УДК 004.31

Жабин В. И., д-р техн. наук
Макаров В.В., канд. техн. наук

МЕТОД ВЫЧИСЛЕНИЯ ПОКАЗАТЕЛЬНОЙ ФУНКЦИИ ПРИ ПОРАЗРЯДНОМ ВВОДЕ И ВЫВОДЕ ИНФОРМАЦИИ

Национальный технический университет Украины "Киевский политехнический институт"

Рассматривается метод вычисления показательной функции в избыточной системе счисления при поразрядном вводе операнда и выводе результата, начиная со старших разрядов. Показана возможность применения указанного метода для уменьшения времени реализации последовательности операций, зависимых по данным

Введение

Известны итерационные методы вычисления элементарных функций «цифра за цифрой» [1-3]. Такие методы разрабатывались для применения в операционных устройствах (ОУ) параллельного типа, вычисления в которых в общем случае могут осуществляться только при наличии всех цифр операнда перед началом выполнения операции. В связи с этим при выполнении последовательности зависимых по данным операций (когда результат предыдущей операции является операндом для последующей) нельзя начинать выполнение очередной операции до тех пор, пока не будет получен параллельный код результата предыдущей операции, то есть указанные операции нельзя выполнять в режиме совмещения. Указанное совмещение можно осуществить при использовании квазипараллельных ОУ [4], обладающих следующими свойствами.

На каждом шаге вычислений в ОУ вводится по одному разряду операнда и с задержкой на несколько шагов формируется один разряд результата, начиная со старших разрядов. Если операции, принадлежащие последовательности, выполнять с помощью указанных ОУ, то разряд промежуточного результата, полученный на i-м шаге, в одном ОУ, может быть использован на $i+1$ -м шаге в другом ОУ в качестве очередного разряда операнда. В этом случае выполнение каждой последующей операции начинается не после завершения предыдущей, а сразу после получения первого (старшего) разряда результата этой операции. В связи с этим операции, принадлежащие последовательности нераспараллеливаемых операций, выполняются в режиме частичного совмещения, что создает предпосылки для уменьшения общего времени вычислений по сравнению с ОУ параллельного типа. Методы выполнения ряда операций в режиме совмещения рассмотрены, например, в работах [4, 5]. Ниже рассматривается возможность выполнения в указанном режиме операции $Y = a^X$.

Теоретические обоснование метода

Как известно [3], при вычислении функции $Y = a^X$ методом «цифра за цифрой» в параллельных ОУ, когда код аргумента представлен перед началом операции всеми разрядами, аргумент может быть преобразован к виду

$$X = \log_a Y_0 + \sum_{i=0}^{n} q_i \log_a (1+2^{-i}), \qquad (1)$$

где $0 \le X < 1$, $Y_0 \le Y$, $q_i \in \{0,1\}$, а функцию можно получить как

$$Y = Y_0 \prod_{i=0}^{n} (1+2^{-i})^{q_i}. \qquad (2)$$

В данном случае с каждым шагом значение суммы

$$R_j = \sum_{i=0}^{j} q_i \log_a (1+2^{-i}) \qquad (3)$$

приближается к X, не превышая его, а результатом преобразования аргумента является последовательность $q_1, q_2, \ldots q_j$, причем, $q_i \in \{0,1\}$. Такой процесс называется итерационным процессом со знако-

начиная с которого должен производиться вывод результата, определяется выражением

$$N = p -]\log_2 a[+ 1. \quad (8)$$

Максимальная погрешность вычислений составляет $\Delta = \Delta_1 + \Delta_2$, где Δ_1 – погрешность первой части вычислений (преобразование аргумента), Δ_2 – погрешность второй части вычислений (непосредственное вычисление функции).

В свою очередь $\Delta_1 = \varepsilon_1 + \varepsilon_2$, где ε_1 – погрешность, возникающая из-за отбрасывания членов бесконечной суммы преобразования аргумента, для которых $i > n$, а ε_1 – погрешность, возникающая из-за того, что константы $\log_a(1+2^{-i})$ не являются точными величинами.

Погрешность ε_1 составляет [3]

$$\varepsilon_1 < (1/\ln a)2^{-n}. \quad (9)$$

Определим ε_2, учитывая, что константы $\log_a(1+2^{-i})$ представлены с погрешностью, не превышающей $2^{-(n+1)}$, причем, в каждом цикле может быть k итераций:

$$\varepsilon_2 \le nk2^{-(n+1)}. \quad (10)$$

Добавив l дополнительных разрядов для представления констант $\log_a(1+2^{-i})$, можно уменьшить ε_2 до любого наперед заданного числа, например, $2^{-(n+1)}$. В этом случае из (10) получим $2^{-(n+1)} = k(n+l)2^{-(n+l+1)}$. Отсюда

$$l =]\log_2(k(n+l))[. \quad (11)$$

Погрешность Δ_1 без добавления дополнительных разрядов с учетом (9) и (10) составляет

$$\Delta_1 < (1/\ln a)2^{-n} + nk2^{-(n+1)}. \quad (12)$$

Добавив l дополнительных разрядов в соответствии с (11), получим

$$\Delta_1 < (1/\ln a)2^{-n} + 2^{-(n+1)}. \quad (13)$$

Погрешность Δ_2 возникает из-за выхода младших разрядов одного из слагаемых за пределы разрядной сетки в результате сдвига. Количество сдвигов m, которые не вносят этой погрешности, определим из выражения $k\sum_{i=1}^{m} i \le n$. Здесь $k\sum_{i=1}^{m} i$ – количество разрядов, на которые единица из (-1)-го разряда продвинется вправо в результате выполнения m циклов, каждый из которых состоит из k итераций. Так как $\sum_{i=1}^{m} i = (m^2+m)/2$, то m найдем из решения неравенства $(m^2+m) \le 2n/k$. Погрешность будут вносить $n-m$ циклов вычислений. Учитывая, что погрешность при сложении не превышает единицу младшего разряда, а в каждом цикле может быть k итераций, получим

$$\Delta_2 \le k(n-m)2^{-n}. \quad (14)$$

Погрешность Δ_2 можно уменьшить, например, до $2^{-(n+1)}$, добавив в сумматор l_1 дополнительных разрядов. Тогда $2^{-(n+1)} = k(n+l_1-m)2^{-(n+l_1)}$. Отсюда

$$l_1 =]\log_2(2k(n+l_1-m))[. \quad (15)$$

Таким образом, погрешность вычисления $Y = a^X$ без добавления дополнительных разрядов с учетом (12) и (14) составит

$$\Delta < (1/\ln a + 1.5nk - mk)2^{-n},$$

а погрешность с учетом дополнительных разрядов для представления констант (13) и дополнительных разрядов сумматора (15) определяется выражением

$$\Delta < (1/\ln a + 1)2^{-n}.$$

Таким образом, путем незначительного увеличения объема оборудования можно получить более высокую точность вычислений.

Аппаратная реализация метода

Рассмотрим устройство для вычисления функции $Y = e^X$. Условие (5) в этом случае удовлетворяется при $k = 2$, то есть в каждом цикле может быть от 0 до 2-х итераций. В соответствии с (7) и (8) определяем задержку формирования разрядов результата $p = N + 1 = 3$.

Таблиця 1

i	Номер такта	x_{i+1}	Константа	CM_1 (P1)	CM_2 (P2)	c_i	r_i	y_i
0	1	1		0.0000000 +0.1000000 0.1000000				
1	1 2 3	0	 $-\ln(1+2^{-1})$ $-\ln(1+2^{-1})$	0.1000000 +1.1001011 0.0001011 +1.1001011 1.1010110	 01.0000000 +00.1000000 01.1000000			
2	1 2	0	 $-\ln(1+2^{-2})$	0.0001011 +1.1100100 1.1101111	 **0**1.1000000	0	0	0
3	1 2 3	2	 $-\ln(1+2^{-3})$ $-\ln(1+2^{-3})$	0.0001011 +0.0010000 0.0011011 +1.1110001 0.0001100 +1.1110001 1.1111101	 0**1**.1000000 +00.0011000 0**1**.1011000	0	1	1
4	1 2 3	2	 $-\ln(1+2^{-4})$ $-\ln(1+2^{-4})$	0.0001100 +0.0001000 0.0010100 +1.1111001 0.0001101 +1.1111001 0.0000110	 0**1**.**1**011000 +00.0001101 01.1100101 +00.0001110 01.**1**110011	0	1	1
5	1 2 3	1	 $-\ln(1+2^{-5})$ $-\ln(1+2^{-5})$	0.0000110 +0.0000010 0.0001000 +1.1111101 0.0000101 +1.1111101 0.0000010	 01.1**1**10011 +00.0000111 01.1111010 +00.0000111 10.0**0**00001	1	0	2
6	1 2 3	1	 $-\ln(1+2^{-6})$ $-\ln(1+2^{-6})$	0.0000010 +0.0000001 0.0000011 +1.1111110 0.0000001 +1.1111110 1.1111111	 10.0**0**00001 +00.0000100 10.0**0**00101	0	0	0
7	2 3	-	 $-\ln(1+2^{-7})$ $-\ln(1+2^{-7})$	0.0000001 +1.1111111 0.0000000 +1.1111111 1.1111111	 10.0000**1**01 +00.0000010 10.0000111	0	0	0
8					10.0000**1**11	0	1	1
9					10.000011**1**	0	1	1
10					10.0000111	0	1	1

Выводы

Предложенный метод позволяет выполнять последовательности зависимых по данным операций в режиме частичного совмещения. Благодаря поразрядному вводу и выводу данных разряд промежуточного результата, полученный на i-м шаге, в одном ОУ, может быть использован на $i+1$ -м шаге в другом ОУ в качестве очередного разряда операнда.

В этом случае выполнение каждой последующей операции начинается не после завершения предыдущей, а сразу после получения старшего разряда результата этой операции, то есть с небольшой задержкой в числе шагов. Это создает предпосылки для уменьшения общего времени вычислений по сравнению с ОУ параллельного типа.

Благодаря последовательному вводу и выводу информации сокращается необходимое число внешних выводов ОУ, что дает дополнительные преимущества по сравнению с ОУ параллельного типа. В частности, при использовании ПЛИС сокращается необходимое число ячеек ввода-вывода. В свою очередь, увеличение числа свободных выводов ПЛИС повышает возможности использования функционального ресурса микросхемы для другой аппаратуры.

Список литературы

1. *Volder J.E.* The CORDIC trigonometric computing technigue // IRE Trans. on Electr. Comp. – 1959. – Vol. 8, No 3. – P. 330–334.

2. *Байков В.Д., Смолов В.Б.* Специализированные процессоры: итерационные алгоритмы и структуры. – М.: Радио и связь, 1985. – 288 с.

3. *Лапыгин Е.Д.* Аппаратные методы ускорения вычисления некоторых элементарных функций // Вопросы радиоэлектроники. – 1964. – Сер. 11, № 7. – С. 3–17.

4. *Жабин В.И., Корнейчук В.И., Тарасенко В.П.* Некоторые машинные методы вычисления рациональных функций многих аргументов // Автоматика и телемеханика. – 1977. – № 12. – С. 145–154.

5. *Жабин В.И., Корнейчук В.И., Тарасенко В.П.* Методы вычисления некоторых функций при поразрядном вводе и выводе информации // Известия ВУЗов. Приборостроение. – 1978. – № 2. – С. 64–69.

6. Устройство для вычисления функции $y = e^x$: А.с. 662937 СССР, МКИ G06F 7/38 /*В.И.Жабин, В.И.Корнейчук, В.В.Макаров, В.П.Тарасенко.* – № 2398563/18 – 24; заявлено 16.08.76; обубл. 15.05.79, Бюл. 18. – 7 с.