одному блоку MAC и по одному блоку ПАП, а также дополнительную управляющую логику.

3. Приведенные в данной работе приемы и конструкции языка VHDL могут быть использованы и в других проектах, связанных с реализацией цифровых устройств на ПЛИС, в частности для реализации устройств сигнальной обработки.

Список литературы

1. *Емец С.* Verilog — инструмент разработки цифровых электронных схем // Компоненты и технологии. Вып. 2. М., 2002. www.compitech.ru/html.cgi/arhiv/01_02/stat_86.htm/.
2. *Стешенко В.*, *Шишкин Г.*, *Евстифеев А.*, *Седякин Ю.* Школа разработки аппаратуры цифровой обработки сигналов на ПЛИС. Занятие 4. Язык описания аппаратуры VHDL. http://www.chipnews.ru/html.cgi/arhiv/00_01/stat-28.htm/.
3. *Степченков Ю.А.*, *Петрухин В.С.* Особенности гибридного варианта реализации на ПЛИС рекуррентного обработчика сигналов // Наст. сб. С. 118–129.
4. *Суворова Е.А.*, *Шейнин Ю.Е.* Проектирование цифровых систем на VHDL. СПб: БХВ-Петербург, 2003.
5. *Сергиенко А.М.* VHDL для проектирования вычислительных устройств. Киев: ДиаСофт, 2003.
6. *Бибило П.Н.*, *Авдеев Н.А.* VHDL. Эффективное использование при проектировании цифровых систем. М.: Солон-Пресс, 2006.
7. Quartus II Development Software Handbook Vol. 7.2 (Complete Five-Volume Set). http://www.altera.com/literature/hb/qts/quartusii_handbook.pdf.
8. Cyclone III FPGAs: An Unprecedented Combination of Power, Functionality, and Cost. http://www.altera.com/products/devices/cyclone3/cy3-index.jsp.
9. The Cyclone III FPGA Handbook. http://www.altera.com/literature/hb/cyc3/cyclone3_handbook.pdf.

УДК 681.324.192

ВЫБОР ЯЗЫКОВЫХ СРЕДСТВ ПРЕДСТАВЛЕНИЯ ПАРАЛЛЕЛЬНЫХ АЛГОРИТМОВ ДЛЯ РЕКУРРЕНТНОГО ОБРАБОТЧИКА СИГНАЛОВ

Ю.А. Степченков, *В.С. Петрухин*, *Д.В. Хилько*

В статье анализируются современные методологии реализации параллельных алгоритмов в различных парадигмах программирования. Рассмотрено несколько конкретных языков программирования и описаны их функциональные возможности. Выработаны критерии, в соответствии с которыми определяется пригодность предложенных языков программирования для реализации параллельных алгоритмов рекуррентным обработчиком сигналов.

1. Введение

Идея параллельных вычислений возникла еще до появления фон-неймановской архитектуры компьютера. Фактически появлению фон-неймановской архитектуры предшествовали различные предложения по реализации архитектур, ориентированных как на последовательные, так и на параллельные вычисления. Но экономически эффективным оказалось лишь предложение фон Неймана.

Однако уже тогда были предложены различные решения и модели архитектур, ориентированных на параллельные вычисления [1], что не осталось незамеченным математиками, программистами и проектировщиками и дало им обильную пищу для размышлений. Пожалуй, именно к тому времени можно отнести возникновение терминов «параллельное программирование», «параллельные алгоритмы».

2. Новая вычислительная парадигма и постановка задачи исследования

В ИПИ РАН ведутся исследования новой вычислительной парадигмы. Была разработана модель рекуррентного обработ-

чика сигналов (РОС) [2], представляющая собой архитектуру гибридного типа: управляющий, или верхний, уровень — процессор общего назначения на базе фон-неймановской архитектуры, операционный, или нижний, уровень — специализированный процессор на базе рекуррентной архитектуры. Последний состоит из четырех независимых (параллельных) секций, в каждую из которых входит память адресной проверки, вычислитель (на базе 16-разрядного АЛУ и умножителя с накоплением MAC) и преобразователь тегов.

Рекуррентный процессор выполняет специальные программы — капсулы. Каждая капсула содержит набор самоопределяющихся данных, т.е. собственно данные и указания по их обработке. Чтобы достичь наибольшего эффекта в использовании параллелизма, капсула должна быть правильно сформирована.

Управляющий уровень, помимо основных функций, должен готовить капсулы. Само формирование капсулы — процесс далеко не тривиальный. Собственно процедуре формирования капсулы должно предшествовать выявление параллельных блоков в исполняемом алгоритме и формирование шаблона капсул. Для решения первой задачи предполагается использовать компилятор языка программирования высокого уровня, результатом работы которого будет векторизированный граф. Для решения второй задачи будет разработан транслятор под названием ТОПОГРАФ-К (Трансляция Ориентированного ПОтокового ГРАФа в Капсулу). Его основная цель — трансляция графовой или текстовой формы исполняемого алгоритма в капсульную форму представления в соответствии с требованиями спецификации на интерфейс управляющего и операционного уровней РОС.

Данная статья посвящена задаче выбора наиболее подходящего языка программирования, а следовательно, и компилятора.

3. Основные парадигмы программирования с точки зрения параллельной обработки данных

Развитие программирования как дисциплины в целом породило две принципиально различные парадигмы: императивное и декларативное программирование. Наиболее важными направлениями декларативного программирования в свою очередь являются

функциональное и логическое программирование. В настоящее время границы между этими направлениями настолько явны, что их относят к разным парадигмам [3].

Императивное программирование основано на представлении программы как последовательности инструкций или команд, исполняемых процессором. Программируя в императивном стиле, мы должны пошагово описать процесс решения. Сама концепция императивного программирования затрудняет реализацию параллелизма.

Один из способов достижения параллелизма — использование процессно-ориентированного подхода [4]. Однако он связан с трудностями технического характера. Перед разработчиками возникает целый ряд проблем по реализации параллельных алгоритмов в рамках императивной парадигмы программирования; в частности — определение естественного параллелизма и его эффективное использование.

Декларативное программирование основано на представлении программы как совокупности утверждений [3]. Декларативные языки хорошо подходят для программирования параллельных компьютеров, не требуя специальных методов описания параллелизма. Параллелизм программы определяется неявно её информационными связями [3].

Причиной разделения логического и функционального программирования стала «основная единица» языка: в случае функционального программирования речь идет о функции, в случае логического — это отношение или предикат (другими словами, используется логика первого порядка).

Иначе говоря, логическое программирование основано на логике [5], а в основе функционального программирования лежит понятие функции, ключевое не только в этом стиле, но и в математике, λ-исчислениях [6]. Одно из наиболее интересных и важных свойств функции — композиция нескольких совершенно независимых друг от друга простых функций в одну, более сложную (и, соответственно, декомпозиция) [7]. Именно это свойство является одним из определяющих при выборе парадигмы функционального программирования, поскольку непосредственно отражает естественный параллелизм алгоритма на уровне отдельных функций, которые, в некотором смысле, становятся аналогами процессов в императивном стиле.

4. Анализ функциональных возможностей существующих языков программирования с точки зрения параллельной обработки

Один из основных критериев выбора языка программирования высокого уровня — возможность реализации параллельных алгоритмов. В исследованиях ИПИ РАН [8] был проведен анализ языков программирования для компьютеров с потоковой архитектурой и выдвинуты требования к этим языкам с точки зрения рекуррентной парадигмы вычислений. Даны также подробные описания некоторых языков программирования и установлено, что класс декларативных, а точнее — функциональных, языков программирования наиболее соответствует постановке задачи.

С учетом этих предпосылок и некоторого опыта создания параллельных программ были выбраны три языка программирования — Haskell, SISAL и РЕФАЛ. Далее приводится описание их возможностей с точки зрения реализации параллельных программ.

4.1. Язык Haskell. Язык функционального программирования Haskell (назван по имени математика Хаскелла Карри, Haskell Curry) появился и развивался параллельно с развитием теории λ-исчислений. Это наиболее распространенный типовой язык функционального программирования. Как и большинство функциональных языков программирования, он обладает типовыми свойствами: краткостью и простотой, строгой типизацией, модульностью, чистотой (отсутствием побочных эффектов и детерминированностью), возможностью отложенных (ленивых) вычислений; функции — это значения и объекты вычисления. Для языка было создано большое количество компиляторов. Haskell позволяет описывать решение задач функционального программирования, имея в своем составе некоторый функционал императивных языков (ввод, вывод и др.). Подробное описание языка можно найти в [6].

С точки зрения параллельного программирования и параллельной обработки данных возможности Haskell несколько ограничены. Обеспечивая возможность параллелизма на уровне отдельных функций, Haskell не предоставляет широкого спектра инструментов для создания, отладки, исполнения параллельных программ. Можно отметить один из компиляторов, созданных для этого языка, — GHC (Glasgow Haskell Compiler). Этот

компилятор позволяет не только запускать программу на одном исполнителе, но и задавать параметр числа исполнителей. Однако существенным недостатком является тот факт, что получить доступ к внутреннему представлению (внутреннему графу программы) практически невозможно или очень сложно. Это сильно затрудняет выявление параллельных блоков алгоритма, препятствует автоматизации данного процесса. А автоматизация выявления параллельных блоков и получение внутреннего графа — первостепенная задача исследования по тематике РОС.

4.2. Язык РЕФАЛ. Язык функционального программирования РЕФАЛ (Recursive Functions Algorithmic Language, REFAL) появился позже языка Haskell и предназначен для выполнения одной из типовых задач, решаемых методами функционального программирования, — эквивалентной трансформации программ.

Задача эквивалентной трансформации программ возникла после переосмысления достижений теории построения трансляторов и получила продолжение в рамках оптимизации программ на уровне исходного кода. Этот процесс называется суперкомпиляцией. По своей сути трансформация программы — это перевод ее из одного языка в другой. Компилятор SCP4, используемый для программ, написанных на некотором языке, справляется с этой задачей вполне успешно.

Подробное описание языка можно найти на одноименном сайте, а также в [9]. Язык достаточно распространен, для него создан мощный компилятор. Немаловажен тот факт, что РЕФАЛ разрабатывается российской группой специалистов, следовательно, открыты и доступны его текущие и новые версии. В работе [9] подробно описывается РЕФАЛ-граф некоторой программы — первая стадия ее компиляции. Этот граф ориентирован на адекватное описание временной эффективности и доступен для анализа, однако его суть — скорее описывать все возможные пути в программе. Поскольку граф представляет собой дерево прогонки, можно с его помощью выделять параллельные блоки.

Один из основных недостатков языка — его синтаксис. Программы на языке РЕФАЛ представляют собой выражения, записанные в БНФ-нотации [10], что затрудняет программирование и чтение текста программ.

4.3. Язык SISAL. Из языков параллельного программирования достаточно известен язык функционального программирования SISAL. Название языка расшифровывается как Streams and Iterations in a Single Assignment Language, он представляет собой дальнейшее развития языка VAL, получившего распространение в середине 70-х годов XX в. Среди целей разработки языка SISAL следует отметить наиболее характерные, связанные с функциональным стилем программирования:

- создание универсального функционального языка;
- разработка техники оптимизации для высокоэффективных параллельных программ;
- достижение эффективности исполнения, сравнимой с императивными языками типа Fortran и C;
- внедрение функционального стиля программирования для больших научных программ.

Функциональная программа — параллельная, если ее можно писать, освободившись от большинства сложностей параллельного программирования, связанных с выражением частичных отношений порядка между отдельными операциями уровня аппаратуры. Пользователь языка SISAL получает возможность сконцентрироваться на конструировании алгоритмов и разработке программ в терминах крупноблочных и регулярно организованных построений, опираясь на естественный параллелизм уровня постановки задачи.

Несомненное преимущество языка SISAL в том, что он разрабатывался как язык параллельного программирования и непосредственно использует скрытый параллелизм. Кроме того, компилятор языка OSC позволяет не только компилировать и исполнять программы с различной степенью параллелизма, но и строить текстовые графы промежуточных представлений, отражающие как чистую, математическую параллельность, так и указанную вручную.

Язык SISAL довольно широко распространен. Более того, его исследованием и доработкой занимаются в Новосибирском отделении РАН. В сборниках, подготовленных в Сибири, можно найти статьи, посвященные этому языку. Синтаксис языка интуитивно понятен и схож с синтаксисом языка Pascal; SISAL наследует также основные свойства языков декларативного стиля.

В Приложении приведены две реализации основной операции алгоритма быстрого преобразования Фурье, одна из которых построена компилятором языка SISAL, а другая — в РОС.

4.4. Сравнительная оценка выбранных языков. Для определения наиболее подходящего языка программирования были разработаны критерии выбора и составлена сводная таблица (с использованием работы [8]).

Таблица 1

Язык	Распространенность	Наличие компиляторов	Доступность и поддержка	Простота программирования	Краткость описания	Удобство выявления параллелизма	Построение внутреннего графа программы
Haskell	+	+	+	+	+	–	–
РЕФАЛ	+	+	+	–	–	+	+
SISAL	+	+	+	+	+	+	+

5. Заключение

Резюмируя все приведенные доводы и аргументы, можно сделать вывод, что наиболее подходящими языками являются SISAL и РЕФАЛ. Однако, опираясь на данные табл. 1, предпочтителен SISAL. Граф, предоставляемый компилятором языка РЕФАЛ, очень специфичен, поскольку содержит, по сути, синтаксический разбор исходного текста программы, а сам компилятор трансформирует язык, на котором написана исходная программа, в другой язык. Язык капсульного программирования, как внутренний язык РОС, сам по себе не тривиален, не обладает гибкостью, присущей языкам высокого уровня, и на данный момент не может быть четко формализован.

Между тем компилятор языка SISAL предоставляет текстовые графы алгоритма, которые могут быть визуализированы и использованы в дальнейшем на этапе работы программного средства ТОПОГРАФ-К.

Однако не следует снимать с рассмотрения язык РЕФАЛ, поскольку, решив трудности технического характера, можно получить отечественный, постоянно совершенствуемый инструмент.

Список литературы

1. *Гергель В.П.*, *Стронгин Р.Г.* Основы параллельных вычислений для многопроцессорных вычислительных систем. Н. Новгород: Изд-во нижегородского университета. `http://www.software.unn.ru/ccam/files/HTML_Version/part1.html#a1/`.
2. *Степченков Ю.А.*, *Петрухин В.С.* Особенности гибридного варианта реализации на ПЛИС рекуррентного обработчика сигналов // Наст. сб. С. 118–129.
3. *Дехтяренко И.А.* Декларативное программирование. `http://www.softcraft.ru/paradigm/dp/dp01.shtml/`.
4. *Джефри Р.* Windows для профессионалов. Создание эффективных Win-32 приложений с учетом специфики 64-разрядной версии Windows. СПб: Питер, 2004. С. 48–97.
5. *Дубинин В.Н.*, *Зинкин С.А.* Языки логического программирования в проектировании вычислительных систем и сетей. Пенза: Изд-во Пензю госю техн. ун-та, 1997. С. 3–5.
6. *Душкин Р.В.* Функциональное программирование на языке Haskel. М.: ДМК, 2007. С. 273–330.
7. *Филд А.*, *Харисон П.* Функциональное программирование. М.: Мир, 1993. С. 11–19.
8. Исследование новой вычислительной парадигмы и разработка на её основе логического проекта динамического многопоточного процессора обработки сигналов (промежуточный за этап 1). Шифр «Сигнал», № г.,Р. 01.2.00 104927. М.: ИПИ РАН, 2001. С. 2.34–2.46.
9. *Немытых А.П.* Суперкомпилятор SCP4. М.: Изд-во ЛКИ, 2007. — 152 с.
10. *Кузин Л.Т.* Основы кибернетики. Т. 1. М.: Энергоатомиздат, 1994. — 576 с.
11. Исследование программируемости архитектурно-алгоритмических и схемотехнических проблем проектирования рекуррентных компьютеров (заключительный отчет). Шифр «ПАРСЕК». № г.,Р. 01.20.0412412. М.: ИПИ РАН, 2006. С. 47–55.

Приложение

В работе [11] рассмотрены некоторые особенности языка SISAL, а также граф «Бабочки» (базовой операции быстрого преобразования Фурье — БПФ), проиллюстрированный на рис. 1. В работе [2] приведена РОС-реализация этого же алгоритма (рис. 2). Сравнивая графы, нетрудно заметить, сколь невелико различие между ними. Однако в настоящее время алгоритм «Бабочка» реализован в РОС одной командой BUTT (BUTTerfly). Введение этой команды существенно упрощает

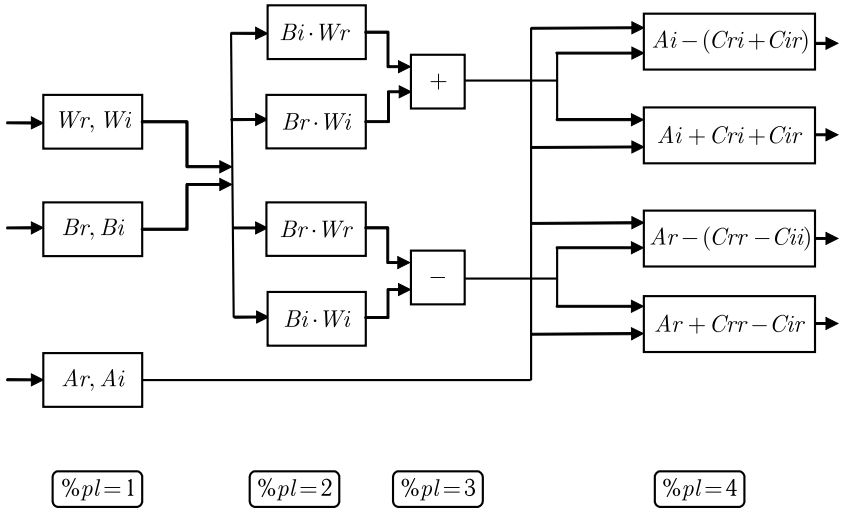

Рис. 1. Граф алгоритма «Бабочки»–БПФ, SISAL-реализация

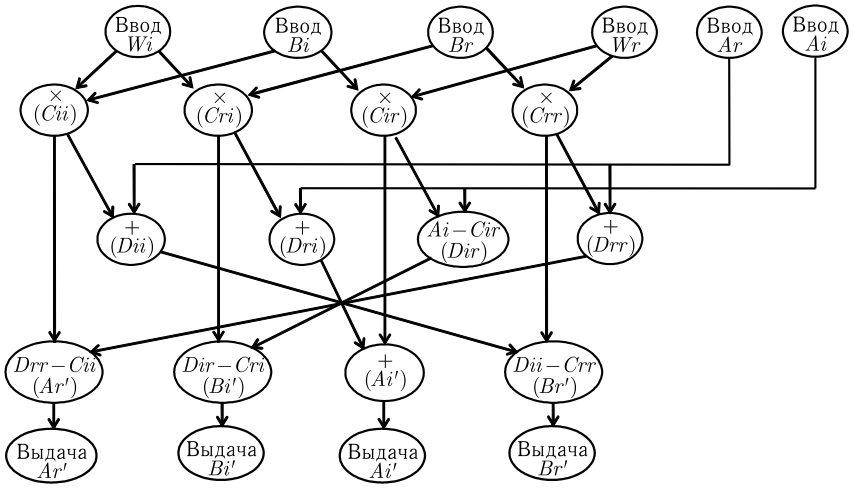

Рис. 2. Граф алгоритма «Бабочки»–БПФ, РОС-реализация

структуру, уменьшает размер капсулы и, кроме того, в четыре раза уменьшает время реализации БПФ.

Комплексное БПФ «Бабочка» (*по основанию 2 с прореживанием по времени*). Исходное уравнение:

$$A' = A + BW; \quad B' = A - BW.$$

Все значения — комплексные.

Алгоритм преобразования уравнения:

1)	2)	3)
$Crr = Br \cdot Wr$	$Drr = Ar + Crr$	$Ar' = Drr - Cii$
$Cri = Br \cdot Wi$	$Dri = Ai + Cri$	$Ai' = Dri + Cir$
$Cii = Bi \cdot Wi$	$Dii = Ar - Cii$	$Br' = Dii - Crr$
$Cir = Bi \cdot Wr$	$Dir = Ai + Cir$	$Bi' = Dir - Cri$

Здесь Ar, Br, Wr — действительные части входных комплексных данных; Ai, Bi, Wi — мнимые части; Ar', Ai', Br', Bi' — действительные и мнимые части преобразованных комплексных отсчетов A, B.

Входные данные: Wr, Wi, Ar, Ai, Br, Bi.

В приведенном примере все индексированные числа — действительные. Метка %*pl* позволяет определить, в каком по счету «слое» находится текущий узел. Все узлы, находящиеся на одном «слое», могут быть выполнены параллельно. Полное описание реализации данного алгоритма может быть найдено в работе [11].

Часть IV

ИНФОРМАТИЗАЦИЯ ОБЩЕСТВА. ПРИМЕНЕНИЕ ИНФОРМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ В НАУЧНЫХ ИССЛЕДОВАНИЯХ

УДК 004.82

ИНФОРМАЦИОННЫЕ РЕСУРСЫ И ИНДИКАТОРЫ ДЛЯ ОЦЕНКИ ИННОВАЦИОННОГО ПОТЕНЦИАЛА НАПРАВЛЕНИЙ НАУЧНЫХ ИССЛЕДОВАНИЙ [1]

И.М. Зацман, О.А. Курчавова, И.В. Галина

Рассматривается методика вычисления количественных индикаторов взаимосвязей направлений научных исследований и приоритетных областей технологического развития для прогнозирования изменения этих взаимосвязей и выбора приоритетных направлений научных исследований, ориентированных на решение прикладных проблем. Анализируется зарубежный опыт вычисления значений количественных индикаторов взаимосвязей с использованием полнотекстовых патентных информационных ресурсов. Предлагается подход к адаптации зарубежной методики к вычислению индикаторов взаимосвязей на основе отечественных информационных ресурсов.

1. Введение

В статье рассматриваются методологические вопросы оценки инновационного потенциала направлений научных исследований, основанной на вычислении количественных индикаторов взаимосвязей этих направлений с приоритетными областями технологического развития. Подобные вычисления и оценки позволяют прогнозировать изменения этих взаимосвязей. В рамках национальной инновационной системы эти вопросы отнесены к зада-

[1] Работа выполнена при частичной финансовой поддержке РГНФ (грант № 06-02-04043а).