Применение отечественных специализированных процессоров семейства КОМДИВ в научных расчётах

П.Б. Богданов, О.Ю. Сударева

Аннотация. В статье на примере трёх алгоритмов из NAS Parallel Benchmarks исследуется принципиальная возможность использования аппаратно-программных решений ФГУ ФНЦ НИИСИ РАН в расчётных задачах общего назначения. Для различных реализаций выбранных алгоритмов — как референсных, так и оптимизированных под микропроцессоры семейства КОМДИВ — приводятся результаты замеров производительности на уже выпускаемых микропроцессорах, а также на ПЛИС-прототипе перспективного микропроцессора. Приводятся также оценки ожидаемой производительности на многопроцессорных комплексах. Делаются выводы о принципиальной применимости таких комплексов для научных расчётов и даются рекомендации к дальнейшей оптимизации аппаратных компонент.

Ключевые слова: КОМДИВ, CP2, CPV, RapidIO, NAS PB, MG, CG, FT

Введение

В течение последних десятилетий в ФГУ ФНЦ НИИСИ РАН (далее НИИСИ РАН) ведётся разработка отечественной элементной базы для вычислительных комплексов специального назначения. Это включает в себя семейство микропроцессоров КОМДИВ на базе RISC-архитектуры MIPS [1], контроллеры и коммутаторы высокопроизводительной коммуникационной среды на базе протокола RapidIO [2], контроллеры высокоскоростных шин VME [3] и PCI\PCI Express [4, 5], а также широкий спектр периферийных контроллеров для SoC-компоновки. На сегодняшний день решения от НИИСИ РАН в сфере специализированных вычислений типа цифровой обработки сигналов смело конкурируют (а в некоторых задачах существенно превосходят) с решениями мировых производителей, например, таких как семейство микропроцессоров TigerSHARC от Analog Devices, Inc. [6].

Помимо специальных приложений и аппаратно-программных решений, НИИСИ РАН ведутся исследования по применению разрабатываемого оборудования в вычислениях общего назначения.

В работе рассматривается вычислительный комплекс на базе разрабатываемой НИИСИ РАН линейки микропроцессоров КОМДИВ (в частности, КОМДИВ128-РИО, со встроенным математическим сопроцессором CP2 — см. документацию [7]), а также контроллеров и коммутаторов высокопроизводительной коммуникационной сети RapidIO. Такой комплекс может быть использован для распределённых вычислений: на каждом процессоре вычисления оптимизируются тем или иным способом, а обмены между процессорами осуществляются через программный интерфейс MPI [8]. В рамках этой схемы возможна эффективная реализация многих алгоритмов, используемых при решении различных практических задач.

В качестве тестовых задач в работе были выбраны три алгоритма из пакета NAS Parallel Benchmarks [9], а именно: алгоритм многосеточного метода (MG), алгоритм метода сопряжённых градиентов (CG), а также быстрое преобразование Фурье (FT). Данные алгоритмы широко распространены и используются при решении вычислительных задач из различных областей естествознания. Применимость той или иной вычислительной системы для этих алгоритмов может служить показателем её применимости для решения широкого класса реальных задач.

Для выбранных алгоритмов имеются референсные реализации: как последовательные, для одного процессора, так и параллельные, в частности, с использованием MPI. На основе последовательных версий кодов был разработан ряд реализаций для процессоров линейки КОМДИВ. Кроме того, в настоящее время в НИИСИ ведутся работы по оптимизации библиотеки MPI на коммуникации по RapidIO [11]. Поддержка MPI позволяет получить на основе оптимизированных однопроцессорных кодов реализации для многопроцессорного комплекса в целом.

Все однопроцессорные реализации были протестированы на доступной аппаратуре. На основе имеющихся данных о производительности коммуникационной среды RapidIO и результатов первоначального тестирования MPI-RIO [11] были также сделаны оценки производительности для многопроцессорных реализаций. По итогам проведённой работы делаются выводы о специфике конкретных алгоритмов и узких местах аппаратных решений, которые нуждаются в доработке. Помимо этого, приводится теоретическая оценка ожидаемой эффективности вычислений на рассматриваемых многопроцессорных комплексах.

1. NAS Parallel Benchmarks

Рассматриваемые в работе алгоритмы MG, CG и FT входят в состав пакета NAS Parallel Benchmarks (Numerical Aerodynamics Simulation Parallel Benchmarks, или NPB). NPB — это набор тестов, разработанных в центре NASA для оценки производительности параллельных суперкомпьютеров. Тесты представляют собой некоторые упрощённые базовые алгоритмы, на основе задач вычислительной гидродинамики, позволяющие оценить характеристики вычислительной системы, существенные при решении реальных задач. Первая версия тестов вышла в 1994 году; пакет тестов продолжает пополняться и уточняться, на текущий момент последняя версия — 2012 года.

Классический состав NPB — 5 «вычислительных ядер»: MG, CG, FT, а также IS (целочисленная сортировка) и EP («чрезвычайная параллельность»), и три «псевдоприложения» — модельных задачи вычислительной гидродинамики. Алгоритм IS основан на ветвлениях и условных переходах, поэтому его реализация на процессорах SIMD-архитектуры нецелесообразна. Алгоритм EP тестирует пиковую производительность вычислительной системы. Эти два алгоритма, а также псевдоприложения, ввиду их специфичности, в данной работе не рассматриваются.

Все алгоритмы из NPB имеют так называемую «бумажную» спецификацию: они полностью описаны в тексте [9], и программисту предлагается реализовать их для конкретной системы любым удобным способом. Накладывается ряд ограничений на используемые средства: например, не разрешается использовать язык ассемблера и подключать нестандартные библиотеки. Цель тестов состоит в том, чтобы проверить, какой производительности вычислений сможет достичь на той или иной системе рядовой программист при решении своих задач.

В тексте также задаются входные данные и эталонные результаты, для проверки корректности реализации алгоритмов. Все вычисления над числами с плавающей точкой требуется производить с двойной точностью.

В NPB вводится понятие классов задач — от S до F. Классы соответствуют различным объёмам входных данных, количествам итераций внутренних циклов в алгоритмах и т.п. Для каждой задачи с увеличением класса объём вычислительной работы возрастает экспоненциально. Соответственно, разные классы рассчитаны на вычислительные системы разных масштабов:

S — маленькие задачи, для тестирования и отладки;

W — задачи для рабочих станций 90х годов;

A, B, C — типичные размеры задач для современных персональных компьютеров;

D, E, F — задачи для распределённых высокопроизводительных вычислительных комплексов.

Имеются готовые референсные реализации алгоритмов, доступные на сайте разработчика [10], последняя редакция — NPB 3.3.1. Представлены различные версии кодов: в основном на языке FORTRAN, частично на C; последовательная версия и параллельные с использованием моделей программирования MPI и OpenMP; версии на Java и HPF, и др. Коды сделаны как образец для облегчения работы программисту; никакая совместимость с ними не требуется.

В некоторых реализациях алгоритмов для процессоров линейки КОМДИВ, представленных в данной работе, есть несколько условных отступлений от требований, заявленных NPB. Во-первых, при реализации ряда из них использовалась вспомогательная библиотека обработки сигналов (`dsplib`) для КОМДИВ128-РИО и вычислительные ядра для арифметического сопроцессора СР2 на языке ассемблера. Во-вторых, в настоящее время сопроцессор СР2 не поддерживает вычислений с двойной точностью. Поэтому в работе, наряду с референсными реализациями алгоритмов, рассматриваются реализации для вещественных чисел одинарной точности. Соответственно, погрешность результата вычислений в этом случае оказывается существенно больше, чем требуется в тестах NPB. Отметим, однако, что описываемые реализации в будущем можно путём незначительных правок адаптировать к вычислениям с двойной точностью — общая схема работы и принципиальные оценки эффективности останутся без изменений.

1.1. Алгоритм MG

Многосеточный метод в настоящее время широко применяется для решения многих задач, в частности, для предобуславливания СЛАУ с разреженными матрицами. В NPB рассматривается простая версия алгоритма: классический геометрический многосеточный метод. Задача тестирует регулярные обмены данными между процессорами.

Формулировка задачи: найти приближённое решение уравнения Пуассона

$$\nabla^2 u = v \qquad (1)$$

на сетке $N \times N \times N$, с периодическими граничными условиями. Алгоритм, который требуется использовать для решения, полностью описан в документации NPB. На каждой итерации осуществляется коррекция приближённого решения на наборе сеток от самой мелкой ($N \times N \times N$) до самой грубой ($2 \times 2 \times 2$): производится последовательность проекций на всё более и более грубые сетки; на самой грубой сетке вычисляется поправка к приближению, после чего эта поправка последовательно интерполируется на более мелкие сетки (вычисляется интерполяция, невязка и сглаживание). Начальное приближение — нулевое. Также для задачи каждого класса задаётся следующее:

- размер сетки N;
- функция в правой части, v — равна 0 везде, кроме нескольких точек, в которых принимает значение 1 или -1;
- количество итераций алгоритма IT;
- эталонная норма невязки для искомого решения;
- допустимая погрешность.

Алгоритм в сущности представляет собой последовательность трилинейных операторов на сетке (далее для краткости любой такой трилинейный оператор будем обозначать «Tlin»). Tlin переводит трёхмерный массив в трёхмерный массив, каждое новое значение в точке получается в общем случае как сумма 27 соседних старых значений с весовыми коэффициентами (Рис. 1): старое значение в той же точке берётся с весом c_0, в 6 точках, отстоящих от неё на 1 по одной координатной оси — с весом c_1, по двум осям — с весом c_2 (12 точек), по всем трём осям — с весом c_3 (8 точек).

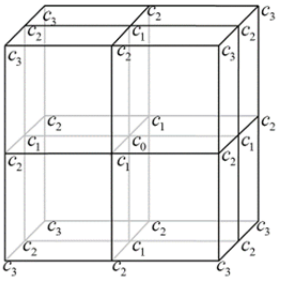

Рис. 1. Tlin: схема вычисления нового значения в точке

Всего используется 4 различных оператора, их коэффициенты заданы и не зависят от уровня:

- A (невязка) — оставляет сетку $M\times M\times M$ неизменной. Его коэффициенты:

$$c_0 = -\frac{8}{3},\ c_1 = 0,\ c_2 = \frac{1}{6},\ c_3 = \frac{1}{12};$$

- P (проекция) — уменьшает сетку: $M\times M\times M \mapsto M/2\times M/2\times M/2$. Коэффициенты:

$$c_0 = \frac{1}{2},\ c_1 = \frac{1}{4},\ c_2 = \frac{1}{8},\ c_3 = \frac{1}{16};$$

- Q (интерполяция) — увеличивает сетку: $M\times M\times M \mapsto 2M\times 2M\times 2M$. Коэффициенты:

$$c_0 = 1,\ c_1 = \frac{1}{2},\ c_2 = \frac{1}{4},\ c_3 = \frac{1}{8};$$

- S (сглаживание) — оставляет сетку $M\times M\times M$ неизменной. Коэффициенты:

$$c_0 = -\frac{3}{8},\ c_1 = \frac{1}{32},\ c_2 = -\frac{1}{16},\ c_3 = 0.$$

При вычислении значений на границе требуется учитывать периодические граничные условия: по каждому из трёх измерений $u_{N+1} = u_1,\ u_0 = u_N$ — соответственно, на входе необходимы значения с противоположной грани обрабатываемого трёхмерного массива. Иллюстрация для различных операторов (для простоты, в двумерном случае) приводится на Рис. 2. В левой части для каждого оператора пунктиром показаны копии противоположных границ. В правой части тёмно-серым цветом выделены вычисляемые значения. Для операторов P и Q светлым для удобства показаны соответствующие позиции в более мелкой из двух сеток (выходной для проекции и входной для интерполяции). Примеры групп соседних точек, которые суммируются с весами, и соответствующие выходные точки выделены чёрным. Как видно из рисунка, для оператора P шаблон суммирования такой же, как для A и S,

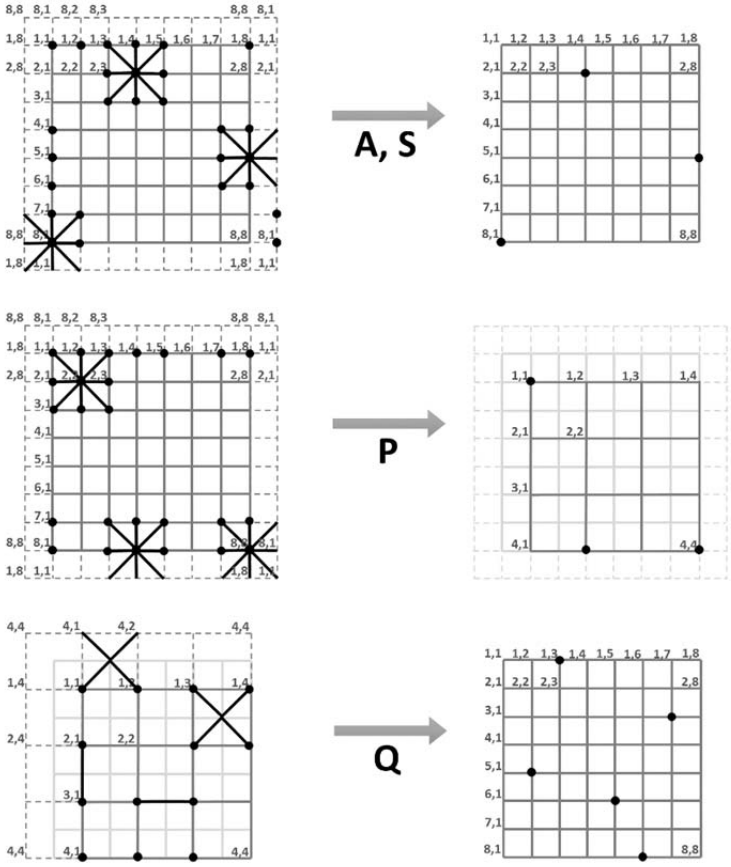

Рис. 2. Tlin: учёт граничных условий

но вычисляется вдвое меньше точек по каждой из осей (т.е. в 8 раз меньше в совокупности для трёхмерного оператора). Оператор Q отличается тем, что имеется несколько различных шаблонов суммирования, в зависимости от положения вычисляемого узла мелкой сетки относительно узлов крупной сетки.

В ходе вычислений каждый следующий оператор применяется к результату предыдущего. Некоторых шаги алгоритма включают также накопление: результат поэлементно прибавляется или вычитается из одного из полученных ранее массивов, поэтому в ходе работы требуется хранить целый набор промежуточных массивов, а все вычисления выполнять out-of-place.

При работе на многопроцессорном комплексе массив разбивается на блоки, и каждый процессор обрабатывает один блок. При вычислении Tlin на входе требуются значения не только в точках самого блока, но и, для получения значений на границе, копии граничных плоскостей соседних блоков либо противоположной границы того же блока, в зависимости от способа разбиения массива на блоки. Поэтому после вычисления каждого Tlin для каждого процессора требуется произвести обмен граничными плоскостями блока с шестью соседними процессорами. В случае, когда процессор один, процедура обмена заменяется на процедуру копирования внутри памяти процессора. Загрузку блоков целиком в память процессоров достаточно произвести один раз, перед началом вычислений. Насколько существенной будет доля времени обменов в общем времени работы алгоритма, зависит от сбалансированности производительности одного вычислительного узла и коммуникационной системы.

1.2. Алгоритм CG

В алгоритме метода сопряжённых градиентов наиболее затратной с вычислительной точки зрения является операция SpMV (умножение разреженной матрицы на вектор) — ключевая операция во многих алгоритмах из различных областей естествознания. Задача тестирует нерегулярный доступ к памяти.

Формулировка задачи из NPB: дана симметричная положительно определённая разреженная матрица A порядка $n \times n$; найти её наибольшее собственное значение. В NPB задаётся процедура генерации матрицы на языке FORTRAN, а также, для задачи каждого класса:

- параметры этой процедуры;
- порядок матрицы, n;
- начальное приближение собственного значения;
- количество итераций алгоритма;
- эталонное собственное значение для генерируемой матрицы и допустимая погрешность.

Алгоритм решения задачи, описанный в тексте NPB — метод обратных итераций со сдвигом, который сводится к решению СЛАУ с разреженной матрицей методом сопряжённых градиентов. Наиболее вычислительно-ёмкая операция SpMV выполняется на каждой итерации внутреннего цикла метода сопряжённых градиентов, а также при вычислении нормы невязки в конце каждой итерации метода обратных итераций. При этом важно, что для получения входного вектора каждого следующего SpMV требуется результат предыдущего. Все прочие вычисления представляют собой поэлементные операции над векторами, скалярные произведения векторов и операции над скалярами. Таким образом, базовой операцией алгоритма CG является SpMV; замеры, проведённые для референсных кодов, показали, что SpMV занимает порядка 95% от общего времени вычислений по алгоритму.

В процедуре генерации из NPB матрица формируется в формате CSR, однако допускается перед началом вычислений переупаковать её в любой другой формат — время на переупаковку при замерах времени вычислений не учитывается. Выбор формата хранения разреженной матрицы, оптимального для конкретной вычислительной системы, представляет собой самостоятельную задачу.

Во всех операциях SpMV используется одна и та же матрица. Её можно один раз загрузить в память процессоров перед началом вычислений, поэтому при вычислении на многопроцессорном комплексе требуется производить обмены только фрагментами векторов длины n или отдельными числами. Как правило, время на такие обмены несущественно по сравнению со временем выполнения процедуры SpMV на

процессорах. Производительность алгоритма зависит, главным образом, от производительности подсистемы памяти на одном процессоре, т.к. операция SpMV в общем случае требует доступа к памяти в случайном порядке.

1.3. Алгоритм FT

Быстрое преобразование Фурье (БПФ) является основой всех спектральных алгоритмов и библиотек цифровой обработки сигналов. Задача тестирует эффективность вычисления БПФ небольших векторов на узле, а также межузловые обмены большими объёмами данных.

В задаче, сформулированной в NPB, требуется решить уравнение в частных производных

$$\frac{\partial u(x,t)}{\partial t} = \alpha \nabla^2 u(x,t), \quad x \in \mathbb{R}^3 \qquad (2)$$

с помощью дискретного преобразования Фурье.

В документации для задачи каждого класса задаются:

- размеры обрабатываемых комплексных массивов, представляющих собой приближения функций на $\mathbb{R}^3$ — $N_1 \times N_2 \times N_3$;
- значение $\alpha = 10^{-6}$;
- количество итераций алгоритма;
- генератор псевдослучайных чисел для формирования входного массива;
- эталонные контрольные суммы для проверки результатов и допустимая погрешность.

Для решения этой задачи вычисляется трёхмерное комплексное быстрое преобразование Фурье (БПФ) от входного массива. Далее на каждой итерации, соответствующей значению t, результат БПФ поэлементно домножается на набор экспонент, после чего вычисляется обратное БПФ и контрольная сумма полученного массива. Здесь наиболее затратной операцией является трёхмерное БПФ (прямое и обратное вычисляются полностью аналогично), причём все три размерности массива являются натуральными степенями двойки.

В одномерном случае ДПФ комплексного вектора x длины N — это комплексный вектор y длины N, в котором

$$y_k = \sum_{j=0}^{N-1} x_j e^{-\frac{2\pi k j}{N} i}, \quad k = \overline{0, N-1}. \qquad (3)$$

В случае, когда $N = 2^n$, БПФ вычисляется по известному алгоритму Кули-Тьюки.

В двумерном случае дана матрица порядка $N_1 \times N_2$, и алгоритм БПФ состоит из двух шагов: на первом шаге вычисляется одномерное БПФ длины N_1 от каждого столбца матрицы, а на втором вычисляется одномерное БПФ длины N_2 от каждой строки полученной промежуточной матрицы. В трёхмерном случае дан массив порядка $N_1 \times N_2 \times N_3$ и требуется вычислить, соответственно, $N_2 \times N_3$ БПФ длины N_1, $N_1 \times N_3$ БПФ длины N_2 и $N_1 \times N_2$ БПФ длины N_3.

При вычислении на многопроцессорном комплексе на каждом шаге алгоритма каждый процессор вычисляет одномерное БПФ для некоторого подмножества векторов (длины N_1, N_2 и N_3 соответственно). После этого на первом и втором шаге требуется весь объём данных с каждого процессора передать другим процессорам: на следующем шаге каждый входной вектор содержит элементы из всех выходных векторов предыдущего шага.

2. Аппаратно-программные средства НИИСИ РАН

2.1. Аппаратура

В контрольном тестировании участвовало несколько поколений микропроцессорных архитектур разработки НИИСИ РАН.

2.1.1. Микросхема 1890ВМ6Я (КОМДИВ64-РИО, ВМ6)

Микросхема 1890ВМ6Я представляет собой 64-разрядный суперскалярный микропроцессор со встроенными системным и периферийными контроллерами, кэш-памятью второго уровня и дополнительными функциями для цифровой обработки сигналов. Изготавливается по проектным нормам КМОП 0,18 мкм.

Перечень основных компонентов микросхемы:

- контроллер динамической памяти DDR SDRAM 166/333МГц, до 2 Гбайт;
- контроллер интерфейсов RapidIO;
- контроллер PCI 33/66 МГц.

2.1.2. Микросхема 1890ВМ7Я (КОМДИВ128-РИО, ВМ7)

Микросхема 1890ВМ7Я [7] представляет собой микропроцессор, предназначенный для применения в многопроцессорных вычислительных комплексах, ориентированных на

большой объём вычислений с 32-разрядными вещественными числами, в том числе в системах цифровой обработки сигналов. Изготавливается по проектным нормам КМОП 0,18 мкм.

Микропроцессор содержит 128-разрядный арифметический сопроцессор (СР2), имеющий SIMD-архитектуру. Архитектура ядра управляющего процессора — упрощённая версия архитектуры ВМ6.

Сопроцессор СР2 включает 4 вычислительных секции, каждая из которых обладает собственной локальной памятью и набором регистров вещественной арифметики с плавающей точкой. Обмен данными между памятью DDR2 и локальной памятью сопроцессора осуществляется через канал прямого доступа к памяти — DMA — с пиковой скоростью 3,2 ГБ/с (на частоте 200 МГц). Архитектура СР2 позволяет выполнение на сопроцессоре до 40 операций с 32-разрядными вещественными числами за один такт рабочей частоты. В набор команд СР2 входит специализированная команда «бабочка Фурье», пиковая производительность на которой составляет 8 ГОП/с (также на частоте 200 МГц). Объединение процессоров в многопроцессорный комплекс осуществляется, как и для ВМ6, через каналы RapidIO.

Микросхема содержит:

- 128-разрядный специализированный сопроцессор СР2, содержащий:
  - 4 вычислительные секции АЛУ;
  - статическое ОЗУ данных объёмом 64 Кбайт в каждой вычислительной секции;
  - регистровый файл объемом 64 64-разрядных регистра (FPR) в каждой вычислительной секции;
- управляющий процессор с архитектурой КОМДИВ64;
- контроллер памяти типа DDR2;
- контроллер шины RapidIO;

Дополнительные функциональные блоки микросхемы, необходимые для целостного функционирования системы и отладочных целей:

- DMA-контроллер;
- накристальная статическая память SRAM объёмом 32 Кбайт, с разрядностью данных 32 и возможностью 8-разрядного доступа.

2.1.3. Микросхема 1890ВМ9Я (КОМДИВ128-М, ВМ9)

Перспективная микросхема 1890ВМ9Я представляет собой 256-разрядный высокопроизводительный микропроцессор, совместимый с микропроцессором КОМДИВ128-РИО. Микросхема изготавливается по технологии TSMC 65 нм. В настоящий момент ВМ9 находится на стадии разработки. Имеются ПЛИС-прототипы микросхемы. Проектная частота реального процессора составит 1000 МГц.

Микросхема КОМДИВ128-М — двухъядерный микропроцессор, в котором сочетаются два управляющих ядра архитектуры КОМДИВ64, являющихся усовершенствованием ядра ВМ6, и два 128-разрядных специализированных сопроцессора СР2, аналогичных сопроцессору в составе ВМ7, также с рядом доработок. Сопроцессоры имеют раздельные DMA-каналы доступа к памяти. Коммуникационные каналы RapidIO также раздельные для двух управляющих процессоров. Кроме того, каждое управляющее ядро поддерживает расширение CPV [12] для векторных вычислений с одинарной и двойной точностью. Набор инструкций векторного расширения CPV является полностью разработкой НИИСИ РАН.

Основные функциональные узлы микросхемы:

- 2 ядра процессора для обработки 64-разрядных чисел с фиксированной запятой;
- 2 арифметических сопроцессора для обработки чисел с плавающей точкой одинарной и двойной точности (CPV);
- 2 специализированных сопроцессора, ориентированных на задачи обработки сигналов и поддерживающих вычисления с одинарной точностью (СР2);
- кэш-память инструкций первого уровня объемом 32 Кбайт для каждого ядра на базе DICE ячеек повышенной сбоеустойчивости;
- регистровый файл, содержащий 32 64-разрядных регистра, выполненный на базе DICE ячеек повышенной сбоеустойчивости;
- кэш-память данных первого уровня объемом 16 Кбайт для каждого ядра;
- встроенная кэш-память второго уровня объёмом 512 Кбайт для каждого ядра;

- контроллер кэш-памяти второго уровня, реализующий сбоеустойчивый протокол работы с кэш-памятью;
- системный контроллер, включающий два контроллера динамической памяти, реализующих сбоеустойчивый протокол работы с внешней памятью;
- четыре последовательных дуплексных канала Serial RapidIO 4X/1X с производительностью 3,125 Гбит/с;
- параллельный 8-разрядный канал RapidIO с частотой передачи 250 МГц.

2.1.4. Векторный сопроцессор CPV

Векторный сопроцессор CPV [12] предназначен для расширения функциональных возможностей универсального RISC-микропроцессора при выполнении арифметических операций. Основными характеристиками сопроцессора являются:

- поддержка комплексных и векторных типов данных, представленных числами с плавающей точкой одинарной и двойной точности;
- максимальная ширина вектора 128 бит;
- регистровый файл на 64 128-разрядных регистра;
- выполнение до 10 арифметических операций над вещественными числами двойной точности и до 20 арифметических операций над вещественными числами одинарной точности за один такт;
- расширенный набор векторных команд;
- возможность загрузки/сохранения вектора через кэш-память 1-го уровня или пары векторов через кэш-память 2-го уровня.

Векторный сопроцессор использует семейства инструкций MDMX и MDMX2 в наборе команд MIPS64 release 2.0 [13].

2.1.5. Коммуникационная среда RapidIO

RapidIO — стандартизированная архитектура высокопроизводительной коммуникационной среды, основанная на коммутации пакетов.

Создателем RapidIO и организацией, контролирующей её развитие, является некоммерческая ассоциация RapidIO Trade Association, основанная компаниями Mercury и Motorola (Freescale) в 2000 году.

При разработке архитектуры RapidIO её создатели ставили перед собой несколько целей:

- создать архитектуру коммуникационной среды для систем, нацеленных на параллельную обработку данных; количество абонентов коммуникационной среды может колебаться от нескольких десятков до нескольких десятков тысяч;
- создать единую архитектуру коммуникационной среды для соединения СБИС на печатной плате, соединения нескольких печатных плат в блок и соединения нескольких таких блоков в единый комплекс;
- обеспечить низкие накладные расходы при передаче данных;
- уменьшить сложность создания конечных систем за счёт разделения спецификации коммуникационной среды на части.

Основное назначение коммуникационной среды — обеспечение передачи данных между устройствами, которые осуществляют отправку или приём данных — оконечными устройствами (англ. endpoint).

Подробный анализ стандарта RapidIO и его реализации в аппаратуре НИИСИ РАН приведён в [14-16].

2.2. Программный стек

Для работы с имеющейся аппаратурой в НИИСИ РАН поддерживаются модификации стандартных компиляторов GCC [17]. Все коды, исполняемые на микропроцессорах НИИСИ РАН, должны быть собраны при помощи этих компиляторов.

ВМ6 представляет собой универсальный процессор — на нём могут исполняться типичные вычислительные процедуры, написанные, например, на языке C или FORTRAN.

Для написания вычислительных процедур, оптимизированных для математического сопроцессора СР2 процессора ВМ7 и многопроцессорных комплексов на его базе в НИИСИ РАН был разработан ассемблер [18], ряд управляющих программных механизмов и библиотек специализированных функций. Ниже приводится краткое описание механизмов и библиотек, наиболее важных при разработке прикладного программного обеспечения.

Разрабатываемый в настоящее время микропроцессор ВМ9 представляет собой следующее поколение микропроцессорной архитектуры НИИСИ РАН, обратно совместимой с ВМ7. Кроме того, в управляющем процессоре (CPU) процессора ВМ9 предусмотрена поддержка векторного расширения CPV, которое также полностью является разработкой НИИСИ РАН. Поддержка векторного сопроцессора CPV введена на уровне компилятора GCC по аналогии с популярными векторными расширениями SSE и AVX для семейства архитектур x86.

2.2.1. Программная поддержка контроллера DMA ВМ7

Загрузки данных из DDR2 в локальную память секций математического сопроцессора СР2 К128 и выгрузки обратно осуществляются при помощи механизма дескрипторов контроллера DMA. Дескриптор DMA — это объект, который описывает некоторую операцию пересылки данных. Контроллер DMA имеет «2,5-мерную» архитектуру: поддерживаются пересылки одно-, двух- и, с некоторыми ограничениями, трёхмерных массивов данных (документация [7]). Например, дескриптор, описывающий пересылку двумерного массива, содержит следующую информацию:

- описание области памяти в DDR2: начальный адрес, количество непрерывных строк, длину строки, шаг в памяти между началами строк;
- описание области локальной памяти: начальный адрес, общий для четырёх секций, длина и количество строк в каждой секции, шаг между началами строк;
- направление пересылки;
- некоторую дополнительную информацию.

В целом работа с DMA выглядит следующим образом. Со стороны CPU, который является составной частью микропроцессора ВМ7, формируется очередь дескрипторов из операций загрузки и выгрузки. Имеется функция постановки операции в очередь, но момент фактического начала пересылки программно не контролируется. Также есть функция ожидания завершения всех операций, находящихся в очереди на текущий момент; после её вызова очередь опустошается.

2.2.2. Программная поддержка сопроцессора СР2 ВМ7

Математический сопроцессор СР2 ВМ7 исполняет «ядра» — вычислительные процедуры, написанные на языке ассемблера СР2 ВМ7.

Программно реализован следующий механизм управления заданиями СР2. Управляющий процессор загружает код того или иного задания в память инструкций СР2. Имеются функции запуска задания из памяти инструкций и ожидания завершения текущего задания. Вычисления на СР2 выполняются асинхронно по отношению к работе управляющего процессора: во время выполнения задания на СР2 можно продолжать выполнять те или иные операции на CPU — например, формирование последующих заданий СР2 или очереди дескрипторов DMA.

Таким образом, программисту доступна явная синхронизация с CPU: как завершения пересылок по DMA, так и завершения заданий на СР2. Это позволяет организовать работу так, чтобы вычисления на СР2 с некоторой порцией данных выполнялись параллельно с пересылкой другой порции данных по DMA, и таким образом добиться высокой производительности вычислений.

Входные массивы данных для заданий (вычислительных ядер) СР2 располагаются в его локальной памяти, туда же помещаются массивы результатов. Помимо входных массивов, загружаемых через DMA, в процедуры могут передаваться дополнительные аргументы (размеры массивов, количество итераций циклов и т.п.) — через набор регистров СР2, доступных для записи со стороны CPU. Для записи и чтения таких регистров предусмотрены специальные процедуры.

2.2.3. Программные эмуляторы

Реализованы программные эмуляторы универсальных процессорных ядер ВМ(6-7-9), сопроцессора СР2 ВМ7, контроллера DMA ВМ7, а также векторного сопроцессора CPV. Использование такого рода инструментов позволяет программисту разрабатывать программный код на локальном компьютере, обращаясь к реальному процессору только на этапе тестирования готовых программ.

Программные эмуляторы предоставляют возможность предустанавливать начальное состояние конкретного аппаратного блока (например, значения регистров CP2), а также выводить потактовую информацию о выполненных операциях того или иного вычислителя или DMA-контроллера.

2.2.4. Библиотека цифровой обработки сигналов (БЦОС)

Механизмы, описанные в пунктах 2.2.1 и 2.2.2, реализованы в рамках библиотеки цифровой обработки сигналов (БЦОС, `dsplib`) для ВМ7.

В состав этой библиотеки входит также библиотека оптимизированных математических процедур (`libcp2m`), включающая основные векторные и матричные операции линейной алгебры, быстрое преобразование Фурье и ряд других широко используемых функций численного анализа. Высокая эффективность процедур достигается благодаря использованию аппаратных возможностей CP2 и DMA. Вследствие этого, каждая процедура имеет ряд ограничений (например, выравненность в памяти) на входные и выходные массивы данных.

2.2.5. Обмены по RapidIO

Для программирования обменов данными через каналы RapidIO разработана специализированная библиотека (`libpart`) (рассматривается в [19]).

Кроме того, ведутся работы по обеспечению поддержки стандарта MPI [11].

3. Производительность алгоритмов NPB на аппаратуре НИИСИ РАН

3.1. Референсные реализации NPB

На аппаратуре НИИСИ РАН были протестированы референсные однопроцессорные реализации алгоритмов MG, CG и FT. Реализации являются последовательными и написаны на языке Fortran 77 авторами пакета NPB; все вычисления выполняются с двойной точностью.

Для дальнейшей разработки реализаций, оптимизированных под процессоры линейки КОМДИВ, референсные коды были дополнены вариантами с одинарной точностью. При этом, разумеется, возросла и допустимая погрешность при проверке корректности результатов.

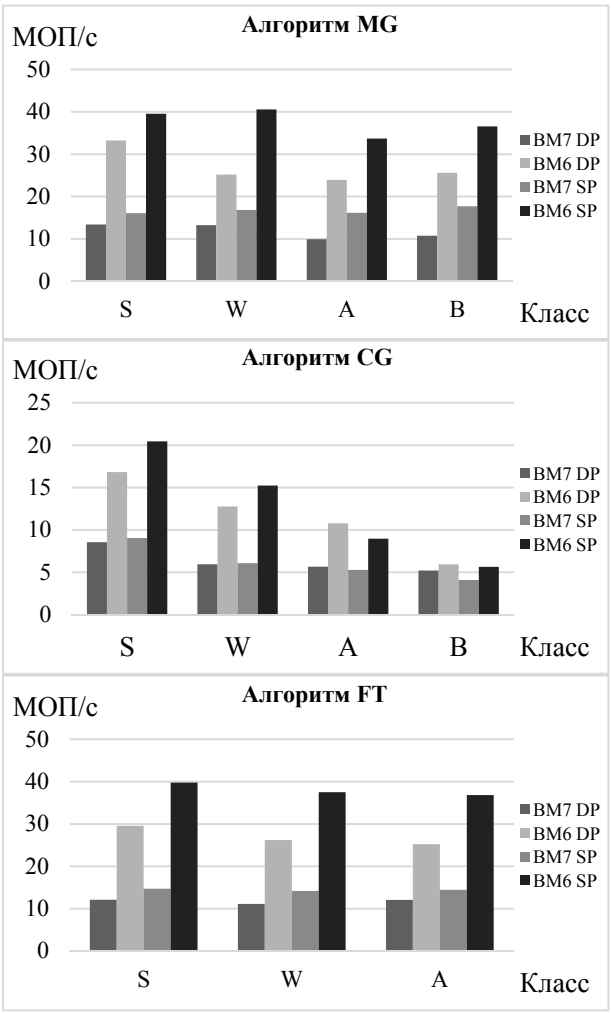

Рис. 3. Референсные реализации MG, CG и FT на FPU процессорных ядер ВМ6 и ВМ7, частота 200 МГц

На Рис. 3 приводятся производительности референсных реализаций алгоритмов MG, CG и FT на FPU процессорных ядер ВМ6 и ВМ7, при работе на частоте 200 МГц. В Табл. 1 для удобства те же данные представлены в числовом виде. DP соответствует вычислениям с двойной точностью (референсный код без модификаций), SP — вычислениям с одинарной точностью (портировано НИИСИ РАН). Здесь и далее производительность приводится в МОП/с — количество операций над числами с плавающей точкой в секунду, делённое на 10^6. Символ (*) в некоторых графах означает, что объёма памяти процессора недостаточно для решения задачи данного класса. Согласно документации, процессорное ядро ВМ7 на одной частоте является менее производительным,

Табл. 1. Референсные реализации MG, CG и FT на FPU процессорных ядер BM6 и BM7, частота 200 МГц

BENCH	PR	ARCH	S	W	A	B
MG	DP	BM7	13.4	13.25	9.91	10.75
		BM6	33.23	25.2	23.92	25.6
	SP	BM7	16.09	16.81	16.13	17.69
		BM6	39.53	40.57	33.7	36.55
CG	DP	BM7	8.55	5.94	5.66	5.2
		BM6	16.82	12.77	10.77	5.93
	SP	BM7	9.04	6.07	5.27	4.09
		BM6	20.46	15.23	8.97	5.63
FT	DP	BM7	12.09	11.11	12.07	*
		BM6	29.57	26.19	25.21	*
	SP	BM7	14.71	14.17	14.45	*
		BM6	39.77	37.51	36.84	*

нежели ядро процессора BM6, поскольку BM7 ориентирован на вычисления с использованием математического сопроцессора CP2. Это отражается и в приведённых результатах замеров: разница в производительности составляет примерно 2 с лишним раза на всех алгоритмах. Падение производительности CG с ростом класса является существенной особенностью алгоритма: вычисления требуют случайного доступа к памяти, и при увеличении объёма данных увеличивается количество кэш-промахов.

Реализации были также протестированы на ПЛИС-прототипе процессора BM9. Замеры на ПЛИС производились с тактовой частотой 62,5 МГц. Что касается реального кристалла BM9, то его проектная тактовая частота составит 1000 МГц. На Рис. 4 и в Табл. 2 приводятся результаты замеров производительности SP-версии кодов для FPU на BM6, BM7 и ПЛИС BM9, промасштабированные до 1000 МГц. Это позволяет оценить ожидаемую производительность выбранных алгоритмов на реальном процессоре BM9 и увидеть влияние на производи-

Табл. 2. Референсные SP-реализации тестов на FPU BM6, BM7 и BM9, в пересчёте на 1000 МГц

BENCH	ARCH	S	W	A	B
MG	BM7	80.45	84.05	80.65	88.45
	BM6	197.65	202.85	168.5	182.75
	BM9	250.56	258.08	245.92	265.12
CG	BM7	45.2	30.35	26.35	20.45
	BM6	102.3	76.15	44.85	28.15
	BM9	159.04	128.64	116.0	77.6
FT	BM7	73.55	70.85	72.25	*
	BM6	198.85	187.55	184.2	*
	BM9	236.96	230.56	235.04	*

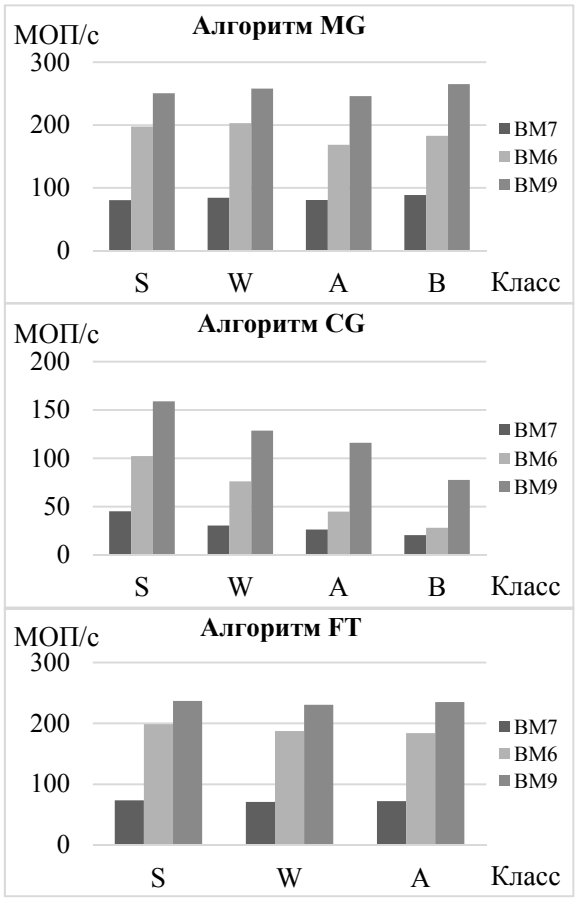

Рис. 4. Референсные SP-реализации тестов на FPU BM6, BM7 и BM9, в пересчёте на 1000 МГц

тельность качественных изменений в ядре управляющего процессора.

Можно видеть, что эффективность вычислений на перспективном микропроцессоре ВМ9 в 3 раза выше, чем на ВМ7, для всех рассматриваемых алгоритмов, и выше, чем на ВМ6, на 45% для алгоритмов MG и CG и на 20% для алгоритма FT.

3.2. Оптимизированные однопроцессорные реализации

В процессоре ВМ9 предполагается два основных способа повышения производительности вычислений. Первый способ — использование следующей итерации математического сопроцессора СР2. Второй — использование векторного расширения CPV, при выполнении вычислений непосредственно в основном потоке команд CPU.

В настоящее время модификации компиляторов GCC НИИСИ РАН поддерживают авто-векторизацию для CPV: код для CPU компилируется со специальными ключами, и на выходе получаются объектные файлы, которые используют машинные инструкции CPV. На Рис. 5 и в Табл. 3 приводятся результаты запусков на ПЛИС ВМ9 SP версий с авто-векторизацией (CPV) и, для сравнения, без неё (FPU). Частота — 62,5 МГц.

Прирост производительности за счёт использования CPV составляет порядка 25% на всех алгоритмах. Теоретический максимум производительности вычислений с использованием CPV — в 4 раза выше, чем вычислений только на FPU, поскольку CPV использует 128-битный вектор, что соответствует четырём вещественным числам одинарной точности. Этот максимум, однако, в реальности недостижим, поскольку любой алгоритм может быть векторизован лишь частично. С учётом того, что для ВМ9 использовались автоматические средства векторизации, а не вручную написанные оптимизированные процедуры, полученные результаты можно считать удовлетворительными.

Для алгоритма MG была также разработана оптимизированная версия кода с использованием математического сопроцессора СР2 — для ВМ7 и для ВМ9. Реализация проведена следующим образом: написан набор необходимых вычислительных ядер на ассемблере СР2 и соответствующих процедур для CPU. Вызов этих процедур встроен в код референсных реализаций, вместо вычислений на CPU. Часть кода, относящаяся к генерации данных, проверке результата и замерам производительности, осталась без изменений.

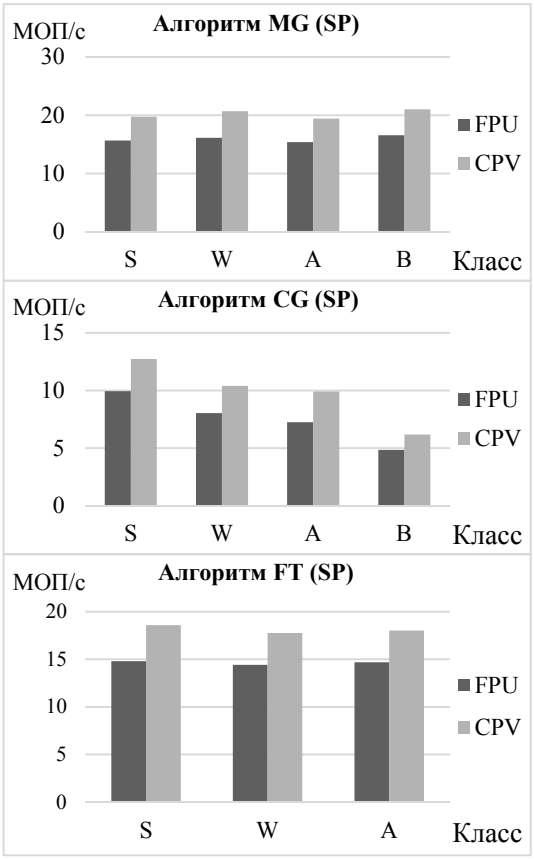

Рис. 5. Авто-векторизация CPV на ПЛИС ВМ9, частота процессорного ядра 62,5 МГц

Табл.3. Авто-векторизация CPV на ПЛИС ВМ9, частота процессорного ядра 62,5 МГц

BENCH	UNIT	S	W	A	B
MG	FPU	15.66	16.13	15.37	16.57
	CPV	19.76	20.67	19.41	21.02
CG	FPU	9.94	8.04	7.25	4.85
	CPV	12.74	10.41	9.92	6.18
FT	FPU	14.81	14.41	14.69	*
	CPV	18.59	17.77	18.01	*

На Рис. 6 и в Табл. 4 приводятся результаты замеров производительности версии MG для СР2 на ВМ7, с тактовой частотой 200 МГц, и на ПЛИС ВМ9, промасштабированные до частоты существующего оборудования 200 МГц (в отли-

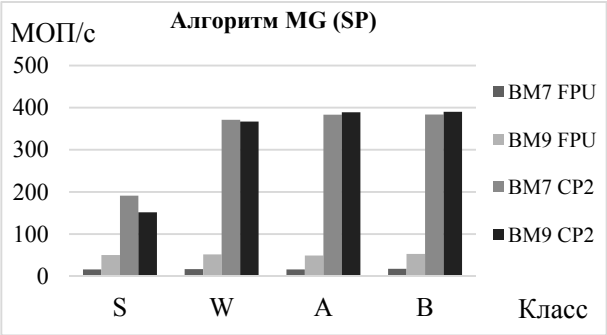

Рис. 6. Алгоритм MG на CP2 BM7 и BM9, частота процессорного ядра 200 МГц

Табл. 4. Алгоритм MG на CP2 BM7 и BM9, частота процессорного ядра 200 МГц

BNCH	ARCH	UNIT	S	Q	A	B
MG	BM7	FPU	16.09	16.81	16.13	17.69
		FPU +CP2	191.28	370.98	383.26	383.65
	BM9	FPU	50.11	51.62	49.18	53.02
		FPU +CP2	151.8	367.12	388.92	390.16

чие от Рис. 4, где масштабирование производилось к проектной частоте BM9). Для сравнения приводятся также производительности референсных реализаций без использования CP2.

Использование CP2 требует трудоёмкой работы высококвалифицированного программиста, однако даёт многократный прирост производительности, по сравнению с вычислениями на CPU: более чем в 20 раз на BM7, более чем в 7 раз на BM9. Прирост производительности на BM9 меньше, чем на BM7, за счёт более производительного CPU.

Высокая производительность вычислений на CP2 достигается благодаря трём фактором. Во-первых, разработанные на языке ассемблера ядра максимально эффективно используют аппаратные возможности сопроцессора. Во-вторых, канал доступа к памяти через DMA достаточно производителен (до 75% эффективности утилизации подсистемы памяти, Рис. 7), чтобы время подгрузки необходимых данных в ходе работы было существенно меньше времени вычислений на CP2. В-третьих, используется возможность совмещения вычислений с пересылками данных, и таким образом большая часть пересылок оказывается вынесена на фон вычислений.

Версия с использованием CP2 была также разработана для процедуры SpMV, которая является базовой в алгоритме CG. Разреженная матрица была упакована в формате, который можно условно назвать «sliced CSR». Здесь используется та же идея, что в формате «sliced ELLPACK», предложенном в [22]: матрица нарезается на «слои» из некоторого фиксированного количества рядов, и в каждом слое ряды дополняются нулями так, чтобы длина всех рядов в одном слое была одинаковой. Однако каждый слой упаковывается в формате не ELL, а классическом CSR (например, [23]). Формат ELL обеспечивает выравненный доступ к элементам матрицы, однако на BM7, в отличие от графических ускорителей, это не имеет большого значения, т.к. скорость чтения данных из локальной памяти CP2 в регистры не зависит от последовательности адресов. Для распределения данных по вычислительным секциям при загрузке в локальную память CP2 оказывается более удобен формат CSR. Дополнение нулями необходимо, чтобы обеспечить выравненность адресов в хостовой памяти при пересылках по DMA, и кроме того, сократить накладные расходы на пересылки и запуски ядра за счёт единообразной обработки больших массивов данных.

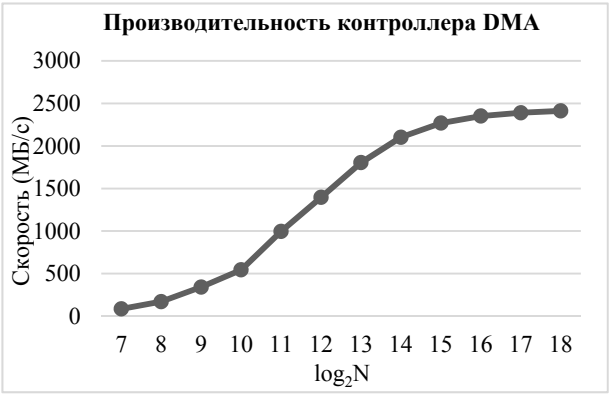

Рис. 7. Производительность контроллера DMA BM7 с DDR2 RAM PC2-3200 (DDR2-400), на частоте 200 МГц; N — объём данных в Байт, пиковая скорость передачи данных 3200 МБ/с

Выбранный формат накладывает ограничения на размеры матриц, которые могут быть обработаны на одном процессоре. Для вычисления SpMV на CP2 необходимо хранение ко-

пии входного вектора целиком в обеих половинах локальной памяти каждой секции. В частности, из всех классов NPB только матрица из алгоритма CG класса S удовлетворяет этому ограничению.

Разработанная процедура SpMV на CP2 и, для сравнения, простейшая процедура SpMV для матрицы в формате CSR на ядре управляющего процессора (CPU) были протестированы на микропроцессоре ВМ7 и на ПЛИС ВМ9. Рассматривалась матрица из алгоритма CG класса S, а также матрицы, сгенерированные со случайным расположением и значениями ненулевых элементов, с коэффициентом заполнения 0,06 (т.е. матрицы, в которых 6% всех элементов являются ненулевыми). Замеры показали, что при обработке различных случайных матриц одного и того же порядка производительность варьируется не более чем на 1-2%. Кроме того, производительность процедуры на CP2 зависит от выбранной ширины слоя H (количества строк в одном слое): наибольшая производительность достигается, когда H является целой степенью 2 и максимально возможное, с учётом того, что один слой должен помещаться в память секций CP2. Результаты замеров приводятся на Рис. 8 и в Табл. 5, в которой в скобках указаны также оптимальные значения H.

Несмотря на описанные трудности, разработанная процедура SpMV для CP2 на достаточно

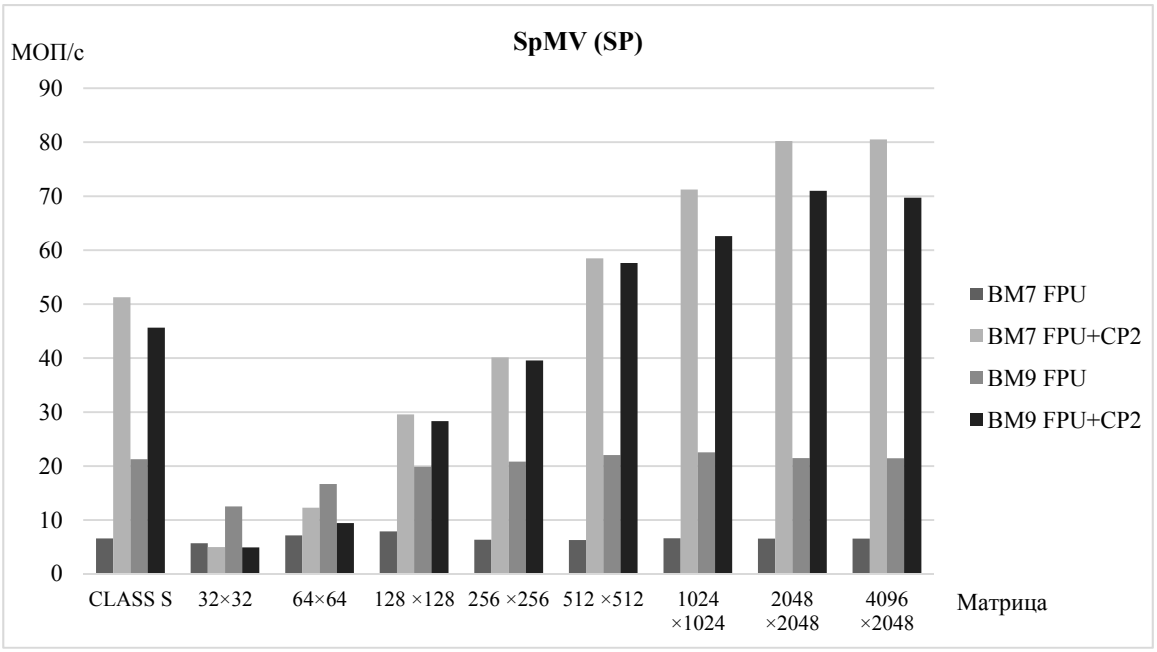

Рис. 8. SpMV на ВМ7 и ВМ9, частота процессорного ядра 200 МГц

Табл. 5. SpMV на ВМ7 и ВМ9, частота процессорного ядра 200 МГц

BNCH		SPMV			
ARCH		BM7		BM9	
UNIT		FPU	FPU+CP2	FPU	FPU+CP2
MATRIX	CLASS S	6.6	51.28 (64)	21.28	45.64 (64)
	32×32	5.71	4.98 (32)	12.52	4.92 (32)
	64×64	7.15	12.28 (64)	16.68	9.44 (64)
	128×128	7.89	29.58 (128)	19.88	28.32 (128)
	256×256	6.36	40.14 (256)	20.84	39.56 (256)
	512×512	6.31	58.5 (256)	22.04	57.64 (256)
	1024×1024	6.61	71.25 (64)	22.56	62.6 (128)
	2048×2048	6.57	80.21 (32)	21.48	71.0 (32)
	4096×2048	6.57	80.52 (32)	21.44	69.72 (32)

больших матрицах показывает более высокую производительность, чем вычисления на управляющем процессоре. Вычисления на FPU выполняются быстрее только на малых объёмах данных: на матрицах порядка 256×256 рост производительности останавливается, в то время как производительность на СР2 продолжает повышаться с ростом объёма вычислений.

На ВМ9, с более производительным CPU, чем у ВМ7, использование СР2 даёт выигрыш более чем в 2 раза на матрице из алгоритма CG класса S, и более чем в 3 раза на больших случайных матрицах. Некоторое снижение производительности на СР2 ВМ9, по сравнению с СР2 ВМ7, связано с текущей доработкой контроллера памяти.

В ядре SpMV на СР2 самой затратной операцией оказывается чтение элементов входного вектора (x). Для каждого элемента матрицы номер столбца, т.е. индекс соответствующего элемента x, передаётся в процедуру во входном массиве и заранее не известен. Эти индексы в общем случае различны для элементов матрицы, которые обрабатываются одновременно в четырёх секциях, и более того, для пары элементов в каждой секции, составляющих одно 64-битное слово. Другими словами, на одну загрузку элементов матрицы приходится 8 загрузок элементов x, и, в силу SIMD-архитектуры СР2, эти загрузки могут осуществляться только последовательно. Замеры, проведённые для процедуры SpMV на одном ВМ7, показали, что в среднем вычисления в ядре на СР2 занимают в 4 раза больше времени, чем соответствующие пересылки по DMA, при этом в ядре 94% всех тактов приходится на зачитку элементов x.

В настоящий момент ведётся доработка процедуры SpMV. В частности, одна из возможных оптимизаций — предварительная перестановка строк и/или столбцов матрицы с последующим разделением матрицы на блоки. Ожидается, что это позволит загружать входной вектор в локальную память по частям, и таким образом, обрабатывать более крупные матрицы. Также за счёт перегруппировки можно уменьшить количество дополнительных нулей в упакованной матрице. В дальнейшем процедуру SpMV на СР2 планируется встроить в референсный код алгоритма CG, аналогично тому, как это было сделано для MG.

Версия алгоритма FT для СР2 к настоящему моменту также находится в процессе реализации. Можно ожидать, что такая реализация покажет высокую производительность: математический сопроцессор СР2 ориентирован на вычисление БПФ, в частности, имеет инструкцию «бабочка Фурье», позволяющую выполнять 10 арифметических операций за такт в каждой секции, и память коэффициентов БПФ. Имеются реализации для ВМ7 других процедур, сводящихся к двумерному БПФ [20], которые работают с эффективностью порядка 45% от пика, что в несколько раз превосходит показатели массовых универсальных процессоров, для которых эффективность на операциях БПФ варьируется в пределах 5-15%. Например, в [21] двумерное БПФ порядка 256×256 реализовано на процессоре Intel Xeon E5-2670 с производительностью 22 ГОП/с, что составляет 13% от теоретического пика (166,5 ГОП/с для вычислений с двойной точностью). Процедура длинного одномерного БПФ длины 65536, сводящаяся к тому же двумерному БПФ порядка 256×256, реализованная на ВМ7, имеет эффективность 49% от пика. Трёхмерное БПФ, являющееся базовой операцией алгоритма FT, может быть реализовано по описанным в [20] принципам, аналогично двумерному.

При вычислениях на ВМ9 представляется оправданным совместное использование СР2 и CPV. Для этого необходимо выделить в алгоритме как можно большую часть работы, которая сводится к однотипным вычислениям над большими объёмами данных, и реализовать эту часть в виде процедуры для СР2. Оставшаяся часть алгоритма — например, вычисления, содержащие большое количество условных переходов или обращений к массивам по не известным заранее адресам — реализуется на CPU и по возможности оптимизируется за счёт использования CPV. Вопрос об оптимальном распределении работы между СР2 и CPU требует дополнительного исследования для каждого конкретного алгоритма.

3.3. Оценки производительности для многопроцессорного комплекса

Помимо последовательных версий кодов, имеются также параллельные референсные реализации алгоритмов NPB с использованием

MPI. При условии поддержки стандарта MPI для пересылок по каналам RapidIO, эти реализации могут быть запущены на многопроцессорных комплексах НИИСИ РАН. Также могут быть разработаны и версии, в которых вычисления на процессоре оптимизированы с использованием CP2 и/или CPV, а обмены данными осуществляются с использованием стандарта MPI. Для алгоритма MG такая версия уже разработана и верифицирована на эмуляторе, для 1-8 MPI-процессов: используются те же ядра CP2, что и в однопроцессорной версии, а референсные MPI-обмены заменены на аналогичные, но для массивов в специальном формате, удобном для вычислений на CP2.

Разработанные MPI-версии алгоритма MG — референсная SP-версия и версия с использованием CP2 — были также протестированы на двух ПЛИС ВМ9. Производительности на одном и двух ПЛИС приводятся в Табл. 6, для рабочей частоты ПЛИС 50 МГц.

Как показывают полученные результаты, для задачи класса S, с малым объёмом обрабатываемых данных, более выгодно производить вычисления на одном процессоре, чем разделять на несколько, т.к. накладные расходы оказываются слишком велики. Задачи классов A и B демонстрируют хорошую масштабируемость на два процессора — с коэффициентом приблизительно 1,75 для версии с CP2.

В настоящий момент в НИИСИ РАН идут работы, связанные с аппаратной и системной поддержкой комплексов из 4 и более ПЛИС ВМ9. В [11] приводятся результаты тестирования MPI-обменов по RapidIO для процессоров ВМ6, в том числе с использованием теста NPB. Зная объёмы данных, пересылаемых в ходе работы алгоритмов, и значения скоростей, а также производительность вычислений на одном узле, можно оценить производительность на многопроцессорном комплексе. Из всех тестов, проведённых в [11], наиболее приближенными к реальной ситуации являются обмены между двумя процессорами в обе стороны одновременно. Замеры показали, что скорость передачи данных зависит от их объёма. Приведём на Рис. 9 проектные оценки скоростей MPI-обменов по RapidIO между процессорами ВМ9 в виде промасштабированных до 1000 МГц

Табл. 6. MG на двух ПЛИС ВМ9, с MPI-обменами по RapidIO, частота процессорного ядра 50 МГц

BENCH	UNIT	NP	S	W	A	B
MG	FPU	1	11.54	12.06	11.64	12.59
		2	9.27	21.86	21.91	23.53
	FPU + CP2	1	37.81	90.79	97.47	97.69
		2	5.74	105.82	170.78	171.25

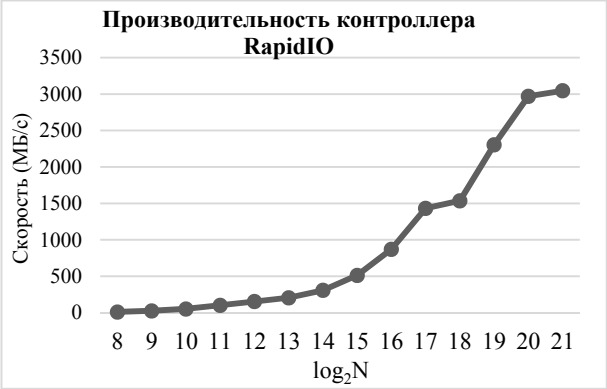

Рис.9. Оценки скорости MPI-обменов по RapidIO между ВМ9, частота процессорного ядра 1000 МГц; N — объём данных в Байт

значений, полученных на ВМ6 на частоте 200 МГц. Предполагаемый пик скорости передачи данных составляет 3,2 ГБ/с.

Оценки производительности для 2-8 процессоров могут быть получены следующим образом. Время вычислений считается равным времени вычислений на одном процессоре, делённому на общее количество процессоров. Время пересылок по RapidIO оценивается как сумма времён всех пересылок, выполняемых в алгоритме; для каждой пересылки определяется объём данных и соответствующая скорость по графику на Рис. 9. Общее время работы оценивается как сумма времени на все вычисления и на обмены — отсюда находится и производительность, как количество арифметических операций (по формуле из референсного кода), делённое на полученную оценку времени работы. В Табл. 7 полученные оценки для двух процессоров сравниваются с результатами реальных замеров из Табл. 6, промасштабированными до 1000 МГц. В последней колонке указано, сколько реальная производительность составляет от теоретической оценки.

Табл. 7. Алгоритм MG на двух BM9 с RapidIO, частота процессорного ядра 1000 МГц

NP	UNIT	CLASS	THEOR	PRACT	RATIO
2	FPU	S	387.4	185.4	48%
		W	594.0	437.2	74%
		A	580.2	438.2	76%
		B	627.4	470.6	75%
	FPU + CP2	S	692.0	114.8	17%
		W	3325.0	2116.4	64%
		A	3819.2	3415.6	89%
		B	3827.8	3425.0	89%

На Рис. 10 приводятся значения производительности алгоритма MG на комплексе из 1-8 процессоров BM9, соединённых через RapidIO, при работе на частоте 1000 МГц. На первой диаграмме представлены оценки для референсной SP-версии кодов, на второй — для версии с использованием CP2. Для одного и двух процессоров приводятся промасштабированные реальные значения, для четырёх и восьми — теоретические оценки.

Время на обмены данными по RapidIO оказывается незначительным по сравнению со временем вычислений на FPU управляющего процессора, поэтому с ростом количества процессоров производительность масштабируется линейно. При использовании CP2 ситуация меняется: сами вычисления происходят на порядок быстрее, чем на FPU, поэтому на маленьких размерах задач (классы S и W), когда данные передаются небольшими объёмами (а значит, с малой скоростью), пересылки существенно замедляют исполнение алгоритма. Однако с ростом объема вычислительной работы время на вычисления возрастает в несколько раз быстрее, чем время на пересылки, и таким образом, для задач класса A или B можно ожидать масштабируемость, близкую к линейной.

Следует иметь в виду, что данные оценки являются ориентировочными оценками сверху. При их получении, во-первых, накладные расходы на разделение работы на несколько MPI-процессов были учтены только эвристически. Во-вторых, топология соединения процессоров в комплексе предполагалась такой, что между любыми двумя процессорами, между которыми требуется производить обмены, существует

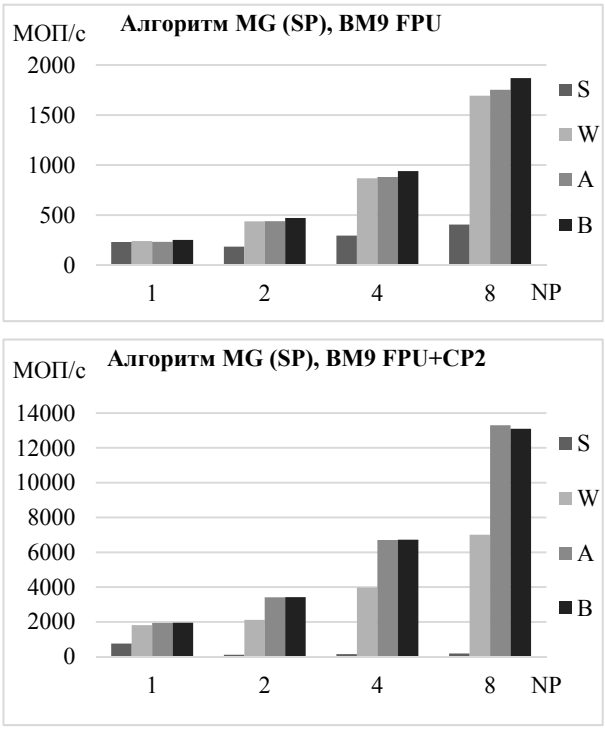

Рис. 10. Алгоритм MG на комплексе из 1-8 BM9 с RapidIO, частота процессорного ядра 1000 МГц

прямой канал передачи данных. Для алгоритма MG последнее означает, что процессоры в общем случае соединены между собой в трёхмерный тор: каждый процессор обменивается данными с шестью соседними (с двумя по каждой координатной оси). Для 2, 4 и 8 процессоров трёхмерный тор вырождается, соответственно, в отрезок, квадрат или куб, в которых любые два соседних процессора соединены двумя каналами — такие топологии проиллюстрированы на Рис. 11.

Если в реальности топология будет неудачной, и скорость обменов окажется, например, ниже в 10 раз, то на задаче класса A на 8 процессорах время пересылок окажется приблизительно равным времени вычислений с использованием

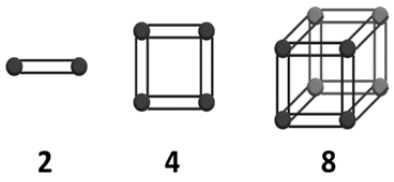

Рис. 11. Схема соединения 2, 4 и 8 процессоров, оптимальная для алгоритма MG

СР2. Таким образом, производительность упадёт вдвое по сравнению с полученной теоретической, однако даже в этом случае она будет в несколько раз выше производительности вычислений на одном процессоре, и тем более — вычислений без использования СР2.

Практика решения различных задач на комплексах из микропроцессоров ВМ7 показала, однако, что даже при существующих технологических ограничениях возможно подобрать такую топологию RapidIO-сети, что потери в скорости пересылок составляют не более 25 %.

Аналогичные оценки производительности были проведены для референсных SP-версий алгоритмов CG и FT. Результаты приводятся на Рис. 12. Из диаграмм видно, что в обоих алгоритмах время обменов по RapidIO оказываеся пренебрежимо мало по сравнению со временем вычислений на управляющем процессоре, поэтому алгоритмы масштабируются линейно с ростом числа процессоров. Однако вычисления в алгоритмах CG и FT могут быть оптимизированы с использовнием СР2, аналогично тому, как это было сделано для алгоритма MG.

В алгоритме CG доля обменов в общем времени работы меньше, чем в алгоритме MG. С другой стороны, уже разработанное ядро SpMV на СР2 даёт меньший относительный прирост производительности по сравнению с CPU, чем ядра алгоритма MG. Отсюда можно сделать вывод, что при вычислении CG с использованием СР2 на многопроцессорном комплексе производительность по-прежнему будет определяться временем вычислений, а не обменов, и масштабирование на больших классах задач будет близко к линейному.

Другая ситуация в алгоритме FT. Архитектура СР2 ВМ7 и ВМ9 изначально спроектирована для эффективного вычисления БПФ, поэтому использование СР2 даст многократный прирост производительности вычислений на одном процессоре. Однако при работе на многопроцессорном комплексе в ходе вычислений требуется производить обмены большими объёмами данных. За счёт того, что на больших объёмах каналы RapidIO используются с эффективностью, близкой к теоретически максимальной, время на вычисление БПФ на одном процессоре оказывается сопоставимо со временем пересылок входных и выходных данных; в [20] приводятся результаты замеров для других алгоритмов, основанных на БПФ, которые подтверждают эту оценку. Для повышения эффективности обменов на ВМ9 имеется аппаратная возможность выполнения вычислений параллельно с обменами по RapidIO, благодаря чему часть обменов оказывается «скрыта» за вычислениями. Однако наиболее важным фактором для алгоритма FT является топология соединения процессоров: снижение скорости обменов в несколько раз повлечёт соответствующее падение общей производительности вычислений.

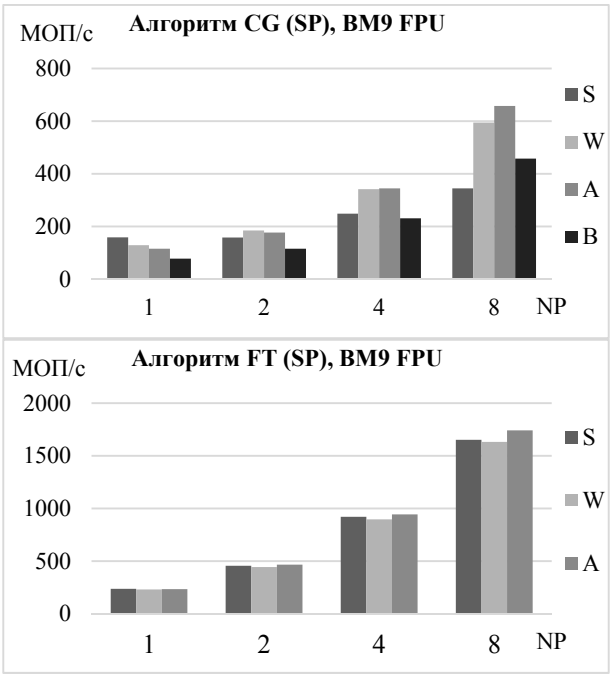

Рис. 12. Алгоритмы CG и FT на комплексе из 1-8 ВМ9 с RapidIO, частота процессорного ядра 1000 МГц

4. Выводы

4.1. Характеристики алгоритмов

В работе рассмотрены три класса алгоритмов, приближенных к реальным расчетным задачам. Все эти алгоритмы могут быть эффективно реализованы на аппаратуре НИИСИ РАН: на одном микропроцессоре линейки КОМДИВ и на многопроцессорном вычислительном комплексе. Микропроцессоры НИИСИ РАН имеют два основных средства повышения производительности вычислений: векторное расширение

CPV и математический сопроцессор CP2. Наибольший прирост производительности обеспечивается именно использованием CP2, однако перспективной возможностью является также совместное использование CPV и CP2, в зависимости от специфики конкретных вычислительных стадий алгоритмов.

Производительность алгоритма MG на одном микропроцессоре определяется в первую очередь производительностью вычислительного ядра на CP2: высокой пропускной способности подсистемы памяти с использованием DMA-контроллера достаточно для выгрузки и загрузки всех необходимых данных на сопроцессор на фоне вычислений. В многопроцессорном комплексе пропускная способность RapidIO также достаточно высока, чтобы время обменов было незначительно по сравнению со временем вычислений. Наиболее подходящей для данного алгоритма топологией соединения процессоров является трёхмерный тор.

Сходная ситуация и для алгоритма CG: производительность также будет напрямую зависеть от производительности вычислительного ядра. В ядре большие накладные расходы связаны с чтением данных из локальной памяти, поскольку CP2 является SIMD-сопроцессором, а в алгоритме CG требуется, напротив, доступ к данным по случайным, не известным заранее адресам. Время на пересылки данных, как по DMA, так и по RapidIO между процессорами, будет пренебрежимо мало по сравнению со временем работы ядра.

Для алгоритма FT эффективность вычислений на одном процессоре будет достаточно высока, благодаря специализированной архитектуре CP2 и сбалансированности производительности CP2 и DMA для алгоритмов, основанных на БПФ. На многопроцессорном комплексе производительность будет существенно зависеть от производительности обменов по RapidIO, в частности, от топологии соединения процессоров.

В целом можно сделать вывод, что для всех рассмотренных алгоритмов значительное влияние на эффективность вычислений оказывает производительность подсистемы памяти, на том или ином уровне: чтение данных из локальной памяти на CP2, либо пересылки между CP2 и системной памятью по DMA, либо обмены по RapidIO между процессорами. Стоит особо отметить, что существенный прирост производительности при вычислениях на CP2 большей частью обеспечен за счёт крайне высокой утилизации подсистемы памяти: на уровне 75% от всей ширины канала доступа к DDR2 через DMA.

В дальнейшем планируется разработка версий алгоритмов CG и FT с использованием CP2 — как последовательных, для одного процессора, так и параллельных с использованием MPI, для многопроцессорного комплекса.

4.2. Сравнение разработок НИИСИ РАН и мировых производителей

В Табл. 8 приводятся сравнительные результаты замеров производительности референсных кодов MG, CG, и FT на BM9 (ПЛИС 62.5 МГц) с одним ядром современного процессора Intel Xeon E5-2690v2 с поддержкой векторного расширения AVX. Частота BM9 промасштабирована до частоты E5-2690v2, которая составляет 3 ГГц.

В Табл. 9 показано соотношение производительности референсной реализации MG одинарной точности на E5-2690v2 и оптимизированной версии для CP2 BM9 в пересчёте на частоту ядра в 3 ГГц.

Приведённые значения показывают, что управляющий процессор BM9 с использованием CPV в настоящее время существенно уступает универсальным процессорам от Intel (от 3-х до 8-ми раз в зависимости от задачи). Однако пример алгоритма MG показывает, что использование математического сопроцессора CP2 позволяет добиться эффективности вычислений, сопоставимой с вычислениями на современных универсальных процессорах от Intel топовых моделей.

Показатели эффективности вычислений на BM9, приведённые в Табл. 4 и Табл. 5, не являются окончательными и могут быть существенно улучшены путём доработки как кодов, так и архитектуры самого процессора. Ожидаемая эффективность версии алгоритма CG, оптимизированной с использованием CP2, превышает приведённую в 2 и более раз. Алгоритм FT может быть реализован на CP2 с эффективностью порядка 50% от пика, что составляет 20 ОП/с на 1 Гц и во много раз превышает эффективность вычисления на одном ядре Xeon E5-2690v2.

Табл. 8. Сравнение референсных реализаций MG, CG и FT на BM9 и Intel Xeon E5-2690v2, частоты приведены к 3 ГГц

BNCH	PR	ARCH	S	W	A	B
MG	DP	BM9 CPV	815	793	722	776
		E5-2690v2 AXV (6 вхождений)	3896	4070	3980	4282
	SP	BM9 CPV	948	992	931	1009
		E5-2690v2 AXV (6 вхождений)	5902	8407	7769	8406
CG	DP	BM9 CPV	539	444	432	144
		E5-2690v2 AXV (6 вхождений)	2285	2244	1954	672
	SP	BM9 CPV	611	499	476	296
		E5-2690v2 AXV (6 вхождений)	2408	2279	2160	900
FT	DP	BM9 CPV	743	657	703	*
		E5-2690v2 AXV (6 вхождений)	2825	2694	2355	2257
	SP	BM9 CPV	892	853	864	*
		E5-2690v2 AXV (6 вхождений)	3515	3340	3263	3287

Табл. 9. Сравнение реализаций MG: одинарной точности (SP) на CP2 BM9 и референсной SP-версии на одном ядре Intel Xeon E5-2690v2 AVX; частоты приведены к 3 ГГц

ARCH	S	W	A	B
BM9 CP2	2268	5448	5850	5862
E5-2690v2	5902	8407	7769	8406

4.3. Предложения к дальнейшей оптимизации микропроцессорной архитектуры НИИСИ РАН

Возможности оптимизации вычислений по тому или иному алгоритму зависят от архитектурных особенностей как отдельно взятого вычислителя, так и распределенного вычислительного комплекса в целом.

Дополнение набора инструкций математического сопроцессора CP2 специальными командами позволит сократить время на вычисления в ядре алгоритма MG и повысит производительность до двух раз.

Для повышения эффективности ядра SpMV и других процедур с нерегулярным доступом к памяти необходима возможность загрузки данных из локальной памяти четырёх секций CP2 параллельно по различным адресам. Ожидаемый прирост производительности — в 4 раза.

При пересылке данных по DMA в ряде алгоритмов, таких как MG и CG, присутствует некоторая избыточность. Чтобы её избежать, требуется ослабить требование к выравниванию данных: необходима возможность пересылки данных 32-разрядными словами, причём адрес в DDR2 должен быть выравнен только на 4 байта.

Увеличение объёма накристальной памяти сопроцессора CP2 позволит обрабатывать данные бóльшими блоками, благодаря чему повысится эффективность вычислений для больших классов задач.

Для того чтобы удовлетворить требованиям к точности вычислений, предъявляемым в современных расчётных задачах, необходима поддержка вычислений на CP2 с двойной точностью.

Для эффективных обменов данными между процессорами требуется возможность соединения процессоров в трёхмерный тор — соответственно, каждый процессор должен обла-

дать как минимум шестью внешними коммуникационными портами.

Помимо архитектурных усовершенствований, важна также разработка соответствующего программного обеспечения. Так, разработка библиотеки оптимизированных математических процедур для CPV, по аналогии с `dsplib` для CP2, позволит в полной мере использовать возможности векторного расширения и повысить эффективность вычислений на управляющем процессоре. Поддержка MPI упростит разработку нового и портирование существующего ПО на многопроцессорные комплексы НИИСИ РАН, что будет способствовать более широкому применению разработок НИИСИ РАН в области высокопроизводительных вычислений.

Литература

1. Dominic Sweetman. See MIPS Run, 2nd Edition. — Morgan Kaufmann Publishers, 2006. — 512 pp.
2. RapidIO Specifications // URL: http://www.rapidio.org/rapidio-specifications/
3. VMEbus Technology FAQ // URL: http://www.vita.com/page-1855175
4. Doug Abbott. PCI Bus Demystified, 2nd edition. — Newnes, 2004. — 250 pp.
5. Ravi Budruk, Don Anderson, Tom Shanley. PCI Express System Architecture. — Addison-Wesley Professional, 1999. — 832 pp.
6. TigerSHARC Processor Benchmarks // URL: http://www.analog.com/ru/design-center/landing-pages/001/tigersharc-benchmarks.html
7. Микросхема интегральная 1890ВМ7Я (КОМДИВ128-РИО). Указания по применению. ЮКСУ.431281.104Д4. — Москва: НИИСИ РАН, 2009. — 371 с.
8. MPI: A message-passing interface standard, version 3.0. // Message Passing Interface Forum, 2012. URL: http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf
9. D. Bailey, E. Barszcz, J. Barton, D. Browning, R. Carter, L. Dagum, R. Fatoohi, S. Fineberg, P. Fred-erickson, T. Lasinski, R. Schreiber, H. Simon,V.Venkatakrishnan and S.Weeratunga. The NAS Parallel Benchmarks // RNR Technical Report RNR-94-007, March 1994.
10. NAS Parallel Benchmarks // URL: https://www.nas.nasa.gov/publications/npb.html
11. Кулешов А.С. Поддержка протокола MPI в ядре ОС Linux для многопроцессорных вычислительных комплексов на базе высокоскоростных каналов RapidIO // Программные продукты и системы, № 4, 2015. - С. 93--98.
12. Бобков С.Г., Аряшев С.И., Зубковский П.С. Арифметические сопроцессоры микропроцессоров с архитектурой КОМДИВ // 6-ой Московский суперкомпьютерный форум. Москва, 2015.
13. MIPS Extension for Digital Media with 3D. — MIPS Technologies, Inc., 1997. — 29 pp.
14. Павлов А. Н. Обзор коммуникационной среды RapidIO // Моделирование и визуализация. Многопроцессорные системы. Инструментальные средства разработки ПО / Сборник статей под редакцией академика РАН В. Б. Бетелина. — М.: НИИСИ РАН, 2009. — с.105 –122.
15. Павлов А.Н. Формальная модель RapidIO // Моделирование и визуализация. Многопроцессорные системы. Инструментальные средства разработки ПО / Сборник статей под редакцией академика РАН В. Б. Бетелина. — М.: НИИСИ РАН, 2009. — с.123 –131.
16. Павлов А. Н. Программная поддержка RapidIO // Моделирование и визуализация. Многопроцессорные системы. Инструментальные средства разработки ПО / Сборник статей под редакцией академика РАН В. Б. Бетелина. — М.: НИИСИ РАН, 2009. — с.132 –147.
17. GCC online documentation // URL: https://gcc.gnu.org/onlinedocs/
18. Программное изделие Ассемблер для специализированного сопроцессора СР2 в составе микропроцессора КОМДИВ128-РИО (АССК128). Руководство программиста. ЮКСУ.90986-01 33 01. — Москва: НИИСИ РАН, 2013. — 67 с.
19. Райко Г.О., Павловский Ю. А., Мелькановић В.С. Технология программирования многопроцессорной обработки гидроакустических сигналов на вычислительных устройствах семейства «КОМДИВ» // Гидроакустика. Вып. 20 (2). — СПб.: ОАО «Концерн "Океанприбор"», 2014. — 118 с.
20. Сударева О.Ю. Эффективная реализация алгоритмов быстрого преобразования Фурье и свёртки на микропроцессоре КОМДИВ128-РИО. — Москва: НИИСИ РАН, 2014. — 266 с.
21. Daisuke Takahashi. An Implementation of Parallel 2-D FFT Using Intel AVX Instructions on Multi-core Processors // Algorithms and Architectures for Parallel Processing: 12th International Conference (ICA3PP 2012), September 4-7, 2012, Fukuoka, Japan. Proceedings, Part II, pp.197–205.
22. A. Monakov, A. Lokhmotov, A. Avetisyan. Automatically tuning sparse matrix-vector multiplication for GPU architectures // High Performance Embedded Architectures and Compilers, vol.5952, pp.111–125, 2010.
23. Баландин М.Ю., Шурина Э.П. Методы решения СЛАУ большой размерности. — Новосибирск: Изд-во НГТУ, 2000. — 70 с.

Богданов Павел Борисович. Младший научный сотрудник НИИСИ РАН. Окончил МГУ в 2003 году. Автор 13 печатных работ. Область научных интересов: высокопроизводительные вычисления, OpenCL, низкоуровневая оптимизация. E-mail: bogdanov@niisi.msk.ru

Сударева Ольга Юрьевна. Младший научный сотрудник НИИСИ РАН. Окончила МГУ в 2010 году. Автор 3 печатных работ, в том числе одной монографии. Область научных интересов: высокопроизводительные вычисления, GPGPU, OpenCL, цифровая обработка сигналов, БПФ, низкоуровневая оптимизация. E-mail: sudareva@niisi.msk.ru