ISSN 2079-3316 ПРОГРАММНЫЕ СИСТЕМЫ: ТЕОРИЯ И ПРИЛОЖЕНИЯ т. 10, № 4(43), с. 201–217

ББК 3971.32-043:22.151.511
ГРНТИ 50.09.33, 50.33.04
УДК 004.272.25:004.272.44

Н. И. Дикарев, Б. М. Шабанов, А. С. Шмелёв

Мелкоструктурный параллелизм и более высокая производительность процессорного ядра: преимущества векторного потокового процессора

Аннотация. В настоящее время резервы в повышении производительности современных процессоров практически исчерпаны, что проявляется в отсутствии роста, как тактовой частоты, так и числа команд, выполняемых в такт, которые определяют скалярную производительность процессорного ядра. В разрабатываемом векторном процессоре с архитектурой управления потоком данных (векторном потоковом процессоре) производительность процессорного ядра может быть повышена до 256 флоп в такт на ядро, что в 8 раз выше по сравнению с последними процессорами Intel Xeon. Это достигается за счет более высокой доли векторных вычислений. В работе показано, что отношение реальной производительности к пиковой на программах битонной сортировки, умножения матриц и 2D Stencil у векторного потокового процессора выше, чем у лучших процессоров традиционной архитектуры.

Ключевые слова и фразы: векторный процессор, архитектура управления потоком данных, многопроцессорная система с общей памятью, оценка производительности.

Введение

Векторная производительность в последних процессорах Intel Xeon благодаря использованию системы команд AVX-512 была повышена до 32 флоп в такт на ядро, что в 8 раз превышает их скалярную производительность. Поскольку резервы в повышении скалярной производительности у современных процессоров практически исчерпаны, то дальнейшее увеличение векторной производительности процессорного ядра также нецелесообразно. Действительно, в векторном процессоре, как и любой гетерогенной системе, в соответствии с законом Амдала,

Работа выполнена при поддержке грантов РФФИ 17-07-01457, 19-07-00987, и с использованием вычислительных ресурсов МСЦ РАН.

10.25209/2079-3316-2019-10-4-201-217

максимальное ускорение за счет векторизации ограничивается той долей, которая в общем объёме вычислений приходится на скалярную обработку. Эта доля в большинстве современных процессоров, использующих обработку коротких векторов, не может быть ниже 15% — 20%, поскольку на одну выполненную векторную команду, как правило, приходится не меньше чем одна скалярная. Такое ограничение объясняется тем, что даже в векторизуемых циклах с большим числом итераций их приходится нарезать на куски, равные аппаратной длине вектора, что выполняется с помощью скалярных команд.

Другими недостатками современных процессоров являются, снижение производительности при увеличении времени выборки из памяти, превышающем 12 — 15 тактов, и отсутствие ускорения при распараллеливании на мелкие программные блоки по процессорным ядрам. Всё это вместе с невозможностью повысить производительность одного процессорного ядра приводит к низкой эффективности выполнения многих пользовательских программ в высокопроизводительных вычислительных системах, построенных на процессорах традиционной архитектуры.

Как правило, реальная производительность пользовательских программ при их выполнении на суперкомпьютерах составляет лишь 7% - 15% от пиковой производительности вычислительной системы. Только на библиотечном тесте Linpack, который сильно оптимизирован поставщиками, суперкомпьютеры имеют производительность больше 70% от их пиковой производительности. При этом программа блочного умножения матриц, на которую приходится основной объём вычислений в тесте Linpack, при её компиляции с языка высокого уровня с оптимизацией только установкой ключей компилятора имеет эффективность порядка 20% при выполнении на одном процессорном ядре [1].

Цель данной работы — показать, что разрабатываемый в МСЦ РАН векторный потоковый процессор (ВПП) может обеспечить не только значительно более высокую производительность одного процессорного ядра по сравнению с процессорами традиционной архитектуры, но и более высокую эффективность при выполнении программ битонной сортировки, умножения матриц и Stencil.

В разделе 1 будут рассмотрены преимущества и сложности реализации потокового процессора. В разделе 2 — особенности реализации ВПП и эффективность его работы на программе умножения матриц. В разделе 3 мы сравним производительность ВПП и Intel Skylake

на программе битонной сортировки. Наконец, в заключении приведены полученные ранее данные о более высокой эффективности ВПП при выполнении программ 2D и 3D Stencil и одинаковой реальной производительности с единицы площади процессорного кристалла у ВПП и графических ускорителей NVIDIA.

1. Преимущества и недостатки потокового процессора

Разрабатываемый ВПП имеет архитектуру управления потоком данных. В потоковом процессоре поиск команд с готовыми операндами производится в ассоциативной памяти (АП) по совпадению контекста у токенов операндов. Напомним, что программой в потоковом процессоре является однонаправленный граф, и данные между командами (узлами графа) передаются по дугам в виде пакетов — токенов. Помимо значения результата команды передатчика токен содержит контекст, в который входят номер команды приёмника в графе программы и ещё несколько полей «цвета» токена. «Цвет» токена включает номера итераций вложенных циклов и вызовов подпрограмм, что позволяет одновременно выполнять команды с одинаковым номером, но из разных итераций циклов и вызовов подпрограмм [2].

Расслоение АП на M модулей позволяет потоковому процессору выполнять до M команд в такт, и за счет увеличения числа модулей M — наращивать производительность. Суммарная ёмкость модулей АП определяет окно поиска готовых команд в потоковом процессоре, которое можно сделать значительно больше, чем в процессоре традиционной архитектуре. Так в каждом ядре ВПП планируется установить 8 модулей АП с суммарной ёмкостью 16 тысяч команд, то есть окно поиска команд с готовыми операндами в ВПП почти в 100 раз больше, чем в лучших современных процессорах. Это позволяет в процессе выполнения программы выявлять значительно больше параллелизма, причем мелкоструктурного, и увеличить реальную производительность процессора, например, за счет устранения простоев при увеличении времени выборки данных из памяти.

Процессоры с архитектурой управления потоком данных активно исследовались и разрабатывались до 2000 года, однако ни один из проектов потокового процессора не смог составить конкуренцию процессорам традиционной архитектуры, быстрое совершенствование которых также происходило в те годы. Основными причинами неконкурентоспособности потокового процессора являются слишком мелкий параллелизм (уровня команд), в 2 — 3 раза большее число выполняемых

команд, сложность работы с массивами данных и необходимость АП большой ёмкости и высокого быстродействия [3]. Из современных проектов реализации потокового процессора отметим вычислительную систему с автоматическим распределением ресурсов [4], в которой для укрупнения гранул параллельной работы используются макрооператоры с хранением промежуточных результатов в регистровом файле. В этом проекте для хранения массивов данных предлагается использовать модули АП, что создаёт серьёзные проблемы при её реализации из-за необходимости обеспечения большой ёмкости и высокого быстродействия памяти с ассоциативным поиском.

Заметим, что в более ранних проектах потокового процессора, например в [2], старались этого не допускать, и для хранения массивов использовалась память I-структур на основе обычной линейно-адресуемой памяти. Для исключения конфликтов запись после записи в этой памяти разрешалась лишь однократная запись элементов массива, а для исключения конфликтов чтение после записи - команды чтения, поступившие раньше времени, хранились в специальных буферах до момента записи элемента. Требование однократной записи приводило к избыточному копированию массивов и являлось одной из причин увеличения числа выполняемых команд по сравнению с процессором традиционной архитектуры, а наличие специальных буферов приводило к росту аппаратных затрат и увеличению времени выборки из памяти.

2. Особенности реализации ВПП и эффективность его работы

В ВПП для хранения массивов имеется аналог оперативной памяти на микросхемах динамической памяти, которая называется память векторов (ПВ), и память меньшей ёмкости — локальная память векторов (ЛПВ) на процессорном кристалле. При этом реализация памяти в ВПП имеет ряд существенных отличий от современных процессоров традиционной архитектуры. Так, в ВПП отсутствуют регистровые файлы и кэш данных.

Значения операндов для скалярных команд в ВПП поступают на вход исполнительного устройства (ИУ) из АП и уничтожаются после вычисления результата, как это принято в потоковых процессорах. Операнды векторных команд читаются преимущественно из ЛПВ, и результаты векторных команд записываются обратно в ЛПВ, которая в ВПП выполняет функцию регистрового файла и кэш данных одновременно.

Отличие ЛПВ от кэш данных в том, что она не является продолжением оперативной памяти, а имеет своё обособленное адресное пространство. Поскольку производительность разрабатываемого ВПП (256 флоп в такт) в 8 раз выше, чем в последних процессорах Intel Skylake (32 флоп в такт), то и ёмкость ЛПВ (4 MB) в несколько раз больше, чем кэш второго уровня у процессорного ядра Skylake. Столь большая ёмкость ЛПВ требуется для того, чтобы эффективно использовать большое окно поиска готовых команд в ВПП, в том числе векторных, а также копировать массивы данных по частям из ПВ в ЛПВ, как это делается в блочных алгоритмах, чтобы локализовать вычисления в пределах находящейся в ЛПВ части массива.

Однако главным отличием ВПП от процессоров традиционной архитектуры является аппаратное распределение ресурса памяти, а именно ПВ и ЛПВ. Чтобы каждый результат векторной команды имел свой уникальный адрес для его использования в качестве операнда другими векторными командами, было предложено аппаратно выделять новый адрес в памяти для записи вектора результата [5]. Это позволило устранить в ВПП адресные вычисления, которые в современных процессорах необходимы для определения адреса вектора в большом массиве. Например, для матрицы, состоящей из строк, нужно вычислить адрес начального элемента вектора конкретной стоки.

Другим преимуществом такого решения является возможность до записи массива (матрицы) целиком начинать читать его отдельные вектора строки для последующих вычислений, что повышает параллелизм векторной обработки, и, следовательно, производительность ВПП. Возникающая при этом проблема разрешения конфликтов информационной зависимости RAW — читать уже записанные элементы вектора — решается в ВПП тем, что токен с указателем вектора результата (адрес начального элемента, тип памяти (ПВ или ЛПВ) и число элементов VL) посылается в АП лишь после того, как вектор записан в память. Тогда команды, ожидающие в АП этот вектор для его использования в качестве операнда, после выдачи из АП, будут читать уже записанный вектор.

В ВПП аппаратная длина вектора VLmax=256 элементов по 8 байт каждый, чтобы в качестве элементов хранить числа с плавающей запятой двойной точности, соответственно для записи вектора в память требуется свободный фрагмент равный 2 KB. При фиксированной длине выделяемых фрагментов быстрое распределение памяти можно организовать используя список свободных векторов для каждого

уровня памяти (ПВ и ЛПВ) [6]. Входящее в состав ВПП устройство распределения памяти выделяет для записи результата векторной команды свободный вектор из нужного списка, и векторная команда, использующая вектор в последний раз, возвращает его в список свободных. Для хранения в памяти массивов произвольной длины в [5] было предложено использовать вектора-указатели, то есть вектора, элементами которых являются указатели векторов подмассивов, и лишь самый нижний уровень такой древовидной структуры содержит ссылки на вектора, элементами которых являются числа.

Там же было показано, что одной командой "Формирование Потока"(ФП) можно считать указатели строк, содержащиеся в векторе-указателе матрицы, и сформировать из них поток токенов для выполнения одинаковых операций над всеми строками матрицы. Эти операции над векторами, содержащими строки, можно выполнять одновременно, так как у токенов с указателями строк команда ФП проставляет разные значения номера итерации в поле контекста.

Использование команд ФП в программе умножения матриц позволило уменьшить общее число выполняемых команд по сравнению с векторным процессором традиционной архитектуры в 2,7 раза, причем число скалярных команд ещё сильнее — в 5 раз [6]. Заметим, что увеличение аппаратной длины вектора VLmax в процессоре приводит к снижению доли скалярных операций при выполнении программы. Следовательно, использование команд ФП позволяет значительно снизить долю скалярных операций в программах, выполняемых в ВПП, по сравнению с классическими векторными процессорами с той же VLmax, и еще в большей степени по сравнению с микропроцессорами, поскольку в них аппаратная длина вектора меньше в 32 раза. В свою очередь, это позволяет повысить производительность векторной обработки в ВПП по отношению к скалярной производительности, сохраняя при этом высокую эффективность, то есть высокое отношение реальной производительности к пиковой.

На рисунке 1 приведены значения производительности ВПП на программе блочного умножения матриц в зависимости от размера матрицы и аналогичные данные для вычислительной системы из 1, 8 и 16 процессорных ядер Intel Xeon E5. Для ВПП приведены результаты с двумя вариантами реализации ИУ: раздельные схемы умножителя и сумматора с плавающей запятой и слитная (fused) схема умножителя и сумматора (FMA). Такие (fused) ИУ используются в большинстве современных процессоров, поскольку это позволяет уменьшить число

выполняемых арифметических команд при той же производительности процессора, и число портов в регистровом файле (в ЛПВ применительно к ВПП) [7].

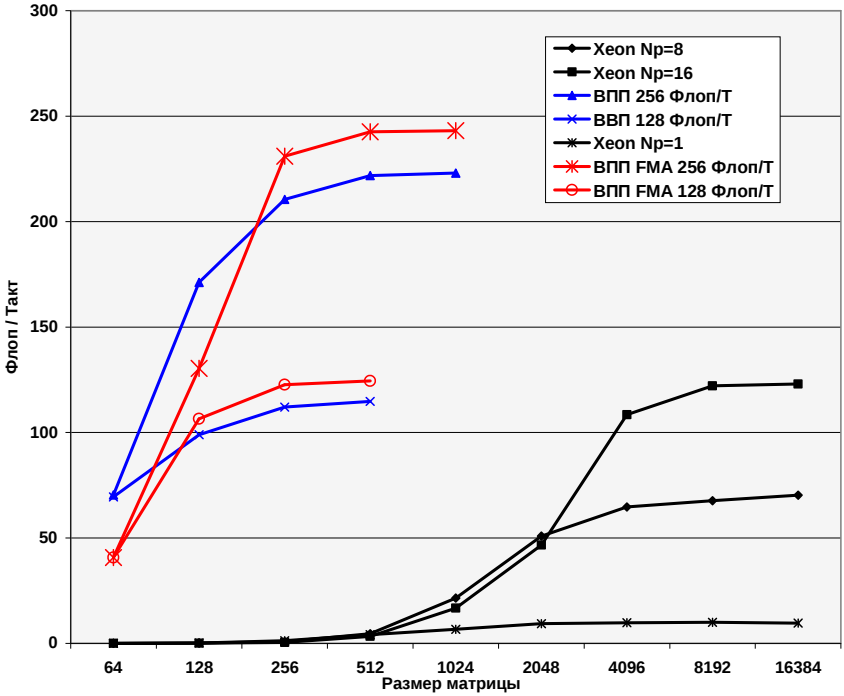


Рисунок 1. Производительность ВПП и Xeon ES-2690 на программе блочного умножения матриц в зависимости от размера матрицы

Как видно из рисунка 1, использование FMA увеличивает реальную производительность ВПП, достигая 97% и 95% эффективности при пиковой производительности 128 и 256 флоп в такт соответственно. Примерно ту же эффективность (96%) имеет и вычислительный узел, содержащий 16 процессорных ядер Intel Xeon ES-2690, при выполнении программы умножения матриц из библиотеки Intel MKL.

Различие в том, что в ВПП использовалась простая программа блочного умножения матриц с одним уровнем разбиения матрицы на блоки, а в узле на процессорах Intel Xeon — гораздо более сложная, библиотечная версия этой программы. В ней разбиение осуществляется

по нескольким уровням памяти, начиная с регистрового файла и до наиболее ёмкого кэша, и используются блоки разного размера.

Как уже отмечалось выше, на той же по сложности алгоритма программе блочного умножения матриц, что использовалась в ВПП, процессоры Intel Xeon имеют лишь 20% эффективность [1]. Другое преимущество ВПП, как было показано в [6], заключается в сохранении высокой производительности при уменьшении размера матрицы на этой программе.

3. Сравнение эффективности ВПП и Intel Skylake на программе битонной сортировки

Программа битонной сортировки является одной из самых быстрых параллельных программ сортировки, и часто используется в процессорах с параллельной обработкой данных, например в графических ускорителях [8]. На рисунке 2 приведена схема сети, поясняющая алгоритм битонной сортировки на примере сортировки массива, состоящего из двух векторов по 8 элементов каждый.

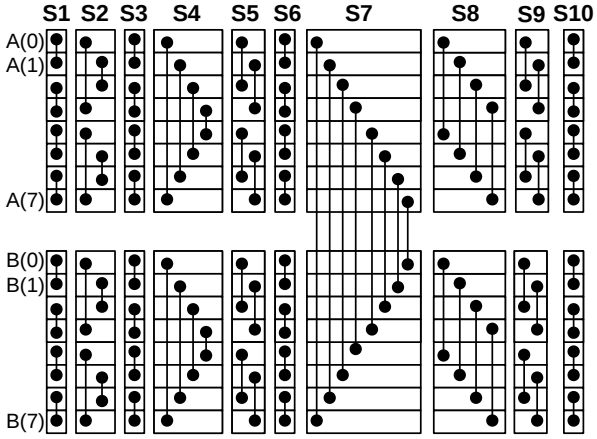


Рисунок 2. Сеть битонной сортировки двух векторов по 8 элементов каждый

Процесс сортировки состоит из последовательного прохождения ступеней S1 — S10. На каждой ступени разбитые на пары элементы массива сравниваются между собой, и производится их перестановка, так чтобы нижний из двух элементов, соединенных на рисунке 2 вертикальной линией, был больше верхнего. В результате на выходе ступени

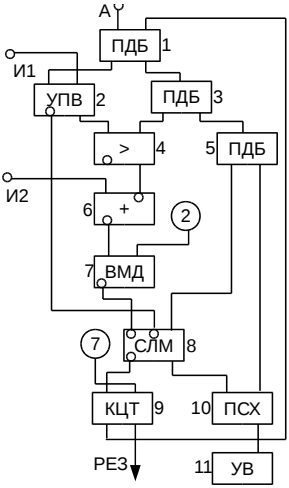
S1 получаем отсортированные фрагменты по 2 элемента, на выходе ступени S3 — по 4 элемента, на выходе ступени S6 — по 8 элементов. Далее можно продолжить сортировку упорядоченных по возрастанию элементов векторов A и B с выхода ступени S6, и на выходе ступени S10 получим отсортированный массив из 16 элементов, когда вектор A будет содержать упорядоченные по возрастанию минимальные элементы двух исходных векторов, а вектор B — максимальные. Заметим, что имеется и другой вариант битонной сортировки, когда в одних парах необходимо переставлять элементы так, чтобы верхний был больше нижнего, а в других — нижний был больше верхнего. Мы будем рассматривать вариант с одним направлением упорядочивания элементов в парах, как более простой и быстрый для процессоров с векторной обработкой данных. Именно такая сортировочная сеть, показанная на рисунке 2, использовалась в работе [9] для сортировки массивов небольшого размера на процессоре Intel Skylake с системой команд AVX-512.

Как видно из приведённой на рисунке 2 сортировочной сети, операции сравнения и перестановки в каждой паре элементов в пределах одной ступени можно выполнять независимо друг от друга, то есть с помощью векторных команд. Однако продвижение по ступеням сети выполняется последовательно, и параллелизм операций в этой программе растет прямо пропорционально размеру сортируемого массива N, в то время как в программе умножения матриц — пропорционально N^3. Применительно к ВПП параллелизм можно повысить, увеличивая число элементов N в пределах одного вектора, пока N не превысит аппаратную длину вектора (256 элементов), и далее пропорционально числу векторов в составном векторе-указателе сортируемого массива. В процессоре Intel Skylake аппаратная длина вектора значительно меньше, а именно 512 бит или 8 чисел с плавающей запятой двойной точности.

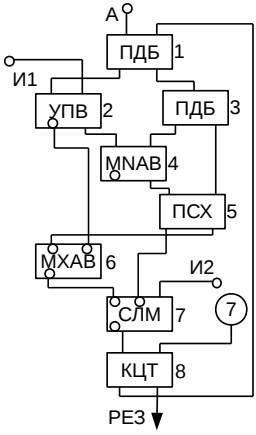
Поскольку число операций сравнения и перестановки в программе битонной сортировки при увеличении размера массива N растёт как $O(N \log^2 N)$, то эта программа эффективна лишь для малых значений N [8]. Так в [9] она использовалась только для сортировки массивов не более чем из 16 векторов по 8 чисел в каждом, поскольку для обработки большего числа векторов, по-видимому, не хватает имеющегося числа векторных регистров в регистровом файле процессора. Соответственно в [9] основные результаты, характеризующие производительность процессорного ядра Intel Skylake, приведены для программы Quicksort, которая выполняет рекурсивное разделение входного массива на блоки с возрастающими значениями элементов, но не упорядоченными в пределах блока.

Лишь после этого производится сортировка элементов в каждом блоке с помощью сортировочной сети, показанной на рисунке 2. У нас же использовалась программа битонной сортировки в диапазоне до 32-х векторов по 256 элементов, то есть до N=8192. Большая ёмкость ЛПВ в ВПП позволяет реализовать такую сортировку.

Целью данной работы являлось использование в ВПП той же самой программы битонной сортировки, что и в [9], лишь представленной в виде графа из команд ВПП. На рисунке 3 приведены два варианта графа программы битонной сортировки для N<=256, то есть в пределах одного вектора.



(*а*) с использованием команды векторного сравнения



(*б*) с использованием команд формирования вектора из минимальных и максимальных элементов двух векторов

Рисунок 3. Граф программы битонной сортировки

Исходный граф программы, приведённый на рисунке 3*а*, более подробное описание работы которого приведено в [10], был доработан для большего соответствия программе битонной сортировки из [9]. А именно, введены команды формирования вектора из минимальных (MNAB) и максимальных (MXAB) элементов двух сравниваемых векторов, которые есть в системе команд AVX-512 у последних процессоров Intel Xeon. Они были использованы в программе битонной сортировки, предложенной автором работы [9], и граф программы на рисунке 3*б* реализует в ВПП тот же самый алгоритм.

В исходной программе сравнение элементов исходного вектора и вектора результата команды «Упорядочить Вектор» (УПВ) 2 осуществляется командой векторного сравнения 4. Команда УПВ 2 упорядочивает вектор, поступивший на вход ступени (команду дублирования ПДБ 1), в соответствии с индексами у вектора с входа И1. Эти управляющие вектора с индексами формируются заранее для каждого номера ступени, начиная с S1 до S6, если длина сортируемого вектора N равна 8, как в Intel Skylake, и с S1 до S36 — для N=256 в ВПП.

Усовершенствованный граф, показанный на рисунке 3*б*, содержит меньшее число команд и требует меньшего времени для прохождения одной ступени сортировки. Команды MNAB 4 и MXAB 6 были добавлены в систему команд ВПП, как и в VHDL описание модели векторного арифметического ИУ, в котором они выполняются.

На рисунке 4 приведены значения производительности одного ядра ВПП и Intel Skylake при сортировке чисел с плавающей запятой в зависимости размера сортируемого массива.

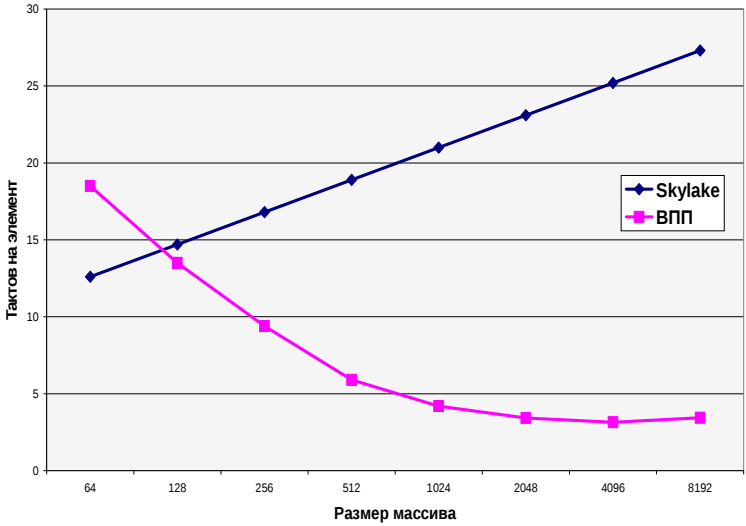


Рисунок 4. Производительность ВПП на программе битонной сортировки и Intel Skylake на программе Quicksort в зависимости от размера массива

Значения производительности получены из времени выполнения

данной программы на VHDL модели ВПП и времени выполнения программы Quicksort процессорным ядром Intel Skylake, результаты которого приведены в [9].

Как видно из рисунка 4, число тактов, затрачиваемых на сортировку одного элемента массива, у одного ядра Intel Skylake монотонно растёт с увеличением размера массива, а у ВПП — сначала падает и, достигнув минимума при сортировке 4096 элементов, тоже начинает расти. Такой характер зависимости производительности от размера массива у ВПП объясняется тем, что в графе программы на рисунке 3 справа все команды кроме ПДБ 3 выполняются последовательно. В таком режиме потоковый процессор значительно проигрывает по производительности процессору традиционной архитектуры, поскольку период следования команд определяется не только задержкой вычисления результата в ИУ, но ещё и временем прохождения токена по кольцу управления. (С выхода ИУ в АП и обратно.) В результате при малом размере сортируемого массива N даже одно векторное ИУ, обрабатывающее вектора страницами по 32 элемента в такт, работает далеко не с полной нагрузкой, а таких арифметических ИУ в ВПП четыре.

При увеличении размера массива N нагрузка арифметических ИУ растёт, и при одновременной сортировке 16 векторов по 256 элементов (N=4096) — приближается к полной. При этом максимальная производительность ВПП, равная 2,96 такта на элемент, в 8,5 раза превосходит производительность одного ядра Intel Skylake при сортировке массива из 4096 чисел с плавающей запятой. Несмотря на то, что пиковая производительность одного ядра ВПП выше, чем у Intel Skylake в 8 раз, отношение реальной производительности к пиковой на программе битонной сортировки у ВПП хотя и не на много, но выше. Здесь нужно учесть, что эффективность выполнения программы с увеличением производительности процессора падает.

Заключение

Результаты моделирования ВПП показали, что на программах блочного умножения матриц и битонной сортировки, несмотря на большую производительность процессорного ядра, эффективность выполнения этих программ выше, чем у процессоров традиционной архитектуры. Более высокая эффективность работы была показана ВПП и на программах 2D и 3D Stencil, причем разрыв в производительности по сравнению процессорным ядром Intel Xeon KNL достигал 30 раз [11]. Отметим, что в отличие от ранее рассмотренных программ,

ограничение производительности на программах 2D и 3D Stencil обусловлено недостаточной пропускной способностью памяти. Столь большой разрыв в производительности отчасти объясняется разной пропускной способностью оперативной памяти у процессорного ядра KNL и ПВ у ВПП, но и при одинаковой пропускной способности производительность ВПП выше почти в 3 раза [11].

Таким образом, на всех рассмотренных выше программах ВПП показал более высокое отношение реальной производительности к пиковой по сравнению с процессорами Intel Xeon.

Отметим ещё одно важное преимущество ВПП. Как известно универсальные многоядерные процессоры, в частности Intel Xeon, примерно в 3 раза уступают в производительности графическим ускорителям при одинаковой площади процессорного кристалла. Предварительная оценка площади процессорного кристалла, занимаемого одним ядром ВПП, показала, что при технологии 22 нм максимальная удельная производительность на программе умножения матриц достигается при пиковой производительности 384 флоп в такт и составляет 3,2 флоп/мм^2 в такт [12].

Ту же самую удельную производительность на программе умножения матриц имеют графические ускорители фирмы NVIDIA, изготовленные по той же технологии 22 нм. Заметим, что пиковая производительность ВПП в 1,7 раза меньше чем у графического ускорителя, но отношение реальной производительности к пиковой — во столько же раз выше. Таким образом, полученные результаты подтверждают целесообразность продолжения работ по практической реализации ВПП. В первую очередь, планируется доработка VHDL описания ВПП для синтеза его схемы на ПЛИС.

Список литературы

[1] М. С. Клинов, С. Ю. Лапшина, П. Н. Телегин, Б. М. Шабанов. «Особенности использования многоядерных процессоров в научных вычислениях», *Вестник УГАТУ*, **16**:6 (51) (2012), с. 25–31. ↑202,208

[2] Arvind, R. S. Nikhil. “Executing a program on the MIT tagged-token data-flow architecture”, *IEEE Transactions on Computers*, **39**:3, pp. 300–318. ↑203,204

[3] G. V. Papadopoulos, K. R. Traub. “Multithreading: A revisionist view of dataflow architectures”, *Proc. 18-th Ann. Symp. on Computer Architecture* (30 May 1991, Toronto, Canada), 1991, pp. 342–351. ↑204

[4] А. В. Климов, Н. Н. Левченко, А. С. Окунев, А. Л. Стемпковский. «Вопросы применения и реализации потоковой модели вычислений»,

Проблемы разработки перспективных микро- и наноэлектронных систем. Т. II, МЭС-2016 (3–7 октября 2016 года), ИППМ РАН, М., 2016, с. 100–106. ↑204

[5] Н. И. Дикарев, Б. М. Шабанов. «Векторный потоковый процессор», *Известия ТРТУ*, 2005, №10 (54), Тематический выпуск «Интеллектуальные и многопроцессорные системы», с. 80–85. ↑205,206

[6] Н. И. Дикарев, Б. М. Шабанов, А. С. Шмелёв. «Векторный потоковый процессор: оценка производительности», *Известия ЮФУ. Технические науки*, 2014, №12 (161), Тематический выпуск: Суперкомпьютерные технологии, с. 36–46. ↑206,208

[7] Н. И. Дикарев, Б. М. Шабанов, А. С. Шмелёв. «Использование «сдвоенного» умножителя и сумматора в векторном процессоре с архитектурой управления потоком данных», *Программные системы: теория и приложения*, **6**:4(27) (2015), с. 227–241. URL ↑207

[8] N. Satish, M. Harris, M. Garland. “Designing efficient sorting algorithms for manycore GPUs”, *2009 IEEE International Symposium on Parallel & Distributed Processing* (Rome, 23-29 May 2009), 2009, pp. 1-10. doi ↑208,209

[9] B. Bramas. “A Novel Hybrid Quicksort Algorithm Vectorized using AVX-512 on Intel Skylake”, *International Journal of Advanced Computer Science and Applications*, **8**:10 (2017), pp. 337–344. doi ↑209,210,212

[10] Н. И. Дикарев, Б. М. Шабанов, А. С. Шмелёв. «Быстрые алгоритмы сортировки для векторного потокового процессора», *Суперкомпьютерные технологии*, Материалы 5-й Всероссийской научно-технической конференции СКТ-2018. Т. 1 (17–22 сентября 2018 г., с. Дивноморское), Изд-во ЮФУ, Ростов-на-Дону–Таганрог, 2018, с. 87–91. ↑210

[11] Н. И. Дикарев, Б. М. Шабанов, А. С. Шмелёв. «Недостаточная пропускная способность памяти на программе Stencil: преимущество векторного потокового процессора», *Программные системы: теория и приложения*, **9**:4(39) (2018), с. 399–415. doi ↑212,213

[12] Н. И. Дикарев, Б. М. Шабанов, А. С. Шмелёв. «Выбор оптимальной производительности ядра векторного потокового процессора», *Суперкомпьютерные технологии*, Материалы 4-й Всероссийской научно-технической конференции СКТ-2016, в 2 т. Т. 1 (19–24 сентября 2016 г.), Изд-во ЮФУ, Ростов-на-Дону, 2016, с. 36–41. ↑213

Поступила в редакцию 19.11.2019
Переработана 20.12.2019
Опубликована 26.12.2019

Рекомендовал к публикации *д.ф.-м.н. Н. Н. Непейвода*

Пример ссылки на эту публикацию:

Н. И. Дикарев, Б. М. Шабанов, А. С. Шмелёв. «Мелкоструктурный параллелизм и более высокая производительность процессорного ядра: преимущества векторного потокового процессора». *Программные системы: теория и приложения*, 2019, **10**:4(43), с. 201–217.

10.25209/2079-3316-2019-10-4-201-217

http://psta.psiras.ru/read/psta2019_4_201-217.pdf

Об авторах:

Николай Иванович Дикарев

Кандидат технических наук, ведущий научный сотрудник МСЦ РАН. Научные интересы: высокопроизводительные вычислительные системы, архитектура управления потоком данных, векторные процессоры, параллельная и векторная обработка.

0000-0002-7857-3250

e-mail: nic@jscc.ru

Борис Михайлович Шабанов

Кандидат технических наук, доцент, лауреат Государственной премии Российской Федерации в области науки и техники. Директор МСЦ РАН.

0000-0002-5238-366X

e-mail: shabanov@jscc.ru

Александр Сергеевич Шмелёв

Научный сотрудник МСЦ РАН. Научные интересы: Суперкомпьютеры, архитектура управления потоком данных, векторные процессоры.

0000-0002-1941-7792

e-mail: guest8993@rambler.ru

CSCSTI 50.09.33, 50.33.04
UDC 004.272.25:004.272.44

Nikolay I. Dikarev, Boris M. Shabanov, Aleksandr S. Shmelev. *Fine-grained parallelism and higher core performance: advantages of vector dataflow processor.*

Abstract. Currently, the reserves in increasing the performance of modern processors are almost exhausted. The stagnation is evidenced by the absence of growth, both the clock frequency and the number of instructions executed per clock, which determine the scalar performance of the processor core. In vector dataflow processor under development, processor core performance looks increased up to 256 flops per clock, which is eight times higher than the latest Intel Xeon processors due to a higher fraction of vector execution. We show that that vector dataflow processor has a higher ratio of real performance to peak on programs such as bitonic sorting, matrix multiplication, and 2D Stencil compared to the best traditional architecture processors.

Key words and phrases: vector processor, dataflow architecture, shared-memory multiprocessor, performance evaluation.

2010 *Mathematics Subject Classification*: 65Y05; 68Q10, 08-04

References

[1] M. S. Klinov, S. Yu. Lapshina, P. N. Telegin, B. M. Shabanov. "Multicore processing features in scientific computing", *Vestnik UGATU*, **16**:6 (51) (2012), pp. 25–31 (in Russian). ↑202,208

[2] Arvind, R. S. Nikhil. "Executing a program on the MIT tagged-token data-flow architecture", *IEEE Transactions on Computers*, **39**:3, pp. 300–318. ↑203,204

[3] G. V. Papadopoulos, K. R. Traub. "Multithreading: A revisionist view of dataflow architectures", *Proc. 18-th Ann. Symp. on Computer Architecture* (30 May 1991, Toronto, Canada), 1991, pp. 342–351. ↑204

[4] A. V. Klimov, N. N. Levchenko, A. S. Okunev, A. L. Stempkovskiy. "The application and implementation issues of dataflow computing system", *Problemy razrabotki perspektivnykh mikro- i nanoelektronnykh sistem.* V. II, M·YeS-2016 (3–7 oktyabrya 2016 goda), IPPM RAN, M., 2016, pp. 100–106 (in Russian). ↑204

[5] N. I. Dikarev, B. M. Shabanov. "Vector DataFlow processor", *Izvestiya TRTU*, 2005, no.10 (54), Tematicheskiy vypusk "Intellektual'nyye i mnogoprotsessornyye sistemy", pp. 80–85 (in Russian). ↑205,206

[6] N. I. Dikarev, B. M. Shabanov, A. S. Shmelëv. "Vector DataFlow processor: performance evaluation", *Izvestiya YuFU. Tekhnicheskiye nauki*, 2014, no.12 (161), Tematicheskiy vypusk: Superkomp'yuternyye tekhnologii, pp. 36–46 (in Russian). ↑206,208

10.25209/2079-3316-2019-10-4-201-217

[7] N. I. Dikarev, B. M. Shabanov, A. S. Shmelëv. "Fused multiply-adders using in vector dataflow processor", *Program Systems: Theory and Applications*, **6**:4(27) (2015), pp. 227–241 (in Russian). ↑207

[8] N. Satish, M. Harris, M. Garland. "Designing efficient sorting algorithms for manycore GPUs", *2009 IEEE International Symposium on Parallel & Distributed Processing* (Rome, 23-29 May 2009), 2009, pp. 1-10. ↑208,209

[9] B. Bramas. "A Novel Hybrid Quicksort Algorithm Vectorized using AVX-512 on Intel Skylake", *International Journal of Advanced Computer Science and Applications*, **8**:10 (2017), pp. 337–344. ↑209,210,212

[10] N. I. Dikarev, B. M. Shabanov, A. S. Shmelëv. "Fast sorting algorithms for vector dataflow processor", *Superkomp'yuternyye tekhnologii*, Materialy 5-y Vserossiyskoy nauchno-tekhnicheskoy konferentsii SKT-2018. V. 1 (17–22 sentyabrya 2018 g., s. Divnomorskoye), Izd-vo YuFU, Rostov-na-Donu–Taganrog, 2018, pp. 87–91 (in Russian). ↑210

[11] N. I. Dikarev, B. M. Shabanov, A. S. Shmelëv. "Insufficient memory bandwidth on Stencil code: the advantage of a vector dataflow processor", *Program Systems: Theory and Applications*, **9**:4(39) (2018), pp. 399–415 (in Russian). ↑212,213

[12] N. I. Dikarev, B. M. Shabanov, A. S. Shmelëv. "Choosing optimal performance of vector dataflow processor core", *Superkomp'yuternyye tekhnologii*, Materialy 4-y Vserossiyskoy nauchno-tekhnicheskoy konferentsii SKT-2016, v 2 t. V. 1 (19–24 sentyabrya 2016 g.), Izd-vo YuFU, Rostov-na-Donu, 2016, pp. 36–41 (in Russian). ↑213

Sample citation of this publication:

Nikolay I. Dikarev, Boris M. Shabanov, Aleksandr S. Shmelev. "Fine-grained parallelism and higher core performance: advantages of vector dataflow processor". *Program Systems: Theory and Applications*, 2019, **10**:4(43), pp. 201–217. (*In Russian*). 10.25209/2079-3316-2019-10-4-201-217

http://psta.psiras.ru/read/psta2019_4_201-217.pdf