УДК 004.272.32

Применение конвертируемых режимов адресации для повышения производительности сопроцессоров цифровой обработки сигналов в составе многоядерной СнК

А.Ю. Пантелеев, И.И. Шагурин

НИЯУ "МИФИ", apanteleev87@gmail.com, ishagurin@inbox.ru

***Аннотация* — В докладе предлагается способ организации векторной памяти, позволяющий конвертировать режимы адресации данных для ускорения выполнения алгоритмов цифровой обработки сигналов (DSP). Описана структура векторно-конвейерного сопроцессора (ВКС) с конвертацией режимов адресации, который выполняет основные процедуры DSP с высокой эффективностью: например, при вычислении БПФ-256 загрузка вычислительных блоков достигает 89%. Рассмотрены особенности архитектуры ВКС и реализация алгоритмов БПФ, свертки и умножения вектора на матрицу. Представлены результаты синтеза ВКС для реализации в составе «Систем на Кристалле» (СнК), ориентированных на выполнение DSP.**

***Ключевые слова* — векторно-конвейерный сопроцессор, конвертация режимов адресации, цифровая обработка сигналов, быстрое преобразование Фурье (БПФ).**

I. Введение

Основными алгоритмами, которые применяются в цифровой обработке сигналов (DSP), являются БПФ, КИХ-фильтры (свертка) и умножение вектора на матрицу для построения нейросетей. Эффективная реализация фильтров и БПФ на векторных архитектурах является достаточно сложной задачей в связи с нерегулярной адресацией данных – эти проблемы рассмотрены в работах [1], [2], [3] и ряде других.

Типовая структура СнК для цифровой обработки сигналов (рис. 1) содержит следующие блоки:

- RISC-процессор для обработки целочисленных данных и управления системой;
- интерфейсные блоки для работы с периферией, такие как UART, АЦП;
- внутренняя память SRAM для хранения обрабатываемых данных;
- специализированные блоки цифровой обработки данных;
- контроллеры прямого доступа к памяти (DMA) для быстрой передачи массивов данных.

Для реализации типовых алгоритмов DSP в состав СнК часто включаются узкоспециализированные блоки, выполняющие с высокой скоростью определенные процедуры, например, БПФ - такие блоки описаны в работах [4], [5] и многих других. Другим вариантом является использование программируемых DSP-сопроцессоров [6], [7], способных выполнять широкий набор задач, но с более низкой производительностью из-за проблем, связанных с реализацией требуемых режимов адресации данных и обеспечением эффективной загрузки вычислительного конвейера.

В данной работе представлен подход к проектированию векторно-конвейерного сопроцессора (ВКС), в котором реализуется конвертация режимов адресации памяти, что позволяет существенно ускорить выполнение типовых процедур DSP (вычисления БПФ и свертки, умножения вектора на матрицу) по сравнению с векторными процессорами, использующими только простые векторные регистры.

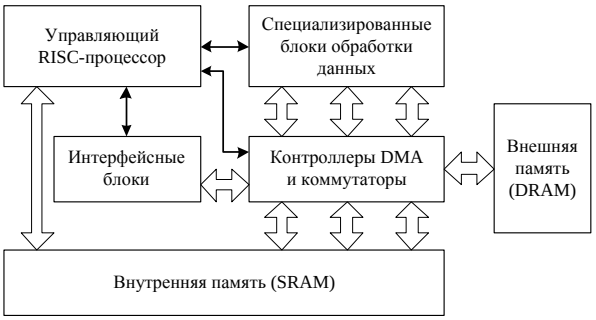

Рис. 1. Типовая структура СнК для DSP

II. Архитектура ВКС

Структурная схема, разработанного ВКС, представлена на рис.2. Большую часть сопроцессора составляют три одинаковых страницы векторной памяти данных объемом по 32 КБ каждая. Порты памяти соединяются через коммутаторы с векторным вычислительным блоком и блоками ввода-вывода, которые обеспечивают интерфейс с контроллерами DMA. Запросы на чтение и запись в память поступают от планировщика вычислений и блоков ввода-вывода.

ВКС обрабатывает данные, загруженные во внутреннюю память внешними устройствами по двум входным шинам, и выдает полученные результаты по одной выходной шине шириной 64 бита каждая. До-

пускается загрузка и выгрузка данных одновременно с обработкой другой задачи.

Вычисления производятся в соответствии с программой, заранее загружаемой во внутреннюю память кода, которая находится в планировщике вычислений. Эта программа состоит из инструкций, которые могут описывать только последовательные вычисления – никаких условных переходов в вычислительной программе быть не может. Планировщик вычислений декодирует инструкции и обеспечивает их выполнение, генерируя управляющие сигналы для вычислительного блока и команды чтения/записи для памяти с учетом зависимостей по данным в программе.

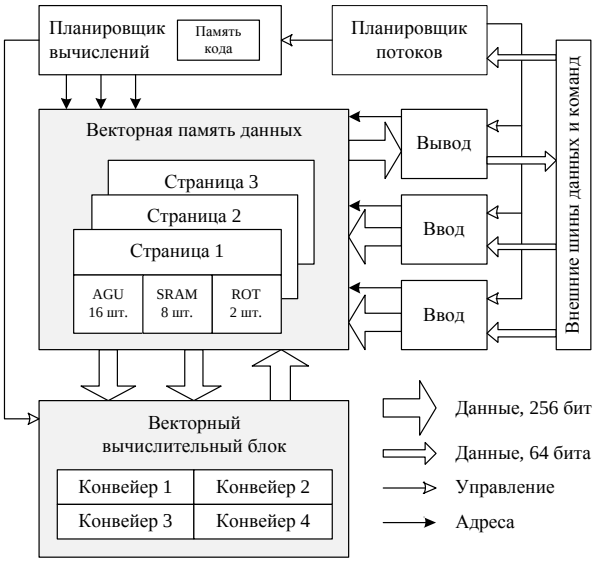

Рис. 2. Общая структура ВКС

Каждый конвейер векторного вычислительного блока содержит 2 умножителя и 3 сумматора, работающих с 32-битными числами формата «с плавающей точкой», соответствующего стандарту IEEE 754. Производительность и длина конвейера зависит от выполняемой операции. Обеспечивается возможность выбора количества конвейеров в вычислительном блоке сопроцессора при синтезе RTL-модели для получения требуемой производительности.

Запуском вычислений и загрузкой данных и программ управляют команды, которые подаются в планировщик потоков от управляющего процессора по входной шине команд. Этот блок определяет возможные зависимости между потоковыми командами и запускает выполнение команд тогда, когда это не приведет к неопределенным результатам (например, не допускается одновременная загрузка и выгрузка данных для одной области памяти). Благодаря этой особенности управляющему процессору не надо синхронизировать каждый шаг с ВКС, а достаточно выдать последовательность потоковых команд и настроить контроллеры DMA, которые загрузят необходимые данные и заберут результаты.

Количество страниц памяти данных выбрано так, чтобы было возможно производить вычисление БПФ в потоковом режиме – т.е. одновременно вычислять одно преобразование, загружать данные для следующего и выгружать результаты предыдущего. В таком режиме на первой странице располагаются данные текущего преобразования, и оба ее порта – чтения и записи – заняты вычислениями. На второй странице располагаются коэффициенты преобразования. Их не следует располагать на одной странице с данными, так как это приведет к некоторой потере производительности вычислений (10% для БПФ-256). Порт записи второй страницы остается свободен, и в нее можно загружать коэффициенты для следующего преобразования, если это требуется (например, при вычислении фрагментов преобразования очень большой размерности). Третья страница используется для асинхронной загрузки и выгрузки данных, и после этого меняется назначением с первой страницей.

Объем памяти данных выбирается таким образом, чтобы было возможно производить вычисление БПФ в потоковом режиме. Для этого на каждой странице должны полностью помещаться два преобразуемых вектора. Объем страницы 32 КБ позволяет выполнять БПФ-2048 в потоковом режиме, или БПФ-4096 без асинхронной загрузки данных.

III. Организация векторной памяти

Вычислительная программа работает с данными, упакованными в векторные «регистры». Фактически все данные располагаются в векторной памяти, поэтому термин «регистр» здесь применим лишь условно. Регистры содержат векторные элементы, количество которых во всех регистрах одинаково и задается при запуске программы. Векторный элемент – это множество действительных или комплексных чисел (скалярных элементов) фиксированной размерности, которые можно считать или записать в векторную память одновременно. То есть, для сопроцессора с 4 конвейерами, один векторный элемент – это 4 комплексных числа или 8 действительных чисел.

Регистры сгруппированы в сегменты – области памяти с независимой адресацией. В каждом сегменте может быть до 64 регистров, а всего предусмотрено 8 сегментов. Преобразование номера регистра во множество адресов в векторной памяти осуществляется блоками AGU и управляется с помощью сегментных регистров, которые располагаются в планировщике потоков и загружаются при помощи специальных команд.

При обращении любого из блоков (планировщик вычислений, блоки ввода-вывода) к векторной памяти происходит выбор страницы по номеру сегмента и генерация множества адресов в банках памяти для одного векторного элемента. Обращения в память всегда адресуют один векторный элемент, а для выполнения операции над содержимым регистра целиком производится последовательное обращение к каждому векторному элементу этого регистра.

Структурная схема одной страницы векторной памяти представлена на рис. 3. Страница состоит из N банков двухпортовой статической памяти (SRAM), на каждый из которых установлено два адресных генератора – на чтение (AGU-R) и на запись (AGU-W). Данные на входе и выходе проходят через блоки циклического сдвига. Каждый блок циклического сдвига является комбинационной схемой с N входными и N выходными 32-битными шинами и передает на выходную шину по номеру K значение с входной шины по номеру (K±R) mod N, где R – значение сдвига. Блок сдвига, обрабатывающий данные для записи, производит сдвиг в сторону больших номеров (использует знак минус), а блок сдвига, обрабатывающий прочитанные данные, – в сторону меньших номеров (использует знак плюс).

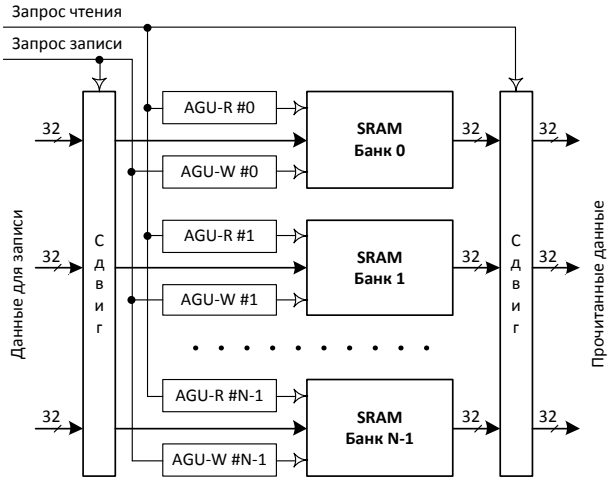

Рис. 3. Схема одной страницы векторной памяти

Генерация адресов происходит в соответствии с выбранным для данного сегмента режимом адресации. Реализуется 5 режимов адресации, разделенных на две группы по совместимости. Рассмотрим эти группы по отдельности.

A. Простой, скалярный и сверточный режимы

Простой режим. Память в этом режиме представляется в виде матрицы, где каждая строка является отдельным регистром.

Скалярный режим предназначен только для чтения. При чтении регистра содержимое одной ячейки памяти используется во всех элементах получаемого вектора. Этот режим служит для замены непосредственных констант в коде программы, например, коэффициентов при выполнении БПФ. Для загрузки данных в такой сегмент используется простой режим адресации.

Сверточный режим служит для реализации алгоритма свертки (КИХ-фильтра). В этом режиме регистры занимают перекрывающиеся адреса в памяти, т.е. например, скалярные элементы 1-7 регистра 0 - это то же самое, что скалярные элементы 0-6 регистра 1. В сверточные сегменты, в отличие от скалярных, можно производить запись; однако для загрузки и выгрузки данных следует использовать простой режим адресации.

Соотношение между тремя вышеописанными режимами показано на рис. 4. Адреса, указанные в ячейках, сформированы по формуле «Банк + 8 * Адрес-в-банке».

Рис. 4. Соответствие адресов в памяти регистрам в простом, скалярном и сверточном режимах адресации

B. Матричные режимы

Матрицы в векторной памяти размещаются таким образом, что возможна параллельная адресация векторных элементов строк или столбцов матрицы. Это размещение известно как скошенная адресация (англ. skewed addressing) и достаточно давно применялось в некоторых процессорах на аппаратном уровне [8]. Подобная схема используется и при транспонировании матриц на современных графических процессорах (GPU) [9]. Отличие используемой в ВКС схемы размещения матриц в том, что она не приводит к образованию неиспользуемых ячеек памяти, работает для матриц произвольного размера (который округляется вверх до ближайшей степени двойки), и легко реализуется аппаратно без необходимости формировать адреса программно и использовать специальные устройства для обнаружения конфликтов доступа к банкам памяти.

Размещение элементов матрицы размерностью 8x8 в 8-банковой памяти показано на рис. 5. Строки матрицы обозначены буквами от A до H, столбцы – цифрами от 0 до 7. Первый столбец выделен серым цветом фона. Если количество столбцов матрицы больше, чем количество банков памяти, то векторные элементы каждой строки записываются по последовательным адресам, т.е. на рис. 5 одна строка матрицы займет несколько последовательных строк в памяти с одинаковым сдвигом.

Прямой матричный режим позволяет поставить в соответствие регистрам строки матрицы. Один векторный элемент регистра располагается по одинаковому адресу во всех банках памяти. Значение сдвига, подаваемое на блоки ROT, соответствует номеру строки матрицы, взятому по модулю количества банков памяти.

Транспонированный матричный режим позволяет поставить в соответствие регистрам столбцы матрицы. Один векторный элемент регистра располагается по последовательным адресам во всех банках памяти. Значение сдвига соответствует номеру столбца матрицы, взятому по модулю количества банков памяти.

Адреса \ Банки	0	1	2	3	4	5	6	7
0	A0	A1	A2	A3	A4	A5	A6	A7
1	B7	B0	B1	B2	B3	B4	B5	B6
2	C6	C7	C0	C1	C2	C3	C4	C5
3	D5	D6	D7	D0	D1	D2	D3	D4
4	E4	E5	E6	E7	E0	E1	E2	E3
5	F3	F4	F5	F6	F7	F0	F1	F2
6	G2	G3	G4	G5	G6	G7	G0	G1
7	H1	H2	H3	H4	H5	H6	H7	H0

Рис. 5. Размещение элементов матрицы размерностью 8x8 в векторной памяти с 8 банками

Таким образом, для транспонирования матриц достаточно лишь изменить режим адресации. Загрузка и выгрузка данных может осуществляться как в прямом, так и в транспонированном матричных режимах.

IV. Реализация БПФ

Реализация БПФ для предложенной архитектуры ВКС основывается на разложении исходного вектора на 2 измерения и выполнении преобразований сначала по одному, а потом по другому измерению [10]. То есть, например, вектор размерностью 128 точек представляется в виде матрицы 8x16, и производятся БПФ малого размера: 8 преобразований на 16 точек, затем поэлементное умножение всей матрицы на матрицу коэффициентов, транспонирование и 16 преобразований на 8 точек в другом направлении. По каждому измерению группы производимых преобразований независимы, что делает такой алгоритм удобным для реализации на процессорах с поддержкой SIMD-инструкций.

Такой алгоритм применяется и для других архитектур, включая DSP (описание его реализации для процессора TigerSHARC дано в работе [11]) и графические процессоры [3], [12]. Этот же алгоритм применяется при реализации БПФ на векторно-конвейерном вычислительном устройстве NeuroMatrix NM6403 производства ЗАО «НТЦ Модуль» [13]. Но в отличие от ВКС, элементарные преобразования (в примере выше – на 8 и 16 точек) производятся на NM6403 в виде ДПФ, а не БПФ, что приводит к значительному снижению быстродействия.

При реализации этого алгоритма с помощью SIMD-процессоров возникает проблема с транспонированием матрицы: либо оно выполняется с помощью соответствующей последовательности команд, работающих над скалярными данными, что приводит к значительному снижению производительности, либо необходимо использование специальной памяти и дополнительных аппаратных средств, позволяющих векторизовать этот процесс. Использование конвертируемых режимов адресации памяти полностью устраняет проблему транспонирования: оно сводится к изменению режима адресации сегмента регистров.

Достоинством данного алгоритма является отсутствие необходимости обращать порядок всех битов номера элемента вектора при загрузке или выгрузке данных. Достаточно при загрузке обратить порядок битов в номерах векторных регистров при загрузке и выгрузке данных, не меняя порядок самих данных – это реализуется аппаратно в блоках ввода-вывода.

Время выполнения БПФ размерностью от 64 до 4096 точек $T_{БПФ}$, полученное для RTL-модели ВКС с 4 и с 8 конвейерами, представлено на рис. 6. Приведено количество тактов, в течение которых выполнялась вычислительная программа, т.е. время передачи данных в/из сопроцессора не учитывается. Точное время вычисления указано цифрами: верхний ряд – для ВКС с 4 конвейерами, нижний – для ВКС с 8 конвейерами.

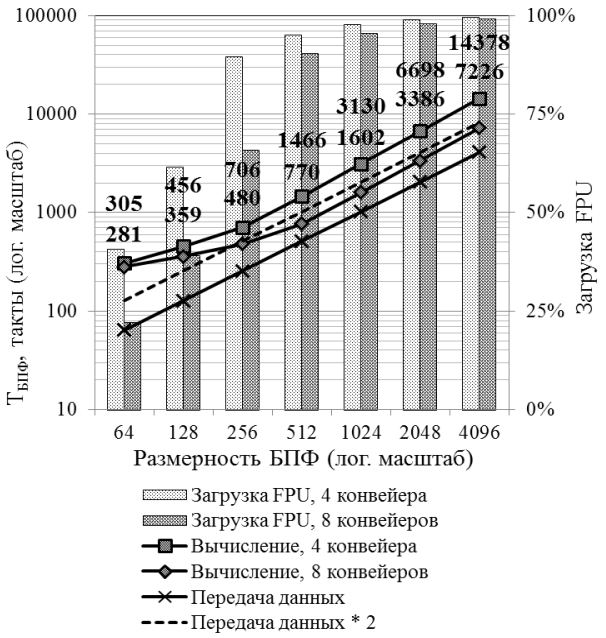

Рис. 6. Время выполнения БПФ и передачи данных; загрузка вычислительного блока (FPU)

На рис. 6 также показана загрузка вычислительного блока (измеряется в процентах) при выполнении соответствующих преобразований. Загрузка FPU измеряется как отношение числа тактов, в которых на вход конвейеров FPU поступали новые данные, к общему времени выполнения вычислительной программы. Видно, что при малой размерности преобразований загрузка невысока, а производительность вычислений ограничена латентностью инструкций: сопроцессор может решать эти задачи быстрее при реализации FPU с более коротким конвейером. На сопроцессор с 8 конвейерами латентность инструкций оказывает существенно большее воздействие, чем на сопроцессор с 4 конвейерами.

Для сравнения на рис. 6 показано время передачи данных в сопроцессор или из него $T_{ПЕР}$ при использовании 64-битных шин данных. Чтобы производительность сопроцессора определялась вычислительными процедурами, а не передачей данных, требуется, чтобы вычисления занимали больше времени, чем загрузка или выгрузка данных. Как видно из графиков, это условие всегда выполняется.

Если бы сопроцессор не поддерживал матричные режимы адресации, то вычисление БПФ пришлось бы прерывать транспонированием матрицы, выполняемым через внешнюю по отношению к ВКС память. В потоковом режиме это приведет только к двукратному увеличению нагрузки на шины данных, что показано пунктирной линией на рис. 6. В таком случае производительность потокового режима ВКС с 8 конвейерами будет ограничена передачей данных. Общее время выполнения (латентность, $T_{ЛАТ}$) одного БПФ-1024 с учетом передачи данных увеличится с 5178 (3130 + 1024*2) до 7226 (3130 + 1024*4) тактов на ВКС с 4 конвейерами, или с 3650 до примерно 5900 тактов на ВКС с 8 конвейерами.

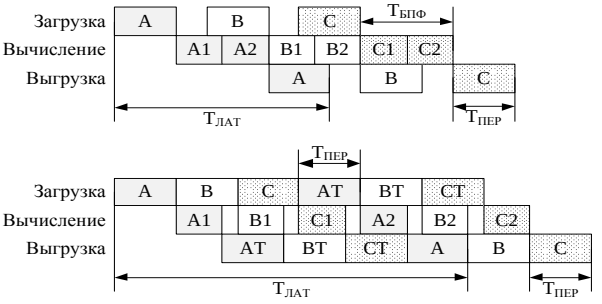

Рис. 7. Выполнение БПФ в потоковом режиме с использованием конвертируемых режимов адресации и без использования таких режимов

Выполнение БПФ в потоковом режиме с использованием конвертируемых (матричных) режимов адресации и без их использования показано на рис. 7. Для каждого случая показано выполнение трех независимых задач – A, B и C. Масштаб по горизонтальной оси (оси времени) соответствует выполнению БПФ-1024 на ВКС с 8 конвейерами, где $T_{БПФ}$ = 1602 такта, а $T_{ПЕР}$ = 1024 такта. Вычисление разделено на 2 примерно равные части, между которыми производится транспонирование: в первом случае оно сводится к переключению режима адресации и происходит очень быстро, а во втором случае оно требует выгрузки и загрузки данных, обозначенных как AT, BT или CT. Применение конвертируемых режимов адресации не влияет на время вычисления $T_{БПФ}$, но в 2 раза уменьшает нагрузку на внешние шины данных и уменьшает общую латентность операции $T_{ЛАТ}$.

Сравнение производительности БПФ на ВКС и ряде других процессоров и сопроцессоров приведено в табл. 1. Для вычисления количества обрабатываемых отсчетов в секунду использовалось БПФ на 1024 точки на большинстве процессоров. Измерения производительности GPU NVIDIA производились с использованием библиотеки CUFFT 4.2 на множестве из 16384 преобразований по 1024 точки.

Представленные в табл. 1 БПФ-процессоры являются специализированными устройствами, которые могут обрабатывать только БПФ определенного размера. Процессор, описанный в работе [4], ориентирован на FPGA и выполняет только БПФ-1024, а процессор, описанный в работе [5], ориентирован на применение в коммуникационных устройствах и выполняет только БПФ-2048. В отличие от этих узкоспециализированных устройств, ВКС является программируемым сопроцессором.

Таблица 1

Сравнение производительности БПФ

Процессор	**Частота, МГц**	**Млн. отсчетов в сек.**
ВКС, 8 конв.	1000	640
ВКС, 4 конв.	1000	327
ADI TigerSHARC [14]	600	66
TI TMS320C674x [15]	456	51
Модуль NM6406	300	17
Модуль NM6403 [13]	40	2.2
ЭЛВИС 1892ВМ5Я [7]	120	23
БПФ-процессор [4]	150	29
БПФ-процессор [5]	300	2400
NVIDIA GK104 (8 блоков SM)	1006	9720 (1215 на SM)

V. Реализация других алгоритмов

Умножение вектора размерности *M* на матрицу размерности *M*N* производится при помощи нескольких регистров-аккумуляторов размерности *N* элементов, в которых накапливаются частичные суммы. Алгоритмически для выполнения этой задачи достаточно одного аккумулятора, но в таком случае при малых значениях *N* не обеспечивается покрытие латентности вычислений, и сопроцессор часть времени простаивает, ожидая, пока результаты предыдущей инструкции будут вычислены и записаны в память. Использование нескольких аккумуляторов существенно снижает минимальное значение *N*, при котором происходит насыщение вычислительного конвейера, ценой дополнительных инструкций, требуемых для сложения частичных аккумуляторов в один. На ВКС с 4 конвейерами умножение вектора из 32 комплексных чисел на матрицу размером 32*32 элемента занимает 570 тактов при загрузке FPU в 91%.

Свертка, или КИХ-фильтр, реализуется при помощи сверточного режима адресации. Один входной вектор (B) хранится в сегменте с прямой адресацией; второй входной вектор (C) хранится в сегменте со скалярной адресацией. Вектор результата вычисления свертки (A) хранится в сегменте со сверточной адресацией и используется как аккумулятор. Программа свертки производит умножение вектора B на каждый элемент вектора C, добавляя результат умножения в аккумулятор A по смещающейся на 1 элемент позиции, что можно описать следующим векторным выражением:

$$\mathbf{A}n \mathrel{+}= \mathbf{B} * Cn.$$

На ВКС с 4 конвейерами свертка комплексных векторов размерностью 8 и 128 элементов выполняется за 548 тактов при загрузке FPU в 93%. Для больших размерностей векторов целесообразно применять теорему о свертке и рассчитывать эту операцию с помощью БПФ.

VI. Результаты синтеза

Результаты синтеза разработанной RTL-модели ВКС с 4 конвейерами, 96 КБ внутренней памяти данных и 4 КБ памяти кода представлены в табл. 2.

Таблица 2

Результаты синтеза RTL-модели ВКС

Тех. норма	**40 нм**	**90 нм**
Макс. частота	1 ГГц	286 МГц
Площадь	1,88 мм^2	8,90 мм^2
Мощность	144,3мВт	327,0мВт

Чтобы показать влияние реализации конвертируемых режимов адресации на параметры кристалла, приведем распределение по основным блокам ВКС площади, занимаемой сопроцессором на кристалле и потребляемой мощности (табл. 3). Режимы адресации реализуются при помощи блоков генерации адресов и циклического сдвига, которые в сумме занимают около 1.7% от площади всего ВКС и потребляют 2.2% мощности. Эти данные показывают, что реализация конвертируемых режимов адресации не вызывает существенного возрастания энергопотребления ВКС и площади, занимаемой на кристалле СнК.

Таблица 3

Распределение площадей и потребляемых мощностей между компонентами ВКС

Компонент	**Площадь**	**Мощность**
Блоки памяти	85%	40%
Вычислительный блок	8%	45%
Блоки генерации адресов	1.7%	2.2%
Блоки циклического сдвига	< 0.1%	< 0.1%

VII. Заключение

Применение векторной памяти с поддержкой конвертируемых режимов адресации является целесообразным решением при проектировании высокопроизводительных процессоров и сопроцессоров, предназначенных для цифровой обработки сигналов. Выгода от применения конвертируемых режимов адресации заключается в упрощении адресации памяти со стороны программы, возможности не тратить ресурсы на транспонирование матриц, и, как следствие, приросте производительности алгоритмов.

Представленный в статье векторно-конвейерный сопроцессор, использующий конвертируемую адресацию памяти, позволяет выполнять типичные для цифровой обработки сигналов задачи, такие как БПФ, КИХ-фильтрация, умножение вектора на матрицу, с высокой производительностью и эффективностью. При высокой размерности этих задач загрузка вычислительных блоков составляет 80 - 90% и более.

Литература

[1] Valero M., Lang T., Ayguadé E. Conflict-free access of vectors with power-of-two strides // 6th international conference on Supercomputing. New York : ACM. 1992.

[2] Shahbahram A., Juurlink B., Vassiliadis S. Efficient Vectorization of the FIR Filter // 16th Annual Workshop on Circuits, Systems and Signal Processing (ProRISC). 2005.

[3] Gutierrez E., Romero S. и др. Memory Locality Exploitation Strategies for FFT on the CUDA Architecture // High Performance Computing for Computational Science - VECPAR 2008. Springer Berlin / Heidelberg.

[4] Patil B., Patil G. и др. FPGA Based Design of a High Speed 32-Bit Floating Point FFT Processor with Performance Comparison with traditional FFT // International Journal of Computer Applications. 2010. Vol. 1. № 29.

[5] Song-Nien Tang, Jui-Wei Tsai, Tsin-Yuan Chang A 2.4-GS/s FFT Processor for OFDM-Based WPAN Applications // IEEE Transactions on Circuits and Systems II: Express Briefs. 2010. Vol. 57. № 6.

[6] Зубковский П.С., Ивасюк Е.В., Аряшев С.И. Сопроцессор комплексных вычислений // Проблемы разработки перспективных микро- и наноэлектронных систем - 2010. Сборник трудов. М.:ИППМ РАН, 2010. С. 356-359.

[7] Солохина Т., Александров Ю. и др. Отечественные трехъядерные сигнальные микроконтроллеры с производительностью 1.5 GFLOPS // Электронные компоненты. 2006. №6. С. 73–78.

[8] Kuck D., Stokes R. The Burroughs Scientific Processor (BSP) // IEEE Transactions on Computers. May 1982. Vol. C-31. № 5.

[9] Ruetsch G., Micikevicius P. Optimizing Matrix Transpose in CUDA. URL: http://developer.download.nvidia.com/compute/cuda/3_0/sdk/website/CUDA/website/C/src/transposeNew/doc/MatrixTranspose.pdf (дата обращения: 19.01.2012).

[10] Burrus C.S. Fast Fourier Transforms // Houston, Texas: Connexions, 2008.

[11] Lerner B. Parallel Implementation of Fixed-Point FFTs on TigerSHARC® Processors. URL: http://www.eetasia.com/ARTICLES/2005JUN/A/2005JUN01_CTRLD_AN01.PDF?SOURCES=DOWNLOAD (дата обращения: 19.01.2012).

[12] Govindaraju N., Lloyd B. и др. High Performance Discrete Fourier Transforms on Graphics Processors // Proceedings of the 2008 ACM/IEEE conference on Supercomputing.

[13] Кашкаров В.А., Мушкаев С.В. Организация параллельных вычислений в алгоритмах БПФ на процессоре NM6403 // Цифровая обработка сигналов. 2001. №1.

[14] TigerSHARC Processor Benchmarks. URL: http://www.analog.com/en/processors-dsp/TigerSHARC/processors/TigerSHARC_benchmarks/fca.html (дата обращения: 23.01.2012).

[15] TMS320C674x Low Power DSP Benchmarks. URL: http://www.ti.com/dsp/docs/dspplatformscontento.tsp?sectionId=2&familyId=1622&tabId=2431 (дата обращения: 26.04.2012).