Math-Net.Ru
Общероссийский математический портал

Ю. А. Степченков, В. Н. Волчек, В. С. Петрухин, А. А. Прокофьев, Р. А. Зеленов, Механизмы обеспечения поддержки алгоритмов цифровой обработки речевых сигналов в рекуррентном обработчике сигналов, *Системы и средства информ.*, 2010, том 20, выпуск 1, 31–47

Использование Общероссийского математического портала Math-Net.Ru подразумевает, что вы прочитали и согласны с пользовательским соглашением
http://www.mathnet.ru/rus/agreement

Параметры загрузки:
IP: 176.15.50.216
7 сентября 2020 г., 13:44:25

УДК 004.272.44

МЕХАНИЗМЫ ОБЕСПЕЧЕНИЯ ПОДДЕРЖКИ АЛГОРИТМОВ ЦИФРОВОЙ ОБРАБОТКИ РЕЧЕВЫХ СИГНАЛОВ В РЕКУРРЕНТНОМ ОБРАБОТЧИКЕ СИГНАЛОВ[1]

Ю. А. Степченков, В. Н. Волчек, В. С. Петрухин, А. А. Прокофьев, Р. А. Зеленов

Аннотация: В статье представлены некоторые результаты разработки многоядерного цифрового сигнального процессора с нетрадиционной рекуррентной потоковой архитектурой для исполнения параллельных алгоритмов. Помимо специфики самой архитектуры, рассматриваются особенности организации отдельного процессорного ядра, ориентированного на эффективное исполнение алгоритмов в области обработки речевых сигналов. Предложен ряд механизмов для уменьшения накладных расходов при организации циклических процедур и переходов.

Ключевые слова: цифровая обработка сигналов; потоковая архитектура; рекуррентность; многоядерность; параллелизм; цикл; переход

1. Введение

В ИПИ РАН ведутся работы по созданию нетрадиционной рекуррентной архитектуры, предназначенной для реализации параллельных вычислений. Данную архитектуру можно рассматривать как развитие парадигмы потока данных (data-flow парадигмы), но построенной на базе других отправных принципов:

- оба потока — данных и команд — интегрированы в один поток самодостаточных данных; традиционная двухпоточность сводится к однопоточности;
- поток самодостаточных данных хранит рекуррентно свернутый (сжатый) алгоритм решения конкретной задачи;

[1]Работа выполнена при частичной финансовой поддержке по Программе фундаментальных исследований ОНИТ РАН на 2010 г.

– в основе парадигмы вычислений лежит графодинамическое представление алгоритмов, рекуррентно свернутых до момента инициации исполнения и саморазворачивающихся в ходе процесса выполнения задач [1].

В качестве практической апробации разрабатываемой архитектуры предлагается структура рекуррентного обработчика сигналов (РОС), предназначенного для эффективного исполнения параллельных алгоритмов в области обработки речевых сигналов. Разработка подобного цифрового сигнального процессора (ЦСП) представляется перспективным и своевременным проектом, соответствующим тенденциям развития области цифровой обработки сигналов (ЦОС) [1]. Многоядерный РОС в текущем исполнении имеет четыре процессорных ядра (ПЯ), способных работать параллельно. Подобный уровень параллелизма можно назвать параллелизмом на уровне ПЯ.

Потребность в параллельных вычислениях в ЦОС связана с тем, что целый класс используемых алгоритмов хорошо поддается распараллеливанию. Традиционные архитектуры, относящиеся к классу архитектур фон Неймана, обладают рядом недостатков, которые, в частности, не позволяют организовывать эффективную поддержку параллельных вычислений. Именно на решение этих проблем и ориентирована разрабатываемая рекуррентная потоковая архитектура, ожидаемые преимущества которой описаны в [1].

Помимо особенностей на уровне вычислительной парадигмы, в РОС присутствует ряд специализированных механизмов, обеспечивающих поддержку алгоритмов обработки речевых сигналов. В частности, в качестве демонстрационной задачи на РОС планируется реализовывать программу распознавания речи. Решение этой и другой подобной задачи (распознавание диктора) сводится к последовательности этапов, на каждом из которых выполняется фильтрация сигнала и независимая обработка промежуточных данных в цикле по полосам спектра сигнала. Эти операции хорошо распараллеливаются и эффективно реализуются на четырехядерном РОС.

Можно выделить следующие основные алгоритмические этапы [2]:

(1) предобработка речевого сигнала на базе использования фильтра Баттерворда четвертого порядка со степенью параллельности, равной четырем;

(2) вычисление барковского спектра с использованием быстрого преобразования Фурье (БПФ) на 128 или 256 точек. Максимальная степень параллельности пропорциональна числу используемых точек;

(3) шумоподавление с помощью преобразования барковского спектра (логарифмирование, взвешивание, RASTA фильтрация, экспонирование) по каждой из 15 барковских полос. Каждая из этих операций допускает распараллеливание, при этом степень параллельности равна числу полос — 15;

(4) квантование вектора параметров на основе векторной кодовой книги для поиска ближайшей модели. В качестве меры близости квантуемого вектора к табличному используется евклидово расстояние. Максимальная степень параллельности равна количеству векторов в кодовой книге (обычно 256);

(5) поиск ближайшей модели, наилучшим образом соответствующей произнесенному слову с использованием метода Витерби на дискретных марковских моделей. В случае распознавания диктора степень параллельности алгоритма Витерби равна количеству «своих дикторов».

Таким образом, реализация перечисленных алгоритмов распадается на последовательность этапов, степень параллельности каждого из которых варьируется от 4 до 256. Четырехядерный вариант реализации РОС с соответствующей аппаратной поддержкой данного класса алгоритмов представляется целесообразным и эффективным.

Поскольку задача создания РОС носит исследовательский характер, чрезвычайно важна возможность гибкого внесения изменений в его структуру при минимальных финансовых и временных затратах. В соответствии с этим ключевым критерием в качестве элементной базы реализации РОС выбраны программируемые логические интегральные схемы (ПЛИС). В результате подробного анализа свойств конкретных семейств ПЛИС [1] предпочтение было отдано микросхемам фирмы Altera — Stratix III (система на кристалле), обладающим высокой производительностью и содержащим многообразную синтезируемую и аппаратную периферию.

Рассмотрим механизмы обеспечения поддержки алгоритмов обработки речевых сигналов в РОС более подробно.

2. Система команд и организация вычислений

Специфика алгоритмов ЦОС накладывает свои определенные требования на систему команд и организацию вычислительного устройства (ВУ) [3]. Большинство команд, поддерживаемых РОС, а также основные функциональные блоки ВУ отдельного ПЯ свойственны традиционным сигнальным процессорам. В качестве аналога для сравнения эффективности ЦОС авторы используют кристалл dsPIC30F компании Microchip — популярный ЦСП нижнего ценового диапазона. Тем не менее, в РОС за счет более эффективной организации вычислений и внедрения специализированных режимов удалось расширить систему команд и повысить эффективность реализации ряда алгоритмов. Рассмотрим более подробно структуру и организацию ВУ в РОС.

Одно из наиболее важных требований к ЦСП — эффективная поддержка операции суммирования результатов умножения. Эта операция одинаково важна для цифровых фильтров, БПФ и множества других алгоритмов ЦОС. Вычислительное устройство РОС оптимизировано для выполнения повторяющихся математических операций, таких как умножение с накоплением. Для этих целей в ВУ РОС, как и в стандартных ЦСП, присутствует аппаратный блок MAC (Multiplication with ACcumulation), позволяющий за один вычислительный такт перемножать два 16-разрядных операнда и накапливать результат умножения в одном из двух внутренних 40-разрядных регистров. Разрядности входных обрабатываемых чисел в 16 бит вполне достаточно для целого ряда применений в области речевой обработки.

Помимо операции умножения с накоплением, блок MAC, входящий в состав РОС, способен выполнять команды арифметического, логического сдвигов и округления результатов. Вычислительное устройство содержит также 16-разрядное арифметико-логическое устройство (АЛУ), выполняющее основные арифметические и все логические команды.

Перечисленные выше блоки являются, по сути, необходимым стандартным минимумом для большинства ЦСП. На примере dsPIC30F компании Microchip был проведен анализ эффективности реализации речевых алгоритмов в традиционных ЦСП нижнего ценового диапазона. Результаты, полученные в ходе анализа, показали неэффективность использования имеющихся ресурсов вычислительного ядра.

Так, например, в dsPIC30F, как и в ряде современных ЦСП, в каждом вычислительном цикле используется только один из функциональных блоков вычислителя; при использовании MAC — простаивает АЛУ, при использовании АЛУ — простаивает MAC. Вычислительное устройство содержит два 40-разрядных регистра-аккумулятора (как и в ряде стандартных ЦСП) и один 16-разрядный регистр — выходной аккумулятор (в силу специфики РОС). В стандартных ЦСП, как правило, в каждом вычислительном цикле используется только один аккумулятор из двух, а при стандартном подходе в РОС — один из трех.

Каждый из перечисленных узлов и регистров ВУ представляет собой независимый аппаратный ресурс, цена простаивания которого достаточно высока. Это обстоятельство натолкнуло на мысль о внедрении поддержки их одновременной работы — *суперскалярности* — в ВУ.

Суперскалярность — это архитектура ВУ, использующая несколько декодеров команд, которые могут нагружать работой множество исполнительных блоков. Планирование динамического выполнения потока команд осуществляется самим ВУ [4]. Таким образом, наделение ВУ свойством суперскалярности позволяет достичь параллельности на уровне операций отдельных команд.

Анализ алгоритмов обработки голосовых сигналов выявил необходимость использования команд сложения/вычитания и умножения в суперскалярном режиме. Потенциально существует возможность выполнять и другие комбинации команд, но практической необходимости в этом в рассматриваемой области сейчас не возникает. Рассмотрим суперскалярную организацию ВУ РОС более подробно.

В РОС отсутствует поток команд и данных. Вместо них есть поток самодостаточных данных, которые несут в себе всю необходимую информацию для выполнения (содержат и команду, и данные). Как следствие, в ВУ всегда анализируются коды операции обоих входных операндов. Если они совпадают, выполняется текущая операция в обычном последовательном режиме с полезной работой только одного аппаратного узла ВУ и одного или двух аккумуляторов. Если коды операций не совпадают — это либо ошибка, либо переход в суперскалярный режим (сложение/вычитание и умножение). При этом один операнд для команды сложения/вычитания отправляется в АЛУ, другой — в блок MAC. Для формирования пары для каждого из посту-

Таблица 1 Эффективность реализации алгоритмов

Алгоритм	Число шагов вычисления	Реализация		
		dsPIC30F	РОС	1/4 РОС[2)]
«Бабочка»	«Бабочки»	9	4 (4)[1)]	4
Rasta-фильтрация	Одного параметра	8	2	8
	Общее	400	27	—
Евклидово расстояние	Одной координаты	2	1 (4)[1)]	1
	Общее	32	13	—
LSP-параметры	Одного параметра	3	2 (4)[1)]	2
	Общее	62	6	—

[1)] В скобках указано, что одновременно вычисляются 4 «бабочки», 4 координаты или 4 параметра.

[2)] Для одноядерного варианта исполнения РОС.

пивших операндов считывается значение из двух 40-разрядных аккумуляторов, присутствующих в блоке MAC, или из памяти коэффициентов (постоянное запоминающее устройство в каждом ВУ).

Однако возможности одновременного использования нескольких аппаратных узлов РОС не ограничиваются только параллельным исполнением двух команд за один такт синхронизации. Поскольку статистические данные свидетельствуют, что БПФ — это базовый алгоритм ЦОС, в том числе и в области голосовых технологий, в состав команд РОС была введена ***специальная многоцикловая команда «Butt»*** (Butterfly), одновременно задействующая максимальное количество функциональных узлов ВУ.

Команда «Butt» реализует базовую операцию алгоритма БПФ по основанию 2 с прореживанием по времени («Бабочка»). В существующих современных ЦСП, например, в том же dsPIC30F от Microchip, этот алгоритм выполняется за 9 вычислительных шагов [5]. В РОС специальная команда «Butt» позволяет сократить количество шагов до четырех при наличии одинакового количества вычислительных ресурсов с большинством существующих ЦСП (табл. 1). Это стало возможным путем оптимального распределения ресурсов при вычислении «Бабочки». При реализации этой команды параллельно задействуются все существующие аппаратные узлы ВУ, а именно — 16-разрядное АЛУ, 16-разрядный умножитель и 40-разрядный сумматор.

Еще одна особенность ВУ заключается во внедрении так называемого ***псевдосуперскалярного*** режима. Суть этого режима заключается в следующем. Обычная операция умножения с накоплением предполагает сложение в 40-разрядном сумматоре ранее накопленного результата с результатом умножения поступивших двух операндов в текущем цикле. В псевдосуперскалярном режиме в текущей операции обрабатываются не два, а три операнда; третий — содержимое 40-разрядного регистра. В рамках этой операции результат умножения не накапливается, а после однократного суммирования (вычитания) с входным операндом уходит на обработку в соседнее ПЯ. Поскольку в таком режиме блоки АЛУ и МАС не задействуются в параллель, этот режим назван псевдосуперскалярным. Он используется, например, в алгоритме Rasta-фильтрации.

Все эти решения при одинаковом, по сравнению со стандартными ЦСП, количестве аппаратных блоков ВУ позволяют достичь ощутимого эффекта в выполнении ряда алгоритмов ЦОС (см. табл. 1).

Несмотря на специфику разрабатываемой архитектуры, ВУ при незначительной доработке может быть использовано и в традиционных изделиях ЦОС. Этому также способствует реализация всего РОС на языке описания аппаратуры VHDL, что позволяет при необходимости модифицировать вычислитель под конкретное аппаратное окружение и использовать его в качестве IP-ядра в других проектах или проектах третьих лиц.

3. Поддержка циклов и переходов

Циклы, позволяющие выполнить некоторый участок программы многократно, являются одной из наиболее употребительных конструкций, в том числе в рамках набора алгоритмов распознавателя слов. Для минимизации накладных расходов при реализации циклических процедур в архитектуру РОС введены следующие средства:

(1) один 8-разрядный счетчик циклов в составе ВУ каждого ядра;
(2) один функциональный блок — память ветвлений (ПВ) в каждом ядре;
(3) четыре формата операндов: загрузчик тегов, загрузчик счетчика циклов, контроллер циклов внутренний и контроллер циклов внешний;

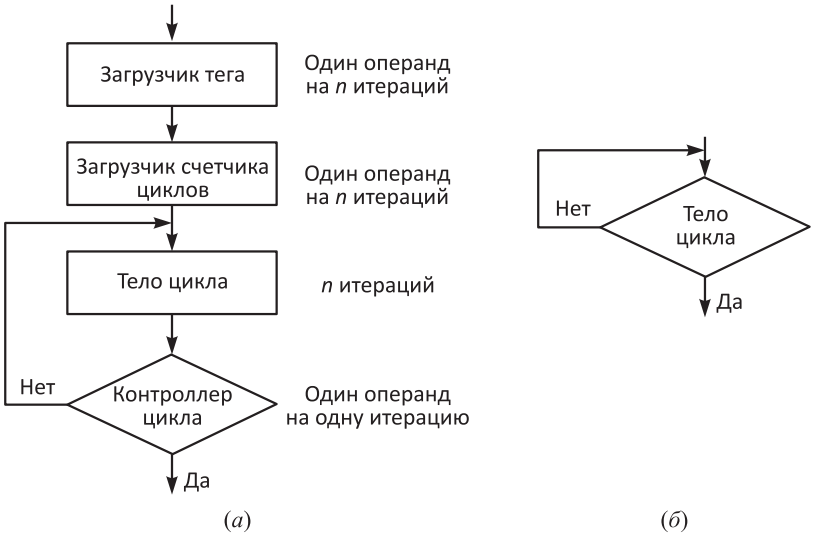

Рис. 1 Варианты реализации цикла в РОС: (*а*) стандартный вариант; (*б*) оптимизированный вариант для внутренних циклов

(4) одно двухразрядное функциональное поле типа операции.

Стандартный вариант реализации цикла в РОС представлен на рис. 1, *а*.

Перед инициацией первой команды в ПВ из тела цикла должен быть помещен загрузчик тега, который будет определять теги выходного результата после отработки требуемого числа итераций. Значение операнда «загрузчик счетчика циклов» обеспечит первоначальную настройку счетчика циклов в ВУ на требуемое число повторений. После окончания обработки последнего операнда в теле цикла на ВУ поступает контроллер цикла, действие которого аналогично команде Loop в системе команд МПx86 или Repeat в системе команд dsPIC30F: декрементация значения счетчика циклов и анализ его состояния.

Если после декрементации значение счетчика циклов больше нуля, идет повторное исполнение тела цикла — традиционное продолжение вычислений в соответствии со стандартной процедурой формирования функциональных полей. В противном случае происходит завершение цикла и переход к исполнению команды, параметры которой заданы в ПВ.

Таблица 2 Инициация внутренних циклов и переходов

Значение подполя O_t		Операция
Символ	Код	
	0	Не специфицировано (резерв)
c	1	Выполнение обычной операции и инициация цикла
b	2	Выполнение обычной операции и инициация перехода
t	3	Обычная операция в соответствии с подполем $[O_c]$

Примечания: c — cycle; b — branch; t — typical.

Чем меньше длина тела цикла, тем больше величина накладных расходов — относительное число служебных процедур по сравнению с собственно вычислительными процедурами. Например, при вычислении коэффициентов автокорреляции тело циклов равно единице и представляет собой операцию перемножения входных операндов с накоплением. В этом случае каждое полезное действие — выполнение тела цикла — будет сопровождаться одним служебным действием. Таким образом, при реализации цикла в соответствии с рис. 1, a величина накладных расходов будет превышать 50%.

Один из путей снижения накладных расходов при реализации циклов и переходов в РОС состоит в следующем. В состав обязательных функциональных полей (тегов) содержательных операндов вводится двухразрядное подполе O_t, которое позволяет детализировать тип (type) выполняемой операции (operation) (табл. 2). Любая стандартная команда, которая является последней в цикле, может сопровождаться установкой значения подполя $O_t = c$.

Приход в ВУ команды с этим установленным подполем вызывает в нем инициацию двух параллельных процессов:

– выполнение требуемой операции в соответствие со значением подполя O_c;
– декрементацию значения счетчика циклов.

Система команд РОС содержит ряд одноопрандных команд и предусматривает возможность объединения в пару двух операндов, инициирующих в ВУ две операции. Например, в пару могут быть объединены операнд, устанавливающий в требуемое значение один из трех аккумуляторов ВУ, и операнд-загрузчик счетчика циклов. Если учесть, что операнд загрузчик тега не

задействует информационные пути ВУ, то реализация циклических процедур может быть сведена к виду, представленному на рис. 1, *б* и характеризующемуся нулевыми накладными расходами.

Аналогичным образом могут быть минимизированы накладные расходы при реализации переходов. Для этого в архитектуру РОС введены:

(1) один функциональный блок в составе ВУ — регистр условий;
(2) один операнд — загрузчик регистра условий;
(3) два формата операнда перехода.

Операнд типа «загрузчик регистра условий» используется для установки флагов по маске: сброса в «0» (символ типа операции f0) или установки в «1» (символ типа операции f1). Каждый разряд в операнде «загрузчик регистра условий» строго соответствует определенному флагу регистра состояний ВУ. При поступлении на обработку операнда со значением $O_t = b$ ВУ выполняет требуемую операцию в соответствии со значением подполя O_c и производит поразрядное сравнение установленных разрядов регистра состояний (результат операции) и регистра условий. При совпадении логических единиц по одному из разрядов осуществляется переход по значению из ПВ.

4. Буферное запоминающее устройство

Архитектура РОС является двухуровневой: верхний (управляющий) уровень — управляющий процессор (УП) с фоннеймановской архитектурой (предполагается использовать 32-разрядный синтезируемый процессор NIOS II); нижний (операционный) — рекуррентное операционное устройство (РОУ), базирующееся на четырех ПЯ. На управляющий уровень РОС возлагается выполнение процедур предварительной подготовки капсул, реализация последовательных частей исполняемой программы. Операционный уровень РОС обеспечивает параллельные вычисления в процессорных ядрах [1].

При взаимодействии между управляющим и операционным уровнями РОС возникает проблема организации быстрой передачи данных. На входы РОУ за один цикл должны поступать до четырех пар 56-разрядных операндов, по два для каждого ПЯ. С целью сокращения объема капсул введен специальный режим работы РОУ, в котором он способен принимать операнды

с упакованной тройкой содержательных данных. При работе в данном режиме в РОУ должно быть отправлено три 56-разрядных операнда. Тем не менее, управляющий процессор не способен передать такое количество данных за один цикл, поэтому потребовалось введение буферного запоминающего устройства (БЗУ), которое накапливает в себе данные и по мере готовности передает их на обработку в РОУ.

Алгоритмы, использующиеся для задач цифровой обработки речевых сигналов, программируются в виде капсул — наборов самодостаточных данных. Во время работы РОС выполняются одни и те же капсулы, изменяются только 16-разрядные содержательные части операндов, в то время как функциональные части остаются без изменения. Поэтому в БЗУ можно хранить шаблоны капсул с заполненными функциональными полями, а содержательные части заполнять перед очередным запуском капсулы на исполнение. Следовательно, единожды записанные в БЗУ шаблоны капсул могут многократно использоваться в процессе работы РОС. Данный механизм позволяет увеличить скорость передачи данных и минимизировать простои РОУ на ожидание готовности очередной порции данных за счет уменьшения трафика между УП и БЗУ.

Теперь возникает проблема организовать БЗУ таким образом, чтобы оно обеспечивало за один цикл отправку четырех операндов в РОУ, а также прием четырех операндов со стороны РОУ, и при этом не замедлило работу конвейера. Ниже описаны решения, позволяющие построить БЗУ оптимальным образом.

На рис. 2 представлена функциональная схема БЗУ, состоящего из:

- буферной памяти (БП);
- памяти признаков (ПП);
- устройства управления (УУ).

Буферная память предназначена для хранения шаблонов капсул. В режиме обмена данными с РОУ происходит одновременное чтение и запись (со стороны РОУ) в БП, поэтому она реализована на основе двухпортового запоминающего устройства (ЗУ). В связи с тем, что оптимальным вариантом является запись в БП 16-разрядных блоков данных, адресное пространство разделяется на 16-разрядные банки.

Память признаков контролирует наличие данных в шаблонах капсул. Если данные в БП готовы для считывания, то в ПП по данному адресу заносится ‘1’, иначе — ‘0’. При обмене

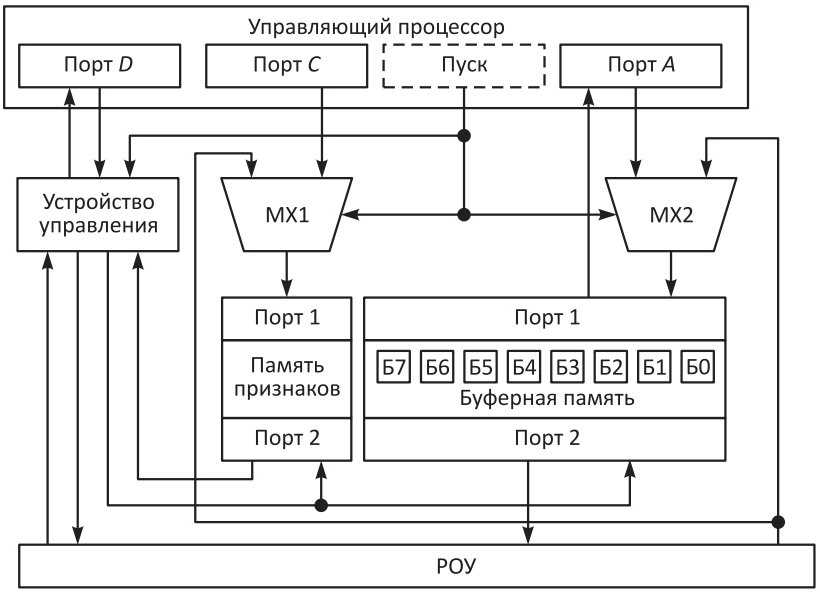

Рис. 2 Функциональная схема БЗУ

данными БЗУ с РОУ возникает необходимость одновременного изменения признаков по двум адресам (обнуление для считанных данных, и выставление ‘1’ для принятых). Поэтому ПП построена на основе 2-портовой памяти, что обеспечивает возможность одновременной записи в порт 1 и чтение из порта 2 или одновременной записи в порт 1 и порт 2.

Максимально допустимая длина цикла работы БЗУ — это 4 обращения к памяти. За это время отрабатывает самая медленная ступень конвейера РОУ. Поэтому, исходя из разрядности содержательной части (16 бит), разрядности операнда (56 бит), количества операндов, которые необходимо считать в РОУ за один цикл (4), максимально допустимого количества обращений к памяти (4) и количества операндов, которые должны быть записаны в БП со стороны РОУ (4), БП целесообразно разделить на восемь банков (Б0–Б7). При этом цикл работы с БЗУ составляет 4 такта, на каждом из которых возможен прием операнда со стороны РОУ. Каждый банк памяти обеспечивает хранение 8 К 16-разрядных значений. Максимальный объем БП составляет 64 К 16-разрядных слов. Выбор банка памяти, к которому

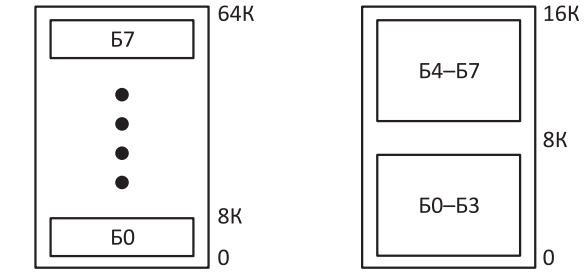

Рис. 3 Схема адресного пространства БП (*а*) со стороны УП; (*б*) со стороны РОУ

обращается УП, осуществляется тремя старшими разрядами 16-разрядного адреса. При такой организации БП содержательные части операндов хранятся в блоках Б0 и Б4 для обычного содержательного операнда, и блоках Б0–Б2 и Б4–Б6 для операндов с упакованной тройкой данных.

Порт 1 БП обеспечивает возможность записи и чтения со стороны УП 16-разрядного значения данных или записи со стороны РОУ 56-разрядного операнда. Адресное пространство БП со стороны УП представлено на рис. 3, *а*, со стороны РОУ — на рис. 3, *б*. Обращение за данными или их запись в БП осуществляется стандартной процедурой обращения к памяти через порт *A*. Возможность выбора направления записи данных от УП или РОУ обеспечивает МХ2. Обращение со стороны РОУ возможно только в режиме пуска РОУ.

Порт 2 БП обеспечивает только чтение 56-разрядных операндов в режиме пуска РОУ. Для этого все банки памяти имеют единое адресное пространство, т. е. за одно обращение к БП по порту 2 считывается два 56-разрядных операнда, за цикл должно быть осуществлено два таких обращения.

В связи с тем, что обмен данными между БЗУ и РОУ носит интенсивный характер, невозможно организовать одновременное взаимодействие БЗУ с РОУ и УП, поэтому БЗУ работает в двух режимах:

(1) в режиме приема данных со стороны УП (сигнал пуска ‘0’) БЗУ по порту *A* принимает 16-разрядные данные и записывает в ячейку определенную 16-разрядным адресом (рис. 3, *а*), признак готовности данных выставляется по

порту C, после записи соответствующего операнда в БП. Конвейер РОУ при этом приостановлен. После того как все необходимые данные записаны в память, УП переводит БЗУ и РОУ в режим пуска;

(2) в режиме пуска (сигнал пуска ‘1’) БЗУ осуществляет обмен данными с РОУ, доступ со стороны УП к памяти при этом закрыт. За цикл осуществляется чтение четырех операндов из БП, также в нее может быть записано до четырех операндов со стороны РОУ. Адрес чтения вычисляется устройством управления, так как данные могут считываться из памяти не по порядку, а по иному закону считывания, установленному операционным уровнем. Также УУ контролирует количество запусков капсулы при ее многократном использовании. По завершении выполнения капсулы или при возникновении ситуации, когда очередной операнд не готов (ПП хранит ‘0’ по соответствующему адресу), УУ приостанавливает конвейер РОУ и передает управление УП.

Порт D управляющего процессора предназначен для организации процесса взаимодействия с БЗУ: через него устанавливается начальный адрес капсулы, количество ее запусков, настраивается закон считывания данных из БП.

Рассмотрим типовой случай работы БЗУ. При первоначальном включении системы сигнал пуска не активизирован; УП через МХ1 и МХ2 имеет доступ к БП и ПП. Он записывает шаблоны капсул: данные в БП и соответствующие признаки в ПП. Для запуска капсулы на исполнение УП через порт D записывает все необходимые настроечные параметры. После этого УП дает команду пуска (запись ‘1’ в регистр пуска). Далее события

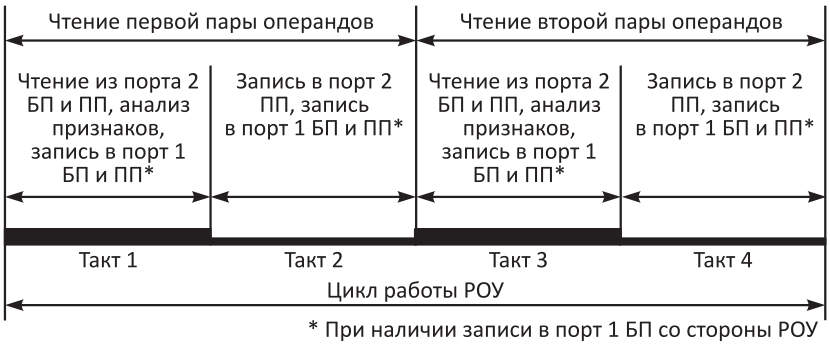

Рис. 4 Временна́я диаграмма работы БЗУ

развиваются в соответствии с временно́й диаграммой, изображенной на рис. 4.

Цикл работы БЗУ состоит из 4 тактов. Причем время цикла работы БЗУ равно времени цикла работы одной ступени конвейера РОУ.

В такте 1 из портов 2 БП и ПП осуществляется чтение операнда и признаков. Устройство управления выполняет смещение адреса. Одновременно с этими действиями в такте 1 осуществляется запись данных из РОУ в БП и ПП по порту 1.

В такте 2 производится сброс ПП по порту 2 в '0', если на предыдущем такте была установлена готовность данных, и они были считаны, также осуществляется запись данных из РОУ в БП и ПП по порту 1.

Такты 3 и 4 выполняются аналогично первым двум тактам действия.

Таким образом, БЗУ является сложным устройством, и от его реализации зависит эффективность работы всего РОС, ведь

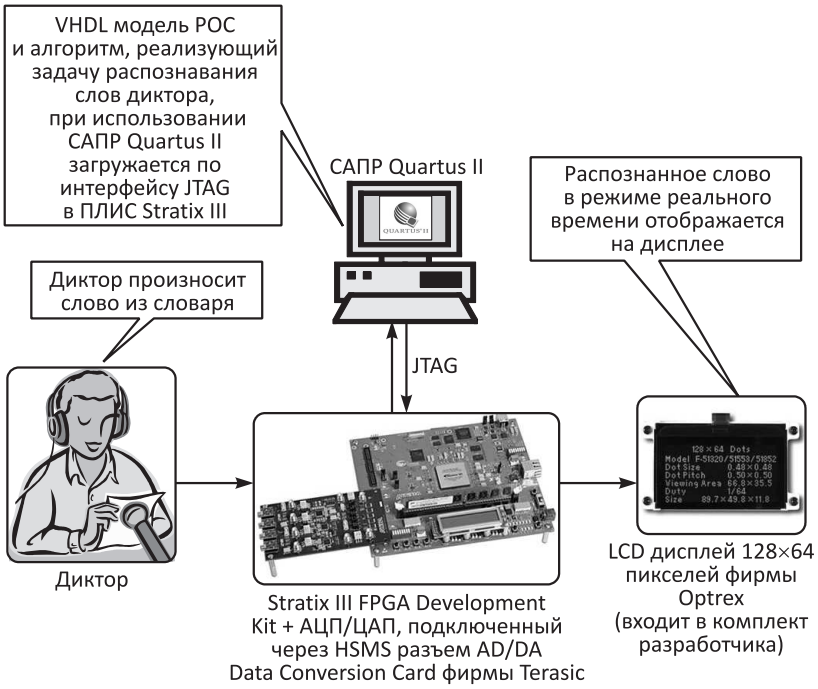

Рис. 5 Предполагаемый демонстрационный стенд

процесс подготовки данных для дальнейшей их обработки в РОУ при неоптимальной организации может потребовать времени много большего, чем время обработки данных. Описанная выше реализация БЗУ нацелена именно на то, чтобы процесс обмена данными между управляющим и операционным уровнями не стал узким местом архитектуры.

Как было сказано выше, в качестве тестовой задачи, решаемой на РОС и демонстрирующей эффективное исполнение алгоритмов обработки речевых сигналов, используется задача распознавания слов диктора в режиме реального времени (рис. 5). Для демонстрации этой задачи и реализации текущего варианта РОС было закуплено следующее оборудование:

– комплект разработчика Stratix III Development Kit;
– АЦП/ЦАП Terrasic AD/DA Data Conversion Card.

5. Заключение

Приведенный перечень механизмов, реализованных в РОС, позволяет повысить производительность вычислений для целого ряда алгоритмов обработки речевых сигналов. Часть этих нововведений, касающихся системы команд и организации режимов их работы, напрямую не связаны с особенностями рекуррентной потоковой архитектуры. Преимущества достигаются за счет максимально эффективного использования аппаратных узлов, имеющихся в большинстве современных ЦСП. Из этого обстоятельства следует вывод — данные решения могут быть также реализованы в других ЦСП.

Еще одной особенностью разрабатываемой рекуррентной потоковой архитектуры является ориентация в перспективе на самосинхронную схемотехнику: самосинхронизация на логическом уровне (по готовности исходных данных) хорошо сочетается с самосинхронизацией на аппаратном уровне (по готовности результатов). Поэтому в настоящее время, несмотря на то, что используется синхронный схемотехнический базис, взаимодействие между функциональными блоками и ступенями вычислительного конвейера осуществляется асинхронно.

Литература

1. *Степченков Ю. А., Петрухин В. С.* Особенности гибридного варианта реализации на ПЛИС рекуррентного обработчика сигналов //

Системы и средства информатики: Доп. вып. — М.: ИПИ РАН, 2008. — С. 118–129.

2. *Рождественский Ю. В., Дьяченко Ю. Г., Морозов Н. В.* Проблемы реализации распознавателя речи на рекуррентном процессоре // Методы и средства разработки информационно-вычислительных систем и сетей (спец. вып.). — М.: Наука, 2004. — С. 66–76.
3. *Кестер У.* Гл. 7. Аппаратура цифровых сигнальных процессоров. URL: http://www.analog.com.ru/Public/7.pdf (дата обращения 15.01.2010).
4. *Таненбаум Э.* Архитектура компьютера. — СПб.: Питер, 2007. 844 с.
5. *Степченков Ю. А., Петрухин В. С.* Перспективы развития цифровых сигнальных процессоров и возможная реализация рекуррентного обработчика сигналов // Системы и средства информатики. Спец. вып. «Методы и средства разработки информационно-вычислительных систем и сетей». — М.: ИПИ РАН, 2004. — С. 92–140.