Реконфигурируемые мультиконвейерные вычислительные системы для решения потоковых задач

И.А. Каляев, И.И. Левин

Аннотация. В работе рассматриваются принципы построения реконфигурируемых вычислительных систем (РВС) на основе вычислительных полей ПЛИС. Показано, что с помощью таких РВС могут создаваться различные мультиконвейерные вычислительные структуры, обеспечивающие максимальный темп обработки при решении потоковых задач обработки информации и управления и, как следствие, малое время решения подобных задач в целом. Приводятся примеры реальных РВС, построенных на основе вычислительных полей ПЛИС, и на конкретных примерах показываются их высокие технические характеристики. Представляется структура комплекса системного программного обеспечения для РВС, описываются общие принципы его организации, а также принципы функционирования и взаимодействия отдельных компонентов.

Ключевые слова: реконфигурируемые вычислительные системы, вычислительные поля ПЛИС, мультиконвейерные вычислительные структуры, комплекс системного программного обеспечения.

Потоковые задачи и способы их решения

Достаточно широкий класс задач обработки информации и управления составляют так называемые потоковые задачи, т.е. задачи обработки некоторого упорядоченного множества (потока) векторов данных по фиксированному алгоритму.

В общем случае потоковую задачу можно сформулировать в следующем виде. Предположим, что существует некоторое упорядоченное множество векторов данных $\mathbf{D}_i =< \mathbf{d}_1^i, \mathbf{d}_2^i, ..., \mathbf{d}_K^i >$, $(i = 1,2,...,N)$, каждый элемент которого должен быть обработан по фиксированному алгоритму, представляемому в виде графа $\boldsymbol{G(Q,X)}$ (Рис. 1).

При этом каждой вершине $q_j \in \boldsymbol{Q}$ графа $\boldsymbol{G(Q,X)}$ поставлена в соответствие некоторая операция O_j, принадлежащая множеству допустимых операций $\boldsymbol{O}$. Любая дуга $x(q_j, q_{j+1}) \in X$ определяет, что результат операции O_j является входным данным для операции O_{j+1}, поставленной в соответствие вершине q_{j+1}.

Задача потоковой обработки заключается в преобразовании кортежа (потока) векторов входных данных $\mathbf{D}_i$ (i = 1,2,…,N) в кортеж (поток) векторов выходных данных $\mathbf{B}_i =< \mathbf{b}_1^i, \mathbf{b}_2^i, ..., \mathbf{b}_L^i >$ (i = 1,2,…,N) в соответствии с графом $\boldsymbol{G(Q,X)}$.

В простейшем случае для решения сформулированной выше задачи можно использовать последовательную ЭВМ с фон-неймановской архитектурой, основным элементом которой является процессор P.

Для этого процессор P предварительно необходимо запрограммировать на последовательное выполнение всех операций O_j (i = 1,2,…,M), соответствующих вершинам графа задачи $\boldsymbol{G(Q,X)}$, и затем подать на вход ЭВМ последовательность векторов $\mathbf{D}_i$ (i = 1,2,…,N).

Время обработки N векторов входных данных при этом составит

$$T_{реш} = N\sum_{i=1}^{M} f(O_i)\tau ,$$

где $M = |\boldsymbol{Q}|$ – число вершин на графе $\boldsymbol{G(Q,X)}$; $f(O_i)$ – число тактов работы ЭВМ при выполнении операции O_i, соответствующей вершине графа $\boldsymbol{G(Q,X)}$; τ – продолжительность такта.

Очевидно, что при достаточно больших значениях основных параметров потоковой задачи (число входных векторов *N*, число вершин на графе) время решения потоковой задачи может оказаться недопустимо велико.

Существенно сократить время решения потоковой задачи возможно путем организации конвейерной обработки данных. Для этого граф задачи $\boldsymbol{G(Q,X)}$ необходимо предварительно разрезать на непересекающиеся подграфы $\boldsymbol{G_1(Q_1,X_1)}$, $\boldsymbol{G_2(Q_2,X_2)}$,…,$\boldsymbol{G_H(Q_H,X_H)}$, (Рис. 2) и каждому подграфу поставить в соответствие свой процессор P_i $(i = 1,2,...,H)$ и соединить их в конвейерную цепочку, как это показано на Рис. 3.

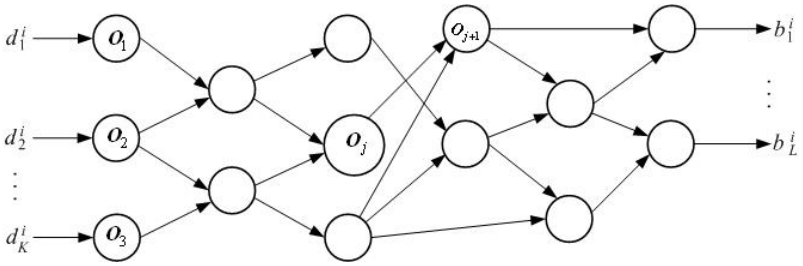

Рис. 1. Граф алгоритма

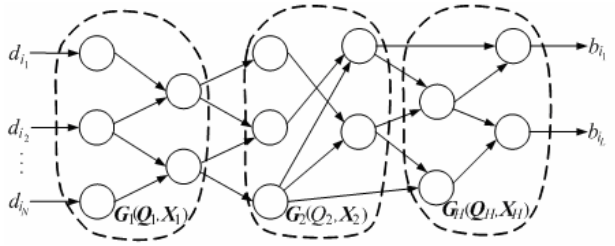

Рис. 2. Разбиение графа алгоритма $\boldsymbol{G(Q,X)}$ на непересекающиеся подграфы

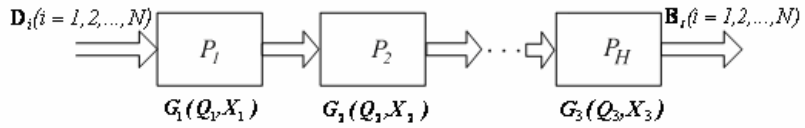

Рис. 3. Конвейер процессоров

Тогда время обработки множества из *N* векторов данных в этой цепочке процессоров будет составлять

$$T_{реш} = \sum_{i=1}^{H} T_{реш}^i + (N-1)\cdot \max_{i=1,...,H}(T_{реш}^i) \approx$$

$$\approx H \cdot \max_{i=1,...,H}(T_{реш}^i) + (N-1)\cdot \max_{i=1,...,H}(T_{реш}^i) =$$

$$=(N+H-1)\,\Delta T$$

где $T_{реш}^i = \sum_{j=1}^{M_i} f(O_j^i)\cdot\tau$ – время реализации подграфа $\boldsymbol{G_i(Q_i, X_i)}$ на процессоре P_i; M_i – число вершин в подграфе $\boldsymbol{G_i(Q_i, X_i)}$; O_j^i – операция, приписанная *j*-ой вершине подграфа $\boldsymbol{G_i(Q_i,X_i)}$; $\Delta T = \max_{i=1,...,H}(T_{реш}^i) = \max_{i=1,...,H}\left(\sum_{j=1}^{M_i} f(O_j^i)\cdot\tau\right)$.

При достаточно большом значении числа *N* величиной *(H-1)* можно пренебречь, вследствие чего значение $T_{реш}$ можно приблизительно определить как

$$T_{реш} \approx N\cdot\Delta T \approx N\cdot\frac{\sum_{j=1}^{M} f(O_j)\cdot\tau}{H}.$$

По сравнению с обработкой множества из *N* векторов данных на одном процессоре в данном случае время решения сокращается приблизительно в *H* раз, где *H* – число процессоров в конвейере.

Как видно из последнего выражения, время обработки массива из *N* данных на конвейере процессоров определяется величиной $\Delta T = \max_{i=1,...,H}(T_{реш}^i)$, т.е. временем обработки дан-

ных на самой «медленной» ступени конвейера. Величину, обратную величине ΔT, принято называть темпом обработки данных в конвейере. Чем больше темп работы конвейера, тем меньшее время потребуется для обработки массива векторов входных данных.

Максимальный темп конвейерной обработки и, как следствие, минимальное общее время $T_{реш}$ обработки всего множества входных данных можно обеспечить, если структура связей между ступенями конвейера будет полностью адекватна топологии связей вершин в графе алгоритма ***G(Q,X)***. Такой способ организации конвейерных вычислений называется *структурным* [1].

Действительно, максимального темпа обработки можно достичь, если каждой операционной вершине q_i $(i = 1,2,...,M)$ графа задачи ***G(Q,X)*** поставить в соответствие свой процессорный элемент P_i $(i = 1,2,...,M)$ и обеспечить связи между P_i согласно топологии информационных дуг графа ***G(Q,X)*** (Рис. 4). Если на вход такой вычислительной структуры последовательно подавать векторы $\mathbf{D}_i$ $(i = 1,2,...,N)$ входного множества, то время их обработки будет составлять

$$T_{реш} = \left(N + H_{\max} - 1\right) \cdot \Delta T ,$$

где $H_{\max}$ – число вершин на критическом пути в графе ***G(Q,X)***, т.е. пути между входной и выходной вершинами, для которого значение $\sum_{j=1}^{K} f(O_j) \cdot \tau$ максимально; $O_j (j = 1,2,...,K)$ – операции, приписанные вершинам критического пути; $\Delta T = \max\limits_{i=1,...,H_{\max}} \left(f(O_j)\right) \cdot \tau$ – время выполнения наиболее длительной операции, приписанной вершинам критического пути.

При большом *N*, а также учитывая, что

$$\Delta T = \max_{i=1,2,...,H_m} f(O_j) \approx \frac{\sum_{j=1}^{M} f(O_j^i)}{M} ,$$

получаем

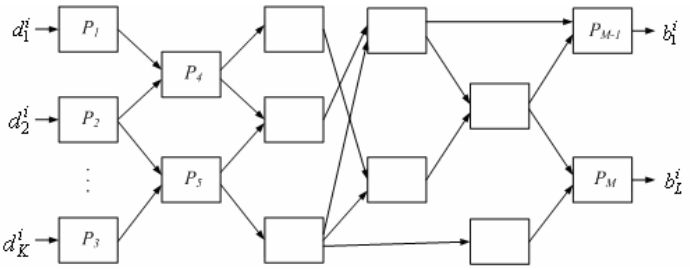

Рис. 4. Мультиконвейерная обработка потока данных

$$T_{реш} = \frac{N}{M} \sum_{j=1}^{M} f(O_j) \cdot \tau .$$

Таким образом, при структурном способе организации вычислений обеспечивается минимальное время обработки массива векторов данных по сравнению со всеми рассмотренными выше способами организации вычислительных систем.

Можно заметить, что вычислительная структура, представленная на Рис. 4, сочетает в себе как параллельный, так и конвейерный способы обработки информации, поскольку входные данные обрабатываются одновременно по различным конвейерным цепочкам процессоров. Конвейерную структуру такого типа в дальнейшем будем называть *мультиконвейерной вычислительной структурой.*

Принципы реализации мультиконвейерных вычислительных структур на основе полей ПЛИС

Можно предложить несколько вариантов технической реализации мультиконвейерных вычислений. Простейший способ – это создание специализированной вычислительной структуры, топологически подобной графу задачи ***G(Q,X)*** (Рис. 4). Такой способ обеспечивает минимизацию непродуктивных накладных расходов при организации мультиконвейерных вычислений и, соответственно, высокую реальную производительность системы. Однако, с другой стороны, понятно, что создание уникального многопроцессорного вычислителя потребует больших временных и финансовых затрат.

Существенно меньших затрат требует реализация мультиконвейерных вычислений на ба-

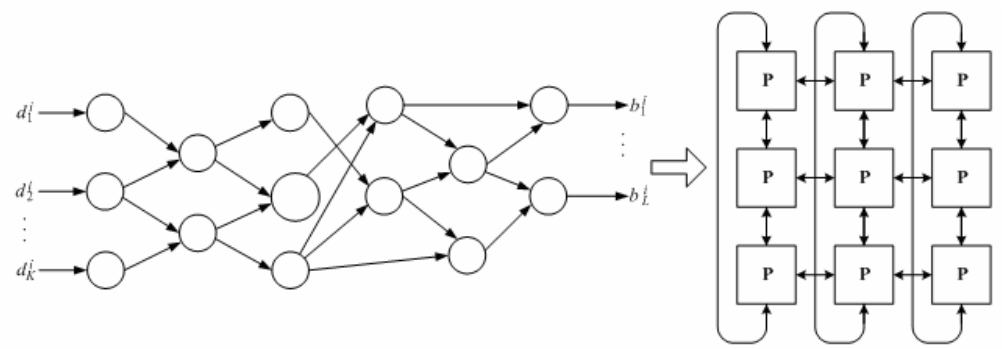

Рис. 5. Проблема отображения графа задачи $G(Q,X)$ в кластерную МВС с жесткой архитектурой

зе МВС кластерного типа, построенных с использованием коммерчески доступных вычислительных узлов и коммутационных сетей. Однако при этом возникают большие непродуктивные накладные расходы, вызванные несоответствием между топологией графа задачи ***G(Q,X)*** и жесткой архитектурой кластерной МВС (Рис. 5). Это, в свою очередь, ведет к резкому снижению реальной производительности кластерных МВС при реализации мультиконвейерных вычислений, которая зачастую не превышает 5÷10% от пиковой производительности [2].

Указанные недостатки специализированных и кластерных МВС позволяет преодолеть концепция создания многопроцессорных вычислительных систем с реконфигурируемой архитектурой [1,3,4]. Суть этой концепции заключается в том, что пользователю предоставляется возможность адаптации архитектуры реконфигурируемой вычислительной системы (РВС) под структуру информационного графа решаемой задачи, что позволяет по сравнению с кластерной МВС существенно снизить непродуктивные накладные расходы в процессе вычислений и, как следствие, повысить реальную производительность системы, а по сравнению со специализированной МВС существенно сократить финансовые и временные затраты на создание мультиконвейерной структуры.

Несмотря на то, что концепция РВС развивается достаточно давно и в 80–90-е годы XX века был создан ряд прототипов вычислительных систем с реконфигурируемой архитектурой, ее широкое применение долгое время сдерживалось отсутствием соответствующей элементной базы. Элементная база РВС должна удовлетворять целому ряду требований, основными среди которых являются: возможность аппаратной (структурной) реализации крупных фрагментов вычислений; возможность программирования (реконфигурации) различных вычислительных структур, отвечающих текущей задаче; наличие систем автоматического проектирования, программирования и реконфигурирования вычислительных структур; приемлемая стоимость и т.д.

Всем этим требованиям отвечают программируемые логические интегральные схемы (ПЛИС) со сверхвысокой степенью интеграции (в английской аббревиатуре FPGA – Field Programmable Gates Array). В ПЛИС изначально заложены возможности реконфигурирования их внутренней структуры, и поэтому они наилучшим образом соответствуют концепции реконфигурируемых вычислительных систем.

В настоящее время ПЛИС находят все более широкое применение в составе высокопроизводительных МВС (суперкомпьютеров) для увеличения их реальной производительности. Приведем два характерных примера. В 2007 году фирма SGI выпустила МВС, содержащую 35 модулей RC100, каждый из которых имеет по две ПЛИС. По данным SGI компьютер решает задачи биоинформатики в 900 раз быстрее, чем кластер из 68 узлов на базе процессоров Opteron. Новой разработкой SGI является модуль SGI RC200-blade, в составе которого используются ПЛИС Altera Stratix III и технология прямого подключения к шине FSB процессоров Xeon под названием Intel QuickAssist [5].

Компания Cray в 2009 году выпустила суперЭВМ Cray XT5h [6]. Он объединяет в единую систему скалярные и векторные процессо-

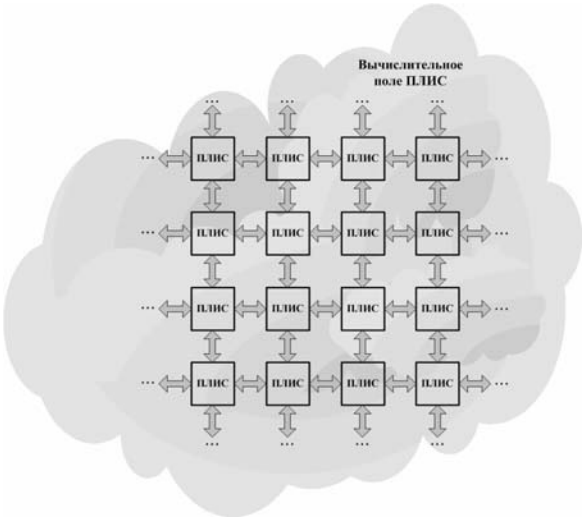

Рис. 6. Организация вычислительного поля ПЛИС

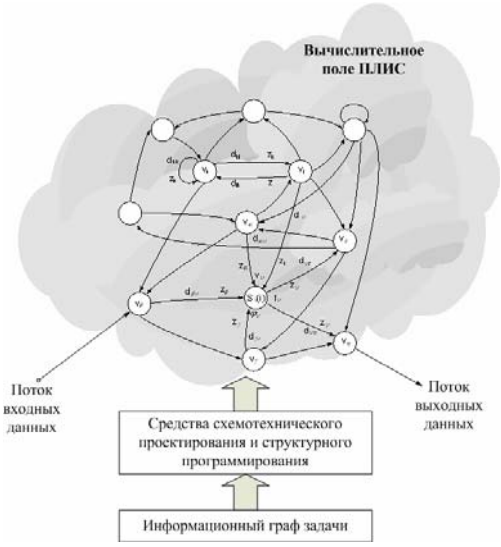

Рис. 7. Отображение графа задачи в вычислительное поле ПЛИС

ры, а также процессоры на основе ПЛИС. Создание такого суперкомпьютера компания Cray обосновывает тем, что скалярные процессоры хорошо подходят для вычислений, не требующих большого числа обращений к памяти, а ПЛИС и векторные элементы предназначены для использования в тех задачах, для которых характерны высокая загруженность памяти и большие объемы обрабатываемых данных. Поэтому их совместное применение может позволить эффективнее решать современные сложные вычислительные задачи. Пиковая производительность суперЭВМ Cray XT5h составляет 2331.00 Тфлопс, и он занимает первую строчку списка TOP500.

Однако практически во всех МВС, в которых используются ПЛИС, они, как правило, выступают в роли сопроцессоров к стандартным вычислительным узлам, реализованным на универсальных микропроцессорах. Концепция же построения РВС открывает более широкие перспективы использования ПЛИС в качестве элементной базы для создания больших вычислительных полей, в рамках которых могут создаваться «мелкозернистые» мультиконвейерные вычислительные структуры, адаптированные под структуру графа $G(Q,X)$ решаемой задачи. В рамках большого вычислительного поля можно осуществить создание и «тонкую» настройку различных вычислительных структур, адекватных информационной структуре решаемой задачи. Именно такой подход обеспечивает возможности достижения максимума реальной производительности системы при решении конкретной потоковой задачи.

Основные принципы создания РВС на основе полей ПЛИС заключаются в следующем.

1. Некоторое множество ПЛИС объединяется в единое вычислительное поле (Рис. 6).

2. С помощью средств автоматического проектирования и структурного программирования в поле ПЛИС создается мультиконвейрная вычислительная структура, адекватная графу $G(Q,X)$ решаемой задачи (Рис. 7).

3. Если аппаратные ограничения поля ПЛИС не позволяют отобразить весь граф решаемой задачи, то последний предварительно разрезается на непересекающиеся подграфы таким образом, чтобы каждый из этих подграфов мог быть структурно реализован в имеющемся поле ПЛИС. Далее организуется последовательная процедура структурной реализации этих подграфов в поле ПЛИС (Рис. 8). При этом очевидно, что чем больше будет поле ПЛИС, тем на меньшее число подграфов необходимо будет разрезать граф задачи, что, в свою очередь, позволит сократить непродуктивные накладные расходы, связанные с реконфигурацией поля ПЛИС и, соответственно, повысить реальную производительность РВС при решении данной задачи.

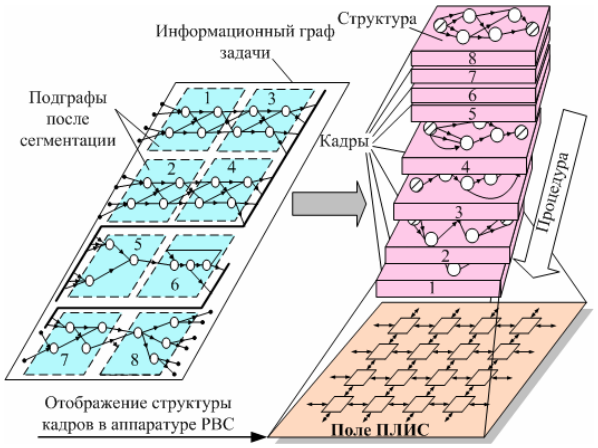

Рис. 8. Структурно-процедурная организация вычислений в поле ПЛИС

Общие принципы организации и функционирования РВС на основе ПЛИС подробно изложены в [2-4].

Базовый модуль РВС

РВС, предназначенная для решения сложных потоковых задач, должна содержать сотни и тысячи ПЛИС высокой степени интеграции, объединенных в общее вычислительное поле. Понятно, что размещение такого количества ПЛИС на одной конструктивной единице (печатной плате) невозможно. Эта проблема решается путем модульного построения вычислительного поля РВС на основе унифицированных конструктивных единиц – базовых модулей. Базовый модуль представляет собой плату, содержащую фрагмент вычислительного поля ПЛИС, а также вспомогательные элементы: интерфейсы межмодульного обмена, блоки распределенной памяти, блоки вторичного питания, подсистему синхронизации, узлы управления, сетевые интерфейсы и т.п. Базовый модуль сам по себе представляет малогабаритную РВС, способную совместно с ПК решать пользовательские задачи, а их комплексирование обеспечивает возможности создания РВС требуемой производительности.

Обобщенная структура базового модуля РВС приведена на Рис. 9.

Основные вычислительные возможности базового модуля сосредоточены в его вычислительном поле, которое содержит некоторое множество ПЛИС большой степени интеграции. Блоки распределенной памяти выполняются на типовых микросхемах ОЗУ SRAM или SDRAM необходимого объема и быстродействия. Управление всеми ресурсами базового модуля осуществляется через контроллер базового модуля (КБМ). Через этот контроллер производится загрузка входных данных и выгрузка результатов решения, загрузка фрагментов параллельных прикладных программ в контроллеры распределенной памяти.

Связи между ПЛИС в пределах вычислительного поля должны обеспечивать скоростную передачу данных между частями вычислительных структур, расположенных в других микросхемах многокристальной реализации вычислительной структуры. Поэтому связи между ПЛИС должны иметь минимальную длину и выполняются на основе скоростных интерфейсов, таких как LVDS (Low Voltage Differential Signaling – дифференциальные сигналы низкого напряжения) или RocketIO.

Преимуществами скоростных интерфейсов являются: низкая потребляемая мощность выходных каскадов, низкий уровень создаваемых электромагнитных излучений, невосприимчивость к синфазным электромагнитным помехам и наличие в микросхемах современных ПЛИС аппаратной поддержки для организации высокоскоростных передач данных.

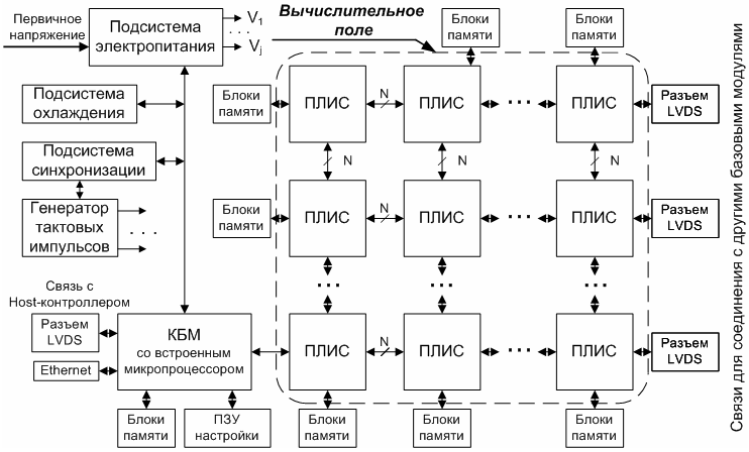

Рис. 9. Структура базового модуля РВС

Физически каждая линия скоростного интерфейса представляет собой пару дифференциальных полосковых проводников, с обоих концов подключенных к специальным выводам микросхем. Темп передачи данных по каждой двухпроводной линии, в зависимости от реализации, составляет до 5 Гбит в секунду.

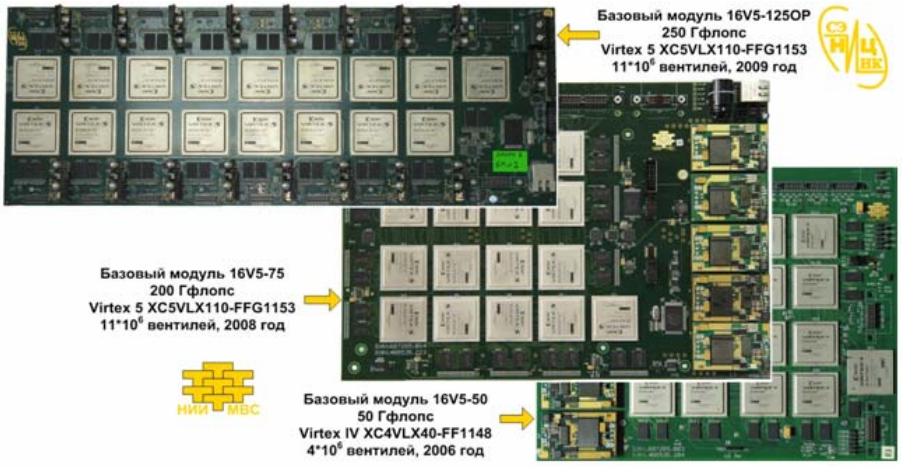

Рис. 10. Базовые модули РВС на основе ПЛИС

Примеры реализации РВС

Некоторые типы базовых модулей РВС, созданные в НИИ многопроцессорных вычислительных систем Южного федерального университета и Научно-исследовательском центре суперЭВМ и нейрокомпьютеров (г. Таганрог), показаны на Рис. 10.

На основе базового модуля 16V5-75 создано семейство РВС различной производительности - от 50 Гфлопс (16 ПЛИС Virtex 5) до 6 Тфлопс (1280 ПЛИС Virtex 5) (Рис. 11) [4, 7].

Модели РВС-5 и РВС-1Р, входящие в состав семейства, содержат соответственно пять и одну 19’ стойку СТ-1Р. Каждая стойка включает четыре блока РВС-0.2-ВБ высотой 6U, имеющих в своем составе до четырех базовых модулей 16V5-75 с пиковой производительностью до 200 Гфлопс каждый на данных с одинарной точностью.

Вычислительное поле РВС-1Р содержит 256 ПЛИС Virtex 5, а РВС-5 – 1280 ПЛИС Virtex 5, объемом 11 миллионов эквивалентных вентилей каждая. Следует отметить, что связи внутри вычислительного поля базового модуля 16V5-75 выполнены по стандарту LVDS и обеспечивают суммарную скорость передачи данных свыше 3 Тбит в секунду на частоте 1200 МГц.

В настоящее время в НИИ МВС ЮФУ выполняется проект по созданию модульно-наращиваемой РВС с производительностью 20 Тфлопс, содержащей 1536 ПЛИС Virtex 5 в 19’ стойке. Основу системы составляют базовые модули 16V5-1250P с производительностью 250 Гфлопс каждый, содержащие 16 ПЛИС Virtex 5 и соединяемые между собой в единый вычислительный ресурс с помощью скоростных LVDS-каналов. Четыре базовых модуля 16V5-1250P объединяются в конструктиве 1U блока «Орион» (Рис. 12). В отличие от базового модуля 16V5-75, где связи для объединения имеются только у четырех ПЛИС из шестнадцати, в базовом модуле 16V5-125OP все ПЛИС вычислительного поля имеют LVDS-каналы для наращивания вычислительного поля. Это обеспечивает большую скорость обмена информацией между базовыми модулями в составе блока «Орион».

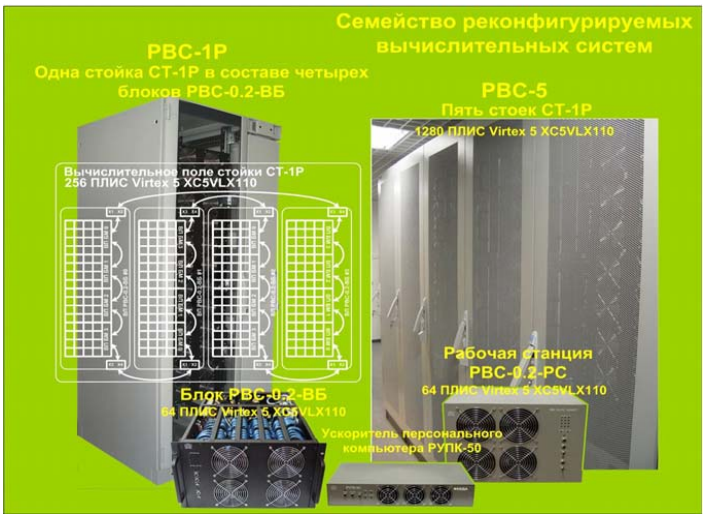

Рис. 11. Семейство РВС на основе полей ПЛИС

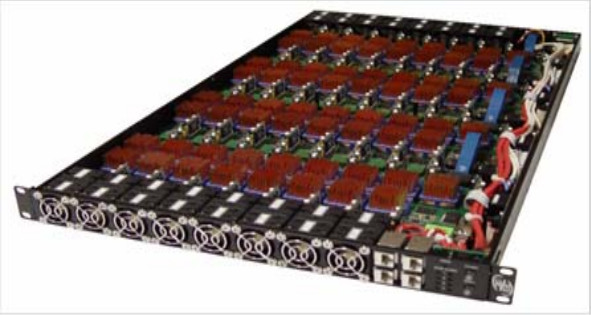

Рис. 12. Вычислительный блок «Орион»

В Табл. 1 приведены значения реальной производительности блока РВС-0.2-ВБ и блока «Орион» на задачах различных классов: символьной обработки, цифровой обработки сигналов, линейной алгебры и математической физики.

Из Табл. 1 видно, что вычислительный блок «Орион», имея практически одинаковый ресурс вычислительного поля ПЛИС с блоком РВС-0.2-ВБ, несколько превосходит последний по производительности. Увеличение производительности блока «Орион» объясняется двумя факторами. Первый из них – это большая пропускная способность каналов, объединяющих вычислительные поля базовых модулей. Второй фактор – б*о*льшее количество каналов распределенной памяти, подключенных к вычислительному полю: 80 каналов в блоке РВС-0.2-ВБ и 128 каналов в блоке «Орион». При этом величины производительностей этих блоков, приходящиеся на единицу объема, существенно различаются в пользу блока «Орион». Это связано с более рациональной компоновкой блока, объем которого составляет примерно 18% от общего объема блока РВС-0.2-ВБ или 32% от объема его вычислительной части. В Табл. 2 приведены значения удельной производительности блока РВС-0.2-ВБ и блока «Орион» на задачах символьной обработки и на задачах, использующих данные одинарной точности с плавающей запятой. Высокая удельная производительность блока «Орион» имеет существенное значение при построении многостоечных высокопроизводительных РВС.

Блок «Орион», как и стойка СТ-1Р, не имеют внешних быстрых LVDS-каналов для дальнейшего наращивания ресурса вычислительных полей. Наращивание производительности систем, составленных из блоков «Орион» и стоек СТ-1Р, выполняется с использованием сетевых технологий посредством каналов Ethernet, как это реализовано, например, в РВС-5 [3, 4, 7].

Табл. 1. Производительность блоков РВС-0.2-ВБ и вычислительного блока «Орион»

Задачи	Символьная обработка, (Операций/ сек)	ЦОС, (Гфлопс)	Линейная алгебра, (Гфлопс)	Мат. Физика (Гфлопс)
РВС-0.2-ВБ	$116{,}2*10^{12}$	647,7	423,2	535,2
«Орион»	$116{,}2*10^{12}$	809,6	528,7	669,0

Табл. 2. Значения удельной производительности блоков

Производительность	Символьная обработка (операций в сек/дм^3)	Задачи общего назначения (Гфлопс/дм^3)
РВС-0.2-ВБ	$2{,}4*10^{12}$	16,7
«Орион»	$7{,}5*10^{12}$	64,9

Программирование РВС

Процесс программирования РВС существенно отличается от программирования МВС традиционной архитектуры. Его можно условно разделить на две составляющие: программирование структурное, в результате которого создаются необходимые мультиконвейерные вычислительные структуры в поле логических ячеек ПЛИС, и программирование процедурное – программирование в традиционном смысле, заключающееся в организации вычислительного процесса в РВС. При этом структурное программирование поля ПЛИС вызывает у пользователей наибольшие трудности. Это связано с тем, что пользователи традиционно привыкли программировать только организацию вычислительного процесса, опираясь на неизменяемую аппаратную поддержку средств вычислительной техники, в то время как для программирования мультиконвейерных вычислительных структур в поле ПЛИС требуются совершенно другие навыки, а именно − навыки схемотехника.

Трудности программирования РВС позволяет преодолеть специальный программный комплекс средств разработки прикладных программ, предоставляющий пользователю возможности создания приложений без привлечения специальных знаний в области схемотехники ПЛИС [3,8]. Про-

граммный комплекс предоставляет пользователю следующие возможности:

– программирование как структурной, так и процедурной составляющих на языке высокого уровня COLAMO;

– создание и изменение мультиконвейерных вычислительных структур для прикладных задач без участия высококвалифицированного схемотехника;

– обеспечение совместимости и переносимости прикладных программ между РВС различных архитектурных платформ;

– масштабирование прикладных программ при увеличении ресурса;

– удаленное использование и удаленный мониторинг вычислительных ресурсов РВС.

Созданный комплекс программного обеспечения по функциональному назначению разделяется на комплекс средств разработки прикладных программ и комплекс средств управления и администрирования вычислительных ресурсов РВС.

Средства разработки прикладных программ содержат: интегрированную среду разработки прикладных программ Argus IDE, поддерживающую языки программирования Argus и COLAMO; транслятор языка программирования РВС высокого уровня COLAMO; транслятор языка ассемблера Argus; среду разработки схемотехнических решений Fire!Constructor для синтеза масштабируемых параллельно-конвейерных структур, оперирующую библиотекой IP-ядер (библиотека вычислительных узлов и интерфейсов).

На Рис. 13 показана обобщенная схема создания прикладной программы для РВС.

Средствами языка программирования высокого уровня COLAMO создается как структурная, так и процедурная составляющие прикладной программы. При этом обеспечиваются создание и изменение вычислительных структур для прикладных задач без участия высококвалифицированного схемотехника и переносимость прикладных задач между РВС различных архитектурных платформ за счет использования библиотек описаний базовых модулей, блоков, РВС (паспорт РВС).

Транслятор COLAMO в процессе трансляции создает четыре компонента параллельной прикладной программы: управляющий, процедурный, потоковый и структурный.

Управляющий компонент транслируется в язык Паскаль и исполняется на управляющих контроллерах, входящих (в зависимости от ап-

Рис. 13. Обобщенная схема создания прикладной программы для РВС

паратной платформы РВС) в состав или базовых модулей и/или вычислительных блоков. Управляющий компонент отвечает за загрузку конфигурации ПЛИС вычислительного поля, загрузку исходных данных и выгрузку результатов из распределенной памяти вычислительных полей, за потоки данных в РВС на уровне сетевых обменов между блоками или базовыми модулями.

Процедурный и потоковый компоненты транслируются в язык ассемблера Argus и, наряду с программой управления, являются составными элементами параллельной пользовательской программы. Процедурная и потоковая составляющие исполняются контроллерами распределенной памяти, задают процедуру смены кадров и организуют параллельные потоки данных в мультиконвейерных вычислительных структурах.

Наиболее интересной особенностью транслятора COLAMO является функция выделения структурной составляющей COLAMO-программы. Транслятор создает структурную составляющую в объектном представлении, которая представляет собой описание информационного графа вычислений и автоматически передается в среду разработки масштабируемых схемотехнических решений Fire!Constructor, синтезирующую вычислительные структуры для всех ПЛИС вычислительного поля.

Среда Fire!Constructor является принципиально новым, не имеющим аналогов программным продуктом, который позволяет полностью исключить схемотехника из процесса создания прикладных программ для РВС. Исходными данными для Fire!Constructor являются:

– структурная составляющая прикладной COLAMO-программы в объектном представлении, выделенная транслятором COLAMO;

– библиотека вычислительных узлов и интерфейсов (IP-ядер), используемых для отображения информационного графа задачи в аппаратуру вычислительного поля РВС; IP-ядра вычислительных узлов и интерфейсов в виде описаний на языке VHDL, составляющие библиотеку, создаются схемотехниками на этапе разработки РВС и подключаются к среде Fire!Constructor;

– библиотека описаний базовых модулей, блоков и РВС (паспорт РВС); наличие библиотеки паспортов РВС позволяет среде Fire!Constructor создавать платформо-независимые многокристальные решения в пределах множества ПЛИС вычислительных полей базовых модулей, блоков и стоек РВС, в том числе и для РВС с гетерогенной структурой, состоящих из разнородных базовых модулей и блоков.

Fire!Constructor в автоматическом режиме синхронизирует информационные потоки в пределах многокристальной реализации мультиконвейерной вычислительной структуры. При этом учитываются особенности реализации вычислительных полей РВС: количество и типы связей между ПЛИС в вычислительных полях базовых модулей и связи между вычислительными полями базовых модулей, блоков и стоек.

Результатом работы Fire!Constructor является следующая информация для каждой ПЛИС вычислительного поля РВС:

– описание на языке VHDL фрагмента мультиконвейерной вычислительной структуры в пределах конкретной ПЛИС в виде файла *.vhd;

– соответствие логических имен внешних сигналов фрагмента мультиконвейерной вычислительной структуры физическим выводам той же ПЛИС в виде файла *.ucf.

Для формирования загрузочных конфигурационных файлов *.bit всех ПЛИС вычислительного поля результаты работы Fire!Constructor отправляются в Xilinx ISE, где и происходит их формирование.

Для программирования РВС необходимо загрузить во все ПЛИС вычислительного поля подготовленные конфигурационные файлы и загрузить компоненты параллельной программы в управляющие процессоры и контроллеры распределенной памяти.

Созданный комплекс программного обеспечения позволяет разрабатывать эффективные прикладные программы для РВС при решении потоковых задач различных предметных областей, обеспечивает удобство программирования и автоматизированный перенос структурного решения с одной архитектуры РВС на другую. Комплекс программного обеспечения в 2-3 раза снижает стоимость и в 3-5 раз сокращает время разработки прикладного решения по сравнению с

подходом, в котором создание вычислительной структуры РВС производится схемотехником.

Заключение

Реконфигурируемые мультиконвейерные вычислительные системы на основе полей ПЛИС являются эффективным средством для решения потоковых задач обработки информации и управления. В отличие от традиционных МВС они предоставляют пользователю возможность создавать в базовой архитектуре поля ПЛИС виртуальные специализированные мультиконвейерные вычислители, адекватные графу решаемой задачи. Это, в свою очередь, обеспечивает высокую эффективность мультиконвейерных вычислений и близкий к линейному рост производительности РВС при наращивании вычислительного ресурса.

Практические применения РВС при решении целого ряда прикладных задач обработки информации и управления в таких областях как:

– цифровая обработка сигналов;

– управление сложными мехатронными объектами ядерной энергетики и нефтегазодобывающего комплекса;

– топографические исследования приповерхностных слоев Земли;

– молекулярное моделирование;

– криптология

и т.п. показали их высокую эффективность в сравнении с традиционными МВС кластерного типа по таким техническим характеристикам, как «реальная производительность/пиковая производительность» (в 5-10 раз), «реальная производительность/потребляемая мощность» (в 10-16 раз) и «реальная производительность/объем (в 10-15 раз).

Литература

1. Каляев А.В., Левин И.И. Модульно-наращиваемые многопроцессорные системы со структурно-процедурной организацией вычислений. – М.: Янус-К, 2003. – 380 с.
2. Аладышев О.С., Дикарев Н.И., Овсянников А.П. и др. СуперЭВМ: области применения и требования к производительности – Известия ВУЗов. Электроника, 2004. – №1. – С. 13-17.
3. Каляев И.А., Левин И.И., Семерников Е.А., Шмойлов В.И. Реконфигурируемые мультиконвейерные вычислительные структуры /Изд. 2-е, перераб. и доп. / Под общ. Ред. И.А. Каляева. – Ростов-на-Дону: Изд-во ЮНЦ РАН, 2009. – 344 с.
4. Дмитренко Н.Н., Каляев И.А., Левин И.И., Семерников Е.А. Семейство многопроцессорных вычислительных систем с динамически перестраиваемой архитектурой. Вестник компьютерных и информационных технологий. – М.: Изд-во Машиностроение, 2009. – Ч. 1. - №6. – С.2-8. –Ч. 2.- № 7. – С.2-10.
5. http://www.ixbt.com/news/hard/index.shtml?09/60/92
6. http://www.parallel.ru/
7. Каляев И.А., Левин И.И., Семерников Е.А. Семейство вычислительных систем с высокой реальной производительностью на основе ПЛИС // Параллельные вычислительные технологии (ПаВТ’2010): Труды международной научной конференции (Уфа, 29 марта-2 апреля 2010 г.) [Электронный ресурс] – Челябинск: Издательский центр ЮУрГУ, 2010. – 723 с.С. 199-210.
8. Левин И.И. Язык параллельного программирования высокого уровня для структурно-процедурной организации вычислений // Труды Всероссийской научной конференции. М.: Изд-во МГУ, 2000, 108-112.

Каляев Игорь Анатольевич. Директор Научно-исследовательский институт многопроцессорных вычислительных систем имени академика А.В. Каляева Южного федерального университета (НИИ МВС ЮФУ). Окончил Таганрогский радиотехнический институт им. В.Д. Калмыкова в 1980 году. Доктор технических наук, член-корреспондент РАН. Автор 230 научных работ. Область научных интересов: многопроцессорные управляющие комплексы, реконфигурируемые вычислительные системы, нейроподобные структуры, интеллектуальная робототехника. E-mail: kaliaev@mvs.sfedu.ru.

Левин Илья Израилевич. Заместитель директора по науке Научно-исследовательского института многопроцессорных вычислительных систем имени академика А.В. Каляева Южного федерального университета (НИИ МВС ЮФУ). Окончил Таганрогский радиотехнический институт им. В.Д. Калмыкова в 1984 году. Доктор технических наук. Автор 221 научной работы. Область научных интересов: многопроцессорные системы, реконфигурируемые вычислительные системы, высокопроизводительные системы, технологии параллельного программирования, ПЛИС.
E-mail: levin@ mvs.sfedu.ru.