УДК 621.391.26

Визор Я.Е., канд. техн. наук

УСКОРЕННАЯ РЕАЛИЗАЦИЯ РЕШЕНИЯ НЕЛИНЕЙНОЙ ФУНКЦИИ В СИНАЛЬНЫХ ПРОЦЕССОРАХ И МИКРОКОНТРОЛЛЕРАХ

Институт кибернетики им. В.М. Глушкова НАН Украины

С целью построения оптимальных по быстродействию и оборудованию арифметических устройств вычислительной техники, проведен анализ итерационных методов решения нелинейного уравнения $f(x)=x^2-a$, выбран оптимальный по быстродействию метод Ньютона. Предложен способ и алгоритм аппаратной реализации вычисления корней нелинейного уравнения в сигнальных процессорах и микроконтроллерах с увеличенным быстродействием и минимальными аппаратными затратами

Введение

Методы ускорения решения нелинейных уравнений в вычислительных системах до недавнего времени были разработаны меньше, чем методы ускорения сложения, вычитания и умножения. По-видимому, это объясняется с одной стороны, "последовательным" характером процедуры при выполнении этих операций и, с другой стороны тем, что эти операции встречалась крайне редко. В некоторых ЦВМ, непосредственное выполнение этих операций реализуется при помощи стандартных подпрограмм, включающих умножение и сложение - вычитание. Однако в настоящее время, анализируя алгоритмы цифровой обработки сигналов (ЦОС) и спектрального анализа, требуется быстрое решение нелинейного уравнения $f(x)=x^2-a$ т.е. извлечь квадратный корень с максимальным быстродействием. Операция извлечения квадратного корня включается в список операций почти всех современных универсальных машин. Необходимость ускорения этой операции следует уже из факта существования весьма эффективных методов ускорения умножения: неускоренная операция извлечения квадратного корня теряет смысл при ускоренных сложениях - вычитаниях и умножениях. В этом случае подпрограмма извлечения квадратного корня может обеспечить большую скорость, чем существующие аппаратные решения, тем более когда речь идет о арифметических блоках сигнальных процессорах (СП), работающих в реальном времени.

Постановка задачи

Синхронные методы ускорения операции извлечения квадратного корня разбиваются на две группы. В первой группе, ускорение достигается в каждом цикле, в котором вычисляются одна или несколько цифр результата, а во второй – выполнением "обходного" пути с использованием умножения. Методы ускорения первой группы достаточно хорошо описаны и реализованы в соответствующей аппаратуре, причем в тех случаях, где не требовалось большого быстродействия. Необходимо рассмотреть вторую группу методов ускорения, в основе которых лежат итерационные методы, использующие операцию умножения, поскольку архитектуры большинства современных СП, для реализации базовой операции MAC, используют быстрый умножитель - накопитель. Итерационные методы используются для решения нелинейного уравнения вида:

$$f(x)=0, \quad x\in[a,b], \qquad (1)$$

где $f(x)$ – непрерывная функция. Уравнение (1) может иметь один или несколько корней. Требуется: 1) установить существование корней уравнения; 2) найти приближенные значения корней; 3) разработать алгоритм аппаратной реализации в арифметическом модуле процессора. Часто первые две задачи решаются одновре-

менно, причем для нахождения корней применяются итерационные методы.

Анализ итерационных методов

Известен простейший метод дихотомии (деление пополам). В этом методе итерационный процесс сходится со скоростью геометрической прогрессии, но при этом он имеет существенный недостаток: при выборе начального отрезка $[x_0, x_1]$, заранее неизвестно к какому корню сойдется процесс, поскольку таких корней на данном отрезке может быть несколько. Второй рассматриваемый метод – это метод простой итерации, который определяется формулой:

$$x_{n+1} = \varphi(x_n), \quad n = 0,1,2,..., \qquad (2)$$

где n – номер итерации, x_0 – произвольное начальное приближение. Требуется найти приближенно решение (корень $x = x^*$) уравнения (2) с относительной погрешностью $\varepsilon > 0$, так, чтобы при всех $n \geq n_0(\varepsilon)$ выполнялось неравенство:

$$\left|x_n - x^*\right| \leq \varepsilon \left|x_0 - x^*\right|, \quad n \leq n_0(\varepsilon). \qquad (3)$$

Это условие может быть выполнено, если последовательность итераций $\{x_n\}$, при $n \to \infty$, сходится к приделу $\lim\limits_{n\to\infty} x_n = x^*$. Если неравенство (3) имеет место, то вычисления можно прекратить при $n = n_0$. В соответствующей литературе [1, 2] доказано, что метод простой итерации сходится со скоростью геометрической прогрессии, причем число итераций, при котором выполняется неравенство (3), определяется из условия $g^n \leq \varepsilon$. Оценка минимального числа итераций $n_0(\varepsilon)$, для выполнения неравенства (3), указывает на их большое количество по сравнению с нижеописанным методом. Метод простой итерации целесообразно использовать в арифметических блоках СП, где разрядность операнда не превышает 8 бит, в остальных же случаях он проигрывает в быстродействии и оборудовании методу Ньютона.

Следующий рассматриваемый итерационный метод называют методом Ньютона или методом линеаризации. Метод определяется формулой:

$$X_{n+1} = X_n - \frac{f(x_n)}{f'(x)}, \quad f'(x_n) \neq 0, \quad n = 0,1,2,... \qquad (4)$$

Представим $f(x)$, для любого $a > 0$, в виде:

$$f(x) = x^2 - a = 0, \qquad (5)$$

и, применив формулу (4) для получения приближенных корней данного уравнения, получим итерационную формулу метода Ньютона :

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{a}{x_n}\right), \quad n = 0,1,2,..... \qquad (6)$$

Известно, что для погрешности $Z = x - x^*$, где x^* – приближенный корень уравнения (5), имеет место соотношение:

$$\lim_{n\to\infty} \frac{Z^{n+1}}{Z^{n^2}} = \frac{f''(x^*)}{2f(x^*)}. \qquad (7)$$

Фактически соотношение (7) означает, что на каждой итерации погрешность возводится в квадрат, т.е. число верных знаков корня удваивается. Если

$$\frac{f''(x^*)}{2f(x^*)} \approx 1,$$

то легко показать, что при $|Z_n| < 0,5$, после пяти – шести итераций погрешность станет величиной порядка 2^{-64}. Это наименьшее возможное значение погрешности при вычислениях на современных ЭВМ даже с удвоенной точностью. Заметим, что для получения столь малой погрешности в методе дихотомии потребовалось бы более 50 итераций.

Метод Ньютона сходится с квадратичной скоростью, что значительно выше,

чем в других методах (метод простой итерации сходится со скоростью геометрической прогрессии). Кроме того, он имеет высокий показатель точности вычислений, при относительно малом количестве итераций. Поэтому метод Ньютона выбран в качестве базового для вычисления операции извлечения квадратного корня в арифметическом блоке сигнального процессора.

Аппаратная реализация метода Ньютона

Применение метода Ньютона в сигнальном процессоре имеет некоторые особенности, так как на любой алгоритм, который реализуется в арифметическом блоке, накладываются жесткие временные ограничения. Отсюда следует, что число итераций должно быть минимальным и достаточным для обеспечения необходимой точности вычисления. Количество итераций, которое нужно выполнить для получения необходимой точности результата, очевидно, зависит от выбора первого приближения. Поэтому для выбора начальных условий в арифметическом устройстве сигнального процессора необходимо использовать блоки ПЗУ, которые смогут обеспечить требуемую точность X_n. Момент окончания операции извлечения корня определяется не путем сравнения разности $|X_{n+1} - X_n|$ с некоторым допуском, а путем подсчета количества итераций, которые заранее известны в зависимости от разрядности используемых операндов [4]. При выборе ПЗУ для хранения начальных условий X_0, необходимо стремиться к минимальной разрядности между числом разрядов операндов и числом адресных входов ПЗУ. Например, если операнды m-разрядные, то, для вычисления X_{n+1} только с одной итерацией необходимо ПЗУ разрядностью $\geq \frac{m}{2}$, так как каждая итерация удваивает число верных разрядов. В большинстве случаях в арифметических блоках СП для операндов используется формат с фиксированной запятой, что имеет некоторые особенности при реализации алгоритма извлечения квадратного корня.

При вычислении квадратного корня, в формуле (6), необходимо вычислить соотношение $\frac{a}{x_n}$. На первой итерации $(n=0)$ имеем: $x_n = x_0$, $x_0 = \sqrt{a_0}$, где a_0 - старшая $(m-k)$ – разрядная часть подкоренного m – разрядного числа a (число k – количество разрядов младшей части числа a, для первой итерации $k = \frac{m-2}{2}$. Тогда формула (6) для первой итерации будет иметь вид:

$$x_1 = \frac{1}{2}\left(x_0 + \frac{a}{x_0}\right), \quad 0 < a < 1. \qquad (8)$$

При подаче a_0 на адресный вход ПЗУ, на выходе получим x_0. Соотношение $\frac{a}{x_0}$ запишем как $a \cdot \frac{1}{x_0}$. Тогда, чтобы не потерять своего качения в разрядной сетке, x_0 должно изменяться в диапазоне $\frac{1}{2} < x_0 < 1$, т.е. x_0 должно быть нормализованным. Нормализацию x_0 можно провести на умножителе, умножив x_0 на код нормализации H_0, при этом сдвинутое в младшую часть аккумулятора умножителя x_0, выдать на шину, но уже с логической «1» в разряде с весом 2^{-1}.

Код нормализации H_0 может храниться в том же ПЗУ, где записаны начальные условия или в отдельно взятой микросхеме ПЗУ. Число таких H_0 столько же, сколько и число начальных условий и равняется 2^{m-k}. Чтобы отношение $\frac{a}{x_0}$ не изменило своего значения, недостаточно нормализовать только x_0, необходимо на такое же количество разрядов

сдвинуть и a, т.е. $\frac{a \cdot H_0}{x_0 \cdot H_0}$. Число a сдвигается на умножителе таким же образом, как и x_0.

Далее, если обозначить $\frac{1}{x_0}$ как некоторое K_0, то получим

$$K_0 = \frac{1}{x_0}. \qquad (9)$$

Если x_0 после нормализации изменяется в диапазоне $0{,}5 \le x_0 \le 1$, то очевидно, $1 \le K_0 \le 2$. Коэффициент K_0, как и x_0, можно получить из ПЗУ, но при этом знаковый разряд с весом 2^0, превращается в информационный. Для вычисления соотношения $\frac{a}{x_0}$ и далее K_0, проводить нормализацию x_0 на умножителе нет необходимости, так как можно заранее просчитать K_0 с нормализованным x_0. Код нормализации H_0 будет нужен только для сдвига a. Анализируя вышесказанное, можно сделать вывод, что с помощью ПЗУ, из a_0 возможно получить переменные x_0, K_0, H_0. Тогда формулу (8) можно записать в следующем виде:

$$x_1 = \frac{1}{2}\left(x_0 + a \cdot K_0 \cdot H_0\right). \qquad (10)$$

Как видно из (10) вычисление сводится к двойному умножению, сумме и сдвигу на один разряд в сторону младших разрядов. Следует заметить, что умножать вначале надо $a \cdot H_0$, потому что результат должен выдаваться из младших разрядов аккумулятора умножителя. Поскольку умножение чисел с фиксированной запятой проводится с коррекцией и без, то в данном случае, для реализации сдвига на один разряд вправо, коррекцию проводить нет необходимости.

Выше рассмотрен случай, когда $a_0 \neq 0$. Если $a_0 = 0$ и $a \neq 0$, то вычисление x_1 будем проводить исходя из следующих рассуждений. Изначальную формулу вычисления квадратного корня $x = \sqrt{a}$ запишем таким образом:

$$x = \sqrt{a} = \sqrt{\frac{a \cdot 2^{k+1}}{2^{k+1}}} = \frac{\sqrt{a'}}{2^{\frac{k+1}{2}}} = \frac{x'}{2^{\frac{k+1}{2}}}, \qquad (11)$$

где $a' = a \cdot 2^{k+1}$ и k – количество разрядов младшей части числа a. После сдвига a на $k+1$ разрядов влево, мы получим новое a_0', которое не будет равно 0. Теперь можно вычислить x_n' по алгоритму, описанному выше, т.е. для случая, когда $a_0 \neq 0$. Сдвинув, вычисленное x_n' на $\frac{k+1}{2}$ разряда вправо, мы получим искомое x. В силу того, что в формате числа с фиксированной запятой нет возможности представить число 2^{k+1} для сдвига влево числа a, сдвигать будем вправо, умножив a на $2^{-(k+1)}$, а результат, т.е. a_0 будем брать из младшей части аккумулятора умножителя. Таким образом, будет реализован сдвиг влево числа a на $k+1$ разрядов. Очевидно, для этого случая, в арифметическом блоке следует предусмотреть устройство, анализирующее разряды a_0, для проверки условия $a_0 = 0$. Таким устройством может быть $(m-k)$ – входовая схема ИЛИ с триггером, выход которого будет выполнять функции флага *F* для местного устройства управления. Кроме того, должна быть какая-либо ячейка памяти для хранения, сдвигающего числа $2^{-(k+1)}$.

С учетом вышесказанного, можно составить обобщенный алгоритм [3], который должно реализовать местное устройство управления процессора, при вычислении квадратного корня. Такой алгоритм представлен на рис. 1.

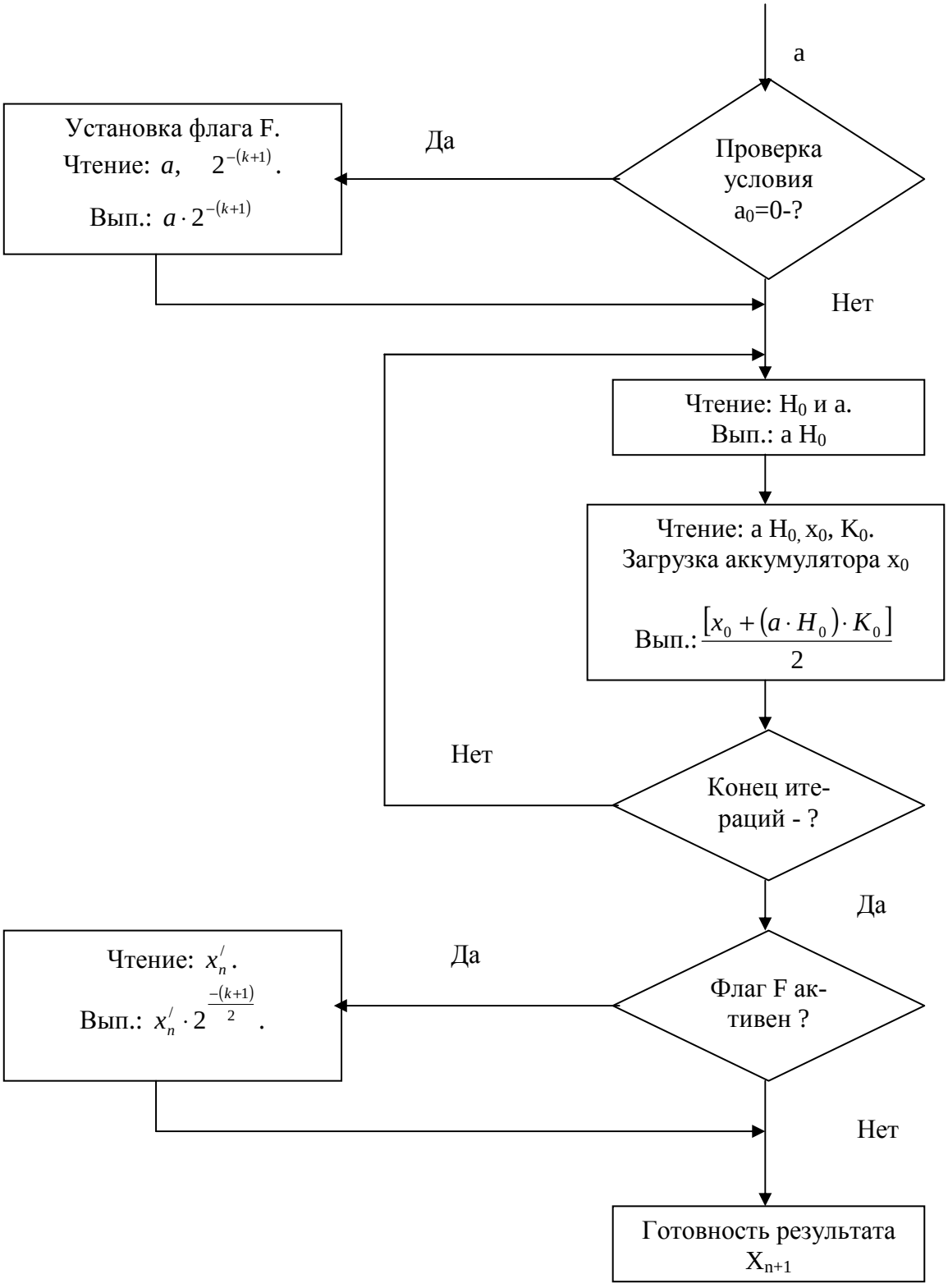

Рис. 1. Блок – схема алгоритма вычисления квадратного корня из действительного числа

Выводы

В предлагаемой работе, с целью построения оптимальных по быстродействию и оборудованию арифметических устройств вычислительной техники, проведен анализ итерационных методов решения нелинейного уравнения $f(x)=x^2 - a$, в результате чего произведен выбор оптимального по быстродействию итерационного метода Ньютона. Предложен способ и алгоритм аппаратной реализации вычисления корней нелинейного уравнения в сигнальных процессорах и микроконтроллерах с увеличенным быстродействием и минимальными аппаратными затратами. Использование метода Ньютона в комплексе с предложенным алгоритмом аппаратной реализации решения нелинейного уравнения позволяет разрабатывать принципиально новые высопроизводителные арифметические модули с минимальными аппаратными затратами выполняющие арифметические операции с действительными и комплексными числами. Аппаратная реализация операции извлечения квадратного корня может выполнять вычисления основного алгоритма в реальном времени без дополнительных временных задержек в командном цикле процессора, так как суммарное время выполнения алгоритма значительно уменьшается по сравнению с программной реализацией. Предложенный способ аппаратной реализации этого метода, позволяет:

– с учетом машинной алгебры [5-15], разработать арифметический модуль с минимальными аппаратными затратами;

– не увеличивать командный цикл процессора;

– получить временной выигрыш по сравнению с программной реализацией;

– вычислять модуль комплексного числа в двухканальных арифметических блоках;

– выполнять вычисления основного алгоритма в реальном времени.

Кроме этого, разработанный арифметический модуль, реализующий предложенный алгоритм вычисления квадратного корня, может быть использован как отдельный вычислительный элемент *core – генератора*, в системе автоматического проектирования *Xilinx* и др.

Список литературы

1. *Бахвалов Н.С.* Численные методы. – М.: Наука, 1975. – 535 с.

2. *Иванов В.В.* Методы вычисления на ЭВМ. Справочное пособие. – К.: Наукова думка, 1986. – 583 с.

3. *Ахо Д., Хопкрафт Д.* Построение и анализ вычислительных алгоритмов. – М.: Мир, 1979. – 535 с.

4. *Визор Я.Е.* Устройство для извлечения квадратного корня.// А.С. №1522198 (СССР). Бюл. №42, 1989.

5. *Бухштаб А.А.* Теория чисел. – М.: Просвещение, 1966. – 379с.

6. *Виноградов И.М.* Основы теории чисел. – М.: Наука, 1972. – 167с.

7. *Куликов В.В.* Алгебра и теория чисел. – М.: Наука, 1980. – 400с.

8. *Ленг С. А*лгебра. – М.: Мир, 1968. – 564с.

9. *Калужнин Л.А.* Введение в общую алгебру. – М.: Наука, 1973. – 239с.

10. *Фрид Э.* Элементарное введение в абстрактную алгебру. – М.: Мир, 1972. – 260с.

11. *Курош А.Г.* Курс высшей алгебры. – М.: Наука, 1976. – 431с.

12. *Воеводин В.В.* Линейная алгебра. – М.: Наука, 1980. – 400с.

13. *Кострикин А.И.* Введение в алгебру. – М.: Наука, 1979. – 495с.

14. 7.*Ван Дер Варден Б.Л.* Алгебра. – М.: Наука, 1986. – 624с.

15. *Скорняков Л.А.* Элементы алгебры. – М.: Наука, 1986. – 239с.