УДК 681.3

Ю.Я. ЛЕДЯНКИН*

ОПЕРАЦИЯ ДЕЛЕНИЯ ДЛЯ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ. Ч. 2

*МП «ПАК» Института кибернетики имени В.М. Глушкова АН УССР, Киев, Украина

Анотація. *Пропонується цифровий метод виконання арифметичної операції ділення (ОД). Алгоритм забезпечує розпаралелювання обчислювального процесу, прискорення його і підвищення точності. Він заснований на виконанні етапів операцій в алгебрі матриць і реалізації методів цифрової арифметики в булевій (можливо, у багатозначній) алгебрі. Розглянуто варіанти виконання операції та приклади рішення. Передбачає розвиток аналогічних робіт у плані створення і використання комбінованих алгоритмів обробки даних шляхом застосування інших алгебр у поєднанні з методами цифрової арифметики.*
Ключові слова: *цифровий метод, операція ділення, паралельні обчислення, обчислювальні системи.*

Аннотация. *Предлагается цифровой метод выполнения арифметической операции деления (ОД). Алгоритм обеспечивает распараллеливание вычислительного процесса, ускорение его и повышение точности ОД. Он основан на выполнении этапов операций в алгебре матриц и реализации методов цифровой арифметики в булевой (возможно, многозначной) алгебре. Рассмотрены варианты выполнения ОД и примеры решения. Предполагается развитие аналогичных работ в плане создания и использования комбинированных алгоритмов обработки данных путём применения других алгебр в сочетании с методами цифровой арифметики.*
Ключевые слова: *цифровой метод, операция деления, параллельные вычисления, вычислительные системы.*

Abstract. *It is proposed the digital method of performing arithmetic division operations (OD). Algorithm provides the parallelizing of computational process, its acceleration and improved accuracy. It is based on carrying out the steps of operations in the matrix algebra and realization of digital arithmetic in Boolean (possibly multi-valued) algebra. The variants of the execution of operation and examples of solutions are regarded. It is suggested the development of similar projects for the creation and using combined data processing algorithms by applying other algebras in combination with the digital arithmetic methods.*
Keywords: *digital method, the division operation, parallel computing, computing systems.*

1. Введение

Создание гетерогенных вычислительных систем (ГВС) высокой производительности, параллельных структур (ПС) особенно для обработки информации, представленной в виде сложных структур данных (ССД), требует нового подхода к численным и арифметическим методам, закладываемым в структуры спецпроцессора (СП) и всей вычислительной системы (ВС) [1–3]. Это особенно важно при реализации вычислительных алгоритмов в едином технологическом (вычислительном) потоке (ЕТП) с помощью процессорных элементов (ПЭ) со скалярным умножителем (СУ) в своём составе.

Один из подходов к трансформации численных методов для реализации алгоритмов решения задач математической физики (МФ) с помощью метода конечных элементов (МКЭ) предложен и описан в (Ч.1, [1, 4–7]). Такой подход предполагает обработку ССД и решение систем линейных алгебраических уравнений (СЛАУ) в режиме ЕТП. Он обеспечивает переход к построению высокопроизводительных архитектур гетерогенного типа на базе ПЭ с СУ в своем составе с HOST-процессором в виде центральной интеллектуальной машины (ЦИМ).

Операция деления (ОД), которая часто является неотъемлемой частью алгоритма вычислительного эксперимента (ВЭ), записанного в аналитическом виде, является и ча-

ISSN 1028-9763. Математичні машини і системи, 2014, № 2

стью набора арифметических операций, которые впоследствии необходимо реализовывать с помощью процессора. Но ОД в цифровом виде является менее точной и выполняется за большее (по отношению ко времени выполнения операции умножения) в несколько раз время. Поэтому её реализацию в АУ процессора стараются избегать, а на уровне алгоритма её выполнение часто закладывают при помощи стандартных подпрограмм, основанных на операциях сложения, вычитания и умножения (или на спецпроцессор, который реализует ОД). Для параллельных устройств ограничения связаны и с последовательным выполнением ОД, когда каждая новая цифра частного, как и новый остаток, не могут быть вычислены раньше, чем будет получен и проанализирован предыдущий остаток.

Статья состоит из двух частей.

В первой части статьи излагается цель работы, суть предлагаемого метода выполнения ОД и на конкретных примерах показывается возможность её выполнения, опираясь на математический аппарат матричной алгебры.

Во второй части статьи показана возможность (и целесообразность) перехода от применения на первом этапе аппарата матричной алгебры к использованию математики числовой арифметики на втором этапе формирования обратной величины (ОВ) кода числа. Показаны варианты сочетания методов матричной алгебры и числовой арифметики на разных этапах формирования обратного значения от делимого. Доказывается правомочность использования математического аппарата числовой арифметики в сочетании с методами математического аппарата матричной алгебры. На конкретных примерах предложены и рассмотрены арифметические методы, обеспечивающие на втором этапе выполнения ОВ её реализацию, показываются преимущества, которые даёт предлагаемый метод, и возможность распараллеливания вычислительного процесса по выполнению ОВ.

Цель работы

В части 1 работы предложен и описан метод и алгоритм реализации ОД. Он основан на реализации арифметических операций сложения, вычитания и умножения. Применение метода в СП исключит деление в классическом понимании этой операции, заменив на операцию умножения. Возможность вычисления ОВ в параллельном режиме работы СП с повышенной точностью с помощью СУ, используемых в ПЭ, в режиме ЕТП, особенно важна при решении СЛАУ (и др. задач вычислительной математики). Практически при решении СЛАУ такой подход обеспечит замену решения исходной матрицы с помощью выражения

$$\vec{a}_i \leftarrow \frac{a_{ij} * \vec{a}_i - a_{ij} * \vec{a}_j}{a_{j-1,j-1}} \tag{1}$$

на реализацию выражения

$$\vec{a}_i \leftarrow (a_{jj} * \vec{a}_i - a_{ij} * \vec{a}_j) * a^{-1}_{j-1,j-1}. \tag{2}$$

В выражении (2) коэффициент $a^{-1}_{j-1,j-1}$ вычисляется от значения диагонального коэффициента предыдущей $a_{j-1,j-1}$, ранее вычисленной строки факторизуемой матрицы (при условии, что $a_{11} \equiv 1$):

$$L_{j-1,j-1} = 1 / a_{j-1,j-1} = a^{-1}_{j-1,j-1}. \tag{3}$$

Вычислив $a^{-1}_{j-1,j-1}$ от известной (и не равной нулю) величины на прямом ходе на структуре, предназначенной для решения СЛАУ в ЕТП, можно полностью “покрыть“ решение класса задач в ЕТП на параллельном СП из ПЭ с СУ в своем составе.

Подход, связанный с вычислением $a^{-1}_{j-1,j-1}$, не исключает использование HOST-процессора, функции которого выполняет центральная интеллектуальная машина (ЦИМ), работающая в архитектуре ГВС в [3] параллельно с СП.

Постановка задач

Предложить такую (в математическом плане) реализацию метода замены ОД операцией умножения на обратную величину значения кода делителя (предложенного автором и описанного в ч. 1 статьи), которая легко бы реализовывалась структурно с помощью цифровых средств, которые закладываются в ПЭ (с СУ в своем составе), и выполнялась с помощью арифметических операций типа сложения, вычитания и умножения.

2. Решение задачи

Варианты вычисления ОВ. Упрощение вычисления ОВ кода числа

Для числа, записанного с помощью РМП в виде верхней треугольной матрицы, для его хранения можно использовать не ПЭ, а отдельные биты, записанные в запоминающем устройстве (ЗУ) ВС. При такой организации хранения операнда (делителя) требуется всего $n^2/2$ бит памяти.

Если использовать отдельное ЗУ (в составе СП), количество битов памяти можно сократить за счет преобразования организации записи и считывания информации из ЗУ. Для этого входные шины записи диагональных (для каждой диагонали отдельно) битов матричного ЗУ следует объединить. Тогда запись кода числа можно выполнить одновременно во всю матрицу. А шины данных (чтения) по каждой строке можно объединить и подключить к регистру формирования обратного кода числа. Это обеспечит одновременное чтение всех коэффициентов (разрядов) при формировании текущего вектора-столбца.

Такая организация хранения операнда позволит варьировать различными способами распараллеливания вычислительного процесса обращения числа.

Но, тем не менее, количество элементов памяти (битов) для хранения операнда (делителя) при записи его с помощью РМП в виде верхней треугольной матрицы избыточно.

Векторный метод вычисления ОВ кода числа

Предложенный ранее вариант вычисления ОВ кода числа, когда используется матрица $[C]$ с записанным в ней с помощью РМП числом и один вектор-столбец единичной матрицы, также избыточен. Для вычисления ОВ кода числа достаточно взять из матрицы $[C]$ только одну (c_{1j}) первую (верхнюю) строку, записанную в обратном порядке (слева – младший $c_{nn} * 2^{-n}$ разряд числа C, а справа – старший $c_{1n} * 2^{-1}$ разряд), например, в регистр сдвига влево (в сторону младшего $L * c_i * 2^1$, $i = \overline{1,n}$ разряда кода числа C). Таким способом расход оборудования можно сократить, а вычислительный процесс упростить.
Рассмотрим алгоритм на примере обращения числа $C = \{.1 \quad 1 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\} \rightarrow$
$\rightarrow \left| 0 * 2^{(-n)=10} \quad 0 * 2^{-9} \quad 0 * 2^{-i-1} \quad 0 * 2^{-i=7} \quad 0 * 2^{-i+1} \quad 0 * 2^{-5} \quad 1 * 2^{-4} \quad 0 * 2^{-3} \quad 1 * 2^{-2} \quad 1 * 2^{-1} \right|$.
Ему для наглядности описания вычислительного процесса поставим в соответствие матрицу $[C^+]$ и вектор-столбец, записанный в виде строки, вида

$$[C^+] = [0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 1 \quad 1]^T. \qquad (4)$$

В целом работает алгоритм, описанный уравнениями (26а) – (26к) в ч. 1 статьи. В описываемом ниже алгоритме реализуется операция

$$[C^+]_i = L* [C^+]_{i+1}\ (c_i = L* c_{i+1} * 2^1),\ \ i = \overline{1,n}, \tag{4a}$$

где $L*$ – оператор сдвига влево (в сторону младшего разряда в записи (4)) на один разряд (2^1).

Формально (4) – это один (последний в записи РМП) n-й столбец. Остальные $(n-1)$ младшие столбцы последовательно формируются n-м столбцом путем сдвига разрядов в сторону младшего n-го разряда числа, формируемого с помощью процедуры типа обратного хода факторизации матрицы. Сдвигая код вектора-столбца, формируем ОВ кода числа делителя C:

$$\left[\begin{array}{rcr} c_{i,10} & = & e_{i,10} \\ 0 & | & 0 \\ 0 & | & 0 \\ 0 & | & 0 \\ 0 & | & 0 \\ 0 & | & 0 \\ 1 & | & 0 \\ 0 & | & 0 \\ 1 & | & 0 \\ c_{1,10}=1 & = & e_{1,10}=1 \end{array}\right] \rightarrow \left|\begin{array}{rrcr} c_{i,10} & c_{i,10} * (k_1 = 1)] & \Rightarrow & e_{i,9} \\ 0 & [(0*1)=0] & \Rightarrow & 0 \\ 0 & [(0*1)=0] & \Rightarrow & 0 \\ 0 & [(0*1)=0] & \Rightarrow & 0 \\ 0 & [(0*1)=0] & \Rightarrow & 0 \\ 1 & [(0*1)=0] & \Rightarrow & 0 \\ 0 & [(1*1)=1] & \Rightarrow & -1 \\ 1 & [(0*1)=0] & \Rightarrow & 0 \\ 1 & [(1*1)=1] & \Rightarrow & -1 \\ & (k_1=1) & \Rightarrow & 1 \end{array}\right| \rightarrow$$

$$\rightarrow \left[\begin{array}{rcr} c_{i,9} & \Rightarrow & e_{i,9} \\ 0 & \Rightarrow & 0 \\ 0 & \Rightarrow & 0 \\ 0 & \Rightarrow & 0 \\ 0 & \Rightarrow & 0 \\ 1 & \Rightarrow & 0 \\ 0 & \Rightarrow & -1 \\ 1 & \Rightarrow & 0 \\ 1 & \Rightarrow & -1 \\ & \Rightarrow & 1 \end{array}\right| \left|\begin{array}{l} e_{10} * 2^{-10} \\ e_9 * 2^{-9} \\ e_8 * 2^{-8} \\ e_7 * 2^{-7} \\ e_6 * 2^{-6} \\ e_5 * 2^{-5} \\ e_4 * 2^{-4} \\ e_3 * 2^{-3} \\ e_2 * 2^{-2} \\ e_1 * 2^{-1} \end{array}\right| \rightarrow \tag{5}$$

$$[c_{i,9} = L* c_{i,10} * 2^1],\ e_{i,9} = (e_{i,10} - c_{i,10}(e_{\ 1} = (k_1 = 1)),\ i = \overline{2,10},\ k_1 = e_1 = 1,$$

$$c_{i,10} * (k_1 = 1)),$$

$$\rightarrow \left[\begin{array}{cc|c|c} c_{i,7} & c_{i,8}*(k_2=-1) & (e_{i,9}-c_{i,8})\Rightarrow & e_{i,8} \\ 0 & [(0*(-1))=0] & (0-0)=0\Rightarrow & 0 \\ 0 & [(0*(-1))=0] & (0-0)=0\Rightarrow & 0 \\ 0 & [(0*(-1))=0] & (0-0)=0\Rightarrow & 0 \\ 1 & [(0*(-1))=0] & (0-0)=0\Rightarrow & 0 \\ 0 & [(1*(-1))=-1] & (0-(-1))=1\Rightarrow & 1 \\ 1 & [(0*(-1))=0] & (-1)-0=-1\Rightarrow & -1 \\ 1 & [(1*(-1))=-1] & (0-(-1))=1\Rightarrow & 1 \\ & k_2=e_{2,8}=-1 & \Rightarrow & -1 \\ & & & 1 \end{array}\right] \left|\begin{array}{l} e_{10}*2^{-10} \\ e_9*2^{-9} \\ e_8*2^{-8} \\ e_7*2^{-7} \\ e_6*2^{-6} \\ e_5*2^{-5} \\ e_4*2^{-4} \\ e_3*2^{-3} \\ e_2*2^{-2} \\ e_1*2^{-1} \end{array}\right. \rightarrow$$

$$[c_{1,8}=L*c_{i,9}*2^1]*(k_2=-1),\ e_{i,8}=(e_{i,9}-c_{i,8}(e_2=(k_2=-1)),\ i=\overline{3,10},$$

$$k_2=e_{2,8}=-1,\ c_{i,10}*(k_1=1)),$$

$$\rightarrow \left[\begin{array}{cc|c|c} c_{i,6} & (c_{i,7}*k_3=1) & (e_{i,8}-c_{i,7})\Rightarrow & e_{i,7} \\ 0 & [(0*1)=0] & (0-0)=0\Rightarrow & 0 \\ 0 & [(0*1)=0] & (0-0)=0\Rightarrow & 0 \\ 0 & [(0*1)=0] & (0-0)=0\Rightarrow & 0 \\ 1 & [(0*1)=0] & (0-0)=0\Rightarrow & 0 \\ 0 & [(1*1)=1] & (0-1)=-1\Rightarrow & -1 \\ 1 & [(0*1)=0] & (1-0)=1\Rightarrow & 1 \\ 1 & [(1*1)=1] & (-1-1)=-2\Rightarrow & -2 \\ & (k_3=e_{3,7}=1) & & 1 \\ & & & -1 \\ & & & 1 \end{array}\right] \left|\begin{array}{l} e_{10}*2^{-10} \\ e_9*2^{-9} \\ e_8*2^{-8} \\ e_7*2^{-7} \\ e_6*2^{-6} \\ e_5*2^{-5} \\ e_4*2^{-4} \\ e_3*2^{-3} \\ e_2*2^{-2} \\ e_1*2^{-1} \end{array}\right. \rightarrow$$

$$[c_{1,7}=L*c_{i,8}*2^1]*(k_3=1),\ e_{i,7}=(e_{i,8}-c_{i,7}(e_{\ 3}=(k_3=1)),\ i=\overline{4,10},\ k_3=e_{3,7}=1,$$

$$\rightarrow \left[\begin{array}{cc|c|c} c_{i,5} & c_{i,6}*(k=-2) & (e_{i,7}-c_{i,6})\Rightarrow & e_{i,6} \\ 0 & [(0*(-2)=0] & (0-0)=0\Rightarrow & 0 \\ 0 & [(0*(-2)=0] & (0-0)=0\Rightarrow & 0 \\ 1 & [(0*(-2)=0] & (0-0)=0\Rightarrow & 0 \\ 0 & [1*(-2)=-2] & (0-(-2))=2\Rightarrow & 2 \\ 1 & [(0*(-2)=0] & (-1-0)=-1\Rightarrow & -1 \\ 1 & [1*(-2)=-2] & (1-(-2))\Rightarrow & 3 \\ & (k_4=e_{4,6}=-2) & & -2 \\ & & & 1 \\ & & & -1 \\ & & & 1 \end{array}\right] \left|\begin{array}{l} e_{10}*2^{-10} \\ e_9*2^{-9} \\ e_8*2^{-8} \\ e_7*2^{-7} \\ e_6*2^{-6} \\ e_5*2^{-5} \\ e_4*2^{-4} \\ e_3*2^{-3} \\ e_2*2^{-2} \\ e_1*2^{-1} \end{array}\right. \rightarrow$$

$$[c_{1,6} = L * c_{i,7} * 2^1] * (k_4 = 1), e_{i,6} = (e_{i,7} - c_{i,6}(e_{\ 4} = (k_4 = -2)),\ i = \overline{5,10},\ k_4 = e_{4,6} = -2,$$

$$\rightarrow \left[\begin{array}{cc|c|c} c_{i,4} & (c_{i,5} * k_5 = 3) & (e_{i,6} - c_{i,5}) \Rightarrow & e_{i,5} \\ 0 & [0*3=0] & (0-0=0) \Rightarrow & 0 \\ 1 & [0*3=0] & (0-0=0) \Rightarrow & 0 \\ 0 & [1*3=3] & (0-3)=-3 \Rightarrow & -3 \\ 1 & [0*3=0] & (2-0)=2 \Rightarrow & 2 \\ 1 & [1*3=3] & (-1-3)=-4 \Rightarrow & -4 \\ & (k_5 = e_{5,5} = 3) & & 3 \\ & & & -2 \\ & & & 1 \\ & & & -1 \\ & & & 1 \end{array}\right] \left|\begin{array}{l} e_{10} * 2^{-10} \\ e_9 * 2^{-9} \\ e_8 * 2^{-8} \\ e_7 * 2^{-7} \\ e_6 * 2^{-6} \\ e_5 * 2^{-5} \\ e_4 * 2^{-4} \\ e_3 * 2^{-3} \\ e_2 * 2^{-2} \\ e_1 * 2^{-1} \end{array}\right. \rightarrow$$

$$[c_{1,5} = L * c_{i,6} * 2^1] * (k_5 = e_{5,5} = 3),\ e_{i,5} = (e_{i,6} - c_{i,5}(e_{\ 5} = (k_5 = 3)),\ i = \overline{6,10},$$

$$k_5 = e_{5,5} = 3,$$

$$\rightarrow \left[\begin{array}{cc|c|c} c_{i,3} & (c_{i,4} * (k_6 = -4)) & (e_{i,5} - c_{i,4}) \Rightarrow & e_{i,4} \\ 1 & [0*(-4)=0 & (0-0)=0 \Rightarrow & 0 \\ 0 & [1*(-4)=-4 & (0-(-4))=4 \Rightarrow & 4 \\ 1 & [0*(-4)=0 & (-3-0)=-3 \Rightarrow & -3 \\ 1 & [1*(-4)=-4 & (2-(-4))=6 \Rightarrow & 6 \\ & (k_6 = e_{6,4} = -4) & & -4 \\ & & & 3 \\ & & & -2 \\ & & & 1 \\ & & & -1 \\ & & & 1 \end{array}\right] \left|\begin{array}{l} e_{10} * 2^{-10} \\ e_9 * 2^{-9} \\ e_8 * 2^{-8} \\ e_7 * 2^{-7} \\ e_6 * 2^{-6} \\ e_5 * 2^{-5} \\ e_4 * 2^{-4} \\ e_3 * 2^{-3} \\ e_2 * 2^{-2} \\ e_1 * 2^{-1} \end{array}\right. \rightarrow$$

$$[c_{1,6} = L * c_{i,7} * 2^1] * (k_6 = e_{6,3} = -4),\ e_{i,4} = (e_{i,5} - c_{i,4}(e_{\ 6,4} = (k_6 = -4)),\ i = \overline{7,10},$$

$$(k_6 = e_{6,4} = -4),$$

$$\rightarrow \left[\begin{array}{cc|c|c} c_{i,2} & (c_{i,3} * k_7 = 6) & \left|(e_{i,4} - c_{i,3}) \Rightarrow\right. & e_{i,3} \\ 0 & [1*6=6] & \left|(0-6) = -6 \Rightarrow\right. & -6 \\ 1 & [0*6=0] & \left|(4-0) = 4 \Rightarrow\right. & 4 \\ 1 & [1*6=6] & \left|(-3-6 = -9\right. & -9 \\ & (k_7 = e_{7,3} = 6) & & 6 \\ & & & -4 \\ & & & 3 \\ & & & -2 \\ & & & 1 \\ & & & -1 \\ & & & 1 \end{array}\right] \left|\begin{array}{l} e_{10} * 2^{-10} \\ e_9 * 2^{-9} \\ e_8 * 2^{-8} \\ e_7 * 2^{-7} \\ e_6 * 2^{-6} \\ e_5 * 2^{-5} \\ e_4 * 2^{-4} \\ e_3 * 2^{-3} \\ e_2 * 2^{-2} \\ e_1 * 2^{-1} \end{array}\right. \rightarrow$$

$$[c_{i,7} = L * c_{i,6} * 2^1] * (k_7 = e_{7,3} = 6),\ e_{i,3} = (e_{i,4} - c_{i,3}(e_{\ 7,3} = (k_7 = 6)),\ i = \overline{8,10},$$

$$(k_7 = e_{7,3} = 6),$$

$$\rightarrow \left[\begin{array}{cc|c|c} c_1 & (c_{i,2} * (k_8 = -9)) & \left|(e_{i,3} - c_{i,2}) \Rightarrow\right. & e_{i,2} \\ & & & \\ 0 & [0*(-9)=0 & \left|-6-0=-6\right. & -6 \\ 1 & [1*(-9)=-9] & \left|4-(-9)=13 \Rightarrow\right. & 13 \\ & (k_8 = e_{8,2} = -9) & & -9 \\ & & & 6 \\ & & & -4 \\ & & & 3 \\ & & & -2 \\ & & & 1 \\ & & & -1 \\ & & & 1 \end{array}\right] \left|\begin{array}{l} e_{10} * 2^{-10} \\ e_9 * 2^{-9} \\ e_8 * 2^{-8} \\ e_7 * 2^{-7} \\ e_6 * 2^{-6} \\ e_5 * 2^{-5} \\ e_4 * 2^{-4} \\ e_3 * 2^{-3} \\ e_2 * 2^{-2} \\ e_1 * 2^{-1} \end{array}\right. \rightarrow$$

$$[c_{i,8} = L * c_{i,7} * 2^1] * (k_8 = e_{8,2} = -9),\ e_{i,2} = (e_{i,3} - c_{i,2}(e_{\ 8,2} = (k_8 = -9)),\ i = \overline{9,10},$$

$$(k_8 = e_{8,2} = -9),$$

$$\rightarrow \left[\begin{array}{cc|c|c} c_1 & c_1 * 13 & \left|(e_2 - c_2)\right. & e_1 \\ 1 & [1*(13)=13] & -6-13=-19 & -19 \\ & (k_9 = e_{1,9} = 13) & & 13 \\ & & & -9 \\ & & & 6 \\ & & & -4 \\ & & & 3 \\ & & & -2 \\ & & & 1 \\ & & & -1 \\ & & & 1 \end{array}\right] \left|\begin{array}{l} e_{10} * 2^{-10} \\ e_9 * 2^{-9} \\ e_8 * 2^{-8} \\ e_7 * 2^{-7} \\ e_6 * 2^{-6} \\ e_5 * 2^{-5} \\ e_4 * 2^{-4} \\ e_3 * 2^{-3} \\ e_2 * 2^{-2} \\ e_1 * 2^{-1} \end{array}\right. \rightarrow$$

$$[c_{i,9} = L * c_{i,8} * 2^1] * (k_9 = e_{1,9} = 13),$$

$$\rightarrow [C^+]^{-1} \rightarrow [C]^{-1} = \left\{C^{-1}\right\} \rightarrow$$

$$\left\{1*2^{-1} \quad -1*2^{-2} \quad 1*2^{-3} \quad -2*2^{-4} \quad 3*2^{-5} \quad -4*2^{-6} \quad 6*2^{-7} \quad -9*2^{-8} \quad 13*2^{-9} \quad -19*2^{-10}\right\}.$$

В результате получаем число, которое записано в позиционной системе счисления. Вес каждого i-го разряда равен $c_i * 2^{-i}$. Причем некоторые разряды записаны в виде $c_i = \pm 1$, а частично в виде многозначных чисел. Если в процессе дальнейших вычислений ОВ не использовать многозначные числа непосредственно, то можно выполнить процедуру преобразования многозначных чисел в двухпозиционные числа с помощью алгоритма, предложенного и описанного ниже.

Для числа разрядностью n предложенный выше алгоритм определения ОВ кода числа можно записать:

1. $(c_{1,n} = 1) = (e_{1,n} = 1)$, $e_n = \{0*2^{-n} ... 0*2^{-2} \quad 1*2^{-1}\}$.
2. $c_{i,n} * (e_n = 1)$.
3. $e_{i,n} - c_{i,n-1}$, $i = \overline{2,n}$.
4. $L * c_{i,n-1} * 2^1 = c_{i,n-1}$.
5. $c_{n-1} = e_{n-1}$.
6. $c_{i,n-1} * (k = e_{n-1} = c_{n-1})$.
7. . . . . .
8. $[c_{i,n-2} = L * c_{i,n-3} * 2^1] * (k_{n-2} = e_{n-2,2})$, $e_{i,2} = (e_{i,3} - c_{i,2}(e_{n-2,2} = (k_{n-2})$, $i = \overline{n-1,n}$, $(k_{n-2} = e_{n-2,2})$.
9. $[c_{i,n-1} = L * c_{i,n-2} * 2^1] * (k_{n-1} = e_{1,n-1})$.
10. $[C]^{-1} = \left\{C^{-1}\right\}$.

Анализ реализации примера по приведенному выше алгоритму ОВ кода числа делителя показывает, что поразрядную операцию умножения i-го вектора-столбца на соответствующий коэффициент $c_i * E_i$ с помощью дополнительного устройства можно считать излишней.

Коррекция алгоритма

Умножение кода числа $E_i * 2^{-i}$ на "1" или "0" дает тот же результат. Поэтому достаточно на входе (или на выходе) каждого разряда регистра сумматоров (СУ) устанавливать по шинному формирователю (ШФ), информационные входы каждого из них подключены к выходам результата соответствующего разряда сумматора, а управляющими входами ШФ является выход соответствующего разряда регистра сдвига, в котором формируется обратный код соответствующего числа (делителя). Такая коррекция дает экономию в n устройств (сумматоров НТ) и операций умножения, оставляя алгоритму выполнение операции сдвига вектора-столбца и $(n-i)$ операций сложения (вычитания) в n-разрядном регистре сумматора $(\sum_{I=1}^{N} \text{НТ}_i)$, что необходимо делать всегда.

Вычисление $\{B\}/\{C\}=[B]*[C]^{-1}=[A]$.

Первый вариант – выполнение операции умножения “в лоб” по правилам алгебры матриц, когда каждый разряд формируемого числа C^{-1} в матричной записи $[C]^{-1}$ рассматривается как многозначное число (взятое со своим знаком и весом), на которое умножается множимое B, записанное как n-разрядное число в позиционной записи, двоичном представлении $("0","1"))$ с помощью РМП.

Следует заметить, что предлагаемый подход замены ОД (в классическом понимании операции) умножением делимого на ОВ от делителя превращает некоммутативную операцию деления в коммутативную операцию умножения. Переход к операции умножения и возможность использования правила коммутативности обеспечивает замену операндов местами, что дает дополнительные преимущества предлагаемого метода, то есть умножать ОВ кода числа на код числа делимого.

Рассмотрим реализацию предлагаемого метода выполнения ОВ на примере умножения $\{C_2\}$ на его обратное значение классическим методом.

Перемножим два числа $\{C_2\}$ и $\{C_2\}^{-1}$ удвоенной (до 10) значности путем добавления пяти младших нулевых разрядов (из табл. в ч. 1 статьи):

$$\{C_2\}=\{1\ \ 1\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\ \ 0\},\qquad (6)$$

$$\{C_2\}^{-1}=\{1\ \ -1\ \ 1\ \ -2\ \ 3\ \ -4\ \ 6\ \ -9\ \ 13\ \ -19\}.$$

В рассматриваемом примере умножение младших пяти нулевых разрядов множимого на $\{C_2\}^{-1}$ не имеет смысла, поэтому будем перемножать старшие пять разрядов числа $\{C_2\}=\underline{[1\ \ 1\ \ 0\ \ 1\ \ 0]}$ на младшие пять разрядов множителя $\{C_2\}^{-1}=\{-4\ \ 6\ \ -9\ \ 13\ \ -19\}$:

$$\begin{array}{rrrrrrrrrr}
 & & & & & 1 & 1 & 0 & 1 & 0 \\
 & & & & & * & -4 & 6 & -9 & 13 \quad -19 \\
\hline
 & & & & -19 & -19 & 0 & -19 & 0 & \\
 & & & 13 & 13 & 0 & 13 & 0 & & \\
 & & -9 & -9 & 0 & -9 & 0 & & & \\
 & 6 & 6 & 0 & 6 & 0 & & & & \\
\underline{-4} & \underline{-4} & \underline{0} & \underline{-4} & \underline{0} & & & & & \\
\sum= -4 & 2 & -3 & 0 & 0 & -28 & 13 & -19 & 0 &
\end{array}\qquad (7а)$$

и на старшие пять разрядов $\{1\ \ -1\ \ 1\ \ -2\ \ 3\}$ множителя:

$$\begin{array}{rrrrrrrrrr}
 & & & & 1 & 1 & 0 & 1 & 0 & \\
 & & & * & 1 & -1 & 1 & -2 & 3 & \\
\hline
 & & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \\
\sum= 1 & 0 & 0 & 0 & 0 & 4 & -2 & 3 & 0 &
\end{array}\qquad (7б)$$

В результате имеем

$$\{C_2\} * \{C_2\}^{-1} = -4 \quad 2 \quad -3 \quad 0 \quad 0 \quad -28 \quad 13 \quad -19 \quad 0 \qquad (7в)$$

$$+1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 4 \quad -2 \quad 3 \quad 0$$

$$1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad (-28 \quad 13 \quad -19 \quad 0).$$

Поскольку правило коммутативности умножения в арифметике (в отличие от правил матричной алгебры) допускает перестановку сомножителей, поменяем их местами и умножим $\{C_2\}^{-1} = [1 \quad -1 \quad 1 \quad -2 \quad 3 \quad -4 \quad 6 \quad -9 \quad 13 \quad -19]$ на

$$\{C_2\} = [1 \quad 1 \quad 0 \quad 1 \quad 0]: \{C_2\}^{-1} * \{C_2\} = 1 \quad -1 \quad 1 \quad -2 \quad 3 \quad -4 \quad 6 \quad -9 \quad 13 \quad -19. \qquad (7г)$$

$$* [1 \quad 1 \quad 0 \quad 1 \quad 0]$$

$$0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0$$

$$1 \quad -1 \quad 1 \quad -2 \quad 3 \quad -4 \quad 6 \quad -9 \quad 13 \quad -19$$

$$0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0$$

$$1 \quad -1 \quad 1 \quad -2 \quad 3 \quad -4 \quad 6 \quad -9 \quad 13 \quad -19$$

$$1 \quad -1 \quad 1 \quad -2 \quad 3 \quad -4 \quad 6 \quad -9 \quad 13 \quad -19$$

$$\sum = 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad (-28 \quad 13 \quad -19 \quad 0).$$

Из примеров (7) следует, что операцию умножения можно выполнять заменой местами сомножителей, то есть умножением полученного кода $\{C_2\}^{-1}$ (делитель) на двоичное представление кода числа множимого $[B]$.

Вычисления в (7а), (7б), (7в) и (7г) равны. Процедура умножения упрощается, потому что в СП в каждом из ПЭ$_{1-n}$ поразрядно от 1 до n уже хранятся все n значений разрядов числа, полученных при факторизации матрицы. Но реализация умножения будет отличаться от правил умножения в алгебре матриц. Если в СП для обоих сомножителей применить дополнительный код, то вариант замены сомножителей местами можно отнести к недостаткам. Применение оптимального метода умножения чисел означает, что код множимого всегда должен быть положительным. Иначе его следует вновь преобразовать в дополнительный код. Но в данном случае умножение кода отрицательного числа (множимого) сводится к одноразовому умноженню на "1". Для процедуры накопления (суммирования частичных призведений) это равносильно операции вычитания многозначного кода числа. Алгоритм будет приведен в статье ниже.

Одновременно заметим, что реализация умножения, описанная выше выражениями (7а), (7б), (7в) и (7г), позволяет распараллелить в целом весь вычислительный процесс.

Формирование ОВ кода числа по правилам цифровой арифметики

Для этого в процессе формирования j-го вектора-столбца матрицы $[C]^{-1}$ по правилам алгебры матриц при вычислении соответствующего (i-го) разряда, представляемого многозначным числом, предлагается использовать алгоритм преобразования его в двухпозиционный код (“0” и “1”) по правилам цифровой арифметики. Переносы, возникающие в текущем разряде (отличающемся от “0” и “± 1”), следует передать на третий вход сумматора старшего разряда. При суммировании текущих значений каждого из разрядов формируемого числа переносы (поступающие в старшие разряды) будут просуммированы с учетом их веса (позиции в пределах разрядной сетки). Тогда многозначный код i-го разряда i-го

вектора-столбца матрицы $[C^*]^{-1}$ будет преобразован (распределен и учтен в старших $(i+l)$-х разрядах) с учетом веса.

Замечание

Из примера (5) формирования ОВ кода числа следует, что умножение всех j-х компонент текущего i-го вектора-столбца (c_{ij}) производится на один коэффициент k_i с его знаком ("+k" или "–k"). Это определяет, что знаки переносов из младшего разряда в старший совпадают и операция суммирования кодов чисел выполняется в соответствии с правилами числовой арифметики.

Для кода числа

$$[C^*]^T = [11010] = (26/32 = 0{,}8125) \qquad (8)$$

(для которого обратная величина $[C]^{-1}$ равна $[C]^{-1} = [1\text{-}11\text{-}23] = (11/32 = 0{,}34375)$) в алгебре матриц пример преобразования ОВ кода числа в код цифровой арифметики $[C^*]^{-1} = \underline{[1 \quad -1 \quad 0 \quad 1 \quad 1]}$ может быть описан с помощью алгоритма:

$$[C^*]^{-1} = \underline{[1 \quad -1 \quad 0 \quad 1 \quad 1]} \qquad (11/32 = 286/512) \qquad (9)$$

$$\begin{array}{lll}
 & * \quad * \quad * \quad 1 \quad 1 & 3_{10}^{-5} = (1*2^{-4} \quad 1*2^{-5})_2 \\
 & * \quad * \quad * \quad -1 \quad * & -2_{10}^{-4} = (-1*2^{-3})_2 \\
 & * \quad * \quad * \quad * \quad 1 & 1_{10}^{-3} = 1_2^{-3} \\
 & * \quad * \quad * \quad * \quad -1 & -1_{10}^{-2} = -1_2^{-2} \\
 & * \quad * \quad * \quad * \quad 1 & 1_{10}^{-1} = 1_2^{-1} \\
[C]^{-1} = & \boxed{1 \quad -1 \quad 1 \quad -2 \quad 3} & (11/32 = 286/512),
\end{array}$$

где

– запись в таблице вида 3_{10}^{-5} означает, что цифра 3 в данном разряде записана в десятичном коде двухпозиционной системы и равна коду $(1*2^{-4} \quad 1*2^{-5})_2$, записанному в двоичном коде той же двухпозиционной системы;

– снизу таблицы записан код числа $[1 \quad -1 \quad 1 \quad -2 \quad 3] = 11/32 = 286/512$ в двухпозиционной системе счисления, полученного в алгебре матриц в процессе факторизации числа (в записи РМП);

– сверху таблицы записан код числа $[1 \quad -1 \quad 0 \quad 1 \quad 1] = 11/32 = 286/512$ в двухпозиционной системе счисления, полученного после преобразования его по правилам цифровой арифметики. Их значения равны. Умножение на любое из них в цифровой арифметике даст равнозначный результат. Но по правилам алгебры матриц умножение вектора (каким будет число в виде РМП записи) на них даст разные значения, за которыми проследим ниже.

Пример. Как векторы они не равны, поэтому и произведение их в алгебре матриц даст разный результат. Это существенное различие получаемого результата формирования $[C^*]^{-1}$.

В первом варианте вычисления, выполняя операции в алгебре матриц, мы получаем (вектор-столбец $[C^*]^{-1}$), умножение матрицы $[C]$ на который в результате даёт единичную матрицу ($[E]$). Во втором варианте вычисления получаем число, умножение матрицы $[C]$ на которое в результате НЕ даёт единичную матрицу ($[E]$).

При умножении двух чисел $[C\ ]*\ [C^*]^{-1} = \underline{[1\ \ 1\ \ 0\ \ 1\ \ 0]} * \underline{[1\ \ -1\ \ 0\ \ 1\ \ 1]}$ $= 26/32 * 11/32 = 286/512$ (пример (7а)) по правилам цифровой арифметики необходимо для положительных разрядов множителя складывать ЧП_i (в виде кода множимого, взятого с учетом веса текущего разряда множителя), а для отрицательных разрядов кода множителя – вычитать код множимого.

Пример. Получим сумму $\sum_{1,3-5} \text{ЧП}_i$ от умножения на положительные (и "0") разряды множителя:

$$\begin{array}{rl} & \underline{[1\ \ 1\ \ 0\ \ 1\ \ 0]} \\ & * \underline{1\ \ -1\ \ 0\ \ 1\ \ 1} \\ & \ \ 1\ \ 1\ \ 0\ \ 1\ \ 0 \qquad \text{ЧП}_5 = 1 * 2^{-5} \\ & 1\ \ 1\ \ 0\ \ 1\ \ 0 \qquad\ \ \text{ЧП}_4 = 1 * 2^{-4} \\ & 0\ \ 0\ \ 0\ \ 0\ \ 0 \qquad\ \ \text{ЧП}_3 = 0 * 2^{-3} \\ \underline{1\ \ 1\ \ 0\ \ 1\ \ 0} & \qquad\qquad\ \ \text{ЧП}_1 = 1 * 2^{-1} \\ \sum_{1,3-5} \text{ЧП}_i = 1\ \ 1\ \ 1\ \ 1\ \ 0 & 1\ \ 1\ \ 1\ \ 0. \end{array} \tag{10}$$

Для получения кода результата необходимо умножить на $(-1 * 2^{-2})$ – отрицательный разряд множителя. Для этого из частичного произведения $\sum_{1,3-5} \text{ЧП}_i$ по алгоритму [3] следует вычесть код множимого. С этой целью следует преобразовать в обратный код $\sum_{1,3-5} \text{ЧП}_i$, сложить его с кодом множимого, а полученный псевдорезультат вновь обратить:

$$\sum_{1,3-5} \text{ЧП}_i \rightarrow \sum_{1,3-5} \overline{\text{ЧП}_i} = 0\ \ 0\ \ 0\ \ 0\ \ 1\ \ 0\ \ 0\ \ 0\ \ 1 \tag{10а}$$

$$\begin{array}{l} +\underline{1\ \ 1\ \ 0\ \ 1\ \ 0} \qquad \text{ЧП}_1 = 1 * 2^{-2} \\ \underline{\underline{0\ \ 1\ \ 1\ \ 1\ \ 0\ \ 0\ \ 0\ \ 0\ \ 1}}\ \text{результат в инверсном коде} \end{array}$$

$1\ \ 0\ \ 0\ \ 0\ \ 1\ \ 1\ \ 1\ \ 1\ \ 0\ \ \text{ЧП}_2$ (результат умножения)=

$= [C\ ]^T * [C^*]^{-1} = [C^*]^{-1} = 286/512.$

Выполним формально преобразование ОВ кода:

$$[C^+]^{-1} = \{\underline{1\ \ -1\ \ 0\ \ 0\ \ 1\ \ 1\ \ 1\ \ -1\ \ 0\ \ -1}\} = 307/1024 \tag{11}$$

$$\begin{array}{lll} 1 & & = 1 * 2^{-1} \\ 0\ \ -1 & & = (-1) * 2^{-2} \\ 0\ \ 0\ \ 1 & & = 1 * 2^{-3} \\ 0\ \ 0\ \ -1\ \ * & & = -2 * 2^{-4} \\ 0\ \ 0\ \ 0\ \ 1\ \ 1 & & = 3 * 2^{-5} \\ 0\ \ 0\ \ -1\ \ 0\ \ * & & = (-4) * 2^{-6} \\ 0\ \ 0\ \ 1\ \ 1\ \ * & & = 6 * 2^{-7} \\ 0\ \ -1\ \ 0\ \ 0\ \ -1 & & = (-9) * 2^{-8} \end{array}$$

$$\begin{array}{rrrrrrrrrrrr} & & & & & 0 & 1 & 1 & 0 & 1 & & = 13 * 2^{-9} \\ & & & & & & -1 & 0 & 0 & -1 & -1 & = (-19) * 2^{-10} \end{array}$$

$$[C]^{-1} = \overline{[1 \quad -1 \quad 1 \quad -2 \quad 3 \quad -4 \quad 6 \quad -9 \quad 13 \quad -19]} = 307/1024$$

$$\{1 \ \text{-1} \ \ 1 \ \text{-2} \ \ 3 \ \ \text{-4} \ \ 6 \ \ \text{-9} \ 13\text{-19}\} \rightarrow \{1 \ \text{-1} \ 0 \ 0 \ 1 \ 1 \ 1 \ \text{-1} \ 0 \ \text{-1}\} = 307/1024. \qquad (12)$$

Из рассмотрения примера (11) видно совпадение знаков во всех разрядах в пределах одной строки. В практике выполнения операции умножения это имеет принципиально важное значение. Кроме этого, из матрицы (11) видна возможность распараллеливания вычислительного процесса.

3. Накопление знакопеременных ЧП

В процессе формирования ОВ кода вектора-столбца (по правилам алгебры матриц) и преобразования (10-10а) вектора в число (по правилам цифровой арифметики) при накоплении знакопеременных ЧП, представленных в дополнительном коде (ДК) в аппаратной реализации стандартных (известных) методов могут возникнуть трудности. Это связано с вычитанием кода множимого, когда по алгоритму складывают уменьшаемое $(\sum_i ЧП_i)$ в исходном коде с вычитаемым $ЧП_{i+1}$ в обратном коде с прибавлением "1" в младшем разряде суммируемого числа. Это суммирование вносит неоднозначность выполняемых операций сложение/вычитание. Предложенный выше алгоритм (10) для одноразового вычитания исключает такую неоднозначность операции, делая её, по существу, однотипной с выполнением операции сложения.

Фактически знак "+" или знак “-“ определяет в сумматоре СУ, по каким шинам (прямым или инверсным) поступает код уменьшаемого, а после выполнения операции суммирования код результата будет записан в регистр сумматора (как результат выполнения операции вычитания).

В устройстве с запоминанием полученных на очередном такте значений результата (B) и переноса (E) в каждом разряде сумматора предлагается использовать следующий алгоритм выполнения операций сложение/вычитание.

Операцию сложения следует выполнять по известным правилам, запоминая значения результата (B) и переноса (E) в каждом разряде сумматора в коде представления – исходном ДК (без преобразования его в обратный). На выходе сумматора СУ на двух регистрах (результата RgB и переноса RgE) запоминают два слагаемых от текущей суммы. Полученные ранее коды суммируем с третьим числом на RgC, получая вновь B' и E' на RgB и RgE. Если при умножении за операцией сложения следует выполнение операции вычитания кода множителя (умножение на отрицательное число) из двух слагаемых результата (B) и переноса E (запомненных в каждом разряде сумматора на RgB и RgE и представляющих собой сумму (или разность от предыдущей операции), третьего числа C с RgC), то нельзя формально использовать алгоритм вычитания:

$$A - C = \overline{\overline{A} + C}, \qquad (13)$$

поскольку $A = B + E$ имеем

$$(B + E) - C = \overline{(\overline{B} + \overline{E}) + C} \neq B' + E', \qquad (14)$$

так как из обоих чисел (B) и (E) вычитаем C.

Необходимо выполнять

$$\overline{\overline{B}+C}+E=B^{'}+E^{'}. \qquad (15)$$

Это означает, что при накоплении знакопеременных чисел полученные в сумматоре значения результата $B^{'}$ необходимо передавать на регистр хранения значений кода результата в обратном (при суммировании – в прямом коде), а значения переносов – всегда в прямом коде. И каждую операцию вычитания (первичную или после операции суммирования) выполнять путем сложения значений кода результата в обратном коде с новым значением кода ЧП.

Пример. 7 -7 +5 = 5➔0.0111= A =7
+ 1.1001= B =-7
+ 0.0101= C =5.

	0.0111	$=A$									
-	0.0111	=+7									
	$\overline{\overline{1.1000}}$	$=\overline{A}$									
+	0.0111	$=B$									
	$\overline{\overline{1.1111}}$	$=B^{'}$		$\overline{\overline{1.1111}}$	$=B^{'}$		$\overline{\overline{0.0101}}$	$=R_1=5$			
	0.0000	$=\overline{B^{'}}$		0.0000	$=\overline{B^{'}}$		1.1010	$=\overline{R_1}$			
+	0.000 –	$=E^{''}$	+	0.000 –	$=E^{''}$		+0.0111	$=A=7$			
+	0.0101	$=C$	+	0.0101	$=C$		$\overline{1.1101}$	$=B^{'}$			
				$\overline{0.0101}$	$=R_1=5$		+0.000 –	$=E^{'}$			
							$\overline{\overline{0.0001}}$	$=\overline{R}$			
							1.1110	$=R=-2.$			

4. Параллельное вычисление ОВ

Вычислительный процесс по реализации ОВ может быть распараллелен.

Он может быть заложен в пределах одного j-го, $j=\overline{n,1}$ столбца, когда вычисления выполняются одновременно во всех i-х, $i=\overline{n,1}$ разрядах (битах числа).

Возможны другие пути распараллеливания вычислительного процесса.

Например. “Разбить” n-мерный $(n \gg k)$ вектор-столбец на k блоков (в пределе $k=n$) и вычислять коэффициенты в каждом из блоков одновременно.

Закладывать параллелизм можно и на одновременное вычисление m, $m=\overline{1,n}$ вектор-столбцов. В этом варианте метода начало вычислений в каждом m-ом столбце можно начинать только со сдвигом на один вычислительный такт (не менее), на котором будет вычислен коэффициент (разряд) определяемого числа. Окончание вычислений будет (практически) одновременно, поскольку длина каждого следующего $(j-1)$-й столбца $(j=\overline{n,1})$ на единицу меньше длины предыдущего j-го, а заканчивается на первой строке $(i=1)$. Однако загрузка устройств в таком варианте распараллеливания будет неравномерной, поскольку в первом вычисляемом $(n-\text{м})$ столбце следует вычислить все i-е, $i=\overline{n,1}$ коэффициенты, в то время как в последнем $(j=1)$ вычисляемом столбце следует выполнить вычисления только один раз. Разбивка вычислений на k блоков в пределах каждого j-го столбца может усреднить загрузку вычислительного устройства.

Вариант вычисления обратной величины $[C^+]^{-1}$ методом вычисления вектора-столбца (5) может быть также распараллелен. Для этого следует разбить вектор-столбец длиной $l=\overline{n,1}$ на k блоков (в пределе $k=n$). После вычисления ОВ очередного $(i$-го) значения разряда вектора-столбца вычисления $[C^+]^{-1}$ можно производить во всех блоках формируемого вектора-столбца на разных блоках ПЭ$_k$ одновременно.

Возможны сочетания процессов распараллеливания вычислений на блоки с параллельным вычислением в каждом столбце в пределах одного блока. Количество строк в матрице, записанной с помощью РМП, для одинаковой загрузки блоков можно рассчитать, исходя из того, что "площадь" начального треугольника $k_1=\overline{n,i}$ факторизуемой матрицы равна "площади" трапеций из строк $k_2=i+\overline{m,...,1}\_...$, "следующих" за ним.

5. Выводы

1. Предлагаемый метод выполнения ОВ позволяет:

– исключить ОД в классическом понимании операции деления, сводя её к выполнению операций сложения, вычитания и умножения;

– за счет приведения операции к умножению сделать её коммутативной;

– распараллелить вычислительный процесс как всего алгоритма, так и на уровне ОВ;

– ускорить выполнение ОД;

– повысить точность ОД.

2. Выполнение метода основано на использовании методов алгебры матриц и цифровой арифметики булевой (возможно, многозначной) алгебры.

3. Применение метода:

– замыкает (в плане создания ЕТП) вычислительный процесс с обработкой ССД на параллельных структурах из ПЭ на базе СУ;

– позволит решать СЛАУ методом факторизации без выполнения ОД, которая имеет низкую точность выполнения, не подлежит распараллеливанию на уровне выполнения операции и из всех арифметических операций является наиболее длительлной по времени выполнения;

– позволит повысить точность выполнения предполагаемой ОД и распараллелить её выполнение не только в СП, но и с помощью универсальных ЭВМ;

– позволит использовать методы решения СЛАУ, предложенные в 2, организовывая ЕТП в универсальных ЭВМ ещё до создания СП для задач МФ;

– кроме решения задач МФ, метод ОВ может успешно применяться в задачах поиска последовательных приближений, обращения матриц, вычисления невязок, при исправлении элементов приближенной матрицы и др. случаях решения СЛАУ, задач МФ и др.

4.Следует провести дополнительный анализ предложенных решений выполнения ОВ (и отдельных этапов её), провести исследования в плане комбинирования методов алгебр матриц и цифровой арифметики для других целей.

5. Можно предположить, что изложенные выше алгоритмы, сочетающие (на разных этапах выполнения операции деления) обработку цифровых данных по правилам алгебры матриц и методам цифровой арифметики, могут положить начало новому направлению в создании цифровых устройств параллельного типа с использованием СУ; при обработке сложных структур данных (ССД); в едином технологическом (вычислительном) потоке (ЕТП). Особенно при создании архитектур гетерогенного типа для моделирования вычислительных экспериментов (ВЭ) с включением в свой состав центральной интеллектуальной машины

(ЦИМ), при решении задач не только МФ, но и других научно-технических и экономических (общих) задач.

Заключение

В рамках выделяемого для статьи объёма трудно предложить новый метод и изложить решения (и все нюансы) отдельных этапов метода, который в данном случае можно считать постановочным в плане нового подхода к выполнению цифровой арифметической операции деления. Предполагается, что метод не является панацеей. Но он даст толчок для дальнейшего его продвижения специалистами, работающими в области разработки, адаптации вычислительных методов и создания численных высокопроизводительных параллельных структур. Особенно параллельных архитектур гетерогенного типа, обеспечивающих сверхвысокую производительность при снижении затрат на обработку единицы информации.

Идея выполнения операции деления (как и метода факторизации матрицы [2]) была предложена автором во время работы в МП ПАК «Вычислительные средства, программирование и технология“ Института кибернетики им. В.М. Глушкова АН УССР, организованном в конце 80-х годов прошлого столетия для создания спецпроцессора новой архитектуры и проработана в ТОО «ФИРМА «ПАК»». В силу форс мажорных обстоятельств работы по спецпроцессору были остановлены. По этой причине была задержана и публикация. Настоящая статья подготовлена к передаче в печать в год 90-летия светлой памяти академика В.М. Глушкова – талантливого ученого-кибернетика и алгебраиста, менеджера и человека, которому наука и мы во многом обязаны.

Считаю приятным долгом выразить свою признательность В.П. Клименко, который стимулировал написание и оказал содействие в издании цикла статей (Ч. 1, [2–8]) в направлении работ, поддержанном в своё время В.М. Глушковым.

СПИСОК ЛИТЕРАТУРЫ

1. Ледянкин Ю.Я.Методы получения суммы парных произведений для решения уравнений математической физики / Ю.Я. Ледянкин // Автоматика. – 1979. – № 5. – С. 78 – 82.
2. Ледянкин Ю.Я. Ускоренный метод умножения чисел, представленных в последовательном дополнительном коде / Ю.Я. Ледянкин // Автоматика. – 1989. – № 3. – С. 76 – 82.
3. Боюн В.П. Об одном способе повышения эффективности процессора / В.П. Боюн, Ю.Я. Ледянкин // Технические средства управляющих машин и систем. – Киев: РИО ИК, АН УССР, 1976. – С. 42 – 54.
4. Ledyankin Yu.Ya. Accelerated Method of Multipying Numbers Represented in Sequential Code / Yu.Ya. Ledyankin // Soviet Journal of Automation and Information Sciens. – 1989. – N 3. – P. 88 – 94.
5. Ledyankin Yu.Ya. Methods of Obtaining the Sum of Pairs of Products in Solving the Equations of Mathematical Phisics / Yu.Ya. Ledyankin // Soviet Automatic Control. – 1979. – N 12. – P. 61 – 64.
6. А.с. № 822181 (СССР). Устройство для умножения чисел в дополнительных кодах / Ю.Я. Ледянкин, Б.Н. Малиновский.

Стаття надійшла до редакції 12.12.2013