СИСТЕМЫ И СРЕДСТВА ИНФОРМАТИКИ

**Научный журнал Российской академии наук
(издается под руководством Отделения нанотехнологий
и информационных технологий РАН)**

Издается с 1989 года
Журнал выходит два раза в год

**Учредители:
Российская академия наук
Институт проблем информатики Российской академии наук**

РЕДАКЦИОННЫЙ СОВЕТ

РЕДАКЦИОННАЯ КОЛЛЕГИЯ

Редакция

Журнал «Системы и средства информатики»
включен в «Перечень российских рецензируемых журналов,
в которых должны быть опубликованы основные научные результаты диссертаций
на соискание ученых степеней доктора и кандидата наук», составленный ВАК

СИСТЕМЫ И СРЕДСТВА ИНФОРМАТИКИ

Том 23 № 2 Год 2013

СОДЕРЖАНИЕ

Скрытые каналы, порожденные метками, в дейтаграммах
А. А. Грушо, Н. А. Грушо, Е. Е. Тимонина **6**

Covert channels generated by tags in datagrams
A. A. Grusho, N. A. Grusho, and E. E. Timonina **19**

Топологическая модель изображений отпечатков пальцев
В. Ю. Гудков, О. С. Ушмаев **22**

Topological model of fingerprint image
V. Yu. Gudkov and O. S. Ushmaev **33**

Метод параллельных цепей для распознавания изображений отпечатков пальцев
В. Ю. Гудков **35**

Method of parallel circuits for fingerprint image recognition
V. Yu Gudkov **48**

Классификация людей по изображению лица на основе сравнительных признаков внешности
В. С. Конушин, Т. М. Лукина, А. И. Кухаренко, А. С. Конушин **50**

Person classification upon face image based on simile classifiers
V. S. Konushin, T. M. Lukina, A. I. Kuharenko, and A. S. Konushin **59**

Возрастная классификация людей по изображению лица на основе метода ранжирования и локальных бинарных шаблонов
А. В. Рыбинцев, Т. М. Лукина, В. С. Конушин, А. С. Конушин **62**

Age estimation upon face image based on local binary patterns and a ranking approach
A. V. Rybintsev, T. M. Lukina, V. S. Konushin, and A. S. Konushin **72**

СИСТЕМЫ И СРЕДСТВА ИНФОРМАТИКИ

Том 23 № 2 Год 2013

СОДЕРЖАНИЕ

Поиск соответствий между ключевыми точками изображений радужных оболочек глаз с помощью метода проекционной фазовой корреляции
Е. А. Павельева **74**

The search for matches between the iris key points using Hermite projection phase-only correlation method
E. A. Pavelyeva **86**

Дискриминантный анализ для классификации и прогнозирования результатов лечения
М. А. Драницына, Т. В. Захарова **89**

Discriminant analysis for classification and forecasting outcomes of the treatment
M. A. Dranitsyna and T. V. Zakharova **95**

Исследование сложной задачи диагностики артериальной гипертензии в методологии искусственных гетерогенных систем
И. А. Кириков, А. В. Колесников, С. Б. Румовская **96**

Study of the complex problem of arterial hypertension diagnostics in the methodology of artificial heterogeneous systems
I. A. Kirikov, A. V. Kolesnikov, and S. B. Rumovskaya **113**

Агрегирование геоконцепций при генерализации карты с учетом логической согласованности и семантической точности
С. К. Дулин, Н. Г. Дулина, П. В. Ермаков **115**

Aggregation geoconcepts for generalization maps, appropriate logical consistency and semantic accuracy
S. K. Dulin, N. G. Dulina, and P. V. Ermakov **131**

Теоретические аспекты разработки методологии программирования рекуррентной архитектуры
Д. В. Хилько, Ю. А. Степченков **133**

СИСТЕМЫ И СРЕДСТВА ИНФОРМАТИКИ

Том 23 № 2 Год 2013

СОДЕРЖАНИЕ

Theoretical aspects of programming methodology development for recurrent architecture
D. Khilko and Yu. Stepchenkov **151**

Методические вопросы формирования системы технического обеспечения информационно-телекоммуникационных сетей
А. А. Зацаринный, Н. Г. Буроменский, А. И. Гаранин **154**

Metogological questions of formation of the system technical support of information-telecommunication networks
A. A. Zatsarinnyy, N. G. Buromienskii, and A. I. Garanin **166**

К вопросу о сравнительной оценке эффективности ситуационных центров
А. А. Зацаринный, Ю. С. Ионенков, А. П. Шабанов **170**

On a comparative evaluation of situational centers efficiency
A. A. Zatsarinnyy, Y. S. Ionenkov, and A. P. Shabanov **184**

Формирование системы целей для ситуационного управления
А. П. Сучков **187**

Formation of the goals system for situational control
A. P. Suchkov **196**

Универсальный паттерн организации ситуационных центров
А. В. Колесников, А. А. Меркулов **198**

Universal pattern of organizations for the situational centers
A. V. Kolesnikov and A. A. Merkulov **219**

СИСТЕМЫ И СРЕДСТВА ИНФОРМАТИКИ, 2013. Т. 23. № 2. С. 133–153

ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ РАЗРАБОТКИ МЕТОДОЛОГИИ ПРОГРАММИРОВАНИЯ РЕКУРРЕНТНОЙ АРХИТЕКТУРЫ*

Д. В. Хилько[1], *Ю. А. Степченков*[2]

Аннотация: Статья посвящена новой рекуррентно-потоковой парадигме вычислений и методологии решения и программирования задач в среде разрабатываемого вычислительного устройства, архитектура которого реализует идеи и принципы описываемой парадигмы. Рассмотрена реализация новой парадигмы в многоядерной потоковой рекуррентной архитектуре (МПРА). Доказана сходимость рекуррентной организации вычислительного процесса с использованием понятий и теорем теории рекурсивных функций. Описана проблема разработки программного обеспечения (ПО), способного функционировать в среде рекуррентной архитектуры. Предложена специализированная рекуррентно-потоковая методология программирования, охватывающая все этапы проектирования ПО. Продемонстрировано применение методологии для решения задачи распознавания изолированных слов (РИС) в среде новой архитектуры. Проведена также поэтапная реализация одного из алгоритмов данной задачи — полосовой фильтрации — в соответствии с временно́й структурой деятельности методологии.

Ключевые слова: парадигма вычислений; методология программирования; рекурсивность; потоковая архитектура

DOI: 10.14357/08696527130210

1 Введение

Развитие идей параллельных вычислений в 1980-х гг. привело к исследованию нового класса архитектур вычислительных систем (ВС) — потоковых архитектур [1, 2]. В архитектурах этого класса поток данных имеет приоритет над потоком команд и является инициатором вычислений. В ряде стран ведутся исследования и разработки систем потоковой архитектуры, но, несмотря на видимые преимущества, такие как отсутствие «узких мест», характерных для фон-неймановской архитектуры [3], и исключение вероятности обработки неподготовленных данных, ряд проблем как технического, так и алгоритмического характера препятствует массовому применению потоковых архитектур.

*Работа выполнена при частичной финансовой поддержке по программам фундаментальных исследований ОНИТ РАН за 2013 г. (проект 1.5) и Президиума РАН (проект 16).

[1]Институт проблем информатики Российской академии наук, dhilko@yandex.ru

[2]Институт проблем информатики Российской академии наук, YStepchenkov@ipiran.ru

Разнообразие задач, решаемых в настоящее время при помощи ВС, очень велико, причем многие из них в том или ином виде содержат рекуррентные вычисления и рекурсию. Это неразрывно связано с понятиями алгоритма и функции, вычисляемой с помощью алгоритма, — основными в рамках теории рекурсивных функций [4]. В Институте проблем информатики РАН ведутся работы по созданию нетрадиционной рекуррентной архитектуры, предназначенной для реализации параллельных вычислений ограниченной размерности в области сигнальной обработки.

В рамках данной статьи для именования архитектуры авторы используют название МПРА, принципы работы которой базируются на новой рекуррентно-потоковой вычислительной парадигме [5]. Для экспериментальной апробации предлагаемой архитектуры разрабатывается рекуррентный обработчик сигналов (РОС), исполняемый в гибридном, двухуровневом варианте — рекуррентном операционном устройстве (РОУ) [6] с ведущим фон-неймановским процессором на управляющем (верхнем) уровне (УУ) и рядом потоковых процессоров на операционном (нижнем) уровне (ОУ).

Основная особенность разрабатываемой архитектуры — способ представления потоков данных и команд, объединенных в единый поток, который рекуррентно сворачивается при подготовке и разворачивается в ходе исполнения. Данное представление, по мнению разработчиков, позволяет эффективно реализовать рекуррентные и рекурсивные вычисления, а также решить другие проблемы, свойственные потоковой и параллельной архитектурам.

Для достижения максимальной производительности разрабатываемой архитектуры необходимо разработать соответствующее ПО, оптимизированное для выполнения в ее среде. Специфические особенности архитектуры, не свойственные другим классам, делают невозможным применение в полном объеме известных методов и технологий программирования. В связи с этим возникла необходимость разработки новой методологической базы для эффективного программирования в среде МПРА.

Данная работа посвящена решению такой задачи, а также доказательству сходимости рекуррентного вычислительного процесса.

2 Доказательство сходимости рекуррентной организации вычислительного процесса

В рамках работ по созданию МПРА необходимо доказать, что новая вычислительная парадигма как минимум будет не менее эффективной, чем уже существующие. Первой и наиболее важной задачей является доказательство того, что рекуррентный вычислительный процесс сойдется и приведет к получению ожидаемых результатов (остановится).

В ходе доказательства будут использованы термины и теоремы теории рекурсивных функций [4]: теорема о нумерации, определение примитивно рекурсивной функции, определение частично рекурсивной функции, s-n-m-теорема, тезис Черча. Фактически необходимо показать, что функциональные преобразования над самоопределяющимися данными можно описать частично рекурсивной функцией нескольких переменных. Необходимо также найти метод построения функции такого рода и обосновать тот факт, что в результате построения будет получена функция, которую можно привести к универсальной частично рекурсивной функции одной переменной (из теоремы о нумерации).

Чтобы корректно построить функцию, описывающую рекуррентную организацию вычислительного процесса в МПРА, необходимо более подробно описать механизмы преобразований (модификаций) значений функциональных полей (тегов) обрабатываемых данных. Суть преобразований заключается в следующем: самоопределяющиеся данные несут в себе служебную информацию для их обработки в виде набора функциональных полей, часть из которых (в количестве пяти) подвергается преобразованиям. Очевидно, что принимаемые полями значения принадлежат множеству натуральных чисел N. Преобразованные значения функциональных полей могут (но не обязательно) определять путь формирования графа вычислительного процесса (так называемая графодинамика [7]). Последовательность новых значений функциональных полей задает рекуррентную цепочку. Таким образом, функциональные преобразования можно представить в виде функции пяти переменных (по числу функциональных полей) $f(x_1, x_2, x_3, x_4, x_5)$, где x_i соответствует функциональному полю, подвергающемуся преобразованию. Необходимо показать, что можно построить такую функцию и она будет частично рекурсивной.

Воспользуемся теоремой о нумерации, согласно которой достаточно показать, что можно построить функцию одной переменной, удовлетворяющей заданным требованиям. Заметим также, что каждое x_i есть функция от номера шага функциональных преобразований (рекуррентной развертки) t. Указанный номер шага отсчитывается с момента начала построения рекуррентной цепочки и не обязательно совпадает с текущим шагом вычислительного процесса. С учетом данных положений задачу можно переформулировать следующим образом: *доказать, что функциональные преобразования поля x_i есть частично рекурсивная функция от номера шага вычислительного процесса*: $t \in [0, n)$, *где n — глубина функциональных преобразований; на каждом шаге эта функция принимает значения, заданные рекуррентной цепочкой.*

Доказательство сходимости. В качестве аргумента искомой функции следует выбрать номер текущего шага рекуррентной развертки t. Тогда искомая функция примет вид функции от значения предыдущего шага рекуррентной развертки и номера текущего шага. Обозначим ее

$$f(t) = f'(f(t-1), t)\,.$$

Обозначим также

$$f(0) = f_0\,;\;\; f(1) = f_1\,;\;\; \ldots;\;\; f(n-1) = f_{n-1}\,. \tag{1}$$

Искомую рекуррентную цепочку натуральных чисел f в соответствии с формулой (1) можно представить в виде вектора

$$f = \{f_0, f_1, \ldots, f_{n-1}\}\,. \tag{2}$$

Обозначим через $\{F\}$ упорядоченное пронумерованное множество мощностью l (n может быть больше l) всех допустимых значений функционального поля, зафиксированных в рамках конкретной реализации архитектуры. Тогда задачу можно интерпретировать следующим образом: найти последовательность номеров значений функционального поля, которая образует требуемую цепочку f.

На множестве $\{F\}$ можно определить функцию выбора $s(q)$, которая для заданного номера q возвращает соответствующее значение. По определению примитивной рекурсии функция выбора примитивно рекурсивна, следовательно, и частично рекурсивна. В силу предложенной интерпретации будем полагать, что формула (2) задает искомую последовательность номеров.

Другими словами, необходимо построить функцию $s(f(t))$, которая строит цепочку f. Причем эта функция будет частично рекурсивной, в соответствии с предыдущим абзацем, если будет частично рекурсивной функция $f(t)$.

Тогда искомая функция $f(t)$ могла бы иметь следующий вид:

$$f(t) = f(t-1) + g(k,t)\,;$$

$$g(k,t) = \begin{cases} f_t - k\,, & \text{если } k = f_{t-1}\,; \\ g(k-1,t)\,, & \text{если } k \neq f_{t-1}\,; \end{cases}$$

$$g(0,t) = 0\,; \quad f(0) = f_0\,.$$

Покажем, что данная функция является частично рекурсивной. Функция $g(k,t)$ — частичная, так как определена только на заданных отрезках $[1,l]$ и $[0,n)$. По определению примитивной рекурсии функции $g(k,t) = f_t - k$ и $g(k,t) = g(k-1,t)$ примитивно рекурсивны; кроме того, функция сравнения также примитивно рекурсивна. Класс общерекурсивных функций включает в себя класс примитивно рекурсивных функций, и, следовательно, по тезису Черча и определению функция $g(k,t)$ является частично рекурсивной. Аналогично $f(t)$, которая является суммой частично рекурсивных функций, по тезису Черча частично рекурсивна.

Таким образом, нашлась такая частично рекурсивная функция от номера шага рекуррентной развертки $s(f(t))$, которая строит корректную цепочку рекуррентных преобразований. По s-n-m-теореме у этой функции существует Геделев

номер (т. е. универсальная частично рекурсивная функция вычисления требуемой цепочки). По построению эта функция сходится к искомому вектору (2).

Итак, получена частично рекурсивная функция одной переменной. Воспользовавшись первоначальными выкладками и теоремой о нумерации, получим частично рекурсивную функцию пяти переменных, которая строит заданную цепочку рекуррентных преобразований. Кроме того, существует метод ее построения. Полученная функция сходится по построению, что и требовалось доказать.

Таким образом, удалось доказать, что рекуррентная организация вычислительного процесса сходится. Следует также отметить, что в рамках работы над МПРА создано специализированное программное средство РЕКУРРЕНТ [8], предназначенное для поиска функций и построения рекуррентных цепочек, а также проверки на достижимость узлов цепочки в этих функциях.

3 Описание рекуррентно-потоковой методологии программирования

Уникальность рекуррентного вычислительного процесса МПРА означает, что ни одна из существующих методологий программирования (структурная, функциональная, объектно-ориентированная и др.) в полной мере не удовлетворяет потребности ее разработчиков. В связи с этим возникла потребность разработки новой методологии, которую авторы МПРА назвали *рекуррентно-потоковой методологией программирования*.

Рассмотрим каждую из составляющих структуры разрабатываемой методологии.

Основаниями данной методологии являются *информатика* и *системный анализ*. В самом деле, информатика включает в себя такой аспект, как разработка языков программирования, что является одной из основных задач в рамках разработки МПРА. В то же время системный анализ используется для определения параметров отдельных компонентов архитектуры, на основе которых разрабатывается специальное ПО, предназначенное для моделирования, проведения экспериментов и отладки.

Принцип рекуррентно-потоковой методологии программирования заключается в представлении исходной задачи в виде совокупности подпрограмм специального типа, называемых *капсулами*. Каждая капсула реализует один или несколько алгоритмов решаемой задачи и представляет собой набор самоопределяющихся данных. Данные такого типа содержат также и всю необходимую управляющую информацию для их обработки. Данное представление программ названо разработчиками *капсульным стилем программирования*.

Особенности капсульного программирования: представление программы в виде самопорождающейся последовательности динамических графов (этот про-

цесс порождения называется рекуррентной разверткой); программы, написанные в капсульном стиле, являются также и потоковыми.

Условием начала вычислительного процесса, помимо управляющего сигнала «пуск», является событие появления входных данных в символьных капсулах. Другими словами, для капсул переопределено «правило срабатывания», которое свойственно большинству потоковых архитектур и программ.

Субъектом, объектом и предметом деятельности в рамках настоящей методологии являются соответственно *программист* — разработчик ПО для МПРА, *капсула* — программа, функционирующая в среде МПРА, и *процесс разработки ПО*.

Важным разделом методологии является описание *временной структуры деятельности*, которую можно представить в виде обобщенной методики программирования в среде МПРА.

3.1 Этапы рекуррентно-потоковой методологии программирования

Этап I. Провести анализ задачи и определить, имеется ли необходимость производить сложные и ресурсоемкие вычисления и какую долю (приблизительно) в решении задачи они составляют.

Этап II. Если имеют место сложные вычислительные задачи и их доля достаточно велика, то декомпозировать исходную задачу на ряд подзадач по принципу:

(1) множество подзадач управления вычислительным процессом;

(2) множество подзадач низкой вычислительной сложности;

(3) множество подзадач высокой вычислительной сложности.

Классы задач 1 и 2 необходимо решать при помощи верхнего УУ архитектуры, третий класс — при помощи нижнего ОУ.

На текущем этапе разработки в качестве основных критериев оценки сложности задачи (подзадачи) были приняты следующие:

– большой объем входного набора (потока) данных (например, цифровые фильтры, свертки и др.);

– скрытый или явный параллелизм вычислительного графа задачи степени 3 или более;

– большой объем потока промежуточных данных;

– большой объем выходного набора (потока) данных.

Этап III. Осуществить реализацию всех подзадач (как для УУ, так и для ОУ). Результатом этого этапа будет управляющая программа, исполняемая средствами УУ, и набор специализированных программ, называемых капсулами

и решающих подзадачи класса 3. Управляющая программа осуществляет подготовку данных для капсул, обработку результатов их вычисления, а также решает подзадачи классов 1 и 2.

Этап IV. Выполнить комплексную отладку набора капсул и управляющей программы.

Ключевыми этапами данной методологии можно считать этапы II и III: первый определяет результирующую декомпозицию задачи, а второй содержит в себе квинтэссенцию парадигмы капсульного программирования.

Рассмотрим этап III более подробно. Этот этап был назван этапом капсульного программирования.

3.2 Методика капсульного программирования

Подэтап III-1. Определение функциональной нагрузки на капсулу.

1. Назначить капсуле функциональность *задача*.
2. Анализировать математическую постановку и модель (если имеется) решаемой при помощи МППА задачи и определить *глобальную степень параллельности* (ГСП) алгоритма.
3. Блоки алгоритма, которые могут быть выполнены параллельно, обозначить *тело*.
4. Провести анализ математической сложности тел (воспользоваться теорией сложности алгоритмов).
5. Провести анализ тел на потенциальный параллелизм и определить их степени параллельности (СП).
6. Если ГСП $\gg 4$, или сложность тел велика, или СП тел велика (*понятие «велика» необходимо уточнять*), то декомпозировать капсулу на последовательность капсул, каждой из которых назначить функциональность *тела*. Иначе этап I завершен.
7. Перейти к анализу капсул в последовательности:
   - 7.1 обозначить ГСП = СП текущей анализируемой капсулы;
   - 7.2 обозначить *алгоритм* — задача, решаемая текущей капсулой;
   - 7.3 вернуться к шагу 3.
8. Корректировать результаты подэтапа III-1 в соответствии с результатами подэтапов III-2 и III-3.

Результат применения данного этапа методологии:

- упорядоченная последовательность капсул, решающая поставленную задачу;
- каждая капсула решает часть общей задачи;
- СП каждой капсулы сравнима с 4.

Подэтап III-2. Разработка капсул.

Инициализация всех капсул статусом *не завершена*.

1. Выбрать очередную незавершенную капсулу; если таковых нет, перейти к подэтапу III-1.
2. Построить потоковый вычислительный граф алгоритма в соответствии с его математическим описанием (или моделью, если имеется).
3. Провести анализ графа с целью выделения повторяющихся и потенциально параллельно исполняемых блоков.
4. Применить существующие методологии распараллеливания алгоритмов и преобразовать граф таким образом, чтобы его СП была не больше 4.
5. Осуществить поиск циклически повторяющегося фрагмента графа и выполнить рекуррентную свертку этого фрагмента в вершины графа. Полученный граф называется *динамическим*.
6. Преобразовать динамический граф в соответствии со спецификацией функциональных возможностей РОУ. Результатом этого преобразования должна стать развернутая (детализированная) граф-капсула.
7. Если полученный граф не эффективен с точки зрения временны́х затрат на конфигурацию, пересылку промежуточных данных и т. п., вернуться к шагу 5. В случае, когда найти приемлемое эффективное решение не удается, перейти к шагу 9 подэтапа III-1.
8. Преобразовать граф-капсулу в символьную капсулу.
9. Установить статус капсулы — *отлаживается* и перейти к шагу 1.

Результат этапа — одна или более капсул, готовых к тестированию.

Подэтап III-3. Отладка капсул.

1. Отладить очередную капсулу со статусом *отлаживается*; если таковых нет, разработка завершена.
2. Анализировать результаты исполнения капсулы на соответствие требованиям точности и времени исполнения.
3. Если требования не удовлетворены, то статус капсулы — *не завершена*; вернуться к подэтапу III-2.
4. Установить статус капсулы — *готова* и вернуться к шагу 1.

Далее будет представлено применение первых двух этапов данной методологии на примере задачи РИС, а применение третьего этапа — на примере одного из алгоритмов распознавания, полосовой фильтрации.

4 Применение новой методологии для реализации распознавателя изолированных слов в среде рекуррентной архитектуры

Для экспериментальной апробации МПРА в результате этапа I методологии была выбрана предметная область распознавания слов. С этой целью в среде

Совокупность алгоритмов распознавателя слов

Алгоритм	Составной алгоритм	Пояснения	Блок	Капсула[1]
Формирование фрейма		Чтение 84 отсчетов из буфера кодека в буфер УУ	УУ	—
Масштабирование памяти		Если необходимо, производится масштабирование памяти фильтра Баттеруорта аналоговой частью коэффициента усиления	УУ	—
Фильтр Баттеруорта	Подготовка данных	Подготовка памяти фильтра и входных данных (для УУ), заполнение капсулы	УУ	~ 100
	1-я секция фильтра	Реализация 1-й секции фильтра, выходные данные становятся входом для 2-й секции	РОУ	
	Подготовка данных	Заполнение капсулы выходными данными 1-й секции	УУ	
	2-я секция фильтра	Реализация 2-й секции фильтра	РОУ	
Автоматический контроль коэффициента усиления (AGC — automatic gain control)	Локальный максимум на фрейме	Поиск по 84 отсчетам локального максимума амплитуды	РОУ	~ 100
	Анализ и обработка локального максимума на фрейме	Нормализация энергии, обновление глобального максимума, сохранение локального максимума в FIFO (First In, First Out), поиск нового максимума в FIFO	РОУ	—
	Масштабный фактор	Вычисление нового масштабного фактора	УУ	—
	Масштабирование	Если необходимо, масштабировать аналоговой частью память фильтра	РОУ	—
	Цифровая часть	Ограничение цифровой части	УУ	—
	Масштабирование	Если необходимо, масштабировать цифровой частью память фильтра	РОУ	—
	Масштабирование буфера	Масштабирование входного буфера	РОУ	100

Окончание таблицы на с. 142

Совокупность алгоритмов распознавателя слов (*окончание*)

Алгоритм	Составной алгоритм	Пояснения	Блок	Капсула[1]
Полосовая фильтрация	Подготовка данных	Подготовка памяти фильтра и входных данных (для УУ), заполнение капсулы	УУ	—
	1-я и 2-я секции фильтра	Реализация секций фильтра	РОУ	80
Извлечение речевых характеристик	Проверка на окончание слова	Если текущий фрейм является паузой, производится проверка на окончание слова	УУ	—
	Нелинейное преобразование	Переход в нелинейный спектр, вычисление логарифма энергии каждой полосы	РОУ	~ 30
	RASTA	RASTA-фильтрация в нелинейном спектре	РОУ	34
	Обратное преобразование	Переход в линейный спектр, вычисление экспоненты энергии каждой полосы	РОУ	~ 30
	Косинусное инверсное дискретное преобразование Фурье (ИДПФ)	Вычисление автокорреляционной свертки с использованием коэффициентов косинусного ИДПФ	РОУ	~ 30
	Рекурсия Дурбина–Скурра	Выделение кепстральных коэффициентов	РОУ	~ 200
	LPC (linear predictive coding) параметры	Вычисление LPC-параметров	РОУ	~ 100
	Дельта-расширение	Дельта-расширение характеристического вектора	РОУ	—
	Квантование ХВ	Вычисление евклидового расстояния	РОУ	22
	Определение конца слова	Определение конца слова	УУ	—
Алгоритм Витерби		Поиск подходящей модели слова	РОУ	~ 300

[1] Размер капсулы в операндах.

МПРА разрабатывается программа РИС. В результате применения этапа II методологии к задаче РИС была получена декомпозиция, представленная в таблице.

В этой таблице приведены также некоторые оценки по затратам на реализацию данных алгоритмов.

В настоящее время в соответствии с этой таблицей осуществляется реализация РИС в среде МПРА по методике, приведенной ниже.

4.1 Методика выделения функций распознавателя в капсулы

1. Выбрать алгоритм из таблицы и найти его в вычислительном графе распознавателя слов.
2. Определить все входные и выходные данные и области их допустимых значений.
3. Провести реализацию алгоритма на РОУ с заданными наборами входных данных и шаблонами выходных данных с учетом всех операций масштабирования, выполняемых внутри заменяемого кода.
4. Заменить часть исходного кода, реализующую выбранный алгоритм, на подпрограмму вызова капсулы.
5. Запустить РОУ для решения задачи распознавания на полном наборе примеров.
6. Если результаты совпали с исходной моделью, приступить к оптимизации капсулы; иначе вернуться к п. 3.
7. Если оптимизации капсулы не требуется, считать задачу решенной.
8. Если в результате оптимизации капсулы возникает необходимость модификации вычислительного графа — изменения входного и выходного наборов данных (ввиду наличия четырех вычислителей в РОУ), внести соответствующие изменения в вычислительный граф и программную модель распознавателя слов. Важно: нельзя изменять основной алгоритм вычислений, только количество данных в наборах (т. е. вычисление нескольких одинаковых тел алгоритма).
9. Если внесенные изменения при запуске обновленной модели приводят к получению тех же результатов, оптимизировать капсулу и вернуться к п. 4.

Для демонстрации этапа III разработанной методологии был выбран один из алгоритмов — полосовая фильтрация. Рассмотрим каждый из пунктов третьего этапа применительно к выбранному алгоритму.

4.2 Подэтап III-1 методики капсульного программирования

1. В качестве *задачи* принимается «реализация полосового фильтра». Предположительно требуется 1 капсула, которой назначается функциональность «полосовой фильтр».
2. Привести математическую постановку.

   Предобработанный речевой сигнал $x_{pf}(n)$, состоящий из 84 отсчетов, фильтруется на банке из 17 полосовых фильтров. В качестве полосовых фильтров

используются двухсекционные биквадратные фильтры. Передаточная характеристика каждого фильтра описывается формулой

$$H(z) = \frac{a}{1 + 2b_{11}z^{-1} + b_{12}z^{-2}} \frac{1}{1 + 2b_{21}z^{-1} + b_{22}z^{-2}}, \tag{3}$$

где a, b_{11}, b_{12}, b_{21}, b_{22} — коэффициенты фильтра, значения которых являются константами и хранятся в отдельной таблице. Таким образом, необходимо рассчитать 17 полосовых фильтров. Отсюда следует, что ГСП = 17.

Раскрыв z-преобразование в формуле (3), получим следующее представление двух последовательных фильтров:

$$\left.\begin{aligned} H_1(z) &\leftrightarrow y_i = ax_i - 2b_{11}y_{i-1} - b_{12}y_{i-2}\,; \\ H_2(z) &\leftrightarrow y'_i = ay_i - 2b_{21}y'_{i-1} - b_{22}y'_{i-2}\,. \end{aligned}\right\} \tag{4}$$

Данные уравнения описывают рекурсивные фильтры второго порядка. Выходные данные первого фильтра являются входными данными для второго, откуда следует, что выделить большее количество ГСП не получится. Кроме того, следует отметить, что фильтруется несмещенный входной вектор.

3. В качестве *тела* обозначить один фильтр, заданный формулой (3).
4. Воспользоваться теорией сложности для оценки вычислительной сложности фильтра.

   Для данного алгоритма $n = 84$. Секции фильтра рассчитываются последовательно, значит, их сложности складываются. Скорости выполнения операций сложения и умножения одинаковые, поэтому оценка времени вычисления одной секции фильтра равна $6n$. Тогда время расчета всего фильтра составит $2 \cdot 6n$. Другими словами, при переходе к оценке асимптотической сложности получим время порядка $O(n)$ (вполне ожидаемое время для алгоритма реального времени).

   Относительно низкая вычислительная сложность позволяет сделать предположение, что выбранное «тело» обладает невысокой СП.
5. Последовательный характер двухсекционного фильтра, казалось бы, препятствует параллельному вычислению обеих секций одновременно. Однако, как было отмечено ранее, данный фильтр обрабатывает несмещенный вектор. Это означает, что результат вычислений первой секции фильтра может быть сразу использован для вычисления второй секции. Таким образом, минимальная локальная СП = 2.

   Рассмотрим одну из секций фильтра более подробно (как следует из формул (4), они идентичны). Нетрудно заметить, что есть три независимых операции умножения; следовательно, СП секции равна 3.

   Таким образом, для фильтра, заданного формулой (3), СП = 6.

6. Имеем: ГСП $= 17$, СП $= 6$. Глобальная степень параллельности на порядок больше 4; следовательно, задачу необходимо декомпозировать на 17 подзадач (17 капсул), каждая из которых рассчитывает отдельный полосовой фильтр.
7. Завершить итерацию этапа I. Назначить *задачу* — «полосовой фильтр». ГСП $= 6$.
8. Обе секции фильтра можно рассчитывать параллельно, поэтому можно выделить два *тела*, каждое из которых соответствует секции фильтра.
9. Сложность каждого тела $O(n)$.
10. СП $= 3$.
11. Имеем: ГСП $= 6$, СП $= 3$. Это означает, что декомпозиция завершена. В результате получена последовательность из 17 капсул, каждая капсула рассчитывает обе секции фильтра.

Из всего сказанного можно сделать вывод, что для расчета каждой секции фильтра требуется два вычислительных устройства. Таким образом, необходимо реализовать алгоритм с СП $= 3$, используя два вычислителя.

4.3 Подэтап III-2 методики капсульного программирования

1. Для краткости опустим подробное описание пп. 1–4 данного подэтапа и приведем фрагмент преобразованного потокового графа с СП $= 4$. Указанный фрагмент изображен на рис. 1.
2. Динамический граф может быть получен путем сворачивания в вершины подграфов графа на рис. 1, выделенных пунктирными прямоугольниками. Нетрудно заметить, что указанные подграфы циклически повторяются.
3. Результат преобразования фрагмента потокового графа во фрагмент граф-капсулы представлен на рис. 2.

4.4 Подэтап III-3 методики капсульного программирования

Выходит за рамки рассмотрения данной статьи.
Символьная капсула полосового фильтра приведена в приложении.

5 Заключение

В ходе работ над новой вычислительной архитектурой для ее экспериментальной апробации был теоретически обоснован факт сходимости рекуррентного вычислительного процесса и создана методологическая база разработки ПО, предназначенного для функционирования в ее среде. Кроме того, была осуществлена апробация предложенной методологии на целостном наборе алгоритмов в

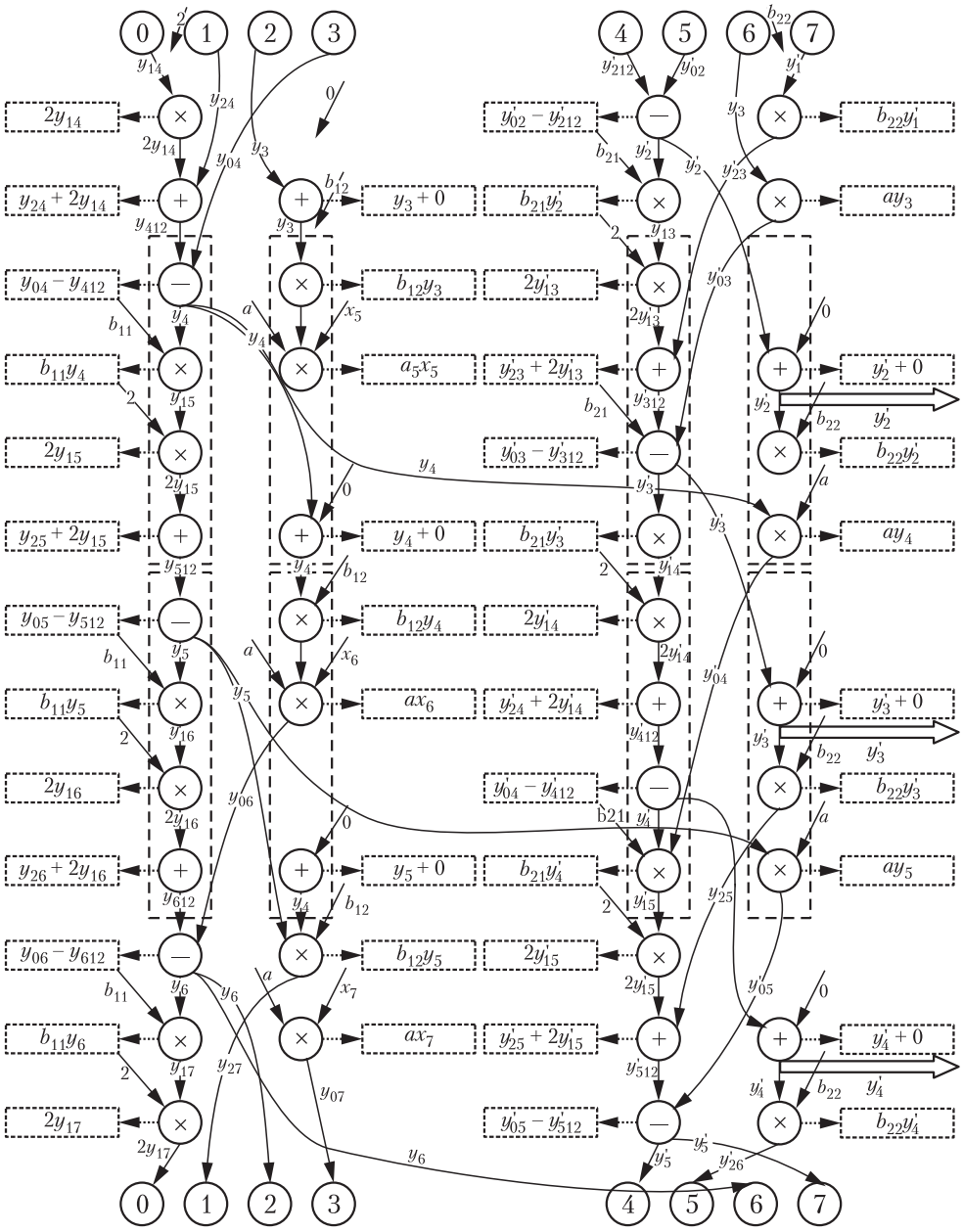

Рис. 1 Фрагмент преобразованного потокового графа полосового фильтра

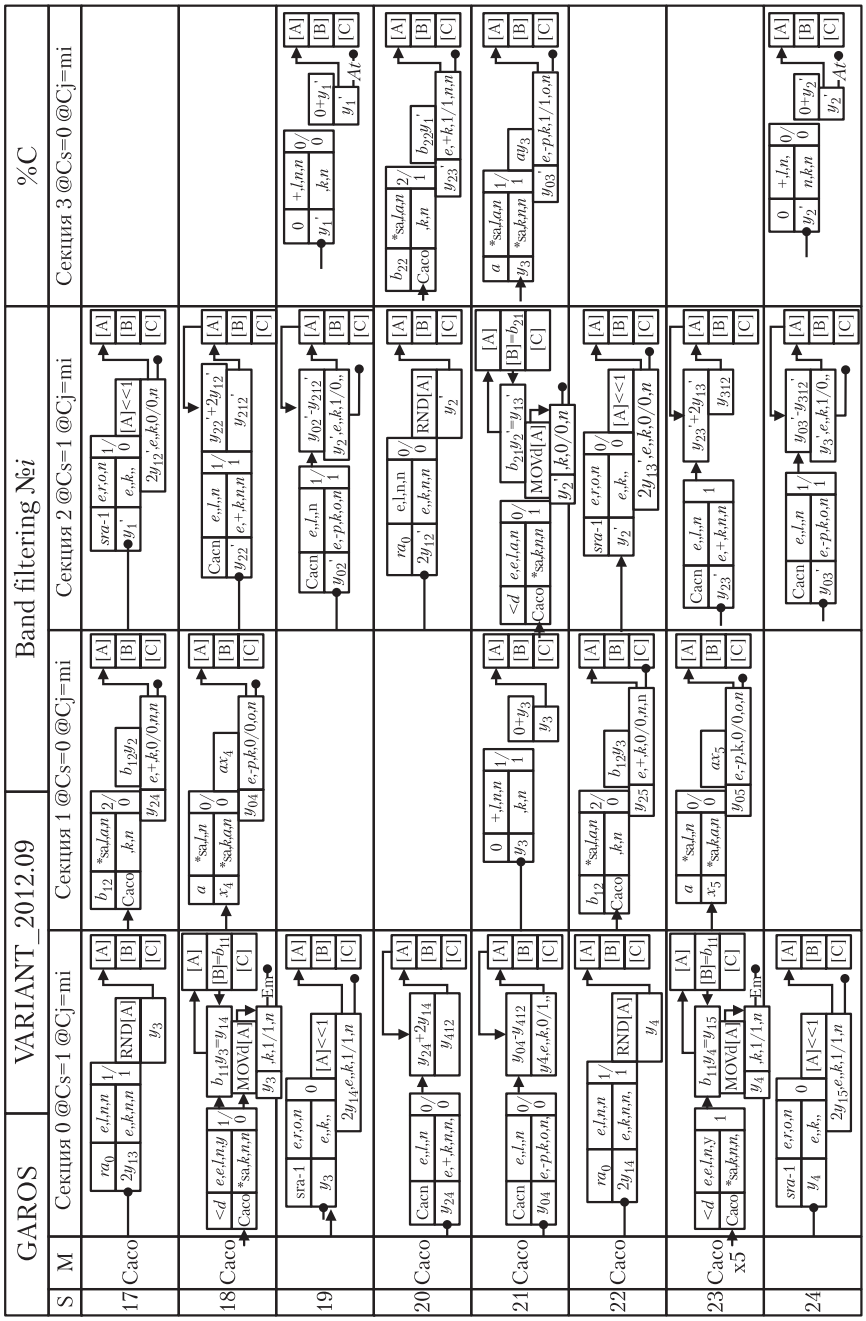

Рис. 2 Фрагмент граф-капсулы полосового фильтра

рамках представительной проблемной области цифровой обработки сигналов — распознавания отдельных слов-команд.

Положительные результаты апробации предложенной методологии могут служить основанием ее использования для создания технологии и программного комплекса проектирования ПО для МПРА. Некоторые компоненты данного комплекса в настоящее время уже разработаны [9–11] или находятся в разработке и отладке [12, 13].

Приложение

Символьная капсула полосового фильтра

```
Acg: Cx=3 Cp=4 Cc= Cr= Ce=ecs Cs=ei;
Ai: IOn=(Y') IOi=s IOs=84;
Am: %Mv=0010 %Mt=e %Mo=0 %Mm=s
At: Tc= Tm= Tu= Tr= Tn=0 To=0 Ts= Te= Ta= [ Sh=0 Sm=0 Dr=1000 ];
Di: @s=s V=b11 [ Oc=>db Ou=l Ot=t Sh=0 Sm=0 Dr=0001 Ds=n De= ];
{@s=s Oc= Ou= Ds= De= Sh= Sm= }
ДФЧ
Di: @s=s V=b12 [ Oc=*sa Ou=l Ot=t Sh=0 Sm=2 Dr=0010 Ds=a De= ];
{@s=e Oc=+ Ou=k Ds=n De=n Sh=0
Sm=0 } ДФЧ
Di: @s=s V=b21 [ Oc=>db Ou=l Ot=t Sh=0 Sm=0 Dr=0100 Ds=n De= ];
{@s=s Oc= Ou= Ds= De= Sh= Sm= }
ДФЧ
Di: @s=s V=b22 [ Oc=*sa Ou=l Ot=t Sh=1 Sm=2 Dr=1000 Ds=a De= ];
{@s=e Oc=+ Ou=k Ds=n De=n Sh=1
Sm=1 } ДФЧ
Ccs: [Oc= Ou=k Sh=0 Sm=0 Dr=0001 Ds= De=] Cj=mi Cl=> Cd= Cb= Cs=2 Cm= Cf=;
Ccs: [Oc= Ou=k Sh=0 Sm=2 Dr=0010 Ds= De=] Cj=mi Cl=< Cd= Cb= Cs=0 Cm= Cf=;
Ccs: [Oc= Ou=k Sh=0 Sm=0 Dr=0100 Ds=n De=] Cj=mi Cl=> Cd= Cb= Cs=2 Cm= Cf=;
Ccs: [Oc= Ou=k Sh=1 Sm=2 Dr=1000 Ds=n De=] Cj=mi Cl=< Cd= Cb= Cs=0 Cm= Cf=;
Cacn: @s=e [Oc= Ou=l Ot=t Sh=0 Sm=0 Dr=0001 Ds= De=n ];
{@s=e Oc= Ou=k Ds= De= Sh=1 Sm=0 } ДФЧ
Di: @s=s V=a [Oc= Ou=l Ot=t Sh=1 Sm=0 Dr=0001 Ds=n De=n ];
{@s=e Oc=-p Ou=k Ds=n De=n Sh=0 Sm=0 }
ДФЧ
Di: @s=s V=a [Oc=*sa Ou=l Ot=t Sh=0 Sm=0 Dr=0010 Ds= De=n ];
{@s=e Oc=-p Ou=k Ds=o De=n Sh=0 Sm=0
} ДФЧ
Di: @s=s V=a [Oc=*sa Ou=l Ot=t Sh=1 Sm=1 Dr=0100 Ds=n De=n ];
{@s=e Oc=-p Ou=k Ds=n De=n Sh=0
Sm=1 } ДФЧ
Di: @s=s V=a [Oc=*sa Ou=l Ot=t Sh=1 Sm=1 Dr=1000 Ds=a De=n ];
{@s=e Oc=-p Ou=k Ds=o De=n Sh=0 Sm=1
} ДФЧ
Di: @s=s V=x0 [Oc=*sa Ou=k Ot=t Sh=1 Sm=0 Dr=0001 Ds=n De=n ];
```

```
{@s=s Oc= Ou= Ds= De= Sh= Sm= }
ДФЧ
Di: @s=s V=x1 [Oc=*sa Ou=k Ot=t Sh=0 Sm=0 Dr=0010 Ds=a De=n ];
{@s=s Oc= Ou= Ds=o De= Sh= Sm= }
ДФЧ
Caco: @s=e @Ad=4 @Au=m [ Oc=*sa Ou=k Ot=t Sh=0 Sm=1 Dr=0001 Ds=n De=n ];
{@s=e Oc= Ou= Ds= De=
Sh= Sm= }ДФЧ
Di: @s=e V=<d [Oc=e Ou=l Ot=t Sh=0 Sm=1 Dr=0001 Ds=n De=y ];
{@s=s Oc= Ou=k Ds=n De=n Sh=1 Sm=1 }
ДФЧ
Di: @s=s V=0 [Oc=+ Ou=l Ot=t Sh=1 Sm=1 Dr=0010 Ds=n De=n ];
{@s=s Oc= Ou= Ds= De= Sh= Sm= } ДФЧ
Di: @s=s V=0 [Oc=+ Ou=l Ot=t Sh=0 Sm=0 Dr=1000 Ds=n De=n ];
{@s=s Oc= Ou= Ds= De= Sh= Sm= } ДФЧ
Cacn: @s=e [Oc=*sa Ou=l Ot=t Sh=0 Sm=1 Dr=0001 Ds=a De=n ];
{@s=e Oc= Ou= Ds= De= Sh= Sm= } ДФЧ
Am: %Mv=0101 %Mt=e %Mo=0 %Mm=s
Di: @s=s V=sra-1 [Oc=e Ou=r Ot=t Sh=1 Sm=0 Dr=0001 Ds=o De=n ];
{@s=e Oc= Ou=k Ds=n De= Sh=1 Sm=1
} ДФЧ
Di: @s=s V=0 [Oc= Ou=l Ot=t Sh=0 Sm=1 Dr=0100 Ds=n De=n ];
{@s=s Oc= Ou= Ds= De= Sh= Sm= } ДФЧ
Di: @s=s V=x0 [Oc=*sa Ou=k Ot=t Sh=0 Sm=0 Dr=0010 Ds=n De=n ];
{@s=s Oc= Ou= Ds=n De= Sh= Sm= }
ДФЧ
Di: @s=s V=ra0 [Oc=e Ou=l Ot=t Sh=1 Sm=1 Dr=0001 Ds=n De=n ];
{@s=s Oc= Ou= Ds= De= Sh= Sm= } ДФЧ
Di: @s=s V=0 [Oc= Ou=k Ot=t Sh=1 Sm=1 Dr=0001 Ds=n De=n ];
{@s=s Oc= Ou= Ds= De= Sh= Sm= } ДФЧ
Cacn: @s=e [Oc= Ou=k Ot=t Sh=0 Sm=2 Dr=0010 Ds=n De=n ];
{@s=e Oc= Ou= Ds= De= Sh= Sm= } ДФЧ
Caco: @s=e @Ad=4 @Au=m [ Oc= Ou=k Ot=t Sh=0 Sm=2 Dr=0010 Ds=n De=n ];
{@s=e Oc= Ou= Ds= De= Sh=
Sm=}ДФЧ
Cacn: @s=e [Oc= Ou=l Ot=t Sh=1 Sm=1 Dr=0100 Ds= De=n ];
{@s=e Oc= Ou=k Ds= De= Sh=1 Sm=0 } ДФЧ
Di: @s=s V=x2 [Oc=*sa Ou=k Ot=t Sh=0 Sm=0 Dr=0010 Ds=a De=n ];
{@s=s Oc= Ou= Ds=o De= Sh= Sm= }
ДФЧ
Di: @s=e V=<d [Oc=e Ou=l Ot=t Sh=1 Sm=0 Dr=0100 Ds=a De=n ];
{@s=e Oc= Ou=k Ds=n De=n Sh=0 Sm=0 }
ДФЧ
Cacn: @s=e [Oc=*sa Ou=k Sh=1 Sm=0 Dr=0100 Ds=n De=n ];
{@s=e Oc= Ou= Ds= De= Sh= Sm= } ДФЧ
Caco: @s=e @Ad=4 @Au=m [ Oc=*sa Ou=k Ot=t Sh=1 Sm=0 Dr=0100 Ds=n De=n ];
{@s=e Oc= Ou= Ds= De=
Sh= Sm=}ДФЧ
Di: @s=s V=x3 [Oc=*sa Ou=k Ot=t Sh=0 Sm=0 Dr=0010 Ds=a De=n ];
{@s=s Oc= Ou= Ds=o De= Sh= Sm= }
ДФЧ
```

```
Di: @s=s V=sra-1 [Oc=e Ou=r Ot=t Sh=0 Sm=1 Dr=0100 Ds=o De=n ];
{@s=e Oc= Ou=k Ds=n De=n Sh=0 Sm=0
} ДФЧ
Di: @s=s V=ra0 [Oc=e Ou=l Ot=t Sh=0 Sm=0 Dr=0100 Ds=n De=n ];
{@s=s Oc= Ou= Ds= De= Sh= Sm= } ДФЧ
Di: @s=s V=0 [Oc= Ou=k Ot=t Sh=1 Sm=1 Dr=0100 Ds=n De=n ];
{@s=s Oc= Ou= Ds= De= Sh= Sm= } ДФЧ
Cacn: @s=e [Oc= Ou=k Sh=1 Sm=2 Dr=1000 Ds=n De=n ];
{@s=e Oc= Ou= Ds= De= Sh= Sm= } ДФЧ
Caco: @s=e @Ad=4 @Au=m [ Oc= Ou=k Ot=t Sh=1 Sm=2 Dr=1000 Ds=n De=n ];
{@s=e Oc= Ou= Ds= De= Sh= Sm= } ДФЧ
Asi: %St=1 @s=s [Oc=*sa Ou=k Ot=t Sm=0 Dr=0001 Ds=a De=n ];
{@s=s Oc= Ou= Ds=o De= Sh= Sm= } ДФЧ
Apdi: V0=x4 [Sh=0] V1=x5 [Sh=0] V2=x6 [Sh=0];
Apdi: V0=x7 [Sh=0] V1=x8 [Sh=0] V2=x9 [Sh=0];
Apdi: V0=x10 [Sh=0] V1=x11 [Sh=0] V2=x12 [Sh=0];
.
Apdi: V0=x79 [Sh=0] V1=x80 [Sh=0] V2=x81 [Sh=0];
Apdi: V0=x82 [Sh=0];
Afi:
Di: @s=s V=x83 [Oc=*sa Ou=k Ot=t Sh=0 Sm=0 Dr=0010 Ds=a De=n ];
{@s=s Oc= Ou= Ds=n De= Sh= Sm= }
ДФЧ
Di: @s=s V=0 [Oc=n Ou=l Ot=t Sh=0 Sm=2 Dr=0010 Ds=n De=n ];
{@s=s Oc= Ou= Ds= De= Sh= Sm= } ДФЧ
Di: @s=s V=a [Oc=*sa Ou=l Ot=t Sh=1 Sm=1 Dr=1000 Ds=a De=n ];
{@s=e Oc=-p Ou=k Ds=n De=n Sh=0 Sm=1
} ДФЧ
Caco: @s= @Ad= @Au=r[ Oc=n Ou= Ot=t Sh= Sm= Dr=1111 Ds=n De=n ];
{@s= Oc= Ou= Ds= De= Sh= Sm= }
ДФЧ
Cacn: @s=e [Oc=*sa Ou=k Sh=1 Sm=0 Dr=0100 Ds=n De=n ];
{@s=e Oc= Ou= Ds= De= Sh= Sm= } ДФЧ
Az:
```

Литература

1. *Agerwala T., Arvind*. Data flow systems // IEEE Computer, 1982. Vol. 15. No. 2. P. 10–13.
2. *Arvind, Culler D. E.* Data flow architectures // Ann. Rev. Comput. Sci., 1986. Vol. 1. P. 225–253.
3. *Arvind, Iannucci R. A.* A critique of multiprocessing von Neumann style // 10th ISCA Proceedings, 1983. P. 426–436.
4. *Роджерс Х.* Теория рекурсивных функций и эффективная вычислимость / Пер. с англ. — М.: Мир, 1972. С. 17–52. (*Rogers, H., Jr.* Theory of recursive functions and effective computability. — McGraw Book Co., 1967.)
5. *Филин А. В.* Особенности обработки сигналов на процессоре с рекуррентно-динамической парадигмой вычислений // Системы и средства информатики, 2001. Вып. 11. С. 262–282.

6. *Степченков Ю. А., Петрухин В. С.* Особенности гибридного варианта реализации на ПЛИС рекуррентного обработчика сигналов // Системы и средства информатики, 2008. Доп. вып. С. 118–129.
7. *Филин А. В.* Динамический подход к выбору архитектуры вычислительных устройств обработки сигналов // Системы и средства информатики, 2001. Вып. 11. С. 247–261.
8. *Степченков Д. Ю.* Программа РЕКУРРЕНТ — инструмент анализа и синтеза целочисленных преобразователей // Системы и средства информатики, 2005. Вып. 15. С. 397–407.
9. *Зеленов Р. А., Степченков Ю. А., Волчек В. Н., Петрухин В. С., Прокофьев А. А., Хилько Д. В.* Система капсульного программирования и отладки (СКАТ). Версия 2. Свид. № 2013610198, 2013.
10. *Хилько Д. В., Степченков Ю. А.* Средства имитационного моделирования потоковой рекуррентной архитектуры (СИМПРА). Свид. № 2013610199, 2013.
11. *Хилько Д. В., Степченков Ю. А., Шнейдер А. Ю.* Программа обработки результатов моделирования потоковой рекуррентной архитектуры (ПРАПОР). Свид. № 2013610199, 2013.
12. *Зеленов Р. А., Степченков Ю. А., Волчек В. Н., Хилько Д. В., Шнейдер Ю. А., Прокофьев А. А.* Система капсульного программирования и отладки // Системы и средства информатики, 2010. Вып. 20. № 1. С. 24–30.
13. *Хилько Д. В., Степченков Ю. А.* Модель потоковой архитектуры на примере распознавателя слов // Системы и средства информатики, 2012. Т. 22. № 2. С. 48–57.

THEORETICAL ASPECTS OF PROGRAMMING METHODOLOGY DEVELOPMENT FOR RECURRENT ARCHITECTURE

***D. Khilko*[1] *and Yu. Stepchenkov*[2]**

[1]IPI RAN, Moscow, Russia, dhilko@yandex.ru
[2]IPI RAN, Moscow, Russia, YStepchenkov@ipiran.ru

Abstract: The paper is dedicated to a new recurrent dataflow computational paradigm and the methodology intended for solving and programming of corresponding problems. These problems are meant to be solved with the computational device that is being developed, which architecture is based on ideas and principles of proposed paradigm. Also, its realization in multicore dataflow recurrent architecture is suggested. Recurrent organization of computational process is theoretically proven to converge by using terms and theorems of the recursive functions theory. The software engineering problem for this architecture is described and its solution — recurrent dataflow programming methodology — is suggested. This methodology is demonstrated on a problem of isolated words recognition with device based on a new architecture. Also, one of the basic algorithms of the mentioned problem — band-pass filtering — is realized by using the proposed methodology step by step.

Keywords: computational paradigm; programming methodology recursiveness; dataflow architecture

DOI: 10.14357/08696527130210

Acknowledgments

The work was partially supported by the Program for Basic Research of the Department of Information Technologies and Computing Systems of the Russian Academy of Sciences in 2013 (Project 1.5) and the Presidium of the Russian Academy of Sciences (Project 16).

References

1. *Agerwala, T., and Arvind.* 1982. Data flow systems. *IEEE Computer* 15:10–13.
2. *Arvind, and D. E. Culler.* 1986. Data flow architectures. *Ann. Rev. Comput. Sci.* 1:225–53.
3. *Arvind, and R. A. Iannucci.* 1983. A critique of multiprocessing von Neumann style. *10th ISCA Proceedings*. 426–36.
4. *Rogers, H., Jr.* 1967. *Theory of recursive functions and effective computability*. McGraw Book Co. 17–52.
5. *Filin, A. V.* 2001. Osobennosti obrabotki signalov na processore s rekurrentno-dinamicheskoj paradigmoj vychislenij [Signal processing features using processor based on dynamic recurrent computational paradigm]. *Systems and Means of Informatics* 11:262–82.
6. *Stepchenkov, Yu. A., and V. S. Petruhin.* 2008. Osobennosti gibridnogo varianta realizacii na PLIS rekurrentnogo obrabotchika signalov [The hybrid variant FPGA realization features of recurrent signal processor]. *Systems and Means of Informatics* Add.:118–29.
7. *Filin, A. V.* 2001. Dinamicheskij podhod k vyboru arhitektury vychislitel'nyh ustrojstv obrabotki signalov [A dynamic approach to computing architecture selection of signal processing devices]. *Systems and Means of Informatics* 11:247–61.
8. *Stepchenkov, D. Yu.* 2005. Programma REKURRENT — instrument analiza i sinteza celochislennyh preobrazovatelej [REKURRENT program as a tool for analysis and synthesis of integer converters]. *Systems and Means of Informatics* 15:397–407.
9. *Zelenov, R. A., Yu. A. Stepchenkov, V. N. Volchek, V. S. Petruhin, A. A. Prokof'ev, and D. V. Khilko.* 2013. Sistema kapsul'nogo programmirovanija i otladki (SKAT). Versija 2 [The system of capsule programming and debugging (SKAT). Ver. 2]. Certificate on official registration of the computer program No. 2013610198.
10. *Khilko, D. V., and Yu. A. Stepchenkov.* 2013. Sredstva imitacionnogo modelirovanija potokovoj rekurrentnoj arhitektury (SIMPRA) [Imitational modeling tools for recurrent dataflow architecture (SIMPRA)]. Certificate on official registration of the computer program No. 2013610199.

11. *Khilko, D. V., Yu. A. Stepchenkov, and A. Ju. Shneyder*. 2013. Programma obrabotki rezul'tatov modelirovanija potokovoj rekurrentnoj arhitektury (PRAPOR) [The program processing the simulation results of recurrent dataflow architecture (PRAPOR)]. Certificate on official registration of the computer program No. 2013610199.
12. *Zelenov, R. A., Yu. A. Stepchenkov, V. N. Volchek, D. V. Hilko, A. Ju. Shneyder, and A. A. Prokofyev*. 2010. Sistema kapsul'nogo programmirovanija i otladki [System of capsule programming and debugging]. *Systems and Means of Informatics* 20(1):24–30.
13. *Khilko, D. V., and Yu. A. Stepchenkov*. 2012. Model' potokovoj arhitektury na primere raspoznavatelja slov [Dataflow architecture model and its usage with a word recognizer program as an example]. *Systems and Means of Informatics* 22(2):48–57.