instructions // IEEE Robotics & Automation Magazine. 2008. June. Vol. 15, № 2. P. 27–36.

10. Yegenoglu F., Erkmen A. M., Stephanou H. E. On-line Path Planning Under Uncertainty. Proc. 27th IEEE Conf. Decis. and Contr. (Austin, Tex., Dec. 7–9, 1988). Vol. 2. New York, 1988. P. 1075–1079.

11. Отчет о госбюджетной научно-исследовательской работе Б6-12 за 1996 г. Алгоритмическое и программное обеспечение интеллектуальных манипуляционных роботов в условиях неопределенности. Теория и моделирование задач управления, планирования траекторий и действий манипуляционными интеллектуальными роботами в условиях неопределенности : отчет о НИР / Сиб. аэрокосмич. акад. ; Ильин В. А. Красноярск, 1997. № ГР 01.9.80.001231. Инв. № 02.9.8.0 0 00526.

12. Лопатин П. К. Компьютерная имитация управления манипуляционными роботами в неизвестной среде на основе точного и упрощенного алгоритмов // Мехатроника, автоматизация, управление. Приложение. 2006. № 8. С. 7–14.

13. Лопатин П. К. Алгоритм2 управления динамическими системами в неизвестной статической среде // Вестник СибГАУ. Вып. 4 (11). 2006. С. 28–32.

14. Лопатин П. К. Алгоритм управления динамическими системами в неизвестной статической среде // Мехатроника, автоматизация, управление. 2007. № 2. С. 9–13.

15. Lopatin P. K. Algorithm of a manipulator movement amidst unknown obstacles // Proc. of the 10 th Intern. Conf. on Advanced Robotics (ICAR 2001) (Budapest, Aug. 22–25, 2001). Hungary, 2001. P. 327–331.

P. K. Lopatin

ALGORITHM FOR AN OBJECT GRASPING BY A MANIPULATOR IN AN UNKNOWN STATIC ENVIRONMENT

An algorithm for n-link manipulating robot (MR) control in environment with unknown static obstacles is considered and theorem, which states that following the algorithm, MR for finite number of steps either grasps the object or gives proved conclusion that the object cannot be grasped in no configuration, is proved in the article.

Keywords: robot, unknown environment, obstacles, reachability.

УДК 004.4’2

А. Ю. Сыщиков

КОНЦЕПЦИЯ И ТЕХНОЛОГИЯ ПАРАЛЛЕЛЬНОГО ПРОГРАММИРОВАНИЯ НЕОДНОРОДНЫХ СИСТЕМ НА КРИСТАЛЛЕ

Неоднородные вычислительные системы на кристалле – это быстрорастущий сегмент распределенных параллельных вычислительных систем. Такие системы не могут эффективно программироваться вручную, тем не менее существуют только базовые инструменты для их программирования. Предлагается концепция и технология параллельного программирования неоднородных систем на кристалле. Концепция представляет визуальный подход к программированию, среднегранулярную организацию параллельных вычислений с последовательными блоками и динамику параллельных вычислений. Представлен разработанный пакет инструментов VIPE (VPL Integrated Programming Environment). VIPE включает в себя формальную математическую модель параллельных вычислений, визуальный язык параллельного программирования для среднегранулярного дизайна программ, транслятор и прекомпилятор схем программ и программный симулятор выполнения параллельных программ на неоднородной системы на кристалле.

Ключевые слова: система на кристалле, параллельное программирование, динамические вычисления, интегрированная среда разработки.

Многопроцессорные системы на кристалле (Multiprocessor Systems-on-Chips, СнК) – одно из ключевых современных приложений технологий сверхбольших интегральных схем. Требования приложений и ограничения реализации вынуждают создавать заказные, нестандартные, неоднородные архитектуры для оптимизации скорости вычислений, энергопотребления, стоимости, размеров и других критериев.

Одним из важнейших вопросов в области разработки СнК является технологии и средства программирования таких систем. В коллективах, создающих СнК, существуют программные средства, позволяющие разработать необходимый код для функционирования системы: компиляторы для ядер, загрузчики, возможно, отладчики. Но такой подход является фрагментарным: части системы программируются независимо, взаимосвязи между ними

скрыты, полная схема программного обеспечения (ПО) СнК является ненаблюдаемой. Отсутствие единой схемы приложения не позволяет обеспечить эффективный процесс сборки и тестирования приложения, простоту адаптации ПО для изменяющихся задач и конфигураций системы, сделать разработку ПО доступной не только их проектировщикам, но и заказчикам.

При таком подходе не удается объединить описание схемы программы (обычно графическое) и кодирование функций (обычно текстовое) в единую среду разработки. Это особенно актуально для разработки параллельных приложений, поскольку не позволяет контролировать ключевые вопросы: эффективность разработанного приложения; возможность модификации при изменении алгоритма или конфигурации платформы; взаимодействие одновременно выполняемых в системе задач, верификацию схемы приложения.

Концепция программирования неоднородных СнК. В данной статье предлагается унифицированная методология разработки параллельных программ для СнК, которая позволяет описывать схему программы независимо от архитектуры СнК, унифицирует механизм разработки ПО и обеспечивает переносимость ПО не только между однотипными системами, но и между различными классами систем.

Ключевыми особенностями предлагаемого подхода являются следующие:

– формальная модель параллельных вычислений как основа детерминированности и алгоритмической полноты вычислений;

– графическое представление и универсальный графический язык программирования для естественного описания схемы взаимодействия между компонентами программы с реализацией функциональных кодов модулей на традиционных языках программирования;

– интегрированная среда разработки приложений для комплексной унифицированной разработки ПО для неоднородных СнК.

Рассмотрим последовательно все уровни предлагаемой концепции.

Формальная АРП-модель параллельных вычислений. В основе предлагаемой концепции лежит формальная математическая модель динамических параллельных вычислений в распределенных архитектурах, АРП-модель [1; 2], которая является параллельной асинхронной моделью с полностью децентрализованным управлением. Общая АРП-модель определяется в терминах схемы параллельной программы. Схема представляется в виде направленного графа. Вершины графа представляют операторы и объекты-данные. Дуги графа соответствуют указателям, которые связывают операторы с соответствующими объектами-данными.

АРП-модель формально определяет наиболее общий тип параллельных вычислений – динамические параллельные вычисления. Схема параллельной программы изменяется, в общем случае, на каждом шаге вычислений – изменяется сам граф, а не только пометки вершин, как это делается, например, в модели потоков данных или сетях Петри.

АРП-модель предоставляет набор механизмов верификации схемы программы и построения согласованных вычислений. Верификация позволяет проверять поведение схемы программы «по построению» по ряду ключевых вопросов: зацикливание программы, наличие тупиков и др. Согласованные вычисления позволяют рассматривать выполнения параллельной программы, например, как последовательное для ее отладки и т. д.

Графический язык параллельного программирования VPL. Язык VPL [3] основан на формальной АРП-модели и предназначен для программирования динамических параллельных вычислений.

В общей АРП-модели каждая операторная вершина может порождать любой фрагмент схемы программы. В языке VPL возможность порождать запрограммированный пользователем фрагмент схемы программы предоставляется только специальному классу управляющих операторов. Управляющие операторы могут включать в себя другие операторы и объекты-данные, которые будут порождены при срабатывании оператора. Эти операторы являются основным инструментом управления параллельным вычислением в программах на языке VPL: они управляют структурой схемы параллельной программы в процессе ее работы. Данный инструментарий также реализует иерархическое структурирование параллельной программы и организацию вычислений, зависящих от значений данных, что является необходимым для алгоритмической полноты языка программирования.

Для остальных типов операторов язык программирования VPL определяет только общую, рамочную интерпретацию: такие операторы при срабатывании не могут порождать новые фрагменты схемы программы. Сама интерпретация будет задаваться программистом на последовательном языке программирования, например на языке C. Такой тип операторов называется «терминальными операторами». Терминальные операторы являются средством описания преобразований данных. Они читают данные, поступившие на вход, производят некоторую обработку данных (реализуют некоторую задачу обработки данных), формируют данные и записывают их на выход. Программа интерпретации оператора определяет функцию, которая будет запущена при вызове экземпляра оператора параллельной программы.

Концепция программирования в АРП-модели на языке VPL. Существенная особенность архитектуры СнК – разнородность, специализация и малая производительность вычислительных модулей. Вследствие этого программы для СнК являются среднегранулярными, т. е. выполняемые блоки кода соответствуют уровню функций и процедур обычных программ.

Для среднегранулярных программ естественным является разделение процесса разработки на программирование схемы параллельной программы и программирование ее компонент. Таким образом, должны использоваться два взаимодействующих уровня программирования:

– программирование схемы параллельной программы в виде сети операторов и объектов-данных в графической нотации языка VPL;

– программирования последовательных программ процессов на традиционных языках программирования (C, Embedded C, Assembler и т. д.).

Все взаимодействия операторов друг с другом или с объектами-данными явно представляются на уровне схемы параллельной программы (рис. 1). Таким образом, они могут контролироваться и верифицироваться.

Операторы взаимодействуют через объекты-данные (дуги в классическом графе потоков данных могут рассматриваться как частный случай объектов-данных). Объекты-данные являются абстракцией хранилищ и коммуникационных каналов, которые существуют вне операторов.

Важным свойством предлагаемой концепции является запрет на управление параллельными вычислениями из программ отдельных операторов. Это предохраняет взаимодействия в параллельной программе от ненаблюдаемости, а схему от хаотичности и невозможности верификации.

Интегрированная среда и практическое применение. Для создания ПО для неоднородных СнК в рамках предлагаемого подхода разработан пакет VIPE: VPL Integrated Programming Environment (рис. 2), который обеспечивает выполнение всего комплекса процедур от исходного алгоритма до получения готовой программы.

Комплекс разработки ПО VIPE состоит из следующих компонент:

– визуальная интерактивная среда разработки схем параллельных программ на языке VPL;

– построитель размещений элементов схем программ на вычислительных модулях неоднородной СнК и расписаний их выполнения [4];

– транслятор схемы программы и функциональных модулей в загрузочные модули программного симулятора платформы;

– компилятор объектных кодов функций и схемы программы, генерация выполняемого на неоднородной СнК кода, линковка загружаемых в СнК модулей;

– программный симулятор неоднородной СнК с интерпретатором VPL.

Комплекс VIPE обладает следующими ключевыми свойствами:

– поддерживает раздельное программирование схемы параллельной программы и последовательного кода ее элементов; поддерживает линовку модулей на традиционных языках программирования;

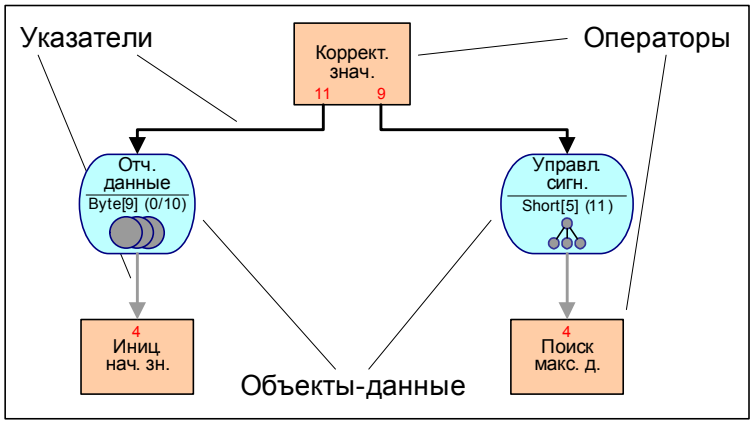

Рис. 1. Параллельная программа – сеть операторов и объектов-данных

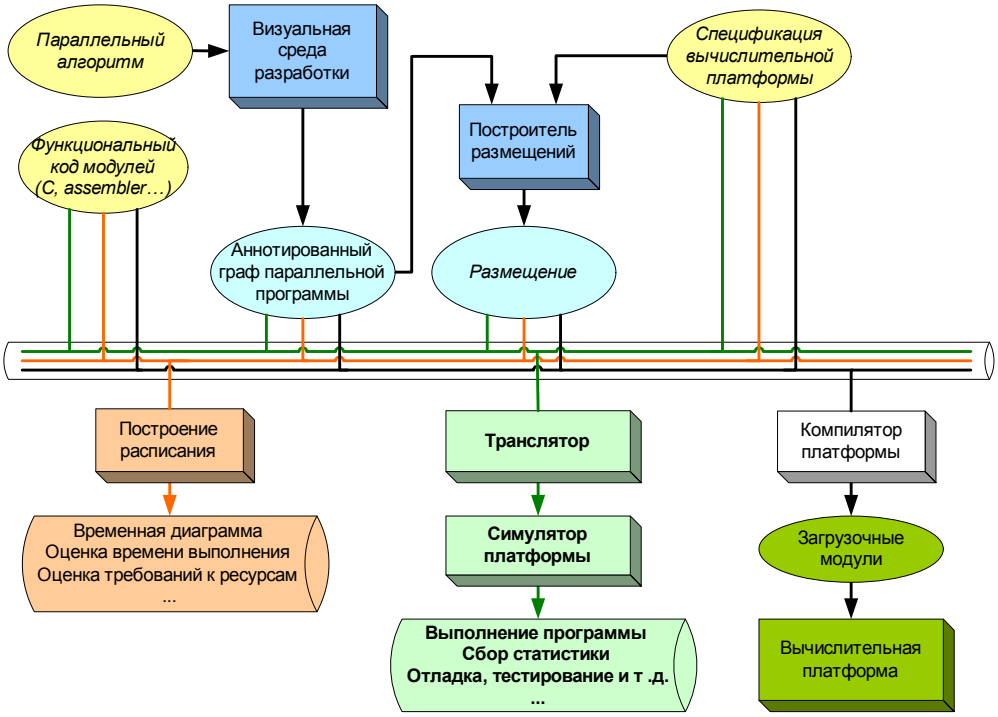

Рис. 2. Схема VPL Integrated Programming Environment

– поддерживает программирование как статических, так и динамических параллельных вычислений;

– поддерживает адаптацию размещения схемы программы под изменения спецификации платформы;

– производит набор трансформаций схемы программы;

– имеет расширяемую модульную структуру, позволяющую разрабатывать новые модули для поддержки различных последовательных языков и спецификаций платформ.

Пример программирования в комплексе VIPE. Рассмотрим применение предложенной концепции программирования неоднородных СнК на примере задачи расчета прохождения набора луча через оптическую систему, которая рассматривается как совокупность оптических поверхностей и источников лучей с заданными характеристиками. Лучи имеют следующие параметры: координаты по осям, углы отклонений по осям, интенсивность, номер поверхности, на которой находится луч.

В качестве объекта распараллеливания выберем лучи, так как при расчете их больше, чем поверхностей, а кроме того, их количество изменяется в ходе вычислений из-за эффектов отражения и поглощения.

Кратко алгоритм будет выглядеть следующим образом:

1. Загружаются из файлов параметры оптических поверхностей и характеристики начального набора лучей.

2. Запускается параллельный цикл, каждая итерация которого рассчитывает прохождение одного луча через одну поверхность:

2.1. Определяется номер поверхности для расчета.

2.2. Характеристики поверхности запрашиваются из хранилища.

2.3. Рассчитываются два новых луча: прямой и отраженный.

2.4. Каждый из них либо удаляется из расчета (ниже пороговой интенсивности), отправляется на выход системы (расчет луча окончен) или возвращается в буфер для дальнейшего расчета.

Поскольку предлагаемый подход к программированию является визуальным, рисовать отдельную схему алгоритма не требуется: результат программирования является одновременно как наглядной схемой алгоритма, так и исполняемой программой. Применяя стиль программирования «сверху вниз» на верхнем уровне получим обобщенную схему программы (рис. 3).

На этой схеме представлены два источника данных: загрузка оптических поверхностей и начального набора лучей. Данные об оптических поверхностях размещаются в локальной памяти процессорного элемента (ПЭ), на котором будет размещен процесс, обрабатывающий запросы на информацию о поверхностях («менеджер поверхностей»). Данные о начальном наборе лучей помещаются в общий буфер лучей для расчета, из которого лучи на обработку будут забираться по мере развертывания итераций цикла («параллельная обработка лучей»).

Цикл разворачивается в количестве одновременно работающих итераций от (1) до (количество лучей) в зависимости от реализации конкретной аппаратной платформы, наличия ресурсов и других факторов. Далее можно детализировать тело цикла независимо, не затрагивая схему верхнего уровня (рис. 4).

Каждая итерация получает на обработку один луч для расчета прохождения через очередную поверхность. Запрашиваются параметры поверхности («Выч-е номера поверхности»), вычисляются два новых луча, прямой и отраженный («Расчет лучей»). Далее проверяются их интенсивности и завершение прохождения («Проверка интенсивности»), результат проверки поступает на вход условного оператора («Выбор места назначения луча»), после чего они удаляются (ветвь «Drop»), попадают в выходной буфер (ветвь «Finish») или вновь в исходный буфер на новую обработку (ветвь «Continue»). Конечно, ветвь «Drop» не обязательно представлять на схеме. Она нарисована исключительно для наглядности отображения обработки.

И, наконец, общая схема (рис. 5). Следует отметить, что не обязательно обобщенная схема должна представляться на одном «листе» программы. Иерархия может быть отображена как на одном экране (на рисунке), так и на различных «листах» и в различных файлах.

Данный пример содержит иллюстрацию возможностей по описанию динамики вычислений и параллелизма в рамках предложенной концепции. Схема программы содержит статические компоненты (терминальные операторы), а также динамическое управление (условные операторы и циклы).

Размещение программ на неоднородных СнК. Отдельно остановимся на возможностях комплекса VIPE по эф-

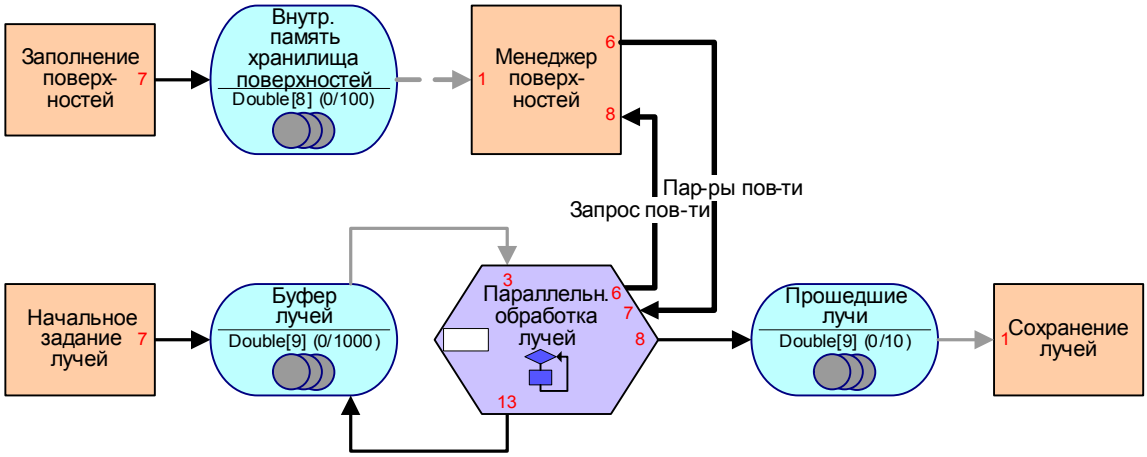

Рис. 3. Схема параллельной программы на верхнем уровне

фективной организации размещения параллельных программ на неоднородных СнК (рис. 6).

Как известно, общая задача размещения программ на однородных ВС в общем случае является NP-полной задачей. Для неоднородных СнК необходимо учитывать еще два фактора: ограничения на размещения и различные параметры выполнения, обусловленные неоднородностью вычислительных модулей и коммуникационной системы.

Комплекс VIPE обеспечивает размещение элементов схемы программы на модулях неоднородной СнК в следующих режимах: автоматическом, ручном и в автоматическом с установленными ограничениями. Представление схемы параллельной программы позволяет легко и наглядно задавать, анализировать и управлять ее размещением.

Таким образом, представлена целостная концепция параллельного программирования неоднородных СнК, описывающая все этапы разработки от неформального описания алгоритма до получения готовой программы, размещения и характеристик выполнения – и реализованная в виде интегрированной среды параллельного программирования VIPE.

Методология и инструментарий основаны на математической модели параллельных вычислений, которая гарантирует получение одинаковых результатов независимо от того, выполняется ли алгоритм виртуально или на модели платформы, а также от конкретной конфигурации платформы.

Интегрированная среда программирования VIPE позволяет:

1. Разработать параллельный алгоритм решения задачи, провести иерархическую декомпозицию до нужной гранулярности, спроектировать параллельный алгоритм, эффективный для выполнения на параллельной СнК как ВС с распределенной архитектурой.

2. На ранних этапах сделать оценку времени выполнения и вычислительной сложности алгоритма. Это позволяет до этапа детального проектирования платформы рамочно оценить требования к вычислительным ресурсам, памяти, коммуникационной среде.

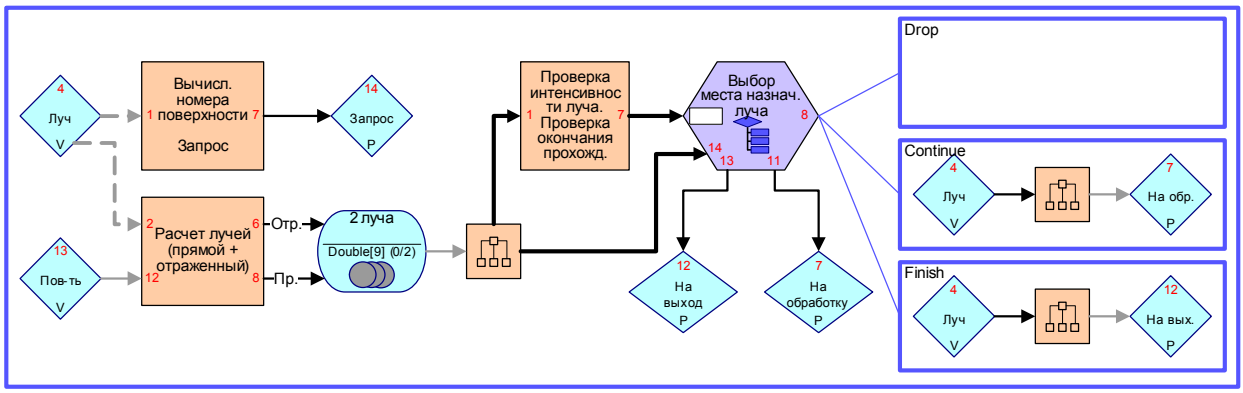

Рис. 4. Цикл расчета прохождения луча через оптическую поверхность

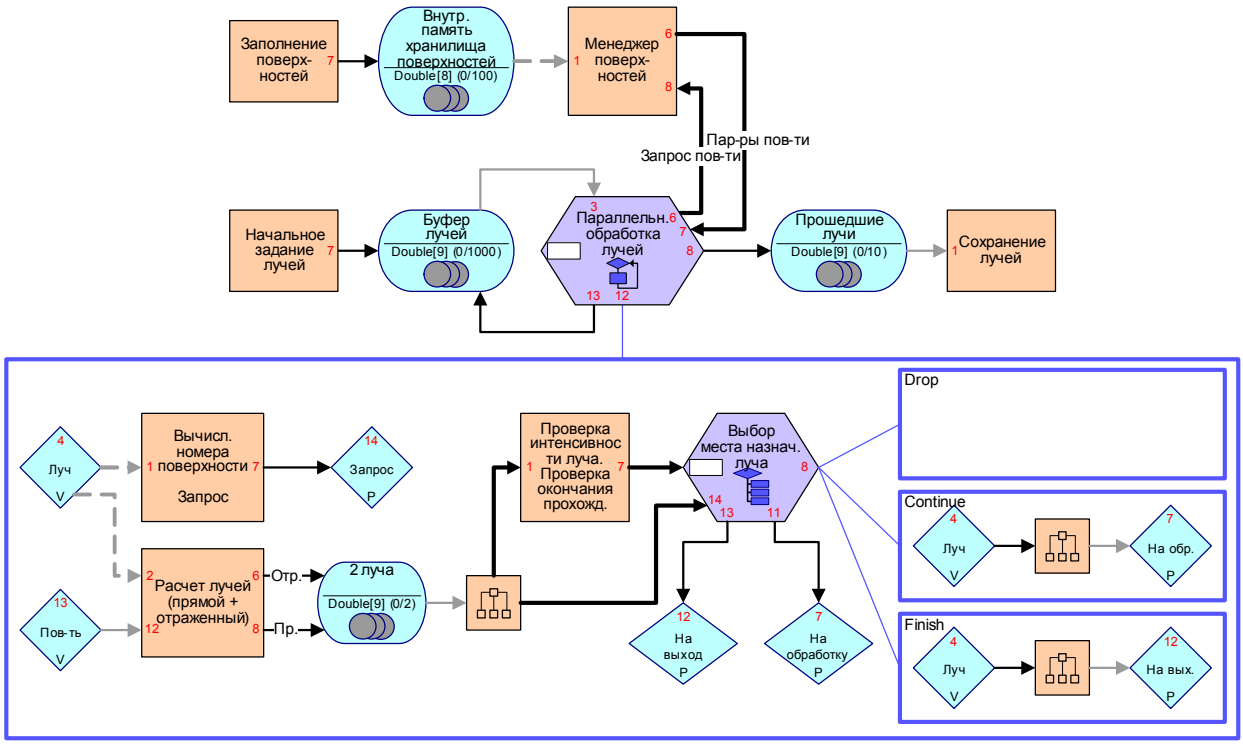

Рис. 5. Полная схема программы расчета прохождения

3. Полностью запрограммировать алгоритм, провести функциональную отладку и тестирование разработанного ПО. Выполнять ПО на различных моделях и конфигурациях вычислительной платформы, более детально получить характеристики выполняемых программ и требований к аппаратуре.

4. Используя компилятор, под конкретную платформу произвести сборку загрузочных модулей для запуска непосредственно на вычислительной платформе.

5. Возможность использовать преимущества конвейерного параллелизма, естественного параллелизма прикладных алгоритмов, естественную динамику выполняемой задачи для улучшения характеристик параллельной программы, повышения производительности (performance, throughput), сокращения времени выполнения (latency).

В рамках апробации концепции разработаны программы для ряда прикладных задач матричной алгебры, обработки сигналов, коммуникационных алгоритмов, алгоритмов вычислительной фотографии и обработки видеопотока, hash-алгоритмов.

Библиографические ссылки

1. Шейнин Ю. Е. Асинхронные развивающиеся процессы – формальная модель параллельных вычислений в распределенных вычислительных структурах // Распределенная обработка информации : тр. Междунар. конф. Новосибирск, 1998. С. 111–115.

2. Ivanov V., Sheynin Y., Syschikov A. Programming model for coarse-grained distributed heterogeneous architecture // Тр. XI симпозиума по проблеме избыточности в информац. системах / под ред. проф. Е. А. Крука. СПб. : ГУАП, 2007. С. 246–250.

3. Шейнин Ю. Е., Сыщиков А. Ю. Язык параллельного программирования крупногранулярных динамических вычислений // Параллельные вычисления и задачи управления : тр. III Междунар. конф. М. : ИПУ РАН, 2006.

4. Сыщиков А. Ю. Программа построения оптимальных расписаний выполнения средне-гранулярных параллельных программ, представленных графом процессов: рег. № ГосФАП – № 6447, 28.06.2006.

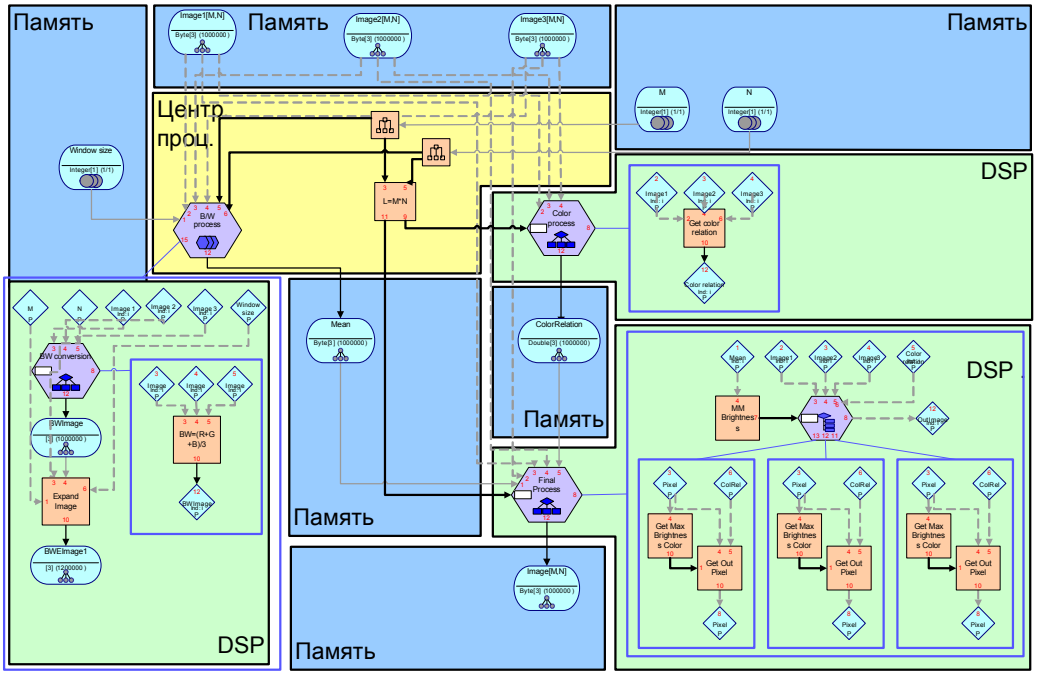

Рис. 6. Пример размещения схемы программы на неоднородной СнК

A. Yu. Syschikov

CONSEPT AND TECHNOLOGY OF PARALLEL PROGRAMMING FOR HETEROGENEOUS SYSTEM ON CHIP

Common Many-Core SoC architectures is a quick-growing segment of distributed parallel computing systems. Such systems cannot be effectively programmed "manually", however there are only basic instruments and tools for programming.

In this article we propose the concept of Many-Core SoC parallel programming. The concept presents visual programming approach, medium-grained parallel program organization with sequential grains and dynamics of parallel computation. The article suggests the set of tools named VIPE: VPL Integrated Programming Environment. VIPE includes strict mathematical formal model basis, visual parallel programming language for medium-grained programs design, toolset for mapping, translating and pre-compiling of the program scheme and the Many-Core SoC simulation software.

Keywords: system-on-Chip, parallel programming, dynamic computations, integrated programming environment.