УДК 004.3:007

МЕТОД ТОЧНОГО ИНФОРМАЦИОННОГО ПОИСКА НА ОСНОВЕ РАЗРЯДНОГО РАСПАРАЛЛЕЛИВАНИЯ

Ромм Я.Е., Белоконова С.С.

Таганрогский институт имени А.П. Чехова (филиал) РГЭУ (РИНХ), Таганрог, e-mail: romm@list.ru, belokonova@mail.ru

В статье представлен метод разрядного распараллеливания операций сравнения, ориентированный на применение в системах информационного поиска. Метод основан на алгоритме алгебраического сложения полноразрядных двоичных чисел, который не использует вычисление переноса. При сравнении применяется аналог дополнительного кода в двоичной системе счисления. Особенности метода позволяют строить его применение одновременно для сравнения чисел и элементов строкового типа. На основе метода могут идентифицироваться совпадающие фрагменты слов с логарифмической оценкой временной сложности. Основная алгоритмическая схема видоизменяется для выполнения параллельного по битам суммирования полноразрядных двоичных полиномов. Полученный сумматор выполняет полное сложение с логарифмической от числа разрядов слагаемых оценкой временной сложности. При этом сумматор состоит из квадратичного от числа разрядов количества логических элементов. Отмечаются возможности применения предложенного метода для ускорения операций поиска, вставки, замены в двоичных и декартовых деревьях структур данных, указаны другие возможные области использования, включая идентификацию бинарных изображений и их фрагментов. Структура предложенных алгоритмов битового распараллеливания делает возможным их применение для повышения точности вычислительных операций за счет удлинения разрядной сетки представления операндов и сокращения потерь значащих цифр в операциях с плавающей точкой.

Ключевые слова: разрядное распараллеливание сложения двоичных чисел, поразрядно-параллельное сравнение слов, параллельная идентификация фрагментов строк, логарифмическая временная сложность сумматора

ACCURATE METHOD OF INFORMATION RETRIEVAL BASED ON BIT PARALLELISM

Romm Ya.E., Belokonova S.S.

Taganrog Institute of A.P. Chekhov (branch) RGEU (RINH), Taganrog, e-mail: romm@list.ru, belokonova@mail.ru

The paper presents a method of bit parallelization of comparison operations focused on the information retrieval systems usage. The method is based on the algebraic addition algorithm for single-digit binary numbers without using transfer computation. When comparing, the equivalent of additional code is used in the binary number system. The features of the method allow to implement it in order to compare numbers and elements of string type simultaneously. On the basis of the method, similar fragments of words with logarithmic estimation of time complexity can be identified. The main algorithmic scheme is modified to perform bit-parallel summation of full-bit binary polynomials. The resulting integrator performs a complete addition to the logarithmic estimation of time complexity on the number of digits of the components. In this case, the integrator consists of a quadratic number of bits of the number of logical elements. The possibilities of the proposed method implementation for speeding up the operations of searching, inserting, replacing data structures in binary and Cartesian trees are noted, other possible uses are indicated, including the identification of binary images and their fragments. The structure of the proposed algorithms of bit parallelization makes it possible to use them to improve the accuracy of computational operations by extending the bit grid of operand representation and reducing the loss of significant numbers in floating-point operations.

Keywords: bit parallel addition n-digit numbers, bit-parallel comparison of words, parallel identification of line fragments, logarithmic time complexity of the integrator

Ускорение, улучшение качества и повышение точности информационного поиска актуально для существующих и проектируемых информационных систем [1, 2]. В качестве алгоритмической основы улучшения поиска в рассматриваемом аспекте остается не раскрытым потенциал битового (разрядного) распараллеливания операций сравнения элементов строкового типа. С целью раскрытия этого потенциала в статье излагается максимально параллельное по разрядам операндов выполнение арифметического сложения чисел с учетом знака операндов. Способ отличается тем, что после элементарного предварительного преобразования операции вычисления переноса в нем становятся взаимно независимыми, отделяются друг от друга, выполняются синхронно по всем разрядам [3, 4], при этом не используются традиционные логические схемы вычисления переноса. Детально излагается распространение этого способа на случай параллельного по битам сравнения чисел и слов строкового типа. В результате представлены основы максимально параллельного выполнения базовых операций поиска, включая сравнение и сортировку. Предлагаемый способ сравнивается

по временной сложности и аппаратным затратам с известными схемами разрядного распараллеливания арифметических операций [5, 6], а также базовых операций в информационных системах, включая операции сравнения, поиска, замены [7–9] с учетом построения структур данных [10, 11]. Как частный случай применения способа дан алгоритм параллельного сложения полноразрядных операндов без вычисления переноса за логарифмическое от числа разрядов время при квадратичном количестве элементов. В [5] декларируется отсутствие прогресса в синтезе сумматоров в течение 40 лет. Ниже исследуется альтернативное построение: в [3, 4] дан алгоритм сложения за время $O(1)$ при абстрактно произвольном количестве разрядов слагаемых.

Преобразование параллельного по разрядам сложения для применения в информационном поиске

Преобразуемый метод сложения и его описание заимствуются из [3, 4]. Сущность метода в том, что после предварительного параллельного по разрядам шага все переносы становятся взаимно отделенными промежуточными нулями. Это дает возможность параллельного по разрядам одновременного преобразования, которое эквивалентно распространению переноса. При наличии параллельных битовых сумматоров время сложения формально имеет единичную оценку независимо от числа разрядов. Если соединить метод с параллельным преобразованием знаков, для этого используется аналог дополнительного кода, то возникает возможность сравнения слов, представленных в двоичном коде, с единичной оценкой времени независимо от длины слова. Для корректного представления преобразований необходимо краткое описание исходного метода. Метод заключается в следующем. Двоичные числа

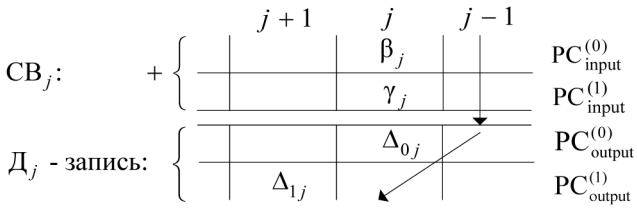
$$P_1 = \sum_{i=0}^{n} \beta_i 2^i, \quad P_2 = \sum_{i=0}^{n} \gamma_i 2^i, \tag{1}$$

где β_i, γ_i – двоичные коэффициенты 0 или 1, располагаются друг под другом согласно весу коэффициентов:

β_n	β_{n-1}	...	β_1	β_0	$\mathrm{PC}^{(0)}_{\mathrm{input}}$
γ_n	γ_{n-1}	...	γ_1	γ_0	$\mathrm{PC}^{(1)}_{\mathrm{input}}$

(2)

Здесь и ниже РС – разрядная сетка. Одновременно, взаимно независимо по индексам (2), выполняется сложение (по вертикали) $\beta_j + \gamma_j$ ($\mathrm{СВ}_j$). Битовая сумма записывается в виде $\beta_j + \gamma_j = \sum_{l=0}^{1} \Delta_{lj} 2^l$, $\Delta_{lj} = \begin{cases} 0 \\ 1 \end{cases}$, $j = 0,1,...,n$, по диагонали от j-го коэффициента ($\mathrm{Д}_j$-запись):



. (3)

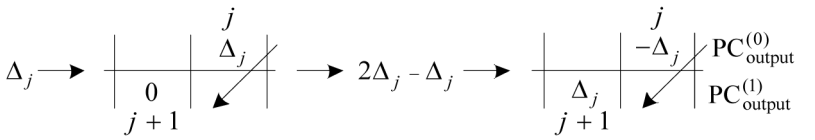
В двухрядном коде суммы слагаемых (1),

Δ_{n+1}	Δ_n	Δ_{n-1}	...	Δ_j	...	Δ_1	Δ_0	$\mathrm{PC}^{(0)}_{\mathrm{output}}$
∇_{n+1}	∇_n	∇_{n-1}	...	∇_j	...	∇_1	∇_0	$\mathrm{PC}^{(1)}_{\mathrm{output}}$

, (4)

все переносы взаимно отделены парами вида $\begin{pmatrix} \Delta_k \\ \nabla_k \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, в $\mathrm{Д}_j$-записи по диагонали от 1 из $\mathrm{PC}^{(0)}_{\mathrm{output}}$ в $\mathrm{PC}^{(1)}_{\mathrm{output}}$ может располагаться только 0 [4]. Оба факта – следствия сложения по вертикали не более двух единиц равного веса. Далее, в (4) каждое $\Delta_j \neq 0$ из $\mathrm{PC}^{(0)}_{\mathrm{output}}$ (и только

из $\mathrm{PC}^{(0)}_{\mathrm{output}}$) представляется в виде тождества $\Delta_j \equiv 2\Delta_j - \Delta_j$, правая часть которого размещается в порядке $Д_j$-записи



(5)

при всех возможных $j = 0,1,..., n$. Корректность преобразования (5) обосновывается отмеченным выше следствием [4]. Параллельное сложение битовых значений равного веса образует в $\mathrm{PC}^{(0)}_{\mathrm{output}}$ окончательную сумму P_1 и P_2 в однорядном коде. Примеры выполнения сложения изложенным способом будут даны ниже. Код суммы окажется знакоразрядным (значения разрядов $\mathrm{PC}^{(0)}_{\mathrm{output}}$ либо 0, либо +1, либо –1). Время сложения T при количестве элементов сумматора R составит

$$T = O(1), \quad R \approx 4(n+1) = O(n). \qquad (6)$$

Чтобы выполнять необходимое для поиска сравнение данных, следует представить и обосновать выполнение вычитания в границах рассматриваемого способа (перейти от арифметического сложения к алгебраическому – с учетом знака слагаемых). Для этого используется обратный код вычитаемого и восстановление правильного результата по аналогии с дополнительным кодом. Пусть от P_1 из (1) вычитается P_2. Для этого P_2 переводится в обратный код. Это равносильно тому, что вместо $P_1 - P_2$ выполняется $P_1 + (2^n + 2^{n-1} + 2^{n-2} + ... + 2^1 + 2^0 - P_2)$. Чтобы конечный результат был правильным, достаточно из него вычесть дополнительное слагаемое $\sum_{i=0}^{n} 2^i$. Поскольку $\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$, из полученного результата достаточно вычесть $2^{n+1} - 1$ или сложить его с $-2^{n+1} + 1$. При этом сложение должно выполняться изложенным согласно (1)–(6) способом. Знак окончательного значения разности дает знак старшего ненулевого разряда. Время выполнения вычитания сохранит оценку (6) с точностью до константы, определяемой двумя дополнительными шагами.

Сравнение элементов (слов) строкового типа

Символы слова представляются в двоичной форме на основе ASCII-кода. В порядке расположения символов в двоичном коде слово интерпретируется как число вида (1). Согласно лексикографическому порядку сравниваемые слова выравниваются по первым слева направо символам. В процессе выравнивания пустые символа меньшего по количеству символов слова заменяются нулями. Остается сравнить два полученных числа изложенным выше способом. Пусть, например, сравниваются слова `'sky'` и `'sun'`. С выравниванием по первым символам получится:

```
01110011011010111011110 01
01110011011000010111011 10
```

В данном расположении из верхнего слова вычитается нижнее. Алгебраическая сумма примет вид

```
000000000000-1101100001-1011
```

В соответствии знаку старшего ненулевого разряда `'sky' < 'sun'`.

В [12] представлены результаты численного моделирования, показывающие, что с помощью изложенного метода могут идентифицироваться строки и текстовые фрагменты произвольного вида. Часть комплекса численного моделирования составляли двоичные полиномы вида (1), при их сравнении и при алгебраическом сложении слова выравниваются по младшему разряду. Численный эксперимент во всех рассмотренных случаях дал положительные результаты. С выравниванием порядков метод переносится на операции с плавающей точкой. Поиск по маске дает совпадение, если сравнение с маской влечет нулевую разность. Таким образом, поиск чисел и слов объединяется в общую алгоритмическую схему. Еще один способ построения единой схемы получается на основе поиска разности `abs(ord() - ord())` для строк и `abs(x - y)` для чисел. В [13] для этой цели используется сортировка слиянием по матрицам сравнений с хранимыми ссылками на элементы входных массивов. Нулевое значение `abs(x - y)`. В этом способе вычисляется приближенно с заданной границей погрешности. Если не учитывать время, требуемое для идентификации знака старшего ненулевого разряда, изложенное сравнение n-разрядных слов имеет временную сложность $T_{string\ com\ parison}(n) \approx 8\,\tau_{binary} = O(1)$, которая формально не зависит от n. Здесь и ниже τ_{binary} – время переключения логического элемента.

Однако идентификация знака сравнения – самостоятельная задача. Ее решение в конечном счете приводит к исключению знаков разрядов на входе и на выходе метода с построением алгоритмической схемы параллельного сумматора.

Алгоритм идентификации знака старшего ненулевого разряда

Знак старшего ненулевого разряда в знакоразрядном двоичном слове можно выделить, в частности, при помощи схем, представленных в [3, 4] и [11]. Ниже для этой цели предлагается алгоритм на основе схемы сдваивания [14]. Сдваивание выполняется для всей последовательности $n+1$ разрядов слова и необходимо включает старший разряд:

Операция $(a_i)\oplus(a_j)$, затем $(a_i^k)\oplus(a_j^k)$ подразумевает выбор и сохранение знака в зависимости от знака сдваиваемых бит. Именно, если $a_i \neq 0$, то узел сдваивания выбирает и сохраняет бит со знаком a_i, если $a_i = 0$ и $a_j \neq 0$, то в узле выбирается и сохраняется a_j, если $a_i = 0$ и $a_j = 0$, то выбирается и сохраняется 0. Алгоритм заканчивает работу на шаге первого соединения в узле сдваивания значения 0 старшего разряда (разряда с индексом n) с битовым значением $a_j^k \neq 0$. После останова алгоритм перемещает $a_j^k \neq 0$ обратно вдоль пути нуля старшего разряда. В результате в разряде с индексом n на входе схемы окажется искомое значение знака старшего ненулевого разряда. Временная сложность алгоритма определяется числом последовательных шагов:

$$T_{sign\ identification}(n) \le 2\lceil \log_2(n+1)\rceil \tau_{binary} = O(\log_2 n). \quad (7)$$

Необходимое дополнение заключается в том, что аналогичным способом можно поразрядно-параллельно выделить все совпадающие фрагменты двух строк. Именно, на первом шаге поразрядно-параллельного вертикального сложения $СВ_j$ двоичных кодов строк пары совпадающих фрагментов дают в $PC_{output}^{(0)}$ цепочки подряд расположенных нулей. Количество нулей цепочки равно числу бит совпавшего фрагмента, расположение цепочек соответствует расположению совпавших фрагментов.

Каждая отдельная цепочка нулей может быть идентифицирована с использованием схемы сдваивания, аналогичной представленной на рис 1. Для пояснения можно привести простой пример: выделяются совпадающие фрагменты слов `'КИТ'` и `'КОТ'`. Двоичный код сравниваемых слов:

```
110010101100100011010010
110010101100111011010010
```

В результате шагов (3)–(4) получится

```
  110010101100100011010010
+ 110010101100111011010010
  ------------------------
  00000000000001100000000   PC(0)output
 110010101100100011010010   PC(1)output
```

Цепочки нулей в $PC_{output}^{(0)}$ дают совпадение фрагментов слов.

С некоторыми оговорками, оценка временной сложности идентификации совпадающих фрагментов двух $(n+1)$-разрядных слов примет вид

$$T_{identification\ of\ all\ matches}((n+1)n/2) \le \lceil \log_2(n+1)\rceil \tau_{binary} = O(\log_2 n).$$

Возросшее по сравнению с (7) количество элементов $R = (n+1)n/2$ объясняется тем, что аналог схемы на рис. 1 строится для каждого разряда сравниваемых слов. Алгоритм аналога отличается тем, что сдваивание начинается от нулевого разряда, слева от которого ненулевой разряд, так же как схема на рис. 1 начинается от старшего разряда.

В $PC_{output}^{(0)}$ цепочки единиц будут означать все не совпадающие фрагменты слов, в примере – несовпадающие части кодов символов `'И'` и `'О'`.

Развитие данных алгоритмов приводит к алгоритму параллельного сложения, использование которого позволяет всю входную и выходную информацию для арифметических операций представлять в дополнительном коде, без использования знаков в значениях разрядов.

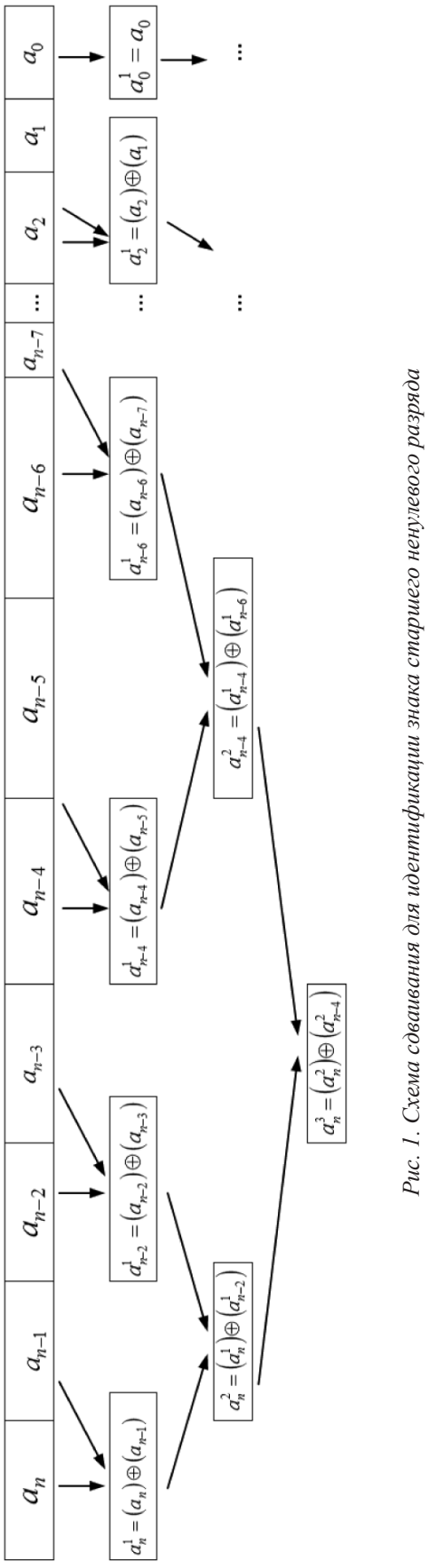


Рис. 1. Схема сдваивания для идентификации знака старшего ненулевого разряда

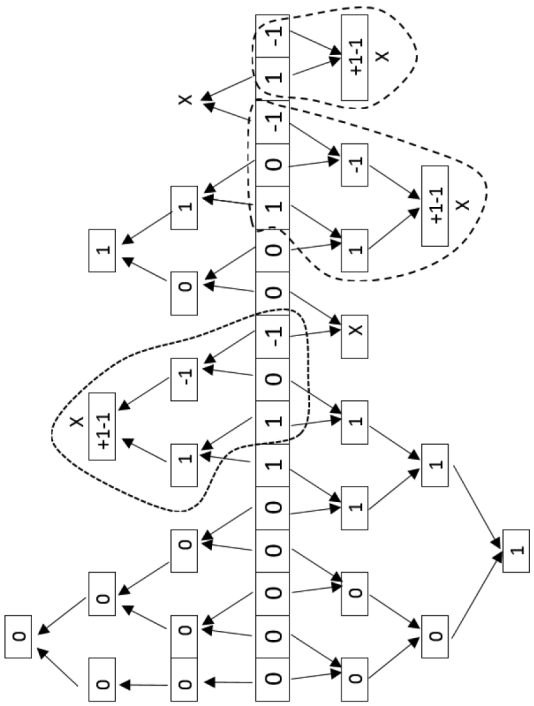


Рис. 2. Схема перевода знакоразрядного кода двоичной суммы в прямой код

Построение параллельного сумматора двоичных чисел

Пусть дана пара целых двоичных полиномов (1) и пусть над ними выполнены преобразования (2)–(5). В результате сумма получится в знакоразрядном двоичном коде, которую можно, как показано ниже, преобразовать в прямой двоичный код. Искомое преобразование удобно пояснить на примере.

Пример 1.

```
  0 1 1 0 1 0 1 1 0 0 0 0 0 0 0 0
+ 1 0 0 1 1 0 1 0 1 0 0 0 1 1 0 1
-------------------------------
  1 1 1 1 0 0 0 1 1 0 0 0 1 1 0 1
0 0 0 0 1 0 1 0 0 0 0 0 0 0 0 0
```

Над двухрядной суммой выполняется описанное ранее преобразование $\Delta_j \equiv 2\Delta_j - \Delta_j$, $j = 0, 1, ..., n$, значения разрядов $PC^{(0)}_{output}$ и $PC^{(1)}_{output}$ примут вид

```
 -1-1-1-1 0 0 0-1-1 0 0 0-1-1 0-1
+1 1 1 1 0 1 1 1 0 0 0 1 1 0 1 0
```

Взаимно независимое суммирование разрядных срезов по вертикали влечет

$$0\,0\,0\,0\,0\,1\,1\,0\text{-}1\,0\,0\,1\,0\text{-}1\,1\text{-}1 \qquad (8)$$

Полученный в знакоразрядном коде результат представляет сумму двух чисел. Она получается с единичным порядком временной сложности (за фиксированное конечное число шагов, не зависящее от числа разрядов слагаемых). Чтобы представить эту сумму в прямом двоичном коде, нужно преобразовать каждую комбинацию вида $2^{m+1} - 1$ к виду $2^{m+1} - 1 = 2^m + 2^{m-1} + 2^{m-2} + \cdots + 2^1 + 2^0$, или, в позиционной системе, комбинацию вида 100...0 – 1 к виду 011...11. Алгоритм иллюстрирует схема, данная для примера 1, затем будут сделаны обобщающие пояснения. Следующая схема переводит код (9) в прямой двоичный код.

Поскольку все комбинации вида 100...0 – 1 взаимно отделены друг от друга (преобразование (2)–(4)) вследствие первого вертикального шага сложения промежуточной парой вертикальных нулей, то все такие комбинации взаимно независимы и могут быть преобразованы в прямой код параллельно. Именно так работают обведенные пунктиром фрагменты схемы преобразования на рис. 2. Алгоритм работы каждого фрагмента такой схемы инвариантен относительно вида и расположения преобразуемой комбинации двоичных цифр, он заключается в следующем. Делается общая для всех разрядов схема сдваивания, которая начинается с младшего разряда справа и отображается сверху вниз. Такая же схема, отображенная вверх, делается для всех разрядов, начиная со следующего после младшего разряда (со сдвигом на единицу влево предыдущей – нижней схемы). В схеме сдваивания допускается перемещение –1 только справа налево. Дерево, в котором она должна идти слева направо, блокируется на первом шаге запрещенного направления. Такая –1 пойдет справа налево, однако по параллельно работающей схеме, изображенной на рис. 2, симметричным отражением вверх со сдвигом на один разряд. В верхней схеме, так же как и в нижней, блокируется создание шагов сдваивания для всех –1, идущих слева направо. В разрешенной схеме (в поддереве) шаги сдваивания выполняются до получения корня поддерева, в котором зафиксируется комбинация +1, –1. Такая комбинация прекращает движение по поддереву и начинает в нем движение в обратном направлении. Обратное движение выполняется снизу вверх в нижнем фрагменте, сверху вниз – в верхнем. При этом –1 из корня поддерева параллельно по всем уровням поддерева, в котором она проходила путь, заменяет все элементы этого поддерева до начальных элементов включительно значением +1, за исключением элементов пути, уже занятых +1. С другой стороны, +1 из корня того же поддерева проделывает обратный путь вдоль своего ранее проделанного движения, заменяя на своем пути все пройденные элементы поддерева до начальных элементов включительно с +1на 0. В результате, путем замены исходной комбинации 100...0 – 1, в разрядной сетке суммы образуется искомая комбинация начальных элементов: 011...11.

Обратное перемещение двоичных коэффициентов от корня поддерева к исходным разрядным значениям можно реализовать многими способами. Для определенности условно подразумевается следующий способ. Все элементы всех поддеревьев (обоих деревьев – верхнего и нижнего) продублированы. Дублирующие элементы инициируется по мере прохождения сигнала по дублируемому элементу в прямом направлении, но открывают свой выход только в обратном направлении. Сигнал в обратном направлении передается только по дублирующим элементам. Изложенные преобразования выполняются одновременно во всех поддеревьях нижней схемы. Такие же преобразования выполняются одновременно во всех поддеревьях верхней схемы. Поскольку переносы не пересекаются, и все такие поддеревья верхней и нижней схемы взаимно отделены, на выходе алгоритма в разрядной сетке образуются полностью

положительная сумма в двоичной форме. В каждом поддереве, не имеющем в корне комбинации +1, –1, по определению исключается движение в обратном направлении, и его входная комбинация двоичных цифр в разрядной сетке не меняется. В результате выполненного преобразования знакоразрядного кода (9) получится искомый прямой код суммы:

0 0 0 0 0 1 0 1 1 0 0 0 0 1 1 0 1

Замечание 1. Изложенный перевод гарантирован для исходного способа сложения, априори взаимно отделяющего все переносы друг от друга парой вертикальных нулей после шага (4). В общем случае, без наличия взаимной отделенности переносов, перевести знакоразрядный код в прямой по предложенной схеме нельзя.

Замечание 2. С использованием описанной схемы параллельного сложения двоичных полиномов входные и выходные данные для арифметических операций окажется возможным представлять в дополнительном коде, без использования знаков в битовых значениях разрядов. Однако для этого необходимо устранить существенное затруднение. Именно после перевода по схеме на рис. 2 комбинаций вида 100...0 – 1 в комбинации вида 011...11 одна комбинация с отрицательной единицей может остаться не преобразованной. Это произойдет в том случае, если старший ненулевой разряд суммы окажется отрицательным, например если комбинация старших разрядов имеет вид 000...0 – 1. В этом случае отрицательную единицу нельзя обработать по предложенной схеме. Кроме того знак «–» нельзя закодировать, поскольку это не только знак суммы, но и знак одного значения разряда конкретно определенного веса (веса старшего ненулевого разряда). Можно различными способами пытаться разрешить это затруднение. Одно из простейших решений заключается в следующем.

Вариант 1. На выходе сложения отрицательная единица, если она появляется, окажется единственной. Ее можно отделить со своим разрядным весом в качестве дополнительного слагаемого (оно отрицательно, состоит из единственного ненулевого разряда). Такое отрицательное слагаемое ничто не мешает представить в обратном коде. В результате можно завершить сложение с использованием того же способа разрядного распараллеливания. Как нетрудно видеть, преобразованная мантисса не будет содержать отрицательных разрядов.

Вариант 2. Еще один способ формально не использует представление данных в дополнительном коде. После вычленения –1 в отдельное слагаемое в разряд этого слагаемого с индексом $n + 1$ записывается значение +1. Получается комбинация вида $2^{n+1} - 2^m = 2^n + 2^{n-1} + 2^{n-2} + \cdots + 2^{m+1} + 2^m$, где m – индекс разряда, содержащего неустраненное значение –1. Тогда все разряды в правой части тождества могут заменить все нулевые старшие разряды, а также разряд с –1, на цепочку из $n - m + 1$ положительных единиц. Чтобы значение окончательного результата оказалось правильным, достаточно вычесть добавленную +1 из разряда с индексом $n + 1$. Получается, что –1 в разряде с индексом $n + 1$ и цепочка из +1 в разрядах с индексами n, $n - 1$,..., $m + 1$, m эквивалентна исходному значению –1 старшего ненулевого разряда с индексом m. Эту комбинацию можно возвратить в исходный результат сложения. В окончательном коде –1 в разряде с индексом $n + 1$ может использоваться в качестве идентификатора знака результата, в то же время это именно значение разряда данного веса. Во всех разрядах с меньшими номерами отрицательных значений не останется. Более точно, преобразование сводится к следующему правилу. При наличии –1 в старшем ненулевом разряде с индексом $m \le n$ этот разряд и все слева от него разряды, до разряда с индексом n включительно, следует заменить на значения +1, а в разряд с индексом $n + 1$ записать значение –1. Если старший ненулевой разряд априори имел индекс $m = n + 1$ и был при этом равен –1, то разряд с индексом $n + 1$ не преобразуется и полученный код считается окончательным кодом суммы полиномов вида (1). Чтобы фактически не выполнять лишних операций, можно –1 старшего ненулевого разряда вычленить в отдельный ряд вместе с нулевыми старшими разрядами. Весь отрезок ряда слева, начиная с –1, параллельно заменяется значениями разрядов +1. Одновременно в разряд с индексом $n + 1$ записывается значение –1. Справа от исходного значения –1 остаются нули. Полученный ряд параллельно складывается с исходным рядом, из которого была вычленена отрицательная единица. Реализация каждого такого шага требует аппаратной поддержки и влечет соответственную задержку, которая имеет единичное значение, тогда как предшествующие преобразования измерялись логарифмическим числом шагов.

Вариант 3. Еще один способ заключается в том, что результат преобразования, описанный в варианте 2, можно получить на этапе работы схемы на рис. 2 с помощью незначительной модификации поддерева старших разрядов этой схемы. Именно, если в разряде с индексом $n + 1$ априори

находилось значение +1, поддерево схемы сдваивания, включающее этот разряд, работает без изменения, как изложено при описании схемы на рис. 2. Если же в разряде с индексом $n+1$ априори находилось значение 0, то вместо этого значения на вход поддерева схемы подается +1. Работа этого поддерева строится по общему правилу. Единственное исключение состоит в том, что в конце перемещения в обратном от корня направлении априорное значение 0 в разряде с индексом $n+1$ заменяется значением –1. В результате исходная комбинация старших разрядов 000...0 – 1 в разрядной сетке суммы заменится на искомую комбинацию –111...11, где –1 является значением разряда с индексом $n+1$.

Замечание 3. С модификацией варианта 3 время работы схемы сдваивания не изменится. Этот вариант наряду со способом сложения двоичных полиномов одновременно дает знак сравнения слов строкового, а также числового типа, представленных в двоичном коде.

Описание схемы выходит за рамки примера 1, с учетом замечаний 1–3 и вариантов 1–3 оно инвариантно относительно разрядности, входного кода слагаемых, наличия и вида преобразуемых комбинаций.

Оценка временной и схемной сложности параллельного сумматора

С выбором варианта 3 временная сложность предложенного алгоритма параллельного сложения чисел вида (1) будет иметь логарифмическую от числа разрядов оценку. В самом деле, число последовательных шагов в направлении корня в поддереве схемы сдваивания не превосходит $\lceil \log_2 i \rceil$, где i – количество входных разрядов, преобразуемых в поддереве. Число шагов в обратном направлении не превосходит того же значения. Все поддеревья схемы выполняют преобразования параллельно. Таким образом, временная сложность преобразования одновременно всех разрядов оценивается из соотношения $T(R) \le \max_i \lceil \log_2 i \rceil \tau_{binary}$, или, $T(R) \le \lceil \log_2 (n+1) \rceil \tau_{binary}$. Количество элементов сумматора R не превзойдет значения пропорционального $n+1$. Именно в поддереве нижней схемы количество сдваиваемых элементов R_i, рассматриваемых по уровням от корня, составит $2^0, 2^1, \ldots, 2^{\lceil \log_2 i \rceil - 1}$, где i – количество разрядов, преобразуемых в поддереве. Отсюда $R_i \le \sum_{\ell=0}^{\lceil \log_2 i \rceil - 1} 2^\ell$, или, $R_i \le 2^{\lceil \log_2 i \rceil} - 1$. В результате $R_i \le 2i - 1$. Поскольку комбинации преобразуемых входных элементов не пересекаются в $(n+1)$-разрядной сетке, то суммарное число элементов поддеревьев не превысит суммы тех значений i разрядов, множество которых покроет разрядную сетку. В итоге $R \le \sum_\ell R_{i_\ell} \le \sum_\ell 2i_\ell - 1$, где $\sum_\ell i_\ell = n+1$. Тогда $\sum_\ell R_{i_\ell} \le 2(n+1)$. Это количество элементов нижней схемы удвоится с учетом предположения о дублирующих элементах. Аналогично оценивается число элементов верхней схемы. Отсюда общее число элементов $R \le 8(n+1)$. Если теперь принять во внимание, что предварительные шаги (2)–(5) выполняются на $O(n+1)$ элементах за время $T(n) = O(1)$, то общая окончательная оценка временной сложности и числа элементов всего параллельного сумматора может быть представлена в виде:

$$T(n) = O(\log_2 n). \qquad (9)$$

Оценка (10) может быть получена путем непосредственного подсчета последовательных шагов общей схемы сдваивания, $O(\log_2 n)$, а также подсчетом числа элементов этой общей схемы, $O(n)$, поскольку оценивавшиеся поддеревья вырезались из конфигурации общей схемы и не меняли количество ее элементов.

Общий результат можно представить в следующей формулировке.

Предложение 1. На основе описанных выше схем алгебраическое сложение двоичных полиномов вида (1) может быть выполнено с временной сложностью (10). С такой же оценкой может быть выполнено сравнение слов строкового и числового типа, представленных в $(n+1)$-разрядном двоичном коде.

Представленная алгоритмическая схема сложения двоичных полиномов отличается от известных схем [5, 6], а также от [15] по структуре и тем, что относится к полному количеству разрядов слагаемых (1), достигая оценки (10). Преимущество обеспечивает использование предварительного шага параллельного по вертикальным срезам сложения слагаемых вида (1), что отсутствует в схемах, с которыми проводилось сравнение.

Области применения

Предложенные схемы могут найти применение при разработке систем информационного поиска. В основу выполнения операций сравнения могут быть положены сравнения с маской, которая состоит из символов на выделенных позициях слова, сравнения можно распространить на конечное множество масок данного вида. При этом в предложенных схемах рассматриваемые

операции могут распараллеливаться по разрядам всего множества слов исследуемых данных, представленных в векторизованной форме. Формально подход можно отнести не только к словам строкового типа, но с оговорками относительно ограничений технологического, технического, программного и структурного характера его можно распространить на различные объекты, представленные в двоичном коде. С использованием способов, описанных в [11], предложенные схемы применимы для ускоренного по сравнению с последовательными методами [7–9] построения и обработки древовидных структур данных. Предложенные схемы дают эквивалентное преобразование двоичных арифметических операций, позволяющее достигать ускорения относительно известных схем суммирования [6, 15] за счет максимального разрядного распараллеливания.

Заключение

Изложен способ разрядного распараллеливания операций сравнения строк и сложения двоичных полиномов. Способ отличается исключением вычисления переноса, поэтому не использует требуемые для выполнения переноса логические схемы. На этой основе могут максимально параллельно реализоваться базовые операции поиска, включая сравнение и сортировку. По структуре, временной сложности и аппаратным затратам предложенный способ улучшает известные схемы разрядного распараллеливания арифметических операций, а также схемы базовых операций поиска. Структура предложенных алгоритмов разрядного распараллеливания делает возможным их применение для повышения точности вычислительных операций за счет удлинения разрядной сетки представления операндов и сокращения потерь значащих цифр в операциях с плавающей точкой.

Список литературы

1. Царев Р.Ю. Структуры и алгоритмы обработки данных. Красноярск: Сибирский федеральный университет, 2013. 160 с.

2. Чернов А.Ф. Ускорение поиска в индексах на основе R-деревьев // Программные продукты и системы. 2012. № 2 (98). С. 46–50.

3. Ромм Я.Е. Метод вертикальной обработки потока целочисленных групповых данных. I. Приложение к бинарным арифметическим операциям // Кибернетика и системный анализ. 1998. № 3. С. 123–151.

4. Ромм Я.Е. Метод вертикальной обработки потока целочисленных групповых данных. II. Приложение к бинарным арифметическим операциям // Кибернетика и системный анализ. 1998. № 6. С. 146–162.

5. Held S. Spirkl S.T. Binary Adder Circuits of Asymptotically Minimum Depth, Linear Size, and Fan-Out Two. ACM Transactions on Algorithms (TALG). January 2018. Vol. 14 Issue 1, Article No. 4.

6. Балака Е.С., Городецкий Д.А., Рухлов В.С., Щелоков А.Н. Разработка высокоскоростных сумматоров по модулю на базе комбинационных сумматоров с параллельным переносом // Известия ЮФУ. Технические науки. 2016. № 6 (179). [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/razrabotka-vysokoskorostnyh-summatorov-po-modulyu-na-baze-kombinatsionnyh-summatorov-s-parallelnym-perenosom (дата обращения: 28.04.2019).

7. Алексеев В.Е. Графы и алгоритмы. Структуры данных. Модели вычислений. Гриф УМО университетов РФ. М.: ИНТУИТ, 2014. 320 с.

8. Ахо А., Джон Э., Хопкрофт Д. Структуры данных и алгоритмы. М.: Вильямс, 2016. 400 с.

9. Вирт Н. Алгоритмы и структуры данных. М.: ДМК Пресс, 2010. 272 с.

10. Гасанов Э.Э. Теория хранения и поиска информации // Фундаментальная и прикладная математика. 2009. № 3–15. С. 47–73.

11. Чабанюк Д.А. Преобразование информационных данных и двоичных структур с минимизацией временной сложности на основе алгоритмов сортировки: автореф. ... канд. техн. наук. Таганрог, 2018. 21 с.

12. Ромм Я.Е., Белоконова С.С. Моделирование разрядного распараллеливания сравнений строковых и числовых элементов // Современные проблемы науки и образования. 2015. № 2. [Электронный ресурс]. URL: http://www.science-education.ru/122-21302 (дата обращения: 17.05.2019).

13. Ромм Я.Е., Белоконова С.С. Детерминированный информационный поиск на основе сортировки с распараллеливанием базовых операций. М.: Научный мир, 2014. 198 с.

14. Ромм Я.Е., Белоконова С.С. Параллельная идентификация совпадающих фрагментов строк с логарифмической временной сложностью // Современные наукоемкие технологии. 2016. № 9–3. С. 437–444.

15. Chen F., Wang G., Chen G., He Q. (2013) A Novel Neural Network Parallel Adder. In: Rojas I., Joya G., Gabestany J. (eds) Advances in Computational Intelligence. IWANN 2013. Lecture Notes in Computer Science. Vol. 7902. Springer, Berlin, Heidelberg.