]Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОИНКИ
(БГУИР)

УДК 621.391; 519.72; 621.391.037.372
681.323; 681.324
№ госрегистрации 20122058
Инв. №

УТВЕРЖДАЮ
Проректор по научной работе БГУИР
д-р техн. наук, проф.
_____А. П. Кузнецов
«__»_____2013 г.

ОТЧЕТ
О НАУЧНО-ИССЛЕДОВАТЕЛЬСКОЙ РАБОТЕ

БЫСТРОЕ ПРОТОТИПИРОВАНИЕ ДИНАМИЧЕСКИ РЕКОНФИГУРИРУЕМЫХ ПОТОЧНЫХ ПРОЦЕССОРОВ ПАКЕТНОГО ДИСКРЕТНОГО ВЕЙВЛЕТ ПРЕОБРАЗОВАНИЯ

(заключительный)

ГБЦ № 12-3132

Науч. руководитель
канд. техн. наук, доц. каф ЭВМ _____ Ал. А. Петровский

Отв. исполнитель
аспирант кафедры ЭВС _____ М.М. Родионов

Минск 2013

СПИСОК ИСПОЛНИТЕЛЕЙ

Руководитель темы,
канд. техн. наук _____ Ал. А. Петровский (введение, заключение)

Отв. исполнитель _____ М.М. Родионов (разделы 1-4)

Нормоконтролер _____ Л. А. Шичко

РЕФЕРАТ

Отчет 37 с., 1 кн., 18 рисунков, 4 таблицы, 10 источников.

ПАКЕТНОЕ ДИСКРЕТНОЕ ВЕЙВЛЕТ ПРЕОБРАЗОВАНИЕ, БАНК ФИЛЬТРОВ, ПРОЦЕССОР, БЫСТРОЕ ПРОТОТИПИРОВАНИЕ, ПРОГРАММИРУЕМАЯ ЛОГИЧЕСКАЯ ИНТЕГРАЛЬНАЯ СХЕМА, FPGA.

Объектом исследования является динамически реконфигурируемый поточный процессор пакетного дискретного вейвлет преобразования. Целью работы является разработка метода быстрого прототипирования архитектур динамически реконфигурируемых процессоров пакетного дискретного вейвлет преобразования.

При выполнении НИР была предложена оригинальная архитектура динамически реконфигурируемого процессора ПДВП, особенностью которой является высокое быстродействие за счет использования поточной архитектуры на лестничных структурах, что при фреймовой обработке данных обеспечивает хороший запас по времени на субполосную обработку. Разработан метод быстрого прототипирования динамически реконфигурируемых процессоров ПДВП по заданной спецификации приложения. Написана библиотека для расчета и моделирования работы алгоритма на фиксированной запятой переменного формата, а также реализовано параметризированное VHDL-описание цифровых блоков процессора

Полученный в результате НИР метод позволит реализовывать эффективные системы обработки мультимедиа данных. Результаты данного исследования могут быть использованы в системах адаптивного шумоподавления, задачах сжатия мультимедиа данных при реализации аппаратных блоков анализа и обработки звуковых сигналов и изображений.

СОДЕРЖАНИЕ

Введение 7

1 Разработка архитектурных решений для построения процессоров ПДВП с учетом специфики целевых мультимедиа приложений 8

1.1 Разработка набора требований к характеристикам процессора ПДВП 8

1.2 Анализ существующих архитектурных решений 9

1.2.1 Общие подходы и проблемы при проектировании процессоров вычисления вейвлет преобразований 9

1.2.2 Двухканальный банк фильтров с использованием лестничных структур 11

1.3 Поточный процессор с динамически реконфигурируемой структурой вычислений 13

1.4 Реализация двухканального банка фильтров анализа/синтеза 15

1.5 Организация блоков памяти 19

1.6 Выводы по разделу 20

2 Разработка метода быстрого прототипирования архитектур процессора 22

2.1 Параметрическое представление цифровых блоков процессора 22

2.2 Метод быстрого прототипирования архитектур процессора 26

2.3 Выводы по разделу 27

3 Инструментальная среда для быстрого прототипирования динамически реконфигурируемых процессоров ПДВП 28

3.1 Параметризированное VHDL описание процессора 28

3.1.1 Пример поточного процессора ПДВП аппроксимирующего шкалу барков 32

4 Анализ параметров качества быстрого прототипирования процессоров ПДВП . 34

4.1 Анализ ошибок вычислений 34

4.2 Анализ свойств сохранения энергии. 35

4.3 Выводы по разделу 36

заключеие 37

Список использованных источников 38

ВВЕДЕНИЕ

Среди различных подходов по реализации многополосной фильтрации особое место занимает алгоритм пакетного дискретного вейвлет преобразования. Вейвлет преобразование, получившее значительное распространение в прикладной математике, находит все новые области применения. Оно позволяет обеспечить большую гибкость при частотно-временной декомпозиции сигналов за счет выбора между многочисленными базисами, банками фильтров и деревьями разложения. Таким образом, наблюдается большой интерес исследователей к данному преобразованию, что, как следствие, приводит к появлению новых алгоритмов, моделей на базе вейвлет декомпозиции. В свою очередь возникает целый ряд нерешенных задач, касающихся эффективной реализации данных алгоритмов. При этом значительный успех достигается не только разработкой аппаратного вычислителя для конкретного приложения, а и проектированием параметризированной архитектуры, закрывающей целый класс задач. Это позволяет использовать полученные результаты для готовых приложений, а также для исследований при разработке новых подходов. Важной задачей также является получение методов и средств быстрого прототипирования встраиваемых процессоров на основе исходного параметризированного описания архитектуры.

В данной работе предлагается метод быстрого прототипирования динамически реконфигурируемых поточных процессоров пакетного дискретного вейвлет преобразования. В данном отчете основные результаты НИР в следующей последовательности. В разделе 1 вначале сформулированы основные требования к результатам НИР и проведен анализ научных подходов по проектированию процессоров вычисления ПДВП. В результате предложена оригинальная архитектура динамически реконфигурируемого процессора ПДВП на арифметике с фиксированной запятой. В разделе 2 рассмотрен метод быстрого прототипирования предложенной архитектуры в виде параметризированного описания цифровых блоков процессора. В разделе 3 изложены практические аспекты реализации инструментальной среды быстрого прототипирования в среде MATLAB. В 4 разделе представлен анализ некоторых параметров качества полученных архитектур и приведены результаты моделирования в инструментальной среде. В конце каждого раздела даны краткие выводы касательно полученных результатов.

1 Разработка архитектурных решений для построения процессоров ПДВП с учетом специфики целевых мультимедиа приложений

1.1 Разработка набора требований к характеристикам процессора ПДВП

В ходе выполнения НИР был произведен анализ характеристик, относящихся к аппаратным реализациям алгоритмов ПДВП в целевых приложениях. В результате сформированы две группы взаимоисключающих требований, предъявляемые в качестве спецификации к разработке. Первая группа требований (функциональная спецификация): базисная вейвлет функция; структура предельного дерева декомпозиции; необходимость динамической реконфигурации структуры дерева декомпозиции; уровень вносимой ошибки в результаты вычислений; пропуская способность. Вторая группа (спецификация на ресурсы системы): ограничения аппаратных затрат на реализацию процессора; максимальная потребляемая мощность системы, сроки разработки системы. Представленный набор требований вызывает большие сложности при реализации, поскольку улучшение (удовлетворение) параметров первой группы приводит к резкому ухудшению характеристик второй группы. Основываясь на данных положениях, был разработан следующий набор требований к характеристикам аппаратной реализации алгоритмов ПДВП на лестничных структурах.

Модульность архитектуры. Данное требование подразумевает разбиение архитектуры на независимые аппаратные модули, с целью лучшей систематизации структуры проекта. Это позволит сосредоточиться на эффективной реализации отдельных компонентов системы, и при этом не потребует значительных модификаций интерфейсов обмена данными при последующих модификациях (улучшениях) некоторых модулей.

Параметризация характеристик. Данное требование приведет к масштабируемости архитектуры, позволит быстро менять заданные параметры системы без необходимости повторного проектирования блоков системы. Данное преимущество позволит модифицировать некоторые функциональные возможности реализации на последних этапах разработки и даже в процессе тестирования системы.

Эффективная реализация отдельных вычислительных модулей системы. Данное требование подразумевает эффективное отображение элементов исходного алгоритма на целевую аппаратную платформу. При этом нельзя забывать и о максимальной параметризации схемных решений.

Таким образом, для оценки архитектурных решений, полученных в ходе выполнения НИР, можно выделить два критерия:
– получение параметризируемой архитектуры аппаратной реализации алгоритма ПДВП, позволяющей гибкое изменение параметров системы без дополнительных затрат времени со стороны разработчика;
– лучшие показатели по основным характеристикам (быстродействие, точность вычислений, аппаратные затраты) по сравнению с аналогичными разработками (в противном случае даже успешное выполнение первого критерия не позволит говорить о положительных результатах, поскольку введение параметризации имеет зачастую малую ценность при неэффективности конечной реализации).

1.2 Анализ существующих архитектурных решений

1.2.1 Общие подходы и проблемы при проектировании процессоров вычисления вейвлет преобразований

Существующие аппаратные решения по реализации алгоритма пакетного дискретного вейвлет преобразования (ПДВП) зависят от ряда факторов: глубина дерева декомпозиции, базисная вейвлет функция (в частности порядок вейвлет фильтра), точность вычислений (арифметика на фиксированной или плавающей запятой), пропускная способность, ограничения аппаратного ресурса, необходимость динамической реконфигурации дерева преобразования [[1]].

Вычислительным ядром большинства представленных в современных работах архитектур является двухканальный банк фильтров, реализованный на основе лестничных структур [[2],[3]]. Данная методика позволяет уменьшить число вычислительных операций почти в два раза.

Для алгоритмов с небольшим числом уровней декомпозиции (до трех) и малыми порядками вейвлет фильтров, применяемых в основном для обработки графики, на сегодняшний день представлено много работ, в полной мере закрывающих проблему. В данных работах в меньшей степени проявляются некоторые сложности аппаратных реализаций вейвлет преобразований.

Другим направлением использования ПДВП являются приложения обработки звука и речи. К особенностям вейвлет преобразования в таких задачах можно отнести большое число уровней декомпозиции и высокие порядки вейвлет фильтров по срав-

нению с приложениями для обработки графики и видео. Так, в работе [[4]] предлагается архитектура на основе фильтров с конечной импульсной характеристикой (КИХ) для вычисления ПДВП с поиском лучшего дерева на основе [[5]]. Данный подход предполагает вычисление полного пакета преобразования, что приводит к увеличению вычислительных затрат по сравнению с предлагаемой архитектурой, реализующей динамически перестраиваемый алгоритм ПДВП. В работе [[6]] представлена поточная архитектура для вычисления ПДВП на основе лестничных структур с использованием целочисленной арифметики, обладающая высокой пропускной способностью и использующая малый ресурс памяти. К недостаткам работы можно отнести тот факт, что разработанная архитектура способна вычислять только полный пакет вейвлет преобразования, а это делает ее затратной с точки зрения аппаратного и вычислительного ресурса для приложений обработки звука и речи с большим числом уровней декомпозиции, не требующих вычисления полного дерева.

Представленные результаты в той или иной степени удовлетворяют заданным требованиям, хотя имеются определенные недостатки, связанные, прежде всего, с направленностью существующих подходов на решение узких задач, что в конечном итоге не позволяет применять данные реализации при некоторых изменениях в требованиях к приложениям без дополнительных затрат на проектирование.

Таким образом, можно сделать выводы касательно аппаратных решений при реализации алгоритмов ПДВП: в большинстве случаев предпочтительной является арифметика с фиксированной запятой, позволяющая при требуемой точности вычислений значительно сократить аппаратные затраты; для уменьшения вычислительных операций целесообразно использовать подход на лестничных структурах; для задач, требующих высокой пропускной способности, наиболее эффективной является конвейерная архитектура, в которой на каждой ступени реализуется один процессорный элемент для вычисления базовой декомпозиции алгоритма на низкочастотную и высокочастотную составляющие. Особое внимание стоит отметить на реализацию алгоритма с использованием лестничных структур на арифметике с фиксированной запятой. Постоянные коэффициенты в данном подходе имеют больший динамический диапазон по отношению к исходному представлению (особенно это сильно проявляется при факторизации вейвлет фильтров с большим порядком), что требует

дополнительных архитектурных решений при реализации на арифметике с фиксированной запятой.

В данной работе предложен метод быстрого прототипирования встраиваемых процессоров вычисления ПДВП анализа/синтеза на лестничных структурах на основе вектора исходных параметров, определяющих базисную вейвлет функцию; структуру предельного дерева декомпозиции; уровень ошибки, вносимый в результаты вычислений; ограничения аппаратных ресурсов на реализацию процессора; необходимость динамической реконфигурации структуры дерева декомпозиции. На выходе метода формируется параметризированный VHDL код блоков процессора, полностью реализующий прототип по заданным требованиям.

1.2.2 Двухканальный банк фильтров с использованием лестничных структур

В алгоритме ПДВП для разложения сигнала на НЧ и ВЧ компоненты в каждом узле дерева необходимо выполнить математическую операцию свертки входного сигнала с импульсными характеристиками вейвлет фильтров $\tilde{h}(z)$ и $\tilde{g}(z)$. Для сокращения вычислительных затрат в данной работе применяется двухканальный банк фильтров на основе лестничных структур [[2]] (в англ. литературе lifting scheme либо ladder structure). В терминах z-преобразования переход к реализации банка фильтров на основе лестничных структур можно рассматривать в два этапа.

Первый этап заключается в переходе к полифазной реализации алгоритма фильтрации [[3]]. При этом процесс вычисления НЧ и ВЧ компонент сигнала $x_{l,n,k}$ в произвольном узле дерева можно записать в виде следующего выражения:

$$[X_{l+1,2n}(z) \quad X_{l+1,2n+1}(z)] = [X^{e}_{l,n}(z) \quad z^{-1}X^{o}_{l,n}(z)] \cdot \tilde{\mathbf{P}},$$
$$\tilde{\mathbf{P}} = \begin{bmatrix} \tilde{h}_e(z) & \tilde{g}_e(z) \\ \tilde{h}_o(z) & \tilde{g}_o(z) \end{bmatrix}, \qquad (1)$$

где $X_{l+1,2n}(z)$ и $X_{l+1,2n+1}(z)$ – представления в z-области НЧ и ВЧ компонент; $X^{e}_{l,n}(z)$, $X^{o}_{l,n}(z)$ – представления последовательностей, состоящих соответственно из четных (even) и нечетных (odd) отсчетов входной последовательности $x_{l,n,k}$; $\tilde{\mathbf{P}}$ – полифазная матрица, элементы которой выступают в следующей зависимости по отношению к исходным вейвлет-фильтрам:

$$\tilde{h}(z) = \tilde{h}_e(z^2) + z^{-1}\tilde{h}_o(z^2),$$
$$\tilde{g}(z) = \tilde{g}_e(z^2) + z^{-1}\tilde{g}_o(z^2) \qquad (2)$$

Данный подход позволяет сократить вычислительные затраты в два раза, и понизить частоту обработки данных вдвое по отношению к частоте подачи исходного сигнала (см. рисунок 1).

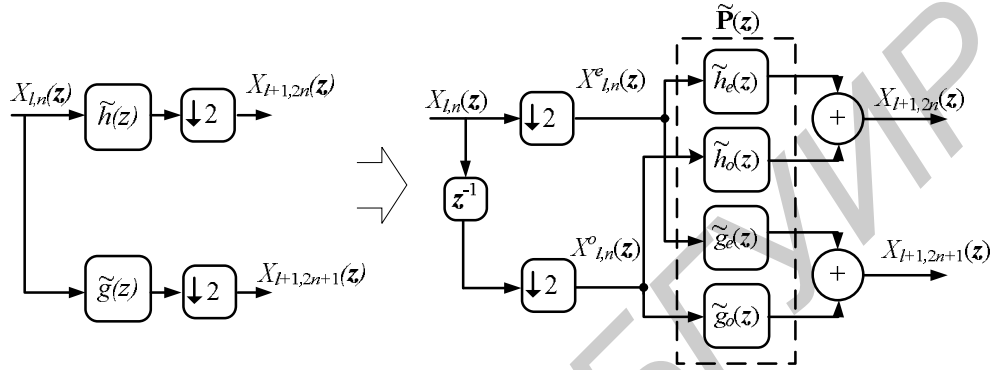


Рисунок 1 – Переход к полифазной реализации банка фильтров анализа

Второй этап представляет собой факторизацию полифазной матрицы на более простые треугольные матрицы, в результате чего в общем случае исходная матрица $\tilde{\mathbf{P}}$ может быть представлена в виде выражения

$$\tilde{\mathbf{P}} = \prod_{i=1}^{\frac{I}{2}} \left(\begin{bmatrix} 1 & \tilde{s}_i(z) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \tilde{t}_i(z) & 1 \end{bmatrix} \right) \begin{bmatrix} c_1 & 0 \\ 0 & c_2 \end{bmatrix}, \qquad (3)$$

где I – число элементарных треугольных матриц, полученных в результате факторизации полифазной матрицы; $\tilde{s}_i(z)$ и $\tilde{t}_i(z)$ – полиномы малого порядка c_1, c_2 – вещественные коэффициенты. Ниже приведена блок-схема реализации двухканального банка анализа на основе лестничных структур (рисунок 2).

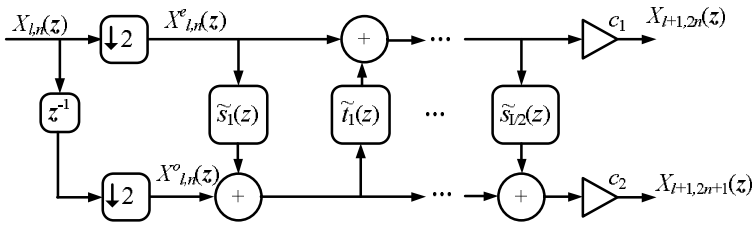


Рисунок 2 – Блок-схема двухканального банка фильтров анализа на основе лестничных структур

В результате факторизации полифазной матрицы общий вид для полиномов $\tilde{s}_i(z)$ и $\tilde{t}_i(z)$ может быть сведен к выражению вида $(b_i^0 + b_i^1 z^{-1})z^u$; где b_i^0, b_i^1 – вещественные коэффициенты, u – целое значение степени z.

При эффективной факторизации матрицы $\tilde{\mathbf{P}}$ двухканальный банк анализа на лестничных структурах может содержать число операций умножения и сложения почти вдвое меньше по сравнению с полифазной реализацией [[3]].

Реализация двухканального банка синтеза с учетом рассмотренной структуры банка анализа также может быть построена на основе лестничных структур. При этом в матричной форме с учетом (1),(3) процесс получения восстановленного сигнала получается, как

$$\left[\hat{X}^e_{l,n}(z) \quad z^{-1}\hat{X}^o_{l,n}(z)\right] = \left[X_{l+1,2n}(z) \quad X_{l+1,2n+1}(z)\right]\begin{bmatrix}\frac{1}{c_1} & 0\\ 0 & \frac{1}{c_2}\end{bmatrix}\prod_{i=\frac{l}{2}}^{1}\left(\begin{bmatrix}1 & 0\\ -\tilde{t}_i(z) & 1\end{bmatrix}\begin{bmatrix}1 & -\tilde{s}_i(z)\\ 0 & 1\end{bmatrix}\right). \qquad (4)$$

Блок-схема двухканального банка синтеза на основе лестничных структур представлена на рисунке 3.

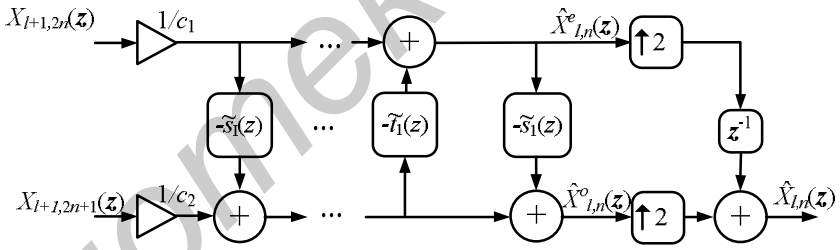


Рисунок 3 –Блок-схема двухканального банка фильтров синтеза

Согласно представленной блок-схеме процедура синтеза реализуется следующим образом. Вначале выполняется умножение на коэффициенты $\frac{1}{c_1}$, $\frac{1}{c_2}$ в каждом канале. Последующие ступени банка синтеза реализуют операции обратные по отношению к операциям алгоритма анализа. При этом используются те же полиномы $\tilde{s}_i(z)$ и $\tilde{t}_i(z)$ из (3), только с противоположными знаками. Восстановленный сигнал $\hat{X}_{l,n}(z)$ получается из рассчитанных последовательностей $\hat{X}^e_{l,n}(z)$, $\hat{X}^o_{l,n}(z)$.

1.3 Поточный процессор с динамически реконфигурируемой структурой вычислений

Предлагается поточная архитектура процессора, представленная на рисунке 4. В описании структуры приняты следующие обозначения: ВБ – вычислительный блок двухканального банка фильтров анализ (синтеза), выполняющий обработку на одном уровне декомпозиции дерева; БП – блок памяти для хранения обрабатываемого сигнала и рассчитанных вейвлет коэффициентов; БУ – блок управления, выполняет функции синхронизации работы ВБ, осуществляет перенастройку системы при динамической реконфигурации дерева ПДВП. В представленной системе все блоки памяти реализуют двойную буферизацию данных (в англ. терминологии ping-pong), при которой данные с предыдущего вычислительного блока записываются в первый буфер и одновременно из второго буфера выдаются данные на следующий уровень декомпозиции. При обработке следующего фрейма буферы меняются местами. Таким образом, устраняется возможность перезаписи данных до их использования на следующем уровне декомпозиции. В процессе обработки входной фрейм накапливается в буфер $БП_0$ после чего сохраненные данные подаются на первый уровень декомпозиции. Для реализации динамически реконфигурируемой системы процессор цифровой обработки сигналов (ПЦОС) по рассчитанным вейвлет коэффициентам принимает решение о дальнейшем росте дерева, после чего данный фрейм может поступить на обработку на второй уровень преобразования. В этот же момент времени на первый уровень поступает новый фрейм данных. Таким образом, в системе реализуется мультифреймовая обработка, при которой декомпозиция (реконструкция) очередного фрейма производится «не дожидаясь» полного завершения обработки ранее поступивших данных. Кроме того в случае сигналозависимой (динамической) перестройки фрейм на определенном уровне l обрабатывается в соответствии с деревом декомпозиции, сформированном на основе анализа фреймов, обрабатываемых на более высоких уровнях. Данная схема позволяет достичь большой гибкости при построении конечных приложений. Структура дерева декомпозиции задается ПЦОС через управляющий интерфейс в блок управления(рисунок 4).

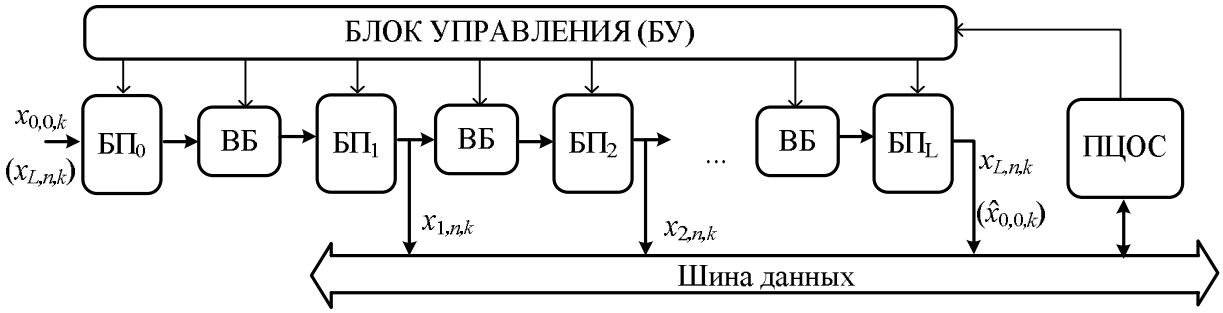


Рисунок 4 – Архитектура поточного процессора ПДВП анализа (синтеза)

1.4 Реализация двухканального банка фильтров анализа/синтеза

Ряд требований к целевому приложению (работа в реальном масштабе времени, большая пропускная способность, и другие) приводит к необходимости применения арифметики с фиксированной запятой для выполнения заданных вычислений. При реализации двухканального банка фильтров на основе лестничных структур с использованием целочисленной арифметики возникает ряд трудностей, связанных с тем, что значения коэффициентов полиномов $\tilde{s}_i(z)$ и $\tilde{t}_i(z)$ могут принимать как дробные, так и достаточно большие целые значения. Данная особенность приводит к увеличению разрядности арифметических блоков и внутренних регистров банка фильтров. Поэтому в данной работе для реализации алгоритма ПДВП на арифметике с фиксированной запятой использован подход [[8],[9]], в соответствии с которым формат представления чисел, участвующих в промежуточных вычислениях, является переменным. Данный метод подразумевает, что число битов, отводимых под целую и дробную части числа, в различных узлах алгоритма отличается.

В соответствии с данным подходом любое число, представленное в формате с фиксированной запятой в дополнительном коде, задается в виде выражения:

$$a = ma \cdot 2^{\exp_a}, \text{где } ma = (-1)^s + \sum_{i=0}^{wl-2} a_i \cdot 2^{i-wl+1}. \quad (5)$$

Здесь ma –значение числа, представленное в дополнительном коде, интерпретируемое как дробное в диапазоне $[-1,1)$; $\exp_1 a$ – порядок масштабирующего множителя

; a_i – значение *i*-го бита числа, равное 0 либо 1; s – знаковый бит; wl – разрядность слова. При этом для промежуточных данных в различных участках алгоритма используется свое значение $\exp_1 a$. Таким образом, в зависимости от значения $\exp_1 a$ происходит перераспределение бит для кодирования целой и дробной части (рисунок 7).

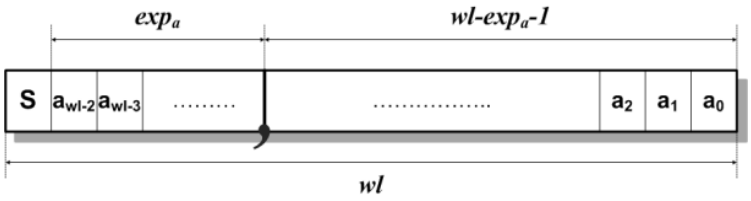


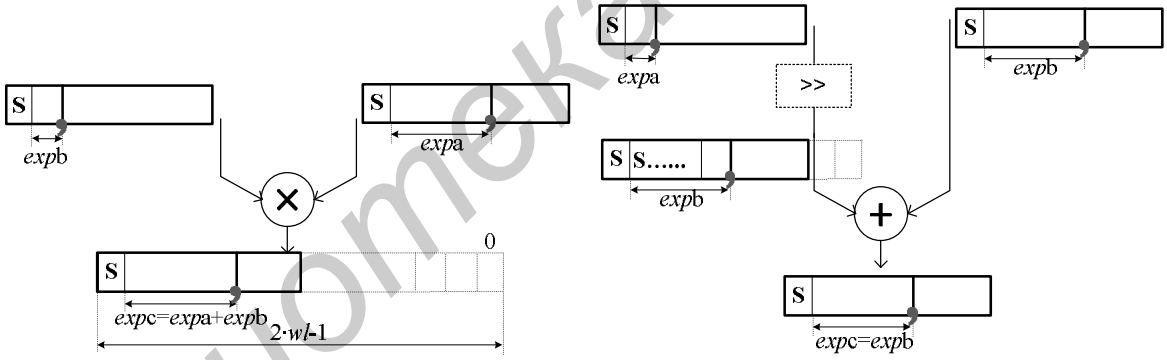
Рисунок 5 – Представление числа в арифметике с фиксированной запятой

Для заданного в (5) формата данных операции сложения и умножения чисел a и b определяются как

$$c = a + b = mc \cdot 2^{\uparrow}(\exp_{\downarrow} c) = (ma \cdot 2^{\uparrow}(\exp_{\downarrow} a - \exp_{\downarrow} b) + mb) \cdot 2^{\uparrow}(\exp_{\downarrow} b), \tag{6}$$

$$c = a \cdot b = mc \cdot 2^{\uparrow}(\exp_{\downarrow} c) = ma \cdot mb \cdot 2^{\uparrow}(\exp_{\downarrow} a + \exp_{\downarrow} b). \tag{7}$$

На рисунке 8 схематично проиллюстрирован процесс выполнения данных операций.



а) *б)*

Рисунок 6 – Выполнение операций: *а)* умножения; *б)* сложения

Таким образом, алгоритм на лестничных структурах с использованием арифметики переменного формата для процедур анализа и синтеза может быть реализован на основе вычислительных элементов, представленных в таблице 1. В данной таблице $x_в[k], x_о[k]$ – входные значения, а $y_в[k], y_о[k]$ – выходные значения соответственно верхнего и нижнего каналов банка фильтров анализа (синтеза) в k-ый интервал времени. Параметры $s0, s1, s2$ определяют значения для операций арифметических сдвигов. Данные параметры вычисляются с учетом (6) для каждого узла алгоритма.

Таблица 1 – Типы вычислительных элементов для алгоритма на лестничных структурах с использованием арифметики переменного формата

Банк анализа		Банк синтеза	
Обозначение	Вычислительные операции	Обозначение	Вычислительные операции
S1	$y_e[k] = x_e[k]$; $y_o[k] = x_o[k] \gg s0 + [(x]_e[k] \cdot mb^0$	$\mathbf{S1^{-1}}$	$y_e[k] = x_e[k]$; $y_o[k] = \left(x_o[k] - [(x]_e[k] \cdot mb^0\right) \gg s1$
S2	$y_e[k] = x_e[k]$; $y_o[k] = x_o[k] \gg s0 + [(x]_e[k] \cdot mb^0$	$\mathbf{S2^{-1}}$	$y_e[k] = x_e[k]$; $y_1o\,[k] = (x_1o\,[k] - [(x]\ _1e\,[k] \cdot mb^1$
T1	$y_e[k] = x_e[k] \gg s0 + [(x]_o[k] \cdot mb^0$	$\mathbf{T1^{-1}}$	$- [(x]\ _1e\,[k-1] \cdot mb^1) \gg s2) \ll$ $y_e[k] = [(x]_e[k] - [(x]_e[k] \cdot mb^0) \gg s1$
T2	$y_o[k] = x_o[k]$; $y_e[k] = x_e[k] \gg s0 + [(x]_o[k] \cdot mb^0$	$\mathbf{T2^{-1}}$	$y_o[k] = x_o[k]$; $y_e[k] = \left(x_e[k] - \left(x_e[k] \cdot mb^0\right) \gg s1 -$
	$+x\ \ k-1\ \ mb\ \ \gg\ \ s2;$		$y_o[k] = x_o[k]$;
	$y_o[k] = x_o[k]$;		

Для разъяснения практических аспектов, связанных с применением предложенной арифметики, ниже рассмотрен пример двухканального банка анализа на базе материнской вейвлет функции db4[[10]]. В результате выполнения факторизации полифазной матрицы для заданного вейвлет базиса в среде MATLAB получено следующее выражение:

$$\tilde{\mathbf{P}} = \begin{bmatrix} 1 & b_1^0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ (b_2^0 + b_2^1 z^{-1})z & 1 \end{bmatrix} \begin{bmatrix} 1 & (b_3^0 + b_3^1 z^{-1})z^{-1} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ (b_4^0 + b_4^1 z^{-1})z^2 & 1 \end{bmatrix} \begin{bmatrix} 1 & b_5^0 z^{-2} \\ 0 & 1 \end{bmatrix} \begin{bmatrix} c_1 & 0 \\ 0 & c_2 \end{bmatrix}. \quad (8)$$

Значения коэффициентов b_m^n, а также их параметры mb и $expb$, рассчитанные в соответствии с (5), представлены в таблице 2.

Таблица 2 Параметры лестничной структуры для исходных вейвлет фильтров Db-4 (8 коэффициентов)

Номер шага ЛС, i	Тип	u	b_i^0			b_i^1		
			значение	mb	$expb$	значение	mb	$expb$
1	$s_1(z)$	0	-3,1029	-	2	0	0	-

				0,7757				
2	$t_2(z)$	1	-0,0763	-0,6104	-3	0,2920	0,5840	-1
3	$s_2(z)$	-1	5,1995	0,6499	3	-1,6625	-0,8313	1
4	$t_4(z)$	3	3,1769	0,7942	2	0,0379	0,6064	-4
5	$s_3(z)$	-3	0,3141	0,6282	-1	0	0	-

На рисунке 7 приведена блок-схема реализации двухканального банка фильтров анализа для рассматриваемого примера. В данной схеме кроме вычислительных элементов S1, S2, T2 (см. таблицу 1) в верхнем канале банка выставлены регистры задержки (элементы $z^{-l}, l \in \mathbf{Z}$) для выполнения условия казуальности системы.

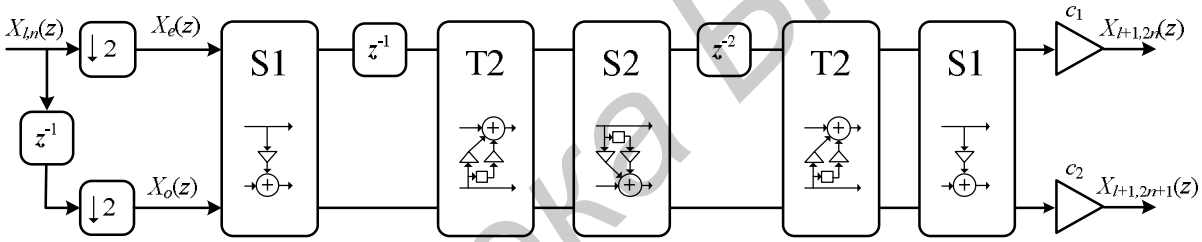
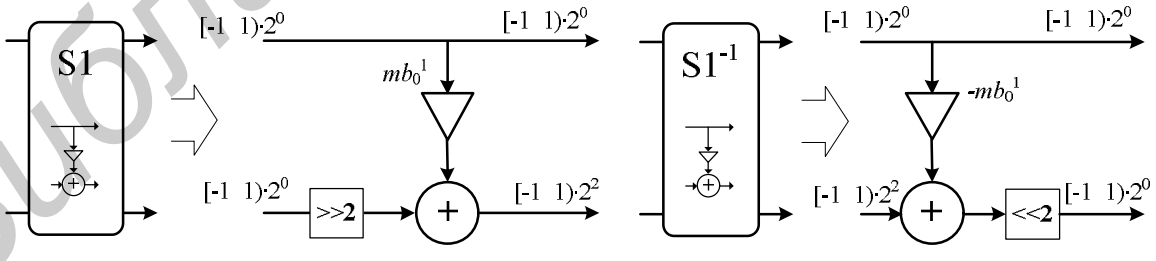


Рисунок 7 – Блок-схема лестничной структуры двухканального банка фильтров на основе Db-4 (табл. 1)

Ниже более подробно рассмотрен первый шаг структуры анализа (рисунок 8а) и обратный по отношению к нему последний шаг структуры синтеза (рисунок 8а).



а) б)

Рисунок 8 – Реализация первого шага лестничной структуры банка анализа (а) и последнего шага (б) банка синтеза

Как видно из рисунков, вычислительные блоки процедур анализа и синтеза с точки зрения реализации отличаются лишь знаками постоянных коэффициентов умножения и положением операций арифметических сдвигов.

Базируюсь на материале, описанном выше, конкретная реализация двухканального банка может быть представлена в виде вектора параметров, содержащего набор постоянных коэффициентов умножения, параметров сдвига, а также некоторой дополнительной информации касательно элементов задержки в промежуточных узлах алгоритма. В соответствии с предложенным подходом в программной среде MATLAB была реализована библиотека функций для расчета параметров банка фильтров.

1.5 Организация блоков памяти

В предложенной архитектуре на каждый уровень декомпозиции ставиться блок памяти, объем которой зависит от размера обрабатываемого фрейма, а также от предельной структуры дерева (в случае динамически реконфигурируемой системы). На каждом уровне l мультиплексор справа от ВБ (см. рисунок 9) позволяет организовать запись двух рассчитанных НЧ и ВЧ коэффициентов в блок памяти на уровне $l+1$.

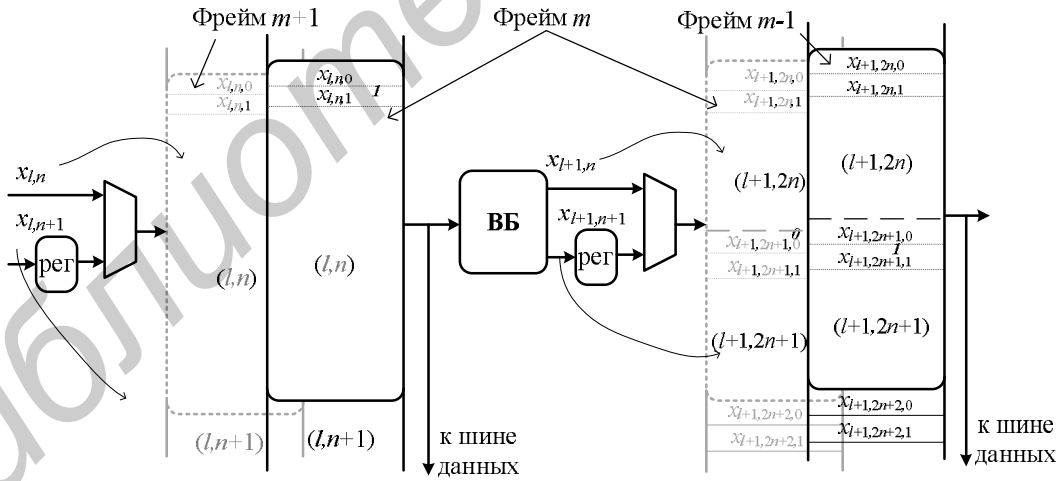


Рисунок 9 – Организация блоков памяти в поточной архитектуре

При определении общего объема необходимой памяти используется дерево декомпозиции, задающее предельный вычислительный ресурс, требуемый в приложении. Данное дерево математически выражается через множество всех узлов

$$E = \{\mu_j\}, j = 1..J, \mu_j = (l_j, n_j); \quad J, J \in \mathbb{N}, \qquad l_j, n_j \in \mathbb{Z} \tag{9}$$

где J – количество всех узлов; μ_j –j -й элемент множества, определяющий конкретный узел, l_j задает уровень декомпозиции, а n_j порядковый номер (слева направо) j-го узла.

На основании (9) выведена формула для расчета объема необходимой памяти M_V (с учетом требования двойной буферизации) и числа вычислительных блоков L системы

$$M_V = 2 \cdot \sum_{j=1}^{J} \frac{K}{2^{l_j}}, \qquad L = \max_{j=1..J} l_j \tag{10}$$

На рисунке 10 показан пример для дерева декомпозиции, задаваемого множеством узлов . Здесь же схематично проиллюстрирован принцип распределения блоков памяти для данной структуры дерева.

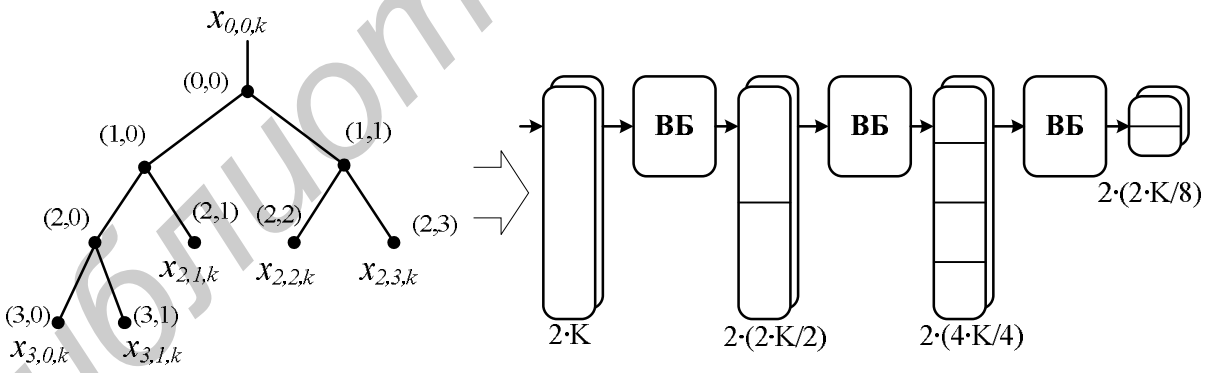


Рисунок 10 – Поточная архитектура для трехуровневого неполного дерева ПДВП

1.6 Выводы по разделу

В данном разделе отражены следующие результаты работы:

– сформулированы требования к результатам НИР:

1) получение параметризируемой архитектуры аппаратной реализации алгоритма ПДВП

2) лучшие показатели по основным характеристикам (быстродействие, точность вычислений, аппаратные затраты) по сравнению с аналогичными разработками

3) быстрое (автоматизированное) получение аппаратного прототипа процессора (инструментальная среда разработки).

– проведен анализ современных научных работ в данной области;

– предложен метод реализации алгоритма на лестничных структурах с использованием арифметики с фиксированной запятой переменного формата.

– предложена архитектура процессора ПДВП для вычислений как на основе фиксированного дерева декомпозиции так и для динамически изменяемого алгоритма.

2 Разработка метода быстрого прототипирования архитектур процессора

2.1 Параметрическое представление цифровых блоков процессора

В результате выполнения процедуры расчета математической модели на арифметике с фиксированной запятой формируется следующий набора параметров двухканального банка фильтров:

– wl – разрядность данных, поступающих на вход и соответственно разрядность результатов всех арифметических операций и буферов для хранения промежуточных результатов;

– J – индекс, задающий уровень декомпозиции, на котором используется данный операционный блок (банк фильтров);

– N – число элементарных шагов лестничной структуры;

– $V_{PE}(i) \rightarrow \{id, type, b_0, ind_x, sh_{x1}, sh_{x2}, [sh_{s2}, b_1, delayed]\}, i = 1..N$ – массив из N векторов, каждый из которых задает параметры i-го элементарного шага лестничной структуры (сами параметры будут рассмотрены ниже);

– $V_{BUF}(i) \rightarrow \{d_{xe}, d_{xo}\}, i = 1..N$ – массив из N векторов, каждый из которых задает число элементов задержки d_{xe}, d_{xo} соответственно в верхнем и нижнем каналах после i-го элементарного шага;

– $V_C \rightarrow \{c_1, c_2, n_{ye}, n_{yo}\}$ – вектор, задающий значения квантованных коэффициентов c_1 и c_2 для операции умножения на матрицу $\mathbf{C}$ (3), а также номера старших бит n_{ye}, n_{yo}, начиная с которых будут выданы результаты перемножения сигналов на коэффициенты c_1 и c_2 соответственно.

На рисунке 11 приведена блок-схема двухканального банка фильтров анализа, поясняющая принципы параметризации предложенной архитектуры.

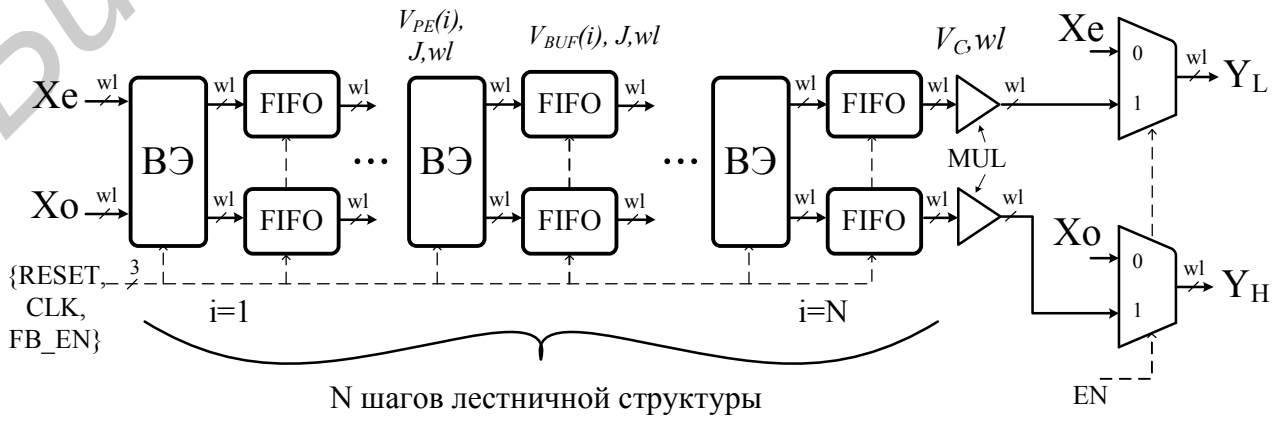


Рисунок 11 – Структурная схема ВБ

Рассмотрим подробнее структуру ВБ. В таблице 3 приведены входные выходные сигналы данного блока.

Таблица 3 – Внешние сигналы операционного блока

Название	Тип	Разрядность	Описание
FB_EN	вход	1 бит	Сигнал разрешения работы арифметических блоков и блоков хранения промежуточных данных. Если равен 1, значит блоки активны, если равен 0, то в данный момент времени операционный блок не используется, при этом на выходные значения Y_L Y_H передаются входные значения X_e, X_o.
RESET	вход	1 бит	Сигнал сброса в нулевое значение всех элементов памяти.
CLK	вход	1 бит	Сигнал тактирования элементов памяти
X_e	вход	wl бит	Входная шина данных
X_o	вход	wl бит	Входная шина данных
Y_L	выход	wl бит	Выходная шина данных
Y_H	выход	wl бит	Выходная шина данных

Принцип работы операционного блока заключается в следующем. На входные шины X_e и X_o поступают четный и нечетный отсчеты входной последовательности, после чего начинается процесс вычисления вейвлет коэффициентов. Результат высчитывается в вычислительных элементах (ВЭ) на основе текущих входных значений и предыдущих значений, хранящихся в буферных блоках. Данные блоки представляют собой регистровые файлы, работающие по принципу FIFO. Число регистров в каждом таком блоке определяется как $n_{xe} \cdot 2^J$ (либо $n_{xo} \cdot 2^J$), где значения n_{xe}, n_{xo} определяют задержку сигнала, соответствующую задержке $z^{-n_{xe}}, z^{-n_{xo}}$ в исходном алгоритме, а

2^J

значение определяет число тактов между приходом входных отсчетов из одной субполосы. Данный прием, предложенный в [[6]], необходим для того чтобы значения предыстории на выходах блоков FIFO относились к той же субполосе, что и текущие входные сигналы.

Пара результатов, вычисленная в последнем элементе ВЭ, умножается на константы c_1 и c_2, и далее подаётся на выход операционного блока.

Как уже говорилось раньше, для параметризации *i*-го вычислительного элемента ВЭ формируется специальный вектор $V_{PE}(i)$. Ниже даны разъяснения для всех параметров, включенных в данный вектор.

Параметр id (identifier). Идентификатор, задающий один из двух типов процессорного элемента для реализации элементарного шага лестничной структуры: $id = 'PE0'$ при значении коэффициента $b_1^i = 0$ (рисунок 12), в противном случае $id = 'PE1'$ (рисунок 13).

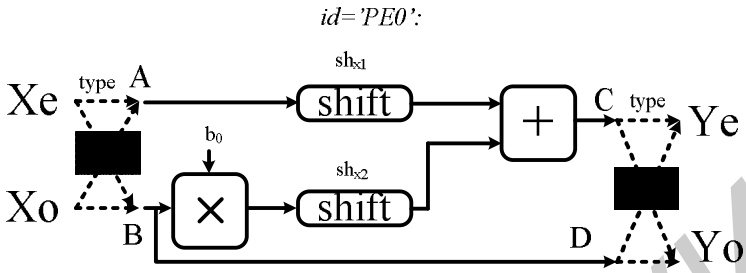


Рисунок 12– Структурная схема процессорного элемента при $id = 'PE0'$.

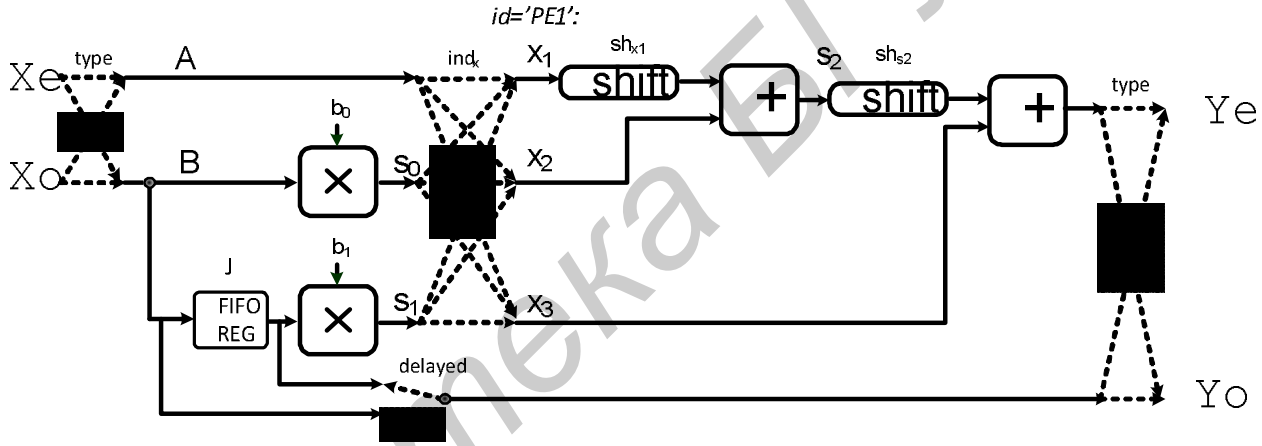


Рисунок 13 – Структурная схема процессорного элемента при $id = 'PE1'$.

Параметр $type$ задает тип элементарного шага, реализуемого PE: $type = 's'$ если это прямой шаг, и $type = 't'$ если шаг дуальный.

Параметры b_0, b_1 могут принимать значения от «минус» до «плюс» $[(2]^{wl-1} - 1)$ и соответствуют значениям mb_0, mb_1 для квантованных коэффициентов b_0^i, b_1^i. В случае $id = 'PE0'$ параметр b_1 игнорируется.

Параметр ind_x представляет собой массив из трех чисел, принимающих значения 1,2, либо 3. Данный параметр определяет очередность суммирования сигналов A, s_1 s_2 для ВЭ с $id = 'PE1'$. К примеру, если параметр $ind_x = [2\,3\,1]$, то для получения

выходного результата сначала выполняется операция $A + s_2$ и далее к получившемуся результату прибавляется значение s_1.

Параметры $sh_{x1}, sh_{x2}, sh_{s2}$ определяют число арифметических сдвигов для внутренних сигналов процессорного элемента x_1, x_2 и s_2 соответственно.

Параметр $delayed$. Для того чтобы разъяснить назначение данного параметра, рассмотрим участок исходного алгоритма (рисунок 14), на котором сначала выполняется прямой шаг $s(z) = b_0 + b_1 z^{-1}$ и далее следует задержка сигнала z^{-1}. Как видно из рисунка, сигнал $Y_e(n-1)$, который должен быть на выходе всей схемы, уже сформирован внутри процессорного элемента. Поэтому при значении $delayed = 1$ на выход ВЭ выдается не сигнал $Y_e(n)$, а сигнал, $Y_e(n-1)$, что позволяет сэкономить для данной

2^J

схемы на J-ом уровне декомпозиции регистров.

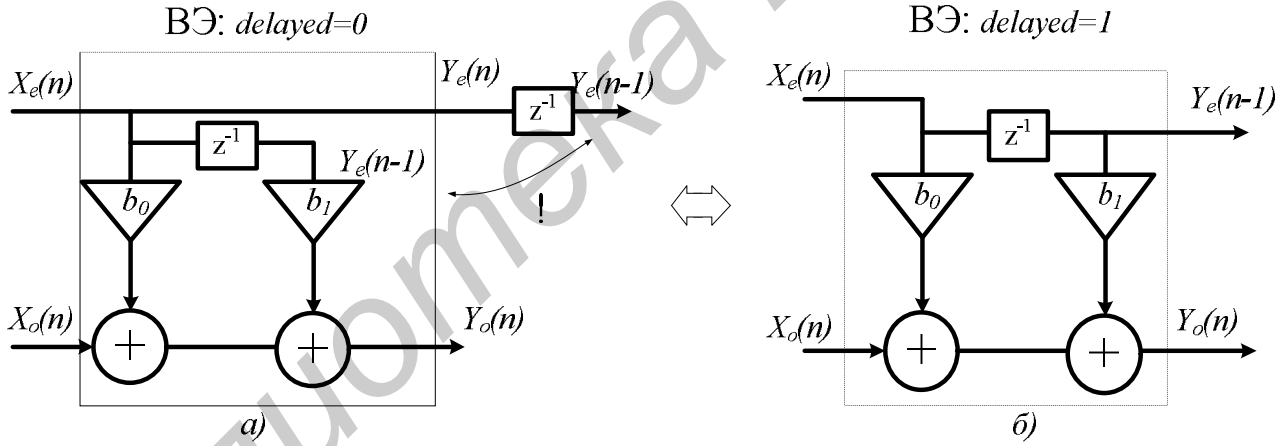


Рисунок 14 – Структура схема процессорного элемента PE при $delayed = 0$ (а) и $delayed = 1$ (б).

Отметим, что блоки арифметического сдвига *shift*, а также коммутаторы SW_1, SW_2, SW_3, SW_4 не используют аппаратного ресурса системы. Операция арифметического сдвига вправо сводится к смещению сигнальных линий перед подачей их на вход сумматора. А указанные выше коммутаторы в соответствии с параметрами $type, ind_x, delayed$ реализуются в виде замыкания соответствующих линий.

Таким образом, процессорный элемент для случая $id = 'PE0'$ в конечном итоге будет реализован на одном сумматоре и одном умножителе, а процессорный элемент

для $id = 'PE1'$ – на двух сумматорах, двух умножителях, и регистровом файле, со-

2^J

стоящем из регистров. Все используемые элементы имеют разрядность равную wl .

Блоки промежуточной памяти и логика управления параметризируются на основе предельной структуры дерева декомпозиции.

Таким образом, прототип процессора ПДВП может быть задан в виде параметров, описывающих структуру двухканального банка фильтров (ВБ), а также вектора, определяющего предельное дерево декомпозиции.

2.2 Метод быстрого прототипирования архитектур процессора

В результате структура метода быстрого прототипирования может быть описана следующей последовательностью действий. Расчет параметров лестничной структуры двухканального банка фильтров на основании исходной базисной вейвлет функции; трансляция полученной математической модели на арифметику с фиксированной запятой с учетом требований точности и ограничением аппаратных ресурсов (разрядности регистров и вычислительных блоков); формирование вектора параметров конфигурирования прототипа процессора; расчет аппаратных затрат на реализацию прототипа; расчет оценочных характеристик прототипа; формирование выходных файлов синтезируемого VHDL-описания процессора.

Для быстрого прототипирования архитектур процессоров вычисления ПДВП в зависимости от конкретного приложения авторами предлагается среда проектирования, представленная на рисунке 15. Входными параметрами для среды являются: структура (либо предельная структура для реконфигурируемой системы) дерева ПДВП; коэффициенты базисных вейвлет-фильтров, тип используемой памяти, разрядность вычислений. При помощи программного обеспечения, реализованного в среде MATLAB, рассчитываются параметры банка анализа на основе лестничных структур, осуществляется трансформация алгоритма на арифметику с фиксированной запятой, и далее генерируется VHDL-пакет параметров описания ПДВП процессора. Данный пакет объединяется с описаниями компонентов процессора в синтезируемое

VHDL-описание, которое при помощи специализированных САПР реализуется на конкретной микросхеме FPGA.

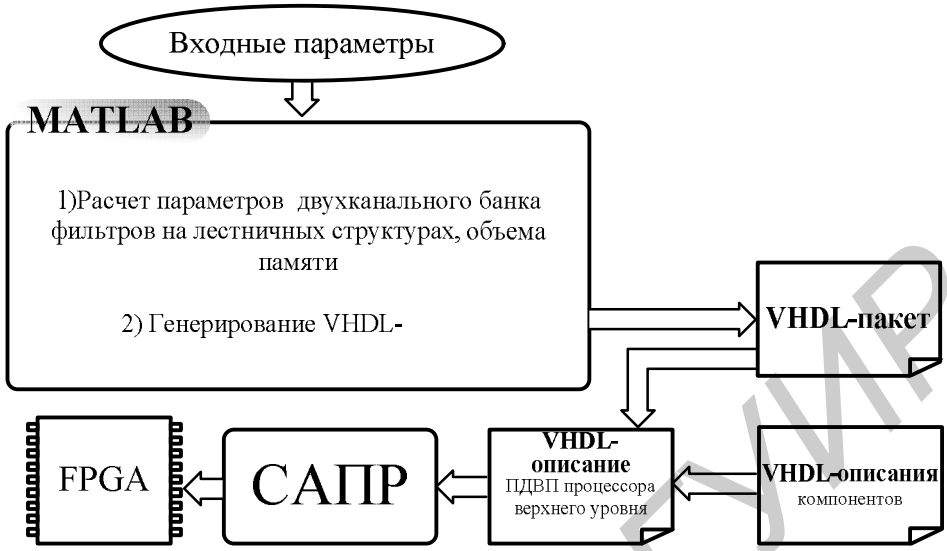


Рисунок 15 – Среда разработки

2.3 Выводы по разделу

В ходе разработки метода быстрого прототипирования архитектур процессора ПДВП был выполнен аналитический обзор требований, предъявляемых к аппаратным реализациям алгоритмов ПДВП целевыми приложениями. Также проведен анализ предложенных ранее аппаратных реализаций процессоров ПДВП на предмет возможности параметризации различных блоков относительно исходного математического представления на арифметике с фиксированной запятой переменного формата.

Предложена общая структура метода быстрого прототипирования архитектур процессоров с исходными данными в виде спецификации, определяемой целевым приложением в виде вектора численных значений, конкретизирующих исходные требования, рассмотренные выше. На выходе метод выдает синтезируемое VHDL-описание прототипа процессора, а также набор количественных значений технических характеристик.

3 Инструментальная среда для быстрого прототипирования динамически реконфигурируемых процессоров ПДВП

3.1 Параметризированное VHDL описание процессора

Ниже приведены особенности практической реализации метода, связанные с формированием VHDL-описания двухканального банка фильтров, как основного модуля процессора.

В качестве примера ниже приведен код VHDL-пакета с параметрами для двухканального банка на лестничных структурой для вейвлет фильтров db9.

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

package pdwt_core_pack is
constant N      : integer := 9; --число шагов лестничной структуры
constant wl          : integer := 16; -- разрядность данных

type type_ind_array is array(natural range <>) of integer range 0 to 15 ;
type type_char_array is array(natural range <>) of character ;
type type_data_array is array(natural range <>) of integer range -2**wl
to (2**wl)-1 ;

-- параметры лестничных структур
--'s'->прямой шаг          't'->обратынй шаг
-- тип лестничных структур банка
constant vect_id: type_char_array(1 to N):=('s', 't', 's', 't', 's', 't',
's', 't', 's');
--PE0->0        PE1->1
constant vect_pe_type: type_ind_array(1 to N):=(0, 1, 1, 1, 1, 1, 1, 1,
0);
constant indx1: type_ind_array(1 to N):=(1, 2, 1, 3, 3, 3, 3, 3, 1);
constant indx2: type_ind_array(1 to N):=(2, 1, 3, 1, 2, 1, 2, 1, 2);
constant indx3: type_ind_array(1 to N):=(3, 3, 2, 2, 1, 2, 1, 2, 3);
constant vect_shx1: type_ind_array(1 to N)      :=(2, 1, 0, 3, 0, 0, 1,
1, 1);
constant vect_shx2: type_ind_array(1 to N)      :=(0, 1, 2, 1, 4, 0, 1,
1, 1);
constant vect_delayed: type_ind_array(1 to N)   :=(0, 0, 1, 0, 1, 0, 1,
0, 0);
constant vect_b1: type_data_array(1 to N):=(     0,
                                                19133,
                                               -27239,
                                                19867,
                                                19133,
                                               -27239,
                                                19867,
                                                20000,
                                                0);
constant vect_b0: type_data_array(1 to N):=( -25419,
                                               -20002,
                                                21297,
                                               -28196,
```

```
                                                    20585,
                                                   -25419,
                                                   -20002,
                                                    21297,
                                                   -28196);
-- constants of the Vbuf vector
constant dxe: type_ind_array(1 to N):=(1, 0, 1, 0, 1, 0, 1, 0, 0);

constant dxo: type_ind_array(1 to N):=(0, 0, 0, 0, 0, 0, 0, 0, 0);

-- constants of the VK vector
constant K1:integer:=30274;
constant K2:integer:=17734;
constant exp_ye:integer:=2;
constant exp_yo:integer:=3;
end pdwt_core_pack;

package body pdwt_core_pack is
end pdwt_core_pack;
```

Данный пакет подключается к остальным VHDL-модулям процессора. Таким образом обеспечивается конфигурирование всех блоков под заданную спецификацию. Далее для примера приведено несколько описаний блоков процессора.

Описание двухканального банка:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
library work;
use work.pdwt_core_pack.all;

entity LIFT_DChFB is
    Port ( iX : in  STD_LOGIC_VECTOR (wl-1 downto 0);
           CLK : in  STD_LOGIC;
           iRESET : in  STD_LOGIC;
           iEN : in  STD_LOGIC;
           YL : out  STD_LOGIC_VECTOR (wl-1 downto 0);
           YH : out  STD_LOGIC_VECTOR (wl-1 downto 0));
end LIFT_DChFB;

architecture Behavioral of LIFT_DChFB is

signal xe,xo,ye,yo: type_signal_array_wl(1 to N+1);
signal en_reg: STD_LOGIC:='0';
signal yl_s,yh_s: STD_LOGIC_VECTOR (2*wl-2 downto 0):=ZEROS(2*wl-2 downto
0);
signal en,reset: STD_LOGIC:='0';
signal x: STD_LOGIC_VECTOR(15 downto 0):=x"0000";
begin
in_reg: process(CLK)
begin
     if rising_edge(CLK) then
          x<=iX;
          reset<=iRESET;
          EN<=iEN;
     end if;
```

```
end process;
p_split:  SPLIT
    Port map ( X=>X,
                              CLK=>CLK,
                              EN=>EN,
                              RESET=>RESET,
                              EN_OUT=>en_reg,
                              Xe=>xe(1),
                              Xo=>xo(1));
p_lift_gen: for i in 1 to N generate
     begin
p_pe_i:    PE
     Generic map(ID=>vect_id(i),
                              PE_TYPE=>vect_pe_type(i),
                              b0=>vect_b0(i),
                              b1=>vect_b1(i),
                              SHIFT_X1=>vect_shx1(i),
                              SHIFT_X2=>vect_shx2(i),
                              IND_X=>(indx1(i),indx2(i),indx3(i)),
                              DELAYED=>vect_delayed(i),
                              J=>0)
    Port map ( Xe=>xe(i),
                              Xo=>xo(i),
                              CLK=>CLK,
                              EN=>en_reg,
                              RESET=>RESET,
                              Ye=>ye(i),
                              Yo=>yo(i));
p_delay_ye:     FIFO_REG
     Generic map(    SIZE_QUEUE=>dxe(i))
    Port map (  D_IN=>ye(i),
                              CLK=>CLK,
                              EN=>en_reg,
                              RESET=>RESET,
                              Q_OUT=>xe(i+1));
p_delay_yo:     FIFO_REG
     Generic map(    SIZE_QUEUE=>dxo(i))
    Port map (  D_IN=>yo(i),
                CLK=>CLK,
                EN=>en_reg,
                RESET=>RESET,
                Q_OUT=>xo(i+1));
     end generate;
p_mult_k1: MULT_FS
     Generic map(    wl=>wl)
          Port map (      A=>xe(N+1),
                              B=>CONV_STD_LOGIC_VECTOR(K1,wl),
                              S=>yl_s);
p_mult_k2: MULT_FS
     Generic map(    wl=>wl)
          Port map (      A=>xo(N+1),
                              B=>CONV_STD_LOGIC_VECTOR(K2,wl),
                              S=>yh_s);
out_reg: process(CLK)
begin
     if rising_edge(CLK) then
          YL<=yl_s(2*wl-2-exp_ye downto wl-1-exp_ye);
          YH<=yh_s(2*wl-2-exp_yo downto wl-1-exp_yo);
```

```
     end if;
end process;
end Behavioral;
```

Описание блока, выполняющего обработку на одном шаге лестничной структуры:

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;
use IEEE.STD_LOGIC_ARITH.ALL;
use IEEE.STD_LOGIC_UNSIGNED.ALL;
library work;
use work.pdwt_core_pack.all;

entity PE is
     Generic(   ID:        character:='t';-- s or t
                PE_TYPE:   integer range 0 to 1:=1;
                     b0:        integer :=-32768;
                b1:        integer :=8192;
                SHIFT_X1:integer range 0 to 31:=0;
                SHIFT_X2:integer range 0 to 31:=0;
                IND_X:     type_ind_array:=(1, 2, 3);
                DELAYED:   integer range 0 to 1:=0;
                --WL:      integer range 1 to 1:='0';
                J:              integer range 0 to 7:=0);
    Port (      Xe : in  STD_LOGIC_VECTOR (15 downto 0);
                Xo : in  STD_LOGIC_VECTOR (15 downto 0);
                CLK : in  STD_LOGIC;
                EN : in  STD_LOGIC;
                RESET : in  STD_LOGIC;
                Ye : out  STD_LOGIC_VECTOR (15 downto 0);
                Yo : out  STD_LOGIC_VECTOR (15 downto 0));
end PE;

architecture Behavioral of PE is

signal a,b,c,d: STD_LOGIC_VECTOR(wl-1 downto 0):=(others=>'0');

BEGIN
---- Перестановка (SW1) входных данных
gen_in_id_s: if ID='s' generate
     begin
          a<=Xo;
          b<=Xe;
     end generate;
gen_in_id_t: if ID='t' generate
     begin
          a<=Xe;
          b<=Xo;
     end generate;

---- Процессорный элемент  PE0 или PE1
gen_pe0:if PE_TYPE=0 generate
     begin
p_pe0: PE0
     Generic map (   SHIFT_A=>SHIFT_X1,
                              SHIFT_S0=>SHIFT_X2,
```

```
                                    b0=>b0)
    Port map(   An=>a,
                            Bn=>b,
                            C=>c,
                            D=>d);
     end generate;
gen_pe1:if PE_TYPE=1 generate
     begin
p_pe1: PE1
     Generic map (
                        SHIFT_X1=>SHIFT_X1,
                        SHIFT_S2=>SHIFT_X2,
                        ind_x=>IND_X,
                        delayed=>DELAYED,
                        b0=>b0,
                        b1=>b1,
                        J=>J)
    Port map ( An=>a,
                            Bn=>b,
                            RESET=>RESET,
                            CLK=>CLK,
                            EN=>EN,
                            C=>c,
                            D=>d);
     end generate;

---- Перестановка (SW2) выходных данных
gen_out_id_s: if ID='s' generate
     begin
          Ye<=d;
          Yo<=c;
     end generate;
gen_out_id_t: if ID='t' generate
     begin
          Ye<=c;
          Yo<=d;
     end generate;

END Behavioral;
```

3.1.1 Пример поточного процессора ПДВП аппроксимирующего шкалу барков

В задачах обработки речевых сигналов, основанных на перцептуальном восприятии информации человеком, необходимо применение банка фильтров, который осуществляет частотную декомпозицию в соответствии со шкалой барков. Ниже рассмотрена реализация поточной архитектуры процессора, вычисляющего ПДВП для фиксированного дерева, аппроксимирующего шкалу барков для частотного диапазона сигнала 0-8000 Гц (рисунок 16).

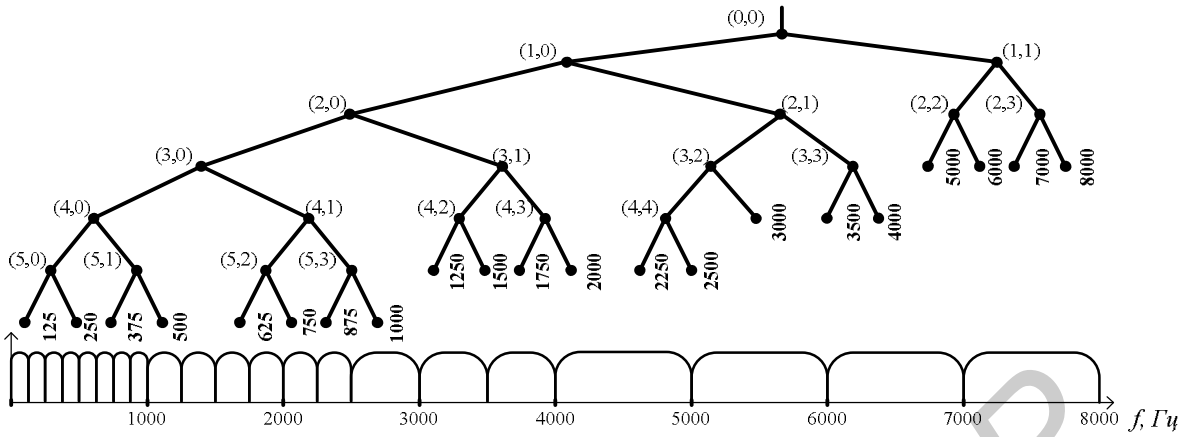


Рисунок 16 . Структура дерева ПДВП, аппроксимирующая шкалу барков в полосе частот 0-8000 Гц

Процессор состоит из шести уровней декомпозиции. Хранение промежуточных данных организовано на основе регистровых файлов. Разрядность внутренних вычислений 18 бит, разрядность входных данных 16 бит. Латентность получения выходных результатов 12 тактов. Базисная вейвлет-функция – Db10 (20 коэффициентов). Аппаратные затраты на основные типы ресурсов приведены в таблице 4.

Оценочная частота тактирования для данной реализации, полученная в САПР Xilinx ISE для кристалла семейства Virtex-4 xc4vlx25ff668-12, составила 8 МГц. При фреймовой обработке и наличии предварительной буферизации для данных, поступающих с частотой дискретизации 16 кГц, используется лишь 0,2 % от имеющегося времени, что тем самым высвобождает больше времени для субполосной обработки.

Таблица 4 – Аппаратные затраты на реализацию поточного процессора

Тип ресурса	Количество
Регистры, разрядность 18 бит	547
Сумматоры, разрядность 18 бит	160
Умножители 18?18	176
Мультиплексоры 2-1 (демультиплексоры 1-2)	17

4 Анализ параметров качества быстрого прототипирования процессоров ПДВП

4.1 Анализ ошибок вычислений

Для анализа полученных решений была написана библиотека функции, моделирующая вычислительный процесс в банке фильтров с произвольной структурой дерева.

На рисунке 17 приведена оценка дисперсии ошибки восстановления сигнала в зависимости от выбора разрядности внутренних регистров в результате прохождения через двухканальный банк фильтров анализа/синтеза (в примере применялись вейвлет фильтры db8). На данном рисунке также представлены результаты эксперимента при использовании КИХ фильтров, лежащих в основе алгоритма ПДВП. Можно отметить, что при использовании одной и той же разрядности регистров КИХ фильтрация дает лучшие результаты, но при этом требует в два раза больше вычислений. Так для достижения уровня дисперсии ошибки в -70 Дб при реализации на лестничных структурах понадобится 16 бит, что примерно на два бита больше по сравнению с КИХ реализацией. Но данный недостаток компенсируется за счет значительного сокращения арифметических операций по сравнению с прямой реализацией. Таким образом, можно сделать вывод, что предложенный подход более эффективен при аппаратной реализации алгоритма ПДВП в сравнении с банком на базе КИХ фильтров.

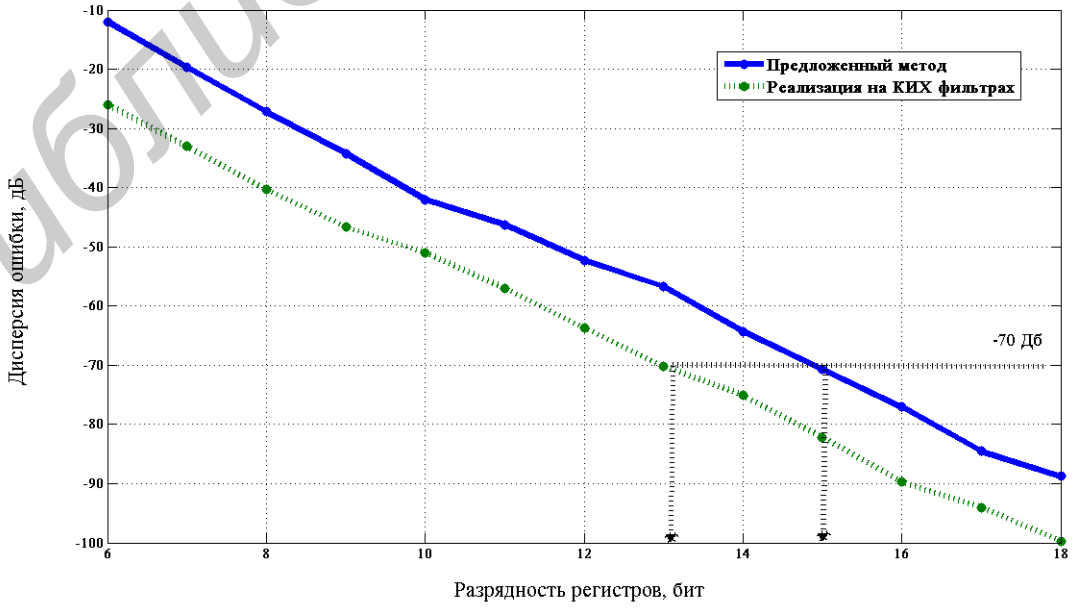
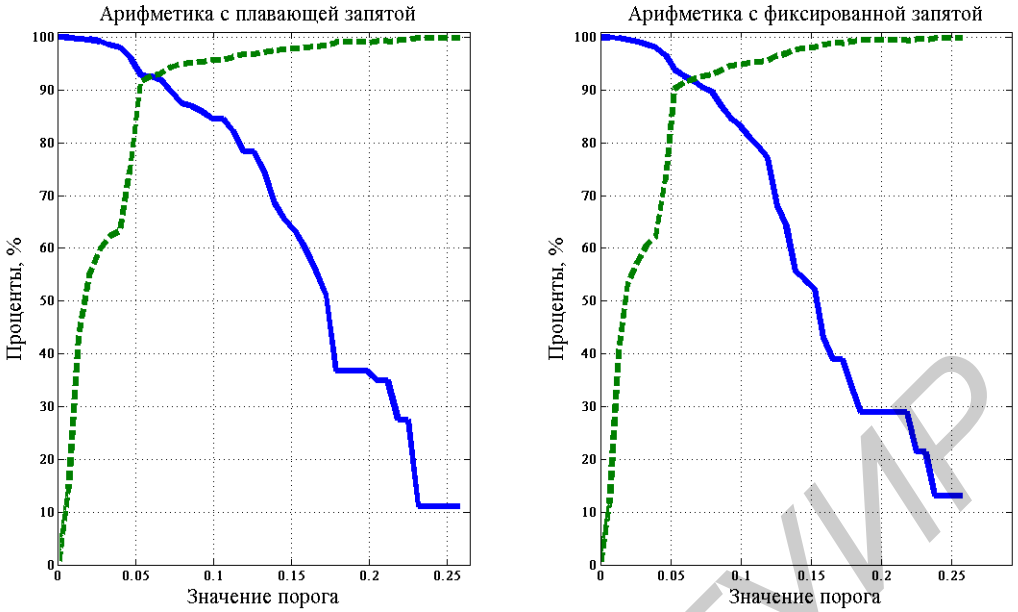


Рисунок 17 – Зависимость дисперсии ошибки восстановления сигнала от разрядности регистров в системах анализ/синтез на базе двухканального банка фильтров (вейвлет фильтры db8): сплошной линией обозначены результаты для предложенного алгоритма на лестничных структурах; пунктирной – целочисленная реализация той же системы с использованием КИХ фильтрации

4.2 Анализ свойств сохранения энергии.

Ниже рассмотрен еще один эксперимент, демонстрирующий сохранение свойства локализации энергии при использовании целочисленной реализации алгоритма ПДВП на лестничных структурах. Для эксперимента был сгенерирован полигармонический сигнал и пропущен через пятиуровневое дерево декомпозиции быстрого вейвлет преобразования (деление по дереву осуществляется только в области низкочастотных компонент). В качестве примера было выбрано семейство вейвлет функций db2. Весь диапазон амплитуд в области вейвлет коэффициентов был разделен на 40 пороговых значений. Данные значения отложены на оси абсцисс графиков, представленных на рисунке 18. Каждому пороговому значению был сопоставлен вектор, состоящий из полученных в результате анализа вейвлет коэффициентов, при условии, что эти коэффициенты больше данного порога. В противном случае значения коэффициентов были заменены нулями (т.е. в каждом векторе были отброшены незначимые относительно данного порога значения). Для всех векторов была выполнена процедура восстановления исходного сигнала. На рис. 18а, 18б для реализаций на арифметике с плавающей и фиксированной запятой соответственно показаны: сплошной линией – график соотношения энергии восстановленного сигнала к исходному (в процентах) при различных значениях порога; пунктирной линией – процент «отброшенных» вейвлет коэффициентов в зависимости от выбранного порога.



а) б)

Рисунок 18 – Оценки энергии восстановления сигнала в зависимости от порога значимых вейвлет коэффициентов для алгоритмов ПДВП на арифметике с а) плавающей запятой и б) с фиксированной запятой переменного формата.

Основываясь на полученных результатах, отмечаем практически полное соответствие модели на основе плавающей запятой и предложенного целочисленного подхода.

4.3 Выводы по разделу

По результатам экспериментов, проведенных в данном разделе можно отметить следующее:

– предложенный подход более эффективен с точки зрения минимизации вносимого уровня ошибки при аппаратной реализации алгоритма ПДВП в сравнении с банком на базе КИХ фильтров.

– реализация алгоритма ПДВП на фиксированной запятой переменного формата сохраняет свойство локализации энергии, присущее вейвлет пакетам.

ЗАКЛЮЧЕИЕ

Ниже сформулированы основные результаты проделанной НИР:

– предложена оригинальная архитектура динамически реконфигурируемого процессора ПДВП, особенностью которой является высокое быстродействие за счет использования поточной архитектуры на лестничных структурах, что при фреймовой обработке данных обеспечивает хороший запас по времени на субполосную обработку;

– разработан метод быстрого прототипирования динамически реконфигурируемых процессоров ПДВП по заданной спецификации приложения;

– написана библиотека для расчета и моделирования работы алгоритма на фиксированной запятой переменного формата, а также реализовано параметризированное VHDL-описание цифровых блоков процессора;

По результатам научных исследований авторами была подготовлена глава «Dynamic reconfigurable on the lifting steps wavelet packet processor with frame-based psychoacoustic optimized time-frequency tilling for real-time audio application » в монографии «Design and architecture of digital signal processing » под редакцией Dr. Gustavo Ruiz, ISBN 980-953-307-610-7, издательство INTECH Publishers 2012.

Применение арифметики с фиксированной запятой переменного формата используется в учебном процессе на кафедре ЭВС в курсе ПЭВСДРА. По данной теме написан раздел «Реализация алгоритмов ЦОС на основе арифметики с фиксированной запятой переменного формата» методического пособия «Проектирование ЭВС с динамически реконфигурируемой архитектурой Методическое пособие для студентов специальности 1-40 02 02 «Электронные вычислительные средства» дневной формы обучения».

Все этапы НИР выполнены своевременно и в полном объеме согласно ТЗ.

Уровень предложенных научных решений соответствует современным зарубежным подходам, о чем свидетельствуют публикации авторов по данной теме в рецензируемые зарубежные издания.

Результаты НИР могут быть использованы в ОКР для создания высокопроизводительных приложений обработки мультимедиа данных.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Petrovsky, Al. Dynamic algorithm transforms for reconfigurable real-time audio coding processor / Al. Petrovsky, A. Petrovsky // The proc. of the Intern. Conf. on Parallel Computing in Electrical Engineering, PARELEC'02, Warsaw, Poland, Sep. 22-25, 2002, IEEE Computer Society Press. – Los Alamitos, California, 2002. – P.422-424.

[2] Sweldens, W. The lifting scheme: A construction of second generation wavelets / W. Sweldens // Siam J. Math. Anal. – 1997. – Vol. 29(2) – P. 511-546.

[3] Daubechies, I. Factoring wavelet transforms into lifting steps / I. Daubechies, W. Sweldens, // Journal of Fourier Anal. Appl. – 1998. – Vol. 4, № 3, – P. 247-269.

[4] Architecture for wavelet packet transform with best tree searching / M. A. Treans [at al.] // 12th IEEE International Conference on Application-Specific Systems, Architectures and Processors (ASAP'00). – 2000. – P.289-298.

[5] Coifman, R. R. Entropy-based algorithms for best basis selection / R.R. Coifman, M.V. Wickerhauser // Trans. Inform. Theory. –1992. – Vol. 38. – P. 1713-1716.

[6] Wang, C. Efficient VLSI Architecture for Lifting-Based Discrete Wavelet Packet Transform / C. Wang, W.S. Gan // IEEE Transactions on circuits and system – II: Express briefs. – 2007. – Vol. 54(5). – P. 422-426.

[7] Mallat, S. A wavelet tour of signal processing: 2nd edition / S. Mallat. – Academic Press, 1999 – 637 p.

[8] Design and DSP implementation of fixed-point systems / M. Coors [at al.] // EURASIP journal on applied signal processing. – 2002– №9 – P. 908-925.

[9] Menard, D. Floating-to-fixed-point conversion for digital signal processors / D. Menard, D. Chillet, O. Sentieys // EURASIP J. on Applied Signal Processing. – 2006. – Vol. 2006 –P. 1-19.

[10] Daubechies, I. Orthogonal bases of compactly supported wavelets / I. Daubechies // Communications on pure and applied mathematics, – 1988. –Vol. 41. – P. 909-996.