УДК 004.2

ЖАБИН В.И.,
ЖАБИНА В.В.

ПОВЫШЕНИЕ ЭФФЕКТИВНОСТИ ПАРАЛЛЕЛЬНЫХ ВЫЧИСЛЕНИЙ В ПОТОКОВЫХ СИСТЕМАХ

Исследуются параллельные системы для реализации мелкозернистых алгоритмов на базе вычислительных модулей, выполняющих операции в избыточных системах счисления с фиксированной и с плавающей запятой при поразрядном вводе и выводе данных. Показано, что применение таких вычислительных модулей позволяет частично совмещать во времени выполнение зависимых по данным операции и упростить коммутационную среду систем. Выполнение операций с плавающей запятой позволяет расширить диапазон представления чисел и обеспечивает асинхронный режим передачи данных, что дает возможность ускорить вычисления.

In respect of fine-grained algorithms implementation there are considered parallel systems on the base of processors which perform operations using fixed and floating point arithmetic as well as redundant numerical systems for bit-by-bit data input and output. It is shown that such computing modules enable to partly execute simultaneously sequences of operations as well as to simplify respective system communication structure. The use of floating point arithmetic for operation execution enables to extend the range of number presentation as well as to provide the asynchronous mode for data transfer giving an opportunity to make calculations faster.

Введение

Время решения задач в параллельных вычислительных системах во многом зависит от реализуемого уровня параллелизма, который определяется зернистостью представления графа задач. В системах реального времени, например, связанных с траекторными вычислениями, необходимо обеспечить высокую скорость решения систем алгебраических уравнений, интерполяции функций различными методами и т.д. [1]. Наиболее высокая степень распараллеливания может быть достигнута при мелкозернистом представлении алгоритмов, когда вершинам графа задачи соответствуют отдельные операции. При этом максимально увеличивается число параллельных ветвей, что дает потенциальную возможность использовать в системе большее количество параллельно работающих вычислительных модулей (ВМ).

Однако, скорость обработки данных в параллельных системах связана не только с длительностью выполнения операций, но и с затратами времени на обмен информацией между параллельными ветвями. Как известно, увеличение степени распараллеливания вычислений сопряжено с ростом интенсивности обмена данными между ВМ. Этот фактор является весьма важным и должен учитываться при выборе архитектуры систем и организации вычислений.

Одна из возможностей уменьшения затрат времени на обмен данными состоит в использовании потоковых систем с непосредственными связями (ПСНС) между ВМ [2, 3]. В ПСНС выходы одних ВМ подключаются к входам других в соответствии с графом потока данных (ГПД). В процессе вычислений данные пересылаются от одних ВМ к другим, преобразуясь на каждом шаге в соответствии с определенной операцией, заданной ГПД. При такой организации вычислительного процесса не требуются сложные процедуры пересылки данных между ВМ, что создает предпосылки к уменьшению непроизводительных затрат времени в процессе обмена информацией.

Уменьшение сложности средств коммутации, в свою очередь, позволяет упростить и ускорить проектирование систем на базе заказных и программируемых СБИС с использованием современной технологии SoC (System on a Chip – система на кристалле). Однако со стороны элементной базы накладывается ряд ограничений, связанных с числом выводов микросхем, наличием встроенных функциональных узлов и устройств. Например, ресурсов одной микросхемы может не хватить для погружения всей системы, то есть возникает необходимость применения нескольких корпусов микросхем, связанных между собою внешними линиями связи. При передаче информации параллельным кодом возникают проблемы, связанные с возможной нехваткой выводов микросхем. Кроме того, снижается надежность систем, так как контактные соединения между микросхе-

мами относятся к ненадежным элементам. Возрастает также энергопотребление и увеличиваются габаритные размеры системы.

Учитывая важность этих проблем, в начале 90-х годов компания Virtual Machine Works обнародовала свою новую технологию под названием VirtualWire (виртуальные соединения), представленную как технология производства больших устройств, для реализации которых приходится использовать несколько ПЛИС [4]. Идея, заложенная в основе технологии VirtualWire, заключается в следующем. Часть ресурсов микросхем используется для реализации специальных цепей, которые позволяют подключать выводы микросхем попеременно к разным источникам сигналов внутри ПЛИС. Данная технология, хотя и помогает решить проблему нехватки выводов микросхем, не позволяет в полной мере использовать незадействованные ресурсы микросхем для решения заданной задачи. Кроме того, мультиплексирование создает дополнительные временные задержки продвижения потоков данных, что противоречит самой идее потоковой модели вычислений.

Одним из эффективных подходов к решению проблемы сокращения средств коммутации в ПСНС является использование в качестве ВМ операционных устройств, выполняющих операции в неавтономном режиме при поразрядном вводе и выводе данных, начиная со старших разрядов. Благодаря поразрядному обмену данными существенно сокращаются аппаратурные затраты на средства коммутации. Однако в работах, посвященных использованию неавтономных вычислений в ПСНС, рассматриваются операции с фиксированной запятой, что приводит к ряду недостатков.

В статье анализируются недостатки применения в ПСНС арифметики с фиксированной запятой и исследуется возможность повышения эффективности вычислений за счет выполнения арифметических операций с плавающей запятой.

Организация неавтономных вычислений в ПСНС

При реализации конкретных алгоритмов в ПСНС требуется обеспечить передачу данных (промежуточных результатов) от одних ВМ к другим в соответствии с ГПД. В реконфигурируемых ПСНС (рис. 1) необходимое соединение между ВМ обеспечивает коммутационная среда, которая настраивается в соответствии с ГПД. Проблемам реконфигурации систем посвящено много научных работ, в том числе, монографий, например, [5, 6]. В системах с жесткими связями ВМ соединяются между собой непосредственно на стадии изготовления, что ограничивает функциональные возможности систем, делая их фактически специализированными. Известны формальные методики перехода от ГПД к структуре системы, например, [2].

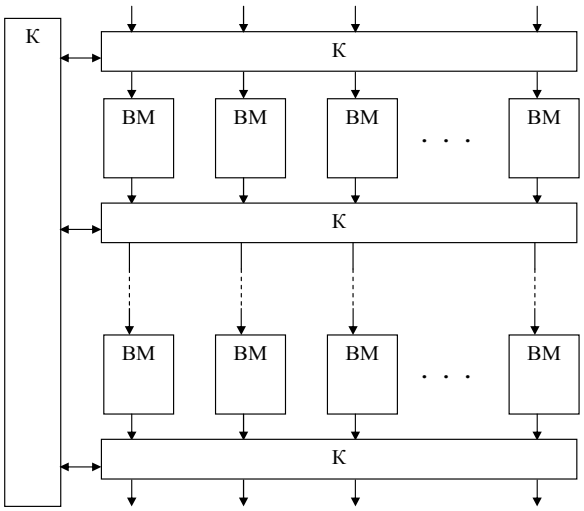

Рис. 1. Реконфигурированная параллельная вычислительная система: К – коммутатор

Любой путь в ГПД представляет собой цепочку связанных между собой по данным операций, которые нельзя распараллелить, так как предыдущая операция формирует операнд для следующей. Параллельно во времени могут выполняться только операции, которые принадлежат разным путям. Арифметические операции со старших разрядов выполняются с использованием однородной избыточной системы счисления с целым основанием k. Известны алгоритмы обработки данных как в симметричных [7, 8], так и в смещенных [9, 10] системах счисления соответственно с цифрами $\{\overline{-g, g}\}$ и $\{\overline{0, g}\}$.

Последовательность зависимых по данным операций выполняется в неавтономном режиме с помощью цепочки ВМ, которые позволяют совмещать выполнение зависимых операций на уровне обработки разрядов слов следующим образом. На каждом шаге в ВМ вводится по одному разряду операндов и формируется один разряд результата, начиная со старших разрядов. При этом разряд промежуточного результата, полученный на i-м шаге в одном ВМ при выполнении j-й операции, может быть исполь-

зован на $(i+1)$-м шаге в другом ВМ при выполнении $(j+1)$-й операции. При таком режиме вычислений выполнение последующей операции будет начинаться не после завершения выполнения предыдущей операции, а сразу же после получения первого разряда результата этой операции. Каждый ВМ начинает формировать разряды операндов с некоторой задержкой на p_j шагов после получения первых разрядов операндов. В таких ВМ информация представлена последовательным кодом только на входах и выходах. Между узлами (регистрами, сумматорами) информация передается параллельным кодом, в связи с чем такие устройства называют "квазипараллельными". Режим работы ВМ называют неавтономным, так как для выполнения последовательности операций необходима совместная работа нескольких ВМ, которые синхронно обмениваются информацией в процессе работы. Если ГПД содержит критический путь из N операций, то при максимальном распараллеливании алгоритм может быть реализован за время [3]

$$T = \left[n - 1 + \sum_{j=1}^{N} \left(p_j + 1 \right) \right] T_{\text{ц}},$$

где n – разрядность операндов, $T_{\text{ц}}$ – длительность цикла формирования цифры результата в ВМ.

Частичное совмещение выполнения зависимых операций дает потенциальную возможность ускорения обработки данных в потоковых системах на базе квазипараллельных ВМ, по сравнению с параллельными ВМ [2, 3]. Для сравнения времени решения задач в системах с параллельными и квазипараллельными ВМ необходимо определить значения указанных выше параметров k, p_j и $T_{\text{ц}}$ для конкретной структурной организации модулей. Можно указать следующие недостатки синхронных вычислений с фиксированной запятой:

– существенные ограничения на диапазон представления чисел при заданной разрядности устройств;

– накопление погрешности при выполнении последовательности операций вплоть до полной потери значимости результатов вычислений;

– сложность масштабирования исходных операндов для обеспечения необходимого соотношения их величин в определенном месте цепочки операций (например, при делении делимое должно быть меньше делителя, а при сложении разряды слагаемых должны иметь одинаковый вес);

– необходимость обеспечения одновременного поступления разрядов операндов на входы операционных устройств из разных источников, что требует выбора максимального по длительности такта из всех возможных для разных операций.

Для повышения эффективности ПСНС целесообразно реализовать следующее:

– разработать методы неавтономного выполнения операций с плавающей запятой с целью повышения точности вычислений и расширения диапазона представления чисел;

– разработать методику реализации вычислений в асинхронном режиме с целью упрощения системы синхронизации и ускорения вычислений.

Обработка данных в форме с плавающей запятой в неавтономном режиме

Число X в позиционной канонической (неизбыточной) системе счисления с основанием k можно записать в форме с плавающей запятой как

$$X = k^P M \ ,$$

где M -мантисса, а P - порядок числа X.

В канонической системе счисления порядок P является целым числом, а мантисса – нормализованной дробью, то есть

$$k^{-1} \le |M| \le 1.$$

Если для записи порядка используется m разрядов, а для записи мантиссы – n разрядов, то максимальное по модулю число в форме с плавающей запятой может быть представлено следующим образом

$$k^{k^m - 1}(1 - k^{-n}) = k^{k^m - 1} - k^{k^m - n - 1}.$$

В свою очередь, минимальное по абсолютной величине число, которое отличается от нуля, записывают как

$$k^{-k^m + 1} k^{-1} = k^{-k^m}.$$

Для компьютеров разработан стандарт ANSI/IEEE 754 представления чисел с плавающей запятой. В нем используют скрытый старший разряд мантиссы с весом 1 (разрядность мантиссы увеличивается на один разряд). Для упрощения работы с порядками при выполнении операций используется смещенный порядок, позволяющий работать с целыми

числами без знаков. Заметим, что при $k > 2$ старший значащий разряд мантиссы может иметь различные значения, то есть не может быть скрытым из-за неоднозначности.

Алгоритмы выполнения операций с плавающей запятой сложнее, чем с фиксированной. Однако при этом увеличивается точность вычислений, диапазон представления чисел, а также устраняются трудности, связанные с масштабированием данных.

Рассмотрим возможный вариант представления чисел с плавающей запятой в избыточной системе счисления.

С порядками операции могут выполняться в автономном режиме, то есть в каждом вычислительном модуле операции выполняются независимо. В связи с этим нет необходимости использовать для представления порядков избыточный код. Порядок может быть представлен в виде целого числа в любом коде, в том числе, в канонической двоичной системе. Обработка порядков, которая сводится к сложению, вычитанию и сравнению кодов, обычно выполняется быстрее, чем поразрядная обработка мантисс и, как правило, может с ней совмещаться.

Более сложной является обработка мантисс. При выполнении арифметических операций возникает проблема, связанная с возможным нарушением нормализации. В канонических системах нормализация заключается в сдвиге мантиссы влево или вправо и коррекции порядка. При неавтономных вычислениях разряды мантиссы формируются поочередно, начиная со старших. Сдвиг можно осуществить только путем задержки разрядов при приеме в вычислительный модуль, что эквивалентно сдвигу кодов вправо. Сдвиг влево таким же образом невозможен.

Вторая проблема заключается в том, что наличие значащего разряда мантиссы справа от запятой не означает, что мантисса является дробным числом. Это связано с тем, что цифры в избыточном представлении могут превышать основание системы счисления. Например, в двоичной системе с цифрами {2,1,0} число 0,22102=1,10110 больше единицы, а в системе с цифрами {-1,0,1} число $0,1\bar{1}\bar{1}\bar{1}\bar{1} = 0,00001$ значительно меньше единице. Все это требует специальных алгоритмов обработки чисел в избыточных системах.

Будем считать, что числа с плавающей запятой имеют вид

$$X = k^P M ,$$

где порядок P – целое число в неизбыточном коде, а мантисса M представлена в избыточном коде и находится в пределах

$$k^{-1} \le |M| < M_{\max} .$$

Значение $M_{\max}$ определяется основанием системы счисления и диапазоном изменения значений цифр.

Пусть мантисса числа X представлена в форме

$$M_x = \sum_{i=1}^{n} x_i k^{-i} ,$$

где $x_i \in \{\overline{g,0}\}$ или $x_i \in \{\overline{-g,g}\}$.

В обоих случаях $M_{\max}$ будет иметь вид

$$0,gg...g = g\sum_{i=1}^{n} k^{-1} . \qquad (1)$$

Определим предел суммы в правой части (1), представленной в виде ряда

$$\frac{1}{k} + \frac{1}{k^2} + \frac{1}{k^3} + ... + \frac{1}{k^n} .$$

Находим частичные суммы:

$$s_1 = \frac{1}{k},\ s_2 = \frac{k+1}{k^2},\ s_3 = \frac{k^2+k+1}{k^3},\ldots,$$
$$s_n = \frac{k^{n-1} + k^{n-2} + ... + k + 1}{k^n} .$$

Исследуя сумму s_n, можно найти

$$\lim_{n\to\infty} s_n = \frac{1}{k-1} .$$

С учетом полученного предела суммы и (1) определим диапазон изменения мантиссы в избыточном представлении как

$$k^{-1} \le |M| < \frac{g}{k-1} . \qquad (2)$$

Рассмотрим возможные интервалы изменения значений мантисс результатов различных операций.

Мантисса произведения, исходя из диапазона изменения мантисс (2), находится в пределах

$$k^{-2} \le |M| < \frac{g^2}{(k-1)^2}. \qquad (3)$$

Анализируя (3), можно показать, что при умножении, в отличие от канонической системы счисления, нормализация относительно (2) может быть нарушена как вправо, так и влево.

В соответствии с (2) мантисса частного от деления изменяется в диапазоне

$$\frac{k-1}{gk} \le |M| < \frac{gk}{k-1}. \qquad (4)$$

Можно показать, что нарушение нормализации может произойти как влево, так и вправо. Следует указать, что при делении мантисс необходимо, чтобы делимое было меньше делителя. Поэтому разряды делимого надо принимать в ВМ с задержкой, величина которой определяется значениями g и k. При каждой задержке надо прибавлять единицу к порядку результата.

При сложении (вычитании) мантисс результат может находиться в пределах

$$0 \le |M| < \frac{2g}{k-1}. \qquad (5)$$

Влево нормализация может быть нарушена на один разряд, а вправо – на все n разрядов.

В соответствии с (2) можно определить интервал изменения значений мантиссы для других функций. Например, для квадратного корня получим

$$\sqrt{k^{-1}} \le M < \sqrt{\frac{g}{k-1}}. \qquad (6)$$

Учитывая диапазоны изменения чисел (3), (4), (5) и (6), рассмотрим алгоритмы выполнения операций в ВМ.

Считаем, что ВМ содержит блоки для обработки мантисс и порядков, которые взаимодействуют между собой. Цифры мантисс принимаются в модуль из предыдущего модуля, начиная со старших разрядов. Передача первой значащей цифры стробируется. До начала обработки разряды операндов могут накапливаться в буфере модуля. Порядки могут передаваться и обрабатываться в неизбыточном коде. Обработка порядков во времени может совмещаться с обработкой мантисс.

Алгоритм операции умножения $Z = YX$**.**

1. Принять порядки P_x и P_y.

2. Получить предварительный порядок результата $P_z = P_x + P_y + 1$.

3. Принимать в цикле цифры мантисс операндов x_i и y_i; формировать цифры мантиссы результата z_i; пока $z_i = 0$ корректировать порядок $P_z := P_z - 1$; цифру $z_i = 0$ не выдавать.

4. При формировании первой ненулевой цифры результата z_i выдать окончательный порядок P_z и данную первую ненулевую цифру мантиссы результата.

5. Принимать очередные цифры мантисс операндов и выдавать очередные цифры мантиссы результата до получения n разрядов мантиссы результата.

Здесь считаем, что нарушение нормализации влево возможно на один разряд, что определяется значениями g и k.

Алгоритм операции деления $Z = Y / X$**.**

1. Принять порядки P_x и P_y.

2. Получить предварительный порядок результата $P_z = P_y - P_x + 1$.

3. В первом цикле, принять старшую цифру мантиссы делителя и 0 в качестве старшей цифры мантиссы делимого.

4. Принимать в цикле цифры мантисс операндов x_i и y_i; формировать цифры мантиссы результата z_i; пока $z_i = 0$ корректировать порядок $P_z := P_z - 1$; цифру $z_i = 0$ не выдавать.

5. При формирования первой ненулевой цифры результата z_i выдать окончательный порядок P_z и данную первую ненулевую цифру мантиссы результата.

6. Принимать очередные цифры мантисс операндов и выдавать очередные цифры мантиссы результата до получения n разрядов мантиссы результата.

Считаем, что для рассматриваемой системы счисления делимое достаточно задержать на один разряд.

Алгоритм операции сложения $Z = Y + X$**.**

1. Принять порядки P_x и P_y.

2. Получить предварительный порядок результата $P_z = \max(P_x, P_y) + 1$ и разность порядков $\Delta P = P_{\max} - P_{\min}$.

3. Пока $\Delta P \neq 0$ принимать в цикле цифру мантиссы операнда с большим порядком и 0 в качестве цифры другого операнда; выполнять $\Delta P := \Delta P - 1$.

4. В последующих циклах ($\Delta P = 0$) принимать цифры обоих слагаемых и формировать цифру мантиссы результата z_i.

5. Если $z_i \neq 0$, то выдать окончательный порядок P_z и z_i в качестве старшей цифры мантиссы результата; в противном случае корректировать порядок $P_z := P_z - 1$, цифру $z_i \neq 0$ не выдавать.

6. Принимать очередные цифры мантисс операндов и выдавать очередные цифры мантиссы результата до получения n разрядов мантиссы результата.

Возможность переполнения разрядной сетки учитывается в п. 2 добавлением 1 к порядку. Выравнивание порядков выполняется в п. 4. Нормализацию результата обеспечивает п. 5. При нулевом значении n разрядов результата в ВМ формируется отображение машинного нуля.

Алгоритм извлечения квадратного корня $Y = \sqrt{X}$**.**

1. Принять порядок операнда P_x.

2. Если порядок P_x нечетный, то принять 0 в качестве старшей цифры мантиссы операнда; скорректировать порядок $P_x := P_x + 1$, иначе коррекция порядка не нужна.

3. Сформировать и выдать порядок результата $P_z := k^{-1} P_x$.

4. Принимать цифры операнда x_i, формировать и выдавать цифры мантиссы результата z_i до получения n разрядов мантиссы результата Z.

Коррекция порядка операнда в п. 2 обеспечивает получения целого значения порядка результата.

При использовании симметричных систем счисления возможны ситуации, когда первая значащая цифра результата в каждом цикле фактически сдвигается вправо, то есть в данном цикле должна иметь нулевое значение. Например, $\bar{1}\bar{1}\bar{1}\bar{1}1 = 00001$. В этом случае выдача цифры из ВМ задерживается, а в следующем цикле проверяется значение следующей за ней цифры. Цифра выдается из модуля, когда превращение ее в 0 невозможно. Аппаратная реализация этого процесса достаточно проста.

Таким образом, применение плавающей запятой расширяет диапазон представления чисел по сравнению с фиксированной запятой. Если для записи порядка используется m разрядов, а для записи мантиссы – n разрядов, то число X в форме с плавающей запятой в системе с основанием k может находиться в пределах

$$k^{-k^m+1} k^{-1} \leq X < k^{k^m-1} \frac{g}{k-1}.$$

При одинаковой длине разрядной сетки обеспечивается повышение точности вычислений по сравнению с фиксированной запятой.

Предложенная методика выполнения операций с плавающей запятой позволяет асинхронно передавать данные между ВМ, то есть каждый ВМ может иметь собственную частоту тактирования. Это создает предпосылки для ускорения вычислений в ПСНС.

Выводы

В работе исследована возможность повышения эффективности параллельных вычислений в потоковых системах с непосредственными связями между вычислительными модулями, работающими в избыточных системах счисления в неавтономном режиме.

Разработаны алгоритмы неавтономного выполнения основных арифметических операций с плавающей запятой, позволяющие совмещать процессы поразрядного ввода операндов, их обработки и поразрядного вывода результата, начиная со старших разрядов.

Благодаря этому имеется потенциальная возможность ускорения вычислений за счет выполнения цепочек зависимых по данным операций в режиме совмещения, что нельзя обеспечить в автономном режиме.

Показано, что по сравнению с использованием фиксированной запятой применение операций с плавающей запятой в потоковых системах позволяет расширить диапазон представления чисел, повысить точность получения результатов при заданной длине разрядной сетки и ускорить вычисления за счет обработки информации в асинхронном режиме.

Список литературы

1. Байков В.Д. Решение траекторных задач в микропроцессорных системах ЧПУ / В.Д.Байков, С.Н.Вашкевич. – Л.: Машиностроение, 1986, 105 с.
2. Жабин В.И. Построение быстродействующих специализированных вычислителей для реализации многоместных выражений / В.И.Жабин, В.И.Корнейчук, В.П.Тарасенко // Автоматика и вычислительная техника. – 1981. - №6. – с. 18-22.
3. Жабин В.И. Выполнение последовательностей зависимых операций в режиме совмещения / В.И.Жабин. // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка: Зб. Наук. Пр. – К.: Век+. – 2007. - №46. – С. 226-233.
4. Максфилд К. Проектирование на ПЛИС. Архитектура, средства и методы / К.Максфилд. – М.: Издательский дом «Додэка-XXI», 2007, 408 с.
5. Палагин А.В. Реконфигурируемые вычислительные системы: Основы и приложения / А.В.Палагин, В.Н.Опанасенко. – К.: Просвіта, 2006, 280 с.
6. Каляев И.А. Архитектура семейства реконфигурируемых вычислительных систем на основе ПЛИС / И.А. Каляев, И.И. Левин, Е.А. Семерников // Искусственный интеллект. – 2008. - № 3. – с. 663-674.
7. Жабин В.И. Некоторые машинные методы вычисления рациональных функций многих аргументов / В.И. Жабин, В.И.Корнейчук, В.П.Тарасенко // Автоматика и телемеханика. – 1977. - №12. – С. 145-154.
8. Жабин В.И. Методы быстрого неавтономного воспроизведения функций / В.И.Жабин, В.И.Корнейчук, В.П.Тарасенко // Управляющие системы и машины. – 1977. - №3. – С. 96-101.
9. Дичка И.А. Совмещение зависимых операций на уровне обработки разрядов операндов / И.А.Дичка, В.В.Жабина. // Искусственный интеллект. – 2008. - №3. – С. 649-654.
10. Дичка И.А. Метод вычисления функций в неавтономном режиме / И.А.Дичка, В.В.Жабина // Искусственный интеллект. – 2009. – №4. – С. 409-414.