УДК 519.6

РАЗЛОЖЕНИЕ АРИФМЕТИЧЕСКОЙ ОПЕРАЦИИ «ДЕЛЕНИЕ» В РЯД ПОБИТОВЫХ СДВИГОВ

О.В. СУХОРУКОВ
(Полоцкий государственный университет)

Для упрощения реализации арифметической операции «деление» при проектировании арифметико-логического устройства (АЛУ) разработан метод разложения этой операции в ряд побитовых сдвигов. Показаны преимущества предложенного метода, достигнутые использованием двух ключевых операций «сдвиг» и «суммирование», – высокая скорость расчёта, задаваемая точность результата, надёжность и устойчивость. Побитовый сдвиг вправо (влево) не требует применения логических элементов (И, ИЛИ, НЕ), а значит, является одной из самых простых и быстрых процедур. Простота внедрения метода в процесс проектирования достигается использованием «сдвига» в связке с базовой операцией большинства АЛУ – «суммированием».

Принципиальная схема АЛУ, вплоть до отдельно взятого компонента, строится на элементарных базисных элементах (И, ИЛИ, НЕ), подчиняется бинарной логике, а значит, принцип работы схемы базируется исключительно на булевой алгебре.

Одна из самых простых операций при проектировании схемы – реализация побитового сдвига. При сдвиге каждого бита в разрядной сетке первоначального числа на n разрядов влево исходное значение умножается на 2^n, а вправо – делится на 2^n. Иными словами, побитовый сдвиг – это реализация арифметической операции «умножение» («деление») первоначального значения на 2^n, где n – число разрядов сдвига.

Пример

Пусть задано первоначальное число $5_{10} = 0101_2$. Тогда побитовый сдвиг на один разряд влево ($01010_2 = 10_{10}$) соответствует умножению $5 \cdot 2^1$. А побитовый сдвиг на один разряд вправо ($0010{,}1_2 = 2{,}5_{10}$) соответствует делению $\frac{5}{2^1}$.

Для реализации операции побитового сдвига при проектировании принципиальной схемы абсолютно не требуется применения логических элементов (рис. 1), достаточно ввести дополнительный разряд(ы) на выходе (разряд $B0$) и сместить биты разрядной сетки входного значения относительно соответствующей разрядной сетки выходного значения на число добавленных разрядов. Схема на рисунке 1 реализует умножение входного четырехразрядного числа ($A0$ – $A3$) на число 2. Результатом операции является пятиразрядное число ($B0$ – $B4$).

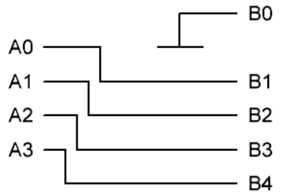

Рис. 1. Умножение четырехразрядного числа на число 2

Реализация арифметической операции «умножение» $x \cdot y$, где $x \in R$, $y \in R \setminus \{0, 2^n\}, n \in N$, также не вызывает затруднений. Достаточно разложить умножение на две отдельных операции: сдвиг и суммирование.

Допустим, необходимо умножить входное четырехразрядное число x на число 3. Для этого достаточно произвести побитовый сдвиг x на один разряд влево, что соответствует умножению на 2, и прибавить значение x: $x \cdot 3 = x + 2 \cdot x$. На выходе получим шестиразрядное число. На первое слагаемое ($S1$) сумматора (SM) посылается x, на второе слагаемое ($S2$) – $2x$. Принципиальная схема представлена на рисунке 2.

Таким образом, сдвиг и суммирование при реализации арифметической операции «умножение» есть разложение умножения в сумму сдвигов (ряд сдвигов).

В предыдущем примере произведение $x \cdot 3$ представлено суммой двух сдвигов $x \cdot 1$ и $x \cdot 2$ $(x \cdot 3 = x \cdot 1 + x \cdot 2)$. То есть сформирован ряд сдвигов $x \cdot 3 = \sum_{k=1}^{2} k \cdot x$.

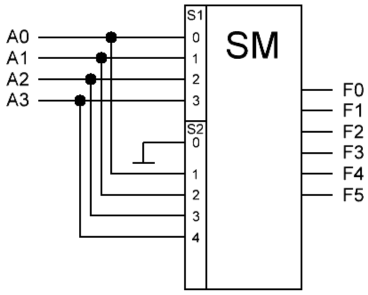

Рис 2. Реализация умножения
четырёхразрядного числа x на число 3

Пример

Умножение произвольного числа x на 7 можно представить как $x \cdot 7 = x + 2x + 4x = \sum_{k=0}^{2} 2^k x$.

Реализовать арифметическую операцию «деление» $\frac{x}{y}$, где $x \in R$, $y \in R \setminus \{0, 2^n\}$, $n \in N$, уже не так просто. Безусловно, существует целый ряд способов, однако даже если взять за основу один из классических алгоритмов деления двоичных чисел:

- алгоритм деления в прямых кодах с восстановлением (без восстановления) остатка [1, с. 38];
- алгоритм деления в дополнительных кодах [1, с. 40];
- реализация операции средствами базового набора компонентов становится крайне трудоемким процессом.

Поэтому основная задача заключается в получении более простого (по отношению к существующим) метода реализации арифметической операции «деление» при обязательном условии высокой скорости вычисления.

Принимая в расчет все поставленные требования, а также учитывая, что с помощью побитового сдвига вправо можно вычислить частное $\frac{x}{y}$ при $x \in R$, $y \in \{2^n\}$, $n \in N$, обратимся к вышеописанному методу разложения умножения в ряд побитовых сдвигов влево и, рассуждая аналогично, разложим деление в ряд побитовых сдвигов вправо.

Рассмотрим случай, когда делитель известен, а делимое $x \in R$. Для примера возьмем частное $\frac{x}{3}$. Тогда

$$\frac{x}{3} = \frac{x}{2} + k, \tag{1}$$

где k есть разность результата первого сдвига $\left(\frac{x}{2}\right)$ из искомого отношения $\left(\frac{x}{3}\right)$:

$$k = \frac{x}{3} - \frac{x}{2} = -\frac{x}{6}.$$

Уравнение (1) примет следующий вид:

$$\frac{x}{3} = \frac{x}{2} - \frac{x}{6}. \tag{2}$$

Выразим $\frac{x}{6}$ через следующий сдвиг вправо, разделив исходное значение x на четыре:

$$\frac{x}{6} = \frac{x}{4} + k_1.$$

Отсюда

$$k_1 = -\frac{x}{12}; \quad \frac{x}{6} = \frac{x}{4} - \frac{x}{12}.$$

Тогда уравнение (2) примет следующий вид:

$$\frac{x}{3}=\frac{x}{2}-\frac{x}{4}+\frac{x}{12}. \tag{3}$$

Представим $\frac{x}{12}$ как разность $\frac{x}{8}-\frac{x}{24}$.

Тогда уравнение (3) принимает вид:

$$\frac{x}{2}-\frac{x}{4}+\frac{x}{8}-\frac{x}{24}\approx\frac{x}{3}. \tag{4}$$

Выполняя эту процедуру бесконечное число раз, получаем разложение частного $\frac{x}{3}$ в геометрический знакопеременный ряд:

$$\frac{x}{3}=\sum_{n=1}^{\infty}\left((-1)^{n+1}\cdot x\cdot\left(\frac{1}{2}\right)^{n}\right)\frac{x}{12}. \tag{5}$$

Ряд сходится, то есть абсолютная величина следующего члена ряда меньше абсолютной величины предыдущего. Поэтому разность абсолютных величин соседних членов ряда всегда положительна. Это крайне важно с точки зрения оценки конечного алгоритма (скорость вычисления, простота реализации), так как постоянный контроль знака, имеющий место в иных алгоритмах [1] двоичного деления, не только понижает скорость вычисления, но также доставляет значительные неудобства в реализации самих методов.

Как правило, микросхемы ограничены единовременным числом возможных разрядов обрабатываемых данных. Пусть делимое x – шестиразрядное двоичное значение с фиксированной запятой (бит 0), тогда x_{max} = $11111{,}1_2$.

Значит, для заданного шестиразрядного значения x целесообразно применить лишь пять сдвигов вправо. При пятом сдвиге исходное значение будет равным $00000{,}1_2$, при шестом и далее уже $00000{,}0_2$, то есть дальнейшие сдвиги не влияют на общий результат.

В итоге ряд обработки шестиразрядного значения с фиксированной запятой – бит 0, с точностью до одного знака после запятой (в двоичной системе) примет следующий вид:

$$\frac{x}{2}-\frac{x}{4}+\frac{x}{8}-\frac{x}{16}+\frac{x}{32}\approx\frac{x}{3}.$$

Отметим важную деталь: в конце каждого уравнения (2), (3), (4) при формировании очередного сдвига $\left(\frac{x}{2},\ \frac{x}{4},\ \frac{x}{8}\right)$ соответственно присутствует еще по одному значению $\left(\frac{x}{6},\ \frac{x}{12},\ \frac{x}{24}\right)$ – это погрешности. Так как ряд сходится, погрешность с каждым очередным сдвигом вправо будет убывать, а точность вычисления, соответственно, расти и выражаться в количестве дополнительных знаков после запятой (в двоичной системе счисления).

Поэтому при необходимости более точного результата достаточно добавить нужное число разрядов в дробную часть значения. Это увеличит число сдвигов и тем самым увеличит число элементов ряда, уменьшая влияние погрешности на конечный результат.

Зная делитель, всегда можно разложить операцию «деление» в ряд соответствующих сдвигов разрядной сетки делимого. Этот ряд можно получить аналогично ряду (6) для частного $\frac{x}{3}$.

Пример. Пусть делимое задано пятиразрядным двоичным значением x с фиксированной запятой (бит – 0), делитель равен 5. Требуется найти частное $\left(\frac{x}{5}\right)$ с точностью до четырех знаков после запятой.

$$\frac{x}{2}+k_1=\frac{x}{5},\text{ где } k_1=\frac{x}{5}-\frac{x}{2}=\frac{2x-5x}{10}=-\frac{3x}{10}.$$

Тогда

$$\frac{x}{5}=\frac{x}{2}-\frac{3x}{10}.$$

Здесь последнее слагаемое $\frac{3x}{10}$ – текущая погрешность.

Разложим $\frac{3x}{10}=\frac{x}{4}+k_2$:

$$k_2=\frac{12x-10x}{40}=\frac{x}{20},\ \frac{3x}{10}=\frac{x}{4}+\frac{x}{20}.$$

Тогда $\frac{x}{5}=\frac{x}{2}-\frac{x}{4}-\frac{x}{20}$, где $\frac{x}{20}$ – погрешность (уменьшилась).

Разложим $\frac{x}{20}$ на $\frac{x}{8}+k_3=\frac{x}{20}$:

$$k_3=\frac{8x-20x}{160}=-\frac{3x}{40}, \quad \frac{x}{8}-\frac{3x}{40}=\frac{x}{20}.$$

Ряд примет следующий вид:

$$\frac{x}{2}-\frac{x}{4}-\frac{x}{8}+\frac{3x}{40}=\frac{x}{5}, \text{ где } \frac{3x}{40} \text{ – погрешность (уменьшилась).}$$

Продолжая череду сдвигов, получим ряд, ограниченный семью сдвигами (до $\frac{x}{128}$). Оптимальное количество необходимых сдвигов равно пяти разрядам входного x **плюс** три дополнительных разряда дробной части (точность вычисления) **минус** один:

$$\frac{x}{2}-\frac{x}{4}-\frac{x}{8}+\frac{x}{16}+\frac{x}{32}-\frac{x}{64}-\frac{x}{128}\approx\frac{x}{5}.$$

Относительная погрешность ряда равна $\frac{7x}{640}$. Абсолютная погрешность будет зависеть уже от самого x. При произведении дальнейших расчетов в двоичной системе погрешность не повлияет на результат, так как будет всегда меньше значения последнего элемента ряда.

Так же как и при делении на три, здесь прослеживается зависимость в чередовании знаков от делителя, отличающаяся лишь периодом.

Пример. Пусть x равен $1011{,}1_2$. Тогда эталон для проверки результата будет иметь следующее двоичное значение (двойная точность):

$$\frac{11{,}5_{10}}{5_{10}}=2{,}3_{10}=0010{,}01001100_2.$$

Решение. Для начала заполним таблицу 1, переведем элементы ряда со знаком минус в дополнительный код.

Таблица 1

Сдвиг	Значение	Доп. код
X	1011,1000	
***x*/2**	0101,1100	
***x*/4**	0010,1110	1101,0010
***x*/8**	0001,0111	1110,1001
***x*/16**	0000,1011	
***x*/32**	0000,0101	
***x*/64**	0000,0010	1111,1110
***x*/128**	0000,0001	1111,1111

Просуммируем:

$$\begin{array}{rll}
 & 0101{,}1100 & (x/2)\\
+ & \underline{1101{,}0010} & (x/4)\\
 & 0010{,}1110 & \\
+ & \underline{1110{,}1001} & (x/8)\\
 & 0001{,}0111 & \\
+ & \underline{0000{,}1011} & (x/16)\\
 & 0010{,}0010 & \\
+ & \underline{0000{,}0101} & (x/32)\\
 & 0010{,}0111 & \\
+ & \underline{1111{,}1110} & (x/64)\\
 & 0010{,}0101 & \\
+ & \underline{1111{,}1111} & (x/128)\\
 & 0010{,}0100 &
\end{array}$$

Полученный ответ ($0010{,}0100_2$) абсолютно точно повторяет эталон ($0010{,}01001100_2$) вплоть до четырех знаков после запятой (с точностью, заявленной в условии задачи).

Такой тесной связи делителя и делимого, как в классических реализациях деления, уже не прослеживается. Деление состоит лишь из ряда сдвиговых операций, результаты которых суммируются. Делитель влияет только на чередование знаков членов ряда, от самой операции деления полностью абстрагирован. Число разрядов делимого плюс число дополнительных разрядов дробной части (точность вычисления) указывает на оптимальное количество итераций сдвигов, однако данной информацией можно пренебречь и продолжить ряд сдвигов с заведомо большим количеством элементов, чем необходимо. На окончательный результат это не повлияет. То есть если оптимальное количество сдвигов равняется пяти, а было произведено сто сдвигов, результат не изменится.

Выходит, что операцию деления различных чисел можно представить рядами, состоящими из одних и тех же членов $\left(\frac{x}{2} \pm \frac{x}{4} \pm \frac{x}{8} \pm \frac{x}{16} \ldots\right)$, где каждый отдельный знак члена будет регулироваться делителем. А так как при выборе знака очередного члена ряда прослеживается некоторая систематичность («эффект волны»), можно предположить, что знание значения делителя и вывод по этому значению ряда будет лишь частным случаем общей формы любого деления любых двух действительных чисел (конечно, при условии, что делитель не равен нулю).

Попробуем выразить общую форму для деления действительных чисел.

Пусть делимое будет x, а делитель y. Тогда путем первого сдвига $\frac{x}{2}$ деление x на y можно трактовать следующим образом: $\frac{x}{y} = \frac{x}{2} + k_1$, где $k_1 = \frac{x}{y} - \frac{x}{2} = \frac{2x - yx}{2y}$.

Очевидно, что знак k_1 зависит не от всего полученного выражения, а лишь от множителя $(2 - y)$: если $y > 2$, k_1 – отрицательное число; если $y < 2$, k_1 – число положительное; если $y = 2$, $k_1 = 0$, необходимое число членов в ряде, представляющем деление, уже достигнуто. Дальнейшее продолжение ряда не имеет смысла.

Все последующие сдвиги $\frac{x}{2^n}$ будут выражаться как разность погрешностей ($k_n - k_{(n-1)}$). Значит, очередной сдвиг будет отрицательным при $ay \geq b$ и положительным при $ay < b$, где b равно общему знаменателю ряда – последнему элементу последовательности $\{2^n\}, n \in N$.

Например, для ряда $\frac{x}{2} + \frac{x}{4} - \frac{x}{8}$, $b = 8$.

Параметр a будет равным коэффициенту при переменной x.

Например, для ряда $\frac{x}{2} + \frac{x}{4} - \frac{x}{8}$, $\frac{4+2-1}{8} = \frac{5}{8}$ $a = 5$. Выражение ay в данном случае можно записать следующим образом: $ay = 2(2y + y) - y = 5y$.

Обобщенная формула разложения операции «деление» в ряд сдвигов примет вид:

$$\frac{x}{y} = \sum_{n=1}^{\infty} \left(p_{(n-1)} \cdot x \cdot \left(\frac{1}{2}\right)^n \right), \quad \text{где} \begin{cases} m_1 = y, p_0 = 1; \\ m_n = 2m_{(n-1)} + p_{(n-1)}y; \\ p_n = 1 \; при, \; m_n < 2^n; \\ p_n = (-1) \; при, \; m_n \geq 2^n, \end{cases} \tag{6}$$

Формула трактуется следующим образом: параллельно циклическому делению x на два y, наоборот, умножается на два плюс еще один y, если очередной член ряда положительный; либо минус y, если очередной член отрицательный. Полученный параметр y сверяется с текущим значением сдвига x, равным 2^n. Если y больше либо равен 2^n, то следующий член основного ряда отрицательный, если меньше – положительный.

Продемонстрируем сказанное выше на *примере*.

Первое условие всех классических алгоритмов двоичного деления гласит, что делитель не должен быть больше делимого, иначе произойдет переполнение разрядной сетки. Чтобы обойти эту проблему, нужно видоизменить делимое или делитель и привести разрядные сетки к общему виду.

Проигнорируем это ограничение и попробуем разделить 101_2 на 1001_2 с точностью до четырех знаков после запятой. Для большей наглядности все предварительные расчеты выполним в таблице 2.

Таблица 2

n	Сдвиг $x/2^n$	Значение	Дополнительный код	Равенство y и 2^n	Расчет y	Значение	(2^n)
–	x	101,0000			y	001001	–
1	$x/2$	010,1000		>	y	001001	0000010
2	$x/4$	001,0100	110,1100	>	$2y - y = y$	001001	0000100
3	$x/8$	000,1010	111,0110	>	$2y - y = y$	001001	0001000
4	$x/16$	000,0101	111,1011	<	$2y - y = y$	001001	0010000
5	$x/32$	000,0010		<	$2y + y = 3y$	011011	0100000
6	$x/64$	000,0001		<	$6y + y = 7y$	111111	1000000

Получили следующий ряд:

$$\frac{x}{2} - \frac{x}{4} - \frac{x}{8} - \frac{x}{16} + \frac{x}{32} + \frac{x}{64} \approx \frac{x}{9}\,.$$

Суммируем:

$$\begin{array}{rll}
 & 010{,}1000 & (x/2) \\
+ & \underline{110{,}1100} & (x/4) \\
 & 001{,}0100 & \\
+ & \underline{111{,}0110} & (x/8) \\
 & 000{,}1010 & \\
+ & \underline{111{,}1011} & (x/16) \\
 & 000{,}0101 & \\
+ & \underline{000{,}0010} & (x/32) \\
 & 000{,}0111 & \\
+ & \underline{000{,}0001} & (x/64) \\
 & 000{,}1000 &
\end{array}$$

Проверяем: $\dfrac{5_{10}}{9_{10}} \approx 0{,}556_{10} = 0{,}1000_2\,.$

Был получен быстрый и абсолютно точный ответ.

В качестве дополнения в таблице 3 приведены некоторые ряды, полученные из обобщенной формулы разложения (6) для первых 16 сдвигов.

Таблица 3

y	$\frac{x}{2^1}$	$\frac{x}{2^2}$	$\frac{x}{2^3}$	$\frac{x}{2^4}$	$\frac{x}{2^5}$	$\frac{x}{2^6}$	$\frac{x}{2^7}$	$\frac{x}{2^8}$	$\frac{x}{2^9}$	$\frac{x}{2^{10}}$	$\frac{x}{2^{11}}$	$\frac{x}{2^{12}}$	$\frac{x}{2^{13}}$	$\frac{x}{2^{14}}$	$\frac{x}{2^{15}}$	$\frac{x}{2^{16}}$
3	+	–	+	–	+	–	+	–	+	–	+	–	+	–	+	–
5	+	–	–	+	+	–	–	+	+	–	–	+	+	–	–	+
6	+	–	–	+	–	+	–	+	–	+	–	+	–	+	–	+
7	+	–	–	+	–	–	+	–	–	+	–	–	+	–	–	+
9	+	–	–	–	+	+	+	–	–	–	+	+	+	–	–	–
10	+	–	–	–	+	+	–	–	+	+	–	–	+	+	–	–
11	+	–	–	–	+	–	+	+	+	–	+	–	–	–	+	–
12	+	–	–	–	+	–	+	–	+	–	+	–	+	–	+	–
13	+	–	–	–	+	–	–	+	+	+	–	+	+	–	–	–
14	+	–	–	–	+	–	–	+	–	–	+	–	–	+	–	–

Таким образом, получен быстрый, простой в применении, компактный (в отличие от имеющихся аналогов) метод реализации арифметической операции «деление» путем разложения последней в ряд побитовых сдвигов.

Связь делителя и делимого стала условной, так как деление сведено к ряду сдвиговых операций, результаты которых суммируются. Делитель влияет лишь на чередование знаков членов ряда, от самой операции деления абстрагирован. Любые изменения значения делимого не влияют на основной каркас ряда. Количество возможных итераций сдвигов не ограничено. Разрядность делимого плюс количество разрядов дробной части конечного результата (заявленная точность) минус один – формула оптимального количества итераций сдвигов.

Достигнута высокая точность результата. Каждый добавочный разряд в дробной части конечного результата увеличивает минимальное количество необходимых сдвигов, тем самым увеличивая ряд, что ведет к уменьшению погрешности.

Знак конечного результата деления целесообразно проверять лишь после получения самого результата деления. Поэтому в методе используются абсолютные величины делителя и делимого.

ЛИТЕРАТУРА

1. Луцик, Ю.А. Арифметические и логические основы вычислительной техники: учеб. пособие для студ. спец. «Вычислительные машины, системы и сети» всех форм обуч. / Ю.А. Луцик, И.В. Лукъянов. – Минск: БГУИР, Минск, 2004. – 120 с.

Поступила 28.08.2012

**DECOMPOSITION OF ARITHMETIC OPERATIONS “DIVISION”
IN THE RANGE OF BITWISE SHIFTS**

O. SUKHORUKOV

To simplify the implementation of an arithmetic operation “division” when designing an arithmetic logical device (ALD) there was developed a method of decomposition of this operation into a range of bitwise shifts. High speed of calculation, the given accuracy of the result, reliability and stability – these are the advantages of the proposed by the author method, achieved by the use of two key operations “shift” and “summation”. Bitwise shift the right (left) does not require the use of logical elements (AND, OR, NOT), and thus is one of the most simple and fast procedures. Ease of implementation of the method in the design process is achieved by using a “shift” in conjunction with the basic operation of the majority ALD – “summation”.