С. С. Андреев, С. А. Дбар, А. О. Лацис, Е. А. Плоткина
Институт прикладной математики им. М. В. Келдыша РАН
тел.: +7(499)250-79-72
e-mail: inform@kiam.ru

Макет гибридной реконфигурируемой вычислительной системы и реализация на нем вычислений с повышенной точностью.

Работа выполнена при частичной поддержке РФИИ, грант 09-01-05005-б за 2009 год.

Введение.

Общепринятым сегодня в мировом суперкомпьютерном сообществе рубежом, на котором следует ожидать появления экзафлопсного суперкомпьютера, считается 2018 год. Однако, чем ближе этот рубеж, тем больше возникает сомнений в разрешимости этой самопоставленной мировым суперкомпьютерным сообществом задачи.

Не единственным, но важнейшим препятствием для ее решения являются ограничения по энергетике [1,2].

Самый быстродействующий на сегодня суперкомпьютер Tianhe-2 имеет пиковую энергоэффективность, равную 3 Гфлопс/Ватт [3], самый «зеленый» - Eurora – 3.2 Гфлопс/Ватт [4]. Исходя из общепризнанного ограничения общего энергопотребления суперкомпьютера 20 мегаваттами [1], для создания экзафлопсной машины необходима энергоэффективность в 50 Гфлопс/Ватт, как минимум. Даже самые оптимистичные из экспертов суперкомпьютерной отрасли не могут сегодня, по нашим сведениям, прогнозировать появления до 2018 года вычислителей с таким уровнем энергоэффективности, готовых к использованию в составе суперкомпьютера. Самые оптимистичные прогнозы дают оценку возможного роста энергоэффективности до 2018 года примерно в 4-5 раз, в то время как надо – в 15-20.

Также немаловажной представляется проблема практического использования «слишком большой» вычислительной системы. Имеющиеся сегодня проекты экзамасштабных суперкомпьютеров предусматривают количество вычислительных узлов порядка ста тысяч. Опыт реального использования машин с так сильно распределенной вычислительной мощностью на сегодня невелик, если вообще существует, сомнения в том, что на этом компьютере, занимая его целиком, сможет работать что-либо, кроме теста HPL, на наш взгляд, более чем оправданы.

Таким образом, необходимость действительно радикальных архитектурных изменений в суперкомпьютерной отрасли становится сегодня актуальной, как никогда. Потенциал архитектуры многопроцессорных систем из однородных многоядерных вычислительных узлов исчерпан окончательно. Появление систем из гибридных вычислительных узлов с такими ускорителями, как GPGPU или Xeon Phi, было важным и своевременным, но далеко не достаточным шагом в правильном

направлении. Приведенные выше значения энергоэффективности «маяков» суперкомпьютерной отрасли уже учитывают эффект применения этих технологий. Нужны еще гораздо более эффективные (в частности, более энергоэффективные) архитектуры.

При этом, на наш взгляд, ошибочно было бы полагать, что работы по поиску новых архитектур актуальны исключительно в контексте гонки за рекордами при строительстве уникальных, сверхбольших вычислительных систем. Новые архитектуры потребуют новых технологий их использования по всей цепочке, от алгоритмов и численных методов до языков программирования и «железа», а такие комплексные по своей природе технологии всегда рождаются на макетах и опытных образцах совсем не рекордного размера. Попытка ограничиться при строительстве малых и средних систем исключительно старыми подходами, в принципе позволяющими, в данном, конкретном случае, достичь необходимого быстродействия, ведет в среднесрочной перспективе к технологическому застою, и может обойтись очень дорого как в переносном, так и в прямом смысле этого слова.

Таким образом, сегодня настоятельно необходимы вычислительные архитектуры, позволяющие выиграть у имеющихся решений хотя бы один порядок по энергоэффективности и, по возможности, заметно сократить количество вычислительных узлов.

1. Возможно ли в принципе требуемое улучшение архитектуры?

Убедимся, что используемые сегодня архитектурные подходы действительно допускают возможность улучшения, в частности, по энергетике.

Затраты энергии на стандартный 64-битный флоп сегодня составляют примерно **70 пикоджоулей**, с перспективой довольно скорого улучшения на порядок.

Коммуникационные затраты на обслуживание этого флопа, то есть на перемещение данных (64*3=192 бита) в арифметическое устройство и из него – **примерно 1000 – 10000 пикоджоулей** для фоннеймановской машины [2]. Это дает нам **1 Гфлопс/Ватт** [1,2] (у лучших современных серверных процессоров пиковая энергоэффективность – около полутора Гфлопс/Ватт, у GPGPU и Xeon Phi – около четырех).

Учитывая, что архитектура вычислителя – это, в первую очередь, конкретный способ организации его внутренней коммуникационной системы, легко видеть, что потенциал архитектурного совершенствования, действительно, очень велик: сократив удельные коммуникационные затраты в расчете на полезный флоп, можно добиться очень многого.

2. Самая лучшая архитектура.

Острота стоящей проблемы требует радикальных решений. Самая лучшая, в принципе, архитектура вычислителя – это, в известном смысле, отсутствие архитектуры как таковой, то есть построение «процессоров одной задачи» для прямой схемной реализации конкретной вычислительной процедуры. Если такой «процессор одной задачи» построен правильно, то

накладных расходов на коммуникации в нем нет или почти нет, поскольку арифметические устройства, выполняющие полезные операции, соединены друг с другом оптимальным именно для данной вычислительной процедуры образом. Техническая возможность гибкой реализации процессоров одной задачи в машине общего назначения обеспечивается использованием ускорителей на FPGA. Такой ускоритель представляет собой универсальный «конструктор» произвольных цифровых электронных схем, в частности, схем вычислительного характера, причем время «зарядки» даже очень большой FPGA конкретной схемой редко превышает 3-4 минуты. Возможности построения на FPGA очень быстрых и эффективных схемных реализаций вычислений с плавающей точкой демонстрировались неоднократно. Наиболее распространенное объяснение того, почему ускорители на FPGA сегодня используются крайне мало, состоит в очень большой длительности и трудоемкости проектирования самих схем вычислительных сопроцессоров. Однако, как раз в области сокращения сроков и снижения трудозатрат при разработке схем, реализуемых в FPGA, в частности, схем вычислительного характера, многими коллективами разработчиков в последние годы достигнут значительный прогресс [5,6,7]. Тем не менее, технология FPGA-ускорителей в суперкомпьютерах общего назначения по-прежнему не используется. Помимо вполне очевидных сложностей, связанных с человеческим фактором, должна быть еще какая-то принципиальная проблема технического характера. Попробуем ее сформулировать.

3. Применение FPGA-ускорителей само по себе проблемы не решает.

Возможность строить эффективные во всех отношениях схемы вычислительного характера в FPGA решающим образом базируется на использовании накристальной блочной памяти. Память такого рода находится непосредственно в микросхеме FPGA, «нарезана» мелкими блоками, и является синхронной, со временем доступа, равным времени доступа к регистру. Именно использование многочисленных (десятки, сотни) отдельных блоков очень быстрой синхронной памяти в составе «конструктора» и позволяет создавать из него схемы, в которых все арифметические устройства «накормлены» оптимальным образом, и почти не простаивают. Подавляющее большинство примеров очень быстрой и эффективной обработки данных в FPGA демонстрирует обработку данных, размещенных именно во внутренней быстрой памяти. Проблема заключается в том, что памяти этой – мало, примерно столько же, сколько кеша в современных процессорах общего назначения, в то время как гибридный вычислительный узел должен обрабатывать десятки и сотни гигабайт данных. При попытке же выйти за пределы кристалла FPGA большая часть преимуществ внутренней реализации вычислений теряется. Скоростной выигрыш, полученный при, действительно, очень быстрой обработке данных внутри FPGA, с лихвой «съедается» накладными расходами на частое копирование данных из процессора в ускоритель и обратно мелкими порциями. Теоретически можно пытаться расширить возможности

реконфигурации под конкретную задачу с единичного FPGA-ускорителя на суперкомпьютер в целом, но в среднесрочной перспективе такое вряд ли возможно как технически, так и экономически, по крайней мере, в классе суперкомпьютеров общего назначения.

Таким образом, применение FPGA-ускорителей само по себе не решает проблемы эффективной организации коммуникаций внутри вычислителя, а лишь выносит ее за пределы кристалла FPGA, где она (проблема) проявляется с новой силой. Мы снова вернулись к исходной точке: слишком велики удельные коммуникационные затраты в расчете на полезную арифметическую операцию с плавающей точкой.

4. Так как же ее решить?

Присмотримся еще раз, внимательно, к формулировке проблемы: «Слишком велики коммуникационные затраты на обслуживание стандартного (64-разрядного) флопа». Попытки уменьшить коммуникационные затраты, в первом приближении, успехом не увенчались. Тогда, может быть, следует попытаться увеличить стандартный флоп? Почему бы не сделать его, например, 128-разрядным? Прежде чем переходить к анализу такого решения с технической точки зрения, необходимо ответить на главный вопрос: нужны ли (и насколько) прикладным программистам суперкомпьютеров вычисления с повышенной точностью?

5. На экзафлопсном суперкомпьютере вычисления с повышенной точностью не нужны. Они там – ОЧЕНЬ нужны.

Вопрос о необходимости и применимости вычислений с повышенной точностью в суперкомпьютерных расчетах ставит многих прикладных программистов в тупик. В самом деле, велик ли резон задумываться о применимости того, чего в выпускаемых промышленностью вычислительных устройствах все равно нет? Применительно к FPGA-ускорителям, в которых все, в том числе – арифметика, делается своими руками, и делается в том виде, в каком это действительно нужно, резон задуматься об этом, очевидным образом, появляется.

Чем вообще определяется необходимая точность вычислений с вещественными числами? Как известно, сеточные задачи сводятся к системам линейных алгебраических уравнений, порядок которых равен общему числу ячеек в сетке. Итерационные методы решения таких СЛАУ нередко подразумевают многократное скалярное умножение строк таких матриц, при котором накопленная погрешность, в общем случае, пропорциональна длине строки. Тем самым, необходимая точность вычислений, грубо говоря, пропорциональна общему числу ячеек в используемой сетке.

Обратимся к истории. На компьютере БЭСМ-6 трехмерный расчет на сетке размером 20x20x20 считался решением серьезной вычислительной задачи, точность представления вещественных чисел была 48 (одинарная) или 96 (двойная) разрядов. Размеры (в ячейках) расчетных сеток, ради обработки которых имеет смысл строить экзамасштабные суперкомпьютеры,

превосходят указанные в миллиарды раз. Проводить расчеты на этих сетках предполагается со «стандартной» вещественной точностью в 64 разряда. Даже мы, вовсе не являясь специалистами в вычислительной математике, легко видим, что здесь «что-то не сходится».

А что говорят специалисты?

Одному из авторов настоящей работы приходилось слышать мнения очень грамотных специалистов на этот счет. В частности, одно из них заключалось в том, что стандартной 64-разрядной точности, вообще говоря, мало даже для расчетов на современных суперкомпьютерах, а достоверность результатов многих из таких расчетов, именно по этой причине, оставляет, мягко говоря, желать много лучшего.

Практически все опрошенные специалисты отмечали, что вычисления с повышенной точностью во многих случаях позволяют ускорить сходимость или снизить порядок аппроксимации, что означает прямую замену нескольких операций с меньшей точностью на одну операцию с большей точностью. Тот же эффект достигается при использовании экономных по числу операций, но недостаточно устойчивых численных методов, которые становятся устойчивыми при повышении точности вычислений (например, ортогонализация Грама-Шмидта). Все эти приемы уменьшения числа операций и/или упрощения алгоритмов хорошо известны математикам, но на практике не применяются, поскольку вместо экономии сулят проигрыш. Современный процессор общего назначения вычисляет с четверной точностью не менее чем в 10 раз медленнее, чем с двойной. В этих условиях, наоборот, высшей доблестью считается умение свести задачу к вычислениям с как можно меньшей точностью, что не только не всегда просто, но и не всегда корректно.

Несмотря на большой накопленный опыт выражения алгоритма в терминах операций с как можно меньшей точностью, это не всегда получается. Тогда прикладному программисту уже просто приходится, в обязательном порядке, использовать крайне медленно работающую, программно реализованную арифметику четверной (и более) точности современных процессоров общего назначения.

Таким образом, арифметика повышенной точности:

- для некоторых численных методов – просто категорически необходима, в силу свойств самого метода,
- для многих других численных методов или уже необходима, или станет необходимой, в силу роста масштаба задач, даже если не все математики, использующие эти методы сегодня, уже об этом догадываются,
- для некоторых численных методов, в принципе позволяющих обходиться меньшей точностью, может служить приемом сокращения числа арифметических операций, необходимых для решения конкретной задачи.

Словом, экзамасштабный суперкомпьютер либо будет машиной для расчетов с повышенной точностью, хотим мы этого или нет, либо сможет им быть, если мы этого специально захотим, для довольно широкого круга приложений.

В обоих случаях использование каждой операции с повышенной точностью эквивалентно использованию нескольких операций со стандартной точностью. Пока операции с повышенной точностью реализованы программно, как в процессорах общего назначения, вынужденный переход к повышенной точности является бедствием, многократно замедляющим расчет, а специальный, сознательный переход к повышенной точности лишен смысла. Ситуация была бы в точности противоположной, если бы арифметика повышенной точности работала так же быстро и эффективно, как арифметика стандартной точности. Вынужденный переход к повышенной точности тогда прошел бы без катастрофического падения быстродействия, а сознательный переход к повышенной точности стал бы действенным методом повышения быстродействия. В обоих случаях, обмен нескольких стандартных операций на одну операцию с повышенной точностью, стал бы «выгодной сделкой». Причем выгодна эта сделка не только по быстродействию, но и по энергетике. Ведь коммуникационная «цена» арифметической операции при переходе к повышенной точности возрастает всего лишь вдвое, в то время как «покупаемая» этой «ценой» работа – в большее количество раз.

Итак, мы хотели утяжелить арифметическую операцию, чтобы снизить удельную коммуникационную цену выполняемой в ней работы, и мы этого добились, причем в очень общем случае – не для какой-то конкретной архитектуры, а для любой вычислительной работы, основанной на арифметических операциях с плавающей точкой. Теперь дело за малым – реализовать в FPGA «настоящую» плавающую арифметику с четверной точностью.

6. Чего стоит повышение точности вычислений в FPGA, и что оно дает.

По сравнению с вычислительными устройствами, реализованными в жесткой логике, FPGA обладают очень низкой рабочей частотой (при автоматизированном построении схем вычислительного характера обычно используется частота 100 – 125 Мгц), которая уже несколько лет не растет, как и рабочие частоты процессоров общего назначения. Размер же кристаллов FPGA (количество «деталей конструктора», из которых можно строить схемы), напротив, растет, примерно соответствуя сложности «настоящих» микропроцессоров. Частотный разрыв с ядрами процессоров общего назначения, превышающий один порядок, несколько осложняет попытки использовать FPGA для ускорения счета, но вовсе не делает их безнадежными. Выигрыш в эффективности использования оборудования (тех самых «деталей конструктора»), получаемый при переходе от процессоров с программным управлением к процессорам одной задачи, настолько велик, что зачастую удается не только наверстать частотный разрыв, но и получить весомое ускорение. Возможность кратного ускорения вычислений со стандартной вещественной двойной точностью на одной FPGA по сравнению с современным ей двухпроцессорным сервером демонстрировалась неоднократно[5,6,8]. Поэтому, для анализа возможностей реализации

арифметики четверной точности в FPGA нам достаточно просто сравнить ее с реализацией стандартной двойной точности, которую мы считаем достаточно хорошо исследованной и опробованной на практике.

Основной режим работы эффективных процессоров одной задачи – длинный конвейер. Он строится из элементарных арифметических устройств (сложителей, вычитателей, умножителей и т. п.), каждое из которых само по себе работает в конвейерном режиме, то есть на каждом такте рабочей частоты принимает новые операнды и выдает новый результат. Выдаваемый на очередном такте результат получается из операндов, принятых некоторое фиксированное число тактов назад. Поскольку даже суммарная задержка конвейера из десятков ступеней обычно много меньше тех объемов данных, которые надо обработать, на итоговое реальное быстродействие ее величина влияет мало. Производительность конвейера, таким образом, грубо равна одному готовому результату вычисления по некоторой большой, сложной формуле за один такт рабочей частоты. Если на кристалле хватает места, можно построить конвейер из нескольких (2, 4, 8, …) «ниток», увеличивая производительность в соответствующее число раз [5,8].

В приведенных здесь пояснениях нигде не присутствует понятие точности вычислений, из чего можно сделать вывод, что быстродействие реализуемых в FPGA процессоров одной задачи напрямую вообще не зависит от точности вычислений. Сделать такой вывод не только можно, но и нужно, поскольку он вполне соответствует действительности.

Косвенно производительность вычислительных сопроцессоров, конечно, зависит от точности вычислений, поскольку арифметические устройства большей точности занимают на кристалле FPGA гораздо больше места. Но при условии, что требуемую схему разместить на кристалле удалось, ее реальное быстродействие от точности вычислений, действительно, не зависит.

Поскольку размер кристаллов FPGA, выпускаемых промышленностью, неуклонно растет, наблюдается ситуация идеального распараллеливания на микро-уровне: занимаемая схемой площадь кристалла, все возрастающая по мере выпуска новых моделей, просто «разменивается» на итоговое быстродействие сопроцессора. При этом получающиеся сопроцессоры, за счет низкой рабочей частоты, оказываются, по сравнению с вычислителями на жесткой логике, очень «холодными».

Немного конкретных цифр.

Нами была выполнена реализация комплекта элементарных арифметических устройств арифметики с плавающей точкой четверной точности. По сравнению с аналогичными устройствами двойной точности, изготовленными «под ключ» при помощи схемотехнической САПР фирмы Xilinx, сложитель занял примерно в 5 раз больше места, умножитель – в 20. Сравнение не вполне корректное, поскольку высокопрофессиональная коммерческая реализация для двойной точности сравнивалась с экспериментальной, «самодельной» реализацией четверной точности, но некоторое представление о порядке сравниваемых величин оно все же дает.

Теперь обратимся к результатам сравнения быстродействия и, что особенно интересно, энергоэффективности.

В качестве примера хорошо подходящего для реализации в качестве сопроцессора алгоритма была выбрана матричная операция GEMM из библиотеки BLAS [9]. Эта операция выполнят обобщенное перемножение матриц по формуле: **C = alpha*A*B + beta*C**, где **alpha** и **beta** – скаляры, **A, B** и **C** – матрицы. Был реализован FPGA-сопроцессор, выполняющий эту операцию с вещественными числами четверной точности, для блоков матричных строк, способных разместиться во внутренней памяти сопроцессора. Для этого сопроцессора была написана управляющая программа, передающая данные блоками из памяти гибридного вычислительного узла в сопроцессор и обратно, чтобы можно было перемножать матрицы произвольного размера.

Производительность такого гибридного приложения на матрицах размером порядка гигабайта превзошла производительность процессорного ядра с частотой 2.6 Ггц в 12-24 раза, в зависимости от конкретного размера матрицы. Таким образом, использованная при этом FPGA xc6vlx240t фирмы Xilinx (далеко не самая современная и большая, но способная вместить реализацию GEMM четверной точности с заметным запасом) работала на этом тесте примерно как два современных серверных процессора. Два современных серверных процессора при такой работе потребляют 260 – 270Вт, указанная же FPGA – не более 30Вт. Поставленная в начале нашего исследования задача улучшения энергоэффективности на порядок, таким образом, для данной задачи успешно решена, причем без потерь в быстродействии.

В качестве примера плохо подходящего для реализации в качестве сопроцессора алгоритма была выбрана задача, часто используемая в качестве демонстрационной в параллельных вычислениях: решение сеточного аналога задачи Дирихле для уравнения Пуассона итерационным методом Якоби, на двумерной прямоугольной сетке [10]. Эта задача характеризуется довольно малым объемом вычислений в расчете на одно обрабатываемое значение: всего 7 полезных операций с плавающей точкой. Как и в случае операции GEMM, было реализовано гибридное приложение с блочной подкачкой данных, чтобы можно было обрабатывать сетки, не умещающиеся в памяти сопроцессора. Для сокращения числа обменов данными между процессором и сопроцессором использовался известный прием искусственного расширения теневых граней, в данном случае теневая грань шириной 1 расширялась до 11. Несмотря на это, на стандартной двойной точности обогнать процессорное ядро не удалось. На четверной точности было достигнуто восьмикратное ускорение по сравнению с процессорным ядром, что означает кратный выигрыш по энергоэффективности практически без потери быстродействия.

Следует отметить, что значительную часть времени работы гибридного приложения (во втором случае – половину) составляет время ожидания завершения обменов данными между процессором и сопроцессором, которые

на используемой вычислительной платформе пока реализованы очень плохо, и легко могут быть ускорены, как минимум, в 4-5 раз.

Для тех задач, к которым действительно применимы методы вынужденного и/или искусственного повышения точности вычислений, напрашивается очень простой вариант получения таким путем экзамасштабной производительности. Надо просто построить машину для вычислений с повышенной точностью, а посчитать ее быстродействие в эквивалентных флопсах стандартной двойной точности.

7. Гибридно-параллельная реконфигурируемая вычислительная система МВС-М.

На наш взгляд, приведенные здесь общие соображения и примеры неопровержимо свидетельствуют о том, что в рассматриваемом направлении следует работать, в первую очередь – наращивать набор реализованных модельных приложений. Привлечение к этой работе отдельных отважных пользователей – математиков также было бы весьма желательным. В этой связи на первый план выходит вопрос о том, насколько разработка гибридных приложений «упакована» в отработанную и документированную технологию, а используемые при этом аппаратные компоненты (процессоры, ускорители) – в единое аппаратно-программное решение, в вычислительную систему. Без такой «упаковки» говорить о расширении фронта работ в сторону исследования методов и алгоритмов вряд ли реалистично. Несколько лет назад коллеги из НИИ «Квант» любезно предоставили нам учебную гибридно-реконфигурируемую вычислительную систему РВС-1 собственной разработки, на базе которой мы пытались создать экспериментальный суперкомпьютер коллективного пользования. К сожалению, вскоре выяснилось, что система эта по целому ряду параметров для серьезных экспериментов не годится. В этом году мы построили собственную гибридно-параллельную вычислительную систему МВС-М, на которой, в частности, были получены упомянутые выше результаты измерения производительности. Ниже представлена структурная схема этой машины.

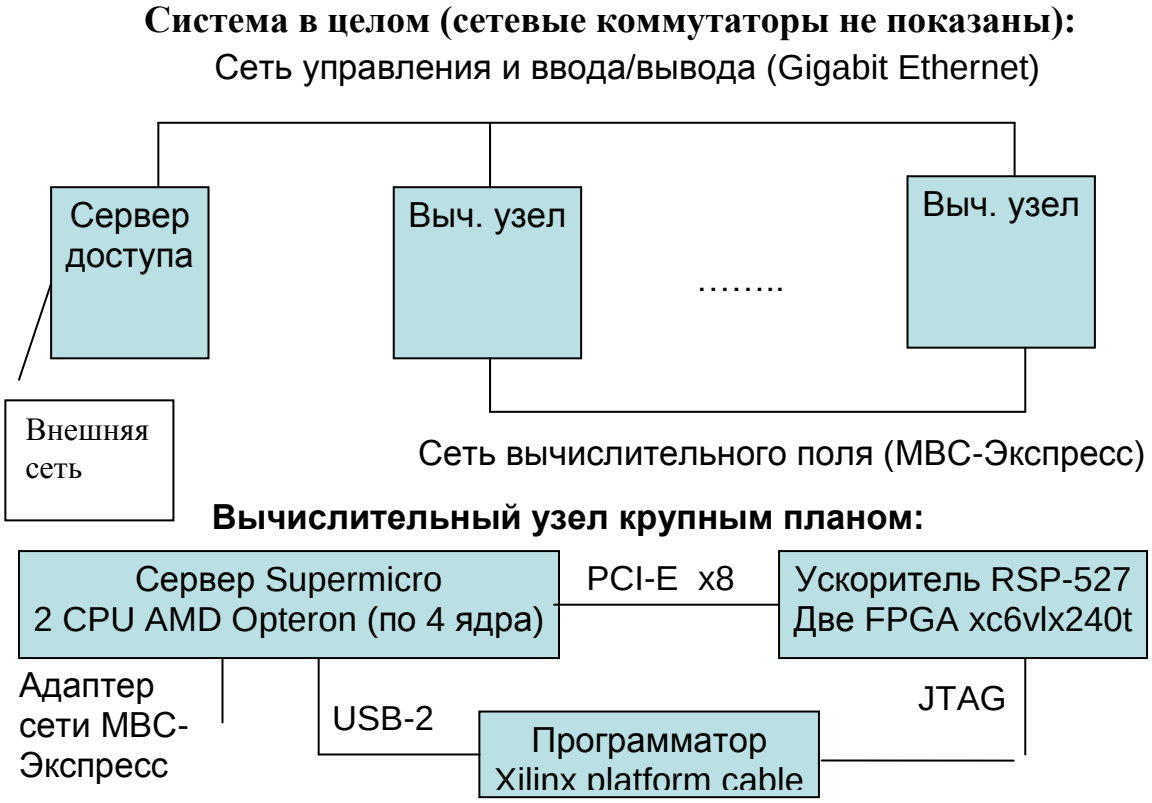

Рис. 1. Общая структура МВС-М.

Коротко о машине. В настоящее время в составе вычислительного поля 3 гибридных узла, но допускается расширение до 23-х. Каждый узел оснащен ускорителем из двух независимых FPGA. Все оборудование, кроме сети управления, программаторов и самих серверов, отечественное.

Ускорители на FPGA и аппаратура сети вычислительного поля МВС-Экспресс – производства НПО «Роста».

Базовое программное и схемное обеспечение разработано в ИПМ РАН, ИПС РАН и НИИ «Квант». Программное обеспечение общего назначения включает в себя коммуникационные библиотеки Скиф-MPI (ИПС РАН) [11], shmem-экспресс (ИПМ РАН, НИИ «Квант») [12], RefNUMA (ИПМ РАН) [13]. Система управления сопроцессором опирается на драйвер МВС-Экспресс, и включает в себя стандартное окружение пользовательских схем, реализуемых в ускорителе (для 32-, 64- и 128-разрядной платформ), скрипты конфигурирования FPGA, базовую библиотеку обмена данными с ускорителем, набор элементарных арифметических устройств с плавающей точкой четверной точности (ИПМ РАН).

Приведенные сведения о базовом программном обеспечении показывают, что МВС-М является вполне типичным кластером из гибридных вычислительных узлов. Технология программирования и использования таких кластеров хорошо известна на примере гибридных кластеров на базе GPGPU [10]. Чтобы сравнить сложность разработки гибридно-параллельных приложений для МВС-М со сложностью разработки таких же приложений для кластера с GPGPU, достаточно сопоставить языки программирования

вычислительных ядер: все остальное хорошо известно и совпадает. Если окажется, что система программирования вычислительных ядер для ускорителей на FPGA по сложности использования хотя бы сопоставима по порядку величины, например, с OpenCL, то МВС-М можно, в первом приближении, считать готовым аппаратно-программным решением для экспериментирования с модельными задачами.

Для разработки схем сопроцессоров (вычислительных ядер) используется система прикладного схемотехнического проектирования Автокод HDL (ИПМ РАН) [5,8]. Сколько-нибудь полный рассказ об этой системе, о ее входном языке мог бы быть темой отдельной статьи, и выходит за рамки настоящего текста. Однако, некоторые основные черты языка Автокод HDL все же стоит перечислить.

Разработка схем ведется в текстовом виде, в той же Linux-консоли, что и разработка программ, и не требует от разработчика специальных знаний в области электроники. Интерфейс сопроцессора с управляющей программой на универсальном процессоре унифицирован, его использование в значительной степени автоматизировано соответствующими конструкциями языка. Язык описания схем – двухуровневый: расширенные возможности языка транслируются в базовые, а те, в свою очередь, в VHDL.

Базовый язык, по сравнению с VHDL, упрощен, минимизирован и специализирован для решения вычислительных задач, но использует ту же самую модель программирования, что и VHDL. Это позволяет поддерживать на высоком уровне эффективность создаваемых схем – напрямую язык не вносит в схему никаких «накладных расходов на удобство описания».

Расширенные возможности языка посвящены автоматизации построения интерфейса с управляющей программой, но главным образом – автоматическому построению правильно согласованных конвейеров, выполняющих в поточном режиме вычисления над вещественными значениями согласно заданному арифметическому выражению. Расширенные возможности языка построены в рамках той же, заимствованной из VHDL, модели программирования, что и базовые.

Язык поддерживает три варианта разрядности чисел с плавающей точкой: 32, 64 и 128 разрядов.

Таким образом, разработка вычислительных ядер упрощается и ускоряется, по сравнению с ручным кодированием на VHDL, не за счет отказа от присущей VHDL модели программирования, а за счет «придания ей человеческого лица». Это позволяет сохранить эффективность, присущую ручному кодированию, при многократном сжатии текста и повышении его читабельности.

Благодарности:

- Коллегам из НИИ «Квант» за неоднократные консультации и помощь в сложных вопросах схемотехники,
- И. В. Простову за участие в тестировании компонентов арифметики четверной точности,
- Ю. А. Климову за большой вклад в разработку базового ПО,

- Г. П. Савельеву за помощь в наладке и эксплуатации макета.

Литература:

1. Л.К.Эйсымонт, В.С.Горбунов. На пути к экзафлопсному суперкомпьютеру: результаты, направления, тенденции. Первый Национальный суперкомпьютерный форум, Переславль-Залесский, 2012г.
2. http://spectrum.ieee.org/computing/hardware/nextgeneration-supercomputers. Дата обращения 21.10.2013.
3. www.top500.org. Дата обращения 21.10.2013.
4. www.green500.org. Дата обращения 21.10.2013.
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. Труды Всероссийской суперкомпьютерной конференции «Научный сервис в сети Интернет», сентябрь 2009г., Новороссийск.
6. А. И. Дордопуло, И. И. Левин, В. А. Гудков. Методы автоматического определения эффективности реализации фрагментов прикладных программ на языке COLAMO. Труды Международной суперкомпьютерной конференции «Научный сервис в сети Интернет», сентябрь 2012г., Новороссийск.
7. http://calypto.com/en/products/catapult/overview. Дата обращения 21.10.2013.
8. Абрамов С. М., Дбар С. А., Климов А. В., Климов Ю. А., Лацис А. О., Московский А. А., Орлов А. Ю., Шворин А. Б. Возможности суперкомпьютеров "СКИФ" ряда 4 по аппаратной поддержке в ПЛИС различных моделей параллельных вычислений. Материалы Седьмой Международной научной молодежной школы "Высокопроизводительные вычислительные системы ВПВС-2010, 27 сентября--2 октября 2010, Дивноморское, Геленджик, Россия.
9. www.netlib.org/BLAS. Дата обращения 21.10.2013.
10. http://www.kiam.ru/MVS/documents/k100/steps.html Дата обращения 21.10.2013.
11. http://www.kiam.ru/MVS/documents/mvse/skifmpiuserman.html Дата обращения 21.10.2013.
12. http://www.kiam.ru/MVS/documents/mvse/shmemprogman.html Дата обращения 21.10.2013.
13. http://www.kiam.ru/MVS/research/refnuma/index.html Дата обращения 21.10.2013.