Технология программирования вычислительных систем гибридного типа

И. А. Каляев[1], А. И. Дордопуло[2,*], И. И. Левин[3], В. А. Гудков[1], А. А. Гуленок[1]

[1]НИИ многопроцессорных вычислительных систем ЮФУ, Таганрог, Россия
[2]Южный научный центр РАН, Ростов-на-Дону, Россия
[3]Научно-исследовательский центр супер-ЭВМ и нейрокомпьютеров, Таганрог, Россия
[*]Контактный e-mail: scorpio@mvs.tsure.ru

Рассмотрены методы редукции производительности, составляющие наряду с языком программирования высокого уровня COLAMO основу разрабатываемой технологии ресурсонезависимого программирования вычислительных систем гибридного типа, содержащих реконфигурируемые и микропроцессорные вычислительные узлы. Язык программирования высокого уровня COLAMO позволяет описывать различные формы организации параллельных вычислений — от структурной формы, характерной для реконфигурируемых вычислительных узлов, через структурно-процедурную и мультипроцедурную до процедурной формы организации вычислений, используемой для унифицированных процессорных узлов. Преобразования между различными формами организации вычислений для быстрой автоматизированной адаптации прикладной программы под изменившуюся конфигурацию аппаратного ресурса вычислительной системы выполняются с помощью методов редукции производительности.

Ключевые слова: редукция производительности, язык программирования высокого уровня, программирование вычислительных систем гибридного типа, технология ресурсонезависимого программирования.

Введение

Одними из перспективных способов достижения высокой реальной производительности вычислительной системы при решении задачи являются адаптация архитектуры системы под структуру решаемой задачи и создание специализированного вычислительного устройства, эффективно реализующего алгоритм вычислений. Большинство практических задач требует совмещения в едином вычислительном контуре как последовательных, так и параллельных вычислительных фрагментов для эффективной реализации структурных и процедурных вычислений [1]. Решение этой проблемы многие разработчики видят в создании вычислительных систем с гибридной организацией вычислений, содержащих различные по архитектуре вычислительные узлы, объединенные каналами передачи данных и позволяющие реализовать структурные и процедурные вычисления в едином вычислительном контуре. Сочетание узлов различной архитектуры в одной вычислительной системе теоретически позволяет повысить ее реальную

производительность за счет возможности эффективной реализации как структурных, так и процедурных фрагментов вычислений на узлах различной архитектуры.

Широкое применение таких вычислительных систем для решения практических задач существенно ограничивается высокой сложностью их программирования, поскольку для эффективного использования архитектурных преимуществ всех вычислительных узлов программисту необходимо не только хорошо знать разные языки программирования и среды разработки для вычислительных узлов различных типов, но и самостоятельно синхронизировать вычисления в едином контуре.

1. Программирование вычислительных систем гибридного типа

По архитектуре используемых вычислительных узлов и типу организации вычислений современные высокопроизводительные вычислительные системы (ВС) можно отнести к одному из четырех основных типов: гомогенной, гетерогенной, гибридной и ВС гибридного типа.

Гомогенная ВС содержит одинаковые по архитектуре вычислительные узлы одного типа с одинаковыми типом организации вычислений и способом обработки информации. Наиболее ярким примером таких ВС являются широко распространенные кластерные вычислительные системы на основе серийно выпускаемых микропроцессоров.

Гетерогенная ВС содержит одинаковые по архитектуре вычислительные узлы, но разных типов с одинаковыми типом организации вычислений и способом обработки информации. Типичным примером таких систем являются системы, содержащие одновременно микропроцессоры и графические процессоры.

Гибридная ВС содержит различные по архитектуре вычислительные узлы с различными способами обработки информации и организации вычислений. Примером таких систем являются вычислительные устройства с цифровыми и аналоговыми подсистемами в едином управляющем контуре.

Вычислительная система гибридного типа (далее — ВСГТ) содержит различные по архитектуре вычислительные узлы с различным типом организации вычислений и одинаковым способом обработки информации. Вычислительные устройства с микропроцессорами и кристаллами программируемых логических интегральных схем (ПЛИС), связанные пространственной коммутационной системой в единый вычислительный контур, являются наиболее характерным примером таких систем.

Такие вычислительные системы гибридного типа могут содержать реконфигурируемые вычислительные узлы и узлы универсальных микропроцессоров, графических процессоров или ускорителей Intel Xeon Phi. Поэтому для программирования ВСГТ зачастую используются технологии программирования гетерогенных вычислительных систем (CUDA, OpenACC, OpenCL и др.), в основе которых лежат расширения языков программирования C, C++, FORTRAN, учитывающие архитектуру специализированного микропроцессорного узла. К основным недостаткам этих технологий программирования относятся следующие:

- существенное замедление при доступе к общей памяти и сложности при взаимодействии различных узлов ВСГТ между собой. Обмены между узлами организуются при помощи хост-процессора, поэтому при числе одновременно обменивающихся узлов ВСГТ более восьми наблюдается значительное падение реальной производительности;

- ориентация на программирование узлов ВСГТ одной архитектуры, а не множества различных устройств, поэтому разбиение задачи на фрагменты, каждый из которых выполняется на своем узле ВСГТ, возлагается на программиста и существенно зависит от текущей конфигурации ВСГТ. Любые изменения конфигурации приводят к необходимости корректировки прикладной программы;
- все технологии ориентированы на поддержку ряда архитектур с фиксированной внутренней структурой и заранее определенным набором команд, что усложняет масштабирование прикладной программы ВСГТ, особенно в случае сокращения доступного аппаратного ресурса;
- плохая переносимость готовых решений между вычислительными системами различных архитектуры и конфигурации и плохая масштабируемость программ.

Основной причиной указанных недостатков является существующий подход к программированию ВСГТ, при котором разбиение задачи на фрагменты осуществляется программистом с учетом количества доступных узлов ВСГТ и их архитектуры. Каждый фрагмент реализуется на выбранном вычислительном узле независимо от других фрагментов в структуре задачи, в результате чего изменение конфигурации ВС или исходного кода прикладной программы приводит к необходимости повторного разбиения задачи на фрагменты и создания локальных программ для каждого узла ВС.

Можно сформулировать основные особенности программирования современных ВСГТ, содержащих реконфигурируемые и микропроцессорные вычислительные узлы:

- программируемые логические интегральные схемы и микропроцессоры программируются на разных языках программирования независимо друг от друга;
- прикладная программа пишется под текущую конфигурацию ВСГТ, любое изменение структуры системы приводит к изменениям программы;
- синхронизация информационных потоков в структуре задачи возлагается на программиста;
- портирование прикладной программы на другую систему с похожей конфигурацией приводит к полной переработке программы;
- время программирования и отладки прикладной задачи для ВСГТ составляет от 6 до 12 мес.

Поэтому для программирования ВСГТ необходимы как средства описания различных вариантов организации вычислений в едином для различных архитектур языковом пространстве, так и средства трансляции параллельных прикладных программ, объединенные в технологию ресурсонезависимого программирования ВСГТ.

2. Технология ресурсонезависимого программирования ВСГТ

Под технологией программирования понимается совокупность обобщенных и систематизированных научных знаний об оптимальных способах (приемах и процедурах) проведения процесса программирования [2], обеспечивающего в заданных условиях получение программной продукции с заданными свойствами. Под технологией ресурсонезависимого программирования будем понимать совокупность знаний, методов, технологических приемов и средств, которая обеспечивает возможность гибкого изменения и масштабирования программы под новую вычислительную архитектуру или конфигурацию вычислительной системы.

Для обеспечения функционирования унифицированных процессорных и реконфигурируемых вычислительных узлов в едином контуре языковые средства разрабатываемой технологии ресурсонезависимого программирования должны обеспечивать возможность описания фрагментов с различными типами организации вычислений, в том числе работающих с различными частотой, скважностью и разрядностью обрабатываемых данных, а также содержать гибкие и мощные средства масштабирования фрагментов вычислений и числа операций в каждом фрагменте. Методы масштабирования должны обеспечивать возможность как увеличения параллелизма задачи (индукцию) при наращивании аппаратного ресурса, так и возможность уменьшения (редуцирования) при сокращении вычислительного ресурса.

Можно сформулировать ряд требований к технологии ресурсонезависимого программирования ВСГТ. Технология ресурсонезависимого программирования ВСГТ должна обеспечивать:

- поддержку различных форм организации вычислений (структурных, структурно-процедурных, мультипроцедурных вычислений и их различных сочетаний) в едином вычислительном контуре;
- возможность программирования унифицированных процессорных и реконфигурируемых вычислительных узлов в едином вычислительном контуре;
- программирование в едином языковом пространстве на языке программирования высокого уровня;
- возможность простого масштабирования прикладной программы для случаев как увеличения, так и сокращения доступного аппаратного ресурса;
- возможность простой и быстрой адаптации прикладной программы, разработанной для одной конфигурации ВСГТ, под другую конфигурацию, в том числе при добавлении узлов новой архитектуры;
- портацию ресурсонезависимой параллельной программы с помощью перетрансляции без существенной корректировки исходного текста.

Для реализации технологии ресурсонезависимого программирования ВСГТ необходимо выбрать язык программирования, который позволит описывать различные формы организации вычислений и программировать унифицированные процессорные и реконфигурируемые вычислительные узлы в едином вычислительном контуре.

Специализированные языки высокого уровня для программирования реконфигурируемых вычислительных систем (РВС) обладают привычным для большинства программистов персональных ЭВМ синтаксисом языка С и различаются семантическими особенностями вызова и использования операторов. В этих языках для описания параллельных процессов в РВС используется изначально последовательная парадигма языка С, семантика которого ориентирована на взаимодействие последовательных процессов, что не позволяет в полной мере использовать все возможности РВС при разработке параллельных программ на этих языках. Это приводит к семантическому разрыву между исходным информационным графом задачи, его описанием на языке высокого уровня и созданной транслятором схемотехнической реализацией. Результатом этого разрыва является существенное снижение эффективности параллельной программы — как правило, в 3–5 раз более низкая производительность по сравнению с приложениями, разработанными на языках HDL-группы (Hardware Definition Language — язык описания аппаратуры).

Перспективным в области программирования РВС является язык высокого уровня COLAMO [1], разрабатываемый в НИИ МВС ЮФУ. Он предназначен для описания

реализации параллельного алгоритма и создания на основе принципов структурно-процедурной организации вычислений специализированной вычислительной структуры в архитектуре РВС, выполняющей последовательную смену структурно (аппаратно) реализованных фрагментов информационного графа задачи, каждый из которых является вычислительным конвейером потока операндов. Таким образом, приложение (прикладная задача) для РВС состоит из структурной составляющей, представленной в виде аппаратно реализованных фрагментов вычислений, и процедурной составляющей, представляющей собой единую для всех структурных фрагментов управляющую программу последовательной смены вычислительных структур и организации потоков данных.

Для реализации вычислений на универсальных процессорах в языке COLAMO есть средства описания процедурной организации вычислений и возможность быстрого перехода от процедурной реализации вычислений на универсальных процессорах к структурной организации вычислений на реконфигурируемых вычислительных узлах. Конструкция Implicit позволяет неявным образом указать тип организации вычислений (структурный или процедурный) для фрагмента программы. Переопределение способа реализации конструкции Implicit позволяет прикладному программисту без существенного изменения текста параллельной программы перейти от структурной организации вычислений к процедурной и обратно, что дает возможность создавать единую прикладную программу на одном языке для всех узлов ВСГТ. Это позволяет рассматривать язык программирования высокого уровня COLAMO как основу для реализации технологии ресурсонезависимого программирования как реконфигурируемых вычислительных узлов, так и универсальных узлов ВСГТ.

Однако для эффективного программирования ВСГТ языковые средства должны иметь возможность описания фрагментов вычислений, работающих с различными частотой, скважностью и разрядностью обрабатываемых данных для обеспечения масштабирования как фрагментов, так и отдельных устройств не только при увеличении аппаратного ресурса, но и при его сокращении, а также возможность работы с данными переменной разрядности для эффективного использования аппаратного ресурса ВСГТ.

3. Редукция производительности как способ масштабирования вычислений в ВСГТ

Основой для простого масштабирования и адаптации прикладной программы в случае как увеличения, так и сокращения доступного аппаратного ресурса является редукция производительности [3] прикладной программы, под которой понимается пропорциональное сокращение производительности во всех без исключения фрагментах информационного графа задачи с возможным сокращением аппаратных затрат на реализацию вычислительной структуры. Редуцирование производительности параллельных программ позволяет изменять ключевые параметры параллельных программ (число задействованных вычислительных устройств, число каналов памяти, разрядность операндов, частоту и др.), поскольку структурная реализация задачи может привести к нехватке доступного аппаратного ресурса, что особенно актуально при переносе задачи на ВСГТ различных архитектур и конфигураций.

В отличие от традиционных технологий и методов программирования многопроцессорных вычислительных систем (MPI, CUDA, OpenAcc и др.), которые предполагают

распараллеливание базового информационного подграфа прикладной задачи в зависимости от конфигурации доступного вычислительного ресурса, для применения редукции производительности информационный граф задачи описывается в исходной параллельной форме с максимальным присущим задаче параллелизмом. В зависимости от числа доступных вычислительных узлов ВСГТ исходный информационный граф сокращается (редуцируется) с помощью специальных *редукционных* преобразований, которые сбалансированно понижают производительность всех фрагментов информационного графа и в ряде случаев сокращают занимаемый задачей аппаратный ресурс ВСГТ. Можно выделить следующие виды редукции производительности: а) по вычислительным устройствам, б) каналам памяти, в) разрядности, г) частоте.

Редукция производительности по вычислительным устройствам основана на сокращении одновременно выполняемых устройств, реализующих вычислительные операции. Принципы функционирования редукции производительности по вычислительным устройствам на примере операции быстрого преобразования Фурье (БПФ) проиллюстрированы на рис. 1 и 2. На рис. 1, *а* представлен исходный информационный граф операции быстрого преобразования Фурье, содержащий 16 вычислительных устройств, а на рис. 1, *б* — структурная реализация этой операции на РВС с условным заполнением ПЛИС (в намеренном допущении, что два вычислительных устройства занимают ресурс одного условного кристалла ПЛИС). При выполнении редукции производительности по функциональным устройствам осуществляется пропорциональное уменьшение числа одновременно работающих устройств, что приводит к сокращению производительности и занимаемого аппаратного ресурса. Степень редукции производительности задается коэффициентом, который определяет, во сколько раз сокращается производительность фрагмента, а для рассматриваемого примера также определяет, во сколько раз сокращается число функциональных устройств.

Результат редукции производительности по функциональным устройствам со степенью 2 для базовой операции быстрого преобразования Фурье представлен на рис. 2. Видно, что число функциональных устройств сократилось вдвое (до восьми), и результат операции получается не за один такт работы (для структурной реализации, см. рис. 1), а за два такта.

На языке высокого уровня один и тот же информационный граф можно описать различными способами, поэтому одно и то же редукционное преобразование может быть выполнено различными способами, которые будем называть *механизмами* редукции. Так, редукция производительности по вычислительным устройствам для рассмотренного примера может быть реализована в виде:

- однокадровой параллельной программы с использованием мультиплексоров и скважности, равной степени выполняемой редукции;
- многокадровой программы с использованием конструкции Let;
- мультикадровой программы с использованием конструкции MultiCadr.

Выбор механизма редукции осуществляет программист при описании редукции.

Одним из важнейших типов редукции, обеспечивающих возможность ресурсонезависимого программирования ВСГТ, является *редукция производительности по каналам памяти*, которая представляет собой согласованное сокращение числа одновременно используемых каналов памяти для фрагмента информационного графа прикладной задачи. Иллюстрация принципов функционирования редукции производительности по каналам памяти представлена на рис. 3 и 4.

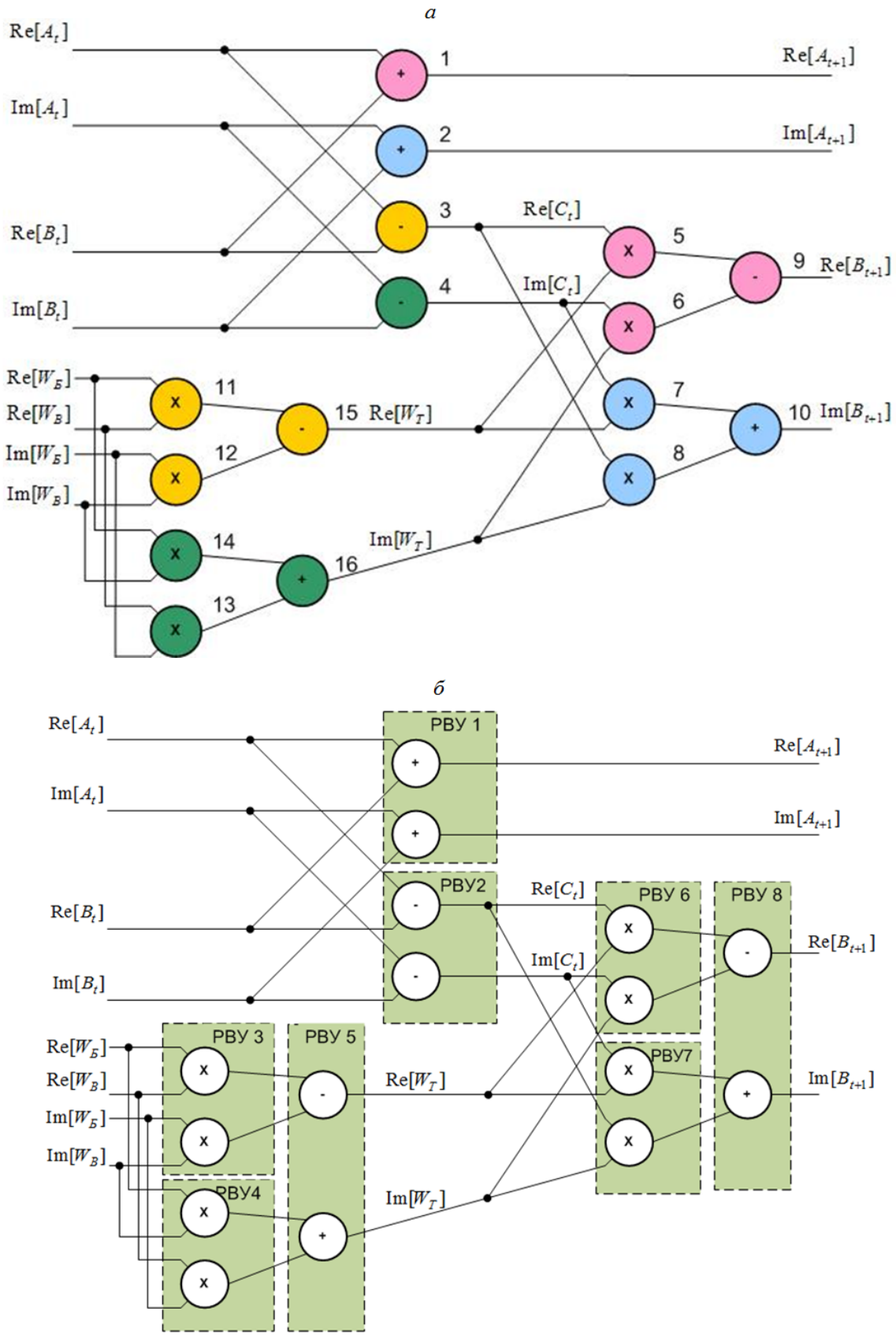

Рис. 1. Принципы функционирования редукции производительности на примере операции БПФ: *а* — исходный информационный граф операции БПФ; *б* — ее структурная реализация на РВС с условным заполнением ПЛИС

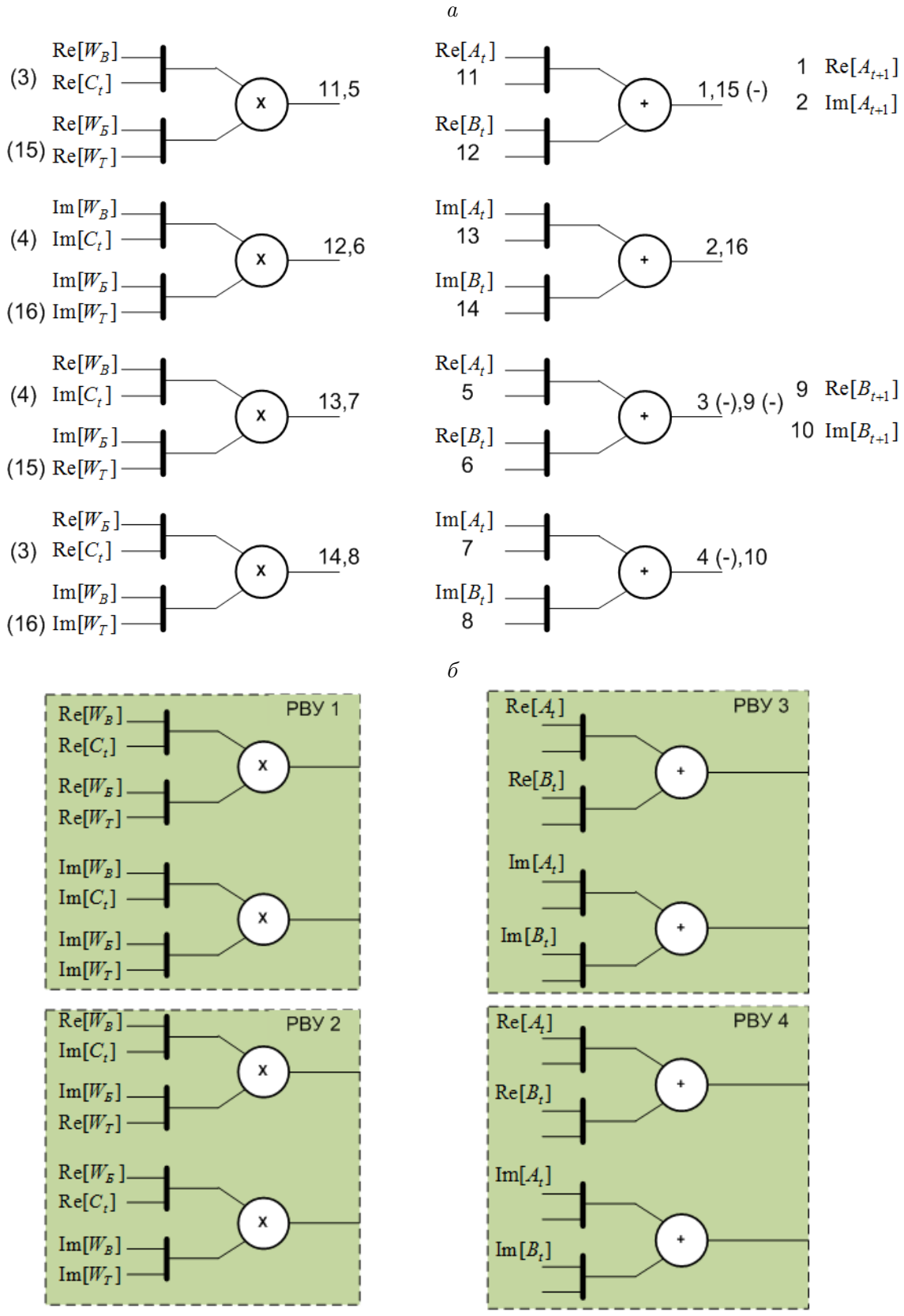

Рис. 2. Результат редукции производительности по функциональным устройствам со степенью 2 для базовой операции БПФ: *а* — редуцированный в два раза информационный граф; *б* — структурная реализация операции БПФ на РВС с условным заполнением ПЛИС

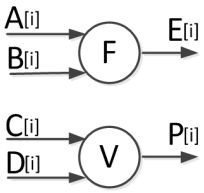

Рис. 3. Исходный информационный граф редуцируемого фрагмента

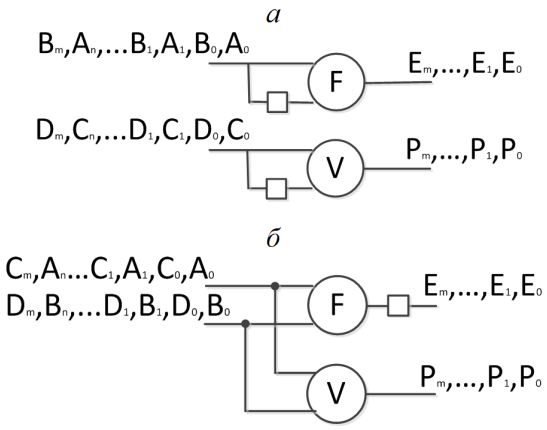

Рис. 4. Результат редукции производительности по каналам памяти со степенью 2: с размещением данных для каждого вычислительного устройства в одном канале памяти (*а*), в разных каналах (*б*)

На рис. 3 представлен фрагмент исходного информационного графа, содержащий четыре входных канала, а на рис. 4 — результат выполнения редукции производительности по каналам памяти со степенью 2. При выполнении редукции производительности по каналам памяти число одновременно работающих устройств не изменяется, но время обработки потоков данных возрастает в два раза, что в два раза сокращает производительность фрагмента.

Редукция производительности по разрядности направлена на сокращение не устройств в вычислительной структуре, а разрядности обрабатываемых данных за счет использования устройств меньшей разрядности, что приводит к увеличению времени обработки и уменьшению аппаратных затрат на реализацию вычислительной структуры. При выполнении редукции производительности по разрядности управление информационными потоками данных осуществляется мультиплексором, а сами данные подаются в вычислительную структуру со скважностью, значение которой равно степени выполняемой редукции.

Редукция производительности по частоте предназначена для увеличения времени обработки потока данных пропорционально степени выполняемой редукции за счет кратного сокращения частоты работы фрагмента, при этом скважность подачи данных в вычислительную структуру остается неизменной.

Редукция производительности является служебным редукционным преобразованием, которое используется не самостоятельно, а применяется для согласования скоростей обработки между редуцированными и нередуцированными фрагментами информационного графа в структуре прикладной задачи.

Для практического использования рассмотренных редукционных преобразований программист должен разметить директивами препроцессора те фрагменты параллельной программы на языке высокого уровня COLAMO, которые могут быть редуцированы. Формат описания директивы редукции может выглядеть следующим образом:

#Reduction of <*вид редукции*> <*степень редукции*>;

Блок операторов

EndReduction;

При трансляции параллельной программы препроцессор в соответствии с расставленными программистом директивами редукции в автоматическом режиме преобразует информационный граф прикладной программы под доступную конфигурацию ВСГТ, что позволит адаптировать программу к текущей конфигурации ВСГТ без существенной модернизации исходного текста программы.

В настоящее время проводятся исследования разработанного экспериментального образца препроцессора языка программирования высокого уровня COLAMO и его апробация при решении тестовых задач цифровой обработки сигналов, символьной обработки и мониторинга компьютерных сетей на вычислительной системе гибридного типа.

Заключение

Для эффективного программирования вычислительных систем гибридного типа предлагается использовать язык программирования высокого уровня COLAMO как основу разрабатываемой технологии ресурсонезависимого программирования ВСГТ. Описание в едином вычислительном контуре различных форм организации параллельных вычислений в совокупности с разработанными методами редукции производительности прикладной программы позволяет обеспечить ресурсонезависимость программирования ВСГТ, т. е. возможность простой и быстрой адаптации прикладной программы под изменившуюся архитектуру или конфигурацию ВСГТ. Предложенное расширение языка COLAMO позволяет рационально использовать ресурсы узлов с разной архитектурой при программировании ВСГТ и обеспечивает пользователя набором необходимых средств для быстрой разработки эффективных ресурсонезависмых масштабируемых параллельных программ в едином языковом пространстве, что снижает сложность программирования ВСГТ и повышает скорость разработки параллельных прикладных программ.

Благодарности. Работа выполнена при финансовой поддержке Министерства образования и науки РФ по Соглашению о предоставлении субсидии № 14.578.21.0006, уникальный идентификатор RFMEFI57814X0006.

Список литературы / References

[1] **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 н/Д.: Изд-во ЮНЦ РАН, 2009. 344 с.
Kalyaev, I.A., Levin, I.I., Semernikov, E.A., Shmoilov, V.I. Reconfigurable multipipeline computing structures / Ed. by I.A. Kalyaev. Rostov-on-Don: SSC RAS Publ., 2009. 344 p. (In Russ.)

[2] **Иванова Г.С.** Технология программирования: Учебник для вузов. М.: Изд-во МГТУ им. Н.Э. Баумана, 2002. 320 с.
Ivanova, G.S. Technology of programming: Uchebnik dlya vuzov. Moscow: Izdatel'stvo MGTU im. N.E. Baumana, 2002. 320 p. (In Russ.)

[3] **Левин И.И., Сорокин Д.А., Мельников А.К., Дордопуло А.И.** Решение задач с существенно-переменной интенсивностью потоков данных на реконфигурируемых вычислительных системах // Вест. компьютер. и информ. технологий. 2012. № 2. С. 49–56.
Levin, I.I., Sorokin, D.A., Melnikov, A.K., Dordopulo, A.I. Solving tasks with considerably variable data flow density on reconfigurable computer systems // Vestnik Komp'yuternykh i Informatsionnykh Tekhnologii. 2012. No. 2. P. 49–56. (In Russ.)

Поступила в редакцию 8 декабря 2015 г.

Programming technology for hybrid computer systems

Kalyaev, Igor A.[1], Dordopulo, Alexey I.[2,*], Levin, Ilya I.[3], Gudkov, Vjacheslav A.[1], Gulenok, Andrey A.[1]

[1]Scientific Research Institute of Multiprocessing Computing and Control Systems, Taganrog, 347928, Russia
[2]Southern Scientific Center RAS, Rostov-on-Don, 344006, Russia
[3]Supercomputers and Neurocomputers Research Center, Taganrog, 347900, Russia

*Corresponding author: Dordopulo, Alexey I., e-mail: scorpio@mvs.tsure.ru

Combination of nodes with different architecture within a hybrid computer system theoretically allows increasing of real performance and efficiency of calculations but considerably complicates programming of the system as a single complex, because programming paradigms of different computational architectures are different. Programming of hybrid computer systems requires a new technology which allows description in one and the same language and automatic transformation of various forms of organization of parallel calculations. The paper addresses the solution of this scientific problem. The paper deals with methods of reduction of the computer system performance for transformation of forms of organization of calculations and scaling of the parallel program by taking into account the variable hardware resource. While reducing the performance, the proportional reduction of the performance is made for all without exception fragments of the task information graph at the expense of reduction of the number of operations performed simultaneously, or reduction of the size of the processing data, or reduction of the number of used data channels. In several cases it leads to a reduction of the hardware resource required for the implementation of computational structure.

The basis of the developed technology of resource-independent programming is a high-level programming language COLAMO, which allows description of various forms of organization of parallel calculations such as structural, structural-procedural, multiprocedural and procedural forms for organization of calculations. Transformations between these various forms of organization of calculations for fast computer-aided adaptation of the application to the modified configuration of hardware resource of the computer system is performed with the help of a special software tool — a pre-processor

of the high-level programming language COLAMO, which reduces the fragments of the parallel program, selected by the developer with the help of special keywords.

At present, we analyze functionality and efficiency of the developed prototype for the pre-processor of the high-level programming language COLAMO and implementing tests of digital signal processing, symbol processing as well as computer network monitoring on a hybrid computer system.

Keywords: performance reduction, high-level programming language, programming of hybrid computer systems, technologies of resource-independent programming.

Acknowledgements. This paper was partly financially supported by the Russian Ministry of Education under grant No. 14.578.21.0006, ID RFMEFI57814X0006.

Received 8 December 2015