УДК 681.325.5

С. Н. ФУРС

АЛГОРИТМ ВЫЧИСЛЕНИЯ ДВОИЧНОГО ЛОГАРИФМА

Известны устройства [1, 2], предназначенные для вычисления двоичных логарифмов, в основу которых положен разностно-итерационный процесс суммирования приращений констант вида $\alpha_i = \log_2(1+q_i 2^{-i})$. Быстродействие таких устройств определяется количеством итерационных шагов, численно равным разрядности аргумента. Кроме того, устройства требуют хранения в ПЗУ двух массивов констант: $\alpha_i = \log_2(1+2^{-i})$ и $\alpha_j = \log_2(1-2^{-j})$. Несколько иной подход к реализации итерационного процесса позволяет сократить объем ПЗУ и значительно ускорить процесс вычисления логарифма.

Пусть заданный аргумент представлен n-разрядным нормализованным двоичным числом $(2^{-1} \leqslant Z < 1)$. Представим Z в форме:

$$Z = 2^{-1} \prod_{i=1}^{k} (1 + 2^{-l_i})^{q_i}, \qquad (1)$$

где $1 \leqslant k \leqslant n$, $1 \leqslant l_i \leqslant n$, $q_i \in \{+1, -1\}$.

Очевидно, что для любого значения Z найдется такая последовательность операторов q_i, что определяемая ими сумма констант вида $\alpha_i = q_i \log_2(1 + 2^{-l_i})$ будет стремиться к значению искомой функции:

$$y = \log_2 Z = -1 + \sum_{i=1}^{k} q_i \log_2 (1 + 2^{-l_i}). \qquad (2)$$

Оператор q_i и число l_i подбираются таким образом, чтобы очередное приближение аргумента, определяемое зависимостью (1), $Z'_{i+1} = 2^{-1} \prod_{j=1}^{i+1} (1 + 2^{-l_j})^{q_j} = Z'_i (1+2^{-l_{i+1}})^{q_{i+1}}$ оказалось как можно ближе к заданному аргументу Z. Оператор q_{i+1} определяется знаком частичного остатка—разностью между заданным аргументом и очередным его приближением;

$$q_{i+1} = \begin{cases} +1, & \text{если } S'_i > 0 \\ -1, & \text{если } S'_i < 0 \\ \text{останов}, & \text{если } S'_i = 0, \end{cases}$$

где $S'_i = Z - Z'_i$.

Очередное приближение аргумента и значение частичного остатка в зависимости от значения оператора q_i, найденного по результатам предыдущей итерации, определяется следующими соотношениями,

$$\begin{gathered} Z'_i = Z'_{i-1}(1 + 2^{-l_i}),\ S'_i = S'_{i-1} - Z'_{i-1} 2^{-l_i}, \text{ если } q_i = +1, \\ Z'_i = \frac{Z'_{i-1}}{1 + 2^{-l_i}},\ S'_i = \frac{S'_{i-1} + Z 2^{-l_i}}{1 + 2^{-l_i}}, \text{ если } q_i = -1 \end{gathered} \qquad (3)$$

с начальными условиями $Z'_0 = 2^{-1}$, $S'_0 = Z - 2^{-1}$.

Для устранения необходимости проведения операции деления в соотношениях (3) при $q_i = -1$, можно вместо величин Z, Z'_i и S'_i пользоваться «деформированными» величинами; $Z^*_i = K_i Z$: $Z_i = K_i Z'_i$, $S_i = K_i S'_i$, где $K_i = \prod_{j=1}^{i} \overline{K}_j$, а $\overline{K}_i = \begin{cases} 1, & \text{если } q_i = +1 \\ (1 + 2^{-l_i}), & \text{если } q_i = -1. \end{cases}$ Иначе говоря, можно вместо очередного приближения аргумента при $q_k = -1$ $Z'_k = \frac{Z'_{k-1}}{1 + 2^{-l_k}}$ рассматривать его «деформированное» значение $Z_k = Z'_k (1 + 2^{-l_k}) = Z'_{k-1}$ при

условии умножения на такой же коэффициент $(1+2^{-l_k})$ заданного аргумента Z, при определении частичного остатка.

Скорость сходимости итерационного процесса в большой степени зависит от правильного определения очередного числа l_{i+1}. Это число следует подбирать так, чтобы очередной частичный остаток, определяемый соотношениями

$$\left.\begin{aligned} S_{i+1} &= S_i - Z_i 2^{-l_{i+1}}, \text{ если } q_i = +1 \\ S_{i+1} &= S_i + Z_i^* 2^{-l_{i+1}}, \text{ если } q_i = -1 \end{aligned}\right\}. \qquad (4)$$

был, по крайней мере, в два раза меньше предыдущего остатка. Это произойдет в том случае, когда при выбранном значении l_{i+1} в процессе сдвига вправо величины Z_i (или Z_i^*) ее старший значащий бит совпадает со старшим значащим битом предыдущего остатка S_i.

Чтобы повысить точность вычисления величины S_i, вместо сдвига вправо величины Z_i (или Z_i^*) целесообразно сдвигать влево на такое же количество разрядов значение предыдущего остатка. В этом случае операция сдвига не сопровождается потерей информации за счет выхода значащих разрядов за разрядную сетку. При этом количество разрядов m_i, на которое надо сдвинуть влево значение предыдущего остатка, чтобы совпадали старшие значащие биты в величинах S_{i-1} и Z_{i-1}, определяет значение очередной степени двойки в выражении (1): $l_i = l_{i-1} + m_i$.

Операцию сдвига частичного остатка до совпадения старших значащих битов в величинах S_i и Z_i можно назвать нормализацией частичного остатка.

Таблица 1

i	q_i	l_i	Z_i	Z_i^*	S_i	y_i
0		0	0,1000000000	0,1110001000	0,0110001000	−1,0000000000
			0,1000000000		0,1100010000	−1,0000000000
1	+1	1	+0,0100000000	0,1110001000	−0,1000000000	+0,1001010111
			0,1100000000		0,0100010000	−0,0110101001
			0,1100000000		0,1000100000	−0,0110101001
2	+1	2	+0,0011000000	0,1110001000	−0,1100000000	+0,0101001010
			0,1111000000		−0,0011100000	−0,0001011111
				0,1110001000	−0,1110000000	−0,0001011111
3	−1	4	0,1111000000	+0,0000111000	+0,1110001000	−0,0001011010
				0,1111000000	+0,0000001000	−0,0010111001
			0,1111000000		0,1000000000	−0,0010111001
4	+1	10	+0,0000000000	0,1111000000	−0,1111000000	+0,0000000001
			0,1111000000		−0,0111000000	−0,0010111000

Таким образом, рекуррентные соотношения алгоритма могут быть представлены в форме:

$$q_i = \begin{cases} +1, \text{ если } S_{i-1} > 0 \\ -1, \text{ если } S_{i-1} < 0 \\ \text{останов, если } S_{i-1} = 0 \end{cases}; \quad l_i = l_{i-1} + m_i$$

Если $q_i = +1$, то;	Если $q_i = -1$, то:
$Z_i = (1 + 2^{-l_i}) Z_{i-1}$,	$Z_i = Z_{i-1}$
$Z_i^* = Z_{i-1}^*$	$Z_i^* = (1 + 2^{-l_i}) Z_{i-1}^*$
$S_i = (S_{i-1})_н - Z_{i-1}$,	$S_i = (S_{i-1})_н + Z_{i-1}^*$
$y_i = y_{i-1} + \log_2 (1 + 2^{-l_i})$.	$y_i = y_{i-1} - \log_2 (1 + 2^{-l_i})$; $i = \overline{1, k}$

с начальными условиями $Z_0 = 2^{-1}$, $Z_0^* = Z$, $S_0 = Z - 2^{-1}$, $y_0 = -1$, $l_0 = 0$. Результат: $y_k = \log_2 Z$. Максимальное значение k номера итерации i — это номер такой итерации, для которой $l_i > n \cdot (S_{i-1})_н$ обозначает нормализованную величину S_{i-1}.

Таблица 2

Разрядность аргумента, n	Среднее число итераций, κ	Среднеквадратическая ошибка, ско	Максимальная погрешность, абс. значение
10	4,2	0,92	3
11	4,6	0,69	2
12	4,9	0,75	2
13	5,3	0,76	2
14	5,7	0,79	3
15	6,1	0,76	2
16	6,5	0,90	3
17	6,9	0,88	3
18	7,3	0,95	4
19	7,7	0,94	3
20	8,1	1,03	4

Последовательное уменьшение частичного остатка (по крайней мере, в два раза после каждой итерации) гарантирует сходимость итерационного процесса. Табл. 1 иллюстрирует порядок выполнения алгоритма для аргумента $Z = (0{,}1110001000)_2$.

Алгоритм был подвергнут экспериментальной проверке, для чего на ЭВМ ЕС-1022 моделировалась специализированная ЦВМ с заданной разрядной сеткой ($n = 10$—20). На этой ЦВМ по описанному алгоритму вычислялась функция $y = \log_2 Z$ для всех 2^n значений аргументов, возможных при данной разрядности n. Результаты эксперимента сведены в табл. 2, в которой погрешности показаны в единицах младшего разряда регистра.

Результаты эксперимента подтвердили значительно более быструю (в среднем в 2,5 раза) сходимость итерационного процесса по сравнению с ранее описанными алгоритмами при уменьшении аппаратурных затрат за счет использования в ПЗУ одного массива констант.

ЛИТЕРАТУРА

1. Оранский А. М., Немытов Б. В. А. с. 448459 (СССР). Цифровое устройство для логарифмирования двоичных чисел.— Опубликовано в Б. И., 1975, № 40, с приоритетом от 02.03.72.

3. Оранский А. М. Аппаратные методы в цифровой вычислительной технике.— Минск, 1977, с. 158.

Поступила в редакцию
05.05.80

Кафедра ЭММ