УДК 004.272.44 DOI: 10.31114/2078-7707-2020-2-73-80

Специализированные преобразователи тегов для рекуррентного обработчика сигналов

Ю.А. Степченков, Д. В. Хилько, Ю.И. Шикунов, Г.А. Орлов

Федеральный исследовательский центр «Информатика и Управление» РАН, г. Москва

ia_ste@mail.ru, dhilko@yandex.ru, YIShikunov@yandex.ru, Orlov.jaja@gmail.com

***Аннотация* — Настоящая статья посвящена исследованию применимости специализированных рекуррентных преобразователей в рекуррентном операционном устройстве для задач цифровой обработки сигналов. Рассматриваются основные особенности и существующие проблемы реализации рекуррентности в операционном устройстве, построенном на основе потокового (data-flow) принципа. Приводится анализ ограниченного подмножества алгоритмов цифровой обработки сигналов с целью построения специализированных рекуррентных цепочек и преобразователей их реализующих. Представлены результаты построения некоторых специализированных преобразователей тегов и реализации демонстрационного алгоритма фильтрации Баттерворта.**

***Ключевые слова* — цифровая обработка сигналов, потоковая вычислительная архитектура, рекуррентность, преобразователь тегов.**

I. Введение

Потоковые вычислительные архитектуры обладают высоким потенциалом производительности на ряде предметных областей. Тем не менее, потоковая модель вычисления не получила широкого распространения ввиду целого ряда проблем, присущих данному классу, таких как: реализация рекурсии, циклов, итераций, работа с константами и др. [1].

В ряде зарубежных экспериментальных проектов, проводившихся с начала 1980-х до середины 2000-х годов, в которых разрабатывались потоковые архитектуры, не нашлось эффективных путей решения перечисленных проблем.

Насколько нам известно, предметно вопросами разработки потоковых архитектур в настоящее время занимаются два коллектива в РФ: Институт проблем проектирования в микроэлектронике РАН (ИППМ РАН) в части создания потоковых систем с массовым параллелизмом и динамическим распределением ресурсов (проект БУРАН [2]) и в ФИЦ ИУ РАН в части использования потоковой парадигмы в области цифровой обработки сигналов (ЦОС) в реальном масштабе времени [1].

Действительно, эффективность проекта БУРАН подтверждена программой моделью, которая базируется на использовании ассоциативной памяти большого объема (наиболее ресурсоемкий ресурс) и тегируемых данных, где размер (разрядность) тегов[1] в несколько раз превышает размер собственно обрабатываемых данных. Указанные особенности исключают возможность использования данного подхода в области ЦОС по экономическим соображениям.

В ФИЦ ИУ РАН разработана концепция универсальной потоковой архитектуры, предназначенной для реализации параллельных вычислительных процессов в области ЦОС. В основе данной архитектуры лежат принципы рекуррентной теории самоорганизации [3], графодинамики, а также парадигмы управления вычислениями потоком активных данных [1]. Данные принципы были объединены в цельный рекуррентно-динамический подход для оценки степени его эффективности в области ЦОС. При этом для реализации были выбраны не отдельные случайные алгоритмы ЦОС, а их совокупность в рамках цельной предметной области - распознавание изолированных слов команд (РИС).

Таким образом, главными критериями разрабатываемой архитектуры являются:

- поиск механизмов эффективной реализации "мелкозернистого" параллелизма на ограниченном числе вычислительных ядер на базе потоковой концепции;
- учет специфики ЦОС-области: минимизация аппаратных затрат в части необходимых типов и объемов памяти и разрядности тегов - необходимого компонента потоковой архитектуры;
- используемые вычислительные ядра не должны требовать какой-либо потоковой специфики и на первом этапе апробации разрабатываемой архитектуры возможно использование ядер традиционной архитектуры.

Центром приложения исследовательских усилий стала разработка рекуррентного операционного устройства (РОУ). РОУ реализует самый нижний (операционный) уровень вычислительного процесса, что позволяет максимально плодотворно оценить жизнеспособность рекуррентно-динамического подхода. На операционном уровне достижимо максимальное

[1] В области потоковых вычислительных архитектур для обозначения набора функциональных (служебных) полей (и каждого поля в отдельности) используются два термина: токен и тег. Дальше в рамках данной статьи будет использоваться термин тег.

распараллеливание вычислительного процесса (уровень «мелкозернистого» параллелизма) [1].

Для отработки новых рекуррентных принципов первичный объект апробации представлен в виде гибридной архитектуры рекуррентного обработчика сигналов (ГАРОС). Т.е. в виде двухуровневой архитектуры: управляющего уровня на базе традиционного фон-неймановского процессора низкой производительности и операционного уровня – РОУ на базе множества однотипных рекуррентных вычислительных ядер [1].

При таком подходе управляющий уровень реализует функции:

- процессора ввода-вывода для связи многоядерного РОУ с внешним миром;
- интерфейса между системным ПО и вычислительным устройством, базирующимся на потоковой парадигме;
- обработки исключительных ситуаций в РОУ;
- вычислителя для последовательных частей алгоритма (при необходимости).

Архитектура РОУ реализует потоковую модель вычислений, но с принципиальными отличиями:

- рекуррентная реализация всех элементов вычислительного процесса;
- повышение статуса операндов и управляющих структур до уровня самодостаточных единиц;
- использование событийной схемы взаимодействия элементов вычислительного процесса;
- использование векторизации для организации естественно-масштабируемого параллелизма.

На базе ГАРОС построены имитационная модель, VHDL прототип, а также создан тестовый стенд с реализацией ГАРОС на ПЛИС [4]. На данном этапе ПЛИС-реализация потоковой архитектуры должна соответствовать двум основным требованиям.

1) Реализация в аппаратуре максимально-возможной алгоритмической степени параллельности, в том числе, за счет широкого использования низкоуровневого (капсульного) стиля программирования операционного уровня.

2) Учет экономических требований ЦОС-реализаций (минимально-возможные объемы используемых типов памяти), минимально-возможная избыточность представления функциональных полей (тегов) и учет специфики алгоритмов ЦОС, позволяющей использовать статические, а не динамические методы распределения данных.

Реализация в изначальной потоковой рекуррентной вычислительной архитектуре этих требований позволила подтвердить вычислительную эффективность РОС, в том числе и по сравнению с высокопроизводительными сигнальными процессорами, предлагаемыми компанией Texas Instruments: TI C55x DSP and TI C66x DSP Core [4]. Однако при этом ряд упомянутых основополагающих принципов был реализован не в полной мере. При этом вычислительная эффективность получена, в том числе, ценой высокой трудоемкости программирования. Во многом требуемая последовательность тегов для реализации алгоритмов обеспечена разработчиками капсул.

Одним из таких решений было использование универсальных рекуррентных преобразователей, позволивших реализовать рекуррентную развертку произвольных цепочек длины не более двух операндов за счет введения избыточности [5].

Более полная реализация в РОС изначальных принципов позволит уменьшить трудоемкость капсульного программирования и снизить избыточность представления тегов. Одним из направлений, которое может приблизить нас к такому развитию, является доработка принципа рекуррентной развертки вычислительного процесса. Первым шагом в этом направлении является разработка библиотеки специализированных преобразователей для конкретной предметной области. В данной статье мы исследуем возможность и практичность использования специализированных преобразователей для решения конкретных задач в предметной области ЦОС.

II. Механизм рекуррентной развертки в РОУ

A. *Принцип рекуррентной развертки*

Особенностью потоковых вычислительных систем является манипуляция двумя потоками: приоритетным потоком входных данных и потоком инструкций. В традиционных вычислительных системах процесс решения задачи представляется в виде полностью запрограммированной трассы (графа) вычислительного процесса. Например, хотя в потоковых системах поток данных и является инициатором вычислений, их трасса описана в качестве потока инструкций, содержащего множество вершин графа вычислительного процесса, представленных в виде пакетов (инструкция + контекст) в памяти.

Фундаментальной для РОУ является концепция графодинамики. В рамках этой концепции фиксированная трасса вычислительного процесса не задана явным образом. При этом в памяти определяется только набор начальных состояний, которые служат источником динамически порождаемого частного случая графа вычислительного процесса.

Второй принцип РОУ, дополняющий концепцию графодинамики – это принцип самодостаточности данных. В соответствии с этим принципом в вычислительном процессе участвует лишь один поток – поток самодостаточных тегированных данных. Последовательность таких данных, решающая поставленную задачу, называется капсулой.

Для порождения графа вычислительного процесса над тегированными данными рекуррентно выполняется функция преобразования. Данный процесс был назван рекуррентной разверткой. Элемент такого графа, порождаемый из одного элемента самодостаточных

данных (ЭСД) - операнда, мы назовем рекуррентной цепочкой.

Объединяя эти два фундаментальных принципа, мы можем сказать, что в РОУ капсула является программой - описанием начального состояния вычислительного процесса, рекуррентно разворачиваемого в динамический граф конкретного вычисления, решающего поставленную задачу.

B. Модуль преобразования тегов

Для обеспечения рекуррентной развертки операндов в РОУ используется специализированный блок – преобразователь тегов (ПТ). В статье [6] приводится подробный анализ возможных реализаций преобразователя тегов. Для изначальной был выбран универсальный преобразователь тегов с кодовой избыточностью.

Универсальный ПТ порождает древовидную структуру графа, которая сходится к нулевому самовозвратному полюсу (нулевой вершине). Этот составной ПТ представляет собой совокупность конкатенируемых двухразрядных элементарных преобразователей, каждый из которых реализует функцию - арифметический сдвиг входного значения функционального поля вправо на один разряд. Граф преобразования такого ПТ представлен на рис. 1.

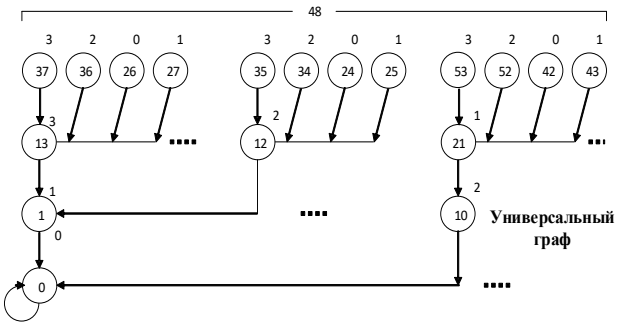

Рис. 1. Граф универсального ПТ

Данное решение позволило избежать необходимости производить времязатратную конфигурацию преобразователя тегов, однако помимо ограничений на длину цепочки потребовало внедрить до 100% избыточности в теговых (функциональных) полях. Кодовая избыточность универсального ПТ заключается в кратном увеличении этих полей.

В изначальном прототипе архитектуры блок ПТ являлся составной частью блока «Вычислитель» и использовался для преобразования данных на выходе из операционного уровня. Как показала практика реализации алгоритмов РОС, универсальный преобразователь получил малую долю практического использования. Основными проблемами в более широком применении функциональных преобразователей в текущей реализации прототипа архитектуры стали:

- конвейеризация вычислительного процесса, которая приводит к большим задержкам в ходе вычисления одной цепочки;
- большое количество внутренних регистров вычислителя, добавленных для повышения производительности алгоритмов РОС;
- малая фиксированная длина цепочки.

Внедрение в блок «Итератор» дополнительного преобразователя тегов, который выполняет ограниченное итеративное рекуррентное преобразование над фиксированным набором тегов, позволило снизить избыточность в представление тегов [7]. Представляется, что дальнейшее снижение избыточности возможно за счет разработки специализированных преобразователей - библиотеки рекуррентных преобразователей, предназначенных для задач предметной области ЦОС.

III. Специализированные рекуррентные преобразователи

A. Анализ алгоритмов РИС

Для апробации прототипа МПРА была выбрана задача распознавания изолированных слов-команд (РИС). Данный выбор обусловлен как наличием реализованной системы, созданной авторами настоящей статьи в рамках совместного проекта с компанией Microchip, так и высокой степенью покрытия алгоритмами, реализованными в РИС, предметной области ЦОС. Выбранная задача была запрограммирована и апробирована как на имитационной модели [5], так и на аппаратной модели и ее реализации в макетном ПЛИС образце [4].

Последующий анализ полученных результатов подтвердил высокий потенциал производительности архитектуры. Однако стремление достичь максимальной производительности с учетом обозначенных проблем привело к меньшей степени использования универсального рекуррентного преобразователя. Создание библиотеки специализированных рекуррентных преобразователей для предметной области ЦОС, позволит не менее эффективно реализовывать алгоритмы и повысить применимость основополагающих принципов рекуррентно-динамического подхода.

В статье [8] приводится описание совокупности алгоритмов распознавания слов-команд. Как известно, основной операцией ЦОС является линейная свертка, которую, условно, можно описать формулой (1):

$$f = a * b + c * d + e * g + \ \ldots \qquad (1)$$

С учетом данного факта была выдвинута и принята без доказательства (на текущем этапе исследований) гипотеза о том, что существует такое ограниченное и небольшое подмножество рекуррентных цепочек и порождающих их преобразователей, которые могут быть применены для организации необходимых рекуррентных вычислений в РОУ. Поиск таких цепочек осуществлялся в соответствии со следующим критерием: искомая цепочка должна присутствовать в схеме вычислений по крайней мере двух алгоритмов РИС.

На основе данного подхода удалось выделить подмножество из четырех алгоритмов РИС: полосовая фильтрация, фильтр Баттерворта, натуральный логарифм и экспонента. Полосовой фильтр и фильтр

Баттерворта представляют собой двухсекционные рекурсивные фильтры второго порядка, в то время как натуральный логарифм и экспонента – степенные ряды Тейлора.

В рамках реализации РИС полосовые фильтры и фильтр Баттерворта имеют схожую структуру. Однако полосовые фильтры требуют одну память для хранения результатов фильтрации предыдущего фрейма, а фильтр Баттерворта – двух памятей. Поэтому в качестве объекта исследования был выбран именно фильтр Баттерворта, структура и назначение которого описывается далее.

Входной речевой сигнал x(n) префильтруется для получения предобработанного сигнала xpf(n) с целью подавления низкочастотного шума и удаления постоянного смещения звукового сигнала, записанного с микрофона, привносимых звуковым оборудованием, каналом и устройством квантования. Для этого используются двухсекционный фильтр высоких частот Баттерворта с частотой отсечки 120 Гц на уровне -3дБ от максимума. Он реализуется в соответствии с формулой (2):

$$H_{\text{Б}}(z) = \frac{1-2z^{-1}+z^{-2}}{1+b_{11}z^{-1}+b_{12}z^{-2}} * \frac{1-2z^{-1}+z^{-2}}{1+b_{21}z^{-1}+b_{22}z^{-2}}. \quad (2)$$

Здесь b_{11}, b_{12}, b_{21}, b_{22} — коэффициенты фильтрации, определяемые частотой отсечки. В данном случае для частоты отсечки 120 Гц коэффициенты фильтра имеют значения:

b_{11} = -1,94921875;

b_{12} =0,95306396484375;

b_{21} = -1,8865966796875;

b_{22} = 0,89031982421875.

Отличие структуры полосовых фильтров от Баттерворта заключается в коэффициентах b_{ij}, а также в том, что их числитель X(z) = a.

Фильтр Баттерворта именно такой структуры реализован в РИС. Мы также программно реализовали данный фильтр для текущей версии ГАРОС. Сравнительные с микроконтроллером компании Microchip результаты имитационного моделирования этой реализации представлены в табл. 1 работы [5], а результаты аппаратной апробации – в табл. 1 работы [4].

На рис. 2 представлена схема вычислительного процесса, реализующая структуру одной итерации вычисления одной секции фильтра Баттерворта для очередного входного отсчета x(i). Вычисление линейной свертки осуществляется с использованием операции умножения с накоплением, а хранение памятей фильтра осуществляется путем прокачки необходимых данных через внутренние регистры РОУ при помощи специально предусмотренных для этого механизмов архитектуры.

Формулы (3) и (4) содержат математическое описание передаточных функций первых секций полосового фильтра и фильтра Баттерворта (после раскрытия z-преобразования), соответственно, а формула (5) – обобщенный вариант степенного ряда (с точностью до констант).

$$\text{H}_{\text{П}}^{1}(z) \leftrightarrow y_i = a * x_i + 2 * (-b_{11}) * y_{i-1} + \\ +(-b_{12}) * y_{i-2} \quad (3)$$

$$\text{H}_{\text{Б}}^{1}(z) \leftrightarrow y_i = {}^{1}\!/_{2} * (x_i + x_{i-2}) - x_{i-1} - b_{11} * y_{i-1} - \\ - b_{12} * y_{i-2} \quad (4)$$

$$\text{G} = \gamma + c_1 * \gamma^2 + c_2 * \gamma^3 + c_3 * \gamma^4 + c_4 * \gamma^5 + c_5 * \gamma^6 + \\ +c_6 * \gamma^7 + c_7 * \gamma^8, \text{ где } \gamma \in \left(-{}^{1}\!/_{2}, 0\right). \quad (5)$$

В процессе анализа математического описания и преобразования уравнений нам удалось выявить повторяющиеся последовательности в каждой из формул (3) – (5). Результаты преобразования уравнений представлены в формулах (6) – (8) (порядок алгоритмов сохранен).

$$\text{H}_{\text{П}}(z) \leftrightarrow y_i = a_0 * x_i + (b_1 * y_{i-1} + b_2 * y_{i-2}) \quad (6)$$

$$\text{H}_{\text{Б}}(z) \leftrightarrow y_i = (a_0 * x_i + (a_1 * x_{i-1} + a_2 * x_{i-2})) + \\ +(b_1 * y_{i-1} + b_2 * y_{i-2}) \quad (7)$$

$$G = (x_0 + c_1 * x_1) + \left(c_4 * x_4 + (c_3 * x_3 + c_2 * x_2)\right) + \\ +\left(c_7 * x_7 + (c_6 * x_6 + c_5 * x_5)\right) \quad (8)$$

Из формул (6) – (8) нетрудно заметить идентичные по своей структуре рекуррентные цепочки. Что, в свою очередь, косвенно подтверждает высказанную нами гипотезу и позволяет построить специализированные рекуррентные преобразователи.

B. Построение рекуррентных цепочек

Осуществлять поиск и построение рекуррентных цепочек мы будем на примере алгоритма фильтра Баттерворта, т.к. его математическое описание содержит обе потенциальные цепочки. Как было отмечено ранее, данный фильтр является двухсекционным рекурсивным фильтром второго порядка. Это означает, что для фильтрации очередного отсчета необходимо хранить в памяти фильтра значения двух предыдущих отсчетов. Кроме того, для данного фильтра также характерно использование двух памятей отсчетов: входных и отфильтрованных (см. формулу (4).

В процессе поиска рекуррентных цепочек мы ориентировались на преобразованные уравнения (7), а также на уже существующую реализацию данного фильтра в капсульном представлении. Следует отметить, что в компоненте «Вычислитель» РОУ содержится набор пользовательских регистров, предназначенных для хранения 40-разрядных данных, в количестве 5 штук. Именно эти регистры и были использованы в существующей реализации для хранения памяти фильтра. Поэтому данный механизм также был заложен при построении рекуррентных цепочек. На рис. 2 представлена вычислительная схема, соответствующая формуле (7).

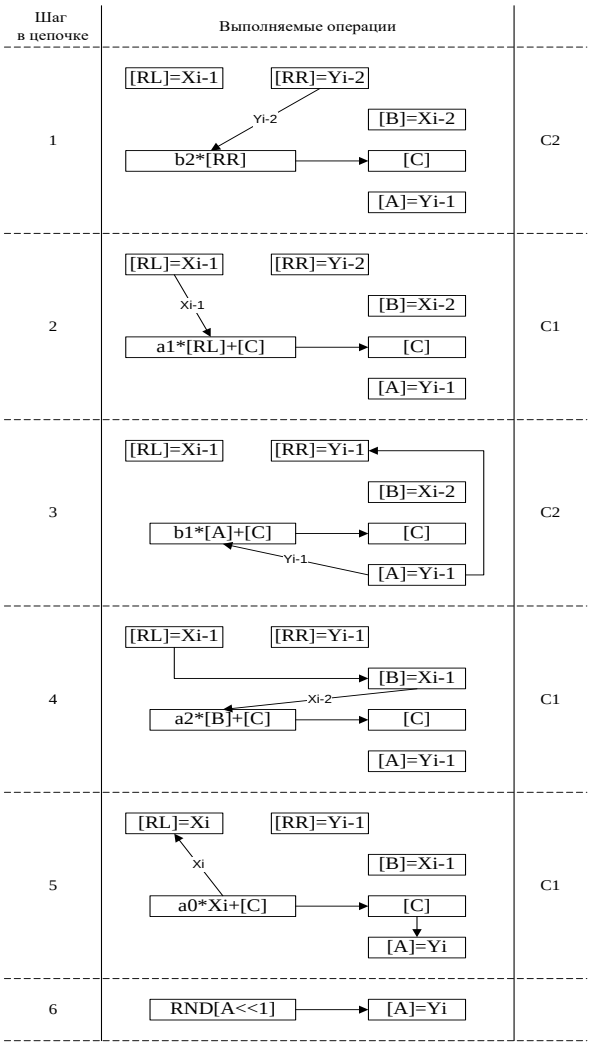

Рис. 2. Схема рекуррентного процесса Баттерворта

Данная схема вычислений была получена путем совмещения двух рекуррентных цепочек, обозначенных на рис. 2 C1 и C2. Кроме того, схема построена в предположении, что исходные данные уже были загружены в регистры соответствующим образом (этап не рекуррентных вычислений). Сравнительный анализ полученной схемы и исходной реализации показал наличие трех проблем: потеря производительности на шаге 2 (за счет выполнения лишней операции умножения с накоплением); проблема округления; длина обеих цепочек равна двум операциям (универсальный преобразователь позволяет строить цепочки длиной в три операции).

В исходной реализации использовалась суперскалярность компонента «Вычислитель», что позволило совместить операции, соответствующие шагам 1 и 2 на схеме. С другой стороны, операция округления является необходимой и выполняется всегда в конце процедуры накопления результата, но включать ее в рекуррентную цепочку не имеет смысла. Это означает, что для ряда алгоритмов необходимо неоднократно выполнить последовательность цепочек и затем округлить результат. Конкретно для данного алгоритма это могло бы привести к значительному увеличению объема капсулы, что противоречит основным целям настоящего исследования.

Анализ полученных результатов и обнаруженных проблем показал, что для достижения поставленных целей необходимо создать более длинные рекуррентные цепочки, предназначенные для конкретных классов задач ЦОС. В рамках настоящей статьи для рассматриваемых примеров мы создали три класса задач и соответствующих им специализированных рекуррентных цепочек:

- рекурсивный фильтр второго порядка с двумя памятями (фильтр Баттерворта);
- рекурсивный фильтр второго порядка с одной памятью (полосовой фильтр);
- взвешенный степенной ряд (ряды Тейлора).

На рис. 3 приводятся схемы их рекуррентных цепочек.

IV. Реализация специализированных преобразователей

A. Преобразователи в «Итераторе»

Чтобы реализовать разработанные цепочки в состав архитектуры необходимо ввести новые специализированные рекуррентные преобразователи. Возможным решением могло бы стать их введение в состав компонента «Вычислитель», который уже содержит блок «Преобразователь Тегов», реализующий универсальное рекуррентное преобразование. Однако перечисленные в разделе 1.2 проблемы говорят о неэффективности такого решения для построенных цепочек. Поэтому было принято решение о введении специализированного преобразователя на ранних ступенях конвейера.

Построенные схемы вычислений характеризуются активным использованием внутренних регистров компонента «Вычислитель». В текущей версии прототипа архитектуры управление данными, хранящимися в данных регистрах, осуществляется при помощи специальных управляющих операндов. В работах [6, 7] приводится описание специального блока «Итератор», предназначенного для генерирования потока управляющих операндов, а также о Преобразователе тегов Итератора (ПТ_И). Данный механизм позволяет, с одной стороны, снизить избыточность капсулы, а с другой – уже содержит ПТ_И. С учетом всего сказанного мы приняли решение доработать ПТ_И и сделать его полноценным специализированным Преобразователем тегов.

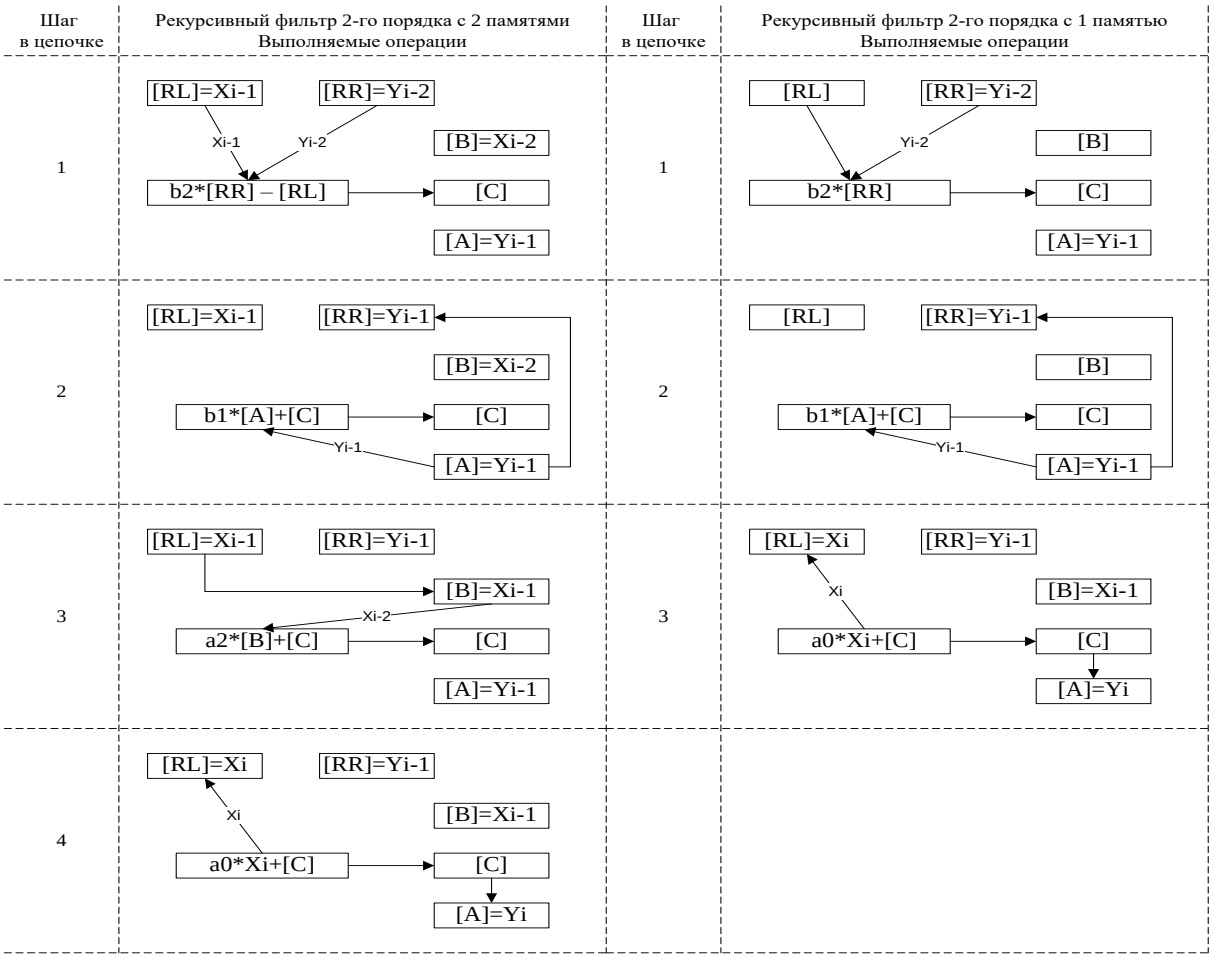

Рис. 3. Специализированные рекуррентные цепочки

Таким образом, схемы, представленные на рис. 2 и рис. 3, могут быть в полном объеме сформированы блоком «Итератор» (как это уже было сделано в текущих версиях реализации рассмотренных алгоритмов, но без активного использования принципов рекуррентности). Однако данное решение оказалось неполным. Несмотря на то, что Итератор мог бы формировать как рекуррентные, так и не рекуррентные цепочки вычислений, особенности его механизма функционирования привели бы к нежелательному прерыванию рекуррентных цепочек не рекуррентными операциями. Для решения проблемы совмещения рекуррентных и не рекуррентных вычислений (например операции округления) нам пришлось доработать архитектуру блока «Итератор». Внутренний буфер Итератора был разделен на две части, одна из которых содержит операнды, формирующие рекуррентные цепочки вычислений, а вторая – операнды, предназначенные для стандартного (уже существовавшего) механизма работы Итератора. При этом в случае конфликта (одновременной выдачи операндов из обоих буферов) приоритет отдается «стандартным» операндам, а рекуррентные цепочки, в свою очередь, ожидают своей очередности для выхода. Данный механизм позволил эффективно совместить рекуррентные и не рекуррентные вычисления без какой-либо потери производительности.

В рамках текущих исследований было принято решение о построении специализированных преобразователей в виде таблиц истинности. Доработанный ПТ_И позволяет переключаться между запрограммированными преобразователями для каждой конкретной цепочки, что повысило гибкость их использования и обеспечило простоту настройки. Также данное решение легко и недорого реализуемо в аппаратуре. Для организации эффективной работы с новыми средствами был доработан формат специализированных вспомогательных и управляющих операндов, предназначенных для настройки Итератора.

B. Результаты реализации фильтра Баттерворта

В качестве демонстрационного алгоритма мы реализовали фильтр Баттерворта с использованием специализированного рекуррентного преобразователя. За счет него нам удалось сократить число настроечных операндов Итератора с 10 до 4, а также удалить несколько вспомогательных операндов, требовавшихся в предыдущей версии. При этом общее время выполнения алгоритма осталось неизменным.

V. Заключение

В ходе работ по созданию специализированных рекуррентных преобразователей нам удалось получить следующие результаты:

- подтверждение работоспособности РОУ на базе библиотеки специализированных преобразователей;
- упрощение процесса программирования за счет использования библиотечных решений;
- потенциал для дальнейшего устранения избыточных полей, используемых универсальным преобразователем и возможность их рационального использования в будущем.

Таким образом, данный подход показал свою состоятельность. С учетом полученных результатов мы продолжим работы по наполнению библиотеки специализированных рекуррентных преобразователей для задач ЦОС. Полученная библиотека будет использована для реализации набора тестовых алгоритмов, аналогичных BTDIMark2000 Benchmark и анализа возможностей РОУ на всей предметной области.

Благодарности

Мы благодарим Рождественского Ю.В. за вклад в теоретические исследования и Дьяченко Ю.Г. за вклад в практическую реализацию.

Поддержка

Исследование выполнено за счет гранта Российского научного фонда (проект 19-11-00334).

Литература

[1] Yu.A. Stepchenkov, Yu.G. Diachenko, D.V. Khilko, V.S. Petrukhin. Recurrent data-flow architecture: features and realization problems // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2016. Proceedings / edited by A. Stempkovsky, Moscow, IPPM RAS, 2017. Part II. P. 52-58.

[2] Д.Н. Змеев, А.В. Климов, Н.Н. Левченко, А.С. Окунев, А.Л. Стемпковский. Потоковая модель вычислений как парадигма программирования будущего // Информатика и её применения. 2015. Т. 9. Вып. 4. С. 29-36.

[3] Клименко А.В. Основы естественного интеллекта. Рекуррентная теория самоорганизации. Версия 3. Ростов н/Д.: Изд-во Рост. ун-та, 1994. 304 с.

[4] Yu. Stepchenkov, Yu. Shikunov, N. Morozov, G. Orlov, D. Khilko. Hybrid Multi-Core Recurrent Architecture Approbation on FPGA // Proceedings of the 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). 2019 IEEE. P. 1705-1708.

[5] D. Khilko, Yu. Stepchenkov, D. Shikunov, Yu. Shikunov. Recurrent data-flow architecture: technical aspects of implementation and modeling results // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2016. Proceedings / edited by A. Stempkovsky, Moscow, IPPM RAS, 2017. Part II. P. 59-64.

[6] Yu. Shikunov, Yu. Stepchenkov, D. Khilko. Recurrent mechanism developments in the data-flow computer architecture // Proceedings of the 2018 IEEE Russia Section Young Researchers in Electrical and Electronic Engineering Conference (EIConRus). 2018 IEEE. P. 1413-1418.

[7] Y.A. Stepchenkov, D.V. Khilko, Y.I. Shikunov and G.A. Orlov. Iterator Component Development for Data Redundancy Solution in Data-Flow Architecture // 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). St. Petersburg and Moscow, Russia, 2020. Pp. 1869-1872.

[8] Хилько Д.В., Степченков Ю.А. Теоретические аспекты разработки методологии программирования рекуррентной архитектуры // Системы и средства информатики. 2013. Т. 23. №2. С. 133-153.

Specialized Tag Transformer for Recurrent Signal Processor

Y.A. Stepchenkov, D.V. Khilko, Y.I. Shikunov, G.A. Orlov

Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences

ia_ste@mail.ru, dhilko@yandex.ru, YIShikunov@yandex.ru, Orlov.jaja@gmail.com

***Abstract* — This paper covers the aspects of the recurrent principle in the digital signal processing domain. The existing implementation of the tag transformer in an operating unit was found lacking in practice. Therefore, we research the applicability of specialized recurrent tag transformers in a recurrent operating unit for the tasks of digital signal processing.**

We analyzed the existing recurrent implementation in an operating device based on the data-flow principles. The discovered deficiencies are described. The hypothesis for specialized recurrent chains efficiency is proposed. An analysis is done for a limited subset of digital signal processing algorithms with the aim of constructing specialized recurrent chains and their respective tag transformers.

The construction results of several specialized tag transformers are presented. The Butterworth demo filtering algorithm implementation using those transformers is presented as well.

The performance of the operating unit on the basis of a library of specialized transformers is confirmed. The programmability simplification is proposed via the introduction of the specialized transformers library. Potential architectural improvements through redundancy required for previous implementation removal is proposed.

***Keywords* — digital signal processing, data-flow, recurrency, tag transformer.**

SUPPORT

The research was funded by a grant from the Russian Science Foundation (Project № 19-11-00334)

REFERENCES

[1] Yu.A. Stepchenkov, Yu.G. Diachenko, D.V. Khilko, V.S. Petrukhin. Recurrent data-flow architecture: features and realization problems // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2016. Proceedings / edited by A. Stempkovsky, Moscow, IPPM RAS, 2017. Part II. P. 52-58.

[2] D.N. Zmeev, A.V. Klimov, N.N. Levchenko, A.S. Okunev, A.L. Stempkovsky. Potokovaya model vichisleniy kak paradigma programmirovanya buduschego (Data-flow computational model as a future programming paradigm), Informatika i ee primenenia. 2015. Vol. 9. № 4. P. 29-36 (in Russian).

[3] Klimenko A. V.K 49 Osnovi estestvennogo intellekta. Rekurrentnaya teoriya samoorganizacii. Versiya 3. (Basics of natural intelligence. The recurrent theory of self-organization. Version 3) Rostov n/D: Izd-vo Rost. un-ta, 1994. 304 c. (In Russian).

[4] Yu. Stepchenkov, Yu. Shikunov, N. Morozov, G. Orlov, D. Khilko. Hybrid Multi-Core Recurrent Architecture Approbation on FPGA // Proceedings of the 2019 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). 2019 IEEE. P. 1705-1708.

[5] D. Khilko, Yu. Stepchenkov, D. Shikunov, Yu. Shikunov. Recurrent data-flow architecture: technical aspects of implementation and modeling results // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2016. Proceedings / edited by A. Stempkovsky, Moscow, IPPM RAS, 2017. Part II. P. 59-64.

[6] Yu. Shikunov, Yu. Stepchenkov, D. Khilko. Recurrent mechanism developments in the data-flow computer architecture // Proceedings of the 2018 IEEE Russia Section Young Researchers in Electrical and Electronic Engineering Conference (EIConRus). 2018 IEEE. P. 1413-1418.

[7] Yu.A. Stepchenkov, D.V. Khilko, Yu.I. Shikunov and G.A. Orlov. Iterator Component Development for Data Redundancy Solution in Data-Flow Architecture // 2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus). St. Petersburg and Moscow, Russia, 2020. Pp. 1869-1872.

[8] Khilko D.V., Stepchenkov Yu.A. Teoreticheskie aspekti razrabotki metodologii programmirovania recurrentnoi arhitecturi (Theoretical Aspects of Recurrent Architecture Programming Methodology Development) // Sistemy i sredstva informatiki. 2013. Vol. 23. № 2. P. 133-156 (in Russian).