УНИВЕРСИТЕТ ИТМО
НАУЧНО-ТЕХНИЧЕСКИЙ ВЕСТНИК ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ
январь–февраль 2015 Том 15 № 1 ISSN 2226-1494 http://ntv.ifmo.ru/
SCIENTIFIC AND TECHNICAL JOURNAL OF INFORMATION TECHNOLOGIES, MECHANICS AND OPTICS
January–February 2015 Vol. 15 No 1 ISSN 2226-1494 http://ntv.ifmo.ru/en

УДК 004.273:004.051

АЛГОРИТМ ВЫБОРА РАЦИОНАЛЬНОЙ ПРОЦЕССОРНОЙ АРХИТЕКТУРЫ

Н.А. Шаменков[a], А.М. Зыков[b], А.А. Карытко[b]

[a] Научно-исследовательский испытательный центр центрального научно-исследовательского института ВВКО Министерства обороны Российской Федерации, Москва, 129345, Российская Федерация
[b] Военно-космическая академия имени А.Ф. Можайского, Санкт-Петербург, 197198, Российская Федерация
Адрес для переписки: kurok134@yandex.ru

Информация о статье
Поступила в редакцию 03.10.14, принята к печати 26.12.14
doi: 10.17586/2226-1494-2015-15-1-94-100
Язык статьи – русский
Ссылка для цитирования: Шаменков Н.А. , Зыков А.М., Карытко А.А. Алгоритм выбора рациональной процессорной архитектуры // Научно-технический вестник информационных технологий, механики и оптики. 2015. Том 15. № 1. С. 94–100

Аннотация. Представлен алгоритм, позволяющий осуществить выбор процессорной архитектуры вычислительного ядра. Такая архитектура обеспечивает максимально возможный темп вычислительного процесса. Работа алгоритма основана на использовании метода скользящего окна, применяемого к фрагментам кода программ, исполнение которых занимает максимальный процент времени – «узким местам». Алгоритм обеспечивает расчет рационального числа арифметико-логических каналов вычислительного ядра процессора в зависимости от типа поддерживаемых им операций. На примере программного кода, реализующего алгоритм расчета тессеральных гармоник гравитационного поля Земли, произведен расчет рационального числа арифметико-логических каналов процессорной архитектуры. В примере учитывались арифметические операции целочисленного и вещественного сложения (вычитания), вещественного умножения, а также операции расчета значений логических предикатов. В соответствии с результатами расчета установлено, что для рассмотренного примера рациональный вариант процессорной архитектуры должен включать два арифметико-логических канала, способных выполнять указанные операции. Разработанный алгоритм целесообразно использовать при решении задач синтеза процессорных архитектур и вычислительных систем, создаваемых на их основе. Максимальный эффект использования результатов работы алгоритма достигается в процессе синтеза вычислительных систем, выполняющих задачи на основе единого математического аппарата.
Ключевые слова: процессорная архитектура, метод скользящего окна, вычислительный конвейер, арифметико-логический канал.

ALGORITHM OF RATIONAL PROCESSOR ARCHITECTURE

N.A. Shamenkov[a], A.M. Zykov[b], A.A. Karytko[b]

[a] Scientific Research Test Department of the Head Central Scientific Research Institute of Aerospace Defence of the Ministry of Defence of the Russian Federation, Moscow, 129345, Russian Federation
[b] Military Space Academy n.a. A.F. Mozhaisky, Saint Petersburg, 197198, Russian Federation
Corresponding author: kurok134@yandex.ru

Article info
Received 03.10.14, accepted 26.12.14
doi: 10.17586/2226-1494-2015-15-1-94-100
Article in Russian
Reference for citation: Shamenkov N.A., Zykov A.M., Karytko A.A. Algorithm of rational processor architecture. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2015, vol. 15, no. 1, pp. 94–100 (in Russian)

Abstract. The paper deals with an algorithm that makes it possible to decide on processor architecture for computational kernel. This architecture provides the maximum possible rate of the computational process. The algorithm is based on a sliding window method applied to bottlenecks - fragments of program code taking a maximum percentage of time for execution. The algorithm calculates a rational number of arithmetic-logic processor core computing channels depending on the type of supported operations. Calculation of the rational number for arithmetic and logical channels of processor architecture is performed on the code example that implements the algorithm for calculating the tesseral harmonics of the Earth gravitational field. Arithmetic operations of integer and real addition (subtraction), real multiplication, as well as the operations of calculating the values of logical predicates, were considered in the example. Calculation results revealed that for considered example, rational variant of processor architecture should include two arithmetic logic channels capable of performing these operations. The developed algorithm is feasible for application in solving the synthesis tasks for processor architectures and computing systems based on them. Maximum effect after using the algorithm results is achieved at the synthesis of computing systems that perform tasks on the basis of a consistent mathematical tool.
Keywords: processor architecture, sliding window method, instruction pipeline, arithmetic logic channel.

Введение

Исследованиям влияния архитектуры процессоров и многопроцессорных многомашинных вычислительных систем на оперативность решения задач сложными техническими системами посвящено значительное количество работ как в России [1–4], так и за рубежом [5, 6]. В соответствии с данными работами дальнейшее повышение оперативности решения задач достигается за счет наращивания вычислительных ресурсов гетерогенных распределенных систем, жесткого закрепления задач, решаемых ее элементами, а также введения в их состав специализированных, в том числе реконфигурируемых, вычислительных модулей.

Одним из критериев выбора процессорной архитектуры базового вычислительного элемента системы служит мера ее приспособленности к выполнению вычислительных задач, решаемых системой. Под мерой приспособленности понимается соответствие аппаратных ресурсов процессора (в данной работе – количество арифметико-логических каналов (АЛК) ядра процессора) требованиям, предъявляемым программно-математическим содержанием «узкого места» программы. Недостаток числа АЛК проявляется при условии доступности к исполнению на одном такте работы процессора нескольких арифметических операций, а также при большом числе условных ветвлений, не позволяющих осуществить верное аннотирование дальнейшего хода вычислительного процесса. При возникновении блокировок в вычислительных конвейерах недостаток АЛК приводит к увеличению времени ожидания операций, готовых к исполнению. Причинами возникновения блокировок являются:

1. изменение последовательности операций в комбинированных командах, поступающих на вход вычислительного конвейера;
2. меньшее по сравнению с числом АЛК количество портов записи и чтения регистрового файла ядра процессора;
3. выполнение АЛК многотактных арифметических операций;
4. отмена выполнения ветви вычислительного процесса в режиме спекулятивных вычислений, арифметические операции которой заполняют вычислительный конвейер в текущий момент времени.

С целью минимизации указанных явлений в современных процессорных архитектурах универсального назначения может быть предусмотрена избыточность АЛК, основанная на специфике задач, решаемых процессором, и частоте проявления различных видов арифметических операций. Вместе с тем излишнее количество АЛК приводит к росту числа портов регистрового файла процессора и общих шин байпаса, обеспечивающих передачу данных между каналами, что, напротив, увеличивает время передачи результатов операций с выходов арифметико-логических каналов и существенно усложняет проектирование систем на кристалле.

Таким образом, в условиях современного разнообразия процессорных архитектур [7] представляет интерес проведение исследований по выбору рационального варианта построения процессорной архитектуры, базового вычислительного элемента вычислительной системы универсального назначения. Проведение этих исследований основывается на анализе взаимозависимостей шагов алгоритма и операторов реализующей его программы [8, 9], выборе способа представления выявленных взаимозависимостей и инструмента для их анализа [10], декомпозиции шагов алгоритма, обладающих высокой вычислительной сложностью [11].

Цель настоящей работы заключается в повышении оперативности функционирования вычислительных систем за счет выбора рациональной процессорной архитектуры базового вычислительного элемента системы с помощью алгоритма, учитывающего особенности динамических профилей программных реализаций вычислительных задач.

Постановка задачи исследования

Исходными данными разработанного алгоритма являются:

- программная реализация алгоритма функционирования технической системы – A;
- множество арифметических операций, имеющих аппаратную реализацию в процессорных архитектурах, $O = \{+i, \times i, +f, \times f, /\}$, где символы i, f соответствуют операциям над целочисленными и вещественными данными;
- множество вариантов процессорных архитектур $H = \{h_1, h_2, \ldots, h_n\}$, каждый из элементов которого характеризуется количеством тактов процессора n, необходимых для выполнения соответствующих элементарных операций множества O.

Требуется определить рациональное число АЛК, поддерживающих конвейерные операции вещественного и целочисленного сложения, умножения, деления, а также вычисления значений логических предикатов.

Обнаружение «узких мест» программ – множества S, чье выполнение занимает наибольший процент времени от времени выполнения всей программы – выполняется средствами стандартных профилировщиков программ. К элементам сформированного множества S применяется разработанный алгоритм.

Алгоритм выбора рациональной процессорной архитектуры

Алгоритм состоит из следующих шагов.

Шаг 1. Макрооперации «узкого места» программы подвергаются процедуре последовательно-параллельной декомпозиции [11], заключающейся в рекурсивном вычитании и объединении множеств вида

$$X_i = \left(X_{i-1} \setminus X_i^{'}\right) \cup \left\{x_{n+i}\right\},$$

где X_i – исходный функционал, вычисляемый на заданном шаге алгоритма; $X_i^{'}$ – исключаемый из функционала X_i блок арифметических операций, соответствующих какому-либо элементу множества O; $\left\{x_{n+i}\right\}$ – множество ранее вырезанных блоков арифметических операций.

Результатом данного шага является формирование множества базовых арифметических операций «узкого места» – U. При этом специальные алгебраические и тригонометрические функции заменяются элементами множества O с условием сохранения тождественности результатов, получаемых в результате использования функционала оригинала и преобразованных выражений.

Шаг 2. Формируется матрица $\mathbf{M}$ инцидентности операндов арифметических операций множества U размерностью $k \times k$, где k – общее число операндов операторов U, при этом создаются временные операнды, содержащие результаты промежуточных операций, элемент матрицы $\mathbf{M}$ – $m(i,j)$ равен единице, если для вычисления значения операнда j требуется вычислить значение операнда i.

Шаг 3. Формируется множество $B = \left\{\langle o_1, r_1, i_1, j_1, t\rangle, \ldots, \langle o_n, r_n, i_n, j_n, t\rangle\right\}$, где n – общее количество арифметических операций $o_j \in O$, $j \in \overline{1,n}$, r – номер строки и столбца матрицы $\mathbf{M}$, ассоциированных с промежуточной переменной, содержащей результат операции o_j, i, j – номера строк и столбцов, ассоциированных с операндами операции o_j, t – номер такта процессора с момента начала исполнения «узкого места» программы.

Шаг 4. На основе матрицы $\mathbf{M}$, а также множества B формируется матрица расстояний $\mathbf{M}_p$ [12] размерностью $k \times k$, в которой единичные элементы $m(i,r)$ матрицы $\mathbf{M}$ заменяются числом тактов процессора $n\left(o_j \middle| h_i\right)$ с архитектурой $h_i \in H$, необходимых для выполнения операции o_j, формирующей значение переменной i.

Шаг 5. Применяя к матрице расстояний $\mathbf{M}_p$ алгоритм поиска в глубину, в поле t элементов множества B записываются значения $\max\limits_{l \in L}\left(l(r)\right)$, где L – множество всевозможных трасс до операнда r.

Шаг 6. Для каждого элемента множества O выполняется подсчет числа $n(t)$ элементов множества B, обладающих одинаковым значением поля t. Таким образом, число АЛК $N_{\max}(o)$, поддерживающих выполнение арифметической операции o и обеспечивающее минимальное время выполнения «узкого места» программы, вычисляется с помощью следующего выражения:

$$N_{\max}(o) = \max\left(n_l(t)\right), l = \overline{1,S},$$

где S – число групп элементов множества B с одинаковыми значениями поля t, характеризующих выполнение арифметической операции o.

Шаг 7. На основе элементов множества B для каждого типа операций o формируются булевы матрицы вызова операций $\mathbf{M}_{bo_i}$ размерностью $d \times n$, где d – максимальное значение поля t, среди элементов множества B, характеризующих выполнение операции o. Элемент $m_{bo}(i,j)$ равен единице, если j-я операция o начинает выполнение на i-ом такте работы процессора.

Шаг 8. Методом скользящего окна осуществляется поиск разреженных областей матрицы $\mathbf{M}_{bo_i}$ (последовательностей нулевых вектор-строк) размером $k \times n$ и более, где k – глубина конвейера, выполняющего арифметическую операцию o. Найденные области удаляются из $\mathbf{M}_{bo}$.

Шаг 9. Методом скользящего окна производится расчет коэффициента загрузки вычислительного конвейера для заданного типа арифметической операции согласно формуле:

$$K_o = \frac{1}{(d-k)} \sum_{i=1}^{d-k} \sum_{j=i}^{k+i} v_i\left(j \middle| z_i\right),$$

где $v_i\left(j \middle| z\right)$ – количество конвейеров, попавших в текущее окно $z_i = k \times n$ и имеющих загрузку j операций.

Шаг 10. Методом скользящего окна производится расчет коэффициента пересечения $K_{\cap}^{os}$ арифметических операций o и s, допускающих аппаратную реализацию в рамках одного арифметического устройства:

$$K_{\cap}^{os} = 1 - \frac{N_{\varnothing}}{N_{\cap}},$$

где $N_{\varnothing}$ – число окон, содержащих такты начала выполнения только операций o или операций o совместно с операциями, отличными от операции s; $N_{\cap}$ – общее число окон, содержащих такты начала выполнения операции o.

Шаг 11. Рассчитывается рациональное число АЛК n_{alu}, выполняющих базовую операцию o, с помощью выражения

$$n_{alu}^{o} = \left\lceil K_o + \sum_{s \in O, s \neq o} K_{\cap}^{os} \cdot K_s \right\rceil,$$

где квадратные скобки соответствуют операции округления вещественных чисел в сторону ближайшего наибольшего целого числа.

На этом работа алгоритма заканчивается.

Результаты экспериментов

Описанный алгоритм применялся для анализа программ, используемых для реализации прогноза движения космических объектов в околоземном космическом пространстве. Результаты профилирования программ свидетельствуют о том, что до 40% общего времени их исполнения выполняется процедура расчета коэффициентов секторальных, тессеральных, а также зональных гармоник [13] гравитационного поля Земли. Фрагмент кода на языке C, представленный ниже, идентичен исходному фрагменту программы с точки зрения его арифметической сложности и порядка следования операторов:

```
1.      while(q<=n){
2.              i=q+ct;
3.              i2=q+ct;
4.              w=(ct*q(i)+ct)*e;
5.              w2=(ct*q(i2)+ct)*z*r;
6.              while(a<=n) {
7.              b1=m*k;
8.              b2=f*k;
9.              c1+=b1*y*(a+ct);
10.             c2+=b2*y*(a+ct);
11.             s+=b1*y;
12.             s2+=b2*y;
13.             if(a>q+ct)
14.             l=((ct*a-ct)*u*w2-(a+q)*w)/(a-q-ct);
15.             if(a!=q) {
16.                     z+=b1*u;
17.                     z2+=b2*u;}      }
18.             if(q!=t) {
19.                     v+=q*(s*p+s2*p2);
20.                     v2+=q*(s2*p2-s*p);      }
21.             zz0+=c1*p+c2*p2;
22.             zz1+=q*(s*p+s2*p2);
23.             zz2+=z*p+z2*p2;         }
```

В представленном фрагменте кода обозначение ct соответствует константе. Ярусно-параллельная форма приведенного фрагмента кода представлена на рисунке. Вершины графа имеют обозначение вида o,t, здесь o – тип арифметической операции, t – операнд, используемый для сохранения результата операции o, значение t, имеющее обозначение «…», характеризует временную переменную, используемую для сохранения промежуточных результатов, прочие обозначения – «*q*», «*n*», «*a*», «*b*», «*c*», «*i*», «*t*», «*l*», «*z*», «*w*», «*v*» – соответствуют переменным рассматриваемого фрагмента кода.

В треугольниках рядом с вершинами указаны номера тактов, на которых возможно выполнение операторов. Дуги представленного графа имеют числовые значения, характеризующие количество тактов процессорной архитектуры, необходимых для выполнения оператора-источника дуги. Указанные значения сформированы на основе анализа соответствующих характеристик процессорных архитектур x86 и e3s (архитектура «Эльбрус»).

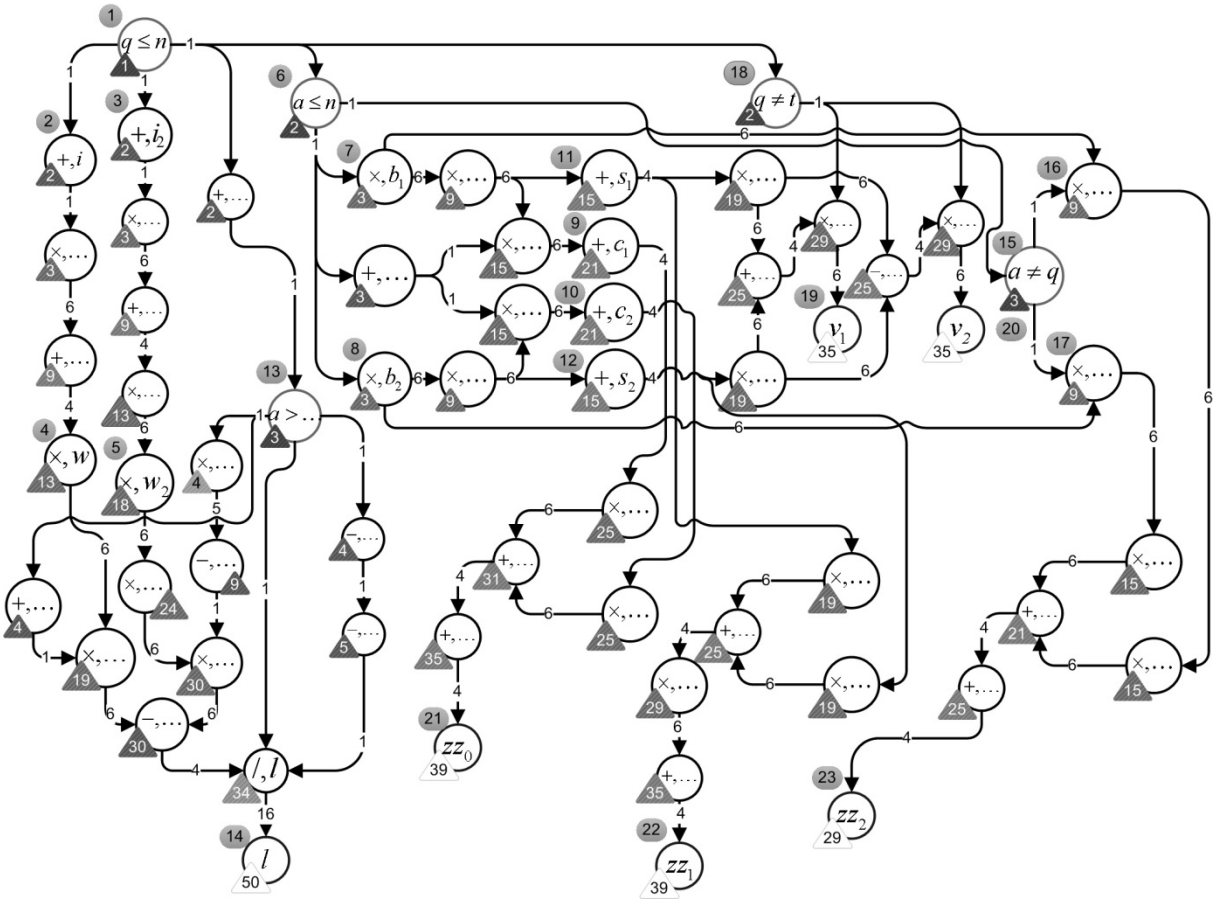

Рисунок. Ярусно-параллельная форма «узкого места» программы расчета коэффициентов тессеральных, зональных и секторальных гармоник

Расчет числа тактов, необходимых процессорной архитектуре x86, для выполнения рассматриваемых типов арифметических операций, производился с использованием ассемблерной инструкции rdtsc. Арифметическая операция размещалась в теле цикла, состоящего из k итераций. С использованием инструкции rdtsc производился замер числа тактов процессора n_1, n_2, выполненных с момента последнего сброса процессора на входе в тело цикла, реализующего процесс тестирования, и выходе из него. При этом учитывалось, что результат, возвращаемый инструкцией rdtsc, содержит по четыре дополнительных такта, затрачиваемых на чтение исходных данных и запись результата каждой операции в регистровый файл процессора, а также декодирование инструкции и доступ к памяти. Оценка числа тактов $n(o)$ процессорной архитектуры x86, расходуемых на выполнение операции o, осуществлялась согласно выражению

$$n_i = \left(n_2^i - n_1^i\right) \cdot k^{-1},\ i \in 1{,}10,$$

$$n(o) = \min_{n_i \in N}\left(n_i\right) - 4,$$

где N – множество результатов наблюдений числа тактов процессора, затрачиваемых на выполнение операции o; k – количество итераций, производимых в теле цикла, для каждой арифметической операции o.

Всего осуществлялось десять процедур тестирования для каждого значения k, принадлежащего множеству значений $\left\{10^2, 10^3, 10^4, 10^5\right\}$. Итоговые значения для процессорной архитектуры x86 составили: для операций целочисленного сложения – 1,4, вещественных операций сложения – 4,8, умножения – 7,7, деления – 18,2 такта на одну операцию. Наличие десятичного знака после запятой и возможное увеличение числа тактов, требуемых для выполнения операций, объясняются отсутствием учета в используемой методике возникающих в ходе выполнения тестовой программы обращений к кэш-памяти процессора и влияния на ход вычислительного процесса работы операционной системы и прерываний, генерируемых устройствами вычислительной системы.

Относительно процессорной архитектуры e3s в соответствии с работой [14], а также общим описанием устройства данной микроархитектуры известно, что целочисленное сложение занимает один такт работы процессора, вещественные операции сложения – до 5 тактов, умножение – от 4 до 6 тактов, деление – от 11 до 20 тактов. Вычисление значений логических предикатов во всех современных процессор-

ных архитектурах реализовано за один такт работы процессора. На основе указанных данных в рассматриваемом примере полагалось, что для выполнения операции вычисления значения логического предиката и целочисленного сложения необходим один такт, для вычисления результата вещественного сложения и целочисленного умножения – 4 такта, результата вещественного умножения – 6 тактов, результата операции деления – 16 тактов.

В соответствии с исходными данными получены следующие значения коэффициентов загрузки вычислительных конвейеров:

- вещественного умножения – 1,61;
- вещественного сложения – 1,27;
- целочисленного сложения – 1,78;
- расчета значений логических предикатов – 1,82.

Операции деления и целочисленного умножения участия в расчетах не принимались во внимание, поскольку они имеют низкий процент использования в рассматриваемом участке программы. Однако, учитывая количество шагов, необходимых для реализации операции деления, совмещение данной арифметической операции с прочими рассмотренными операциями в рамках одного АЛК является нецелесообразным. В рассмотренном примере наиболее часто используемая операция вещественного умножения обладает следующими коэффициентами пересечения с прочими арифметическими операциями:

- вещественного сложения – 0,8;
- целочисленного сложения – 0,086;
- расчета значений логических предикатов – 0,17.

Таким образом, рациональный вариант процессорной архитектуры для выполнения рассматриваемого фрагмента кода состоит из двух АЛК, способных выполнять операции целочисленного и вещественного сложения (вычитания), умножения, вычисления значений предикатов и одного АЛК, обеспечивающего выполнение операции деления. Добавление четвертого АЛК позволяет получить незначительный выигрыш оперативности решения задачи, максимальное значение которого не превышает четырех тактов работы процессора (случай, когда команда вещественного умножения готова к выполнению и АЛК заняты выполнением вещественных операций деления и сложения, а также целочисленного умножения). Наличие числа АЛК, превышающего его рациональное значение, способствует дальнейшему сокращению времени исполнения программы в связи с возможностью обеспечения конвейеризации циклов [15, 16] в случаях отсутствия зависимостей по данным между несколькими последовательными итерациями. Вместе с тем увеличение числа АЛК, поддерживающих выполнение операции o свыше максимального числа таковых операций, способных начать выполнение на одном такте работы процессора, является бесполезным.

Адекватность разработанного алгоритма выбора рациональной процессорной архитектуры подтверждается результатами, полученными на пакете тестов обработки радиолокационной информации для отечественного процессора Эльбрус-2С+ и процессора Core 2 Duo.

Заключение

Описанный алгоритм позволяет формулировать требования к количеству арифметико-логических каналов вычислительных ядер процессорной архитектуры. Результаты работы алгоритма целесообразно использовать на этапе синтеза процессорных архитектур, предназначенных для решения задач, использующих единый математический аппарат. Подобный подход обеспечивает рациональное использование полезной площади кристаллов процессоров и позволяет достичь максимально возможный темп вычислительного процесса в рамках используемой процессорной архитектуры.

При проектировании многопроцессорных многомашинных распределенных вычислительных систем результаты работы алгоритма позволяют максимально адаптировать структуру вычислительной системы к множеству решаемых задач за счет их жесткого закрепления за конкретными узлами вычислительной сети на основе критерия совпадения рационального числа арифметико-логических каналов узла и требуемого числа каналов для решения задачи.

Вместе с тем представляет несомненный интерес проведение дальнейших исследований, посвященных оптимизации системы команд процессорных архитектур и процесса обмена результатами арифметических операций между арифметико-логическими каналами процессора, организации асинхронного доступа к регистровому файлу процессора, количеству блоков регистрового файла и распределению регистров по назначению, входящих в их состав.

Referencesa

1. Trakhtengerts E.A. The effect of architecture and structure of multiprocessor computers on programming languages and translation methods. *Avtomatika i Telemekhanika*, 1986, no. 3, pp. 5–47.
2. Rumyantsev A.S. Metod otobrazheniya zadach na krupnogranulyarnye rekonfiguriruemye vychislitel'nye sistemy [Tasks mapping method for coarse grain reconfigurable systems]. *Scientific and Technical Journal of Information Technologies, Mechanics and Optics*, 2014, no. 1 (89), pp. 76–81.

3. Bogdanov A.Yu., Perekatov V.I., Feldman V.M. Kommunikatsionnye interfeisy mezhmashinnykh svyazei vychislitel'nykh sredstv semeistva «El'brus» [Communication interfaces of Elbrus computer means untermachine liasons]. *Voprosy Radioelektroniki*, 2013, vol. 4, no. 3, pp. 72–83.
4. Myskin A.V., Torgashev V.A., Tsaryov I.V. Protsessory s dinamicheskoi arkhitekturoi na osnove skhem gibkoi logiki [Processors with dynamic architecture based on flexible logic devices]. *Trudy SPIIRAN*, 2002, vol. 1, no. 1, pp. 113–128.
5. Bacon D.F., Grahem S.L., Sharp O.J. Compiler transformations for high-perfomance computing. *ACM Computing Surveys*, 1994, vol. 26, no. 4, pp. 345–420. doi: 10.1145/197405.197406
6. Molka D., Hackenberg D., Schone R., Muller M.S. Memory performance and cache coherency effects on an Intel Nehalem multiprocessor system. *Proc. 18th Int. Conf. on Parallel Architectures and Compilation Techniques*. Dresden, Germany, 2009, pp. 261–270. doi: 10.1109/PACT.2009.22
7. Stegajlov V.V., Norman G.E. Problemy razvitiya superkomp'yuternoi otrasli v Rossii: vzglyad pol'zovatelya vysokoproizvoditel'nykh sistem [Challenges to the supercomputer development in Russia: a HPC user perspective]. *Programmnye Sistemy: Teoriya i Prilozheniya*, 2014, vol. 5, no. 1–1 (19), pp. 111–152.
8. Toporkov V.V. *Modeli Raspredelennykh Vychislenii* [Models of Distributed Computing]. Moscow, Fizmatlit Publ., 2004, 320 p.
9. Tel G. *Introduction to Distributed Algorithms*. 2nd ed. Cambridge University Press, 2000, 612 p.
10. Karpov V.E., Lobanov A.I. *Chislennye Metody, Algoritmy i Programmy. Vvedenie v Rasparallelivanie* [Numerical Methods, Algorithms and Programs. Introduction to Parallelization]. Moscow, Fizmatkniga Publ., 2014, 196 p.
11. Cheburakhin I.F. *Sintez Diskretnykh Upravlyayushchikh Sistem i Matematicheskoe Modelirovanie* [Synthesis of Discrete Control Systems and Mathematical Modeling]. Moscow, Fizmatlit Publ., 2004, 247 p.
12. Reznikov B.A. *Metodologiya Sistemnykh Iissledovanii. Modelirovanie Slozhnykh Sistem* [The Methodology of Systems Research. Modeling of Complex Systems]. Leningrad, MO USSR Publ., 1990, 522 p.
13. Roy A.E. *Orbital Motion*. Bristol, UK, Adam Hilger ltd, 1978.
14. Kim A.K., Perekatov V.I., Ermakov S.G. *Mikroprotsessory i Vychislitel'nye Kompleksy Semeistva «El'brus»* [Microprocessors and Computer Systems of the Family "Elbrus"]. St. Petersburg, Piter Publ., 2013, 272 p.
15. Gergel' V.P., Meerov I.B., Bastrakov S.I. *Vvedenie v printsipy funktsionirovaniya i primeneniya sovremennykh mul'tiyadernykh arkhitektur (na primere Intel Xeon Phi)* [Introduction to the principles of operation and application of modern multicore architectures, for example, Intel Xeon Phi]. Available at: http://www.intuit.ru/studies/courses/10611/1095/info (accessed 30.09.2014).
16. Rau B.R. Iterative modulo scheduling: an algorithm for software pipelining loops. *Professional Engineering*, 1994, vol.7, no. 21, pp. 63–74.

Шаменков Николай Александрович – кандидат технических наук, начальник отдела, Научно-исследовательский испытательный центр центрального научно-исследовательского института ВВКО Министерства обороны Российской Федерации, Москва, 129345, Российская Федерация, Shna810516@mail.ru

Зыков Александр Михайлович – кандидат технических наук, доцент, доцент, Военно-космическая академия имени А.Ф. Можайского, Санкт-Петербург, 197198, Российская Федерация, Amz05@rambler.ru

Карытко Анатолий Александрович – аспирант, адъюнкт, Военно-космическая академия имени А.Ф. Можайского, Санкт-Петербург, 197198, Российская Федерация, kurok134@yandex.ru

Nikolai A. Shamenkov – PhD, Department head, Scientific Research Test Department of the Head Central Scientific Research Institute of Aerospace Defence of the Ministry of Defence of the Russian Federation, Moscow, 129345, Russian Federation, Shna810516@mail.ru

Alexander M. Zykov – PhD, Associate professor, Associate professor, Military Space Academy n.a. A.F. Mozhaisky, Saint Petersburg, 197198, Russian Federation, Amz05@rambler.ru

Anatoliy A. Karytko – postgraduate, Graduated in a military academy, Military Space Academy n.a. A.F. Mozhaisky, Saint Petersburg, 197198, Russian Federation, kurok134@yandex.ru