УДК 681.3.055

З. Б. КРУГЛЯК, Г. Г. МАТУШКИН
(Новосибирск)

ИСПОЛЬЗОВАНИЕ СИСТЕМЫ СЧИСЛЕНИЯ В ОСТАТОЧНЫХ КЛАССАХ ПРИ ТАБЛИЧНЫХ МЕТОДАХ ОБРАБОТКИ

С появлением и развитием голограммных запоминающих устройств, обладающих большим объемом памяти при высокой плотности упаковки 10^6—10^8 бит/см^2, а также электронных ЗУ в интегральном исполнении снова возник интерес к табличным методам обработки цифровой информации [1—3]. Использование методов табличной арифметики позволяет строить арифметические устройства с высоким быстродействием. Однако построение арифметических устройств табличного типа даже для 16-разрядных операндов в двоичной позиционной системе счисления является нереальным, поскольку это потребовало бы запоминающих устройств емкостью $2^{32} \cdot 2^4 = 10^{11}$ бит. Реальные возможности построения устройств обработки информации методами табличной арифметики на основе как ГЗУ, так и электронных ЗУ в интегральном исполнении открывает использование системы счисления в остаточных классах (СОК).

Система счисления в остаточных классах характеризуется тем, что число в ней представляется в виде набора остатков

$$N = (\alpha_1, \alpha_2, \ldots, \alpha_n)$$

по выбранным основаниям $P_1, P_2, \ldots, P_n$, которые представляют ряд положительных целых взаимно-простых чисел. Остаток α_i определяется из выражения

$$\alpha_i = N - \left[\frac{N}{P_i}\right] P_i,$$

где $\left[\frac{N}{P_i}\right]$ означает целую часть от деления N на P_i.

Достоинствами СОК являются [4]:

независимость образования разрядов числа, в силу чего каждый разряд несет информацию обо всем исходном числе. Отсюда независимость разрядов числа друг от друга и возможность их независимой параллельной обработки;

малоразрядность остатков, представляющих число, что дает широкие возможности для использования табличной арифметики.

Основными недостатками СОК являются:

невозможность визуального сопоставления чисел, так как внешняя запись числа не дает представления о его величине;

отсутствие простых признаков выхода результатов операций за пределы диапазона чисел, которые могут быть представлены в СОК при заданных основаниях $P_1, P_2, \ldots, P_n$. Этот диапазон обозначается как $[0, R]$, где $R = P_1, P_2 \ldots P_n$;

получение во всех случаях точного результата операции, что исключает возможность непосредственного приближенного выполнения операций, округления результата;

сложность перевода из СОК в позиционную систему.

Из-за указанных недостатков и отсутствия достаточно удобных алгоритмов, позволяющих обходить или устранять эти недостатки, СОК оставалась малоудобной для реализации табличной арифметики.

В предлагаемой статье сделана попытка разработать приемлемые для табличной арифметики алгоритмы выполнения немодульных операций (деление, сравнение, перевод из СОК в позиционную систему). Немодульные операции являются наиболее громоздкими и неудобными, они — основная причина, сдерживающая применение СОК для табличной арифметики.

Умножение с округлением. Выполнение умножения $A \times B$ связано с представлением чисел в расширенном диапазоне $P = P_1 P_2 \dots P_{n-1} P_n P_{n+1} \dots P_{n+r} > (R-1)^2$, так как $A \in [0, R)$; $B \in [0, R)$, а их произведение находится в диапазоне $A \times B \in [0, (R-1)^2)$.

Умножение в СОК выполняется следующим образом:

$$A = (\alpha_1, \alpha_2, \dots, \alpha_{n+r});$$
$$B = (\beta_1, \beta_2, \dots, \beta_{n+r});$$
$$A \times B (\delta_1, \delta_2, \dots, \delta_{n+r});$$
$$\delta_i = \alpha_i \beta_i (\mathrm{mod}\, P_i).$$

В табличной арифметике умножение выполняется за один такт и отдельно по каждому из оснований. Округление полученного сомножителя путем формального деления на основание системы P_i с дальнейшим раскрытием неопределенности результата α_i по этому основанию [4] является неприемлемым из-за очень большого числа элементарных операций *, что резко снижает быстродействие и требует значительных массивов памяти.

Алгоритм округления, предложенный в [5], пригоден лишь при выполнении вычислений в режиме с фиксированной запятой. Ниже предлагается алгоритм округления полученного результата произведения, который во многом устраняет указанные недостатки и позволяет осуществить операции в режимах с фиксированной и с плавающей запятой. Последовательность выполнения операций при реализации данного алгоритма округления результата произведения следующая:

1. К результату произведения прибавляется $x(P_{n+r})/2$, где

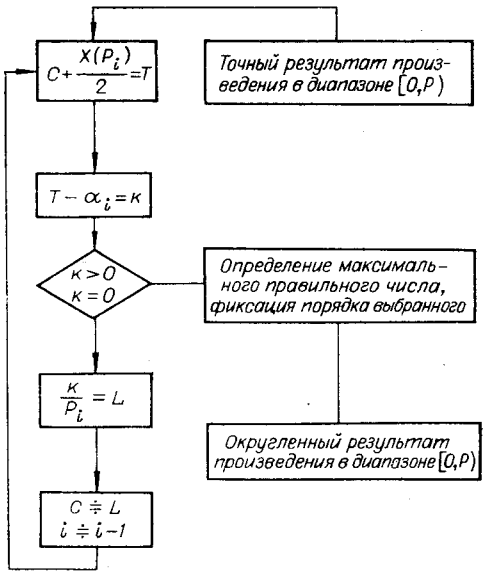
$$x(P_{n+r}) = \begin{cases} P_{n+r}, & \text{если } P_{n+r} \text{ четно;} \\ P_{n+r-1}, & \text{если } P_{n+r} \text{ нечетно.} \end{cases}$$

2. Из полученного результата вычитается остаток этого результата по основанию P_{n+r}.

3. Производится формальное деление полученной разности на основание P_{n+r}. Так как остаток по основанию P_{n+r} после приведенных операций равен нулю, то число делится на P_{n+r} без остатка и формальное деление дает правильный результат. Результат деления запоминается. Неопределенность по основанию не раскрывается, она отбрасывается, и полученный результат представляется в диапазоне $P^1 = P_1 P_2 \dots P_{n+r-1}$. Прибавление $x(P_{n+r})/2$ необходимо для исключения накопления ошибки одного знака. Описанный алгоритм повторяется по основанию P_{n+r-1} и т. д., пока на одном из тактов округления не будут получены по всем основаниям нули. В зависимости от того, на каком такте округления были получены нули, определяется такт округления, при котором было получено наибольшее правильное число. (Под правильным числом понимается число, которое находится в интервале $[0, R)$). Сокращение на ряд оснований фиксируется как порядок числа. Блок-схема описанного алгоритма приведена на рисунке.

---

* Под элементарной операцией в данном случае понимаются операции, выполняемые за один такт выборки из памяти.



Здесь $C=A\times B$, $i=n+r$

Если полученный результат участвует в дальнейших вычислениях, необходимо полученное правильное число A представить в расширенном диапазоне P. Для этого необходимо определить остатки по основаниям, на которые произведено сокращение. Предлагается определять их по следующему алгоритму:

1. Подставляем на место остатков по сокращенным основаниям нули. При этом число принимает вид $(\alpha_1, \alpha_2, \ldots, \alpha_{k-1}, 0, 0, \ldots, 0)$.

2. Проводим нулевизацию до первого основания P_k, по которому произведено сокращение. В результате имеем выражение $(0, \ldots, 0, \alpha_k^{k-1}, \alpha_{k+1}^{(k-1)}, \ldots, \alpha_{n+r}^{(k-1)})$.

3. По полученному в результате нулевизации остатку $\alpha_k^{(k-1)}$ определяем интервал J нахождения числа в диапазоне $R_{k-1}=P_1P_2\ldots P_{k-1}$ по формуле $J=(\alpha_k^{(k-1)}m_k') \pmod{P_k}$, где m_k' — вес ортогонального базиса B_k' в системе оснований $P_1, P_2, \ldots, P_k$.

4. Решаем уравнение относительно x:

$$\frac{J+xm_{k-1}'}{P_k}-\left[\frac{J+xm_{k-1}'}{P_k}\right]=0,$$

где $[\ ]$ означает целую часть числа.

Это уравнение имеет бесчисленное множество решений. Однако в качестве решения берется минимальное положительное значение x, удовлетворяющее указанному уравнению.

Искомое значение α_k для результата произведения $A\times B=C$ есть $(x+m_k') \bmod P_k$.

5. По полученному α^k и $\alpha_k^{(k-1)}$ находится следующая константа нулевизации и определяется $\alpha_{k+1}^{(k)}$ ((k) означает k-й цикл нулевизации).

Для $\alpha_{k+1}^{(k)}$ повторяются пункты 3—5; так продолжается до нахождения α_{n+r}.

Предложенный алгоритм расширения диапазона особенно удобен в табличной арифметике, так как в этом случае можно исключить ряд операций: исчезают пункты 3, 4 и по $\alpha_k^{(k-1)}$ сразу определяется α_k. Предложенный алгоритм округления позволяет производить сокращение на основание системы за $3+\frac{n+k-1}{k}$ тактов, где n — общее число оснований системы, k — количество оснований, на которые производится сокращение.

Алгоритм же, описанный в [4], позволяет выполнять округление на одно основание за $4+n$ тактов и не зависит от числа сокращенных оснований. Помимо этого, при использовании данного алгоритма для каждого сокращаемого числа необходимо знать его минимальный след, что требует дополнительно n тактов. Таким образом, округление числа, выполненное по алгоритму, описанному в [4], требует в общих случаях $4+2n$ тактов на одно основание, т. е. более чем в 2 раза превышает число тактов предложенного алгоритма.

Перевод чисел из СОК в двоичную позиционную систему (ДПС). Известны различные алгоритмы перевода чисел из СОК в позиционную систему [4—6]. Все они требуют значительных затрат времени. Алгоритм перевода и устройство, приведенные в [7], требуют значительных аппаратурных затрат, а полученный в результате перевода полиадический код требует еще и перевода в двоичную или десятичную системы счисления. Указанные алгоритмы перевода не учитывают особенностей табличной арифметики. Поэтому ниже предлагается простой и достаточно быстрый алгоритм перевода из СОК в ДПС, эффективность которого определяется использованием табличных методов осуществления арифметических операций. Последовательность выполнения операций упомянутого алгоритма следующая:

1. Число A, представленное в СОК и требующее перевода в двоичную систему счисления, подвергается нулевизации. Значение констант нулевизации M_{ij} известно как в СОК, так и в ДПС.

2. Параллельно процессу нулевизации на двоичном сумматоре определяется сумма констант, нулевизирующих число.

3. Полученное в результате нулевизации число $A=(0, 0, \ldots, 0\ \alpha_n^{(n-1)})$ находится на границе диапазона нахождения числа A [4], и величина его определяется следующим выражением:

$$A'=\left(\left(\alpha_n^{(n-1)}m_n\right) \bmod P_n\right)\frac{R}{P_n},$$

где $\alpha_n^{(n-1)}$ — остаток по основанию P_n после проведения нулевизации, R — диапазон представления чисел, m_n — вес ортогонального базиса B_n в системе оснований $P_1, P_2, \ldots, P_n$.

4. Для получения величины числа A в двоичной системе счисления достаточно из величины A' вычесть величину сумм констант нулевизации, что также осуществляется на двоичном сумматоре. Преимуществами приведенного алгоритма являются отсутствие относительно сложных операций умножения, определения ранга числа переполнения и т. д., которые имеют место в описанных ранее алгоритмах, затрудняя их реализацию и уменьшая их быстродействие.

Деление. В любой системе счисления выполнение операции деления представляет значительные трудности и требует времени на порядок больше, чем выполнение других арифметических операций [4]. Для систем остаточных классов сложности деления усугубляются тем, что остаток частного по данному основанию не является функцией остатков делимого и делителя по этому основанию, а зависит от значений делимого и делителя в целом.

Использование алгоритмов деления, описанных в [4, 6], требует дополнительного объема памяти. Ниже предлагается алгоритм деления, который строится на основе приведенных выше алгоритмов округления, практически не требует дополнительного объема памяти и в то же время по быстродействию не уступает алгоритмам деления, приведенным в [4].

Пусть A — делимое, B — делитель представлены в расширенном диапазоне $P=P_1, P_2, \ldots, P_{n+r}$. Тогда последовательность операций при выполнении предлагаемого алгоритма деления может быть представлена в следующем виде:

1. Производится сокращение диапазона представления чисел A и B, как описано выше. При этом делимое и делитель делятся на одни и те же основания расширенного диапазона представления. Деление производится до тех пор, пока делитель не станет равным нулю. Первым приближением частного и будет частное от деления делимого C_1:

$$C_1=\frac{A}{P_{n+r}P_{n+r-1}\cdots P_k},$$

где P_k — основание расширенной системы представления чисел, после деления на которое в следующем такте делитель стал равен нулю.

2. Вычисляется невязка L.

3. L представляется в расширенном диапазоне $P=P_1P_2, \ldots, P_{n+r}$.

4. Производится деление L_1 на те же основания системы, что и A. Получаем

$$C_2 = \frac{L_1}{P_{n+r}P_{n+r-1}\cdots P_k}.$$

Если C_2 не равно нулю, то вычисляется вторая невязка $L_2{:}L_2=A-B(C_1+C_2)$. Над L_2 выполняются те же операции, что и над L_1. Процесс продолжается до тех пор, пока невязка C_i не станет равной нулю.

Если числа в СОК представлены в искусственной форме, то знак невязки получается автоматически, в противном случае знак невязки следует определить.

Конечный результат частного $C=C_1+C_2+\ldots+C_i$.

Сложение, вычитание, сравнение чисел. Сложение и вычитание выполняются на отдельных таблицах за один такт. Возможное переполнение при этом может быть округлено по приведенному выше алгоритму.

С точки зрения табличной арифметики в целях экономии памяти наиболее целесообразно раздельное введение знака числа и анализ его с помощью специальной логической схемы. Если обозначить знак «+» через 1, а знак «—», через 0, то можно правило определения знака представить в следующем виде:

1. Если $\operatorname{sign}(A)=\operatorname{sign}(B)$, то $\operatorname{sign}(A+B)=\operatorname{sign}(A)$;

$$\operatorname{sign}(A\times B)=\operatorname{sign}\left(\frac{A}{B}\right)=1; \quad \operatorname{sing}(A-B)=\begin{cases}\operatorname{sign}(A) & \text{при } |A|>|B|;\\ \operatorname{sign}(B) & \text{при } |A|<|B|.\end{cases}$$

2. Если $\operatorname{sign}(A)\neq\operatorname{sign}(B)$, то $\operatorname{sign}(A\times B)=\operatorname{sign}\left(\frac{A}{B}\right)=0$;

$$\operatorname{sign}(A+B)=\begin{cases}\operatorname{sign}(A) & \text{при } |A|>|B|;\\ \operatorname{sign}(B) & \text{при } |A|<|B|;\end{cases} \quad \operatorname{sign}(A-B)=\operatorname{sign}(A).$$

Сравнение чисел по модулю может быть произведено следующим образом. Сравниваемые числа A и B берем со знаком «+» и производим вычитание $A-B$ в расширенном диапазоне. При равенстве нулю всех остатков результата вычитания $|A|=|B|$. Если хотя бы один из остатков разности не равен нулю, проводим нулевизацию по n основаниям. Если в результате нулевизации получен «0» по всем дополнительным основаниям, то $|A|>|B|$, в противном случае $|A|<|B|$.

Рассмотренные выше алгоритмы разрабатывались применительно к использованию табличных методов обработки и позволяют выполнять все арифметические действия в СОК, а также округление результатов и переводы чисел из СОК в ДПС и обратно, используя лишь четыре таблицы и небольшое число констант. Арифметическое устройство при работе в СОК с использованием табличных методов выполняется из отдельных блоков ПЗУ, которые могут быть электронными, оптическими или голограммными в зависимости от необходимых объемов памяти и быстродействия. Для осуществления полного арифметического устройства для операций с 16-разрядными двоичными числами по приведенным выше алгоритмам потребуется 6 модулей ПЗУ, каждый емкостью примерно 10^5 бит. Модули ПЗУ такой емкости в достаточно компактном исполнении можно ожидать в недалеком будущем как в интегральном микроэлектронном исполнении [8], так и оптического типа [9].

ВЫВОДЫ

Представление чисел в системе остаточных классов с использованием предложенных рациональных алгоритмов выполнения операций позволяет применить в АУ, основанных на принципах табличной арифметики, ПЗУ с объемом памяти, вполне реальном в ближайшем будущем ($10^5 \div 10^6$ бит). Использование таких модулей в табличных АУ с временем выборки порядка десятков наносекунд позволяет создать арифметические устройства с фиксированной запятой с быстродействием порядка десятков миллионов операций в секунду независимо от разрядности операндов. Существенным преимуществом АУ табличного типа в системе остаточных классов, кроме быстродействия, является схемотехническая и конструктивная модульность и функциональная простота.

ЛИТЕРАТУРА

1. Ф. Камме. Проектирование сложных логических устройств с помощью типовых ПЗУ.— «Электроника», 1970, т. 43, № 1.
2. А. Хемел. Выполнение операций с помощью ПЗУ.— «Электроника», 1970, т. 43, № 10.
3. И. И. Коршевер, Г. Г. Матушкин, П. Е. Твердохлеб. Цифровые функциональные преобразования на основе оптических запоминающих устройств.— «Автометрия», 1974, № 1.
4. И. Я. Акушский, Д. И. Юдицкий. Машинная арифметика в остаточных классах. М. «Сов. радио», 1968.
5. Л. А. Орлов, Ю. М. Попов. Оптоэлектронное арифметическое устройство в системе остаточных классов.— «Автометрия», 1972, № 6.
6. А. Свобода. Развитие вычислительной техники в Чехословакии. Система счисления остаточных классов.— «Кибернетический сборник». М. «Мир», 1964, № 8.
7. И. Я. Акушский и др. Преобразователь кода из системы остаточных классов в полиадический код.— Авт. свид-во № 328448. ОИПОТЗ, 1972, № 6.
8. Jules H. Gildr. Memories turn to never materials, but care hangs on of the old reliable. — "Electronic Design", 1973, № 11.
9. Оптическое ПЗУ.— «Экспресс-информация. ВТ», 1971, № 44.

Поступила в редакцию 14 июня 1974 г;
окончательный вариант — 6 января 1975 г.

УДК 621.378:681.33

В. И. ФЕЛЬДБУШ, Ю. В. ЧУГУЙ

(*Новосибирск*)

КОГЕРЕНТНО-ОПТИЧЕСКИЕ СИСТЕМЫ ОБРАБОТКИ СИГНАЛОВ НА ОСНОВЕ ПРИМЕНЕНИЯ СИЛУЭТНЫХ ФИЛЬТРОВ

Многоканальная фильтрация сигналов (радиочастотных, оптических, ультразвуковых и т. д.) — одно из перспективных применений когерентно-оптических систем обработки информации. Задача синтеза таких систем сводится, как известно, к задаче синтеза фильтров пространственных частот [1]. Реализация фазовых и амплитудных компонентов фильтров предполагает использование многоградационных фазовых, а также полутоновых транспарантов с большим диапазоном пропускания, изготовление которых связано с серьезными технологическими трудностями. Правда, эти трудности частично преодолеваются при получении фильтров по методу Вандер Люгта [2]. Однако в этом случае удается получать лишь согласованные фильтры.