Министерство науки и высшего образования РФ
Федеральное государственное автономное образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О. В. Непомнящий
подпись инициалы, фамилия
«___» _____ 2019 г.

БАКАЛАВАРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Реконфигурируемый, адаптивный, цифровой фильтр эхо-компенсации
тема

Пояснительная записка

Руководитель	_____ подпись, дата	профессор, зав. каф. ВТ, канд. техн. наук должность, ученая степень	О. В. Непомнящий инициалы, фамилия
Выпускник	_____ подпись, дата		Е. С. Бывшев инициалы, фамилия
Нормоконтролер	_____ подпись, дата	доцент, канд. техн. наук должность, ученая степень	В. И. Иванов инициалы, фамилия

Красноярск 2019

Министерство науки и высшего образования РФ
Федеральное государственное автономное образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О. В. Непомнящий
подпись инициалы, фамилия
«___» _____ 2019 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Бывшеву Егору Сергеевичу
фамилия, имя, отчество

Группа КИ15-08Б Направление (специальность) 09.03.01
номер код

Информатика и вычислительная техника
наименование

Тема выпускной квалификационной работы Реконфигурируемый, адаптивный, цифровой фильтр эхо-компенсации

Утверждена приказом по университету № _____ от _____

Руководитель ВКР О. В. Непомнящий, профессор, заведующий кафедрой Вычислительная техника, к.т.н.
инициалы, фамилия, должность, учёное звание и место работы

Исходные данные для ВКР: Разработать и исследовать программную модель и лабораторный макет эхо-компенсатора, предназначенного для подавления акустических и электрических эхо-сигналов. При реализации и исследовании лабораторного макета выполнять отработку сигналов акустического диапазона.

Перечень разделов ВКР: Анализ и классификация известных методов эхо-компенсации, метод подавления эхо-сигналов на основе рекурсивного цифрового фильтра, программное моделирование и исследование основных характеристик эхо-компенсатора, практическая реализация и исследования лабораторного макета эхо - компенсатора.

Перечень графического материала: презентация в формате PowerPoint, раздаточный материал, функциональные схемы, алгоритмы ПО.

Руководитель ВКР _____ О. В. Непомнящий
подпись инициалы, фамилия

Задание принял к исполнению _____ Е. С. Бывшев
подпись инициалы, фамилия

«___» _____ 2019 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Реконфигурируемый, адаптивный, цифровой фильтр эхо-компенсации» содержит 59 страниц текстового документа, 1 таблицу, 17 иллюстраций, 16 формул, 2 приложения, 32 использованных источников.

АДАПТИВНАЯ ФИЛЬТРАЦИЯ, ЦИФРОВОЙ ФИЛЬТР, ОБРАБОТКА СИГНАЛОВ, СХОДИМОСТЬ, ВЫЧИСЛИТЕЛЬНАЯ СЛОЖНОСТЬ, ЭХО-КОМПЕНСАЦИЯ, ПОЛОЖИТЕЛЬНАЯ ОБРАТНАЯ СВЯЗЬ, MATLAB, ПЛИС

Объект работы – реконфигурируемый адаптивный цифровой фильтр эхо-компенсации

Задачи:

– выполнить анализ существующих методов адаптивной эхо-компенсации;

– разработать программную модель реконфигурируемого адаптивного цифрового эхо-компенсатора;

– выполнить исследование программной модели;

– разработать проект реконфигурируемого адаптивного цифрового эхо-компенсатора на базе ПЛИС;

– выполнить исследование, разработанного проекта.

Полученные результаты:

– проведен анализ существующих методов адаптивной эхо-компенсации, выявлены их основные преимущества и недостатки;

– в среде MATLAB/Simulink разработана программная модель эхо-компенсатора;

– выполнены исследования программной модели;

– на основе ПЛИС Intel/Altera Cyclone IV разработан проект эхо-компенсатора;

– выполнены лабораторные исследования макета.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время удельный вес средств цифровой обработки сигнала в составе систем различного назначения неуклонно возрастает. Особенно такая тенденция характерна для систем радиосвязи, медико-биологических исследований, решении задач навигации при контроле аэрокосмических и морских объектов, сейсмологии, обработки аудио- и видеоинформации, цифровой оптики и в ряде других приложений. Отличительной особенностью этих задач является большой объем вычислений, реализуемых в реальном времени [1].

Для рассматриваемых приложений, одной из приоритетных задач является решение проблем подавления помех, возникающих в результате переотражения волн. Для решения данной проблемы применяются различные подходы, в том числе и основанные на технологиях цифровой адаптивной фильтрации сигналов. Адаптивная эхо-компенсация является эффективным средством борьбы с проблемой эхо-сигналов в различных технических системах и приложениях [2–5].

Таким образом, **актуальность** выбранной темы выпускной квалификационной работы (ВКР) обусловлена широким спектром аппаратных средств, используемых в различных сферах человеческой деятельности, отсутствием универсальных решений подавления данного рода помех, а также современными требованиями к обработке сигналов в режиме реального времени.

Целью данной работы является разработка метода адаптивной эхо-компенсации, его моделирование, а также практическая реализация для определения предельных параметров при обработке сигналов звукового диапазона.

1 Анализ существующих адаптивных методов эхо-компенсации

1.1 Место адаптивных алгоритмов в фильтрации сигнала

Адаптивная обработка сигналов зарождалась в середине прошлого века. Основные принципы адаптивной обработки сигналов сформировали такие ученые, как Винер, Колмогоров, Крейн, Левинсон. Однако адаптивную обработку сигналов на практике впервые применил Бэрнард Уидроу (Bernard Widrow), который в конце 50-х годов прошлого века разработал и реализовал первый адаптивный фильтр на базе Least Mean Square (LMS) алгоритма, также известного как алгоритм по методу наименьшего квадрата (МНК) [6].

Как правило, адаптивную фильтрацию применяют с тех случаях, когда фильтры с фиксированными параметрами не могут выполнить поставленную задачу. Например, если условия фильтрации изменяются, поэтому требования к фильтру не могут быть сформулированы заранее [6].

Основные области применения адаптивной фильтрации – очистка данных от помех и шумов, перекрывающихся по спектру со спектром полезных сигналов, или когда полоса частот помех и шумов неизвестна, переменна и не может быть задана априорно для расчета параметрических фильтров. Так, например, в цифровой связи сильная активная помеха может интерферировать с полезным сигналом, а при передаче цифровой информации по каналам с плохими частотными характеристиками может наблюдаться межсимвольная интерференция цифровых кодов. Адаптивный фильтр является эффективным решением данных проблем [7].

Адаптивный фильтр – это фильтр с изменяемыми в процессе работы параметрами, набор которых во многом зависит от критерия работы адаптивного фильтра. Этим критерием часто является достижение минимума некоторой целевой функции, как правило, квадратичной функции ошибки между так называемым требуемым и выходным сигналами адаптивного фильтра [6]. Достижение минимума целевой функции означает, что выходной

сигнал адаптивного фильтра в определённой степени приближен к требуемому сигналу, физическая природа которого определяется конкретным приложением адаптивного фильтра [8]. В зависимости от решаемой задачи и типа обрабатываемого сигнала адаптивные фильтры могут быть одноканальными или многоканальными с действительными или комплексными весовыми коэффициентами (ВК) [9]. Алгоритмы вычисления ВК могут быть простыми, как, например, LMS-алгоритм, или сложными, как, например, Recursive Least Squares (RLS) алгоритм, также известный как рекурсивный метод наименьших квадратов (РМНК).

1.2 Общие принципы адаптивной обработки сигналов

Общая структура адаптивного фильтра показана на рисунке 1. Входной дискретный сигнал $x(k)$ обрабатывается дискретным фильтром, в результате чего формируется сигнал $y(k)$. Этот выходной сигнал сравнивается с сигналом-образцом $d(k)$, разности между ними образует сигнал ошибки $e(k)$. Задача адаптивного фильтра – минимизировать ошибку воспроизведения образцового сигнала. С этой целью блок адаптации после обработки каждого отсчета анализирует сигнал и дополнительные данные, поступающие из фильтра, используя результаты этого сигнала для подстройки весовых коэффициентов фильтр [10].

Возможен другой вариант адаптации, при котором образцовый сигнал не используется. Такой режим работы называется «слепой адаптацией» [10]. Вне сомнения, «слепая адаптация» является более сложной вычислительной задачей, чем адаптация с использованием образцового сигнала [11].

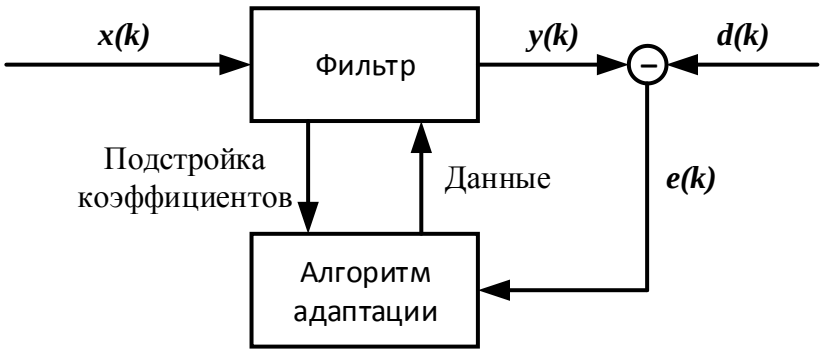


Рисунок 1 – Общая структура адаптивных фильтров

Может показаться, что алгоритмы с использованием образцового сигнала лишены практического смысла, поскольку выходной сигнал должен быть заранее известен. Однако, существует множество практических задач, при решении которых образцовый сигнал оказывается доступен. В ряде случаев при этом полезным сигналом является не выходной сигнал фильтра, а сигнал ошибки, то есть разность между образцовым сигналом и выходным сигналом адаптивного фильтра [10].

1.3 Виды эхо-сигналов

По происхождению, помехи разделяют на акустические и электрические. Соответственно, выделяют два вида эхо-сигналов: электрическое и акустическое эхо. Причиной первого является нарушение условий баланса дифференциальных систем, используемых в проводных системах для развязки двух и четырехпроводных линий, вследствие чего часть энергии принимаемого сигнала в виде искаженной и задержанной копии попадает в цепи передачи, тем самым, ухудшая качественные показатели системы связи [2].

Акустический эхо-сигнал возникает в том случае, когда звуковая волна, отражаясь от близлежащих объектов, возвращается обратно к источнику колебаний. В системах телекоммуникаций это происходит в случаях, когда звуковая волна, источником которой является громкоговоритель абонентского оборудования, или сам абонент, попадает в микрофонную цепь за счет

переотражений или плохой развязки приемной и передающей цепей. Хотя акустическое эхо присутствует как в проводных, так и в беспроводных системах телекоммуникаций, наибольшее влияние этого негативного эффекта сказывается в таких технических приложениях как радиотелефония, системы телеконференций и мобильная связь [2].

Причинами появления этого типа помех являются:

– пульсации выпрямленного тока в источниках питания, электрические и магнитные «наводки» от цепей переменного тока;

– хаотическое движение (флуктуации) электронов или других заряженных частиц в проводниках, резисторах, электронных приборах;

посторонние электрические и магнитные поля, создаваемые трансформаторами, электродвигателями, цепями переменного тока, соседними цепями в многопарных кабелях;

– специфические недостатки носителей записи, обусловленные неоднородностью магнитной и киноленты, грампластинки, а также копирэффект и модуляционный шум. «Шум» определяется как беспорядочные колебания различной физической природы, отличающиеся сложностью временной и спектральной структуры;

– посторонние источники в радиодиапазонах (другие радиостанции, атмосферные электрические разряды, промышленная, медицинская, бытовая электроаппаратура) [12].

1.4 Алгоритмы адаптации

В настоящее время существует множество алгоритмов адаптации, обладающих различными свойствами, однако многие из них являются производными либо модификациями одного из методов поиска экстремума (чаще всего минимума) целевой функции. Выделены четыре основные группы алгоритмов, наиболее часто применяющихся в задачах адаптивной фильтрации [2]:

1. алгоритм LMS,
2. алгоритм RLS,
3. алгоритм аффинных проекций,
4. решетчатые (лестничные) алгоритмы.

Рассмотрим каждый из алгоритмов.

1.4.1 Алгоритм LMS

Одним из наиболее распространенных адаптивных алгоритмов, основанном на поиске минимума целевой функции (1), является метод наискорейшего спуска (Least Mean Square). При использовании данного способа оптимизации вектор ВК фильтра $w(k)$ должен обновляться следующим образом [13]:

$$J(w) = \overline{e^2(k)} \rightarrow min, \tag{1}$$

где $\overline{e^2(k)}$ – средний квадрат ошибки, k – натуральное число, шаг.

$$w(k+1) = w(k) + \mu e(k)u(k), \tag{2}$$

где μ – положительный коэффициент, называемый размером шага, $u(k)$ – вектор-столбец содержимого линии задержки фильтра на k-ом шаге.

Основным достоинством алгоритма LMS является предельная вычислительная простота – для подстройки коэффициентов фильтра на каждом шаге нужно выполнить $N + 1$ пар операций «умножение–сложение». «Платой» за простоту является «медленная» сходимость и повышенная дисперсия ошибки в установившемся режиме – коэффициенты фильтра всегда флуктуируют вокруг оптимальных значений, что и увеличивает уровень выходного шума [13].

1.4.2 Алгоритм RLS

Отличительной особенностью алгоритма RLS (Recursive Least Square) является его целевая функция. Этот алгоритм направлен не на уменьшение среднеквадратичной ошибки, как в LMS, а на минимизацию нормы ошибки (3). Подробно алгоритм RLS рассматривается в [14].

$$J(w) = \sum_{k=0}^{K-1} |e(k)|^2 \rightarrow min, \tag{3}$$

где K – вектор-столбец коэффициентов усиления.

Формула для вычисления коэффициентов выглядит следующим образом:

$$w(k+1) = w(k) + K(k+1)e(k+1). \tag{4}$$

Основным достоинством алгоритма RLS является «быстрая» сходимость по сравнению с LMS алгоритмом. Однако достигается это за счет значительно более высокой вычислительной сложности. При оптимальной организации вычислений для обновления коэффициентов фильтра на каждом такте требуется (2,5N2 + 4N) пар операций «умножение–сложение» [11].

1.4.3 Алгоритм аффинных проекций

Как упоминалось выше LMS алгоритм является достаточно простым с точки зрения реализации, но имеет достаточно «медленную» сходимость, алгоритм RLS, напротив, имеет высокую сложность, но имеет «быструю» сходимость. Алгоритм аффинных проекций занимает промежуточное место по вычислительной сложности и производительности среди рассматриваемых алгоритмов [15].

Для вычисления ВК в методе аффинных проекций применяться формула:

$$w(k+1) = w(k) + \mu X_{NL}(k)e(k), \quad (5)$$

где $X_{NL} = [x_N(k), x_N(k-1), \dots, x_N(k-L+1)]$;

$x_N(k) = [x(k), x(k-1), \dots, x(k-N_m+1]$;

$x_N(k)$ – вектор входных сигналов.

Основным достоинством данного алгоритма является «более быстрая» сходимость, чем у LMS алгоритма. Некоторые модификации этого алгоритма имеют эффективность близкую к эффективности RLS алгоритмов [15]. Однако, такая эффективность достигается посредством увеличения весовых коэффициентов, что влечет за собой увеличение вычислительной сложности [16].

1.4.4 Решетчатые (лестничные) алгоритмы

Лестничные алгоритмы получили свое название из-за структуры процедур вычисления ошибок линейного предсказания и моделирования обрабатываемых сигналов, напоминающих по форме лестницу. Особенностью лестничных алгоритмов является отсутствие вычислений весовых коэффициентов адаптивного фильтра в явном виде. Все вычисления в лестничных алгоритмах являются скалярными и выполняются на каждой k -й итерации по времени в течение N шагов при изменении порядка адаптивного фильтра от 1 до N.

Лестничные алгоритмы следует отнести к «быстрым» алгоритмам адаптивной фильтрации. Вычислительная сложность лестничных алгоритмов несколько больше сложности других RLS-алгоритмов. Однако лестничные алгоритмы известны своей устойчивостью. Кроме того, каскадная структура вычислений, характерная лестничным алгоритмам, способствует не только эффективной программной, но и аппаратной реализации таких алгоритмов в виде СБИС [17].

Основной причиной «непопулярности» данных алгоритмов является требование реализации адаптивного цифрового фильтра (АЦФ) по решетчатой структуре, которая является концептуально более сложной по сравнению с прямой формой построения цифрового фильтра [2].

Основные результаты анализа известных методов адаптивной фильтрации, представлены в таблице 1.

Таблица 1 – Основные методы адаптивной фильтрации

Метод	Скорость сходимости	Вычислительная сложность
Алгоритм LMS	Медленная	Низкая
Алгоритм RLS	Быстрая	Высокая
Алгоритм аффинных проекций	Средняя	Средняя
Решетчатые алгоритмы	Быстрая	Высокая

1.5 Выводы

На основании полученных результатов предварительного анализа можно сделать обоснованное предположение о том, что удовлетворительные результаты могут быть получены при использовании рекурсивного цифрового фильтра и реализации эхо-компенсатора на аппаратном уровне, например, на базе ПЛИС.

Такой подход позволит увеличить скорость обработки сигнала, т.е. обеспечить режим реального времени, что, предположительно, позволит обрабатывать сигналы в большем частотном диапазоне – расширить сферу применения и обеспечить универсальность фильтра в отличии от известных решений, реализованных на базе DSP-процессоров.

Однако, требуется исследования основных характеристик фильтра: сходимости, точности и предельных значений в заданном частотном диапазоне – при практической реализации. Для этого необходимо разработать и исследовать модель фильтра, а также изготовить и исследовать лабораторный

макет для практического подтверждения полученных результатов моделирования.

2 Разработка модели адаптивного фильтра эхо-компенсации

2.1 Требования к модели фильтра эхо-компенсации

При построении модели фильтра следует учесть, что фильтр эхо-компенсации должен функционировать в режиме реального времени и осуществлять автоматическую подстройку на изменение состояния окружающей среды. Скорость адаптации фильтра должна быть достаточной для обеспечения высокого качества выходного сигнала.

Такой адаптивный фильтр должен пропускать без искажения полезные составляющие сигнала и значительно ослабить, а в отдельных случаях полностью исключить помехи, т. е. уменьшить любые искажения входного сигнала [12].

Предположительно, с этой задачей может справиться RLS фильтр. Как отмечалось ранее, среди адаптивных алгоритмов RLS имеет достаточно «высокую сходимость».

2.2 RLS алгоритм

Особенностью RLS фильтра является то, что выходное значение каждого отсчета формируется не только манипуляциями с текущими входными значениями, но и со значениями выходных отсчетов, вычисленных в предыдущих циклах расчетов (рисунок 2) [12]. Это позволяет избежать больших и неоправданных вычислительных затрат на вычисление весовых коэффициентов фильтра.

При использовании адаптивного алгоритма RLS необходимо на каждом временном такте выполнить следующие шаги [11].

При поступлении новых входных данных $x(k)$ производится фильтрация сигнала с использованием текущих коэффициентов фильтра $w(k-1)$ и вычисление величины ошибки воспроизведения образцового сигнала.

$$y(k) = x(k)^T w(k-1), \tag{6}$$

$$e(k) = d(k) - y(k). \tag{7}$$

Рассчитывается вектор-столбец коэффициентов усиления по формуле (8) (следует отметить, что знаменатель дроби в следующих двух формулах является скаляром, а не матрицей):

$$K(k) = \frac{P(k-1)x(k)}{1 + x(k)^T P(k-1)x(k)}. \tag{8}$$

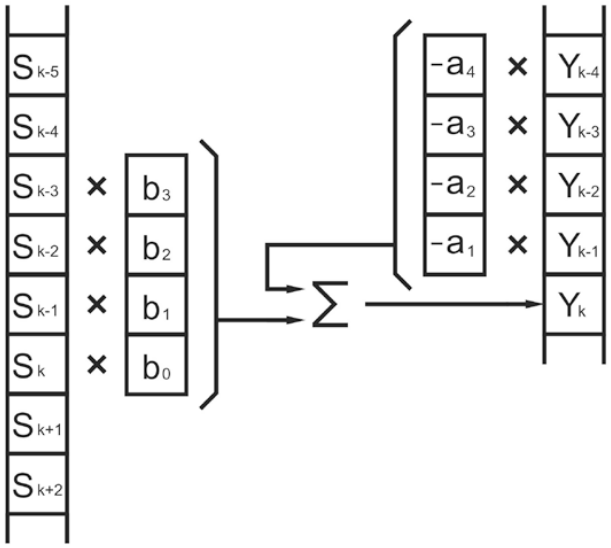


Рисунок 2 – Схема RLS фильтра

Производится обновление оценки обратной корреляционной матрицы сигнала по формуле:

$$P(k) = P(k-1) - \frac{P(k-1)x(k)x(k)^T P(k-1)}{1 + x(k)^T P(k-1)x(k)}. \tag{9}$$

Производится обновление коэффициентов фильтра по формуле:

$$w(k) = w(k-1) + K(k)e(k). \tag{10}$$

Начальное значение вектора весовых коэффициентов w обычно принимается нулевым, а в качестве исходной оценки матрицы P используется диагональная матрица следующего вида:

$$P(-1) = \begin{bmatrix} \infty & 0 & 0 & \cdots & 0 \\ 0 & \infty & 0 & \cdots & 0 \\ 0 & 0 & \infty & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \infty \end{bmatrix}. \tag{11}$$

На практике диагональ матрицы заполняется большими положительными значениями, например, $100/\sigma_x^2$.

2.3 Выбор инструмента для создания модели фильтра

На сегодняшний день существует целый ряд математических пакетов, реализующих разнообразные численные методы и производящих аналитические математические преобразования. Наиболее известными сегодня являются пакеты прикладных программ и математические библиотеки [18,19]:

- MATLAB (фирма The MathWorks);
- Maple (фирма Waterloo Maple Inc.);
- Mathematica (фирма Wolfram Research);
- MathCAD (фирма PTC).

MATLAB (MATrix LABoratory) – это система предназначенный для осуществления любых численных расчетов и моделирования технических и физических систем, а также выполнения научных и инженерных расчетов при работе с массивами данных [20].

Maple – программный пакет, система компьютерной алгебры. Система Maple предназначена для символьных вычислений, хотя имеет ряд средств и для численного решения дифференциальных уравнений и нахождения интегралов. Обладает развитыми графическими средствами [21].

MathCAD является интегрированной системой программирования, ориентированной на проведение математических и инженерно-технических расчетов [22].

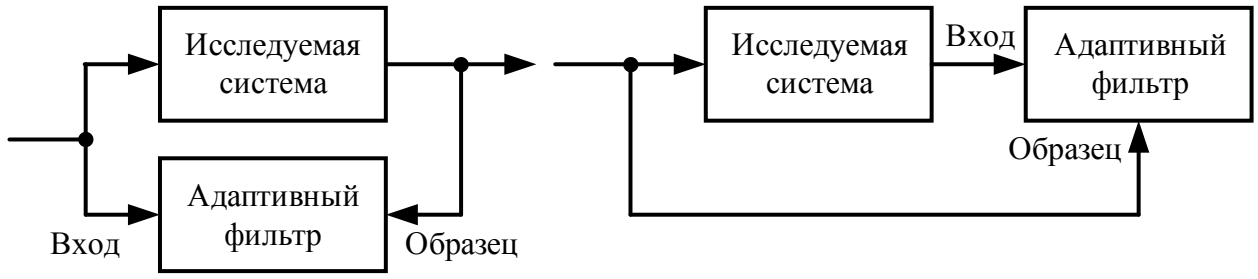
Mathematica – система компьютерной алгебры. Содержит множество функций, как для аналитических преобразований, так и для численных расчетов. Кроме того, программа поддерживает работу с графикой и звуком [23].

Все выше перечисленные математические пакеты имеют обширный функционал, большой набор инструментов и библиотек для работы во многих областях, включая цифровую обработку сигналов [18,19,24]. Однако MATLAB отличается высокой скоростью численных выражений, а матрицы являются основой автоматического решения и составления уравнений состояния динамических объектов и систем [18].

Таким образом, в качестве инструмента моделирования адаптивного фильтра эхо-компенсации был выбран пакет прикладных программ MATLAB.

2.4 Идентификация систем

Все способы применения адаптивных фильтров сводятся к решению задачи идентификации, т. е. определения характеристик некоторой системы. Существующие подходы к решению задачи идентификации отклика некоторого тракта, определения характеристик исследуемой системы можно подразделить на прямую идентификацию и обратную [11,12] (рисунок 3).



а) прямая идентификация б) обратная идентификая

Рисунок 3 – Идентификация характеристик исследуемой системы

В случае прямой идентификации входной сигнал является общим для исследуемой системы и адаптивного фильтра, а выходной сигнал системы служит для адаптивного фильтра образцовым сигналом (рисунок 3, а). В процессе адаптации временные и частотные характеристики фильтра будут стремиться к соответствующим характеристикам исследуемой системы [11].

При обратной идентификации адаптивный фильтр включается последовательно с исследуемой системой (рисунок 3, б). Выходной сигнал системы поступает на вход адаптивного фильтра, а входной сигнал системы является образцом для адаптивного фильтра. Таким образам, фильтр стремится компенсировать влияние системы и восстановить исходный сигнал, устранив внесенные системой искажения [11].

При эхо-компенсации решается задача прямой идентификации тракта распространения эхо. На вход адаптивного фильтра поступает искомый сигнал, а в качестве образцового сигнала используется сигнал, содержащий эхо. Адаптивный фильтр формирует оценку эхо-сигнала, а сигнал ошибки представляет собой принимаемый сигнал, очищенный от эха. Схема эхо-компенсации представлена на рисунке 4.

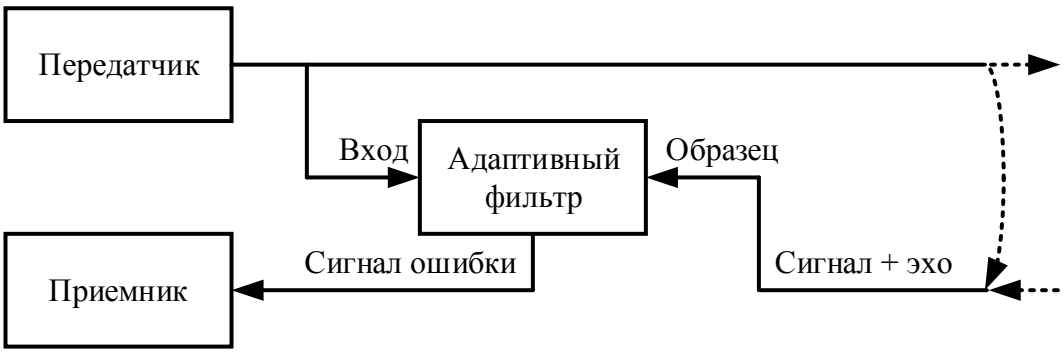


Рисунок 4 – Эхо-компенсация, осуществляемая с помощью адаптивного фильтра

2.5 Моделирование адаптивного фильтра эхо-компенсации

В рамках проводимой работы разработана модель адаптивного эхо-компенсатора. Модель создавалась при помощи библиотеки визуального программирования Simulink с использованием MATLAB System Block, который предоставляет следующие возможности:

– совместное использование одного и того же объекта System в MATLAB и Simulink;

– выделенная интеграция объектов системы с Simulink;

– модульный тест алгоритма в MATLAB перед его использованием в Simulink;

– настройка диалогового окна;

– эффективное моделирование с лучшей инициализацией;

– обрабатывать состояния;

– настройка портов блока с метками;

– доступ к двум режимам симуляции.

Реализованная модель адаптивного эхо-компенсатора приведена на рисунке 5.

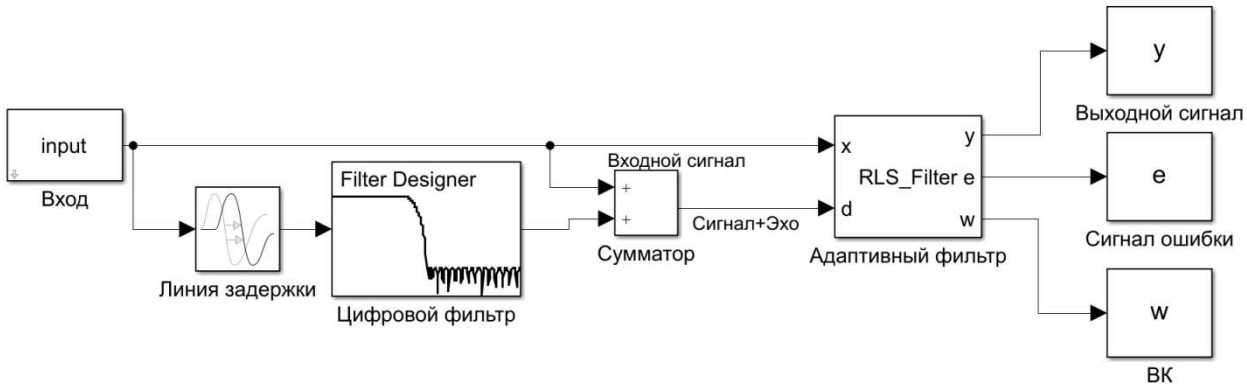


Рисунок 5 – Модель адаптивного эхо-компенсатора

В начале входной сигнал поступает на вход адаптивного фильтра (вход x) и на вход линии задержки. Эхо сигнал, пройдя через цифровой фильтр, на сумматоре складывается с входным сигналов и поступает на вход адаптивного фильтра (вход d). На выходе считываются выходной сигнал y, сигнал ошибки e и вектор весовых коэффициентов w.

Реализованный интерфейс блока адаптивного фильтра представлен на рисунке 6.

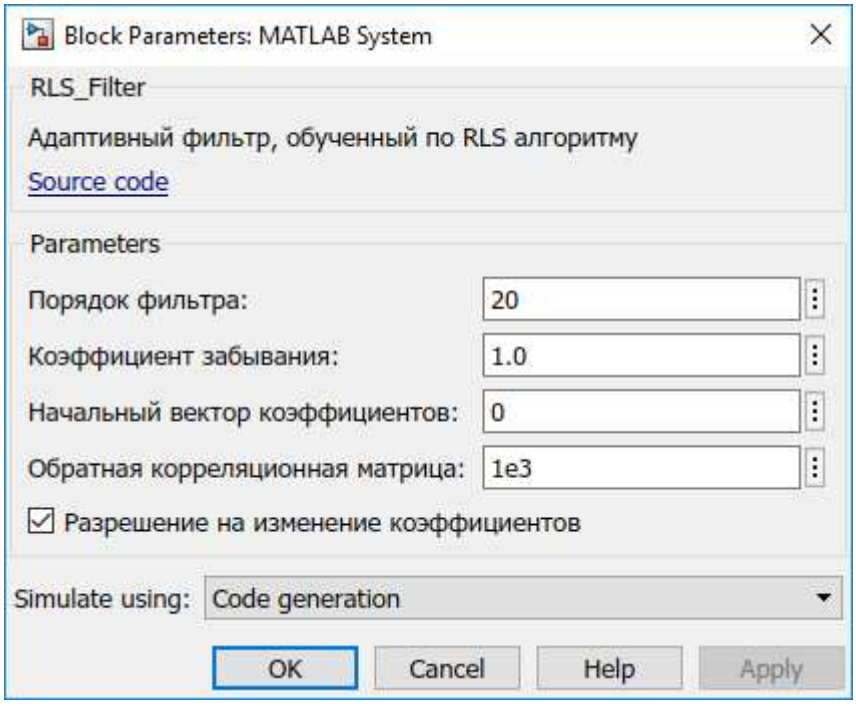


Рисунок 6 – Диалоговое окно блока адаптивного фильтра

В диалоговом окне можно настроить следующие параметры фильтра:

– порядок фильтра (количество весовых коэффициентов);

– коэффициент забывания (экспоненциально уменьшает вес прошлых значений сигнала ошибки);

– начальный вектор коэффициентов (позволяет инициализировать весовые коэффициенты);

– обратная корреляционная матрица (определяет величину значений диагонали обратной корреляционной матрицы);

– разрешение на изменение коэффициентов.

Флаг для разрешения на изменение коэффициентов позволяет включить (отключить) адаптацию, т.е. зафиксировать коэффициенты фильтра. Этот параметр можно использовать в том случае, когда фильтр адаптировался под входной сигнал.

Параметр *Simulate using* позволяет выполнять моделирование посредством интерпретированного выполнения или генерации кода на C/C++.

Исходный код адаптивного фильтра представляет собой класс, разработанный на языке m. Исходный код программы приведен в приложении А.

2.6 Результаты моделирования

При моделировании реконфигурируемого адаптивного цифрового фильтра использовались сигналы разной сложности: сигналы математических функций (синус, косинус и т.д.) разной частоты, акустические сигналы (реальные записи речи и музыки).

На рисунке 7 приведены графики моделирования тригонометрической функции.

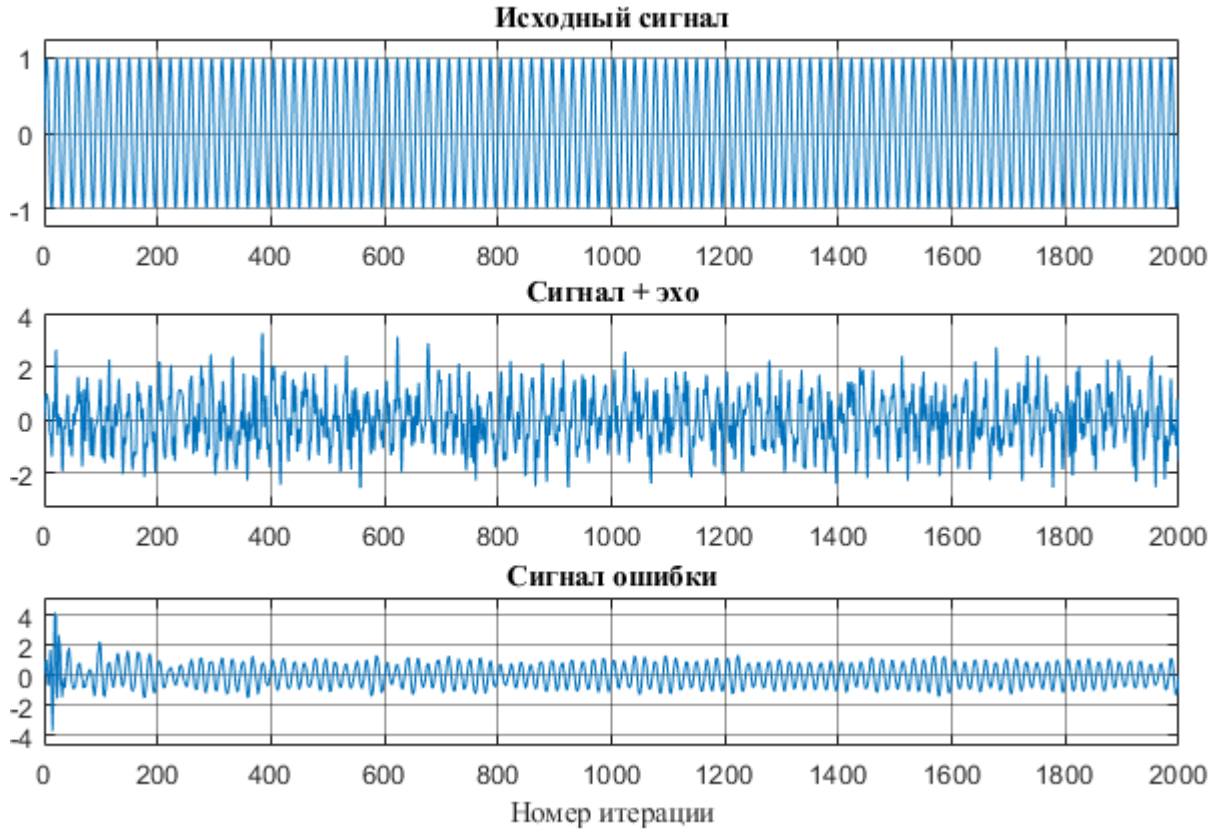


Рисунок 7 – Графики моделирования сигнала тригонометрической функции

Из графика видно, что в начальный момент времени сигналы ошибки и эха идентичны, однако в процессе адаптации ошибка практически полностью исчезает.

На рисунках 8 а, в, д представлены графики весовых коэффициентов в начальный, промежуточный и конечный моменты времени, а на рисунках 8 б, г, е – соответствующие графики амплитудно-частотных характеристик. По графикам видно, что после основной части адаптации значения весовых коэффициентов и амплитудно-частотной характеристики существенного изменения не имеет.

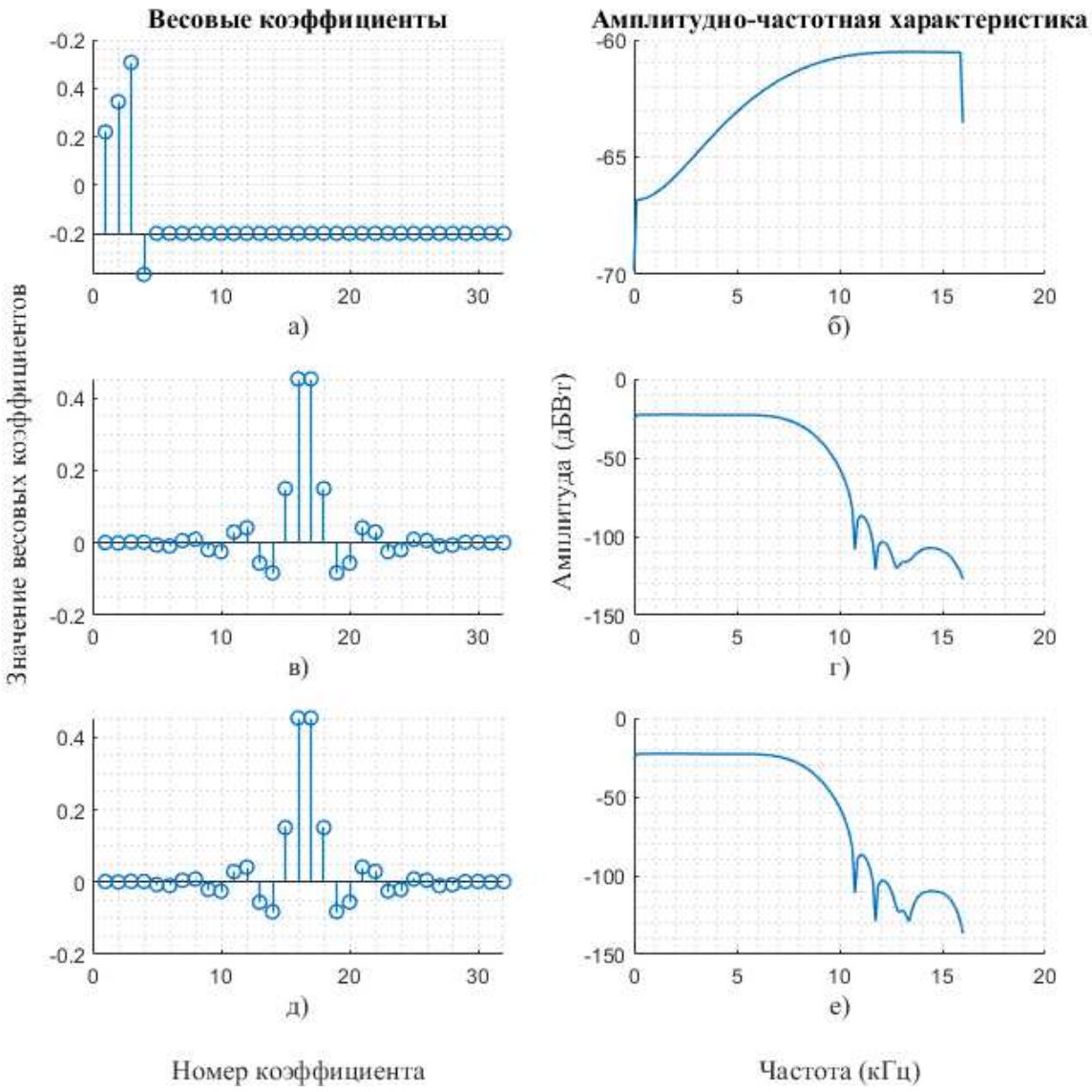


Рисунок 8 – Графики весовых коэффициентов и амплитудно-частотных характеристик адаптивного фильтра

На рисунке 9 приведены графики моделирования сигнала речевого диапазона частотой 8 кГц. При прослушивании записи исходного сигнала отчетливо слышно все звуки, однако при прослушивании записи «сигнал+эхо» почти невозможно разобрать речь. После фильтрации зашумленного сигнала прослушивая записи сигнала на выходе фильтра отчетливо разбираются слова, но на фоне слышен незначительный шум. При моделировании данного сигнала с разными весовыми коэффициентами ошибка в среднем составляла не более 20 % от исходного сигнала.

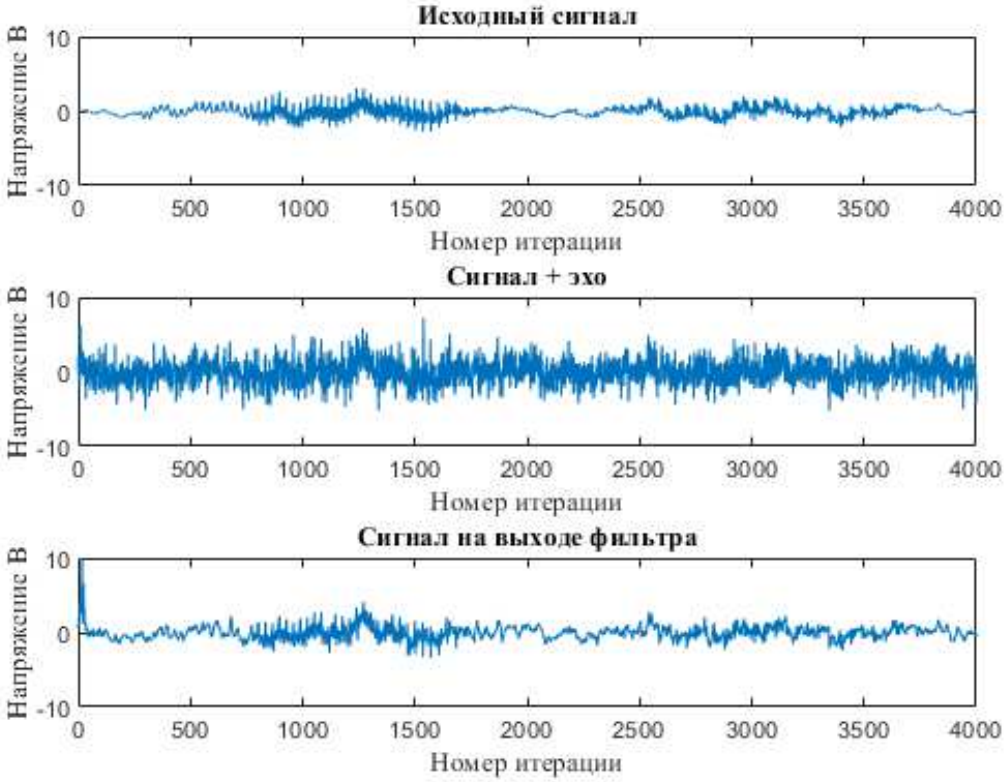


Рисунок 9 – Графики моделирования сигнала на частоте 8 кГц

На рисунке 10 приведены графики моделирования сигнала речевого диапазона частотой 44 кГц. При частоте выше 40 кГц в работе фильтра наблюдаются сбои. В отфильтрованном сигнале почти невозможно определить исходный сигнал.

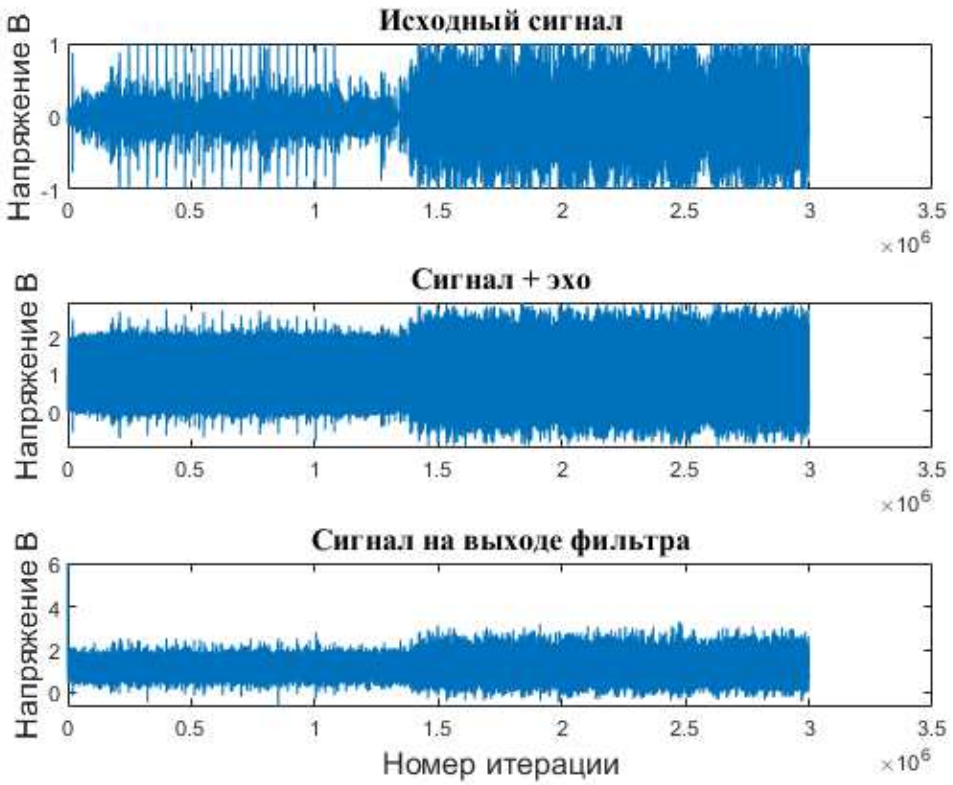


Рисунок 10 – Графики моделирования сигнала на частоте 44 кГц

2.7 Выводы

Определены требования к реконфигурируемому адаптивному фильтру эхо-компенсации. Рассмотрен алгоритм адаптации по методу RLS. Выбрана среда программного моделирования. Разработана программная модель адаптивного фильтра эхо-компенсации. Проведено моделирование фильтра с использованием сигналов разной сложности.

Моделирование показало, что при использовании адаптивного фильтра основная компенсация помехи происходит в течение первых 200 отсчетов. При дальнейшей адаптации весовые коэффициенты фильтра существенной динамики не имеют. Амплитудно-частотная характеристика также практически не изменяется. На частотах акустического диапазона фильтр справляется со своей основной задачей, однако, на частотах выше 40 кГц адаптивный фильтр начинает давать сбои. Это связано с тем, что фильтр при высокой частоте

входного сигнала не успевает адаптироваться, так как вычисление весовых коэффициентов требует некоторого времени, которое зависит от аппаратной или программной платформы, на которой реализован адаптивный фильтр. Предположительно, внесение во входной сигнал пауз, т.е. деление сигнала на части, поможет избежать сбоев на высоких частотах, поскольку пауза позволит адаптироваться под новую часть сигнала.

3 Разработка аппаратной реализации фильтра эхо-компенсации

3.1 Выбор аппаратной платформы для реализации лабораторного макета

Для обеспечения режима реального времени необходимо использовать аппаратную платформу, которая бы позволяла полностью на аппаратном уровне реализовывать все функции адаптивного фильтра.

Рассмотрены 3 варианта реализации адаптивного фильтра:

- на базе заказной (полузаказной) ИС;
- на базе DSP процессора;
- на базе ПЛИС.

Вариант реализации на базе заказной или полузаказной ИС не подходит, поскольку является трудозатратным, длительным и дорогостоящим процессом.

Вариант разработки на базе DSP процессоров не позволит достичь желаемых результатов, поскольку некоторые функции придется реализовывать на программном уровне.

В современном мире для прототипирования подобных систем используется ПЛИС. Стоит отметить, что современные ПЛИС имеют в своем составе DSP процессора, математические сопроцессоры, ускорители и т.д.

Исходя из этого было принято решение использовать ПЛИС для реализации лабораторного макета. Именно ПЛИС позволит определить с высокой точностью предельные значения при использовании фильтра

В качестве базиса были рассмотрены две ведущие компании по производству ПЛИС: Intel, Xilinx. Современные ПЛИС этих компании позволяют реализовывать весь требуемым набор функций для реализации адаптивного фильтра эхо-компенсации.

Сравнительный анализ ПЛИС компаний Intel и Xilinx показал, что ПЛИС компании Intel в большей степени соответствуют требованиям к реализации адаптивного фильтра [25–27]. Поэтому в качестве аппаратной платформы для

реализации лабораторного макета эхо-компенсатора были выбраны ПЛИС компании Intel.

3.2 Особенности реализации адаптивных фильтров на ПЛИС

Эффективная реализация сложных алгоритмов на ПЛИС требует специальных архитектур и модификации алгоритмов.

При программном моделировании адаптивных фильтров вычисления обычно выполняются в арифметике с плавающей точкой. Однако ПЛИС не имеет встроенных модулей по обработке чисел с плавающей точкой. Реализация математических операций с плавающей точкой требует больших аппаратных затрат, что не целесообразно при проектировании систем реального времени на базе ПЛИС.

3.2.1 Переход к арифметике с фиксированной точкой

Хотя вычисление арифметики с плавающей точкой в ПЛИС возможно, это обычно достигается с помощью специального модуля с плавающей точкой [28]. Эти устройства очень дороги с точки зрения логических ресурсов. Из-за этого во всем проекте может использоваться только небольшое количество модулей с плавающей точкой, и они должны совместно использоваться процессами. Это не дает полного преимущества распараллеливания, которое возможно в ПЛИС, и, следовательно, не является наиболее эффективным методом. Поэтому все расчеты должны быть проведены в арифметике с фиксированной точкой, но это может привести к некоторым ошибкам. Основные ошибки в DSP [29]:

– ошибки квантования АЦП – результат представления входных данных ограниченным числом битов;

– ошибки квантования весовых коэффициентов – вызваны представлением весовых коэффициентов конечным числом битов;

– ошибки переполнения – вызваны сложением (умножением) двух больших чисел одного знака, что приводит к результату, который превышает допустимую длину слова;

– ошибки округления – происходит, когда результат умножения округляется (или усекается) до ближайшего дискретного значения или допустимой длины слова.

3.2.2 Модификация RLS алгоритма

Подходящий компромисс для борьбы с потерей точности при переходе от представления с плавающей запятой к представлению с фиксированной запятой заключается в сохранении ограниченного числа десятичных знаков. Обычно достаточно двух-трех десятичных знаков, но число, необходимое для сходимости данного алгоритма, должно быть найдено путем экспериментов. Например, при программном моделировании цифрового фильтра определяется, что для точной обработки данных достаточно двух десятичных знаков. Это может быть легко получено путем умножения коэффициентов фильтра на 100 и усечения до целочисленного значения. Разделив выход на 100, вы получите ожидаемое значение. Поскольку умножение и деление на две степени могут быть легко осуществлены в ПЛИС путем сдвига битов, степень два может быть использована для упрощения процесса. В этом случае можно умножить на 128, что потребует семь дополнительных аппаратных битов.

Для простой свертки, умножение на заданный масштабный коэффициент и последующее деление выходных данных на тот же масштабный коэффициент не влияет на вычисления. Для более сложного алгоритма есть алгоритм, который необходим для работы этой схемы.

Алгоритм корректировки масштабного коэффициента [30]:

– определить масштаб;

– посредством моделирования найти необходимую точность (количество десятичных знаков);

– масштабный коэффициент равный точности округлить до степени двойки;

– разделить на масштабный коэффициент, когда умножены два масштабированных значения;

– умножить на масштабный коэффициент, когда два масштабированных значения разделены.

Примечание, если производится умножение (деление) на значение меньше единицы, деление (умножение) заменяется на обратную величину.

Таким образом, следуя алгоритму корректировки масштабного коэффициента RLS алгоритм принимает следующий вид:

$$y(k) = \frac{x(k)^T w(k-1)}{scale}, \tag{12}$$

$$e(k) = d(k) - y(k), \tag{13}$$

$$K(k) = \frac{scale^2\, P(k-1)x(k)}{1 + x(k)^T P(k-1)x(k)}, \tag{14}$$

$$P(k) = P(k-1) * scale - \frac{K(k)x(k)^T P(k-1)}{1 * scale^2}, \tag{15}$$

$$w(k) = w(k-1) + \frac{K(k)e(k)}{scale}. \tag{16}$$

3.3 Аппаратная модель адаптивного эхо-компенсатора

В настоящее время для программирования ПЛИС компании Intel используется интегрированная среда проектирования Quartus Prime [31] (рисунок 11).

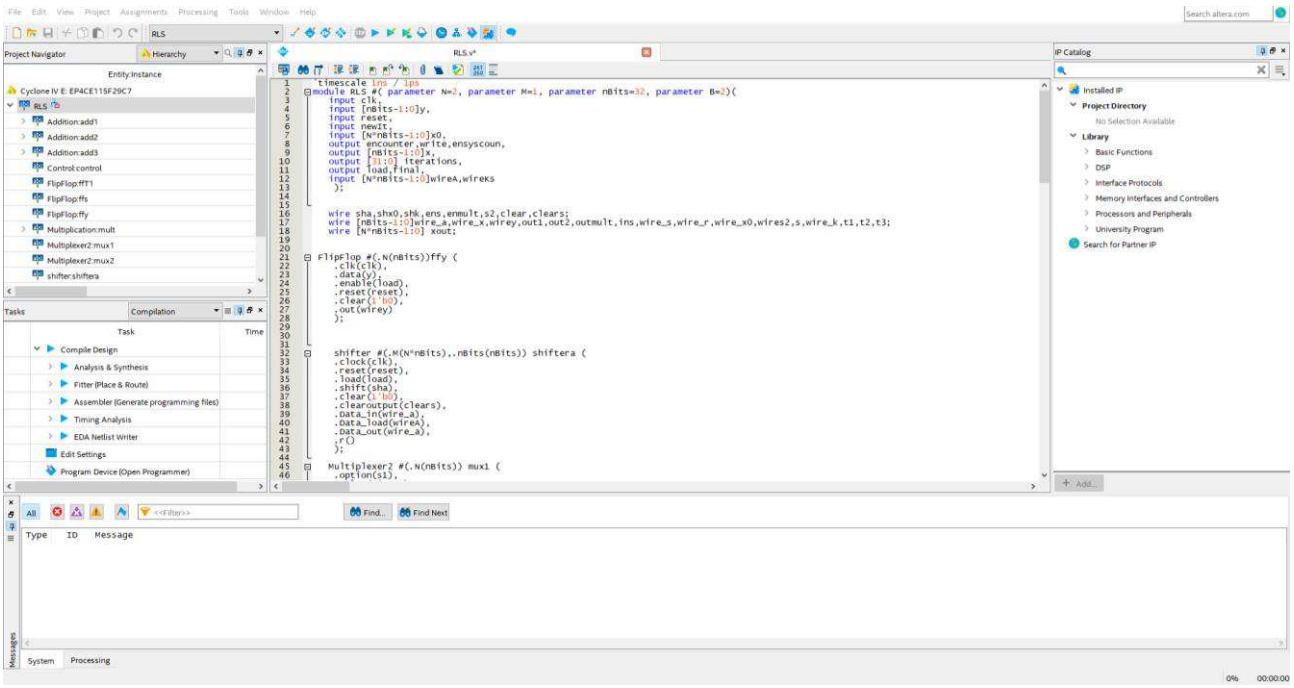


Рисунок 11 – Интерфейс Quartus Prime

Программировать ПЛИС возможно двумя способами:

– использовать графическое (схемное) описание;

– использовать язык описания аппаратуры.

Языки описания аппаратуры имеют явные преимущества перед графическим (схемным) описанием [32]:

– скорость разработки;

– наглядность работы схемы;

– портируемость и переносимость проектов;

– управляемость кода.

Поэтому при реализации реконфигурируемого адаптивного цифрового фильтра на ПЛИС в качестве языка описания аппаратуры использовался Verilog.

На рисунке 12 представлена схема адаптивного фильтра, полученная в результате синтеза проекта. Условно схему можно разделить на две части: управляющий автомат и комбинационную схему. Комбинационная схема представляет собой схему, в которой с помощью управляющих сигналов

управляющего автомата вычисляются переменные по формулам (12-16). С кодом синтезированной схемы можно ознакомиться в приложении Б.

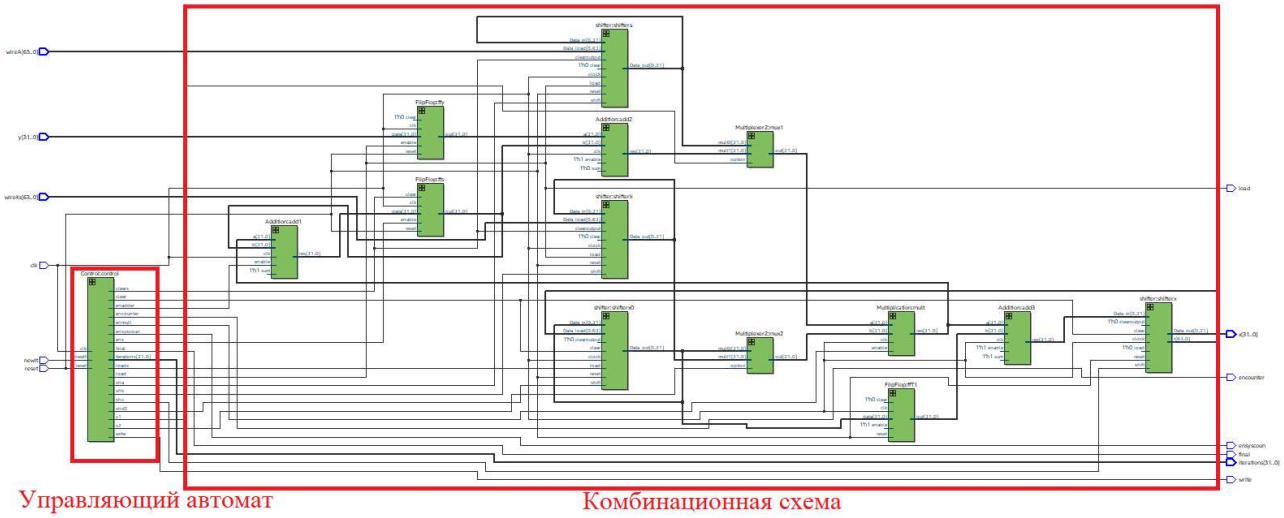


Рисунок 12 – Синтезированная схема реконфигурируемого адаптивно цифрового фильтра

Для тестирования разработанного модуля был собран лабораторный макет, а также оборудовано рабочее место (рисунок 13).

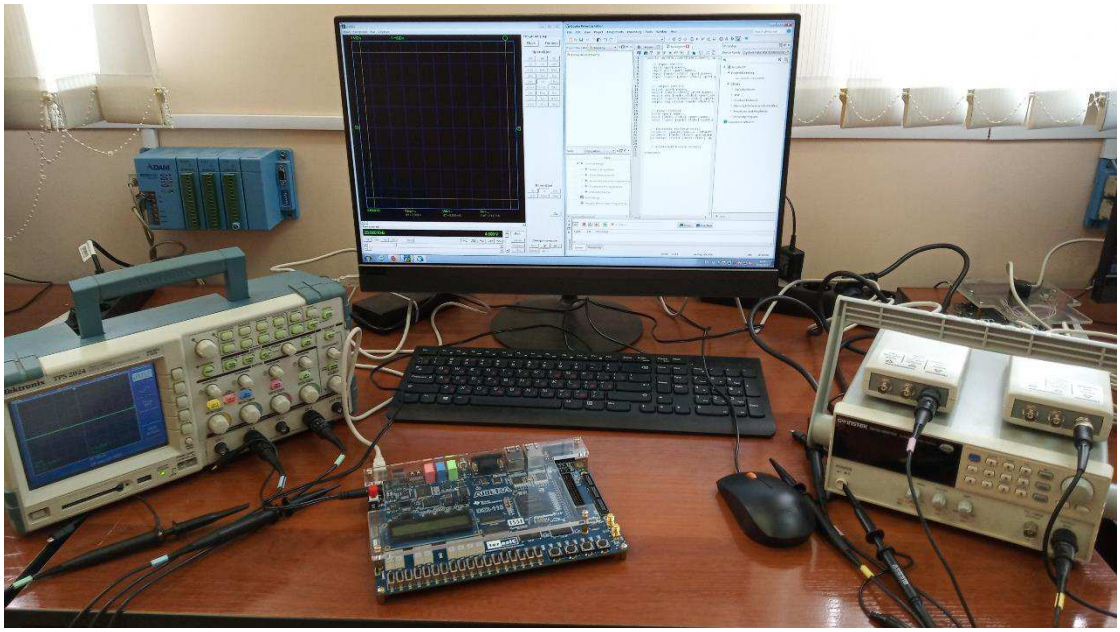


Рисунок 13 – Рабочее место

В состав рабочего места входит:

– компьютер;

– отладочная плата DE2-115;

– осциллограф Tektronix TPS 2024;

– два осциллографа PV6501 (используются для генерации импульсов);

– генератор импульсов GwINSTEK SFG-2010.

Для подачи сигналов с генераторов импульсов использовались входы под Jack 3,5, поскольку в отладочной плате DE2-115 АЦП и ЦАП выведены на данные разъемы (рисунок 14).

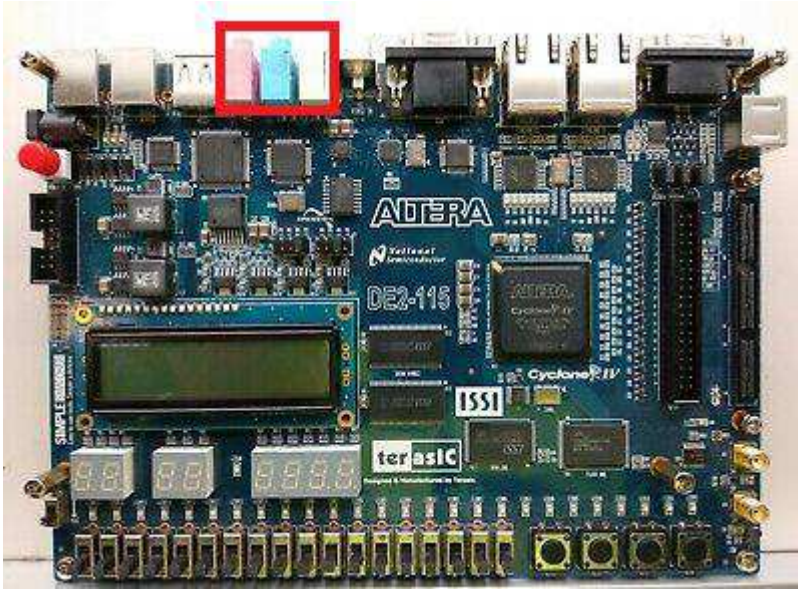


Рисунок 14 – Отладочная плата DE2-115

3.4 Результаты лабораторных испытаний

Тестирование аппаратной модели реконфигурируемого адаптивного цифрового фильтра эхо-компенсации проводилось:

– в среде ModelSim;

– на отладочной плате DE2-115.

В ходе тестирования аппаратной модели эхо-компенсатора использовались сигналы разной сложности: сигналы математических функций (синус, косинус и т.д.) разной частоты, акустические сигналы (реальные записи речи и музыки).

На рисунке 15 приведены графики моделирования в ModelSim. По графику отфильтрованного сигнала отчетливо видно влияние перехода к арифметике с фиксированной точкой.

На рисунке 16 представлен график сигнала ошибки.

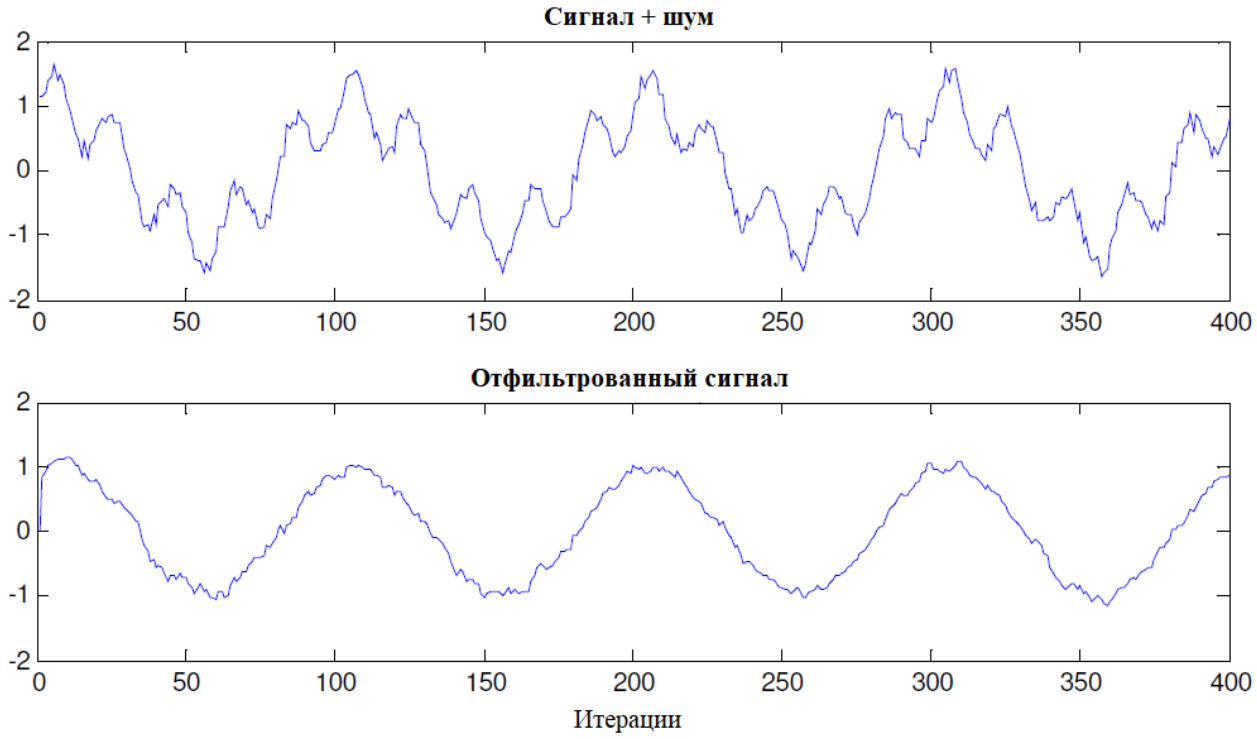


Рисунок 15 – Графики аппаратного моделирования реконфигурируемого адаптивного фильтра в среде ModelSim

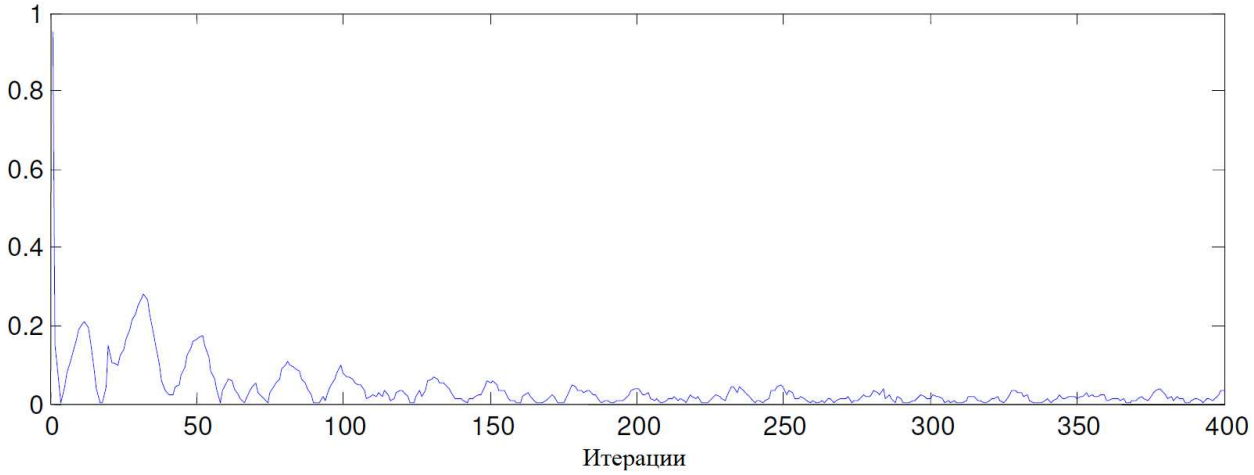


Рисунок 16 – График сигнала ошибки

На рисунке 17 приведены результаты тестирования эхо-компенсатора на отладочной плате DE2-115. Зашумленный сигнал выделен синим цветом, отфильтрованный сигнал – зеленым.

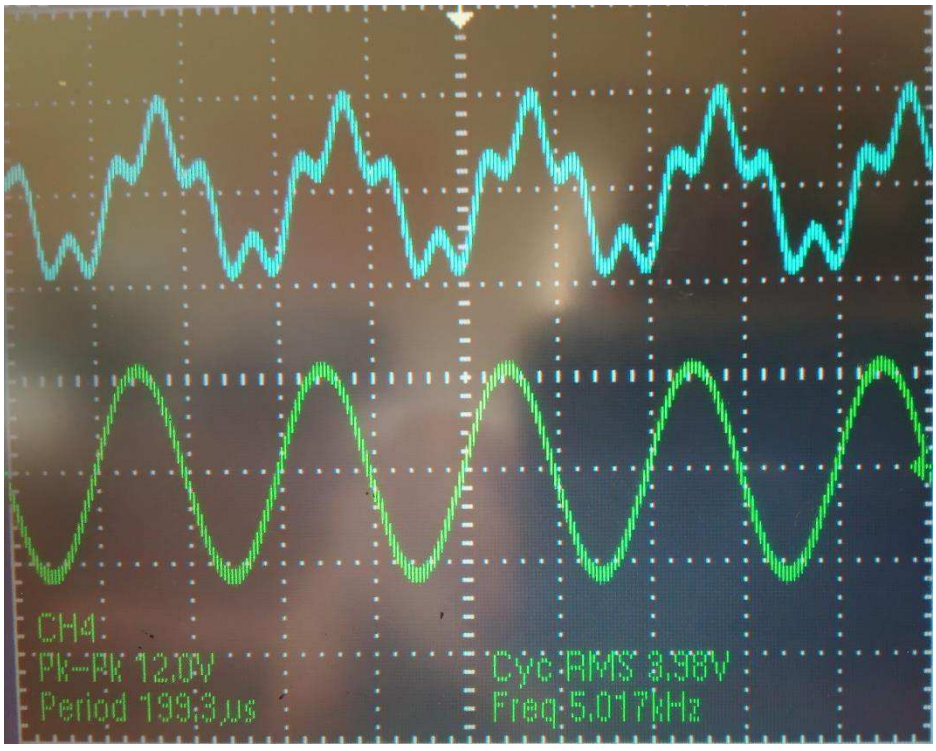


Рисунок 17 – Результаты лабораторных испытаний

В качестве исходного сигнала использовалась синусоида с амплитудой 4 В и частотой 5 кГц. В качестве шума использовался пилообразный сигнал с

амплитудой 4 В и частотой 20 кГц. Тестирование показало, что разработанный модуль адаптивного фильтра работает корректно и успешно справляется фильтрацией сигналов.

3.5 Выводы

Для обеспечения режима реального времени в качестве аппаратной платформы была выбрана ПЛИС. Рассмотрены ПЛИС двух ведущих производителей: Intel, Xilinx. Определены особенности реализации адаптивных алгоритмов на ПЛИС. Основной проблемой является переход к арифметике с плавающей точкой. Для достижения требуемой сходимости RLS алгоритма, была произведена модификация данного алгоритма. Разработана аппаратная модель и лабораторный макет эхо-компенсатора, а также проведены лабораторные испытания.

Лабораторные испытания показали, что фильтр работает в соответствие с моделью, разработанной с помощью Matlab.

ЗАКЛЮЧЕНИЕ

В результате рассмотрения алгоритмов адаптивной фильтрации было выявлено, что алгоритм RLS является наиболее подходящим, поскольку имеет «быструю» сходимость.

Была разработана программная модель реконфигурируемого фильтра эхо-компенсации. Было проведено его моделирование с использованием простых тригонометрических сигналов и сигналов акустического диапазона.

В результате моделирования были получены следующие результаты:

– основная адаптация сигнала происходит в первые 200 отсчетов;

– дальнейшая адаптация фильтра не вносит в значения весовых коэффициентов существенных изменений;

– адаптивный фильтр корректно работает на частотах акустического диапазона, дальнейшее увеличение частоты ведет к сбоям в работе фильтра.

При реализации лабораторного макета была выбрана аппаратная платформа. Определено, что ПЛИС является оптмальным вариантом для реализации лабораторного прототипа эхо-компенсатора вследствие возможности параллельной обработки данных, а также гибкости настройки схемы. Были рассмотрены особенности проектирования адаптивного фильтра эхо-компенсации. Была произведена модификация RLS алгоритма для реализации в арифметике с фиксированной точкой. Был разработан лабораторный макет эхо-компенсатора, а также проведены лабораторные испытания.

Лабораторные испытания показали, что лабораторный макет фильтра корректно работает, в соответствие с моделью, разработанной Matlab.

СПИСОК СОКРАЩЕНИЙ

АЦФ – адаптивный цифровой фильтр;

ВК – весовые коэффициенты;

ИС – интегральная схема;

МНК – метод наименьших квадратов;

ПЛИС – программируемая логическая интегральная схема;

РМНК – рекурсивный метод наименьших квадратов;

СБИС – сверхбольшая интегральная схема;

DSP – digital signal processing;

LMS – least mean squares;

RLS – recursive least squares.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Нгуен, Т. Ф. Обработка радиотехнических сигналов на фоне помех / Т. Ф. Нгуен. – Тамбов: Юком, 2018. – 76 с.

2 Кузнецов, Е. П. Методы и алгоритмы адаптивной эхо-компенсации: сравнительный анализ эффективности применения / Е. П. Кузнецов // Цифровая обработка сигналов. – 2007. – № 2. – С. 26–34.

3 Кузнецов, Е. П. Цифровая обработка сигналов в задачах эхо-компенсации: тематический обзор (часть 1) / Е. П. Кузнецов, В. В. Витязев // Цифровая обработка сигналов. – 2006. – № 3. – С. 8–19.

4 Джиган, В. И. История, теория и практика адаптивной обработки сигналов / В. И. Джиган // Проблемы разработки перспективных микро- и наноэлектронных систем. – 2012. – № 1. – С. 30–37.

5 Кузнецов, Е. П. Цифровая обработка сигналов в задачах эхо-компенсации: тематический обзор (часть 2) / Е. П. Кузнецов, В. В. Витязев // Цифровая обработка сигналов. – 2006. – № 4. – С. 20–28.

6 Джиган, В. И. Адаптивная фильтрация сигналов : теория и алгоритмы / В. И. Джиган. – Москва: ТЕХНОСФЕРА, 2013. – 528 с.

7 Адаптивная фильтрация цифровых данных [Электронный ресурс] : Национальная библиотека им. Н. Э. Баумана. – Режим доступа: https://ru.bmstu.wiki/Адаптивная_фильтрация_цифровых_данных (дата обращения: 26.12.2018).

8 Велигоша, А. В. Обоснование и выбор эффективного метода реализации адаптивых цифровых фильтров / А. В. Велигоша // Теория и техника радиосвязи. – 2012. – № 1. – С. 66–72.

9 Джиган, В. И. Прикладная библиотека адаптивных алгоритмов / В. И. Джиган // Электроника: Наука, Технология, Бизнес. – 2006. – № 1. – С. 60–65.

10 Малахов, О. И. МНК и РНК алгоритмы адаптивной фильтрации / О. И. Малахов, М. М. Трегубенко // Научный вестник Московского

государственного технического университета гражданской авиации. – 2006. – № 98. – С. 85–97.

11 Сергиенко, А. Б. Цифровая обработка сигналов : учебное пособие / А. Б. Сергиенко. – Изд. 3-е. – Санкт-Петербург: БХВ-Петербург, 2011. – 768 с.

12 Кондратьев, К. В. Методы и средства адаптивной компенсации акустической обратной связи при передаче речевой информации : дис. … канд. тенх. наук : 01.04.06 / Кирилл Валерьевич Кондратьев. – Таганрог, 2016. – 149 с.

13 Сергиенко, А. Б. Алгоритмы адаптивной фильтрации: Особенности реализации в Matlab / А. Б. Сергиенко // Математика в приложениях. – 2003. – № 1. – С. 18–28.

14 Haykin, S. Adaptive filter theory fifth edition / S. Haykin. – Изд. 5-е. – Prentice Hall, 2014. – 907 с.

15 Джиган, В. И. Быстрый алгоритм аффинных проекций : полная версия / В. И. Джиган // Проблемы разработки перспективных микро- и наноэлектронных систем. – 2016. – № 1. – С. 202–209.

16 Gonzalez, A. Affine projection algorithms : Evolution to smart and fast algorithms and applications / A. Gonzalez, M. Ferrer, F. Albu, M. de Diego // European Signal Processing Conference. – 2012. – № 20. – С. 1965–1969.

17 Джиган, В. И. Лестничные алгоритмы адаптивной фильтрации – новая составляющая прикладной библиотеки СБИС сигнальных контроллеров серии «Мультикор» / В. И. Джиган // Проблемы разработки перспективных микроэлектронных систем. – 2006. – С. 310–315.

18 Колкер, А. Б. Обоснование выбора программного обеспечения для робототехники / А. Б. Колкер, Д. А. Ливенец, А. И. Кошелева // Автоматика и программная инженерия. – 2012. – № 1. – С. 51–64.

19 Кочегурова, Е. А. Особенности системы MATLAB для решения задач вычислительной математики / Е. А. Кочегурова. – Томск: Издательство Томского государственного университета, 2013. – 110 с.

20 MATLAB [Электронный ресурс] : Официальный сайт компании Mathworks. – Режим доступа: https://www.mathworks.com/products/matlab.html (дата обращения: 01.05.2019).

21 Maple [Электронный ресурс] : Официальный сайт компании Waterloo Maple Inc. – Режим доступа: https://www.maplesoft.com/products/Maple/ (дата обращения: 01.05.2019).

22 MathCAD [Электронный ресурс] : Официальный сайт компании PTC. – Режим доступа: https://www.ptc.com/ru/products/mathcad (дата обращения: 01.05.2019).

23 Mathematica [Электронный ресурс] : Официальный сайт компании Wolfram Research. – Режим доступа: https://www.wolfram.com/mathematica/ (дата обращения: 01.05.2019).

24. Аладьев, В. З. Программирование в пакетах Maple и Mathematica : сравнительный аспект / В. З. Аладьев, В. К. Бойко, Е. А. Ровба. – Гродно: Монография, 2011. – 516 с.

25 Сравнительная таблица некоторых характеристик FPGA производителей Xilinx и Altera [Электронный ресурс] : Лаборатория параллельных информационных технологий научно-исследовательского вычислетельного центара Московского государственного университета. – Режим доступа: https://parallel.ru/fpga/chips.html (дата обращения: 25.04.2019).

26 Xilinx Products [Электронный ресурс] : Официальный сайт компании Xilinx. – Режим доступа: https://www.xilinx.com/products/silicon-devices/fpga.html (дата обращения: 02.05.2019).

27 Intel FPGA [Электронный ресурс] // Официальный сайт компании Intel. – Режим доступа: https://www.intel.ru/content/www/ru/ru/products/programmable/fpga.html (дата обращения: 02.05.2019).

28 Liang, J. Floating point unit generation and evaluation for FPGAs / J. Liang, R. Tessier, O. Mencer // Annual IEEE Symposium on Field-Programmable Custom Computing Machines. – 2003. – № 11. – С. 185–194.

29 Ifeachor, E. C. Digital Signal Processing / E. C. Ifeachor, B. W. Jervis. – Изд. 2-е. – Prentice Hall, 2002. – 960 с.

30 Petrone, J. Adaptive Filter Architectures for FPGA Implementation / J. Petrone. – The Florida State University, 2004. – 86 с.

31 Инструменты и ПО для проектирования [Электронный ресурс] : Официальный сайт компании Intel. – Режим доступа: https://www.intel.ru/content/www/ru/ru/software/programmable/overview.html (дата обращения: 02.05.2019).

32 Графический дизайн или текст Verilog/VHDL [Электронный ресурс] : МАРСОХОД Open Source Hardware Project. – Режим доступа: https://marsohod.org/11-blog/251-sch-or-txt (дата обращения: 01.05.2019).

ПРИЛОЖЕНИЕ А

Листинг программной модели реконфигурируемого адаптивного цифрового фильтра эхо-компенсации

```
classdef RLS_Filter < matlab.System
    % Адаптивный фильтр, обученный по RLS алгоритму

    properties (Nontunable, PositiveInteger)
        % Порядок фильтра
        Length = 24;
    end

    properties
        % Коэффициент забывания
        ForgetFactor = 1.0;
        % Начальный вектор коэффициентов
        InitialCoefficients = 0;
        % Обратная корреляционная матрица
        InitialInverseCorrelation = 1e3;
    end

    properties (Logical)
        % Разрешение на изменение коэффициентов
        LockCoefficients = true;
    end

    properties (DiscreteState)
        % Вектор-столбец коэффициентов фильтра
        Coefficients;
        % Образцовый сигнал
        Desired;
        % Входной сигнал
        Input;
```

```
    % Обратная корреляционная матрица
    InverseCorrelation;
    % Вектор-столбец коэффициентов усиления
    KalmanGain;
end

properties (Access=protected)
    % Данная структура содержит коэффициент забывания (ForgetFactor)
    % и разрешение на изменение коэффициентов (LockCoefficients)
    FilterParameters;
end

properties (Access=protected, Nontunable)
    % Тип входных данных
    InputDataType;
end

methods
    % Конструктор
    function obj = RLS_Filter(varargin)
        setProperties(obj, nargin, varargin{:});
    end
end

methods (Static, Hidden)
    function discreteStates = CoefficientUpdate(error, discreteStates, params)
        X = discreteStates.Input;
        P = discreteStates.InverseCorrelation;
        lam = params.ForgetFactor;

        % Обновление вектора коэффициентов усиления
        XP = X'*P;
        inv = 1/(lam + XP*X);
        K = inv*(P*X);
```

```
        % Обновление обратной корреляционной матрицы
        P = (1/lam)*(P - K*XP);

        % Сохранение результатов вычислений
        discreteStates.KalmanGain = K;
        discreteStates.InverseCorrelation = P;

        % Обновление вектора коэффициентов фильтра
        if params.LockCoefficients
            discreteStates.Coefficients = discreteStates.Coefficients + error*K';
        end
    end
end

methods (Access=protected)
    function setupImpl(obj, x)
        % Определение типа данных
        inputDataTypeLocal = class(x);
        obj.InputDataType = inputDataTypeLocal;
        obj.FilterParameters = getFilterParameterStruct(obj);

        L = obj.Length;
        initInvCorrLocal = obj.InitialInverseCorrelation;

        % Выделение памяти для обратной корреляционной матрицы
        obj.InverseCorrelation= eye(L, inputDataTypeLocal)*initInvCorrLocal;

        % Выделение памяти для вектора коэффициетов усиления
        obj.KalmanGain = zeros(L, 1, inputDataTypeLocal);

        % Выделение памяти для входного сигнала
        obj.Input = zeros(L, 1, inputDataTypeLocal);

        % Выделение памяти для образцового сигнала
        obj.Desired = zeros(L, 1, inputDataTypeLocal);
```

```
    % Выделение памяти для весовых коэффициентов
    obj.Coefficients = zeros(1, L);
end

function resetImpl(obj)
    L = obj.Length;
    initInvCorrLocal = obj.InitialInverseCorrelation;
    inputDataTypeLocal = obj.InputDataType;

    % Инициализация весовых коэффициентов
    coeffLocal = obj.Coefficients;
    coeffLocal(:) = cast(obj.InitialCoefficients, inputDataTypeLocal);
    obj.Coefficients = coeffLocal;

    % Инициализация обратной корреляционной матрицы
    obj.InverseCorrelation= eye(L, inputDataTypeLocal)*initInvCorrLocal;

    % Инициализация вектора коэффициетов усиления
    obj.KalmanGain = zeros(L, 1, inputDataTypeLocal);

    % Инициализация входного сигнала
    obj.Input = zeros(L, 1, inputDataTypeLocal);

    % Инициализация образцового сигнала
    obj.Desired = zeros(L, 1, inputDataTypeLocal);
end

function processTunedPropertiesImpl(obj)
    % Обновление коэффициента забывания и разрешения на
    % изменение коэффициентов при каждом их изменении
    obj.FilterParameters.ForgetFactor = ...
        cast(obj.ForgetFactor, obj.InputDataType);
    obj.FilterParameters.LockCoefficients = ...
        obj.LockCoefficients;
```

```
end

function [y, e, w] = stepImpl(obj, x, d)
    % Получение внутренних значений фильтра
    discreteStates = getDiscreteState(obj);

    discreteStates.Input = updateBuffer(obj, x, discreteStates.Input);
    % Вычисление выходного сигнала
    y = discreteStates.Coefficients*discreteStates.Input;
    % Вычисление ошибки
    e = d - y;
    w = zeros(1, obj.Length);
    % Обновление коэффициентов фильтра
    discreteStates=obj.CoefficientUpdate ...
        (e,discreteStates,obj.FilterParameters);
    w = discreteStates.Coefficients;
    % Сохранение вычисленных значений
    setDiscreteState(obj,discreteStates);
end

function filterParams = getFilterParameterStruct(obj)
    filterParams = struct('ForgetFactor', ...
        cast(obj.ForgetFactor,obj.InputDataType), ...
        'LockCoefficients', obj.LockCoefficients);
end

function buffer = updateBuffer(~, currentInput, buffer)
    buffer(:) = [currentInput(:); buffer(1:end-length(currentInput))];
end

function setDiscreteStateImpl(obj, discreteStates)
    obj.Input = discreteStates.Input;
    obj.Coefficients = discreteStates.Coefficients;
    obj.InverseCorrelation = discreteStates.InverseCorrelation;
    obj.KalmanGain = discreteStates.KalmanGain;
```

```
        end

        function discreteStates = getDiscreteStateImpl(obj)
            discreteStates = struct('Input', obj.Input, ...
                'Coefficients', obj.Coefficients, ...
                'InverseCorrelation', obj.InverseCorrelation, ...
                'KalmanGain', obj.KalmanGain);
        end
    end
end
```

Листинг разработанного ПО в полном объеме приведен на CD-диске.

ПРИЛОЖЕНИЕ Б

Листинг аппаратной модели реконфигурируемого адаптивного цифрового фильтра эхо-компенсации на языке Verilog

Файл RLS.v

```
module RLS #( parameter N=2, parameter M=1, parameter nBits=32, parameter B=2)(
        input clk,
        input [nBits-1:0]y,
        input reset,
        input newIt,
        input [N*nBits-1:0]x0,
        output encounter,write,ensyscoun,
        output [nBits-1:0]x,
        output [31:0] iterations,
        output load,final,
        input [N*nBits-1:0]wireA,wireKs
   );


        wire sha,shx0,shk,ens,enmult,s2,clear,clears;
        wire[nBits-1:0] wire_a,wire_x,wirey,out1,out2,outmult,ins;
        wire[nBits-1:0] wire_s,wire_r,wire_x0,wires2,s,wire_k,t1,t2,t3;
        wire [N*nBits-1:0] xout;


 FlipFlop #(.N(nBits))ffy (
   .clk(clk),
   .data(y),
   .enable(load),
   .reset(reset),
          .clear(1'b0),
   .out(wirey)
   );
```

```
        shifter #(.M(N*nBits),.nBits(nBits)) shiftera (
.clock(clk),
.reset(reset),
.load(load),
.shift(sha),
        .clear(1'b0),
        .clearoutput(clears),
.Data_in(wire_a),
.Data_load(wireA),
.Data_out(wire_a),
.r()
);

       Multiplexer2 #(.N(nBits)) mux1 (
.option(s1),
.mult0(wire_a),
.mult1(wire_r),
.out(out1)
);

        shifter #(.M(N*nBits),.nBits(nBits)) shifterx0 (
.clock(clk),
.reset(reset),
.load(loadx),
        .clear(clear),
.shift(shx0),
        .clearoutput(1'b0),
.Data_in(wire_x0),
.Data_load(xout),
.Data_out(wire_x0),
.r()
);
```

```
        shifter #(.M(N*nBits),.nBits(nBits)) shifterk (
.clock(clk),
.reset(reset),
.load(load),
.shift(shk),
        .clearoutput(clears),
        .clear(1'b0),
.Data_in(wire_k),
.Data_load(wireKs),
.Data_out(wire_k),
.r()
);


       Multiplexer2 #(.N(nBits)) mux2 (
.option(s2),
.mult0(wire_x0),
.mult1(wire_k),
.out(out2)
);

        Multiplication #(.nBits(nBits)) mult (
.clk(clk),
.a(out1),
.b(out2),
.enable(enmult),
.res(outmult)
);




        Addition #(.nBits(nBits)) add1 (
.clk(clk),
.a(outmult),
```

```
.b(s),
        .sum(1'b1),
.res(wires2),
.enable(enadder)
);



        FlipFlop #(.N(nBits))ffs (
.clk(clk),
.data(wires2),
.enable(ens),
.reset(reset),
        .clear(clears),
.out(s)
);


        Addition #(.nBits(nBits)) add2 (
.clk(clk),
.a(wirey),
.b(s),
        .sum(1'b0),
.res(wire_r),
.enable(1'b1)
);

        Addition #(.nBits(nBits)) add3 (
.clk(clk),
.a(outmult),
.b(t3),
        .sum(1'b1),
.res(wire_x),
.enable(1'b1)
);
```

```
        shifter #(.M(N*nBits),.nBits(nBits)) shifterx (
.clock(clk),
.reset(reset),
.load(1'b0),
.shift(shx),
        .clear(clear),
.Data_in(wire_x),
        .clearoutput(1'b0),
.Data_load(),
.Data_out(x),
.r(xout)
);


       Control #(.N(N),.M(M),.B(B)) control (
.clk(clk),
.reset(reset),
.newIt(newIt),
.load(load),
        .write(write),
        .iterations(iterations),
        .ensyscoun(ensyscoun),
        .clear(clear),
        .clears(clears),
.loadx(loadx),
.sha(sha),
.shx0(shx0),
        .final(final),
        .encounter(encounter),
.s1(s1),
.s2(s2),
.enmult(enmult),
.enadder(enadder),
```

```
    .ens(ens),
        .shk(shk),
    .shx(shx)
    );

        FlipFlop #(.N(nBits))ffT1 (
    .clk(clk),
    .data(wire_x0),
    .enable(1'b1),
    .reset(reset),
        .clear(1'b0),
    .out(t3)
    );

endmodule
```

Файл shifter.v

```
module shifter#(parameter M=4*32, parameter nBits=32)
    (
        input clock, reset, load, shift,
        input [0:nBits-1] Data_in,
        input [0:M-1] Data_load,
        input clearoutput,
        input clear,
    output reg [0:nBits-1] Data_out,
        output reg [M-1:0] r
    );




   always @ (negedge reset or posedge clock) begin
     if (reset == 0) begin
         r <= {M{1'b0}};
```

```
                    Data_out <= {nBits{1'b0}};
                    end
                else if (clear==1) begin
                    r <= {M{1'b0}};
                    Data_out <= {nBits{1'b0}};
                end
    else if (load == 1) begin
      r <= Data_load;
                    //Data_out <= Data_load[0:nBits-1];
                    end
    else if (shift == 1) begin
                    Data_out <= r[M-1:M-nBits];
                    r[M-1:nBits] <= r[M-nBits-1:0];
      r[nBits-1:0] <= Data_in;
      end
                end
  end
endmodule
```

Файл Control.v

```
module Control #(parameter N=16, parameter M=20, parameter B=1024)(
        input clk, reset, newIt,
        output reg load, loadx ,sha, shx0,s1,s2,
        output reg final,enmult,enadder,clears,ens,shx,shk,clear,encounter,write,ensyscoun,
        output reg [31:0] iterations);

         parameter Reset=4'd0;
         parameter NewSystem=4'd1;
         parameter NewIt=4'd2;
         parameter Calcs=4'd3;
         parameter WaitForMeasurement=4'd4;
         parameter Calcr=4'd5;
         parameter Calcx=4'd6;
         parameter Savex=4'd7;
```

```
parameter End=4'd8;
parameter CountSys=4'd9;

reg [3:0]state;
reg [31:0]counter;
reg [32:0] numBlocks;


always @ (posedge clk or negedge reset) begin
                if (reset==0) begin
                        state<=Reset;
                end else begin
                        case(state)
                          Reset: begin
                                        load<=0;
                                        loadx<=0;
                                        sha<=0;
                                        shx0<=0;
                                        clear<=0;
                                        final<=0;
                                        s1<=0;
                                        s2<=0;
                                        enmult<=0;
                                        enadder<=0;
                                        ens<=0;
                                        shx<=0;
                                        shk<=0;
                                        encounter<=0;
                                        write<=0;
                                        clears<=0;
                                        ensyscoun<=0;
                                        state <= NewSystem;
                                        numBlocks<=0;
                                        counter<={32{1'b0}};
                                end
```

```
NewSystem: begin
        clear<=1;
        state<=NewIt;
        iterations<={32{1'b0}};
end
NewIt: begin
        load <= 1;
        loadx<=1;
        clears<=1;
        state<=Calcs;

end
Calcs: begin
        sha<=1;
        shx0<=1;
        s1<=0;
        s2<=0;
        enmult<=1;
        enadder<=1;
        ens<=1;
        counter<=counter+1;
        if(counter==N) begin
                state<=WaitForMeasurement;
                sha<=0;
                counter<={32{1'd0}};
        end
end
WaitForMeasurement:begin
        if(newIt==1) begin
                state<=Calcr;
        end
end
Calcr: begin
        state<=Calcx;
        shx0<=1;
```

```
                encounter<=1;
                iterations<=iterations+1;
        end
        Calcx: begin
                s1<=1;
                s2<=1;
                shx<=1;
                shk<=1;
                shx0<=1;
                enmult<=1;
                counter<=counter+1;
                if(counter==N) begin
                        shk<=0;
                        if(iterations==M) begin
                                state<=CountSys;
                                counter<={32{1'd0}};
                        end else begin
                                state<=NewIt;
                                counter<={32{1'd0}};
                        end
                end
        end
CountSys: begin
        numBlocks<=numBlocks+1;
        state<=Savex;
end
Savex: begin
        write<=1;
        counter<=counter+1;
        shx<=1;
        if(counter==32'd0) begin
                write<=0;
        end
        if (counter==N) begin
                counter<=0;
```

```
                                    if (numBlocks==B) begin
                                        state<=End;
                                    end else begin
                                        state<=NewSystem;
                                    end
                                end
                            end
                            End: begin
                                state<=state;
                                final<=1;
                            end
                        endcase


                    end
                end
endmodule
```

Листинг разработанного ПО в полном объеме приведен на CD-диске.

Министерство науки и высшего образования РФ
Федеральное государственное автономное образовательное учреждение высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ О. В. Непомнящий
подпись инициалы, фамилия
«20» июня 2019 г.

БАКАЛАВАРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Реконфигурируемый, адаптивный, цифровой фильтр эхо-компенсации
тема

Пояснительная записка

Руководитель _____ 26.06.19 профессор, зав. каф. ВТ, канд. техн. наук О. В. Непомнящий
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ 26.06.19 Е. С. Бывшев
подпись, дата инициалы, фамилия

Нормоконтролер _____ 26.06.19 доцент, канд. техн. наук В. И. Иванов
подпись, дата должность, ученая степень инициалы, фамилия

Красноярск 2019