняется, что позволяет минимизировать время выполнения операции;
- GSA (Generalized Svoboda division Algorithm) алгоритм рекуррентного деления цифр с основанием n, основанный только на частичном остатке и реализующий логику выбора цифры частного на основе старших битов остатка [10];
- алгоритм Ньютона–Рафсона [11], основанный на замене операции деления на умножение делимого на величину, обратную делителю;
- алгоритм Гольдшмидта [12], в котором операция деления правильных дробей сводится к последовательным операциям умножения числителя и знаменателя на постоянные коэффициенты, что не изменяет соотношение, но позволяет привести знаменатель к единице.

Наша работа посвящена исследованию аппаратной реализации деления на основе алгоритма Гольдшмидта, который мы адаптировали для работы в целочисленных

значениях. Предложенная модификация позволила выполнить аппаратную реализацию устройства, ориентируясь на его реализацию в базисе ПЛИС.

II. Алгоритм Гольдшмидта

Алгоритм деления Гольдшмидта является алгоритмом, основанным на конвергенции, и реализует деление путем умножения как числителя, так и знаменателя на «антиделитель» [8].

A. Описание алгоритма

Основная идея алгоритма Гольдшмидта состоит в следующем: для получения частного Q подбираются коэффициенты $k_1, k_2, \dots k_i, \dots k_m$, на которые умножается и делимое N, и делитель D, такие, что в результате умножения делитель D стремится к единице:

$$Q = \frac{N\prod_{i=1}^{m} k_i}{D\prod_{i=1}^{m} k_i}.$$

k_i рассчитывается следующим образом:

$$k_i = 2 - D_{i-1}, \qquad D_0 = D.$$

Числитель и знаменатель для очередной итерации вычисляются по формуле:

$$\frac{N_{i+1}}{D_{i+1}} = \frac{N_i k_{i+1}}{D_i k_{i+1}},$$

то есть

$$Q \approx N\prod_{i=1}^{m} k_i \text{ когда } D\prod_{i=1}^{m} k_i \approx 1.$$

Для корректной работы алгоритма необходимо, чтобы знаменатель находился в интервале (0, 1). Это можно обеспечить масштабированием числителя и знаменателя. Рассмотренный алгоритм является итерационным, количество итераций m определяет точность результата.

Основной недостаток рассмотренного алгоритма состоит в том, что он не позволяет вычислить остаток. Кроме того, на каждой итерации присутствует умножение, что приводит к необходимости использовать умножители, являющиеся сложными устройствами. Авторы [13] показали, что алгоритм деления Гольдшмидта может быть выражен в виде рекурсивных уравнений, использующих операции умножения и возведения в квадрат на каждой итерации, определив, что такой модифицированный вариант алгоритма эффективен для реализации в программных библиотеках.

B. Аппаратная реализация алгоритма

Авторы [12] предложили вариант аппаратной реализации, основанной на использовании таблицы поиска, которая представляет собой оптимальную взаимно обратную таблицу, в которой хранятся значения коэффициента k для каждого значения D. Для повышения точности вычислений необходимо увеличивать объем памяти для хранения таблицы. Для разрядности n сигнала D емкость таблицы будет равна $n2^n$, то есть требуемый ресурс памяти с увеличением разрядности будет увеличиваться экспоненциально. Решение использовать таблицу вместо вычислений определяется требованиями к точности результата и скорости вычислений. При больших объемах памяти скорость считывания увеличится, что снизит эффективность такого технического решения.

Алгоритм Гольдшмидта включает в себя два этапа, на основе которых строятся основные блоки архитектуры аппаратного делителя, реализующего алгоритм.

Этап 1: Используя знаменатель в качестве адреса, из таблицы считывается значение первого коэффициента k_1, и числитель и знаменатель умножаются на это значение. Результатом операции являются N_1 (являющийся приближением к результату Q_1) и D_1. На этом этапе необходимы: таблица и 2 умножителя.

Этап 2: Определяется коэффициент k_2 путем дополнения D_1 до 2. Используя коэффициент k_2, рассчитываются N_2 (или Q_2) и D_2.

На следующих этапах действия этапа 2 повторяются для следующих значений k_i, N_i (или Q_i) и D_i. Процесс вычисления завершается на шаге m, таком, при котором D_m приблизится к единице с заданной точностью.

C. Аппаратная реализация адаптированного алгоритма

Структура предлагаемого делителя показана на рис. 1. Входные сигналы устройства N и D имеют разрядность n. В схеме можно выделить 4 блока (U_1, U_2, U_i, U_n).

Блок U_1 выполняет расчет коэффициента k_1 по следующей формуле k_1=Const – D с последующим умножением результата на числитель N и знаменатель D, зафиксированные в регистрах 1 и 2, умножителями 4 и 5. Результатом операции являются N_1 (являющийся приближением к результату Q_1) и D_1. Константа Const формируется следующим образом: декодер DC определяет положение старшей единицы кода D, и на основании результата x генератор константы GC формирует константу по формуле: Const=2^{x+1} или

$$\text{Const} = 10\underbrace{00\dots0}_{x}. \qquad (1)$$

Блок U_2 реализует вторую итерацию: выполняется расчет коэффициента k_2=Const$_1$ – D_1 с последующим умножением на N_1 и D_1, хранящиеся в регистрах 1 и 2. Результат операции – N_2 (приблизительный результат Q_2) и D_2. Константа Const$_1$ формируется сдвигом константы Const элементом Shift 6 на количество разрядов 2x. Умножение 2x реализуется сдвигом x влево на один разряд элементом Shift 7. Разрядность регистров и умножителей определяется исходя из правил изменения разрядности при выполнении операции умножения: разрядность результата определяется суммой разрядностей множителей.

Изменение разрядностей по описанным правилам приводит к квадратичному росту разрядности промежуточных результатов. Поэтому такой порядок работы с разрядностями сохраняется на первых j этапах. Далее, когда промежуточные результаты достигнут определенного значения, на каждом этапе будет выполняться масштабирование кодов N_i (Q_i) и D_i. Это показано в блоке Ui.

Блок U_i реализует i-ю итерацию следующим образом: выполняет расчет коэффициента k_i путем вычитания

знаменателя D_i из константы $Const_j$ с последующим умножением на N_i и D_i, хранящихся в регистрах 1 и 2. Константа $Const_j$ формируется в блоке U_j (j<i) сдвигом константы $Const_{j-1}$ на количество разрядов $2^j x$. Начиная с блока j, во всех последующих блоках масштабирование константы не выполняется, и для исключения переполнения масштабируются результаты умножения путем сдвига N_i и D_i на $2^j x$ в сторону младших разрядов.

Последний блок U_m работает аналогично блоку U_i, в вычислениях используется константа $Const_j$, а результат N_m становится результатом деления. При этом значение коэффициента масштабирования указывает на положение десятичной точки в коде результата N_m.

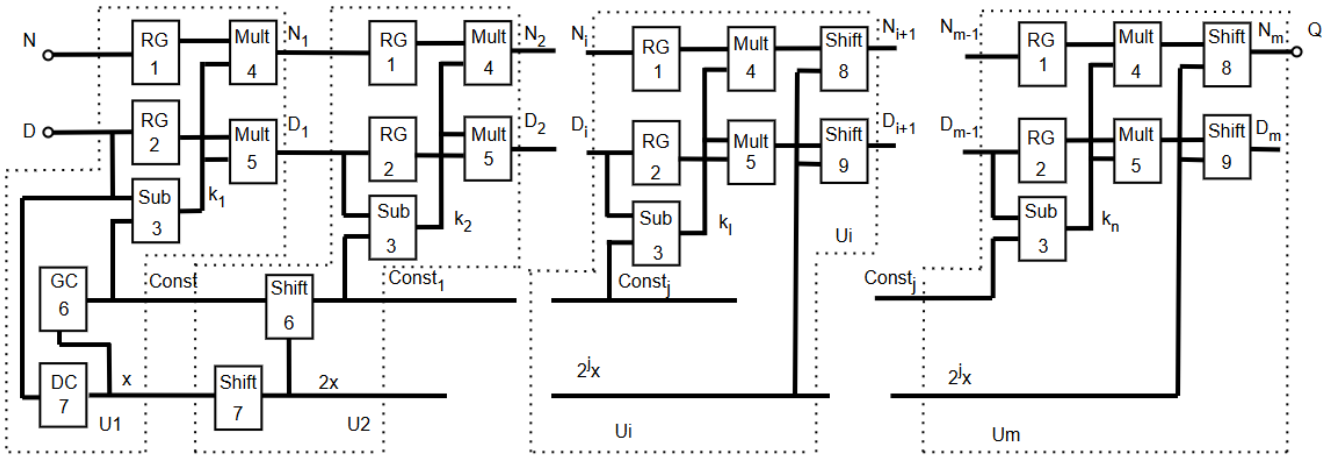

Рис. 1. Архитектура предлагаемого устройства деления на основе алгоритма Гольдшмидта

Результатом операции можно считать значение N_m, такое, для которого знаменатель представлен следующим образом:

$$D_m = 01\underbrace{00\ldots0}_{2^j x}.$$

При таком значении вычитание из константы, сформированной на текущем этапе D_m, не приведет к изменению знаменателя:

$$\mathrm{Const_m} - \mathrm{D_m} = 10\underbrace{00\ldots0}_{2^j x} - 01\underbrace{00\ldots0}_{2^j x} = 01\underbrace{00\ldots0}_{2^j x},$$

или

$$2^{2^j x+1} - 2^{2^j x} = 2\cdot 2^{2^j x} - 2^{2^j x} = 2^{2^j x}.$$

В рассмотренной реализации промежуточные результаты N_r и D_r записываются в регистры общим тактовым сигналом (на схеме не показан). Частота этого сигнала определяется временем срабатывания самой длинной комбинационной цепочки. Все элементы, реализующие сдвиг кода (Shift), являются комбинационными [14].

Анализ схемы показывает, что можно выделить блоки двух типов: в архитектуре рис. 1 – это блоки $U_1 – U_{j-1}$ и $U_j – U_m$. В блоках первого типа вычисления производятся без потери точности, но с увеличением разрядности промежуточных данных. В блоках второго типа все результаты приводятся к определенной разрядности с отбрасыванием младших разрядов. В рассмотренной архитектуре выполняется частичное распараллеливание: вычисления $N_{i+1} = N_i k_{i+1}$ и $D_{i+1} = D_i k_{i+1}$ можно выполнять одновременно. Организация последовательной работы блоков является затратной с точки зрения аппаратных ресурсов, но при этом допускает конвейеризацию: с каждым тактом на вход устройства можно подавать новую пару кодов N и D. При необходимости минимизации аппаратных затрат достаточно иметь два блока и использовать их многократно для обработки одной пары N и D. Для реализации такого режима потребуется разработка управляющего логического блока.

Рассмотренная реализация делителя имеет недостаток, характерный для итерационных, квадратично сходящихся методов, – использование умножителей, являющихся сложными устройствами. Этот недостаток может быть скомпенсирован применением специализированных быстрых умножителей [15, 16].

III. Имплементация делителя в ПЛИС

Для исследования свойств устройства было подготовлено его описание на языке VerilogHDL, ориентированное на имплементацию в программируемые логические интегральные схемы. Декодер DC, определяющий положение старшей единицы кода D, выполнен в виде комбинационной схемы на базе цепочки выявления старшей единицы и шифратора, как показано на рис. 2. Генератор константы GC выполняет формирование константы формата (1) с использованием операции расширения разрядности путем дополнения нулями младших бит. Во всех блоках использованы регистры защелки 1 и 2 для хранения промежуточных значений N_i и D_i, умножители Mult 4 и 5 для вычисления $N_{i+1} = N_i k_i$ и $D_{i+1} = D_i k_i$, а также вычитатели 3. Фрагмент RTL представления устройства, сгенерированный системой проектирования Quartus Prime для имплементации в микросхему Cyclone 10LP, приведен на рис. 3. В представленном фрагменте показаны младшие блоки U_1 и U_2. Определитель положения старшей единицы и декодер представлены на RTL модулем Const_G_inst. При этом значение положения старшей

единицы фиксируется в регистре msb_reg и в последующие блоки передается синхронно с передачей промежуточных данных.

В табл. I приведены возможные частоты тактового сигнала Fmax, рассчитанные с использованием модели Slow1, которая соответствует напряжению – 1200мВ и температуре – 85 °C. Высокое значение Fmax объясняется тем, что предложенный делитель представляет собой конвейер. Максимальная частота ограничена скоростью работы блоков умножителей и вычитателя. При этом для умножения встроенные умножители не используются, а умножение реализуется на ячейках, настроенных на работу в арифметическом режиме.

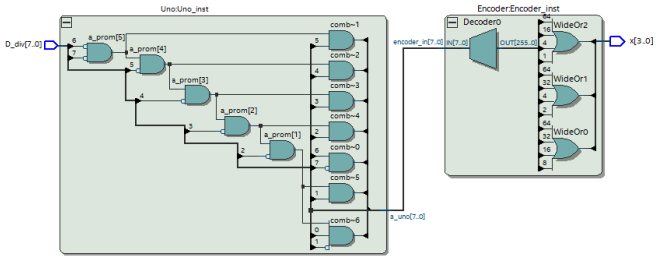

Рис. 2. Декодер – определитель положения старшей единицы кода D и шифратор, формирующий результат в двоичном коде

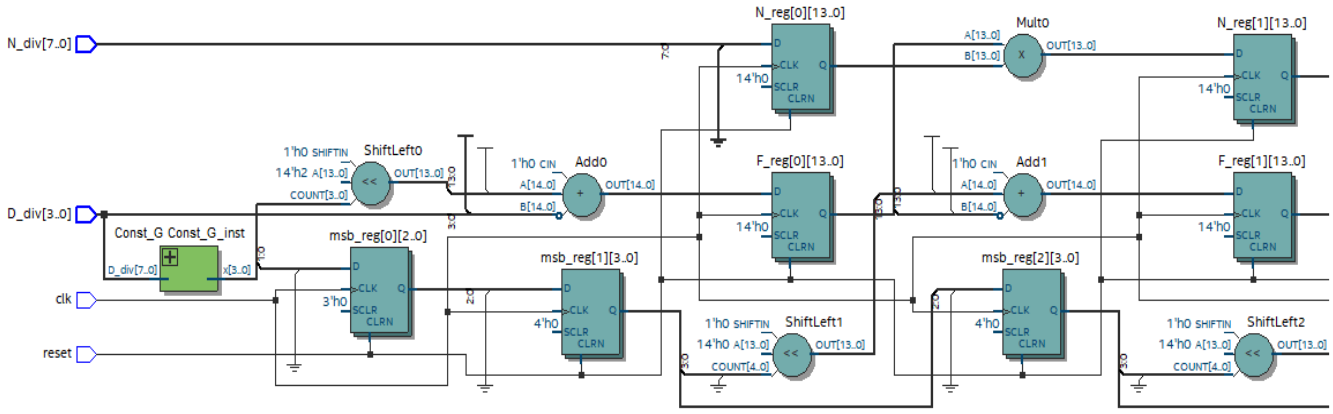

Рис. 3. Фрагмент RTL спроектированного устройства (блоки U_1, U_2)

ТАБЛИЦА I. ХАРАКТЕРИСТИКИ УСТРОЙСТВ ДЕЛЕНИЯ

	Аппаратные затраты			**Fmax, МГц**
	Логическ. элементы	***Регистры***	***Встроенные умножители***	
LPM_Divider	1039	99	-	17
Деление без восстановления остатка	101	75	-	233
Деление методом Ньютона-Рафсона	95	40	9	86
Предложенный делитель	125	120	0	247,77

IV. ЗАКЛЮЧЕНИЕ

В представленной работе показана реализация модулей целочисленного деления для ПЛИС, работа которого основана на подходе, принятом в делителе Гольдшмидта: числитель и знаменатель умножаются на коэффициенты так, чтобы знаменатель стремился к определенному значению. В рассмотренном устройстве это значение определяется степенью 2. Устройство компактно, обладает высокой производительностью и превосходит по некоторым характеристикам известные решения. Подготовленное Verilog описание параметризовано и может быть легко перенастроено для работы с данными другой разрядности. Кроме того, это описание может стать основой для проектирования макроблоков заказных интегральных схем. Предложенное устройство может быть модифицировано для вычисления квадратного корня и обратного квадратного корня, а также использовано для проектирования делителя с плавающей точкой.

СПИСОК ЛИТЕРАТУРЫ

[1] A. Suresh, B. N. Reddy and C. Renu Madhavi, "Hardware Accelerators for Edge Enabled Machine Learning," 2020 IEEE REGION 10 CONFERENCE (TENCON), Osaka, Japan, 2020, pp. 409-413, doi: 10.1109/TENCON50793.2020.9293918.

[2] F. He, K. Ding, X. He, M. Chen, K. Mao and Q. Zhang, "A Novel Self-Attentive Neural Network Hardware Accelerator for Edge Computing," 2024 IEEE 4th International Conference on Data Science and Computer Application (ICDSCA), Dalian, China, 2024, pp. 477-482, doi: 10.1109/ICDSCA63855.2024.10859467.

[3] E. Matthews, A. Lu, Z. Fang and L. Shannon, "Rethinking Integer Divider Design for FPGA-Based Soft-Processors," 2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM), San Diego, CA, USA, 2019, pp. 289-297, doi: 10.1109/FCCM.2019.00046.

[4] S.F. Obermann and M.J. Flynn, "Division algorithms and implementations," in IEEE Transactions on Computers, vol. 46, no. 8, pp. 833-854, Aug. 1997, doi: 10.1109/12.609274.

[5] E. Matthews, A. Lu, Z. Fang and L. Shannon, "Quick-Div: Rethinking Integer Divider Design for FPGA-based Soft-processors," in ACM Trans. Reconfig. Technol. Syst, vol. 15, no. 3, Art 32, 27 pages, Sept. 2022, https://doi.org/10.1145/3502492

[6] M. M. A. Da Rosa, P. Da Costa, G. Paim, E. Da Costa, R. Soares and S. Bampi, "An Energy-Efficient StEFCal VLSI Design with Approximate Squarer and Divider Units," 2023 IEEE 14th Latin America Symposium on Circuits and Systems, Quito, Ecuador, 2023, pp. 1-4, doi: 10.1109/LASCAS56464.2023.10108306.

[7] U.S. Patankar, M.E. Flores and A. Koel, "Division algorithms - From Past to Present Chance to Improve Area Time and Complexity for Digital Applications," 2020 IEEE Latin America Electron Devices Conf.

(LAEDC), San Jose, Costa Rica, 2020, pp. 1-4, doi: 10.1109/LAEDC49063.2020.9073050

[8] U.S. Patankar and A. Koel, "Review of Basic Classes of Dividers Based on Division Algorithm," in IEEE Access, vol. 9, pp. 23035-23069, 2021, doi: 10.1109/ACCESS.2021.3055735.

[9] B. Mehta, J. Talukdar and S. Gajjar, "High speed SRT divider for intelligent embedded system," 2017 International Conf. on Soft Computing and its Engineering Appl. (icSoftComp), Changa, India, 2017, pp. 1-5, doi: 10.1109/ICSOFTCOMP.2017.8280077.

[10] A. Svoboda, "An Algorithm for Division", Information Processing Machines, no. 9, pp. 25-32, 1963.

[11] O.I. Bureneva and O.U. Kaidanovich, "FPGA-based Hardware Implementation of Fixed-point Division using Newton-Raphson Method," 2023 IV International Conf. on Neural Networks and Neurotechnologies (NeuroNT), Saint Petersburg, RF, 2023, pp. 45-47, doi: 10.1109/NeuroNT58640.2023.10175844.

[12] M.D. Ercegovac, L. Imbert, D.W. Matula, J.-M. Muller and G. Wei, "Improving Goldschmidt division, square root, and square root reciprocal," in IEEE Transactions on Computers, vol. 49, no. 7, pp. 759-763, July 2000, doi: 10.1109/12.863046.

[13] M. Joldeş, O. Marty, J.-M. Muller and V. Popescu, "Arithmetic Algorithms for Extended Precision Using Floating-Point Expansions," in IEEE Transactions on Computers, vol. 65, no. 4, pp. 1197-1210, 1 April 2016, doi: 10.1109/TC.2015.2441714.

[14] Patrick W. Bosshart and Quang Dieu An, "Shifter circuit for an arithmetic logic unit in a microprocessor," Patent US5896305A, United States, 07 Feb 1997.

[15] O. Bureneva and S. Mironov, "Fast FPGA-Based Multipliers by Constant for Digital Signal Processing Systems," in Electronics, vol. 12, no. 605, 2023, doi: 10.3390/electronics12030605.

[16] S. Ullah, S. Rehman, M. Shafique and A. Kumar, "High-Performance Accurate and Approximate Multipliers for FPGA-Based Hardware Accelerators," in *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 41, no. 2, pp. 211-224, Feb. 2022, doi: 10.1109/TCAD.2021.3056337.