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

Институт космических и информационных технологий

Кафедра вычислительной техники

БАКАЛАВРСКАЯ РАБОТА

090301 Информатика и вычислительная техника

Разработка цифрового процессора эффектов для музыкальных инструментов

Руководитель _____ А.Г. Хантимиров
подпись, дата

Студент _____ _____ Я.Ф. Кашапов
номер группы, зачетной книжки подпись, дата

Красноярск 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Звук является одним из видов информации, которую человек способен воспринимать. Существует множество музыкальных инструментов, которые способны издавать звуки различного рода, отличающиеся по частоте, спектру, длительности и интенсивности. В зависимости от этих параметров, люди могут испытывать полный спектр эмоций.

В настоящее время есть несколько типов инструментов, которые позволяют получить звуковой сигнал в каком-либо виде: аналоговые и цифровые.

К аналоговым можно отнести все те музыкальные инструменты, которые используют аналоговые устройства для извлечения звука. Например, звук электрогитары снимается с никелированных струн звукоснимателем, который в общем является катушкой индуктивности.

Цифровыми являются те инструменты, звук в которых генерируется цифровыми электронными схемами. Например, цифровой синтезатор.

В современном «цифровом» мире использование цифровых методов обработки звуков имеет очевидное преимущество — цифровой звук значительно легче и дешевле обрабатывать. На сегодняшний день для этого существует большое количество специализированных микросхем и аудио-процессоров, таких как цифровой сигнальный процессор (ЦСП, или DSP – digital signal processor) и т. п. Появление цифровой обработки сигналов в аудио-приложениях постоянно было обусловлено возможностью разнообразить тональности, экспериментируя усилителями и эффектами.

Обычно DSP-устройства являются сложными и комбинируют в себе как аналоговую, так и цифровую логику. Например, они могу содержать один или несколько аналого-цифровых и цифроаналоговых преобразователей, а также аналоговые и цифровые фильтры частот. Важным критерием качества работы цифровых аудио-процессоров является задержка обработанного сигнала. Чем

меньше эта задержка, тем более пригодным является устройство для использования его в реальном времени, например на музыкальном концерте.

Звуковые эффекты, накладываемые на звук, позволяют разнообразить звучание инструмента, добавив дополнительную музыкальную информацию. Например, большая часть музыкального жанра «Рок», который зародился в 60-х вследствие применения аналоговых схем искажения звукового сигнала, невозможна без таких эффектов как «overdrive» и «distorsion». Данные эффекты выделяют такую музыку, и за счёт них образовался сам жанр, а также появились новые техники игры.

Эффекты можно рассматривать как функцию, которая применяется к исходному входному звуковому сигналу. Поэтому эффекты бывают линейные и нелинейные.

На сегодняшний день существует множество коммерческих решений, которые позволяют настраивать звуковые эффекты для любого аналогового звука. Например, решения от компании HeadRush — цифровые процессоры эффектов для электрогитар. Они отличаются высоким качеством, большим количеством настроек и дороговизной. Так же существуют аналоговые «педальки» для электрогитар, которые, как правило, реализуют один конкретный эффект и не являются дорогими. Однако многие музыканты не обходятся только одним эффектом и комбинируют такие устройства, что в итоге ведёт к «удорожанию» звука.

Поэтому существует необходимость разработать устройство, которое будет обладать высокими возможностями по конфигурированию звука, а также сравнительно невысокой ценой.

Для качественного подхода к реализации такого устройства есть смысл использовать программируемую логическую интегральную схему (ПЛИС), так как именно данный способ позволяет обеспечить наименьшие задержки звукового сигнала при его обработке за счёт особенностей ПЛИС.

В этом проекте демонстрируется альтернативный подход к звуковым эффектам с использованием ПЛИС вместо ядер DSP или аналоговых

компонентов для выполнения цифровой обработки сигналов, предназначенных для изменения аудиосигналов. Звуковая обработка была разработана на отладочной плате с системой на кристалле Xilinx Zynq®-7000, совмещающей в себе ПЛИС, а также два ядра ARM Cortex A9. Различные гитарные эффекты были получены из пользовательских сложно-функциональных блоков (СФ-блоков), написанных на Verilog. Программируемая логика (ПЛ) управляется поворотными потенциометрами и кнопкам, с которыми взаимодействует пользователь, а индикация базируется на светодиодах.

1 Анализ предметной области

1.1 Обзор существующих решений

Многие устройства, реализующие цифровые звуковые эффекты, сегодня являются очень сложными и больше напоминают компьютеры. Они состоят из высококоплиментарных DSP-ядер, специально предназначенных для обработки звука, за счёт чего и возможно создание линейных и нелинейных эффектов.

В общем виде можно выделить несколько типов цифровых эффектов с точки зрения их реализации:

– программно-реализуемые эффекты;

– самостоятельные устройства, реализующие цифровые эффекты;

– эффекты, которые идут в составе усилителей звука.

Отдельно стоит выделить классические способы обработки звука — различные блоки эффектов (педали эффектов).

Рассмотрим данные решения более подробно.

1.1.1 Программно-реализуемые эффекты

Для получения таких эффектов достаточно использовать персональный компьютер (ПК). Например, программное обеспечение «Guitar Rig» (Рисунок 1) позволяет подключить сотни разных эффектов к любому источнику звука в ПК.

Рисунок 1 – Интерфейс программы GuitarRig 5

Также популярны программные средства, называемые звуковыми плагинами, которые можно активировать в программах мастеринга звука. Один плагин может реализовывать один или несколько эффектов. Например, плагин «Emissary Ignite» (Рисунок 2). Он содержит в себе два режима звучания («Clean» и «Lead») и отдельные эквалайзеры для каждого из них. Плагины можно комбинировать, выстраивая необходимую цепочку эффектов для получения желаемого звучания.

Рисунок 2 – Звуковой плагин «Emissary Ignite» в режиме Lead Fat

В случае использования программных решений, музыкальный инструмент подключается напрямую к компьютеру либо через внешний аудио-интерфейс, а аудио-обработка осуществляется полностью на центральном процессоре (ЦП). Подобные программы высоко утилизируют процессор компьютера и в своём интерфейсе отображают загрузку ЦП. При таком подходе входной звуковой сигнал может обрабатываться со значительными задержками, вплоть до невозможности использовать подобные решения в реальном времени при игре на инструменте.

Таким образом, плюсы данного подхода:

– много реализованных эффектов;

– нужен только компьютер и звуковая карта.

Минусы:

– заметные задержки при обработке звука;

– неудобно использовать на концертах;

– для обеспечения высокого качества звука необходима дорогостоящая звуковая карта.

1.1.2 Самостоятельные устройства, реализующие цифровые эффекты

Как правило, такие устройства содержат DSP-ядро, аналого-цифровые преобразователи (АЦП), цифро-аналоговые преобразователи (ЦАП) и интерфейс пользователя (кнопки, дисплей и т.п.). По способу ввода и вывода звука могут сильно отличаться. Например, на входе и выходе может быть аналоговый сигнал. Или на входе аналоговый, а на выходе — цифровой, передающийся по шине I2S или USB. Представителем такого типа устройств является HeadRush Pedalboard (Рисунок 3) [1]. Задержка обработанного сигнала сравнительно низкая.

Рисунок 3 – Моделирующий гитарный процессор эффектов HeadRush Pelalboard

Другим примером является ZOOM G1X Four (Рисунок 4). Такой процессор эффектов построен на основе 32-разрядного DSP ZOOM ZFX-3, в составе которого содержится 24-битный АЦП и ЦАП с частотой дискретизации 96 кГц.

Рисунок 4 – Гитарный процессор эффектов ZOOM G1X Four

Плюсы подобных устройств:

– много реализованных эффектов;

– устройство готово для использования на концерте;

– низкие задержки обработанного сигнала.

Минусы:

– высокая стоимость.

1.1.3 Цифровые эффекты, идущие в составе усилителей звука

Некоторые модели комбо-усилителей звука содержат в себе встроенный блок процессора эффектов. Например, Yamaha THR (Рисунок 5). Такие устройства обладают меньшими возможностями по настройке эффектов по сравнению полноценными процессорами эффектов. Входной сигнал аналоговый, передающийся через инструментальный разъём. Задержка обработанного сигнала сравнительно низкая, так как данный вид устройств предназначен для использования в реальном времени.

Рисунок 5 – Гитарный комбоуслитель Yamaha THR с моделирующим блоком настройки эффектов

Плюсы:

– устройство «два в одном»;

– устройство готово для использования на концерте;

– низкие задержки обработанного сигнала;

Минусы:

– высокая стоимость;

– малое количество эффектов.

1.1.4 Блоки эффектов

Блоки эффектов являются классическим вариантом наложения эффектов на звук. Как правило, они реализуют один эффект или несколько одного типа. Чаще всего эти устройства являются аналоговыми. Имеют инструментальные вход и выход.

Рисунок 6 – Комбирированный блок эффектов «Overdrive/Distortion» BOSS OS-2

Плюсы:

– малые габариты;

– устройство готово для использования на концерте;

– низкие задержки обработанного сигнала;

Минусы:

– могут быть дорогими;

– реализуют один эффект;

– цепочка блоков эффектов обладает большими габаритами.

1.2 Обзор элементной базы, используемой в проекте

1.2.1 Отладочная плата EBAZ4205

Отладочная плата EBAZ4205 (Рисунок 7) построена на основе системы на кристалле (СНК) Xilinx ZYNQ 7010. Дополнительные возможности платы [2]:

– 256 Мб DDR3 RAM;

– 128 Мб NAND Flash;

– розетка RJ-45 и Ethernet-контроллер IP101GA;

– 52 GPIO, выведенных на разъёмы DATA1, DATA2, DATA3;

– слот для SD-карт;

– JTAG-разъём для отладки ПЛИС.

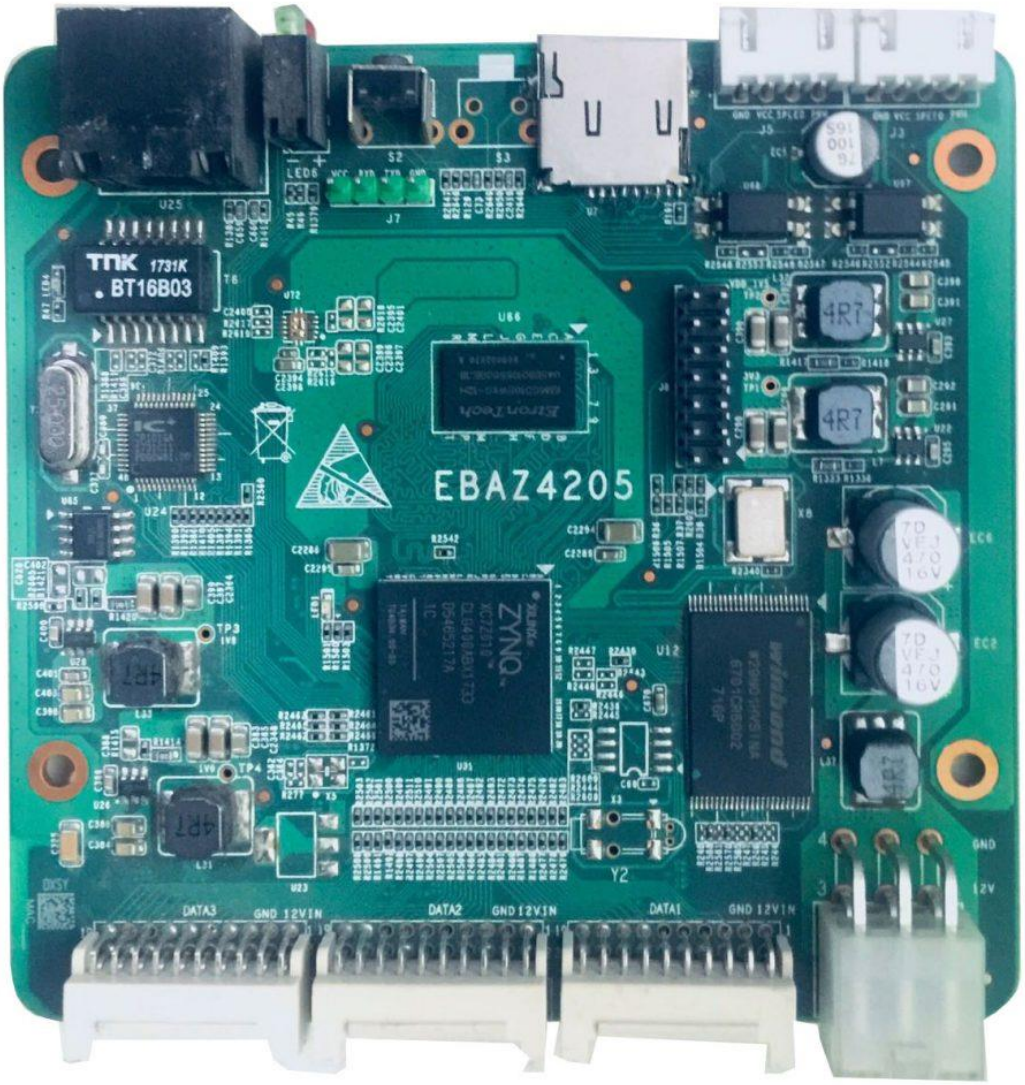

Рисунок 7 – Отладочная плата EBAZ4205

Как уже было отмечено, в основе отладочной платы лежит Xilinx ZYNQ 7010. Это система на кристалле, которая помимо программируемой логики (ПЛ) содержит ещё процессорную систему (ПС) с двумя ядрами ARM Cortex A9. Общая структурная схема ZYNQ 7010 (Рисунок 8) [3] включает множество дополнительных СФ-блоков и соединений типа «процессорная система – программируемая логика». Основные характеристики ПЛ:

– 28 тыс. программируемых логических ячеек;

– 17,6 тыс. таблиц преобразования;

– 2,1 Мб ОЗУ.

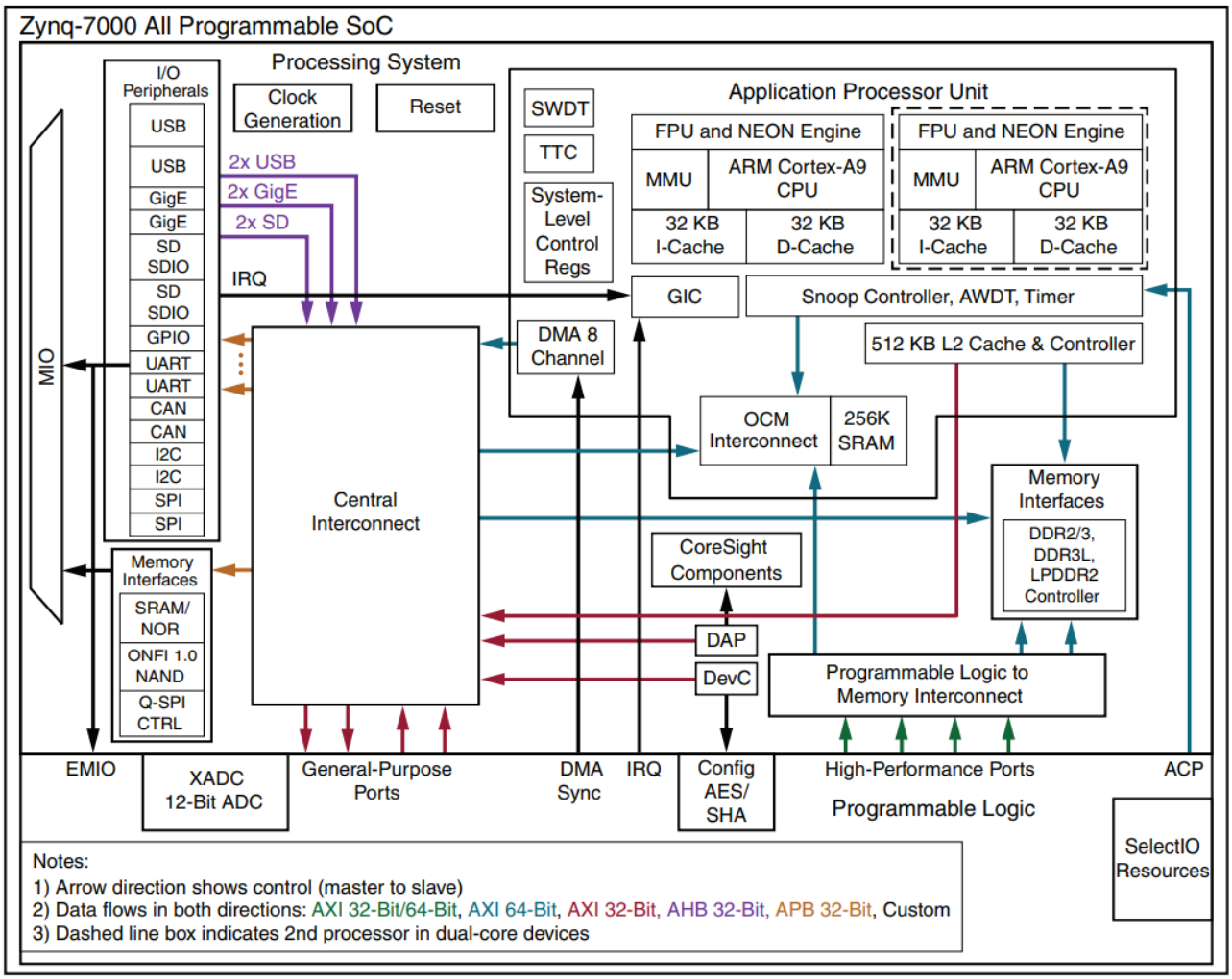

Рисунок 8 – Общая структурная схема ZYNQ 7010

В данном проекте процессорная часть использоваться будет только лишь для отладки и загрузки результата синтеза Verilog-конфигурации в ПЛИС. Так как на процессорной части ZYNQ 7010 запущено Linux-ядро, то через devfs доступно устройство /dev/xdevbit, в которое можно загружать синтезированные проекты (bitstream).

1.2.2 Аналого-цифровой преобразователь PCM1808

АЦП PCM1808 (Рисунок 9) специально спроектирован для обработки звукового сигнала. Типовые примеры применения данного преобразователя [4]:

– Цифровой TV;

– DVD-рекордер.

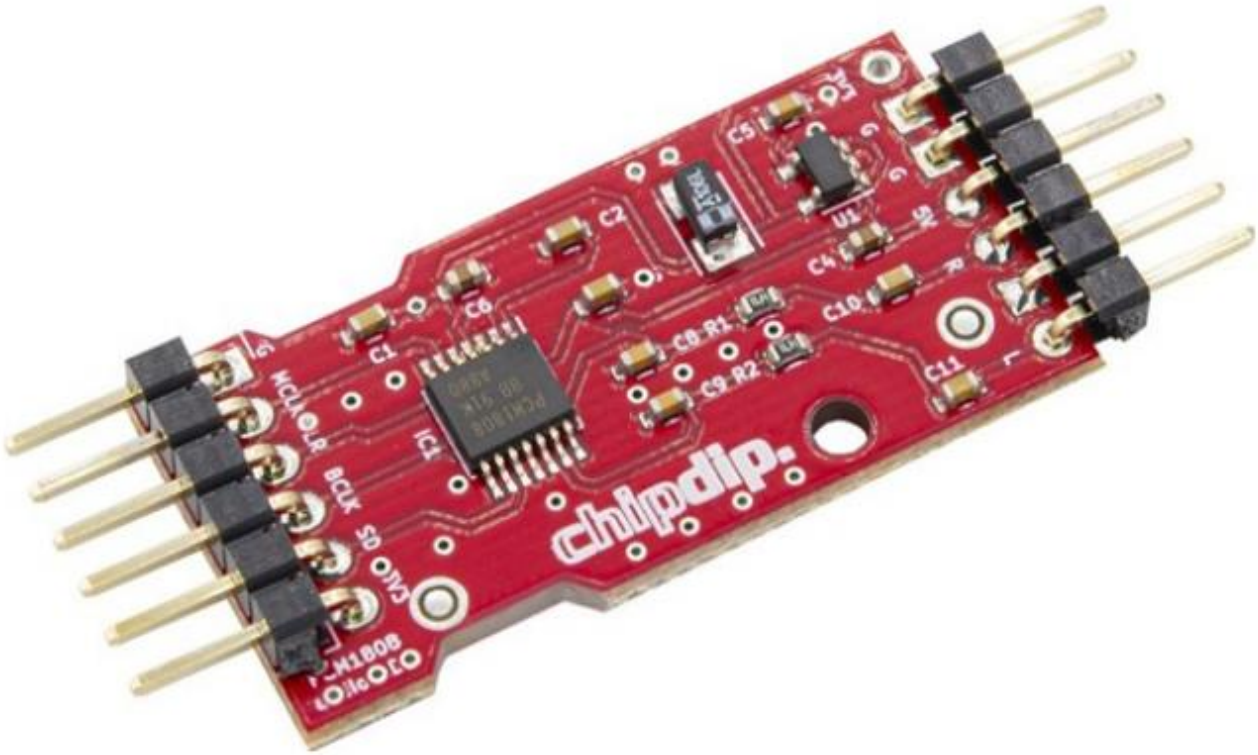

Рисунок 9 – Отладочная плата Chipdip PCM1808

Основные характеристики АЦП:

– разрядность 24 бит;

– настраиваемая частота дискретизации аналогового сигнала до 96 кГц.

– оцифрованный сигнал в формате I2S или Left-justified;

– возможность работы в разных режимах: «master» или «slave».

1.2.3 Цифро-аналоговый преобразователь PCM5102a

ЦАП PCM5102a (Рисунок 10) по своему классу не отличается от АЦП PCM1808. Они могут применяться в одних и тех же проектах. Основные характеристики ЦАП [5]:

– настраиваемая разрядность 16, 24 или 32 бит;

– настраиваемая частота дискретизации аналогового сигнала до 384 кГц.

– оцифрованный сигнал в формате I2S или Left-justified;

– возможность работы в разных режимах: «master» или «slave».

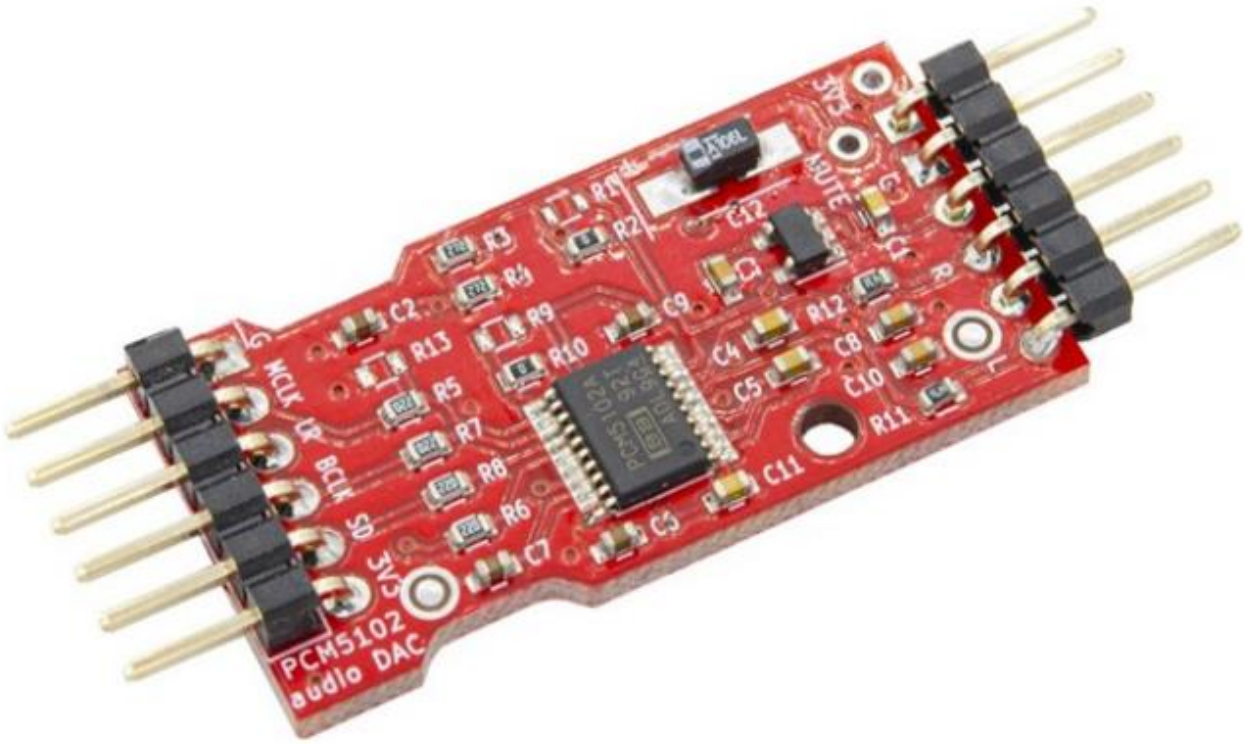

Рисунок 10 – Отладочная плата Chipdip PCM5102a

1.3 Обзор стандарта I2S

Показанные выше АЦП и ЦАП передают и принимают оцифрованный звук в формате I2S. Это специализированный стандарт передачи аудио между цифровыми устройствами [6].

Шина I2S состоит из трёх проводников (Рисунок 11):

– continuous serial clock (SCK, в этом проекте – BCK);

– word select (WS, в этом проекте – LRCK);

– serial data (SD, в этом проекте – DOUT).

Сигнал BCK предназначен для синхронизации передачи данных между устройствами. WS определяет, для какого аудиоканала (левого или правого) передаются данные по последовательной шине SD. Если WS равен логической единице, то данные на SD соответствуют правому каналу, иначе – левому.

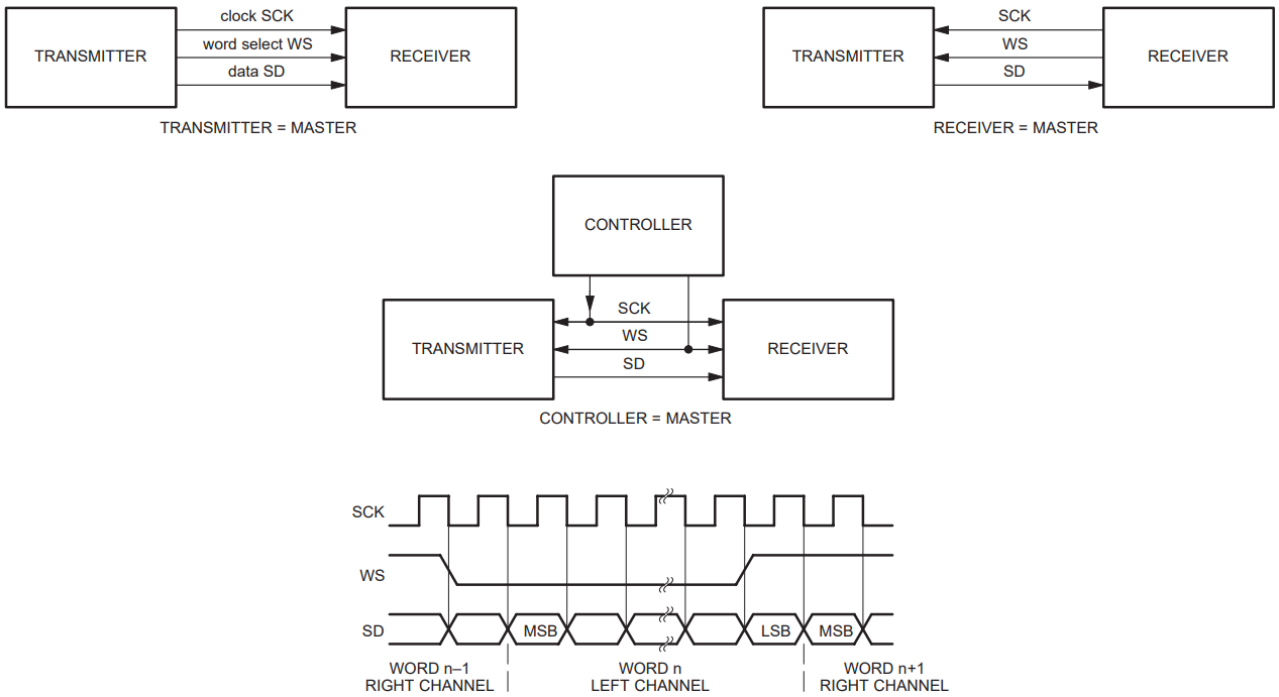

Рисунок 11 – Типовые схемы соединения устройств и передачи данных I2S

Как можно видеть, от того, в каком режиме работает устройство, зависит кто генерирует тактовые сигналы передачи данных. В данном проекте между I2S-передатчиком и I2S-приёмником находится ПЛИС.

1.4 Требования к проектируемому устройству

Проектируемый процессор эффектов должен реализовывать следующие эффекты:

– задержка;

– перегрузка;

– искажения;

– тремоло.

Для выбора эффектов следует организовать интерфейс пользователя на основе кнопок, светодиодных индикаторов, потенциометров и энкодеров.

Verilog-проект должен быть разделён на модули. Взаимодействие с АЦП, ЦАП и эффекты, а также обработку ввода пользователем следует реализовать в виде СФ-блоков.

1.5 Вывод

Были рассмотрены различные варианты устройств, реализующих цифровые звуковые эффекты. Каждый из вариантов имеет свои плюсы и минусы. Однако самым главным критерием для устройства, реализующего эффекты, является задержка обработанного сигнала. Поэтому, для качественного подхода к реализации такого устройства есть смысл использовать ПЛИС, так как именно данный способ позволяет обеспечить наименьшие задержки звукового сигнала при его обработке за счёт возможности параллельно обрабатывать данные.

Также рассмотрена элементная база проектируемого устройства. Высокоточные АЦП и ЦАП позволят преобразовать звуковой сигнал с наименьшими потерями.

Таким образом, задача сводится к реализации набора СФ-блоков, реализующих звуковые эффекты и взаимодействующих с ЦАП, АЦП и т.д.

2 Проектирование и разработка алгоритмов звуковых эффектов

2.1 Эффект «Усиление»

Эффект усиления функционирует за счёт увеличения амплитуды входного сигнала.

$$f(x) = x * G \tag{2.1}$$

где G – коэффициент усиления входного сигнала.

Ниже представлена общая блок-диаграмма, описывающая эффект усиления входного звукового сигнала (Рисунок 12).

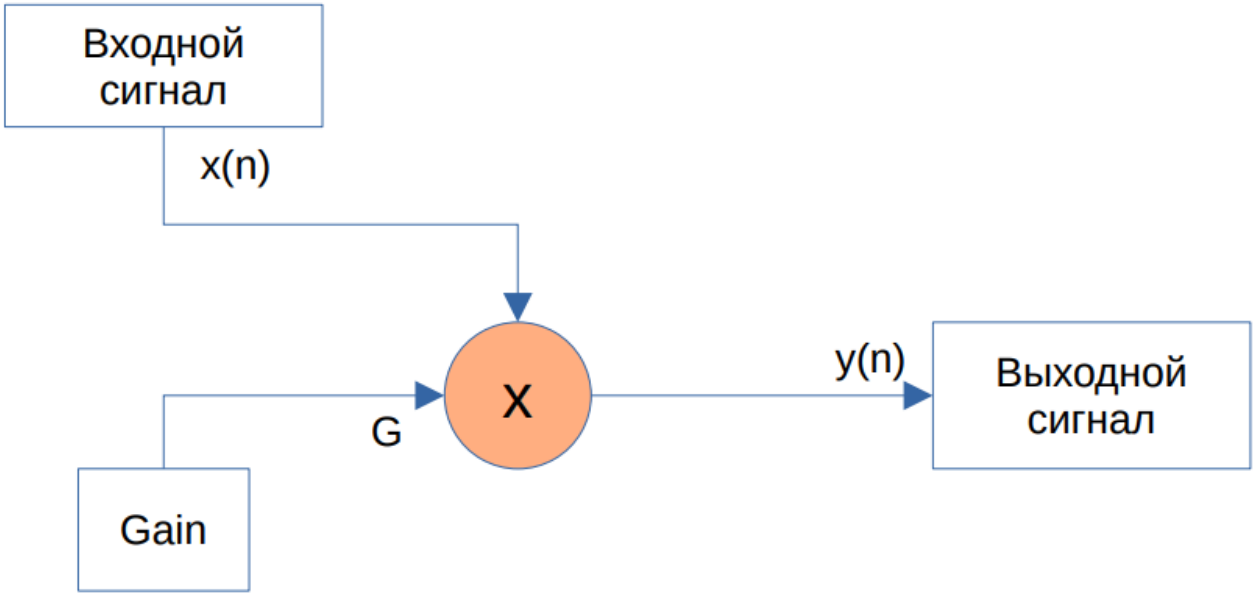

Рисунок 12 – Блок-диаграмма эффекта усиления

Входные параметры, необходимые для работы эффекта:

– входной сигнал;

– коэффициент усиления.

Встроенные функции Matlab, где была построена модель данного эффекта, позволяют считывать музыкальные файлы в формате Waveform Audio File

(WAV), а также проигрывать считанную и обработанную музыку. Для примера был взят отрывок ритм-гитарной партии музыкальной композиции «*Metallica – Sad But True»*, и к нему был применён эффект усиления с коэффициентом равным пяти (Рисунок 13). Частота дискретизации составила 44100 Гц.

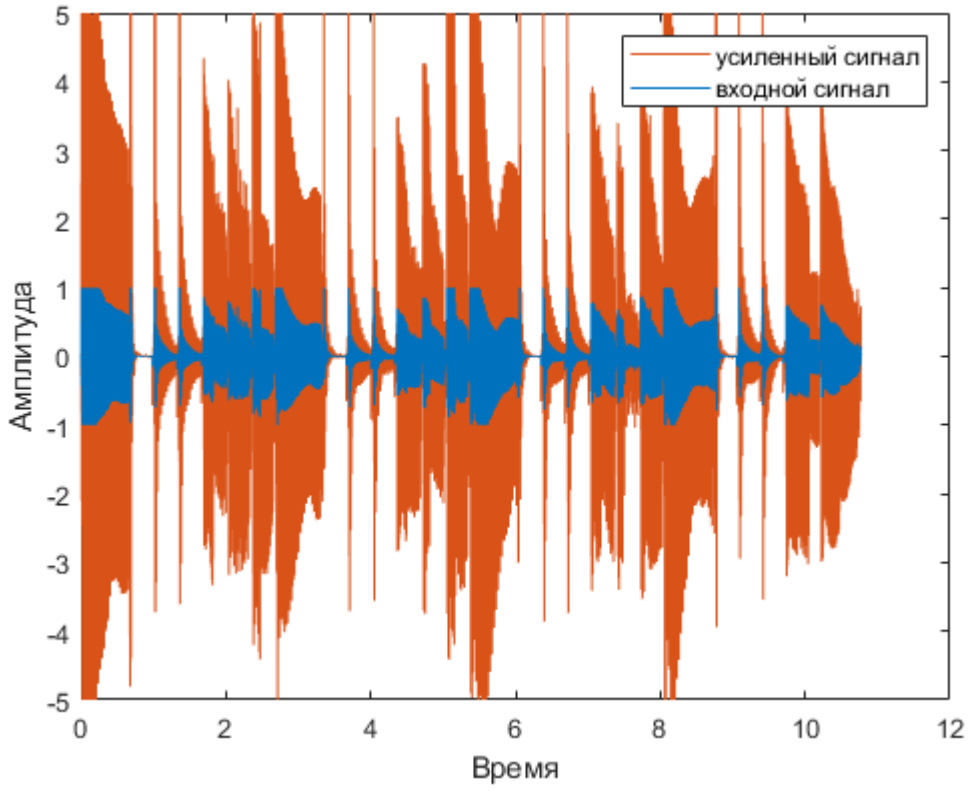

Рисунок 13 – Применение эффекта «Усиление» к музыкальной композиции

2.2 Эффект «Перегруз»

Эффект перегрузки (Overdrive) появился в 60-х вместе с популяризацией электрических гитар. Функционально данный эффект искажает входной сигнал путём ограничения его по амплитуде. Существуют два основных способа ограничения: жесткое (hard-clipping) и мягкое (soft-clipping). Рисунок 14 показывает сравнение этих методов.

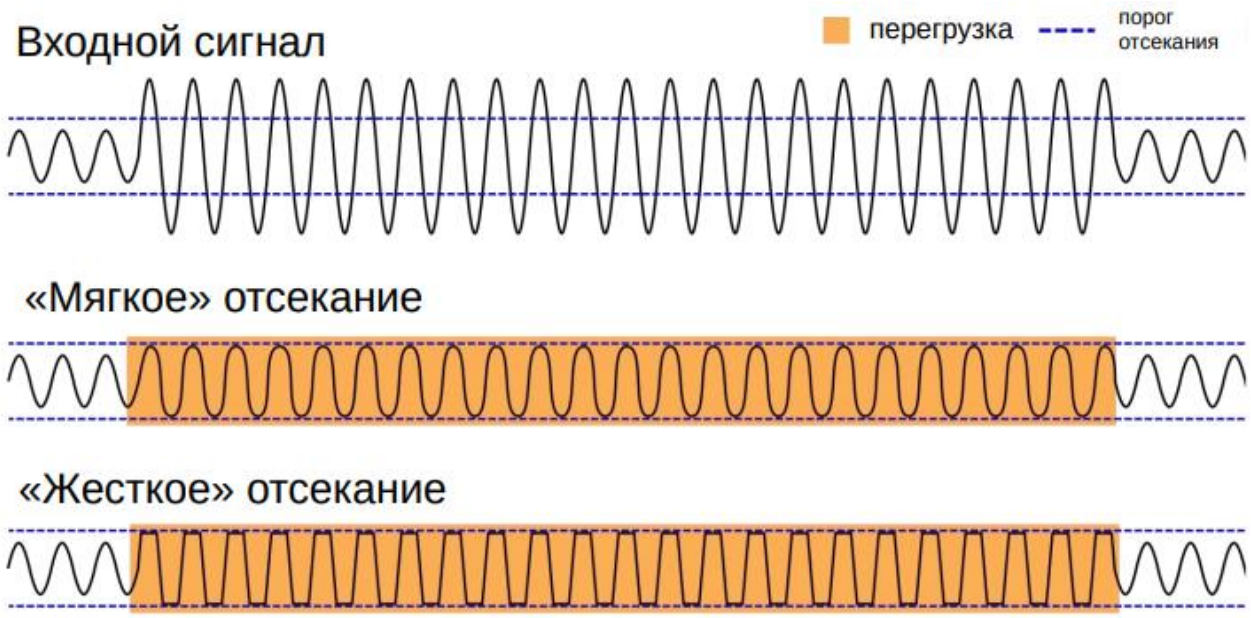

Рисунок 14 – Способы реализации эффекта «Перегруз»

Как можно видеть при «мягком» ограничении входного сигнала, функция сглаживает его значения у порога отсекания. Это усложняет реализацию, но позволяет сделать перегруженный звук более естественным по сравнению с «жестким» ограничением. Ниже представлена общая блок-диаграмма, описывающая эффект перегрузки (Рисунок 15).

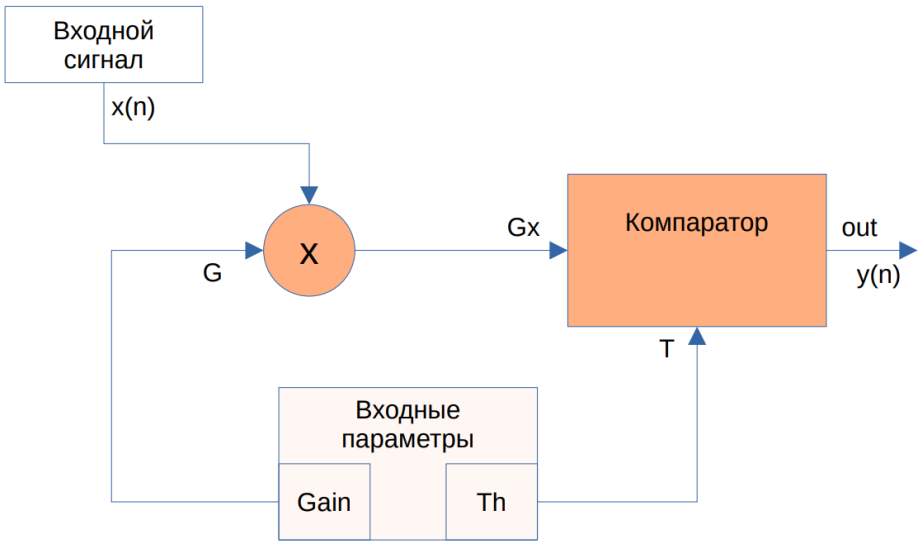

Рисунок 15 – Блок-диаграмма эффекта перегрузки

Для функционирования эффекта необходимы следующие входные параметры:

– входной сигнал;

– значение коэффициента усиления звука (Gain);

– значение порога срабатывания отсекания (Threshold).

После подачи входного аудиосигнала, он умножается на коэффициент усиления, а затем блок-компаратор сравнивает получившийся усиленный сигнал и пороговое допустимое значение. В случае превышения порога, выходом компаратора будет само пороговое значение, иначе – усиленный сигнал. Математически метод «жесткого» ограничения это можно записать следующим образом [7]:

$$f(x) = \begin{cases} threshold, & x > threshold \\ x, & x \leq threshold \end{cases} \quad (2.2)$$

где *threshold* – порог отсекания сигнала.

Метод «мягкого» ограничения описывается [7]:

$$f(x) = \begin{cases} 2x, & 0 \leq x < \frac{1}{3} \\ \frac{3-(2-3x)^2}{3}, & \frac{1}{3} \leq x < \frac{2}{3} \\ 1, & \frac{2}{3} \leq x \leq 1 \end{cases} \quad (2.3)$$

На основании формул 2.2 и 2.3 была построена модель в Matlab с двумя режимами: hard-clipping или soft-clipping. Для наглядности результат работы модели показан на синусоидальном входном сигнале (Рисунок 16 и Рисунок 17).

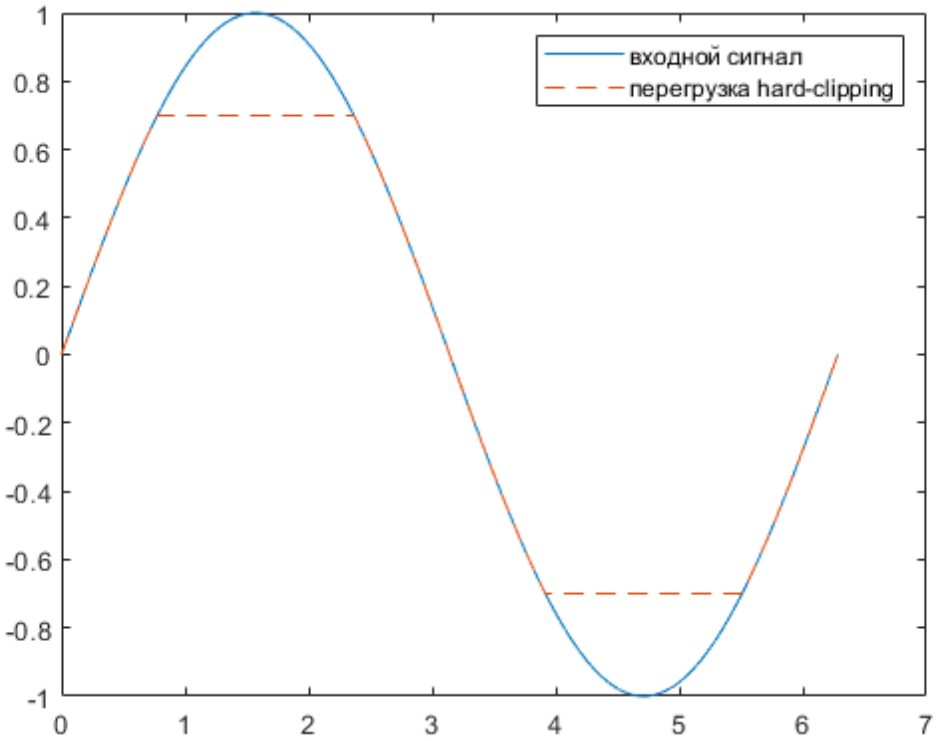

Рисунок 16 – Применение функции «Перегруз» к входному сигналу в Matlab в режиме «жесткого» отсекания

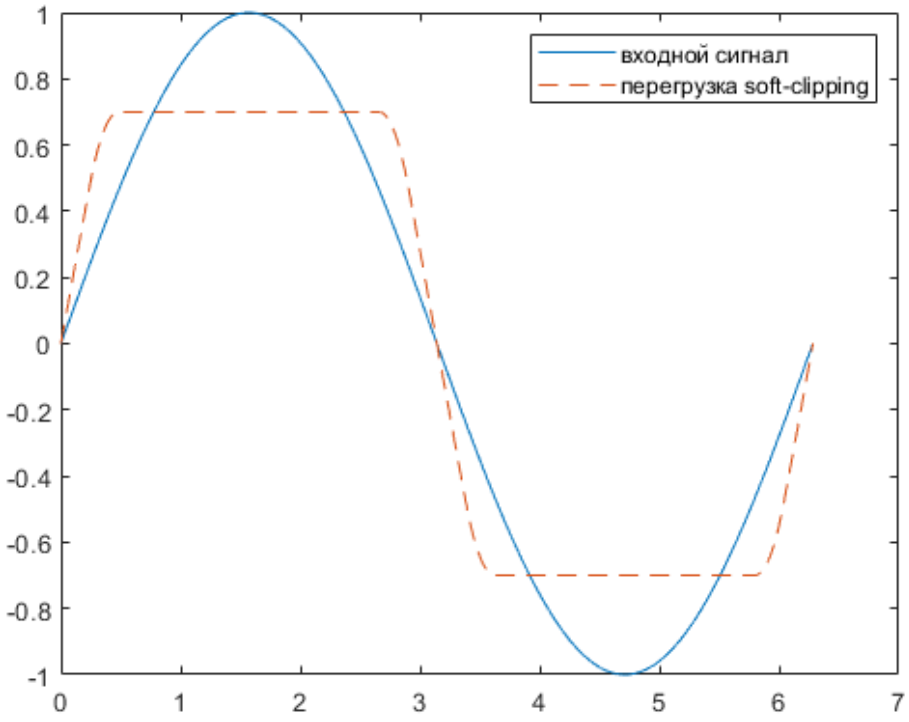

Рисунок 17 – Применение функции «Перегруз» к входному сигналу в Matlab в режиме «мягкого» отсекания

Полученная модель позволяет:

– выбрать режим (soft- или hard-clipping);

– усилить входной сигнал;

– выбрать уровень отсекания сигнала (threshold).

Результат моделирования эффекта в Matlab показан ниже (Рисунок 18).

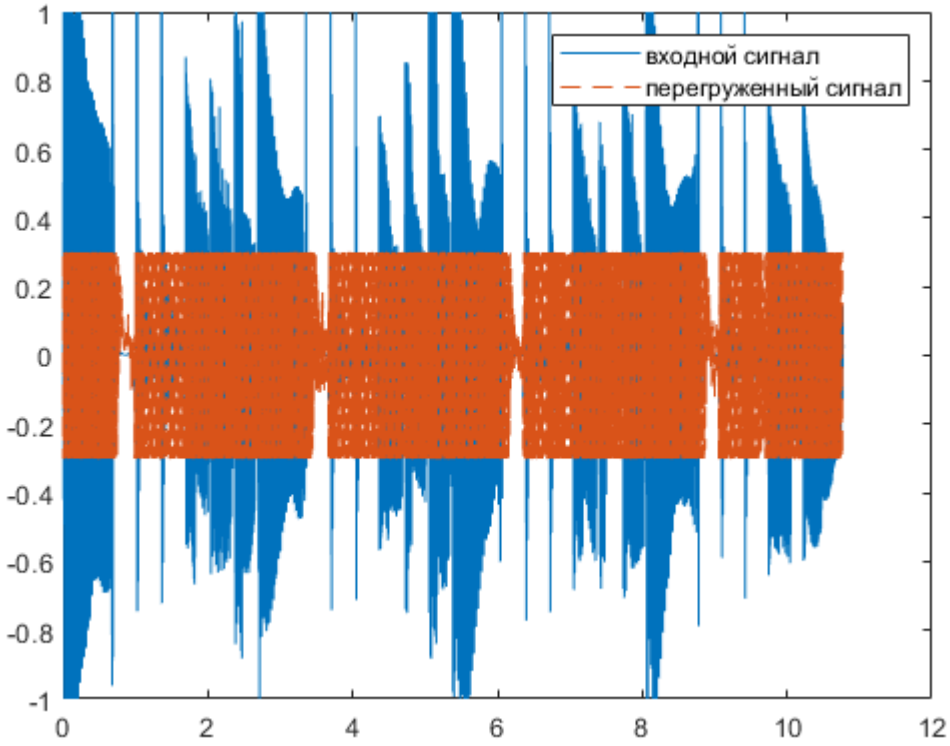

Рисунок 18 – Применение эффекта «Overdrive» к музыкальной композиции

2.3 Эффект «Задержка»

Данный эффект основан на том, что к входному сигналу добавляется его точная копию, но с задержкой на какое-то определённое время. Существует два основных типа реализации повторения сигнала: *infinite impulse response* (IIR) и *finite impulse response* (FIR).

IIR – свойство, которое позволяет системе повторять сигнал бесконечное количество раз. FIR – свойство, которое позволяет системе повторять сигнал конечное количество раз (2.4). В данном проекте реализуется именно этот вариант, а количество повторений равно одному.

$$y[n] = \begin{cases} x[n] + G * x[n - M], FIR \\ x[n] + G * y[n - M], IIR \end{cases} \quad (2.4)$$

где G – коэффициент усиления повторяющегося сигнала;

M – задержка, измеряемая в отсчётах.

Блок-диаграмма, описывающая эффект задержки показана ниже (Рисунок 19).

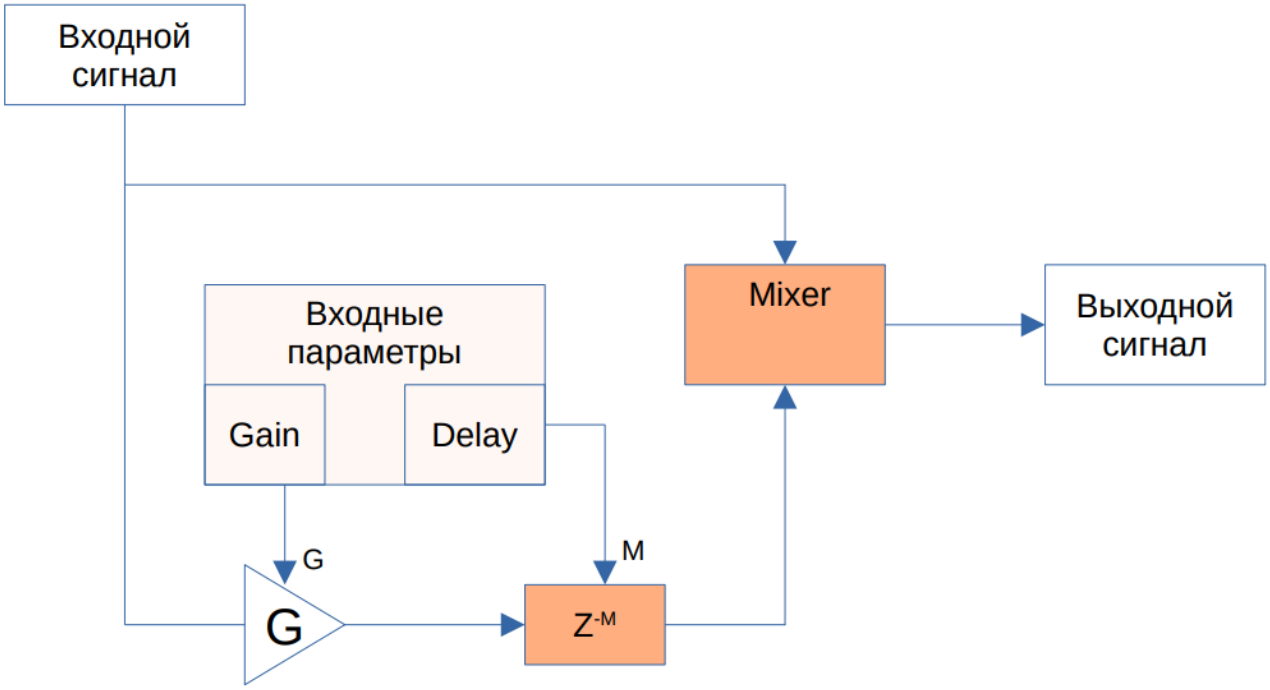

Рисунок 19 – Блок-диаграмма эффекта задержки

Цифровой эффект «Задержка» является зависимым от частоты дискретизации (частоты семплирования) звука. Данный параметр описывает, сколько замеров значений сигнала выполнено в секунду. Стандартное значение для студийной музыкальной записи – 44100 отсчётов в секунду. Расчёт количества отсчётов за определённый промежуток времени в миллисекундах вычисляется следующим образом:

$$delay_{samples} = \left\lfloor \frac{delay_{ms} * F_s}{1000} \right\rfloor \quad (2.5)$$

где $delay_{ms}$ – задержка в миллисекундах;

F_s – частота дискретизации сигнала в Гц;

Построенная модель в Matlab (Рисунок 20) показывает поведение системы при применении данного эффекта к синусоидальному сигналу с задержкой, равной 200 мс.

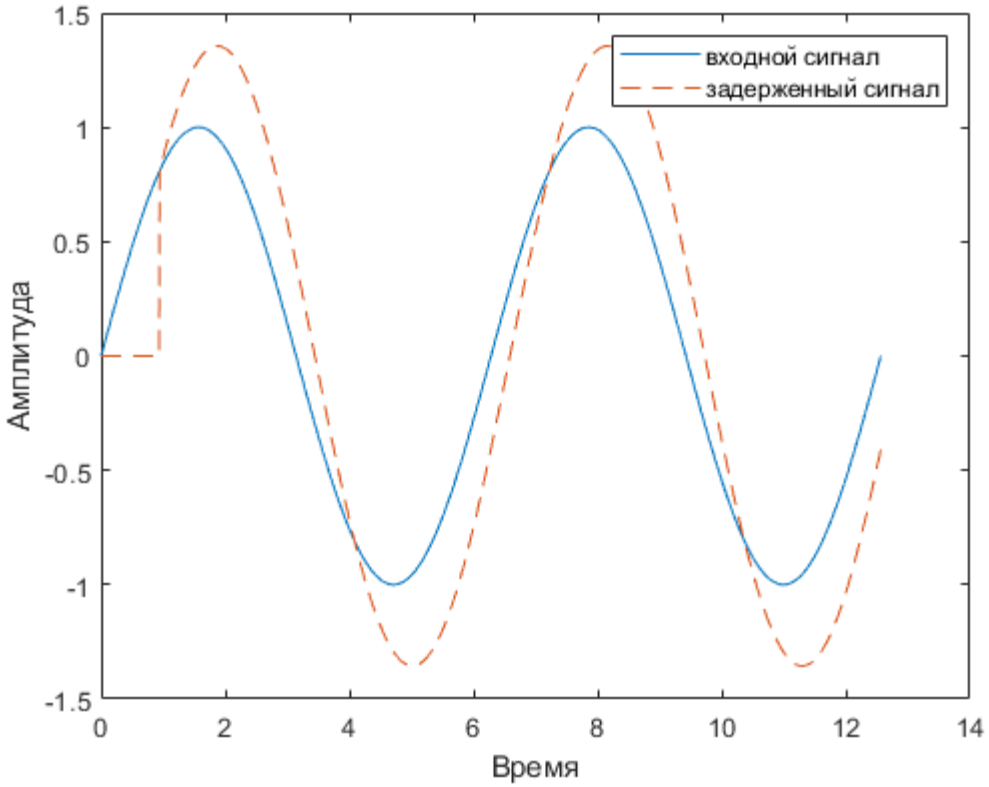

Рисунок 20 – Применение эффекта «Задержка» к синусоидальному сигналу

Видно, что дополнительный сигнал смещён вперёд во времени, иными словами, отстаёт относительно входного сигнала и при этом повторяет его.

2.4 Эффект «Тремоло»

Эффект «Тремоло» основан на амплитудной модуляции сигналов. Модуляцией называется процесс изменения одного или нескольких параметров

высокочастотного несущего колебания по закону низкочастотного информационного сигнала [11]. Схема модуляции сигнала показана ниже (Рисунок 21).

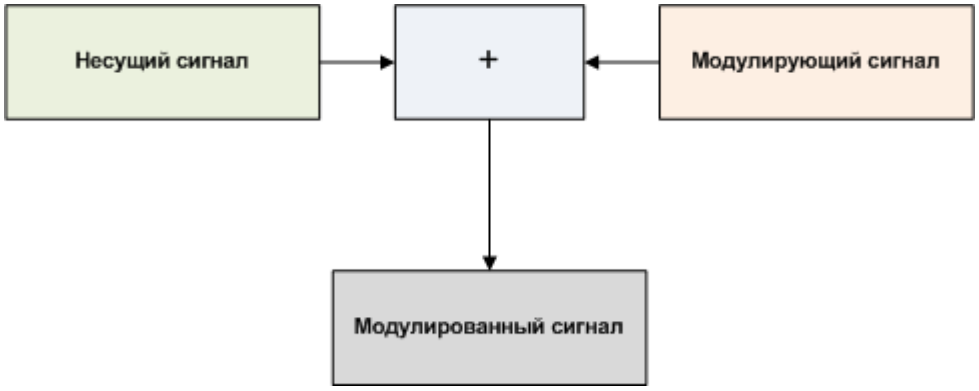

Рисунок 21 – Принцип формирования модулированного сигнала

В случае эффекта «Тремоло» модулирующим сигналом выступает звуковой сигнал, а несущим – сигнал какой-то формы с частотой в диапазоне, как правило, от 0 до 20 Гц. Модулированным сигналом будет выступать сигнал с эффектом. Функция амплитудной модуляции для звукового сигнала:

$$f(t) = input * (1 + depth * \sin(2\pi \frac{F_c}{F_s} t)) \quad (2.6)$$

$$f(t) = input * (1 + depth * \cos(2\pi \frac{F_c}{F_s} t)) \quad (2.7)$$

где *input* – модулирующий сигнал;

depth – глубина затухания модулированного сигнала;

F_c – частота несущего сигнала;

F_s – частота дискретизации моделирующего сигнала.

Однако, при реализации эффекта «Тремоло» в ПЛИС гораздо эффективнее использовать более простую функцию несущего сигнала. Такой функцией может выступать меандр, так как сигнал такой формы легко реализовать в цифровых устройствах и на его реализацию будет потрачено меньше вычислительных

ресурсов. Также несущим сигналом может являться треугольный сигнал. В таком случае моделирующая функция принимает следующий вид:

$$f(t) = input * (1 + depth * triangle(2\pi \frac{F_c}{F_s} t)) \quad (2.8)$$

где *triangle()* – функция генерации треугольной несущей.

Данный эффект был смоделирован в Matlab в разных режимах:

– функция несущего сигнала синусоидальная (depth=1) (Рисунок 22);

– функция несущего сигнала треугольная (depth=0.8) (Рисунок 23);

– функция несущего сигнала пилообразная (depth=0.8) (Рисунок 24).

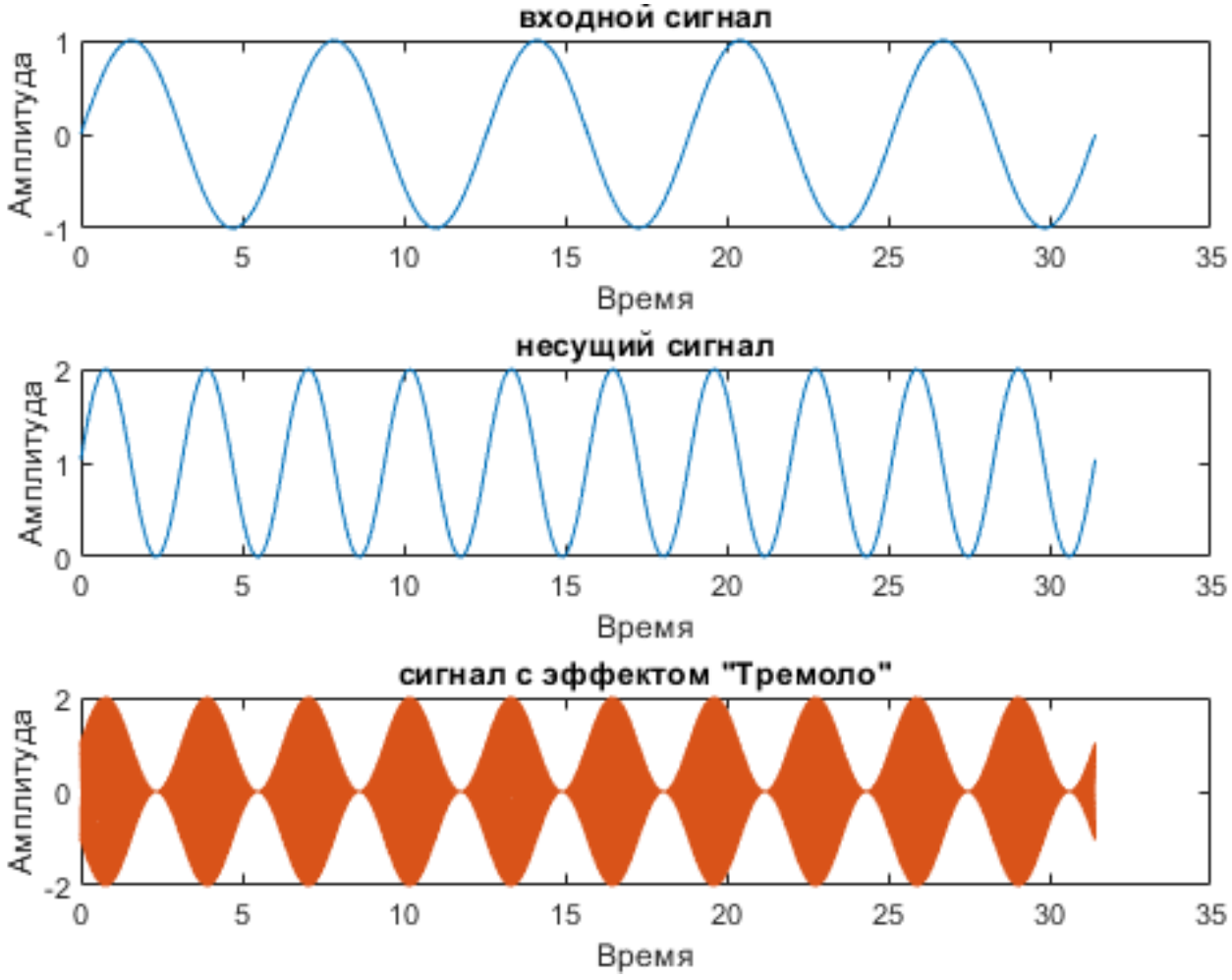

Рисунок 22 – Применение эффекта «Тремоло» с синусоидальной несущей

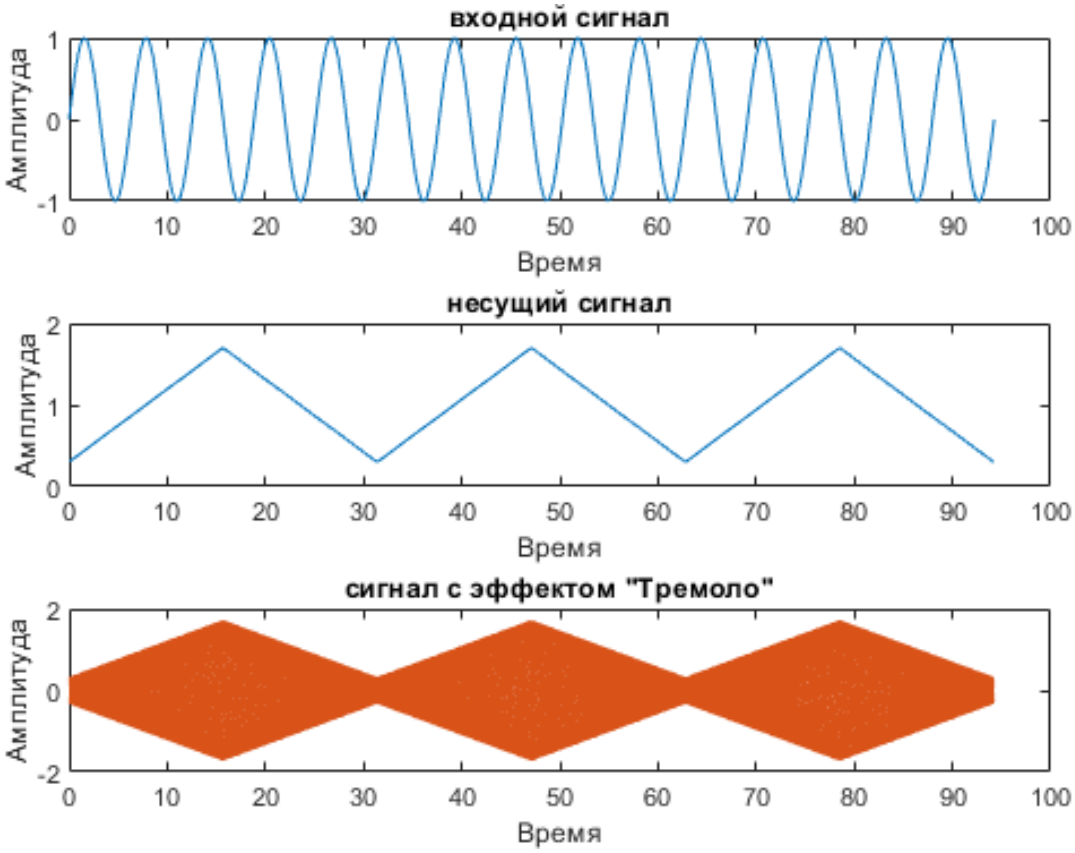

Рисунок 23 – Применение эффекта «Тремоло» с треугольной несущей

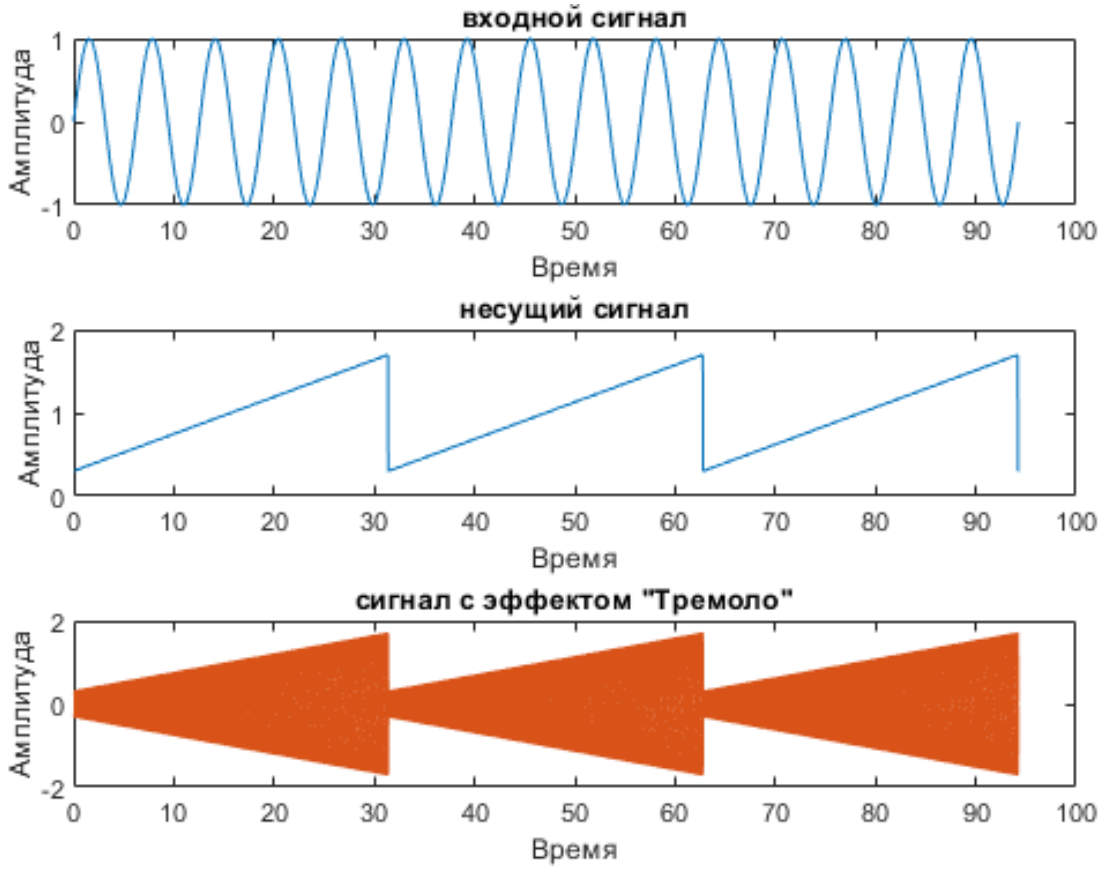

Рисунок 24 – Применение эффекта «Тремоло» с треугольной несущей

Модель с треугольной формой несущего сигнала была применена к звуковой записи (Рисунок 25).

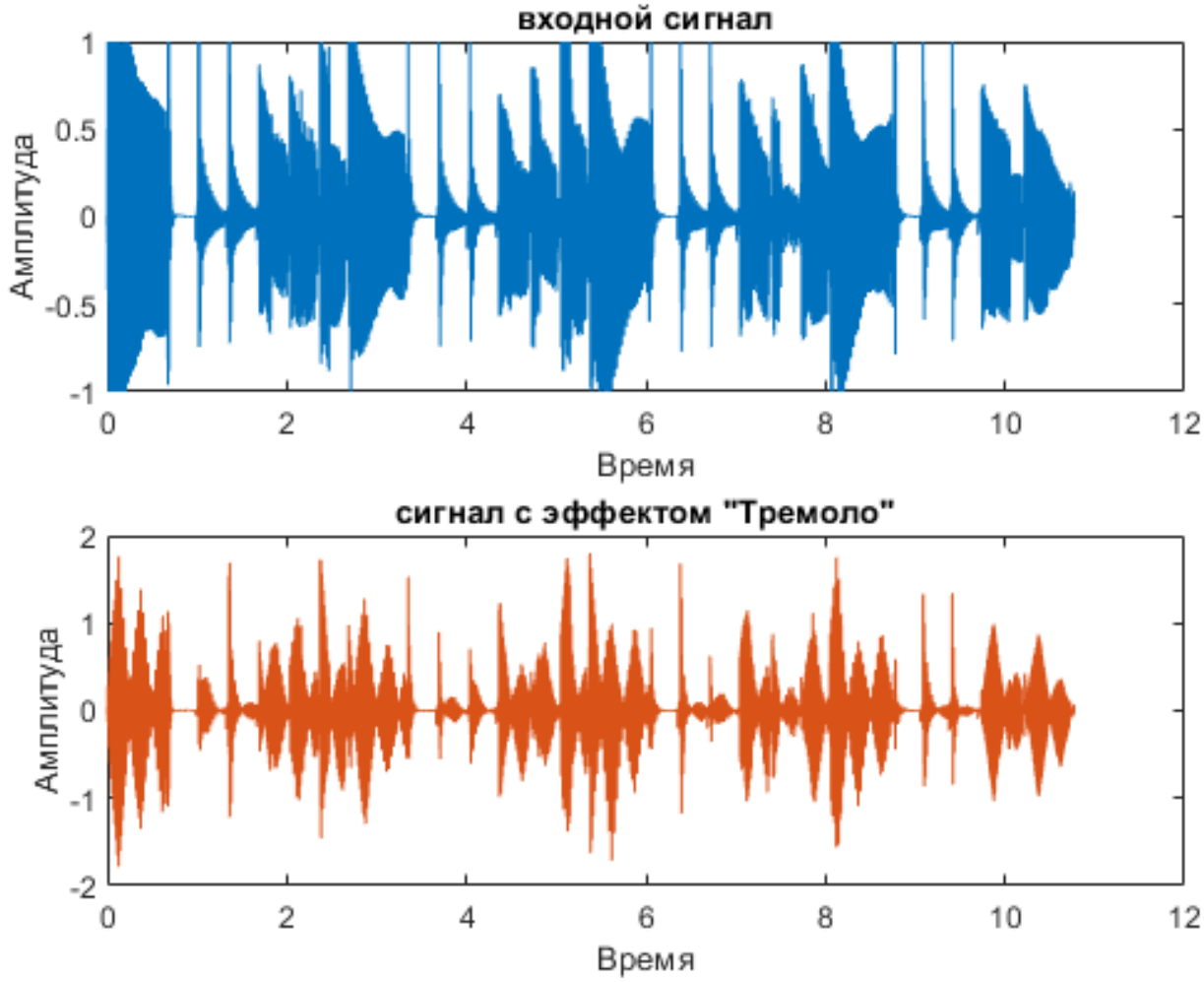

Рисунок 25 – Применение эффекта «Тремоло» к музыкальной композиции

2.5 Эффект «Искажение»

Данный эффект по своей функциональности похож на «Перегруз», однако искажает сигнал в большей степени. Существует множество вариаций реализации данного эффекта [7] (2.9, 2.10, 2.11), однако все они являются сложнореализуемыми в ПЛИС.

$$y = \frac{x}{1+|x|} \tag{2.9}$$

$$y = \begin{cases} -\frac{2}{3}, & x \leq -1 \\ x + \frac{x^3}{3}, & -1 < x < 1 \\ \frac{2}{3}, & x \geq 1 \end{cases} \tag{2.10}$$

$$y = \arctan(x) \tag{2.11}$$

Наиболее оптимальный способ создания цифрового эффекта «искажения» с точки зрения реализации в ПЛИС является метод повторного квантования сигнала. Суть его заключается в том, чтобы уменьшить точность квантования оцифрованного звука.

2.6 Вывод

В этой главе были рассмотрены реализации эффектов разрабатываемого процессора эффектов. Они также были смоделированы в Matlab, что позволило понять то, как они работают. При реализации в ПЛИС стоить учитывать следующие ограничения:

– следует отказаться от вычислений чисел с плавающей точкой;

– следует отказаться от деления чисел.

3 Разработка архитектуры и алгоритмов функционирования процессора эффектов

3.1 Конфигурация аппаратуры

АЦП PCM1808 и ЦАП PCM5102a являются конфигурируемыми: они могут работать в разных режимах [4][5]. Модули Chipdip, которые используются в проекте, заранее определяют эти режимы в своей электрической схеме:

– PCM1808 настроен в режиме ведущего, а PCM5102a - ведомого;

– протокол обмена оцифрованного звука I2S;

В режиме «ведомый» PCM5102a управляется внешним устройством. Это означает, что ЦАП не генерирует тактовые импульсы LRCK, BCK и SCKI. Их должно генерировать устройство «ведущий». В случае этого проекта, эти устройством является Xilinx ZYNQ 7010. Для PCM1808 предоставляется внешний тактовый импульс SCKI.

3.2 Разработка архитектуры процессора эффектов

Рисунок 26 отражает общую архитектуру разрабатываемого процессора эффектов.

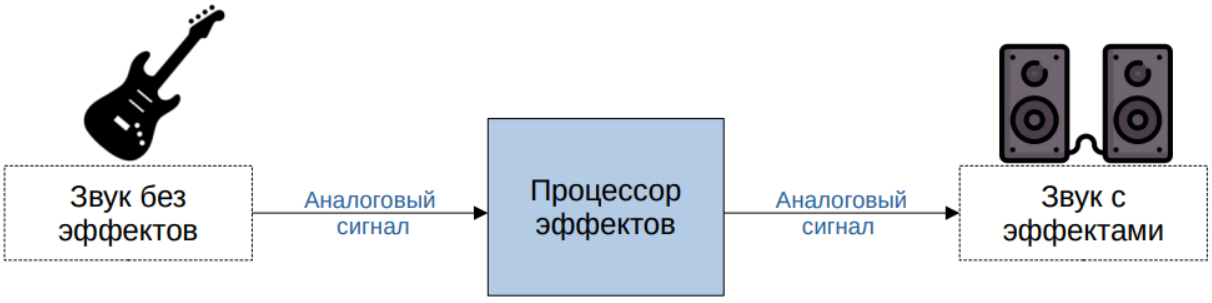

Рисунок 26 – Функциональная схема работы процессора эффектов

Процессор эффектов принимает на вход звуковой сигнал в аналоговом

виде и выдаёт на выход также аналоговый сигнал через инструментальный разъём. Все необходимые преобразования форм сигнала производятся внутри разрабатываемого устройства. Более подробная схема (Рисунок 27) описывает эти преобразования.

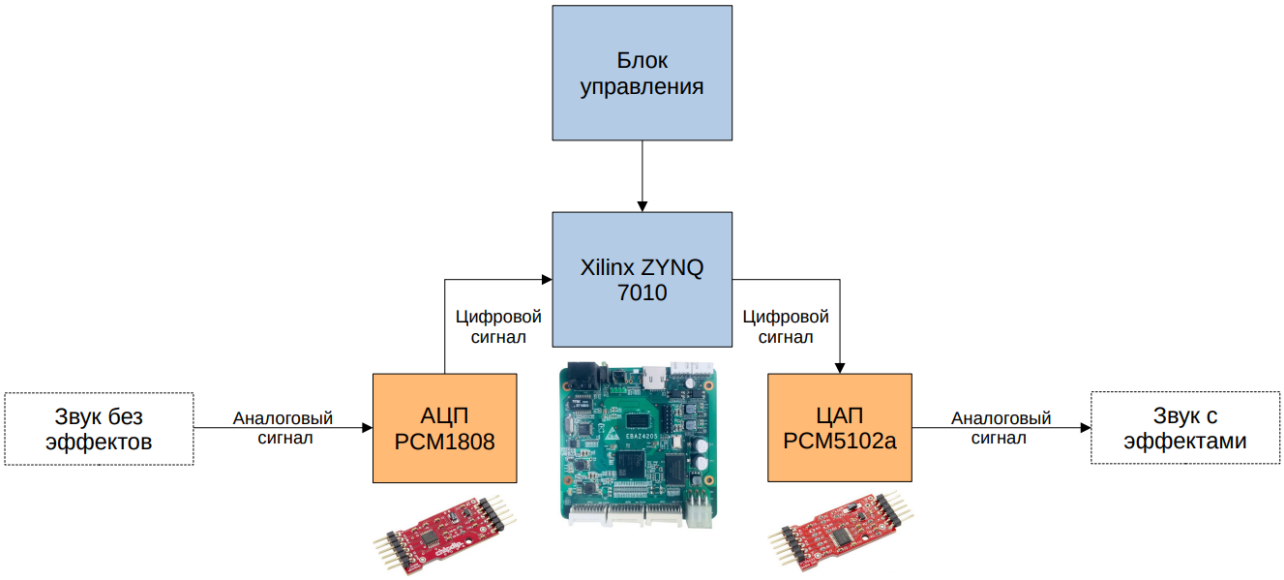

Рисунок 27 – Компонентная схема разрабатываемого устройства

Показанная выше схема описывает, как компоненты процессора эффектов взаимодействуют между собой. Аналогово-цифровой преобразователь PCM1808 принимает входной звуковой аналоговый сигнал, преобразует его в цифровой вид и передаёт по шине I2S на ПЛИС Xilinx ZYNQ 7010, которая реализует цифровые звуковые эффекты. Полученный обработанный сигнал поступает по I2S шине в цифро-аналоговый преобразователь PCM5102a, который преобразует сигнал в конечный аналоговый формат. Компонентная электрическая схема (Рисунок 28) описывает какими сигналами соединены составляющие процессора эффектов.

На этой схеме система Xilinx ZYNQ 7010 разделена на две части: процессорная система (ПС) Xilinx ZYNQ 7010 генерирует тактовые импульсы для программируемой логики (CLK0). АЦП, ПЛ и ЦАП соединяются рядом линий тактовых импульсов:

– LRCK – линия выбора канала звукового сигнала (левый/правый);

– SCKI – линия системных тактовых сигналов для PCM1808 и PCM5102a;

– BCK – линия тактовых сигналов передачи данных для PCM1808 и PCM5102a.

На линиях «I2S DATA» выставляются цифровые аудиоданные, а по шине «Mode» передаётся код текущего режима работы процессора эффектов.

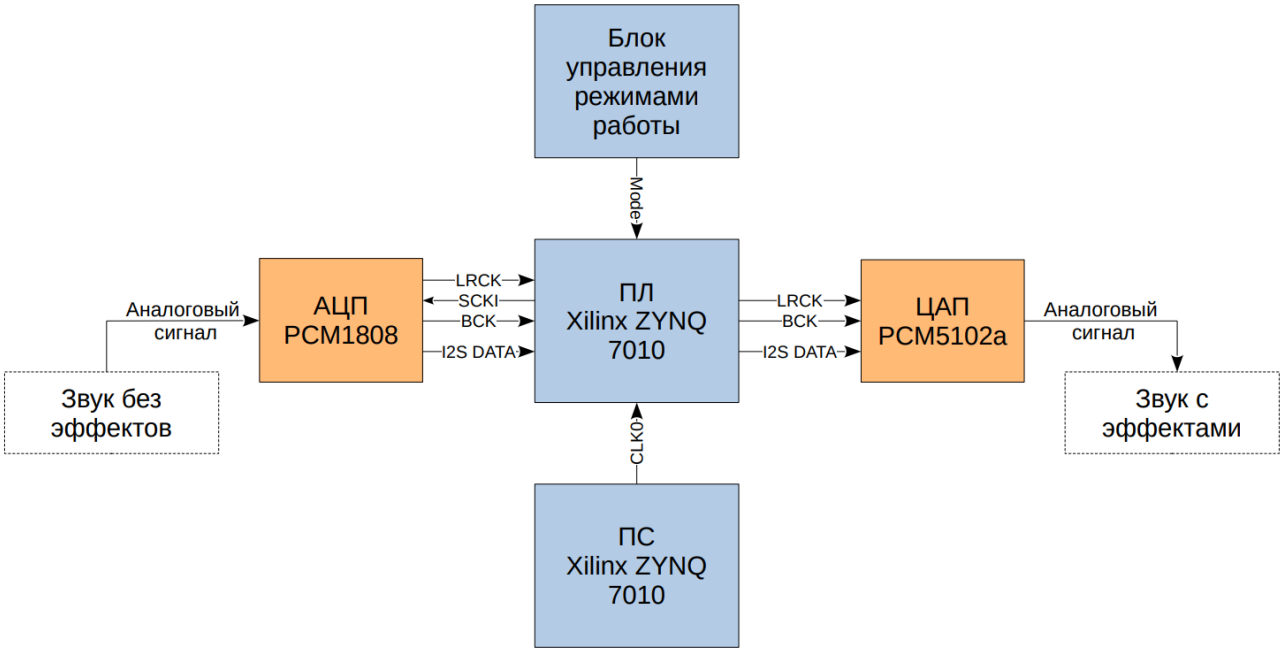

Рисунок 28 – Компонентная электрическая схема процессора эффектов

Далее архитектуру проекта стоит рассмотреть с точки зрения её устройства в виде СФ-блоков.

3.3 Разработка алгоритмов функционирования процессора эффектов

Сложно-функциональные блоки представляют из себя переиспользуемые модули ПЛИС-проектов. Они позволяют выстроить иерархию проекта за счёт использования модулей и сократить количество конфигурации, что в итоге ведёт к снижению количество потенциальных ошибок в коде.

Например, СФ-блок (Рисунок 29), которое реализует эффект «Distortion», можно использовать в других проектах. Достаточно подать на вход оцифрованный сигнал в формате I2S.

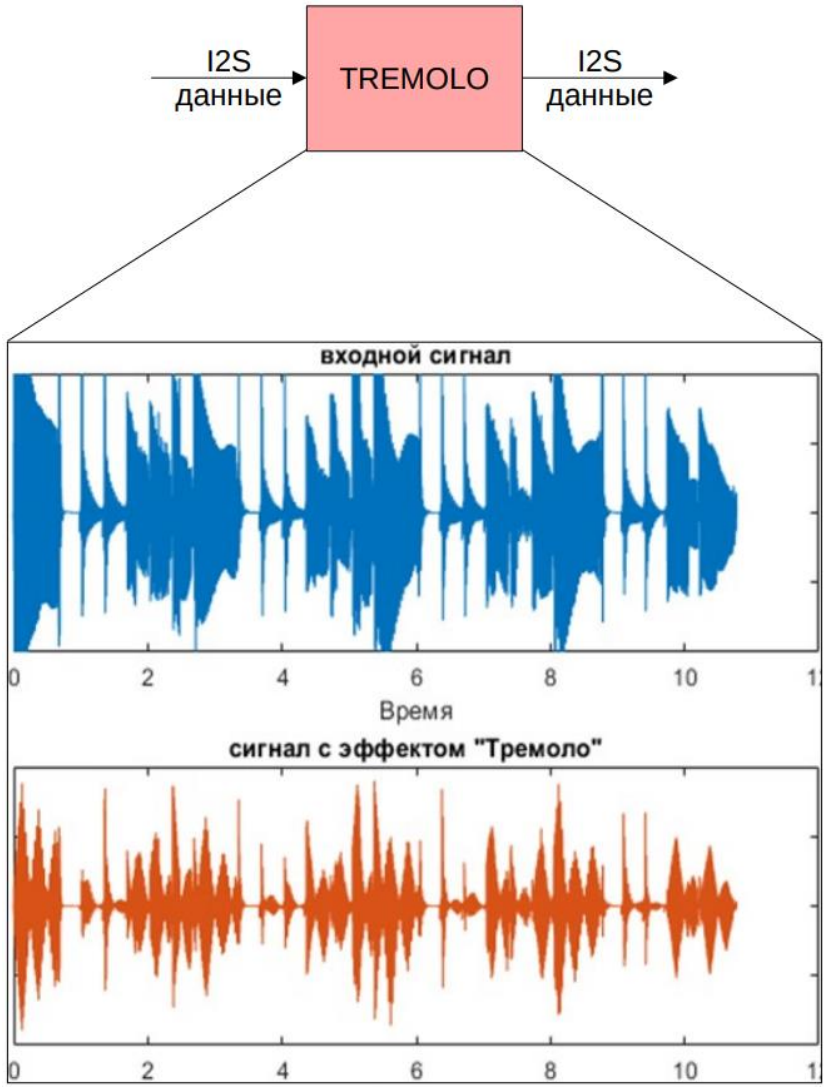

Рисунок 29 – СФ-блок, который применяет эффект «Тремоло» к входному звуковому сигналу

СФ-блоки можно комбинировать в проекте, выстраивая цепочки СФ-блоков. Схема (Рисунок 30) описывает устройство ПЛИС-проекта на уровне СФ-блоков. Таблица 1 описывает назначение каждого СФ-блока.

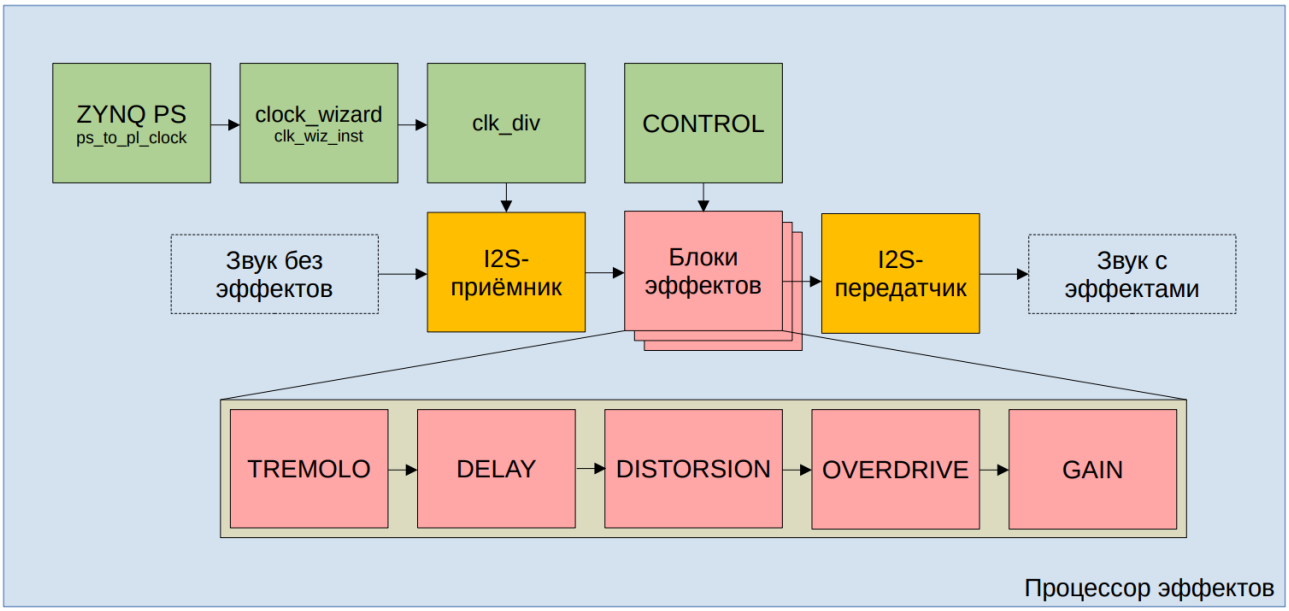

Рисунок 30 – Схема ПЛИС-проекта на уровне СФ-блоков

Таблица 1 – Назначение СФ-блоков ПЛИС-проекта

Название	**Назначение**
I2S-приёмник	Реализует интерфейс работы с модулем АЦП PCM1808
I2S-передатчик	Реализует интерфейс работы с модулем ЦАП PCM5102a
tremolo	Реализует звуковой эффект «тремоло»
delay	Реализует звуковой эффект «задержка»
distortion	Реализует звуковой эффект «искажение»
overdrive	Реализует звуковой эффект «перегрузка»
gain	Реализует звуковой эффект «усиление»
control	Реализует обработку взаимодействия с пользователем
ZYNQ PS	Предоставляет источник тактовых импульсов для ПЛ
clock_wizard	Предоставляет источник тактовых импульсов 30МГц для модулей АЦП и ЦАП.

clk_div	Реализует делитель частот на основе счётчика

ПЛИС-проект представляет из себя цепочку СФ-блоков. Блоки, реализующие эффекты могут быть соединены последовательно, таким образом процессор эффектов позволяет накладывать несколько эффектов одновременно. Диаграмма ниже (Рисунок 31) показывает, как устроена последовательность включения эффектов.

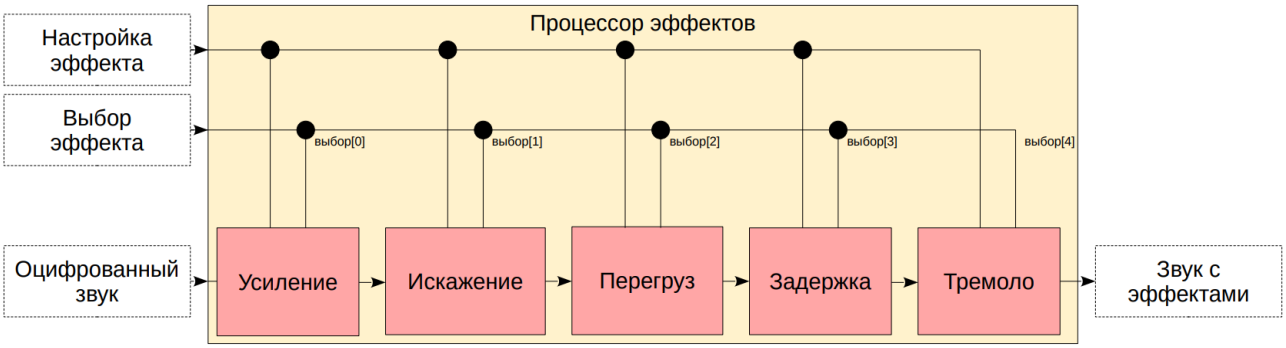

Рисунок 31 – Последовательность включения эффектов

3.4 Вывод

Была рассмотрена архитектура проектируемого устройства. Также определён набор реализуемых сложно-функциональных блоков.

4 Разработка, отладка и отработка сложно-функциональных блоков

4.1 Подсистема тактирования в проекте

4.1.1 Настройка тактирования ПЛ Xilinx ZYNQ 7010

Поскольку на отладочной плате EBAZ4205 отсутствует кварцевый резонатор для тактирования программируемой логики, то источником тактовых импульсов может выступать процессорная система ZYNQ [12]. Для этого необходимо сконфигурировать проект Xilinx Vivado, добавив туда СФ-блок «ZYNQ7 Processing System» с активированным выходом FCLK_CLK0 (Рисунок 32) и созданным портом CLK0 (Рисунок 33), который будет тактировать ПЛ от процессорной части с частотой 50 МГц.

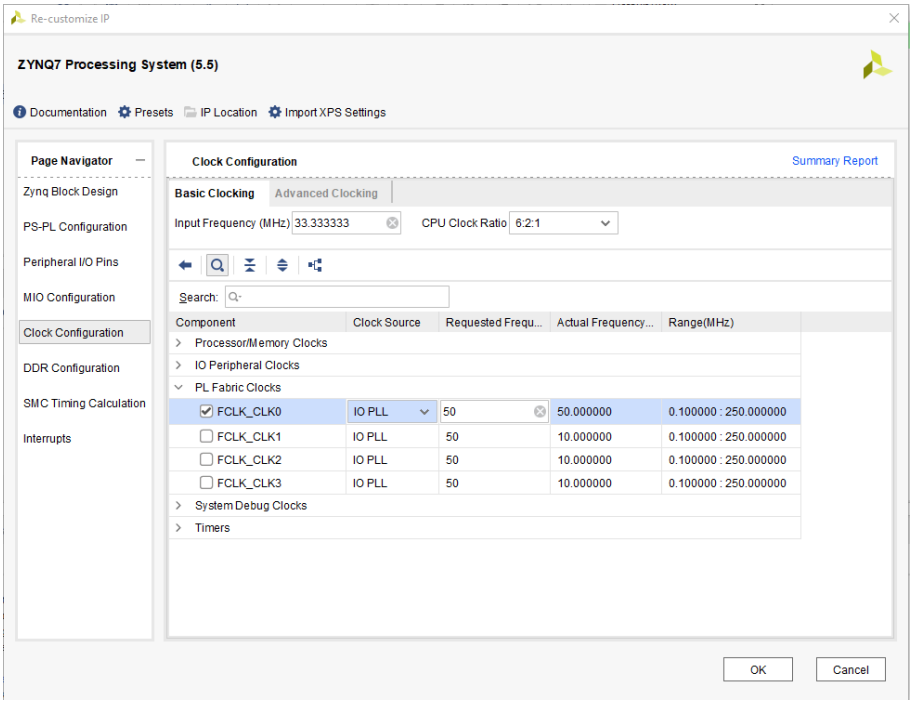

Рисунок 32 – Активация источника тактовых импульсов от процессорной части Xilinx ZYNQ 7010

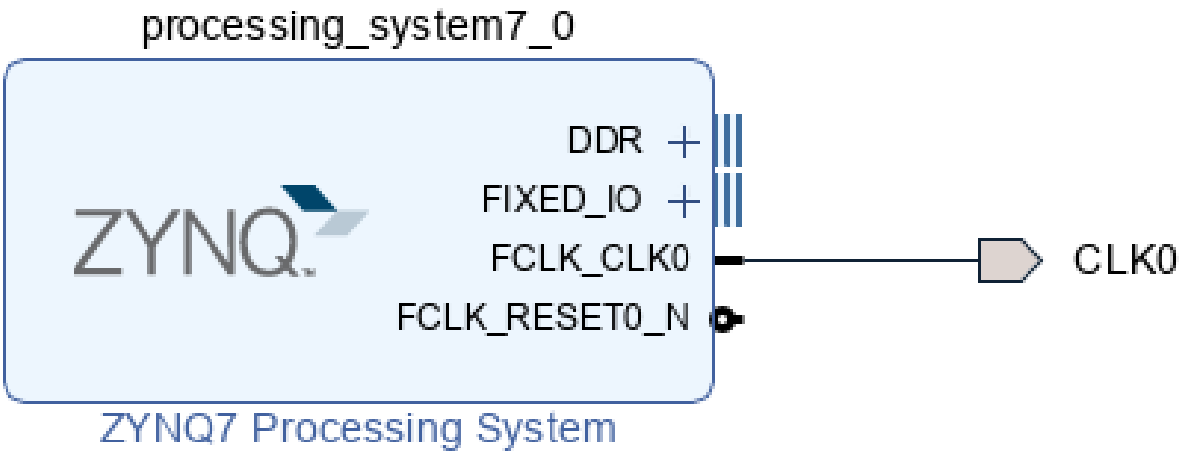

Рисунок 33 – Добавленная в проект процессорная часть Xilinx ZYNQ 7010 с выходом тактовых импульсов

После добавления показанного выше СФ-блока, необходимо создать HDL-обёртку в Xilinx Vivado, а далее модуль можно импортировать (Рисунок 34) в рабочие проекты, где необходим источник тактовых импульсов. Например, можно создать переменную Verilog с типом wire, которая будет инициализировать модуль.

```
wire clk;
ps_to_pl_clock_wrapper this (
    .CLK0(clk)
);
```

Рисунок 34 – Импорт источника тактовых импульсов в проект топ-уровня

4.1.2 Тактирование разрабатываемого процессора эффектов

Отдельно стоит рассмотреть схему тактирования процессора эффектов и тактирование его внутренних составляющих. Модули АЦП PCM1808 и ЦАП PCM5102a синхронно принимают и отдают аудиоданные определённым образом

[4][5]. При различных значениях на системном тактовом входе ЦАП и АЦП, можно получить разные частоты дискретизации звука (Рисунок 35).

Sampling Frequency	System Clock Frequency (f_{SCK}) (MHz)											
	64 f_S	128 f_S	192 f_S	256 f_S	384 f_S	512 f_S	768 f_S	1024 f_S	1152 f_S	1536 f_S	2048 f_S	3072 f_S
8 kHz	–(1)	1.0240(2)	1.5360(2)	2.0480	3.0720	4.0960	6.1440	8.1920	9.2160	12.2880	16.3840	24.5760
16 kHz	–(1)	2.0480(2)	3.0720(2)	4.0960	6.1440	8.1920	12.2880	16.3840	18.4320	24.5760	36.8640	49.1520
32 kHz	–(1)	4.0960(2)	6.1440(2)	8.1920	12.2880	16.3840	24.5760	32.7680	36.8640	49.1520	–(1)	–(1)
44.1 kHz	–(1)	5.6488(2)	8.4672(2)	11.2896	16.9344	22.5792	33.8688	45.1584	–(1)	–(1)	–(1)	–(1)
48 kHz	–(1)	6.1440(2)	9.2160(2)	12.2880	18.4320	24.5760	36.8640	49.1520	–(1)	–(1)	–(1)	–(1)
88.2 kHz	–(1)	11.2896(2)	16.9344	22.5792	33.8688	45.1584	–(1)	–(1)	–(1)	–(1)	–(1)	–(1)
96 kHz	–(1)	12.2880(2)	18.4320	24.5760	36.8640	49.1520	–(1)	–(1)	–(1)	–(1)	–(1)	–(1)
176.4 kHz	–(1)	22.5792	33.8688	45.1584	–(1)	–(1)	–(1)	–(1)	–(1)	–(1)	–(1)	–(1)
192 kHz	–(1)	24.5760	36.8640	49.1520	–(1)	–(1)	–(1)	–(1)	–(1)	–(1)	–(1)	–(1)
384 kHz	24.5760	49.1520	–(1)	–(1)	–(1)	–(1)	–(1)	–(1)	–(1)	–(1)	–(1)	–(1)

Рисунок 35 – Соответствие режимов и частот функционирования PCM5102a

Например, при подаче 16.9344 МГц (SCKI) на этот вход в режиме $384f_s$ частота передачи данных (BCK) составит 2.8224 МГц, а результирующая частота дискретизации составит 44100 Гц (LRCK). Это стандартное студийное значение для звука на компакт-дисках. Однако, существует проблема с тем, чтобы точно сгенерировать такое значение SCKI. Поэтому было принято решение подать на системный вход тактовых импульсов 30 МГц. В режиме $384f_s$ итоговая частота дискретизации звука будет равна 78125 Гц, а BCK – 5 МГц. Данные расчёты одинаково актуальны как для PCM1808, так и для PCM5102a. Рисунок 36 показывает, как генерируются тактовые импульсы различной частоты.

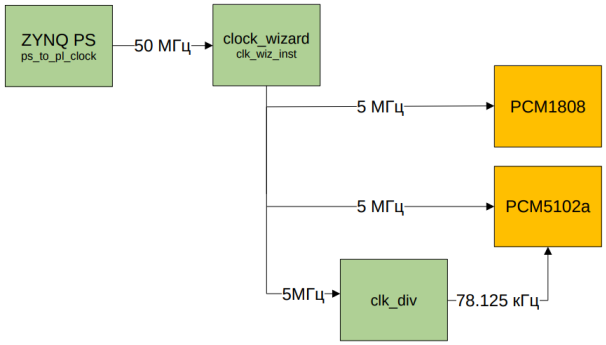

Рисунок 36 – Схема взаимодействия СФ-блоков и модулей АЦП и ЦАП относительно тактовых импульсов

4.2 Разработка СФ-блока «clk_div»

Данный сложный функциональный блок выполняет функцию цифрового делителя частот на основе счётчика. На графическом представлении (Рисунок 37) можно видеть входные и выходные сигналы СФ-блока. Таблица 2 описывает их назначение.

Рисунок 37 – Графическое обозначение СФ-блока «clk_div»

Таблица 2 – Назначение сигналов СФ-блока «clk_div»

Сигнал	Описание
reset_n	Входной сигнал сброса делителя по низкому уровню
clk	Входной опорный сигнал тактовых импульсов
clk_out	Выходной сигнал тактовых импульсов

У СФ-блока «clk_div» есть два входных параметра:

– **Presc**. Определяет значение предделителя частоты;

– **Width**. Определяет разрядность счётчика.

Для тестирования сложнофункционального блока был написан тест-бенч на Verlog. Результаты тестирования (Рисунок 38) при входных параметрах Presc=8, Width=32 показывают, что входной тактовый сигнал с частотой 50 МГц был поделен на 8, т.е. итоговая частота составила 6.25МГц.

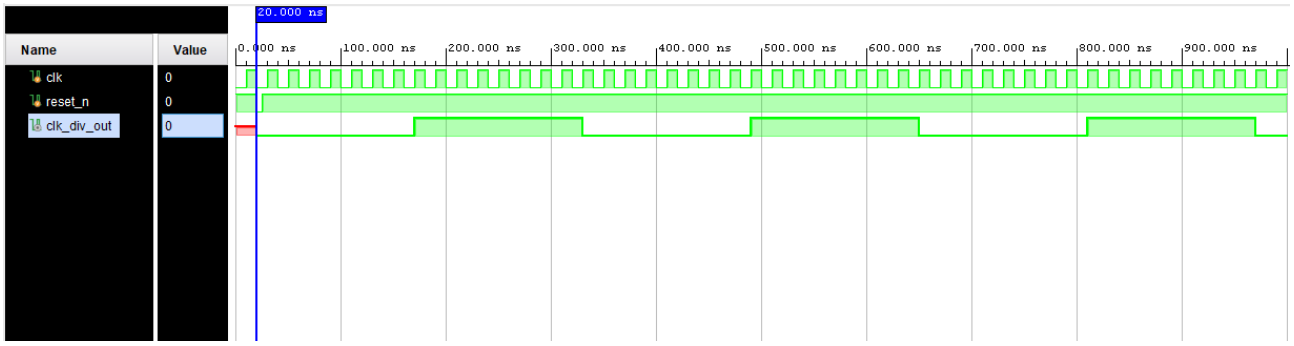

Рисунок 38 – Результат тестирования СФ-блока «clk_div»

4.3 Разработка СФ-блока «I2S-передатчик»

Сложно-функциональный блок i2s_transceiver выполняет функцию отправителя I2S-закодированного аудио согласно документации на ЦАП PCM5102a [5]. Разработанный Verilog-модуль принимает входные массивы аудиоданных для каждого канала и выполняет их сериализацию. Под сериализацией подразумевается преобразование параллельных данных в последовательное представление. На графическом представлении СФ-блока (Рисунок 39) можно видеть входные и выходные сигналы. Таблица 3 описывает их назначение.

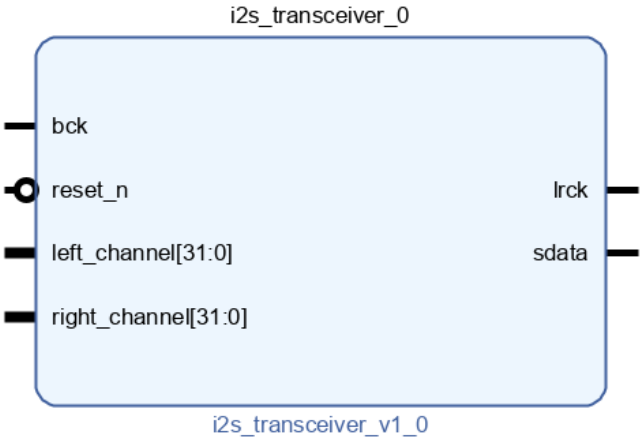

Рисунок 39 – Графическое обозначение СФ-блока «I2S_transceiver»

Таблица 3 – Назначение сигналов СФ-блока «I2S-передатчик»

Сигнал	Описание
reset_n	Входной сигнал сброса по низкому уровню
bck	Входной сигнал тактовых импульсов передачи данных
lrck	Выходной сигнал смены канала отправления
sdata	Выходной сигнал последовательных данных I2S
left_channel	Входная 32-хразрядная шина, передающая значение I2S для левого канала
right_channel	Входная 32-хразрядная шина, передающая значение I2S для правого канала

У СФ-блока «I2S-передатчик» есть один входной параметр:

– **Width.** Определяет разрядность аудиоданных для одного канала.

– **Presc.** Определяет значение предделителя частоты для генерации сигнала LRCK.

Для тестирования сложнофункционального блока был написан тест-бенч на Verilog. Результаты тестирования показаны ниже (Рисунок 40).

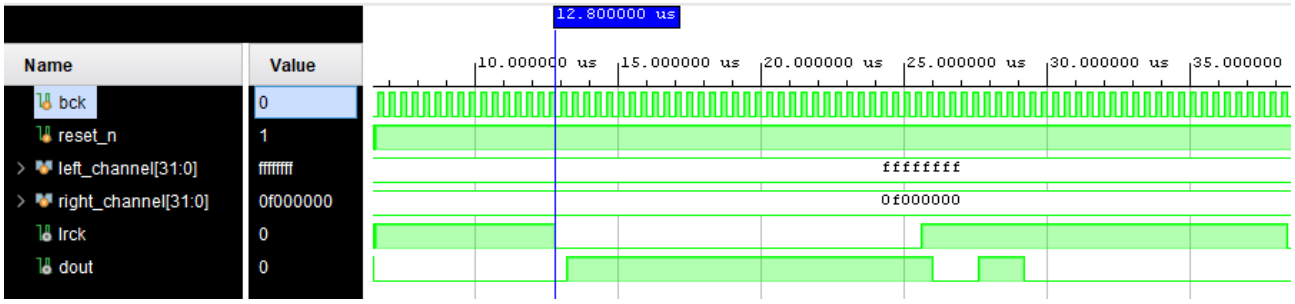

Рисунок 40 – Результат тестирования СФ-блока «I2S-передатчик»

Показанную выше диаграмму можно сравнить с диаграммой из документации на принимающее устройство ЦАП PCM5102a (Рисунок 41). При использовании формата данных I2S во всех режимах, данные для выбранного канала на шине DATA действительны только по истечении такта после переключения шины LRCK.

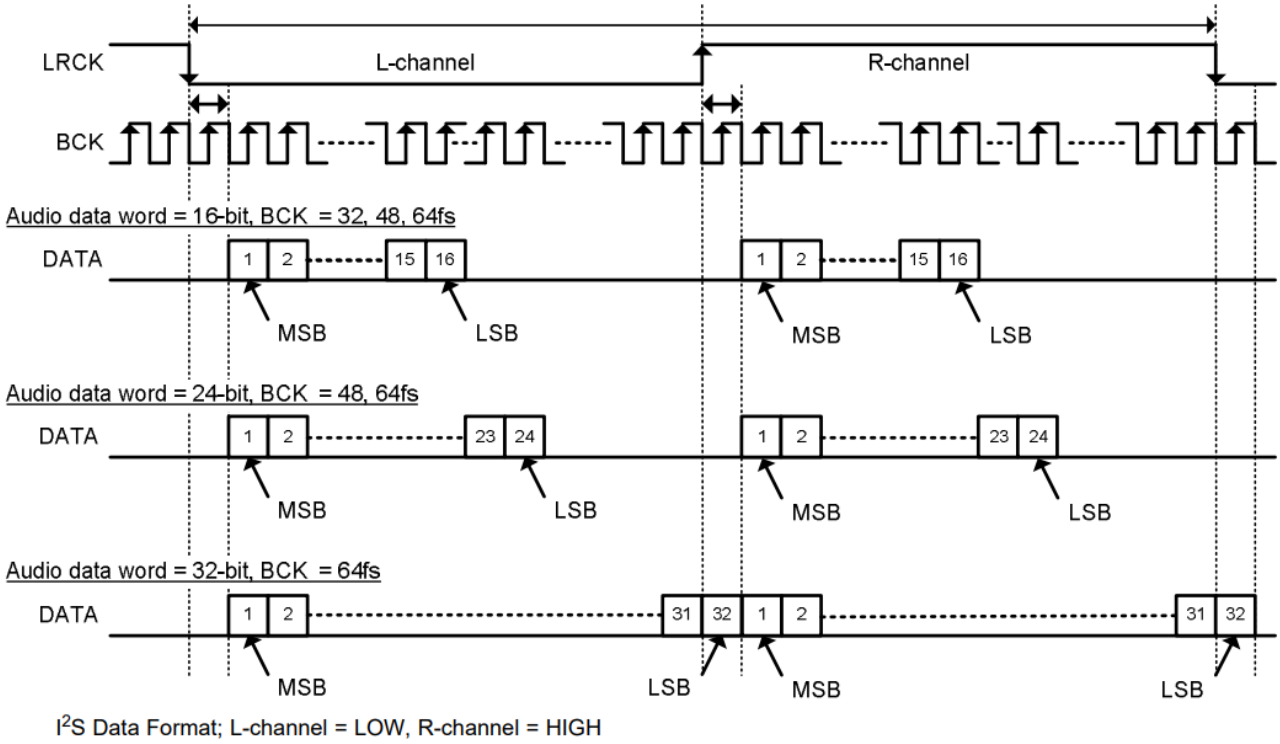

Рисунок 41 – Формат приёма данных PCM5102a

Исходя из результатов тестирования СФ-блока, полученный формат данных полностью соответствует правильному формату, который предусмотрен устройством приёма.

Функционально работа СФ-блока сводится к тому, чтобы по падающему фронту BCK выставлять данные правого и левого аудиоканалов на последовательную шину DOUT и каждые 64 такта BCK переключать сигнал LRCK.

4.4 Разработка СФ-блока «I2S-приёмник»

Сложно-функциональный блок i2s_receiver выполняет функцию считывателя I2S-закодированного аудио, занимая промежуточное место между блоками обработки звука и устройством, этот звук предоставляющим в формате I2S (АЦП). На графическом представлении (Рисунок 42) можно видеть входные и выходные сигналы СФ-блока. Таблица 4 описывает их назначение.

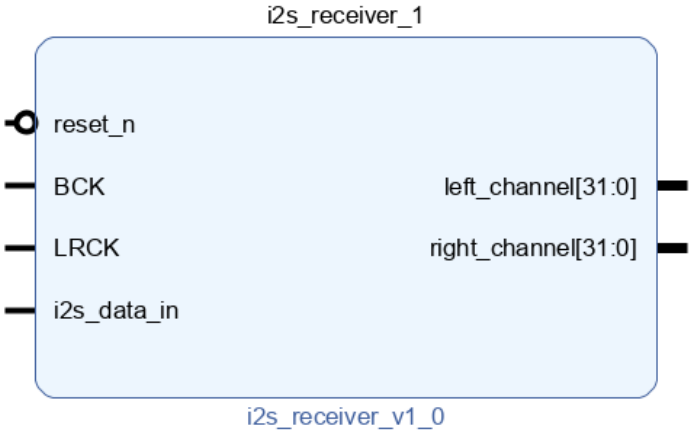

Рисунок 42 – Графическое обозначение СФ-блока «I2S-приёмник»

Таблица 4 – Назначение сигналов СФ-блока «I2S-приёмник»

Сигнал	Описание
reset_n	Входной сигнал сброса по низкому уровню
BCK	Входной сигнал тактовых импульсов передачи данных
LRCK	Выходной сигнал смены канала считывания
i2s_data_in	Входной сигнал последовательных данных I2S
left_channel	Выходная 32-хразрядная шина, передающая считанное значение аудио для левого канала
right_channel	Выходная 32-хразрядная шина, передающая считанное значение аудио для правого канала

У СФ-блока «I2S-приёмник» есть один входной параметр:

– **Width.** Определяет разрядность аудиоданных для одного канала.

Тестирование данного модуля удобнее всего производить в паре с использованием СФ-блока «I2S-передатчик». Блок-диаграмма проекта для тестирования показана ниже (Рисунок 43).

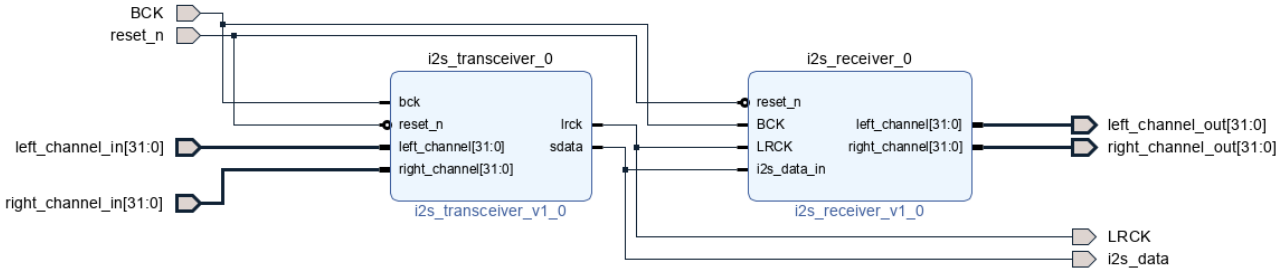

Рисунок 43 – Блок-диаграмма проекта-тестбенча

Так как работа передатчика полностью соответствует формату I2S, то в случае верной работы приёмника, значения в регистрах left_channel_in, left_channel_out, right_channel_in и right_channel_out должны совпадать. Далее для тестирования был написан тестбенч. Частота BCK составила 5 МГц, а LRCK – 78,125 кГц.

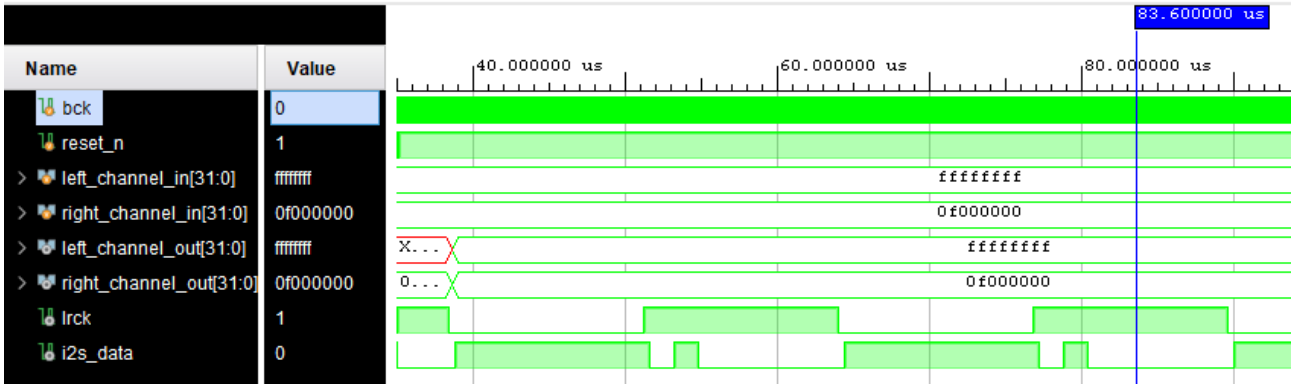

Рисунок 44 – Результат тестирования СФ-блока «I2S-приёмник»

Как можно заметить выше, данные на входе передатчика и выходе приёмника совпали, а каналы не были перепутаны. Это означает, что СФ-блок «I2S-приёмник» работает корректно.

4.5 Разработка СФ-блока «Усиление»

Задача данного СФ-блока сводится к тому, чтобы умножить входящие значение аудиосигнала на какое-либо константное значение.

СПИСОК СОКРАЩЕНИЙ

ЦСП – цифровой сигнальный процессор;

СФ-блок – сложно-функциональный блок;

ПЛИС – программируемая логическая схема;

ПЛ – программируемая логика;

ЦСП – цифровой сигнальный процессор.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 HeadRush Pedalboard: официальный сайт. – URL: https://www.headrushfx.com/products/pedalboard (дата обращения: 15.12.2022)

2 EBAZ4205. Xjtuecho [Электронный ресурс] / – URL: https://github.com/xjtuecho/EBAZ4205 (дата обращения: 21.12.2022)

3 Zynq-7000 All Programmable SoC Data Sheet: Overview. Xilinx [Электронный ресурс] / – URL: https://www.farnell.com/datasheets/2553940.pdf (дата обращения: 21.12.2022)

4 PCM1808 Reference Manual. Analog Devices [Электронный ресурс] / – URL: https://static.chipdip.ru/lib/170/DOC001170929.pdf (дата обращения: 21.12.2022)

5 PCM5102a Reference Manual. Analog Devices [Электронный ресурс] / – URL: https://static.chipdip.ru/lib/145/DOC004145096.pdf (дата обращения: 21.12.2022)

6 I2S Bus Specification. Philips Semiconductors [Электронный ресурс] / – URL: https://www.sparkfun.com/datasheets/BreakoutBoards/I2SBUS.pdf (дата обращения: 22.12.2022)

7 Design of DSP Guitar Effects with FPGA Implementation [Электронный ресурс] / – URL: https://scholarworks.rit.edu/cgi/viewcontent.cgi?article=11514&context=theses (дата обращения: 28.03.2023)

8 Гитарные эффекты: алгоритмы, первый опыт аппаратной реализации [Электронный ресурс] / – URL: https://habr.com/ru/post/369391/ (дата обращения: 22.12.2022)

9 Adding sound effects to your music. Rhea [Электронный ресурс] / – URL: https://www.projectrhea.org/rhea/index.php/Adding_sound_effects_to_your_music (дата обращения: 01.04.2023)

10 Digital Audio Effects. Cardiff University [Электронный ресурс] / – URL: https://users.cs.cf.ac.uk/Dave.Marshall/CM0268/PDF/10_CM0268_Audio_FX.pdf (дата обращения: 01.04.2023)

11 Физический уровень – Модуляции. ИТМО вики-конспекты [Электронный ресурс] / – URL: https://neerc.ifmo.ru/wiki/index.php?title=Физический_уровень_-_Модуляции (дата обращения: 05.04.2023)

12 ZYNQ uses PS to light up PL (detailed version). ProgrammerSought [Электронный ресурс] / – URL: https://www.programmersought.com/article/69677090787/ (дата обращения: 07.04.2023)

ПРИЛОЖЕНИЕ А

Реализация СФ-блока «clk_div»

```
`timescale 1ns / 1ps

module clk_div
#(parameter WIDTH = 32, PRESC = 8) (
    input reset_n,
    input clk,
    output reg clk_out
    );

    reg [WIDTH-1:0] bit_cnt = 1'b1;
    always @(posedge clk or negedge reset_n)
      if (!reset_n)
         bit_cnt <= 1;
      else if (bit_cnt >= PRESC)
            bit_cnt <= 1;
      else
            bit_cnt <= bit_cnt + 1;

    always @(posedge clk or negedge reset_n)
      if (!reset_n)
         clk_out <= 1'b0;
      else if (bit_cnt == PRESC)
            clk_out <= ~clk_out;

endmodule
```

ПРИЛОЖЕНИЕ Б

Тестбенч для СФ-блока «clk_div»

```
`timescale 1ns / 1ps

module clk_div_tb();
    reg clk, reset_n;
    wire clk_div_out;

    initial begin
        clk = 0;
        reset_n = 1;
    end

    initial begin
        #20 reset_n = 0;
        #5 reset_n = 1;
    end

    always
        #10 clk=~clk;

    design_1_wrapper clk_div_inst (
        .clk(clk),
        .clk_out(clk_div_out),
        .reset_n(reset_n)
    );
endmodule
```

ПРИЛОЖЕНИЕ В

Реализация СФ-блока «I2S-передатчик»

```
`timescale 1ns / 1ps

module i2s_transceiver #(
      parameter WIDHT   = 32, PRESC = 32
)(
      input bck,
    input reset_n,

      output reg lrck,
      output reg sdata,

      // Parallel datastreams
      input [WIDHT-1:0] left_channel,
      input [WIDHT-1:0] right_channel
);

reg [WIDHT-1:0]   bit_cnt;
reg [WIDHT-1:0]   right;
reg [WIDHT-1:0]   left;

always @(negedge bck or negedge reset_n)
      if (!reset_n) begin
            bit_cnt <= 1;
            sdata <= 0;
            left <= 0;
            right <= 0;
    end
      else if (bit_cnt >= PRESC)
            bit_cnt <= 1;
      else
            bit_cnt <= bit_cnt + 1;

// Sample channels on the transfer of the last bit of the right channel
always @(negedge bck)
      if (bit_cnt == PRESC && lrck) begin
        right <= right_channel;
        left <= left_channel;
      end

// left/right "clock" generation - 0 = left, 1 = right
always @(negedge bck or negedge reset_n)
      if (!reset_n)
            lrck <= 1;
      else if (bit_cnt == PRESC)
            lrck <= ~lrck;

always @(negedge bck)
      sdata <= lrck ? right[WIDHT - bit_cnt] : left[WIDHT - bit_cnt];

endmodule
```

ПРИЛОЖЕНИЕ Г

Тестбенч для СФ-блока «I2S-передатчик»

```
`timescale 1ns / 1ps

module i2s_transceiver_tb();
    reg bck, reset_n;
    reg [31:0] left_channel, right_channel;
    wire lrck, dout;

    initial begin
        left_channel = 32'hFFFFFFFF;
        right_channel = 32'h0F000000;
        bck = 0;
        reset_n = 1;
    end

    initial begin
        #20 reset_n = 0;
        #5 reset_n = 1;
    end

    always begin
        #200 bck=~bck;
    end

    design_1_wrapper i2s_tx_inst(
        .reset_n(reset_n),
        .BCK(bck),
        .left_channel(left_channel),
        .right_channel(right_channel),
        .LRCK(lrck),
        .DOUT(dout)
    );
endmodule
```

ПРИЛОЖЕНИЕ Д

Реализация СФ-блока «I2S-приёмник»

```
`timescale 1ns / 1ps

module i2s_receiver #(
      parameter WIDTH = 32
)(
    input reset_n,
      input BCK,
      input LRCK,
      input i2s_data_in,

      // Parallel datastreams
      output reg [WIDTH-1:0] left_channel,
      output reg [WIDTH-1:0] right_channel
);

reg [WIDTH-1:0]   left;
reg [WIDTH-1:0]   right;
reg   lrclk_r;
wire lrclk_nedge;

assign lrclk_nedge = !LRCK & lrclk_r;

always @(posedge BCK)
      lrclk_r <= LRCK;

// sdata msb is valid one clock cycle after lrclk switches
always @(posedge BCK)
      if (lrclk_r)
            right <= {right[WIDTH-2:0], i2s_data_in};
      else
            left <= {left[WIDTH-2:0], i2s_data_in};

always @(posedge BCK or negedge reset_n)
      if (!reset_n) begin
            left_channel <= 0;
            right_channel <= 0;
      end
      else if (lrclk_nedge) begin
            left_channel <= left;
            right_channel <= {right[WIDTH-2:0], i2s_data_in};
      end
endmodule
```

ПРИЛОЖЕНИЕ Е

Тестбенч для СФ-блока «I2S-приёмник»

```
`timescale 1ns / 1ps


module i2s_receiver_tb();

    reg bck, reset_n;
    reg [31:0] left_channel_in, right_channel_in;
    wire [31:0] left_channel_out, right_channel_out;
    wire lrck, i2s_data;

    initial begin
        left_channel_in = 32'hFFFFFFFF;
        right_channel_in = 32'h0F000000;
        bck = 0;
        reset_n = 1;
    end

    initial begin
        #20 reset_n = 0;
        #5 reset_n = 1;
    end

    always begin
        #200 bck=~bck;
    end

    design_1_wrapper i2s_rx_tx (
        .BCK(bck),
        .LRCK(lrck),
        .i2s_data(i2s_data),
        .reset_n(reset_n),
        .left_channel_in(left_channel_in),
        .right_channel_in(right_channel_in),
        .left_channel_out(left_channel_out),
        .right_channel_out(right_channel_out)
    );

endmodule
```