УДК 004.315.5

Модули вычисления функций обратной величины и обратного квадратного корня одинарной точности

Е.В. Ивасюк, П.С. Зубковский

Учреждение Российской академии наук
Научно-исследовательский институт системных исследований РАН

zubkovsky@cs.niisi.ras.ru

***Аннотация* — Статья описывает архитектуру и функционирование модулей вычисления функций обратной величины и обратного квадратного корня от вещественных чисел одинарной точности. Реализован метод коррекции результата для соответствия стандарту IEEE754. Проведена оценка быстродействия модулей.**

***Ключевые слова* — Вещественная арифметика, обратная величина, обратный квадратный корень, метод Ньютона, IEEE754.**

I. Введение

Быстрое выполнение сложных инженерных расчетов требует разработки быстродействующих арифметических модулей для процессоров общего или специализированного назначения. Операции вещественного сложения и умножения уже давно реализовываются в микропроцессорах на аппаратном уровне. Проектирование модулей, выполняющих эти операции, хорошо отлажено и в настоящее время доступно множество публикаций на эту тему, например [1], [4].

Такие операции как деление и квадратный корень, являются гораздо более сложными в проектировании, и поэтому часто реализуются программно-аппаратным способом, используя существующие аппаратные возможности процессоров, а именно модули, выполняющие операции умножение с накоплением.

Преимущество такого способа состоит в небольшой сложности реализации и, соответственно, малых аппаратных затратах, требующих добавления специализированных обратных связей. Недостаток заключается в том, что построенная таким образом схема получится очень медленной и неконвейерной, так как один и тот же модуль будет задействован для выполнения всех итераций, в дополнение к этому необходимо еще вычислить некоторое начальное приближение.

Поэтому, актуальной является задача реализации алгоритмов деления и обратного квадратного корня полностью на аппаратном уровне.

II. Выбор базового алгоритма

Существует два основных класса алгоритмов двоичного деления и квадратного корня: методы цифровой рекурсии (SRT, название происходит от фамилий создателей) и методы функциональных итераций. Частным и самым простым случаем SRT-алгоритмов является вычисление “в столбик”. Самым распространенным алгоритмом функциональной итерации является метод Ньютона.

SRT-методы имеют линейную сходимость к результату, т.е. обеспечивают получение фиксированного количества битов мантиссы результата за одну итерацию. Схемы, реализующие подобные алгоритмы, сложнее сделать конвейерными, особенно при обработке чисел двойной точности.

Поэтому в качестве базового алгоритма был выбран метод Ньютона. Он имеет квадратичную сходимость и при точности начального приближения $2^{-12}\ldots2^{-14}$ потребует всего одну итерацию. У метода Ньютона есть существенный недостаток – метод не обеспечивает получение точного остатка и, следовательно, сложно выполнить округление полученного результата по стандарту IEEE754[2]. Ошибка результата будет составлять не более одного младшего значащего бита (unit in last position, ulp) мантиссы. Однако, такой точности достаточно для работы в составе специализированного сопроцессора, например сопроцессора цифровой обработки сигнала. Тем не менее, будет предложен метод получения точного остатка, но требующий дополнительных аппаратных затрат.

При постановке задачи на разработку было три основных критерия, которым должны удовлетворять разрабатываемые модули: они должны быть полностью конвейерными, латентность выполняемых операций не должна превышать 6-ти тактов машинного времени и погрешность результата должна быть не больше одного младшего значащего бита мантиссы.

Такие требования продиктованы будущей областью применения.

III. Описание алгоритма

Как было отмечено в предыдущем разделе, в качестве базового алгоритма выбран метод Ньютона. Формулы для вычисления значений итераций обратной величины и обратного квадратного корня имеют следующий вид:

$$X_{i+1} = X_i \times (2 - d \times X_i),$$

для обратной величины, d – значение исходного операнда,

$$X_{i+1} = \frac{X_i}{2} \times (3 - d \times X_i^2),$$

для обратного квадратного корня, d – значение исходного операнда.

Каждая итерация включает в себя две более простые: умножение и умножение с вычитанием. Причем они зависимы, умножение должно быть выполнено после умножения с вычитанием. В случае обратного квадратного корня, выполнение первой части итерации усложняется тем, что сначала необходимо вычислить квадрат значения предыдущей итерации.

Для вычисления начального приближения X_0 целесообразно использовать метод, который дает точность достаточную для выполнения всего одной итерации, которая обеспечит достижение итоговой точности результата[3]. В качестве такого метода хорошо подходит метод кусочно-линейной аппроксимации первого порядка. Аппроксимация искомых функций осуществляется в диапазоне значений [1, 2), что обусловлено стандартом IEEE754. Формула для вычисления начального приближения имеет вид:

$$X_0 = C_0 - d_{\mathrm{mod}} \times C_1,$$

где C_0 и C_1 – коэффициенты, d_{mod} – модифицированный исходный операнд,

$$d_{\mathrm{mod}} = 1.d_{22}...d_0 - 1.d_{22}...d_{17}0...0.$$

II. Описание модулей

Модули реализованы в виде программных RTL-моделей, описанных на языке Verilog, и имеют полностью конвейерную организацию. Модуль вычисления обратной величины имеет 5 стадий, обратного квадратного корня – 6 стадий. Структурные схемы модулей представлены на рис.1а и рис.1б.

Выборка коэффициентов C_0 и C_1 производится табличным способом в обоих модулях, в качестве адреса таблицы выступают старшие биты мантиссы исходного операнда, располагающиеся после десятичной точки. В данном случае выборка коэффициентов производится по 6-ти старшим битам дробной части мантиссы входного операнда. При вычислении обратного квадратного корня в таблице хранятся по два значения каждого из коэффициентов: для случаев четной и нечетной экспоненты числа.

Ошибка начального приближения, выполненного предложенным методом кусочно-линейной аппроксимации, с выборкой коэффициентов по 6-ти старшим битам мантиссы составляет не более 2^{-14}.

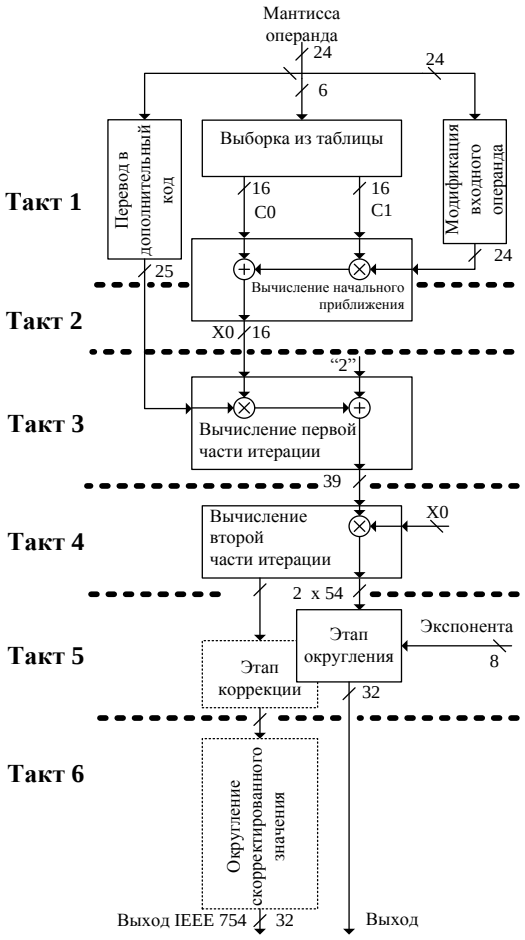

Рис. 1а. Структурная схема модуля вычисления обратной величины

Суммарный объем коэффициентов для модуля обратной величины составляет приблизительно 256 байт, для модуля обратного квадратного корня – 512 байт. Размер таблиц невелик, поэтому целесообразным будет избежать применения блоков памяти и реализовать таблицы на комбинационной логике.

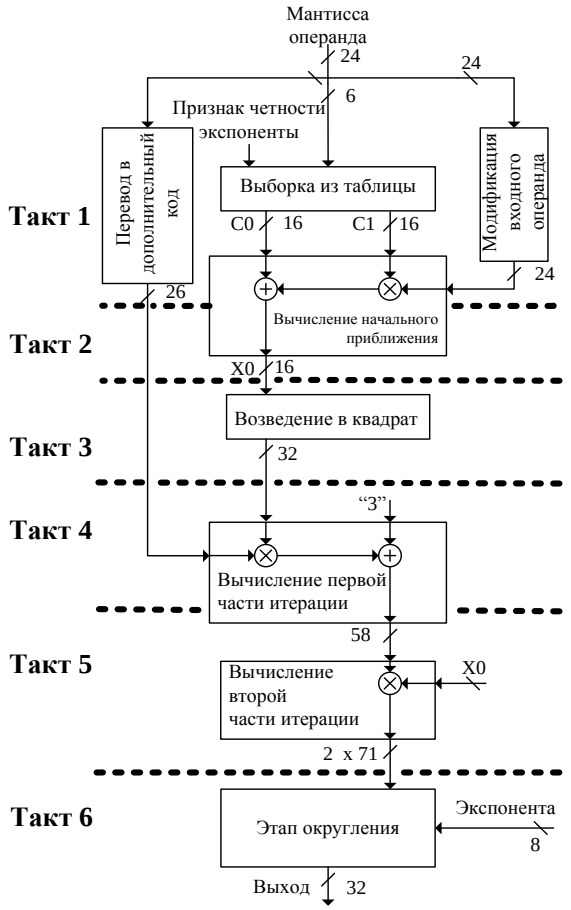

Рис. 1б. Структурная схема модуля вычисления обратной величины

Как было отмечено выше, для вычисления начального приближения необходима дополнительная операция умножения с вычитанием. Данное действие реализовано с помощью подмодуля умножения. Все подмодули умножения, примененные для вычисления функций, реализованы на основе модифицированного алгоритма Бута[4], который предусматривает формирование частичных произведений, а затем их сложение сумматором (деревом Уоллеса[5]). Необходимо отметить, что операция умножения начального приближения знаковая, так как надо получить значение:

$$X_0 = C_0 - d_{\mathrm{mod}} \times C_1 .$$

Согласно правилу умножения знаковых чисел по этому алгоритму, последнее частичное произведение содержит только один бит – инкремент предыдущего. Поэтому сложение константы C_0 может быть реализовано модификацией последнего частичного произведения и, следовательно, количество частичных произведений не изменится, что позволит выполнить начальное приближение за время умножения. Как видно из приведенного рисунка, вычисление начального приближения занимает больше такта машинного времени. Полная разрядность приближения не нужна, так как ошибка существенно больше ulp X_0 и составляет величину 2^{-14}, поэтому, для удобства и запаса в один бит, произведено усечение до 16 бит (1бит целой части, который всегда "0", и 15 бит дробной части - непосредственно приближение).

Единственная итерация Ньютона выполнена с помощью операций умножения с вычитанием, а затем просто умножения. Умножение с вычитанием выполнено аналогично таковому из начального приближения, что опять же позволяет выиграть в скорости без существенных затрат. Итерация занимает приблизительно два такта машинного времени. Необходимо отметить, что у последнего умножения итерации нет необходимости выполнять суммирование с распространением переноса, достаточно всего лишь выполнить компрессию частичных произведений деревом Уоллеса.

Окончательное суммирование с распространением переноса объединено с этапом округления. Такой подход считается распространенным при построении арифметических исполнительных устройств, в которых необходимо округление, и позволяет получить существенный выигрыш в быстродействии. Алгоритм округления представляет собой один из стандартных подходов при выполнении операции вещественного умножения[6]. Он может быть использован и здесь без существенных переделок. Изменения коснутся только разрядности обрабатываемых операндов, которые отличаются от таковых для умножения мантисс одинарной точности.

Все сумматоры с распространением переноса, примененные в модулях, построены по схеме с ускоренным переносом (carry-look ahead). Они использованы на этапах дополнения/модификации входной мантиссы, начального приближения, возведения в квадрат начального приближения для обратного корня, первой части итерации, а также на этапе округления.

IV. Выполнение коррекции

Как было отмечено в разделе II, данный алгоритм не позволяет получить результат, полностью удовлетворяющий стандарту IEEE754 по точности, так как метод Ньютона не обеспечивает получение точного остатка.

Действительно, тестирование моделей показало, что отличия от точных значений составляют не более 1ulp мантиссы результата. Проверка проводилась с помощью перебора операндов со всеми комбинациями мантисс и с фиксированным значением экспоненты. В случае обратного квадратного корня, перебор проводился дважды, с четным и нечетным значениями экспоненты.

Результаты моделирования показали, что расхождения с точным результатом происходят приблизительно в 30 – 40% от всех комбинаций мантиссы. Для повышения точности работы, т.е. для сокращения количества неправильных результатов, можно увеличить точность начального приближения, например, повысив степень аппроксимирующей функции, но расхождения все равно останутся.

Это может оказаться неприемлемым при встраивании модулей в состав блоков вещественной арифметики (FPU) основного процессора, поэтому необходимо предусмотреть этап коррекции, который конечно же потребует дополнительных аппаратных затрат и увеличит время выполнения операций.

Для обеспечения точного округления следует знать точный остаток, который возможно получить, выполнив дополнительные вычисления[7]. Точный остаток равен разности делимого (единица) и произведения неточного частного (его квадрата) на делитель:

$$REM = 1 - RES \times d ,$$

в случае обратной величины,

$$REM = 1 - RES^2 \times d ,$$

в случае обратного квадратного корня.

Важен даже не сам остаток, а его знак: если он положителен, то значит, полученный результат является недооцененным, если он отрицателен – переоцененным.

Вычисление знака точного остатка требует добавления подмодуля умножения, который с помощью небольшой модификации, упомянутой выше, может выполнить операцию умножения с вычитанием. В дополнение к этому, необходимо полное суммирование с распространением переноса для определения знака.

Следовательно, неточный результат следует либо инкрементировать, либо декрементировать в зависимости от режима округления. Такая коррекция была выполнена на примере только модуля обратной величины, так как он имеет большее быстродействие, т.е. латентность 5 тактов против 6 у обратного корня. Она позволила выполнить точное округление и полностью соответствовать стандарту. В случае обратного квадратного корня применение такой коррекции более затруднительно, так как требуется больше арифметических действий, и они не укладываются в отведенный диапазон 6 тактов. Работа по повышению точности работы и быстродействия модуля обратного квадратного корня ведется в настоящее время.

V. Заключение

Спроектированы программные модули вычисления обратной величины и обратного квадратного корня одинарной точности в виде RTL-моделей, описанных на языке Verilog. Алгоритм работы основан на методе Ньютона с кусочно-линейным начальным приближением.

На примере модуля обратной величины, был реализован метод коррекции получаемого результата для соответствия стандарту вещественной арифметики.

Разработанные модули соответствуют предъявленным требованиям и могут быть применены в составе специализированного арифметического сопроцессора, который допускает неполное соответствие стандарту IEEE754. Модуль обратной величины обеспечивает точность результатов согласно стандарту, модуль обратного квадратного корня – ошибку равную ulp мантиссы. Так, модули были включены и тестировались в составе программной RTL-модели сопроцессора комплексной арифметики, который допускает неполное соответствие стандарту.

При завершении работ по дополнению модуля обратного корня этапом коррекции, модули могут быть применены в составе блоков вещественной арифметики (FPU).

Для оценки быстродействия был выполнен схемотехнический синтез разработанных модулей в составе сопроцессора. Результаты синтеза показывают, что быстродействие модулей находится на уровне 250Мгц, при 5(6)/6 этапах конвейера для деления/корня. Синтез выполнялся по проектным нормам 0.18мкм.

Литература

[1] Peter-Michael Seidel, Guy Even, "On the Design of Fast IEEE Floating-Point Adders," arith, pp.0184, 15th IEEE Symposium on Computer Arithmetic (ARITH-15 '01), 2001.

[2] ANSI/IEEE Standard 754-1985: IEEE standard for Binary Floating-Point Arithmetic. Poscataway, NJ: IEEE Press, 1985.

[3] M. Ito, N. Takagi, and S. Yajima, "Efficient Initial Approximation for Multiplicative Division and Square Root by a Multiplication with Operand Modification," IEEE Trans. Computers. Apr. - 1997. - V. 46, - № 4. P. 495-498.

[4] Gary W. Bewick, "Fast Multiplication: Algorithms and Implementation", Ph.D. Thesis, Department of Electrical Engineering, Stanford University, February 1994.

[5] Wallace, C.S.,"A Suggestion for a Fast Multiplier," IEEE Trans. Electronic Computers. - 1964. - V. 13. P. 14-17.

[6] R.K. Yu and G.B. Zyner, "167 MHz Radix-4 Floating Point Multiplier," Proc. 12th Symp. Computer Arithmetic. - 1995. - V. 12. - P. 149-154.

[7] Eric M. Schwarz, "Rounding for Quadratically Converging Algorithms for Division and Square Root," asilomar, pp.600, 29th Asilomar Conference on Signals, Systems and Computers (2-Volume Set), 1995.