ИПМ им.М.В.Келдыша РАН

Абрау-2016 • Труды конференции

С.С. Андреев, С.А. Дбар, А.О. Лацис,
Е.А. Плоткина

Некоторые проблемы реализации вычислений на FPGA-ускорителях

Рекомендуемая форма библиографической ссылки

Андреев С.С., Дбар С.А., Лацис А.О., Плоткина Е.А. Некоторые проблемы реализации вычислений на FPGA-ускорителях // Научный сервис в сети Интернет: труды XVIII Всероссийской научной конференции (19-24 сентября 2016 г., г. Новороссийск). — М.: ИПМ им. М.В.Келдыша, 2016. — С. 9-13. — URL: http://keldysh.ru/abrau/2016/32.pdf

Размещена также презентация к докладу

Некоторые проблемы реализации вычислений на FPGA-ускорителях

С.С. Андреев, С.А. Дбар, А.О. Лацис, Е.А. Плоткина

ФИЦ Институт прикладной математики им. М. В. Келдыша РАН

Аннотация. Процессор общего назначения перестал устраивать суперкомпьютерную отрасль в качестве единственного вычислительного устройства. Необходимы более эффективные альтернативы. В качестве таковых предлагаются, в частности, GPGPU и Xeon Phi. Эти вычислители - шаг в верном направлении, но шаг недостаточный. Эффективность, в частности, энергетическую, надо повысить еще гораздо сильнее. Самый радикальный способ это сделать — перейти к процессорам одной задачи. Таковые могут быть реализованы только в FPGA. Этим объясняется наш интерес к этой уже, казалось бы, забытой технологии высокопроизводительных вычислений. К сожалению, принципиальная возможность вычислять очень эффективно при использовании FPGA не реализуется автоматически. В докладе делается попытка систематизировать принципиальные трудности, стоящие на пути реализации высокопроизводительных вычислений на FPGA-ускорителях.

Ключевые слова: высокопроизводительные вычисления, GPGPU, FPGA, процессор одной задачи, энергетическая эффективность.

Введение

Проблема использования нетрадиционных вычислительных архитектур довольно часто воспринимается по следующей "наивной" схеме:

- существует возможность ускорить вычисления, использовав, помимо процессора общего назначения, некоторые странные вычислительные устройства - сопроцессоры-ускорители;
- в силу каких-то непостижимых причин, программирование этих ускорителей осуществляется на необычных языках, не таких, как просто обыкновенные Си или Фортран;
- изучать такие языки сложно, и потому возможностью ускорить вычисления таким образом лучше не пользоваться;
- если же пользоваться ей все-таки придется, то самое главное - выбрать ускоритель с "наименее необычным" языком.

В действительности все гораздо сложнее и, в некотором смысле, гораздо хуже.

Ускорение вычислений с использованием вычислителей нетрадиционной архитектуры является сегодня и в ближайшей перспективе не столько возможностью, сколько необходимостью. Для того, чтобы это ускорение

получить, совершенно недостаточно освоить язык программирования того или иного сопроцессора-ускорителя, как бы прост или сложен этот язык ни был. Для начала потребуется выбрать численный метод, в принципе способный ускоряться на новых архитектурах, и реализовать его в виде алгоритма, практически пригодного для такого ускорения. Выделенное при такой реализации вычислительное ядро, подлежащее переносу на ускоритель, действительно, придется записать на довольно необычном языке. Необычность эта ни в коей мере не является прихотью или недоработкой инженеров и системных программистов, придумавших данный ускоритель. Она является печальной (для прикладного программиста) необходимостью, платой за возможность действительно использовать потенциал этого устройства для ускорения вычислений. Соответственно, изучение этого языка включает в себя не только и не столько освоение непривычного синтаксиса, сколько освоение непривычных архитектурных понятий. Без этого ускорения не получить.

Таким образом, перенос приложения на суперкомпьютер с нетрадиционной архитектурой подразумевает преобразование программы, изначально предназначенной для обычной MPP-системы, состоящее из нескольких шагов. Первые и главные шаги нацелены на преобразование структуры программы. Эти преобразования выполняются в совершенно традиционных программистских терминах, и могут быть как универсальными, так и нацеленными на совершенно конкретную нетрадиционную архитектуру. Знание языка программирования конкретного ускорителя на этом этапе не требуется, но требуется знание, например, того, насколько велика его внутренняя память. Очень важно понимать, что без этих предварительных шагов любое формальное переписывание части программы на языке программирования ускорителя совершенно бессмысленно, поскольку приведет только к замедлению программы.

Завершающий шаг преобразования программы - переписывание вычислительного ядра, выносимого на ускоритель, на соответствующем языке. Чем больше в принципе потенциал ускорения, заложенный в том или ином типе ускорителя, тем, как правило, сложнее этот язык, и тем проблематичнее получение хорошего вычислительного ядра при механическом использовании систем автоматизации, предоставляющих возможность написать вычислительное ядро на "почти обычном" Си или Фортране.

В предлагаемой работе упомянутые выше проблемы рассматриваются для самого многообещающего в принципе, но и самого тяжелого по характеру предварительной переработки текста программы, типа ускорителей, а именно - для гибридно-параллельных систем с ускорителями на базе FPGA.

1. Требования к численному методу

Начнем с рассмотрения наиболее универсальных требований к численным методам, подлежащим ускорению на нетрадиционных архитектурах. Заодно выясним, чем эти требования обусловлены.

Будем исходить из следующих основных посылок:

- процессор общего назначения неэффективен на вычислительной работе, необходима гораздо более эффективная альтернатива;
- альтернатива в виде GPGPU и Xeon Phi все еще недостаточно эффективна;
- повышение эффективности возможно только за счет оптимизации коммуникаций в системе процессор — память на микро-уровне.

Обоснование этих тезисов выходит за пределы данной работы, но в последние годы может быть легко найдено во множестве источников, например, в [1].

Из приведенных тезисов легко получить ряд простых, но очень важных выводов. Они касаются ограничений на алгоритмы, которые можно надеяться ускорить путем реализации на вычислителях новой архитектуры.

Первый вывод: повышение эффективности за счет оптимизации коммуникаций возможно только для таких алгоритмов, в которых есть, что оптимизировать. Если некоторая величина интенсивно используется в программе, ее можно разместить в небольшой, но более быстрой памяти, сэкономив несколько дорогостоящих обращений к памяти большой, но медленной. Однако, если каждое из прочитанных из большой памяти значений используется всего один раз, экономия невозможна по определению. Значит, выиграть от перехода на новые вычислительные архитектуры смогут только алгоритмы с высоким коэффициентом переиспользования данных.

Второй вывод: весьма вероятно, что новые архитектуры потребуют локализации многократно используемых данных в достаточно мелких блоках. В самом деле, оптимизация коммуникаций в системе процессор-память предполагает, что память способна одновременно снабжать данными много вычислительных устройств, то есть представляет собой, фактически, сложно организованную систему из многих памятей. Но такое осуществимо физически, как правило, внутри одной микросхемы, при условии, что вычислительные устройства находятся в ней же. Это ограничивает объем памяти, доступной для эффективной организации, примерно на 3 порядка по сравнению с объемом памяти хорошего вычислительного сервера.

Третий вывод: локализация обработки, которую мы хотели бы ускорить, в блоках данных настолько мелких, как указано выше, может потребовать поочередного, многократного копирования данных из основной памяти в память сопроцессора и обратно. В этих условиях интерфейс копирования данных становится узким местом, а его достаточная пропускная способность — главным условием возможности получить ускорение.

Ограничения на алгоритмы, которые, возможно, удастся ускорить, мы получили, зная только проблему, которую необходимо решить (плохая система коммуникаций), но не зная ничего о конкретном способе решения (о конкретной альтернативной архитектуре вычислителя). Иными словами, ограничения эти - универсальные, касаются любых альтернативных архитектур.

2. Процессор одной задачи в FPGA как альтернативная архитектура

Пытаясь изобрести эффективную в коммуникационном отношении вычислительную архитектуру, мы закономерно приходим к идее процессора одной задачи. Вряд ли существует способ более экономно организовать коммуникации, чем сделать это для одного, конкретного, алгоритма, и только для него. Это решение на сегодня автоматически означает использование FPGA. Далее ограничимся рассмотрением гибридно-параллельных суперкомпьютеров с FPGA-сопроцессорами. Основу такой машины составляет MPP-система из вычислительных узлов на базе процессоров общего назначения. В составе каждого вычислительного узла имеется небольшое число FPGA для реализации в них сопроцессоров - ускорителей вычислений. Сплошные массивы из большого количества FPGA, соединенных между собой непосредственно, рассматривать не будем.

Легко видеть, что сформулированные в предыдущем разделе требования к численному методу в полной мере относятся к FPGA-ускорителям. Возможности реализации очень эффективной в коммуникационном отношении обработки данных ограничены пределами одной микросхемы FPGA. Внутренняя память такой микросхемы - исключительно гибкий материал, идеально подходящий для оптимизации коммуникаций на микро-уровне, но суммарный объем такой памяти в одной микросхеме составляет всего лишь несколько мегабайт. Скорость копирования данных в память FPGA (и из нее) примерно на порядок ниже, чем скорость работы памяти современных серверов общего назначения.

С FPGA-ускорителями связано еще одно, характерное именно для них, ограничение, связанное с их низкой (в 20-30 раз ниже, чем у процессора общего назначения) рабочей частотой. На первый взгляд, настолько низкая рабочая частота вообще исключает возможность какого-либо ускорения вычислений. Многочисленные примеры успешного переноса на FPGA различных вычислительных ядер показывают, что это неверно. Однако, низкая частота имеет важное следствие, которое усиливает упомянутое выше требование к мелкоблочности обработки. Подключение на таких низких частотах к FPGA внешних микросхем памяти большого объема, вроде глобальной памяти GPGPU, становится практически бессмысленным: для обеспечения разумных скоростей доступа в эту память потребовалось бы столько ножек, сколько в FPGA практически нет и не может быть. Это, в свою очередь, усиливает требование к мелкоблочности обработки.

Следует отметить, что в совокупности перечисленные требования к численному методу оказываются довольно жесткими, а запись алгоритма в мелкоблочной форме - весьма трудоемкой сама по себе.

3. Языки программирования вычислительных ядер для FPGA

На завершающей стадии переноса приложения, реализуемая в FPGA схема сопроцессора-ускорителя должна быть описана на некотором языке,

внешне напоминающем язык программирования. Было бы очень желательно, чтобы эту работу мог делать прикладной программист, подобно тому, как сегодня он зачастую разрабатывает программу, например, для GPGPU. Значит, язык описания схем должен быть как можно ближе к традиционным языкам. Но сама логика описания эффективно работающей схемы сопроцессора объективно такова, что традиционные алгоритмические языки в чистом виде для этого не годятся. Существуют (и вполне годятся для описания эффективных схем) языки, используемые профессиональными схемотехниками (Verilog и VHDL). По целому ряду известных и хорошо изученных причин, языки эти категорически непригодны для быстрого и безболезненного освоения программистами, не имеющими профильной приборостроительной подготовки [2].

В последнее время вновь стала популярной идея использования для описания схем аннотированного языка Си. Популярность эта объясняется недавним появлением транслятора из аннотированного Си в схему весьма высокого качества, разработанного фирмой Xilinx [3]. К сожалению, этот язык обладает тем же врожденным недостатком, что и многие аннотированные языки автоматизированного распараллеливания программ для традиционных MPP-систем. Недостаток этот состоит в том, что аннотированный Си фактически становится совершенно новым языком, для успешного использования которого необходимо владеть в полной мере системой понятий, присущих целевому оборудованию (в данном случае - схемотехнической картиной мира). Таким образом, вопрос о разработке языка (или хотя бы модели программирования) прикладной вычислительной схемотехники на сегодня остается открытым.

Литература

1. P. Kogge. Next-Generation Supercomputers. http://spectrum.ieee.org/computing/hardware/nextgeneration-supercomputers. Дата обращения 12.05.2016.
2. А. О. Лацис Параллельная обработка данных. Издательский центр «Академия», Москва, 2010 г. ISBN 978-5-7695-5951-8 336с.
3. http://www.xilinx.com/support/documentation/sw_manuals/xilinx2014_2/ug871-vivado-high-level-synthesis-tutorial.pdf Дата обращения 12.05.2016.