УДК 004.272.44

Рекуррентная потоковая архитектура: технические аспекты реализации и результаты моделирования

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

Институт проблем информатики Федерального исследовательского центра "Информатика и управление" Российской академии наук

dhilko@yadenx.ru, ystepchenkov@ipiran.ru, shikunovdima@gmail.com, yishikunov@yandex.ru

***Аннотация* — В настоящей статье рассматриваются методы и особенности реализации прототипа архитектуры, основанной на новой рекуррентно-потоковой парадигме вычислений и предназначенной для решения задач цифровой обработки сигналов. Приводится демонстрация ключевых принципов и технических решений, реализованных в новой архитектуре, на примере задачи быстрого преобразования Фурье, а также оценка быстродействия этой задачи относительно ее решения на процессорах традиционной одноядерной и специализированной потоковой многоядерной архитектур. Также приводятся сравнительные оценки эффективности реализации алгоритмов распознавания изолированных слов в среде рекуррентной архитектуры по отношению к фоннеймановской одноядерной.**

***Ключевые слова* — потоковая архитектура, рекуррентность, цифровая обработка сигналов, быстрое преобразование Фурье.**

I. Введение

На сегодняшний день все более актуальными становятся задачи цифровой обработки сигналов (ЦОС), для решения которых применяются специализированные цифровые сигнальные процессоры (ЦСП). Одним из возможных вариантов реализации ЦСП является использование архитектуры потока данных, в силу того, что структура алгоритмов ЦОС и парадигма потока данных обладают высокой синергией и хорошо «подходят» друг другу. Помимо очевидных преимуществ применения потоковых архитектур для задач ЦОС, существует целый ряд проблем, препятствующих их успешному коммерческому внедрению [1-3]. Основными препятствиями являются сложности реализации конвейерной обработки данных, рекурсивных вычислений, циклических процедур, взаимодействия с константами и многократное использование программного кода.

В процессе поиска приемлемого решения перечисленных проблем в отделе «Перспективных архитектур компьютерных систем» Института проблем информатики РАН Федерального исследовательского центра «Информатика и управление» РАН (ИПИ РАН) была разработана концепция новой многоядерной потоковой рекуррентной архитектуры (МПРА) для решения задач ЦОС с ограниченной параллельностью. Основные аспекты архитектуры МПРА описаны в работах [4-5]. Наиболее важными свойствами новой архитектуры являются: *самодостаточные* данные и *рекуррентность*.

Самодостаточными называются данные, которые включают в свою структуру как непосредственно данные, так и инструкции (теги) для их обработки. Другими словами, два традиционных потока – данных и инструкций – объединяются в единый самодостаточный поток. *Рекуррентностью* называется свойство динамического развития траектории вычислительного процесса путем вычисления новых значений теговых полей самодостаточных данных с помощью преобразования тегов.

В данной статье рассматриваются технические аспекты реализации четырехъядерного прототипа МПРА и результаты его экспериментальной апробации для задач ЦОС. В состав прототипа были включены средства поддержки параллелизма на различных уровнях (потоков, команд и операций отдельных команд), а также средства аппаратной поддержки ЦОС (в том числе специализированные инструкции и различные виды памяти констант). В качестве демонстрационной задачи была выбрана задача распознавания изолированных слов (РИС), а также эталонный алгоритм из класса ЦОС – быстрое преобразование Фурье (БПФ).

Для большинства алгоритмов из множества задачи РИС, а также для БПФ, была осуществлена программная реализация и аппаратно-программное моделирование. Приведены сравнительные результаты моделирования данных алгоритмов на архитектуре МПРА по параметру времени вычислений (в количестве логических шагов) с аналогичными результатами для одноядерного ЦСП традиционной архитектуры. Полученные значения коэффициентов ускорения варьируются в диапазоне ***от 2 до 17***, что позволяет оценить потенциальное быстродействие ЦСП, спроектированного на основе МПРА.

II. Описание прототипа многоядерной потоковой рекуррентной архитектуры

В процессе поиска приемлемой реализации идей, заложенных в МПРА, было установлено, что наиболее целесообразным вариантом реализации рекуррентного

обработчика сигналов (РОС) на основе ПЛИС является гибридная двухуровневая архитектура РОС (ГАРОС) с ведущим фон-неймановским процессором на управляющем (верхнем) уровне (УУ) и рядом потоковых процессоров на нижнем уровне – рекуррентном операционном устройстве (РОУ) [6]. При этом взаимодействие между УУ и РОУ осуществляется посредством специальной двухпортовой буферной памяти (БП), обеспечивающей одновременный доступ на чтение и запись как для УУ, так и для РОУ. Конфликты доступа к портам разрешаются в пользу РОУ. Суммарный объем БП составляет 8196 64-разрядных операндов, а также совокупность битов наличия операндов в ячейках.

Ключевым компонентом ГАРОС является РОУ, для которого были созданы как программная имитационная, так и аппаратная VHDL модели [7, 8]. На рис. 1 приведена архитектура РОУ.

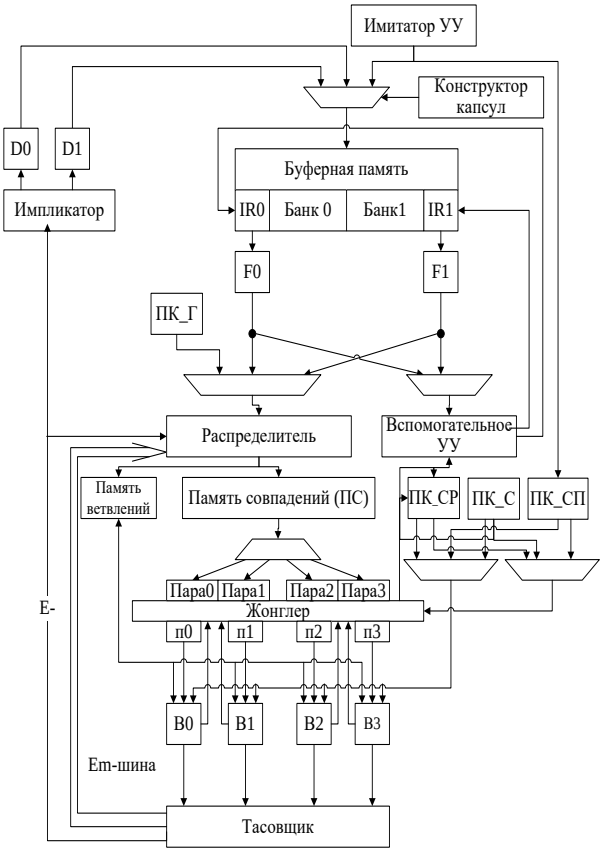

Рис. 1. Архитектура РОУ

Здесь: IR0, IR1 – индексные регистры в БП; F0, F1 – буферные регистры на входе РОУ; D0, D1 – буферные регистры на выходе РОУ; ПК_Г – память констант глобальная; ПК_С – память констант секционная; ПК_СР – память констант секционная регистровая; ПК_СП – память констант секционная подгружаемая; В – Вычислитель (в текущей реализации в количестве 4 единиц), который включает в себя компонент ПТ – преобразователь тегов.

Для большинства потоковых архитектур характерно использование ресурсоемкого компонента, осуществляющего формирование совпадающих пар тегированных данных, – ассоциативной памяти. В текущей реализации РОУ универсальная ассоциативная память была заменена на прямо адресуемую ПС небольшого объема (16 ячеек в данной реализации) с дополнительным битом наличия операнда в адресуемой ячейке. Принятое решение накладывает жесткие ограничения на последовательность входных данных для РОУ, т.к. не поддерживает контекста шага вычислительного процесса. В то же время структура данных компонентов требует значительно меньших накладных расходов, имеет более высокую энергоэффективность и быстродействие по сравнению с ассоциативной памятью.

Самодостаточные данные в текущей версии реализации МПРА представляются в виде операндов специализированных типов: содержательные (несут в себе данные), вспомогательные (глобальные опции конфигурации) и управляющие (тонкая конфигурация или специальное применение). В рамках РОУ исполняемая программа представляет собой последовательность операндов, названная разработчиками капсулой. Буферная память хранит шаблон капсулы, который заполняется данными со стороны УУ. Рис. 2 содержит листинг фрагмента капсулы вычисления БПФ.

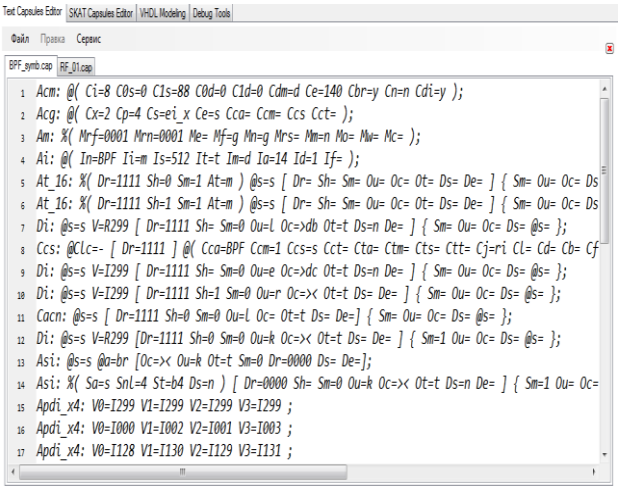

Рис. 2. Листинг фрагмента капсулы БПФ

Свойство рекуррентности представлено в текущей версии реализации МПРА при помощи универсального Преобразователя тегов, осуществляющего логический сдвиг вправо комплекта значений теговых полей на каждом шаге вычислительного процесса. Реконфигурация ПТ на выполнение иной функции преобразований достаточно длительный по времени процесс (как показано в работе [3] для динамически реконфигурируемых ПЛИС). По мнению разработчиков, использование универсального ПТ является достаточным для решения задач из выбранного класса ЦОС. В работе [9] была доказана сходимость рекуррентного вычислительного процесса (универсальный ПТ является частным случаем). Техническая реализация подобного механизма функционирования ПТ потребовала ввести избыточные теговые поля в операнды, выделяемые разделителями ″{″ и ″}″ (рис. 2).

Вычислительный процесс в РОУ организован следующим образом:

1) компонент Распределитель осуществляет выборку операндов из БП и их рассылку в соответствующие потоки (секции), заданные в теговых полях;

2) компонент Память совпадений (ПС) осуществляет сравнение теговых полей и формирование пар совпавших операндов;

3) компонент Жонглер осуществляет разделение единого потока самодостаточных данных на два потока – данных и инструкций, а также их необходимое распределение по входам Вычислителей;

4) компонент Вычислитель осуществляет вычисление результата и преобразование тегов;

5) компонент Тасовщик осуществляет пересылку полученных результатов между параллельными потоками, а также в Импликатор;

6) компонент Импликатор осуществляет запись выходных данных в БП.

Подобная организация обработки инструкций в устоявшемся вычислительном процессе будет осуществляться за 2,5 шага. Таким образом, можно сделать вывод, что двухстадийного конвейера должно быть достаточно для наиболее эффективной реализации заложенного в МПРА механизма. Однако реальные испытания показали, что в случае объединения компонентов Распределитель, ПС и Жонглер в одну ступень конвейера, ее производительность значительно ниже производительности ступени, на которой будут расположены Вычислитель и Импликатор. Вследствие этого компонент Распределитель был вынесен на отдельную (третью) ступень конвейера.

Данное решение позволило существенно расширить функциональные возможности компонента Жонглер и нивелировать ограничения, накладываемые отказом от ассоциативной памяти. Это достигается за счет функциональности разрешения противоречий между полями совпавших операндов пары, обеспечения взаимодействия с Памятью ветвлений и различными подвидами Памяти констант, а также за счет разделения и подготовки данных и инструкций для суперскалярной обработки на Вычислителе.

III. Архитектурные особенности ГАРОС для решения проблем потоковых ЦОС-архитектур

A. Суперскалярная организация Вычислителя. Поддержка рекурсивных алгоритмов

Общей проблемой параллельных и потоковых систем является реализация рекурсивных алгоритмов. Для таких алгоритмов характерна высокая зависимость по данным. В то же время большинство алгоритмов ЦОС используют рекурсию в том или ином виде. Поэтому одной из основных задач, которую требуется решить при разработке ЦСП на основе потоковой архитектуры, является поддержка рекурсивных вычислений.

Основным механизмом поддержки рекурсивных вычислений в РОУ является рекуррентная свертка, осуществляемая на этапе программирования, и рекуррентная развертка (преобразование тегов), осуществляемая ПТ. Состояние рекурсивного вычислительного процесса сохраняется в функциональных полях как хранящихся в ПС, так и промежуточных операндов. Данный подход позволяет реализовать любую необходимую глубину рекурсии. Кроме того, Вычислитель также содержит ряд дополнительных механизмов поддержки рекурсивных вычислений.

В состав компонента Вычислитель входят: набор буферных регистров L, R, B, C и аккумулятор A; блок аппаратного умножения (У); АЛУ16 (в данной реализации 16-разрядное) и 40-разрядное арифметическое устройство (АУ40); логика сдвига и округления (BS&R); блоки поддержки обработки условных переходов. Помимо входных шин L- и R-, каждый из буферных регистров и аккумулятор, а также шина с выхода памяти констант, могут быть источниками данных для бинарных операций. Кроме того, каждый из блоков – У, АЛУ16, АУ40, BS&R – может функционировать одновременно в случае обеспечения необходимыми данными. На рис. 3 приводится детализированная структура компонента Вычислитель.

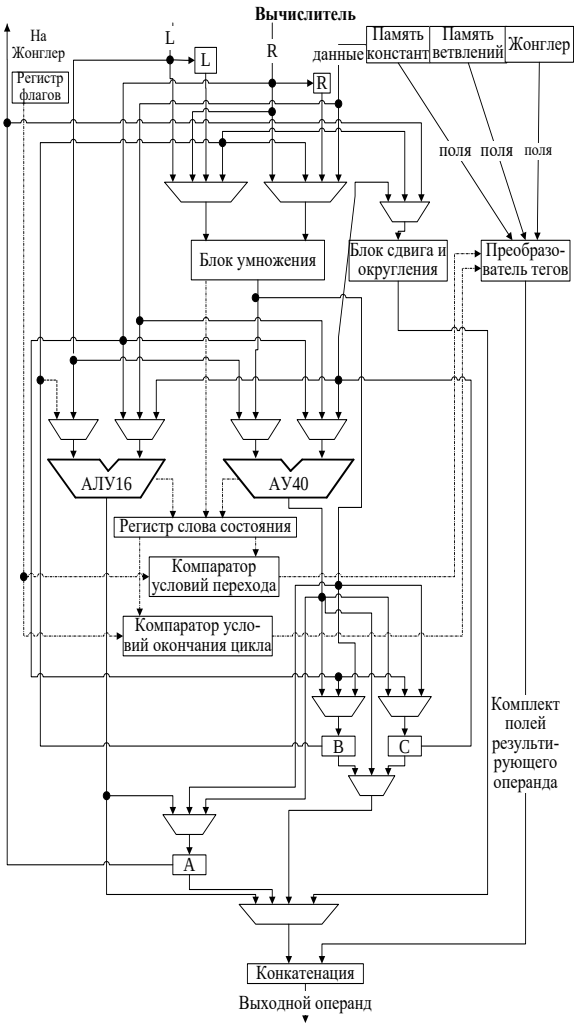

Рис. 3. Структурная схема Вычислителя

Таким образом, Вычислитель имеет суперскалярную архитектуру. В настоящей реализации РОУ поддерживаются операций умножения с накоплением, а также два суперскалярных режима, покрывающих большое многообразие возможных сочетаний пар (в редких случаях троек) операций, которые могут быть выполнены в рамках контекста одного вычислительного шага. Это, в свою очередь, позволяет обеспечить эффективный переход между очередными шагами рекурсивного алгоритма без каких либо потерь в точности или дополнительных логических шагов, необходимых для обеспечения поддержки сильной зависимости по данным.

B. Механизмы работы с константными данными

Большинство алгоритмов ЦОС в том или ином виде используют константы. В этой связи все современные ЦСП используют специализированные типы памяти, которые называются памятью констант. Предлагаемая архитектура в этом смысле также не является исключением. Из рис. 1 видно, что в состав ГАРОС входит четыре различных типа памяти констант, каждая из которых может хранить последовательности констант и комплекты тегов в виде самодостаточных данных.

Каждый из типов памяти констант имеет различные области доступности и механизм функционирования. Далее рассматривается назначение и механизм работы каждого типа памяти констант.

1) ПК_Г – глобальная память констант, доступная на уровне Распределителя. Так же, как и Распределитель, является централизованным ресурсом для всех секций. Используется для загрузки констант, которые необходимо передать по тем же правилам, что и входные данные, поступающие из БП. Такой подход позволяет в требуемые моменты времени загружать полноценные операнды в Распределитель для их дальнейшего использования Жонглером при разрешении противоречий в тегах компонентов пары, поступивших из ПС.

2) ПК_С – секционная память констант, доступная напрямую Вычислителю. Количество модулей этого типа памяти равно количеству секций. Наиболее часто используемый тип памяти констант. Используется для выполнения суперскалярных операций с тремя или более источниками входных данных. При необходимости может также предоставлять набор полей для ПТ.

3) ПК_СР – секционная регистровая память констант, доступная Жонглеру для записи и Вычислителю для чтения. Имеет фиксированный небольшой объем и используется в том случае, если количество используемых констант мало. Загружается специальными управляющими операндами и используется аналогично ПК_С.

4) ПК_СП – секционная подгружаемая память констант, доступная УУ для записи и Вычислителю для чтения. Используется в том случае, если количество констант слишком велико, чтобы хранить их в ПК_С. Примером может служить алгоритм Витерби, который осуществляет распознавание на основе поиска экстремума вероятности, сравнивая ее с параметрами моделей распознаваемых слов, хранящихся в базе. Очевидно, что каждый такой параметр является константой, но их количество исчисляется тысячами. ПК_СП имеет небольшой объем и постоянно обновляется в ходе развития вычислительного процесса.

C. Поддержка циклических процедур в РОУ

Многие алгоритмы ЦОС включают в себя накопление некоторых данных или повторения одного и того же участка программы в течение заданного количества итераций и затем меняют свое поведение. Поэтому в состав ГАРОС были включены специальные средства поддержки циклических операций.

В состав Распределителя был введен компонент Итератор, назначение которого заключается в формировании специализированных управляющих операндов с заданной задержкой. Данные операнды позволяют привлекать значение аккумулятора А в качестве второго компонента пары, а также содержат полный комплект тегов. Эти поля могут быть использованы как для разрешения противоречий на Жонглере, так и для обхода ограничений универсального преобразователя тегов. Другими словами, применяя подобные операнды, можно организовать циклические процедуры с заданным количеством итераций, которые не требуют привлечения данных из капсулы.

Другим ресурсом, обеспечивающим поддержку циклических процедур в РОУ, является блок, заданный на рисунке 4 как «компаратор условий окончания цикла». Этот блок включает в себя настраиваемый счетчик, а также механизм выполнения перехода по достижению счетчиком значения, равного 0. При этом переход в терминах РОУ – это выборка ПТ комплекта тегов для формирования результирующего операнда не с выхода Жонглера, а с выхода Памяти ветвлений. Проверка условия окончания цикла в РОУ может быть выполнена двумя способами: «длинным» и «коротким».

Под «коротким» понимается способ, при котором проверка счетчика итераций и при необходимости переход осуществляются на том же шаге, где происходит модификация счетчика. Данный эффект достигается путем специального режима функционирования Вычислителя. Причем информация об активации этого режима содержится в теговом поле *Ot* (см., например, операнд №7 на рис. 2). Подобная организация проверки счетчика и перехода позволяет максимально эффективно осуществлять циклические процедуры, тела которых невелики (порядка 5-10 вычислительных шагов). Но «короткий» способ имеет также и недостаток, который заключается в том, что значение поля *Ot* является константой (не подвергается преобразованию в ПТ). Данный эффект может привести к возникновению некорректных теговых значений у последующих операндов, что необходимо отслеживать дополнительно.

Под «длинным» понимается способ, при котором инициация проверки значения счетчика итераций на равенство 0 осуществляется с помощью специального

управляющего операнда – контролера циклов. При такой организации проверки тратится дополнительный вычислительный шаг, вследствие чего переход и называется «длинным». Однако подобный подход имеет и свои преимущества. В случае если тело циклической процедуры включает в себя достаточно большое количество вычислительных шагов, то «длинный» переход может быть полезен. Во-первых, он не требует модификации полей *Ot*, а во-вторых – позволяет сформировать единую процедуру выхода из цикла сразу для нескольких секций (например, когда тела циклических процедур одинаковы во всех секциях). Следует также отметить, что организация условных переходов реализована идентичным образом.

D. Многократное повторение программ и повторное использование выходных данных

Часть алгоритмов ЦОС подразумевает вычисление некоторого параметра и его последующее уточнение за некоторое количество итераций. Для алгоритмов такого рода может понадобиться возможность многократного использования программного кода (в случае РОУ – многократного исполнения капсулы), а также механизм перезаписи выходных данных для повторного использования на последующих итерациях.

В рамках РОУ реализован механизм обработки выходных данных в двух режимах. В первом режиме осуществляется накопление только выходных данных, которые должны быть переданы УУ для дальнейшей обработки. Такие данные помещаются в выходной раздел капсулы. Во втором режиме осуществляется перезапись выходных данных в шаблон капсулы по заданным адресам. Таким образом, осуществляется подготовка к следующей итерации капсулы.

Многократное исполнение реализовано в РОУ при помощи Вспомогательного устройства управления (ВУУ), которое отслеживает границы того фрагмента капсулы, который необходимо выполнить несколько раз. Также ВУУ управляет адресацией в БП, что позволяет манипулировать потоком данных, поступающим в Распределитель. Наиболее полно данный режим удалось использовать в рамках РОУ при реализации БПФ. В частности, в РОУ реализована аппаратная поддержка БПФ, в том числе бит-реверсная адресация и примитив 4-х цикловой операции «бабочка».

IV. Демонстрация особенностей ГАРОС на примере быстрого преобразования Фурье

Для демонстрации архитектурных возможностей ГАРОС был выбран алгоритм БПФ Radix 2 (256 отсчетов), т.к. он является эталонным и используется в большинстве задач ЦОС. Суть задачи состоит в вычислении 1024 типовых операций «бабочка» (по 128 операций за 8 ступеней). Эффективность реализации БПФ рассчитывалась на основе показателя времени вычисления преобразования (количество логических шагов). Осуществлялось сравнение показателей скорости вычислений, полученных при реализации БПФ в ГАРОС и в одноядерном микроконтроллере dsPIC30F [10].

Согласно работе [10] библиотечная реализация БПФ на dsPIC30F вычисляется за 476 мкс при производительности в 40 MIPS, т.е. примерно за 19000 инструкций. Причем в данное множество инструкций уже включены все операции перезаписи промежуточных данных.

Для ГАРОС реализации БПФ можно пренебречь затратами на перезапись промежуточных данных, т.к. эти операции обеспечиваются механизмами аппаратной поддержки БПФ и выполняются параллельно основным вычислениям. На рис. 2 приводится фрагмент капсулы БПФ, которая преимущественно состоит из упакованных операндов, что позволяет уменьшить объем капсулы практически в 4 раза. На рис. 4 приводится фрагмент граф-капсулы вычисления БПФ средствами ГАРОС. Схема приведена для двух секции, т.к. в параллельных секциях процесс проходит полностью идентично.

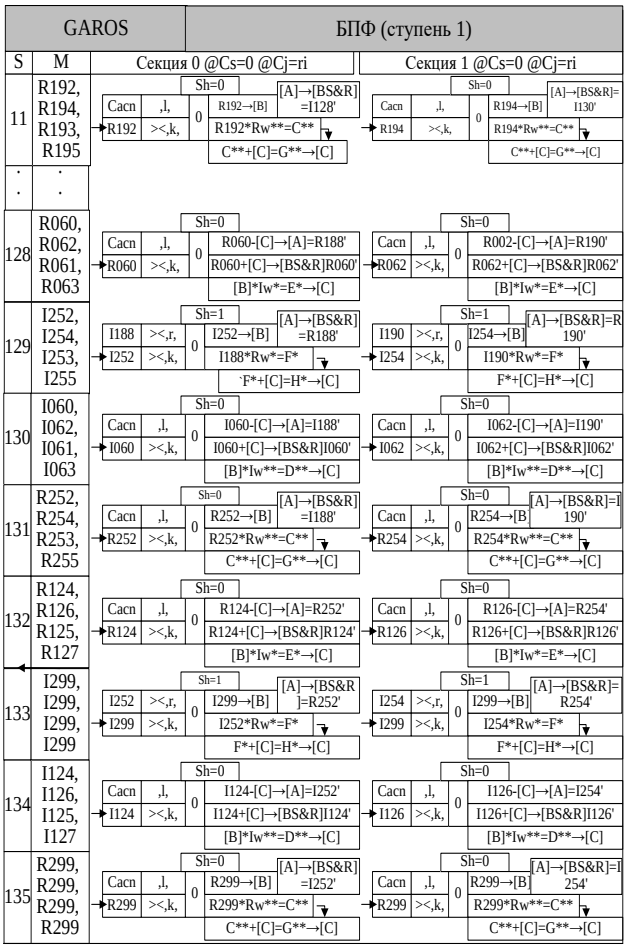

Рис. 4. Фрагмент граф-капсулы БПФ

В каждой секции вычисляется четырехцикловая операция «бабочка», поддерживаемая на аппаратном уровне. При этом за 4 вычислительных шага вычисляется 4 «бабочки». Следовательно, с учетом затрат на преконфигурацию за 130-135 шагов вычисляется 128 «бабочек», что суммарно за 8 ступеней алгоритма дает

1024 бабочки за приблизительно 1100 шагов. Достигается подобный результат за счет использования суперскалярности вычислительных блоков РОУ, памяти констант, а также многократного исполнения капсулы (8 ступеней) и перезаписи промежуточных данных в шаблон капсулы. Согласно рис. 4 на каждом шаге в суперскалярном режиме вычисляются значения сумм, произведений с данными из памяти констант, осуществляется округление и запись данных во внутренние регистры. Тем самым задействуются все технические решения, описанные ранее.

Выбор микроконтроллера dsPIC30F в качестве эталона для сравнения параметров производительности был осуществлен осознанно: именно для этого микроконтроллера сотрудниками ИПИ РАН была осуществлена коммерческая реализация задачи распознавания слов для компании Microchip (в работе [10] это указано в виде библиотеки распознавания). Поэтому полученные оценки являются объективными, т.к. опираются на целостные реальные результаты в области РИС.

V. Результаты аппаратно-программного моделирования ГАРОС

Для экспериментальной апробации ГАРОС была выбрана задача распознавания изолированных слов (РИС), реализованная в рамках ранних проектов на микроконтроллере dsPIC30F для компании Microchip, а также эталонный алгоритм БПФ Radix 2 для 256 отсчетов. В работе [9] приводится специализированная методология разработки капсул для ГАРОС. В результате ее применения удалось реализовать для РОУ практически все алгоритмы РИС. С помощью аппаратно-программной платформы GAROS IDE [11] было осуществлено моделирование каждого из реализованных алгоритмов и получены оценки скорости их выполнения. Следует отметить, что приведенные далее оценки коэффициента ускорения рассчитаны с учетом следующих ограничений:

1) микроконтроллер dsPIC30F основан на фоннеймановской архитектуре;

2) ГАРОС имеет 4 секции и вычислительных ядра соответственно, а dsPIC30F – одно ядро;

3) вычислители ГАРОС имеют значительно более широкие возможности применения суперскалярных режимов, чем dsPIC30F, исполняющий ограниченное множество команд, для которых определена подобная семантика;

4) в процессе исполнения капсулы предполагается, что все данные уже загружены или успевают загружаться управляющим уровнем по ходу вычисления капсулы. В этой связи при расчетах не учитывались временные затраты на подготовку данных как в dsPIC30F, так и в ГАРОС.

Кроме того, некоторые капсулы были реализованы в двух вариантах с целью нахождения наиболее производительного решения. Вариант 1 означает, что алгоритм реализован с использованием всех четырех ВУ, а вариант 2 – что алгоритм реализован для четырех комплектов входных данных с использованием одного ВУ для каждого комплекта. Результаты сравнения реализации демонстрационных алгоритмов представлены в табл. 1. Некоторые из полученных результатов являются предварительными (помечены символом «~»), в силу того, что средства аппаратно-программного моделирования требуют доработки, что, в свою очередь, может потребовать доработки соответствующих капсул.

Таблица 1

Результаты реализации демонстрационных алгоритмов

Название алгоритма	Кол-во шагов для dsPIC30F	Кол-во шагов для РОУ	Коэффициент ускорения
БПФ2_256	**~19000**	**~1100**	**~17,2**
Баттеруорт (одна секция)	679	288	2,36
Баттеруорт*) (две секции)	2760	682	4,05
Полосовой фильтр (одна полоса)	1428	442	3,23
Натуральный логарифм (вариант 1)	36	19	1,89
Натуральный логарифм x4 (вариант 2)	36*4	26	1,38*4
RASTA фильтр	153	45	3,4
Экспоненцирование (вариант 1)	32	13	2,46
Экспоненцирование x4 (вариант 2)	32*4	20	1,6*4
Косинусное ИДПФ	36	17	2,12
Рекурсия Дурбина-Скурра	~440	~72	~6,1
PLP параметры	144	32	4,5
PLP параметры*)	144	22	6,55
Витерби (расчет решетки для текущего N)	91*N-143	99*N	$(1-\frac{8*N+143}{99*N})*4$
* модифицированная версия ГАРОС N – количество наблюдений в векторе наблюдений (N > 5)			

VI. Заключение

Принятые в процессе реализации прототипа МПРА технические решения позволили, с одной стороны, решить часть проблем, свойственных потоковым архитектурам, а с другой стороны – получить существенный прирост быстродействия по сравнению с традиционным ЦСП. Наиболее важным предварительным результатом является оценка эффективности аппаратной поддержки алгоритма БПФ. Комбинирование различных механизмов РОУ позволило достичь коэффициента ускорения порядка ***17*** по сравнению с одноядерным традиционным ЦСП. Кроме того, оценки коэффициентов ускорения большинства алгоритмов РИС лежат в диапазоне ***3 – 6***, что также является неплохим результатом для четырехъядерного исполнения.

Исследование капсул, реализующих различные алгоритмы из класса ЦОС, на предмет степени задействования вычислительных ресурсов позволило получить среднюю оценку покрытия порядка 60-70%. Тем не менее оценка степени задействования вычислительных ресурсов для алгоритмов – БПФ, Косинусное ИДПФ, Витерби, Натуральный логарифм и Экспонента (вариант 2) – приближается к 100%. Полученный результат свидетельствует о высокой эффективности принятых в рамках прототипа МПРА технических решений.

Также планируется реализовать прототип МПРА для 32-разрядных чисел с плавающей точкой на основе самосинхронной элементной базы, что позволит повысить его энергоэффективность и отказоустойчивость. У коллектива исполнителей имеется опыт разработки аппаратных самосинхронных сопроцессоров с плавающей точкой для 32/64-разрядных чисел: делителя и устройства извлечения квадратного корня [12]; устройства умножения с накоплением и устройства совмещенного умножения двух входных операндов и последующего сложения (вычитания) с третьим входным операндом с однократным округлением [13].

Тем не менее полученные результаты свидетельствуют о необходимости дальнейшего развития и совершенствования новой архитектуры для повышения среднего покрытия вычислительных ресурсов и конечной производительности МПРА.

Благодарности

В заключение хотим выразить свои благодарности Морозову Н.В., Дьяченко Ю.Г. и Рождественскому Ю.В. за неоценимый вклад в разработку аппаратной модели МПРА.

Поддержка

Исследование выполнено при частичной финансовой поддержке по подпрограмме № 4 отделения ОНИТ РАН на 2016 г. (проект 0063-2015-0016 III.3).

Литература

[1] E. A. Lee and J. C. Bier. Architectures for Statically Scheduled Dataflow // Parallel Algorithms and Architectures for DSP Applications / edited by Magdy A. Bayoumi. Dordrecht. Kluwer Academic Publishers, 1991. P. 159-190.

[2] V.P. Srini. DFS-SuperMPx: Low-cost Parallel Processing System for Machine Vision and Image Processing // Proc. Third International Conference “Parallel Computing Technologies”, PaCT-95. St. Petersburg, 1995. Vol. 3. P. 356-369.

[3] S. Voigt, M. Baesler, T. Teufel. Dynamically reconfigurable dataflow architecture for high-performance digital signal processing // Journal of Systems Architecture, 2010, no. 56. P. 561-576.

[4] Yu. Shikunov, D. Khilko, Yu. Stepchenkov. Hardware and Software Modelling and Testing of Non-Conventional Data-Flow Architecture // Proceedings of the 2016 IEEE North West Russia Section Young Researchers in Electrical and Electronic Engineering Conference (ElConRusNW), 2016. P. 360-364.

[5] Степченков Ю.А., Волчек В.Н., Петрухин В.С., Прокофьев А.А. Механизмы обеспечения поддержки алгоритмов цифровой обработки речевых сигналов в рекуррентном обработчике сигналов // Системы и средства информатики – М.: ТОРУС ПРЕСС, Вып.20, № 1, 2010. С. 31-47.

[6] Волчек В.Н., Степченков Ю.А., Петрухин В.С., Прокофьев А.А., Зеленов Р.А. Цифровой сигнальный процессор с нетрадиционной рекуррентной потоковой архитектурой // Проблемы разработки перспективных микро- и наноэлектронных систем – 2010. Сборник трудов / под общ. ред. академика А.Л. Стемпковского. М.: ИППМ РАН, 2010. С. 412-417.

[7] Хилько Д.В., Степченков Ю.А., Дьяченко Ю.Г., Шикунов Ю.И., Морозов Н.В. Аппаратно-программное моделирование и тестирование рекуррентного операционного устройства // Системы и средства информатики, 2015 Т. 25 №4. С. 78-90.

[8] Хилько Д.В., Шикунов Ю.И., Степченков Ю.А. Особенности программной реализации имитационной модели потоковой рекуррентной архитектуры // Труды Второй молодежной научной конференции «Задачи современной информатики» – М.: ФИЦ ИУ РАН, 2015. С. 220-227.

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

[10] URL: http://microchip.com.ru/Support/Download/13_64.pdf (дата обращения: 04.04.2016).

[11] Хилько Д.В., Степченков Ю.А., Шикунов Ю.И. Инструментальная среда проектирования ПО для гибридной архитектуры рекуррентного обработчика сигналов (GAROS IDE). Свидетельство о государственной регистрации программы для ЭВМ № 2015614004 от 01.04.15.

[12] Y. Stepchenkov, Y. Diachenko, V. Zakharov, Y. Rogdestvenski, N. Morozov, and D. Stepchenkov, Self-Timed Computing Device for High-Reliable Applications / Proc. International Workshop on power and timing modeling, optimization and simulation (PATMOS'2009), Delft, Netherlands, 2009. P. 276–285.

[13] Соколов И.А., Степченков Ю.А., Рождественский Ю.В., Дьяченко Ю.Г. Самосинхронное устройство умножения-сложения гигафлопсного класса: методологические аспекты // Проблемы разработки перспективных микро- и наноэлектронных систем - 2014. Сб. трудов / под общ. ред. академика РАН А.Л. Стемпковского. М.: ИППМ РАН, 2014. Ч. IV. С. 51-56.

Recurrent data-flow architecture: technical aspects of implementation and modeling results

D.V. Khilko, Yu. A. Stepchenkov, D. I. Shikunov, Yu. I. Shikunov

The Institute of Informatics Problems, Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences

dhilko@yadenx.ru, ystepchenkov@ipiran.ru, shikunovdima@gmail.com, yishikunov@yandex.ru

***Keywords* — data-flow architecture, recurrence, digital signal processing, fast Fourier transform.**

EXTENDED ABSTRACT

New recurrent data-flow computational model and its possible implementation are introduced. The architectural prototype based on this model is positioned as DSP-type hardware.

Research on possible implementations of the concept of multi-core dataflow recurrent architecture has lead to development of a recurrent signal processor incorporating the idea of two-level hybrid interaction between high-level (von Neumann) control unit and a low-level dataflow multi-core operational unit. Solutions to the key technical problems of dataflow DSP computing are demonstrated using the classical algorithm of fast Fourier transform (FFT) as an example. Super-scalar modes, recursion, manipulating constants, cycled operations and repetitive input data usage have been implemented.

In order to show the perspectives of new architecture, the results of comparative analysis of number of logical steps needed to complete certain DSP algorithms are presented. Acceleration factors for the comparison of 4-core hybrid architecture recurrent signal processor to classic single-core microprocessor (dsPIC30F) range from 3 to 6 for word recognition algorithms and is estimated to be as high as 17 for FFT algorithm.

SUPPORT

The study was partially supported by the 2016's subprogram no. 4 of ONIT RAS department. (project 0063-2015-0016 III.3).

REFERENCES

[1] E. A. Lee and J. C. Bier. Architectures for Statically Scheduled Dataflow // Parallel Algorithms and Architectures for DSP Applications / edited by Magdy A. Bayoumi. Dordrecht. Kluwer Academic Publishers, 1991. pp. 159-190.

[2] V.P. Srini. DFS-SuperMPx: Low-cost Parallel Processing System for Machine Vision and Image Processing // Proc. Third International Conference "Parallel Computing Technologies", PaCT-95. St. Petersburg, 1995. Vol. 3, pp. 356-369.

[3] S. Voigt, M. Baesler, T. Teufel. Dynamically reconfigurable dataflow architecture for high-performance digital signal processing // Journal of Systems Architecture, 2010, no. 56, pp. 561-576.

[4] Yu. Shikunov, D. Khilko, Yu. Stepchenkov. Hardware and Software Modelling and Testing of Non-Conventional Data-Flow Architecture // Proceedings of the 2016 IEEE North West Russia Section Young Researchers in Electrical and Electronic Engineering Conference (ElConRusNW), 2016, pp. 360-364.

[5] Yu. Stepchenkov, V. Volchek, V. Petrukhin, A. Prokofyev. Hardwere maintenance for digital processing of speech signals in the recurrent dataflow processor // Systems and means of informatics – TORUS PRESS, Moscow, 2010. P. 31-47. (in Russian).

[6] Volchek V.N., Stepchenkov Yu.A., Petrukhin V.S., Prokofyev A.A., Zelenov R.A. Digital Signal Processor With Non-Conventional Recurrent Data-Flow Architecture // Problemi razrabotki perspektivnih mikro- i nanoelektronnih system (MES), Moscow, IPPM RAS, 2010. pp. 412-417 (in Russian).

[7] Khilko D. V., Stepchenkov Yu. A, Diachenko Yu. G., Shikunov Yu. I., Morozov N. V. "Hardware and Software Modelling and Testing of Recurrent Operational Unit" // Sistemy i sredstva informatiki, 2015 Vol. 25 no. 4, pp. 78-90 (in Russian).

[8] Khilko D. V., Shikunov Yu. I., Stepchenkov Yu. A. "Multi-core Recurrent Data-flow Architecture Imitational Model Implementation Features" // Trudy Vtoroj molodezhnoj nauchnoj konferencii «Zadachi sovremennoj informatiki» - Proc. of the Second youth scientific conference "Modern Problems in Informatics", Moscow, FRC ITCS RAS, 2015. pp. 220-227 (in Russian).

[9] Khilko D.V., Stepchenkov Yu.A. Theoretical Aspects of Recurrent Architecture Programming Methodology Development // Sistemy i sredstva informatiki, 2013, Vol. 23 no. 2, pp. 133-156 (in Russian).

[10] URL: http://microchip.com.ru/Support/Download/13_64.pdf (accessed 04.04.2016).

[11] Khilko D.V., Stepchenkov Yu.A., Shikunov Yu.I. The instrumental software development environment for hybrid architecture of recurrent signal processor (GAROS IDE). Certificate of state registration of computer program No. 2015614004 from 01.04.15.

[12] Y. Stepchenkov, Y. Diachenko, V. Zakharov, Y. Rogdestvenski, N. Morozov, and D. Stepchenkov, Self-Timed Computing Device for High-Reliable Applications / Proc. International Workshop on power and timing modeling, optimization and simulation (PATMOS'2009), Delft, Netherlands, 2009. P. 276–285.

[13] Sokolov I.A., Stepchenkov Yu.A., Rozhdestvenskij Yu.V., Diachenko Yu.G. Speed-Independent Fused Multiply-Add Unit of Gigaflops Rating: Methodological Aspects // Problems of Perspective Micro- and Nanoelectronic Systems Development - 2014. Proceedings / edited by A. Stempkovsky, Moscow, IPPM RAS, 2014. Part IV. P. 51-56. (in Russian).