АППАРАТНАЯ РЕАЛИЗАЦИЯ АРИФМЕТИКИ ПОВЫШЕННОЙ ТОЧНОСТИ В FPGA КАК ПОПЫТКА ВЗГЛЯДА НА ЭКЗАФЛОПС С ДРУГОЙ СТОРОНЫ

С.С. Андреев, С.А. Дбар, А.О. Лацис, Е.А. Плоткина, И.В. Простов

Последние несколько лет ознаменовались интенсивнейшими поисками новых суперкомпьютерных архитектур. Особую значимость этим поискам придает экзафлопсная проблема. Не секрет, что в рамках не только традиционных архитектур 10-15-летней давности, но и реально используемых сегодня гибридных архитектур на базе GPGPU экзафлопс все еще «не срастается», в частности, не укладывается в самопоставленное ограничение потребляемой мощности 20 мегаваттами [1]. Возникает настоятельная потребность в прорывном повышении эффективности на архитектурном уровне.

Хорошо известно, что проблема суперкомпьютерной архитектуры – это проблема коммуникаций, перемещения данных между вычислительными устройствами и системой памяти. Например, важнейшим недостатком традиционной фоннеймановской архитектуры является доминирование времени выполнения вспомогательных операций над временем полезных вычислений [2]. Вспомогательные операции естественно рассматривать как операции по доставке данных в вычислительное устройство из системы памяти и обратно, то есть как частный случай коммуникаций. Недостатком архитектуры в данном случае является тот факт, что соотношение вычислительной и коммуникационной работы не выгодное: слишком малый объем вычислений обслуживается слишком большим объемом коммуникаций. Поскольку этот недостаток наблюдается уже на микро-уровне, при выполнении отдельных процессорных команд, изменение упомянутого соотношения возможно только путем изменения архитектуры, путем создания вычислителя, в котором уже на микро-уровне коммуникации могут быть организованы более эффективно. Например, вместо фоннеймановского процессора можно использовать GPGPU, с его гетерогенной, многоуровневой системой памяти, явно управляемой из программы.

Соотношение между вычислительной и коммуникационной работой можно сместить в нужную сторону не только путем уменьшения объема коммуникаций, но и путем увеличения объема вычислений. При использовании таких приемов, как, например, блочные методы вычислительной линейной алгебры [3], ровно это и происходит – программист стремится выполнить максимально возможный объем вычислений с данными, уже попавшими в небольшую, но быструю память. К сожалению, на микро-уровне этот прием не проходит. Арифметическая операция есть просто арифметическая операция, «увеличить» ее, выполнив «максимально возможный объем вычислений» с данными, уже попавшими на внутренние регистры арифметического устройства, вроде бы не представляется возможным.

Тем не менее, представим себе на минуту, что невозможное осуществилось. Это соответствовало бы ситуации 60-70 годов прошлого столетия, когда время выполнения операции с вещественными числами в процессорах многократно превышало время выполнения любой вспомогательной операции. Проблемы доминирования времени выполнения вспомогательных операций, грубо говоря, еще не существовало [2]. Многие и многие сегодняшние проблемы, вынуждающие нас не только мучительно придумывать новые архитектуры, но и гораздо более мучительно заставлять прикладных программистов их осваивать, не то чтобы решались легко, а просто не возникали. Соотношение между вычислительной и коммуникационной работой было надежно и сильно смещено в правильную сторону уже на самом низком уровне – на уровне выполнения отдельных арифметических команд. Выгодность упомянутого соотношения достигалась сама по себе, безо всяких специальных усилий системных архитекторов и прикладных программистов. Конечно, зачатки сегодняшних проблем постепенно возникали уже тогда, но всякие усилия по их разрешению, вроде использования блочных методов, давали полезный результат, образно говоря, отсчитываемый от гораздо более высокой планки.

«Возврат в золотой век» редко бывает возможным, но в данном случае он не столько возможен, сколько неизбежен. Приведенное двумя абзацами выше утверждение о невозможности «увеличить» арифметическую операцию является намеренно провокационным, и неверно по существу. Арифметическая операция НЕ есть «просто арифметическая операция». Она есть операция, выполняемая арифметическим устройством с некоторой точностью (разрядностью). При увеличении разрядности она, вполне естественным образом, «увеличивается», причем гораздо быстрее, чем объем обеспечивающих ее коммуникаций.

Потребности суперкомпьютерных приложений в вычислениях повышенной точности имеют, как минимум, два источника.

С одой стороны, применение повышенной точности иногда позволяет свести решаемую вычислительную задачу к меньшему количеству арифметических операций. Например, ускорить сходимость, снизить порядок разностной схемы, или вообще воспользоваться более экономным методом, который при обычной точности не дал бы правильного результата. Сегодня такие методы не используются, поскольку вычисления с повышенной точностью сами по себе очень дороги в современных процессорах, но это не значит, что таких методов нет.

С другой стороны, что гораздо важнее, при использовании любых, самых традиционных, вычислительных методов, потребность в точности вычислений часто растет по мере роста объема обрабатываемых данных. Например, при решении сеточными методами задач механики сплошной среды требуемая точность зависит, как минимум, линейно от размера матрицы СЛАУ, к решению которой сводится сеточная задача, то есть от числа ячеек сетки. За годы, прошедшие с момента возникновения и стабилизации общепринятых представлений о «традиционной, обычной» точности вычислений, типичные размеры обрабатываемых сеток уже выросли на 6-7 порядков. Перспективные, экзафлопсные в том числе, системы строятся, не в последнюю очередь, для еще большего увеличения размера обрабатываемых сеток, и речь снова будет идти о порядках. От некоторых пользователей наших машин в ИПМ им. М. В. Келдыша РАН приходилось слышать мнение, что результатам многих современных расчетов на суперкомпьютерах просто нельзя доверять, поскольку для используемых в них сеток стандартной точности double, скорее всего, уже не достаточно.

Наилучшим подтверждением того, что вычисления с повышенной точностью действительно нужны, мы считаем тот факт, что многие наши пользователи используют их уже сегодня. Применяется либо «вынесенный порядок», либо специальные библиотеки [4], реализующие повышенную точность программно. Каковы при этом потери быстродействия? Наши измерения показывают, что переход от аппаратно реализованной точности double к программно реализованной точности quadruple приводит к замедлению примерно в 40-50 раз. А ведь некоторым приложениям мало и четверной точности!

«Возврат в золотой век» можно считать состоявшимся. Обнаружен мощный пласт приложений, для которых соотношение между вычислениями и коммуникациями сильно смещено (или может быть смещено) в сторону вычислений. Для системного архитектора эта ситуация исключительно хороша тем, что ее легко улучшать. Не требуется сверхъестественных усилий по архитектурному снижению объема коммуникаций – достаточно просто ускорить вычисления, поскольку в затратах времени вновь доминируют именно они. Какое отношение все это имеет к экзафлопсной проблематике? Самое прямое.

Экзафлопсные машины почти автоматически потребуют повышенной точности вычислений, просто в силу размеров тех расчетных сеток, которые будет иметь смысл на них обрабатывать. Попытки практического использования таких машин приведут, как мы только что показали, к автоматическому падению **реальной** производительности, **измеренной в реально используемых арифметических операциях**, в те самые 40-50 раз. Не проще ли просто сразу построить машину, **всего лишь петафлопсную, но на повышенной точности**? Эта машина будет решать те же задачи (и за то же самое время), для решения которых с использованием стандартной точности понадобилась бы экзафлопсная производительность. По существу, это и будет экзафлопсная машина.

Построить компьютер, который работал бы на повышенной точности примерно с той же скоростью, с какой современные машины работают на стандартной точности, в принципе, не просто, а очень просто. Конечно, электронная промышленность на это вряд ли пойдет – ведь потребуется выпустить микропроцессор, не годный ни для чего, кроме использования в экзафлопсном суперкомпьютере. Здесь нам на помощь приходит технология сопроцессоров на программируемой логике.

В своих исследованиях последних лет нам удавалось достигать на FPGA-сопроцессорах быстродействие, заметно превышающее быстродействие процессоров общего назначения [5]. Речь шла при этом о вычислениях со стандартной точностью double. Основных препятствий для дальнейшего развития этих работ было два.

С одной стороны, приходилось преодолевать частотный разрыв между программируемой и жесткой логикой. Программируемая логика работает на частотах, примерно на порядок меньших, чем жесткая, и это соотношение в последние годы стабилизировалось. Переход к повышенной точности снижает скорость выполнения арифметических операций в процессоре почти на 2 порядка, а в FPGA ведет лишь к увеличению используемой площади кристалла, но не к падению скорости (рабочая частота остается той же, что и для стандартной точности). При этом доступная площадь кристалла (объем оборудования) современных FPGA быстро растет, почти так же быстро, как 20 лет назад росла рабочая частота процессоров общего назначения.

С другой стороны, объем внутренней памяти FPGA, в которую необходимо поместить обрабатываемые данные, чтобы получить ускорение, зачастую слишком мал. Эффект ускорения при обработке данных уничтожается коммуникационными затратами на подкачку данных и выгрузку результатов. Как мы показали выше, использование повышенной точности очень сильно смещает соотношение между вычислениями и коммуникациями в сторону вычислений, так что и эта проблема становится менее острой.

В докладе будут представлены результаты реализации в FPGA-ускорителях некоторых типовых алгоритмов с четверной точностью, в частности, умножения плотных матриц и модифицированного процесса Грама-Шмидта. Будет рассказано о применении построенных сопроцессоров для решения соответствующих модельных задач блочными методами, с учетом необходимости организации подкачки и выгрузки данных по ходу счета.

ЛИТЕРАТУРА:

1. Л.К. Эйсымонт, В.С. Горбунов. На пути к экзафлопсному суперкомпьютеру: результаты, направления, тенденции. // Труды Третьего Московского суперкомпьютерного форума, Москва, 2012.
2. А.О. Лацис. Параллельная обработка данных. Издательский центр «Академия», Москва, 2010г. ISBN 978-5-7695-5951-8
3. http://www.netlib.org/scalapack Дата обращения 31.05.2013.
4. http://gmplib.org Дата обращения 31.05.2013.
5. С.С. Андреев, С.А. Дбар, А.О. Лацис, Е.А. Плоткина. Система программирования Автокод HDL и опыт ее применения для схемной реализации численных методов в FPGA. Материалы Всероссийской научной конференции "Научный сервис в сети ИНТЕРНЕТ", Новороссийск, сентябрь 2009 г.