УДК 004.27

Использование «сдвоенного» умножителя и сумматора в векторном процессоре с архитектурой управления потоком данных [*]

Дикарев Н.И., к.т.н., E`mail: nic@jscc.ru
Шабанов Б.М., к.т.н., E`mail: shabanov@jscc.ru
Шмелёв А.С., E`mail: guest8993@rambler.ru
Межведомственный Суперкомпьютерный центр РАН, филиал ФГУ ФНЦ НИИСИ РАН, Москва

Аннотация. Процессор с архитектурой управления потоком данных может выполнять до 16 команд в такт по сравнению с 4 – 6 командами в такт у лучших процессоров фон-неймановской архитектуры. Моделирование векторного потокового процессора показало, что его производительность может быть доведена до 256 флоп в такт при выдаче менее 8 команд в такт, и поддерживаться близкой к пиковой производительности на программе перемножения матриц при значительно меньшем размере обрабатываемых матриц. Анализируются преимущества и недостатки использования в этом процессоре на векторной обработке конвейерного «сдвоенного» умножителя и сумматора вместо раздельных умножителей и сумматоров с плавающей запятой.

Введение

Продолжавшийся до последнего времени рост степени интеграции интегральных схем привёл к тому, что сейчас можно на одном кристалле реализовать до двенадцати суперскалярных процессоров (ядер), каждый из которых способен осуществлять поиск команд с готовыми операндами в окне из 100 команд и выполнять до 4 – 6 команд в такт. Такой темп выдачи команд не меняется уже в течение 13 лет, и 10 лет назад тактовая частота процессора перестала расти, т.е. производительность скалярной обработки одного процессорного ядра достигла своего предела. Поэтому дальнейшее увеличение производительности суперЭВМ достигается в основном за счет роста числа процессорных ядер в суперЭВМ, число которых сейчас достигает одного миллиона. Это накладывает серьёзные ограничения на класс задач, которые могут эффективно использовать столь большое число процессоров, поскольку эффект от распараллеливания по процессорам в суперЭВМ проявляется лишь при наличии параллелизма крупных программных блоков, в каждом из которых должно выполняться не менее 100 тысяч команд. Цель данной работы - показать, что разрабатываемый в МСЦ РАН процессор с архитектурой управления потоком данных (потоковый процессор) может обеспечить не только значительно более высокую производительность по отношению к лучшим процессорам традиционной архитектуры, но и поддерживать ее при работе с мелкоструктурным параллелизмом. Кроме того, будет рассмотрен один из возможных способов повышения производительности разрабатываемого векторного потокового процессора (ВПП), заключающийся в использовании на векторной обработке «сдвоенного» умножителя и сумматора с плавающей запятой.

Принцип работы потокового процессора и препятствия на пути его создания

В потоковом процессоре вне зависимости от места двухвходовой команды в графе программы она выдаётся на исполнение по прибытию на вход последнего из пары токенов операндов с одинаковым контекстом. После вычисления результата в исполнительном устройстве (ИУ) новые токены со значением результата отправляются

[*] Работа выполнена при частичной поддержке гранта РФФИ 13-07-00792.

на входы последующих команд согласно графу программы, а использованные токены операндов уничтожаются. Такой принцип работы позволяет исключить конфликты информационной зависимости – основную причину ограничения производительности процессора традиционной архитектуры, в котором до выдачи команды на выполнение нужно проверить записаны ли в память (регистровый файл) результаты предыдущих команд, которые должна читать в качестве операндов выдаваемая команда. Кроме того, в потоковом процессоре в отличие от фон-неймановского отсутствует центральное устройство управления (счетчик команд), и параллелизм выполняемых команд определяется в динамике по приходу операндов на входы команд в децентрализованной схеме, что также повышает его производительность.

На рис. 1 показана структурная схема потокового процессора, в которой устройство поиска готовых к выполнению команд – память поиска пар (ППП) готовых операндов выполнена в виде *K* работающих в параллель модулей, обеспечивающих работой такое же число ИУ. Для этого требуется лишь задать распределение команд из графа программы по модулям ППП, например, использовать младшие разряды номера команды для задания номера модуля, в котором осуществляется поиск. Однако, чаще всего, контекст токена содержит помимо номера команды в графе программы еще несколько полей, например, в разрабатываемом ВПП, это поля индекса, номера итерации и запуска процедуры, которые также должны совпадать у команд, выдаваемых на выполнение из ППП. Это дает возможность одновременного выполнения различных итераций вложенных циклов и различных запусков процедур, создавая для каждой выполняемой параллельно команды копию узла в графе программы. Недостаток такого решения в существенном усложнении реализации ППП, поскольку она должна выполнять ассоциативный поиск. С другой стороны, дополнительные поля в контексте токена дают возможность использовать их ещё и для задания номера модуля ППП, что делает более равномерным распределение токенов по модулям ППП.

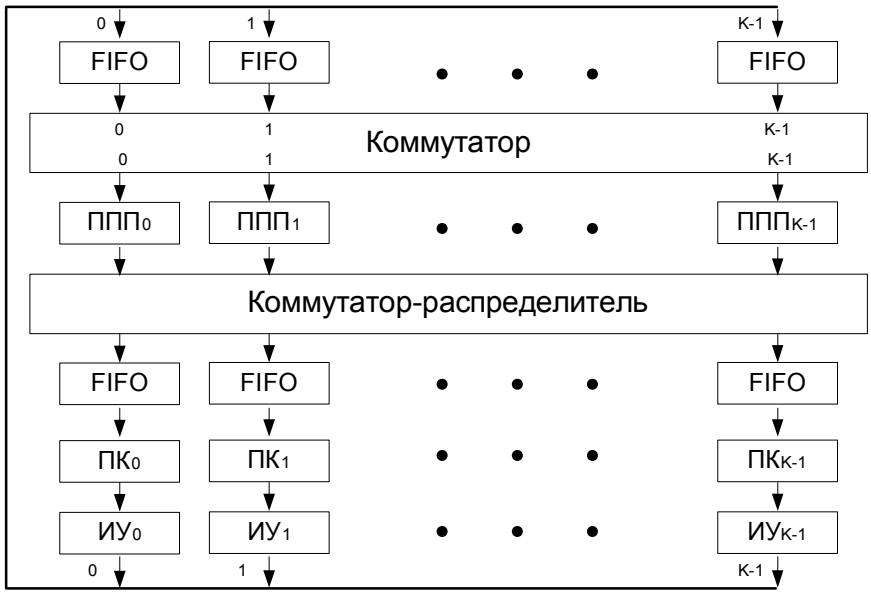

Рис. 1. Структурная схема потокового процессора

Токены со значением результата с выхода любого ИУ передаются в схеме на рис. 1 на вход нужного модуля ППП с помощью коммутатора. Буферы FIFO, установленные на входах коммутатора, предотвращают блокировку выхода ИУ в случае конфликта за выход коммутатора, когда в одном такте несколько ИУ посылают токены в один и тот

же модуль ППП. В этом случае на выход коммутатора проходит токен с одного из входов (с более высоким приоритетом), а остальные сохраняются в буфере FIFO на входе коммутатора для прохождения в последующие такты. Тем самым буферы FIFO используются для сглаживания неравномерности распределения токенов по модулям ППП во времени. Приход первого токена операнда у команды с двумя входами приводит к его записи в свободную ячейку ППП. По приходу второго токена операнда команда выдается на исполнение в ИУ, а ячейка ППП, занятая первым операндом освобождается. Тогда готовая команда в виде пакета из двух значений операндов вместе с контекстом передается с выхода ППП через коммутатор-распределитель на вход того ИУ, где она будет выполняться, предварительно проходя через буфер FIFO и модуль памяти команд (ПК). ПК - это обычное линейно адресуемое запоминающие устройство, из которого по номеру команды в графе читается её код операции и информация о том, сколько токенов со значением результата нужно сформировать и на входы каких команд их направить. Назначение буферов FIFO то же, что и аналогичных буферов на входе коммутатора - они предотвращают блокировку выхода коммутатора-распределителя, если ИУ занято обслуживанием предыдущей команды. Буферы FIFO должны быть достаточной ёмкости, поскольку переполнение любого из них из-за кольцевой структуры управления (см. рис. 1) с высокой вероятностью приводит к блокировке всех устройств в этой цепи, и процессор попадает в неработоспособное состояние.

Потоковый процессор потенциально позволяет достичь во много раз более высокую производительность по сравнению с процессором традиционной архитектуры. Проекты по разработке такого процессора проводились в США, Японии, Великобритании и ряде других стран с конца 70-х до середины 90-х годов прошлого века [1,2,3]. Однако ни один из них не был успешным из-за сложности практической реализации потокового процессора. Отметим несколько главных причин почему при сопоставимых аппаратных затратах не удалось получить более высокую производительность у потокового процессора по сравнению с процессором традиционной архитектуры. Во-первых, это сложность реализации ППП, которая должна обладать большой ёмкостью, поскольку её переполнение недопустимо, высоким быстродействием и осуществлять ассоциативный поиск. Вторая причина связана с использованием коммутатора, с помощью которого осуществляется пересылка токенов с выходов каждого из K ИУ на входы любого из K модулей ППП. Такой коммутатор должен иметь высокую пропускную способность по каждому из входов и одновременно малую задержку передачи до выхода, что трудно выполнимо уже при $K>16$. Если же учесть, что в потоковом процессоре приходится выполнять в 2 - 3 раза больше команд на программах научных задач по сравнению с фон-неймановским процессором [4], то оказывается, что его максимальное преимущество в реальной производительности не превышает 2 раз.

Заметим, что модули ППП, коммутатор, коммутатор-распределитель, буферы FIFO и модули ПК в схеме на рис. 1 должны иметь высокое быстродействие, поскольку вносимая ими задержка при прохождении токена по кольцу управления добавляется к времени выполнения команды в ИУ, определяя тем самым время выполнения команды в цепочке из последовательных команд. Поэтому на чисто последовательном коде потоковый процессор будет заведомо проигрывать по производительности фон-неймановскому процессору, и более высокая производительность может быть получена лишь при наличии существенного параллелизма у выполняемых в программе команд. Однако нельзя допускать и появления избыточного параллелизма, поскольку переполнение буферов готовых команд на входах ИУ или модулей ППП приводит процессор в тупиковую ситуацию (deadlock). Данная проблема характерна для многих научных задач, в которых при увеличении размера обрабатываемых массивов естественный параллелизм может быть столь велик, что обрабатывать все элементы

массивов в параллель невозможно, так как для этого не хватит ресурсов ни в какой реальной системе [5]. Поэтому в потоковый процессор необходимо вводить программное и (или) аппаратное ограничение параллелизма.

Векторный потоковый процессор

В разрабатываемом в МСЦ РАН векторном потоковом процессоре ППП не используется для хранения элементов массивов данных, они хранятся в обычной линейно-адресуемой памяти - памяти векторов (ПВ), что позволяет существенно снизить требования к ёмкости ППП. Векторная обработка позволяет в VL раз сократить число выполняемых команд на векторизуемой части программы, где VL - длина вектора, и в ВПП при VL=256 требования к ёмкости ППП снижены в сотни раз. Причем степень векторизации программ в ВПП выше, чем в векторных процессорах традиционной архитектуры за счет хранения массивов данных в ПВ в виде «векторов-указателей», то есть векторов, элементами которых являются указатели векторов подмассивов. Такое хранение массивов позволяет одной командой выдать указатели всех векторов строк, содержащимися в векторе-указателе матрицы, и выполнить над всеми строками матрицы одинаковые векторные операции. Тем самым удается векторизовать не один, а два вложенных цикла программы, и значительно сократить адресные и другие вычисления, выполняемые в скалярных ИУ (СИУ) ВПП, и соответственно повысить производительность векторных ИУ (ВИУ) относительно СИУ. В среднем число выполняемых команд в ВПП даже на чисто скалярном коде в 2 – 3 раза меньше по сравнению с процессорами традиционной архитектуры [6], что позволяет снизить суммарную ёмкость модулей ППП до 16К команд и повысить её быстродействие. Использование векторной обработки повышает производительность ВПП до 256 флоп в такт и делает структуру ВПП регулярной, с малой средней длиной линий связи между элементами на процессорном кристалле, что должно обеспечить малое потребление мощности при достаточно высокой тактовой частоте. Как показывают предварительные расчеты при реализации на СБИС, ВПП способен обеспечить ту же производительность, что имеют современный высокопроизводительный процессор вместе с ускорителем [7], и в отличие от них способен сохранять высокую производительность на программах с мелкоструктурным и нерегулярным параллелизмом [8].

Высокая производительность векторной обработки в ВПП, анализ работы которого приведен в [8], достигается за счет реализации ВИУ в виде 32 идентичных колец (lanes) обработки, что позволяет обрабатывать вектор страницами по 32 элемента в такт. Каждое из этих колец содержит 4 конвейерных сумматора и 4 умножителя с плавающей запятой, одновременная работа которых обеспечивает пиковую производительность ВПП, равную 256 флоп в такт. Расслоение находящейся на процессорном кристалле в каждом кольце локальной памяти векторов (ЛПВ) на шестнадцать двухпортовых банков даёт возможность доступа к ЛПВ через 32 порта. Векторное АЛУ из четырех сумматоров и четырех умножителей задействуют 24 порта, ещё через три порта к ЛПВ обращается блок выполнения специальных операций (БВСО), то есть используются 27 портов из 32 возможных. БВСО может работать одновременно с векторным АЛУ, выполняя команды чтения и записи одиночных элементов вектора, команды редукции, среди которых вычисление суммы элементов вектора и поиска максимального (минимального) элемента, а также команды сборки – разбрасывания элементов вектора. Наконец, как векторное АЛУ, так и БВСО могут читать и записывать элементы вектора не только в быстродействующей ЛПВ, но и в ПВ большой ёмкости, реализованной на микросхемах динамической памяти, через, находящийся на процессорном кристалле, контроллер ПВ.

На рис. 2 показаны зависимости производительности ВПП на программе перемножения матриц от их размера, полученные на VHDL модели ВПП уровня регистровых станций и

приведенные в [8]. Для сравнения там же приведены аналогичные результаты для вычислительного узла на микропроцессорах Intel Xeon E5 при выполнении той же программы из библиотеки Intel MKL на 1, 8 и 16 процессорных ядрах, из которых следует, что ВПП обеспечивает значительно более высокую производительность при уменьшении размера обрабатываемых матриц. Это достигается за счет отсутствия накладных расходов на создание потоков, синхронизацию и обмен данными между ними в многопроцессорной системе, поскольку ВВП является однопроцессорной системой, хотя и имеет более высокую производительность. Но даже при выполнении программы перемножения матриц на одном процессорном ядре Intel Xeon E5 (см. график Np=1 на рис.2) его производительность падает до половины от пиковой производительности при уменьшении размера матрицы до $N_{1/2}$=650, в то время как ВВП сохраняет высокую производительность при значительно меньшем размере матрицы. Для сравнения $N_{1/2}$ ВВП с пиковой производительностью 128 и 256 флоп в такт составляет примерно 60 и 100 [8]. Такая разница объясняется эффектом «холодного КЭШа» у процессора традиционной архитектуры, когда при уменьшении объема вычислений в выполняемой программе процессор начинает резко терять производительность, в первую очередь, из-за недостаточного накопления данных в КЭШах, что приводит к частым промахам и падению производительности.

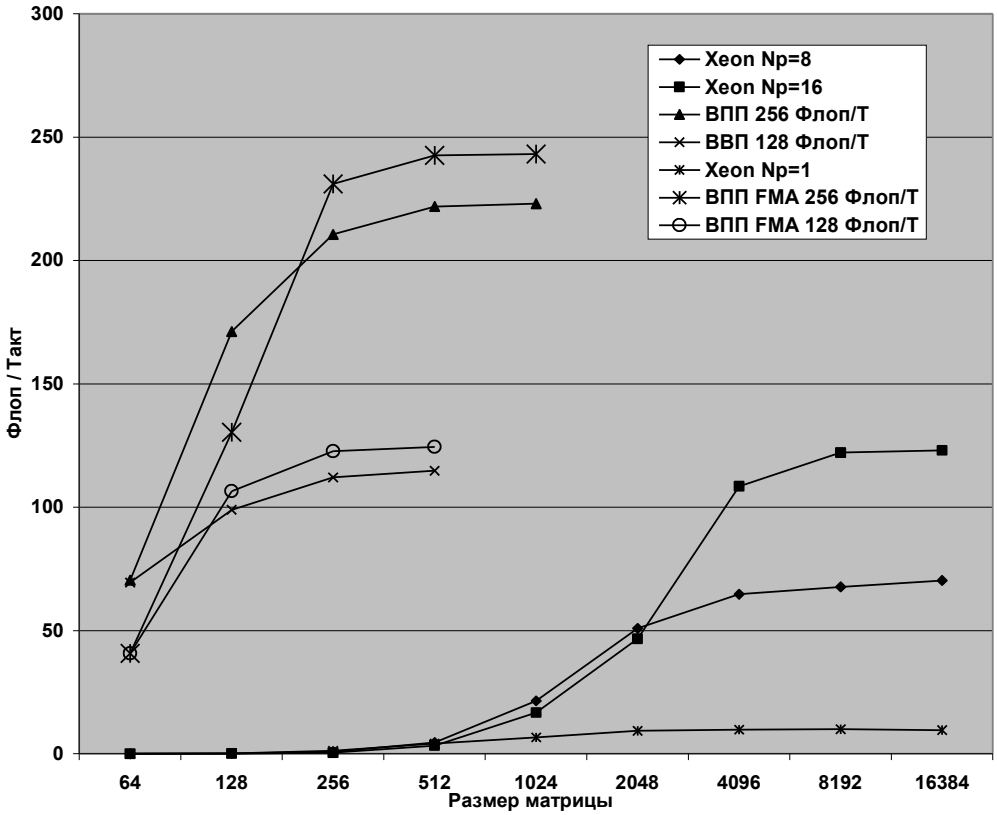

Рис. 2. Зависимость производительности ВПП от размера матрицы

В то же время из приведенных на рис. 2 зависимостей видно, что при одной и той же пиковой производительности, равной 128 флоп в такт, реальная производительность ВПП на программе перемножения матриц составляет 89.6% от пиковой, что ниже в сравнении с 95% у системы на микропроцессорах. Причем с ростом пиковой производительности ВПП до 256 флоп в такт его эффективность падает до 86.8%.

Использование сдвоенных АЛУ в модели ВПП

Для повышения эффективности работы ВПП была проведена оптимизация его модели, заключающаяся в использовании на векторной обработке вместо раздельных умножителя и сумматора одного «сдвоенного» умножителя и сумматора с плавающей запятой (FMA), выполняющего две операции с плавающей запятой в такт по вычислению функцию A*B+C. При использовании четырех FMA в каждом из 32 колец обработки элементов вектора в ВПП число используемых портов доступа к банкам ЛПВ в кольце уменьшается с 24 до 16, что приводит к снижению конфликтов занятого банка в расслоенной ЛПВ. В результате производительность разрабатываемого векторного процессора на программе перемножения матриц удалось повысить с 222.2 флоп в такт до 241.4 флоп в такт или до 95% от пиковой производительности процессора, равной 256 флоп в такт (см. графики ВПП FMA на рис.2).

Впервые «сдвоенный» умножитель и сумматор (Fused Multiply-Add) был применен в процессоре IBM Power1, поскольку такая комбинация двух операций с плавающей запятой часто встречается в реальных программах, и реализация в виде одного ИУ имеет следующие преимущества по сравнению с раздельными сумматором и умножителем [9]. Время вычисления результата в FMA меньше, а точность выше, так как отсутствует ступень округления промежуточного результата при вычислении той же функции в цепочке из двух последовательных ИУ. Кроме того, требуется меньшее число портов в регистровом файле, и соответственно меньше аппаратные затраты в регистровом файле и схемах управления ИУ, работающих с ним.

Заметим, что площадь, занимаемая на процессорном кристалле регистровым файлом, растет пропорционально его ёмкости и квадратично от числа портов [10]. Поскольку при переходе на FMA за счет уменьшения числа портов в регистровом файле до полутора раз можно снизить площадь регистрового файла для чисел с плавающей запятой до двух раз, то «сдвоенные» умножители и сумматоры используются практически во всех современных высокопроизводительных микропроцессорах и ускорителях. Другая не менее важная причина их использования – возможность, не увеличивая числа выдаваемых в такт команд обработки чисел с плавающей запятой вдвое повысить производительность процессора.

В разрабатываемом ВПП регистровые файлы не используются из-за их слишком малой ёмкости, и основным запоминающим устройством, обеспечивающим высокую пропускную способность для работы большого числа ИУ, не только на векторной, но и на скалярной обработке, является ЛПВ. Эта расположенная на процессорном кристалле память имеет высокую пропускную способность и большую ёмкость за счет расслоения на 512 банков, которые образуют двухуровневую структуру. Верхний уровень составляют 32 идентичных кольца, групповая синхронная работа которых позволяет читать и записывать вектора страницами по 32 элемента в такт, и лишь БВСО выполняет одиночные обращения для доступа к произвольным элементам вектора на скалярной обработке. На нижнем уровне (в каждом кольце) ЛПВ состоит из 16 двухпортовых банков, что, как уже упоминалось выше, позволяет через 24 порта читать операнды и записывать результаты нескольким ИУ векторного АЛУ, и ещё три порта выделить для БВСО. В отличие от регистровых файлов, которые имеют ёмкость не более 128 слов, но позволяют без конфликтов обращаться к любому слову или короткому вектору со временем выборки в один такт, расслоенная память имеет большую ёмкость (и время выборки), причем доступ через независимые порты может сопровождаться конфликтами занятого банка (порта). Увеличение числа портов доступа к ЛПВ в каждом кольце так же приводит к квадратичному росту аппаратных затрат в матричном коммутаторе, с помощью которого входы и выходу ИУ подключаются к банкам ЛПВ. Соответственно

при переходе на FMA за счет уменьшения суммарного числа портов так же можно существенно снизить аппаратные затраты в матричном коммутаторе и, кроме того, как было отмечено выше, повысить производительность векторной обработки за счет снижения числа конфликтов занятого банка в ЛПВ.

Помимо снижения аппаратных затрат и увеличения пропускной способности ЛПВ, переход на FMA дает возможность существенно уменьшить число векторов, хранимых в ЛПВ, поскольку результат векторного умножения уже не требуется записывать в ЛПВ и затем читать для операции сложения, как при раздельных командах умножения и сложения. Как показано в [8], число таких векторов, произведённых умножителем и ожидающих в ЛПВ использования с последующим уничтожением сумматором, на программе перемножения матриц в ВПП может достигать 3000, что при длине вектора 256 слов по 8 байт составляет более 6 Мбайт, то есть практически всю доступную ёмкость ЛПВ. Возможность столь существенной экономии ресурса ЛПВ при использовании «сдвоенного» умножителя и сумматора также являлась одной из целей перехода на FMA. Однако FMA является командой с тремя операндами вместо двух, на которые рассчитана работа ППП в разрабатываемом процессоре. Поэтому переход на FMA требовал существенной доработки схемы ППП вместе с увеличением её ёмкости в полтора раза, поскольку вместо хранения одного операнда (слова данных 8 байт и контекста 8 байт) до прихода второго операнда у двухвходовых команд, теперь требовалось хранить в ППП два операнда до прихода третьего. Чтобы не проводить столь существенной доработки схемы ППП, было решено воспользоваться тем, что указатель вектора, состоящий из адреса вектора в ПВ или ЛПВ и длины вектора, содержит значительно меньше разрядов, чем 8 байт, необходимых для числа с плавающей запятой двойной точности. Тогда в одно слово данных можно записать два адреса вектора по 28 бит и значение VL – 8 бит. Такую операцию выполняет введённая в систему команд ВВП новая команда упаковки (ПАК), которая выполняется в СИУ и упаковывает два указателя вектора с первого и второго входов команды в поле данных токена, посылаемого с выхода этой команды на первый вход команды FMA в графе программы. На второй вход этой команды можно подать либо ещё один указатель вектора, либо скалярное значение, которое может быть одним из операндов векторной команды. Таким образом, команда FMA с тремя операндами реализуется в ВПП как последовательность из двух команд ПАК и FMA, каждая из которых имеет 2 входа, причем последняя из них при выполнении в ВИУ имеет всю необходимую информацию для чтения трех векторов операндов либо двух векторов и скаляра.

Результаты выполнения программы перемножения матриц в ВПП при такой реализации FMA, подтверждающие повышение производительности, были приведены выше (см. рис. 2). Следует лишь отметить, что рост производительности в ВПП при использовании FMA сопровождался значительным увеличением числа команд, ожидающих в ППП прихода парного операнда, то есть требуется ППП значительно большей ёмкости. Как было объяснено в [8], наличие зависимости по данным во внутреннем цикле графа программы перемножения матриц для ВПП, когда вектор, созданный сумматором в предыдущей итерации цикла, используется в качестве его операнда в следующей итерации, проявляется в большом времени цикла у команд, выполняемых сумматором. Это следствие известного недостатка потоковой архитектуры, заключающегося в низкой производительности при выполнении последовательных команд. В ВПП этот недостаток компенсируется возможностью осуществлять поиск готовых команд в гораздо большем окне (в 100 раз) по сравнению с лучшими процессорами традиционной архитектуры, что позволяет не останавливать работу умножителя, поставляющего вектора операнды на другой вход сумматора. В результате сумматор начинает выполнять команды из нескольких итераций внешнего цикла, что устраняет простои в его работе и

производительность ВПП приближается к пиковой. При использовании отдельных устройств умножителя и сумматора число команд, ожидающих парного операнда у сумматора в ВПП, растёт пропорционально его производительности, и при производительности ВПП равной 256 флоп в такт требуемая ёмкость ППП достигает 3000 токенов. Поскольку эти токены в программе перемножения матриц представляют собой указатели векторов, созданных умножителем, то и требуемая ёмкость ЛПВ составляет 3000 векторов.

Заключение

С переходом на описанную выше реализацию FMA число векторов в ЛПВ удаётся снизить до 600, однако число токенов в ППП увеличивается до 15000, что близко планируемой ёмкости ППП в 16К токенов. Причины такого «сверхнормативного» роста числа токенов в ППП будут выясняться в ходе дальнейших исследований, что позволит выработать меры по уменьшению числа токенов в ППП и провести очередную оптимизацию схемы (VHDL модели) ВПП.

Литература

1. Arvind and R.S.Nikhil, Executing a program on the MIT tagged-token data-flow architecture. -IEEE Trans. Comput., vol. C-39, 1990, N 3, pp.300-318.
2. Manchester Dataflow Research Project. http://cnc.cs.manchester.ac.uk/progects/dataflow.html
3. Sakai S. et al, An Architecture of a Dataflow Single-Chip Processor. -Proc. 16-th Ann. Symp. on Computer Architecture, 1989, pp. 252-273.
4. G.V.Papadopoulos, K.R.Traub, Multithreading: A revisionist view of dataflow architectures. -Proc. 18-th Ann. Symp. on Computer Architecture, 1991, pp.342-351.
5. D.E.Culler and Arvind, Resource requirements of dataflow programs. -Proc. 15-th Ann. Symp. on Computer Architecture, 1988, pp.141-150.
6. Дикарев Н.И., Шабанов Б.М. Скалярная обработка в процессоре с архитектурой управления потоком данных // Труды международной конференции "Информационные технологии в науке, социологии, экономике и бизнесе (осенняя сессия)", Украина, Ялта-Гурзуф, 30 сентября – 8 октября 2008 года, с.147-149.
7. Дикарев Н.И., Шабанов Б.М., Шмелёв А.С. Проблемы масштабирования производительности в векторном процессоре с архитектурой управления потоком данных // Суперкомпьютерные технологии СКТ-2012. Материалы 2-й Всероссийской научно-технической конференции, 24 - 29 сентября 2012, Дивноморское, Геленджик. – Ростов-на-Дону: изд-во ЮФУ, 2012, с.34-38.
8. Дикарев Н.И., Шабанов Б.М., Шмелёв А.С. Векторный потоковый процессор: оценка производительности // Известия ЮФУ. Технические науки. Тематический выпуск: Суперкомпьютерные технологии. - Ростов-на-Дону: изд-во ЮФУ, 2014. №12 (161), с. 36-46.
9. R.K. Montoye et al, Design of the IBM RISC System/6000 floating-point execution unit. -IBM Journal of Research and Development, vol. 34, pp 59-70, 1990.
10. C.R. Jesshope, Multi-Threaded Microprocessor – Evolution or Revolution // Proc. ACSAC 2003: Advances in Computer Systems Architecture, pp. 21-45.