МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра физики твердого тела

Разработка модели блока цифрового синтезатора частот, управляемого процессорным ядром архитектуры RISC-V

АВТОРЕФЕРАТ МАГИСТЕРСКОЙ РАБОТЫ

магистранта 2 курса
направления 11.04.04 «Электроника и наноэлектроника»
профиль «Формирование и диагностика микро-, нано- и биомедицинских систем»
Института физики

Алексеева Анатолия Игорьевича

Научный руководитель:

профессор, д.ф. – м.н.	_____ 10.06.2025	А. А. Семёнов
должность, уч.степень, звание	подпись, дата	инициалы, фамилия

Зав. кафедрой

профессор, д.ф. – м.н.	_____	А. В. Скрипаль
должность, уч.степень, звание	подпись, дата	инициалы, фамилия

Саратов 2025

Введение. Прямой цифровой синтез (от английского DDS – Direct Digital Synthesizer) представляет собой метод получения аналогового сигнала (обычно синусоидального, пилообразного или последовательности треугольных импульсов) путём генерации последовательности цифровых отсчётов и их последующего преобразования в аналоговую форму с помощью ЦАП. Поскольку сигнал сначала создаётся в цифровом виде, такое устройство обеспечивает быстрое изменение частоты, высокую точность по частотной сетке и возможность работы в широком диапазоне частот.

Аппаратное объединение устройства DDS с быстродействующим управляющим микропроцессорным ядром может позволить создать эффективный синтезатор частот, реализуемый в рамках одной БИС программируемой логической интегральной схемы.

Управляющее устройство и микросхема ЦВС являются отдельными микросхемами то возникает проблема миниатюризации данного устройства.

Соответственно, данные недостатки можно устранить, разместив управляющее устройство и алгоритм ЦВС на одной микросхеме.

Структурная схема данного решения, разработанного в рамках данной дипломной работы, приведена на рисунке 1.

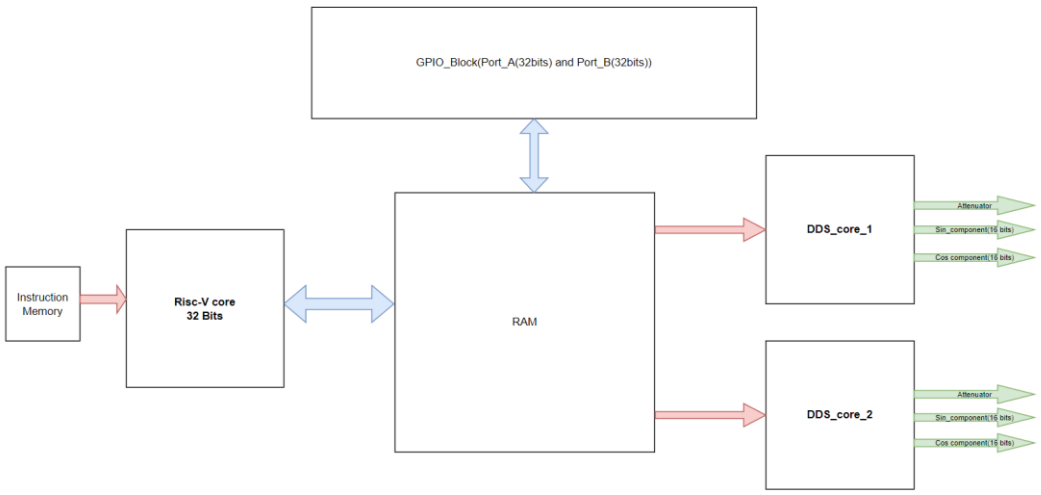

Рисунок 1. Структурная схема реализации микросхемы ЦВС с управляющим устройством.

В данной реализации устройством управления является конвейерное

процессорное ядро RISC-V. Конвейерная реализация необходима для того, чтобы уменьшить критический комбинационный путь в процессоре, что позволит повысить общую тактовую частоту дизайна. Данный процессор обладает базовым набором инструкций RV32I, которое имеет общее адресное пространство с ЦВС блоками. Изменение данных в оперативной памяти ведет моментальную конфигурацию блоков ЦВС. В микросхеме два полностью независимых блока ЦВС, что может быть использовано для квадратурной модуляции выходного сигнала. **Преимуществом данной реализации** является то, что можно сократить расстояние между управляющим устройством и блоками ЦВС, это позволит улучшить временные параметры, что позволит быстрее манипулировать фазой выходного сигнала. Также немаловажным преимуществом данной реализации как было указано выше является возможность уменьшения размеров готового устройства.

Целью магистерской работы являлась разработка модели блока цифрового синтезатора частот, управляемого процессорным ядром архитектуры RISC-V.

Магистерская работа содержит 3 главы:

Глава I МОДУЛЯЦИЯ СИГНАЛОВ

1.1 Амплитудная модуляция

1.2 Частотная модуляция

1.3 Фазовая модуляция

Глава II ПРОЦЕССОР RISC-V

2.1 Общая информация

2.2 Архитектурная реализация RISC-V процессора

2.2.1 Инструкции типа R

2.2.2 Инструкции типа I

2.2.3 Инструкции типа S/B

2.2.4 Инструкции типа U/J

2.3 Микроархитектурная реализация процессора RISC-V

2.3.1 Однотактная реализация RISC-V

1. МОДУЛЯЦИЯ СИГНАЛОВ

Модуляция делится на непрерывную и импульсную. В случае непрерывной модуляции носителем информации служит синусоидальный сигнал — «несущая». Поскольку такой сигнал определяется параметрами амплитуды, частоты и фазы, выделяют три основных типа непрерывной модуляции[1]:

1. Амплитудная модуляция (АМ).
2. Частотная модуляция (ЧМ).
3. Фазовая модуляция (ФМ).

Существуют различные виды этих модуляций (подробнее о них будет рассмотрено в следующих разделах), а также их комбинации, называемые многократными модуляциями. В импульсной модуляции носителем информации служит последовательность импульсов, которая характеризуется такими параметрами, как амплитуда, длительность, временное положение, количество импульсов и другие.

Преимущества модуляции по сравнению с немодулированным сигналом включают:

1. Возможность увеличения числа каналов на одной линии связи.
2. Повышение надежности передачи информации при использовании

[1]Телеконтроль и телеуправление: учебное пособие /С.Н. Ливенцов, Ю.А. Чурсин; Томский политехнический университет – Томск: Изд-во Томского политехнического университета, 2011. – 139 с

помехоустойчивых методов модуляции.

3. Улучшение эффективности излучения сигнала при передаче по радиоканалу.

4. Повышение эффективности каналов связи и снижение стоимости передачи сообщений.

1.1 Амплитудная модуляция

Амплитудная модуляция (АМ) представляет собой процесс формирования сигнала путём изменения амплитуды гармонического колебания («несущей») в соответствии с мгновенными значениями напряжения или тока более низкочастотного электрического сигнала (сообщения).

1.2 Частотная модуляция

При частотной модуляции (ЧМ) мгновенные значения сигнала сообщения (тока или напряжения) изменяют частоту носителя («несущей»), при этом его амплитуда остается постоянной (см. рис. 3):$\omega = \omega_0 + m_{\text{ч}}\Omega\cos(\Omega t)$, где $m_{\text{ч}} = \Delta\omega/\Omega$- коэффициент частотного отклонения, или глубина частотной модуляции;$\Delta\omega$ – девиация угловой частоты, то есть максимальное отклонение частоты носителя от исходного значения; ω_0 – среднее значение угловой частоты переносчика.

1.3 Фазовая модуляция

При фазовой модуляции (ФМ) передаваемая информация влияет на значение фазы φ несущего сигнала. В результате фаза «несущей» φ изменяется в соответствии с мгновенными значениями тока или напряжения модулируемого сообщения (см. рис. 4):

$$\varphi = \Delta\varphi * \sin(\Omega t).$$

2. ПРОЦЕССОР RISC-V

2.1 Общая информация

Архитектура RISC-V была создана как коммерчески востребованная открытая компьютерная архитектура, которая отличается надежностью, эффективностью и гибкостью. В отличие от других архитектур, RISC-V

обладает открытым исходным кодом, использует базовые наборы инструкций для обеспечения совместимости, поддерживает широкий спектр микроархитектур — от встроенных систем до высокопроизводительных вычислительных платформ. Она предлагает как статические, так и настраиваемые расширения, а также включает такие преимущества, как сжатые инструкции и набор инструкций RV128I, что позволяет оптимизировать аппаратную часть и обеспечивает поддержку как текущих, так и будущих технологий, гарантируя долговечность архитектуры. Вокруг RISC-V сформировалось сообщество промышленных и научных партнеров RISC-V International[2], тем самым ускорив инновации и коммерциализацию. Этот консорциум разработчиков также помогает проектировать и ратифицировать архитектуру RISC-V.

Сообщество RISC-V International к 2021 году насчитывает более 500 членов как из академических, так и из промышленных кругов, включая Western Digital, NVIDIA, Microchip и Samsung.

2.2 Архитектурная реализация RISC-V процессора

В архитектуре RISC-V для достижения компромисса применяются четыре формата инструкций[3]: типа R, типа I, типа S/B и типа U/J. Такое ограниченное число форматов обеспечивает унифицированность инструкций и, как следствие, упрощает их аппаратную реализацию. В то же время разные форматы позволяют учитывать различные требования к инструкциям, например, необходимость хранения больших констант внутри них. Инструкции типа R (регистровые), такие как add s0, s1, s2, используют три регистра в качестве операндов. Инструкции типа I (immediate — непосредственные), например addi s3, s4, 42, а также инструкции типа S/B (store/branch — сохранение слова в память или условный переход), такие как sw a0, 4(sp) или beq a0, a1, L1, используют два

[2] About RISC-V International URL: http://riscv.org (дата обращения 29.01.2025)

[3]Сара Л. Харрис, Дэвид Харрис. Цифровая схемотехника и архитектура компьютера: RISC-V / пер. с англ. В. С. Яценкова, А.Ю. Романова; под ред. А. Ю. Романова. – М.: ДМК Пресс, 2021. – 810 с.

регистра и 12- или 13-битную константу.Инструкции типа U/J (upper immediate/jump, старшие разряды константы/безусловный переход), такие как jal ra, factorial, работают с одним регистром и 20- или 21-битной константой.

2.2.1 Инструкции типа R

Инструкции типа R используют три регистра в качестве операндов: два регистра-источника и один регистр-назначение. 32-битная инструкция состоит из шести полей: funct7, rs2, rs1, funct3, rd и op. Каждое поле занимает от трех до семи бит. Операция, выполняемая командой, закодирована в трех полях, выделенных синим цветом: 7-битном поле op (также называемом opcode или кодом операции), а также в полях funct7 и funct3 (называемых функциями). Конкретная операция типа R определяется значениями в полях op и funct; эти биты вместе называют управляющими битами, поскольку они указывают процессору, какую операцию выполнить.

2.2.2 Инструкции типа I

Команды типа I (immediate, непосредственные) используют два регистровых операнда и один непосредственный операнд (константу). К инструкциям типа I относятся addi, andi, ori и xori, операции загрузки (lw, lh, lb, lhu и lbu). Машинный формат похож на формат команд типа R, но вместо полей funct7 и rs2 содержит 12-битное непосредственное поле imm. Поля rs1 и imm представляют собой операнды-источники, а поле rd – регистр-назначение.

2.2.3 Инструкции типа S/B

Аналогично инструкциям типа I, инструкции типа S/B (store/branch, хранение слова в памяти / условный переход) используют два регистровых операнда и один не посредственный операнд (константу). Но в инструкциях типа S/B оба операнда являются регистрами-источниками (rs1 и rs2), тогда как инструкции типа I используют один регистр-источник (rs1) и один регистр-назначение (rd). В отличие от инструкций типа R, здесь поля funct7 и rd заменены на 12-битную константу imm. В коде машинного языка это поле константы разбивается на два битовых блока – биты 31:25 и биты 11:7. Инструкции сохранения слова в памяти используют тип S, а в инструкциях

условного перехода используется тип B. Форматы S и B различаются только тем, как закодирована константа.

2.2.4 Инструкции типа U/J

Инструкции типа U/J (upper immediate/jump, старшие разряды константы/безусловный переход) содержат в своем машинном коде один операнд регистра-назначения rd и 20-битовое поле константы. Аналогично другим видам инструкций, инструкции типа U/J имеют 7-битный opcode. В инструкциях типа U оставшиеся биты предназначены для хранения 20 старших разрядов 32-битной константы. В инструкциях типа J оставшиеся биты выделены под 20 старших бит 21-битной константы смещения безусловного перехода. По аналогии с инструкциями типа B, самый младший значащий бит константы всегда равен 0 и не представлен в коде инструкции типа J.

2.3 Микроархитектурная реализация процессора RISC-V

2.3.1 Однотактная реализация RISC-V

На рисунке 2 показан однотактный процессор с блоком управления, подключенным к тракту данных.

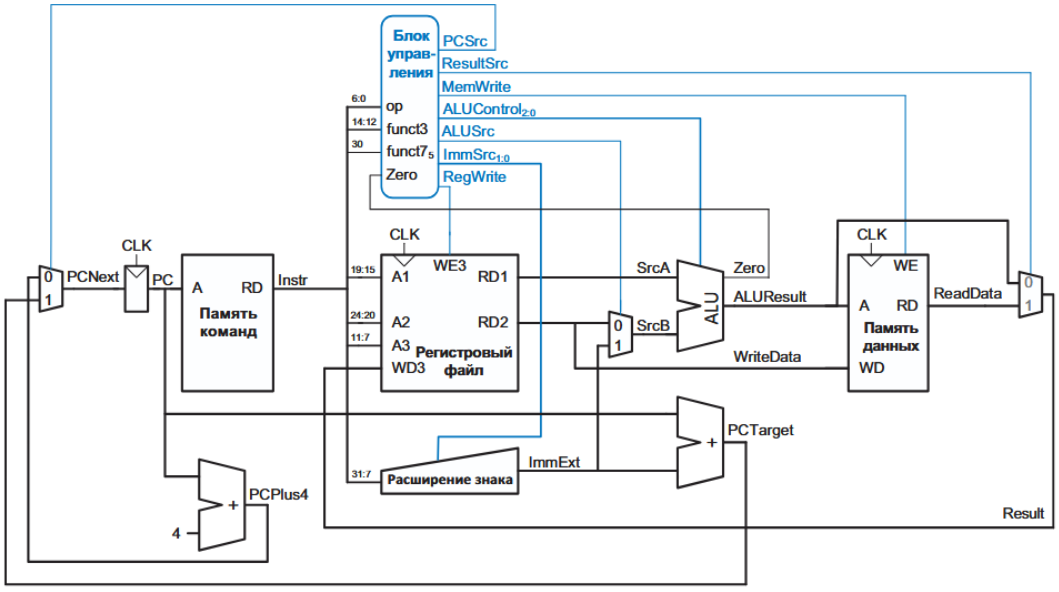

Рисунок 2. Однотактный процессор

Блок управления условно можно разделить на две основные части: главный дешифратор, который генерирует большинство управляющих сигналов, и дешифратор для АЛУ, отвечающий за определение операции,

которую должна выполнить арифметико-логическая единица. В однотактном процессоре применяется отдельная память для команд и данных, поскольку необходимо за один такт одновременно читать команду из памяти и обращаться к памяти данных.

Однотактный процессор обладает тремя основными недостатками. Во-первых, ему необходима отдельная память для команд и данных, что, как правило, трудно реализовать. В большинстве современных систем применяется общая память с возможностью чтения и записи как команд, так и данных. Во-вторых, период тактового сигнала должен быть достаточно большим, чтобы успели выполниться самые медленные команды (например, lw), несмотря на то, что остальные команды работают быстрее. В-третьих, такой процессор требует три сумматора: один для арифметико-логического устройства (АЛУ) и два — для вычисления нового значения счетчика команд; быстрые сумматоры требуют множества логических элементов, что делает их относительно дорогими в реализации.

2.3.2 Многотактная реализация RISC-V

Один из способов устранения проблем, связанных с недостатками однотактного процессора, — применение многотактного процессора, в котором выполнение каждой команды разбито на несколько этапов. Память, АЛУ и регистровый файл являются основными источниками задержек, поэтому для записи в память с примерно одинаковой задержкой на каждом коротком этапе процессор может использовать только один из этих модулей. Такой подход позволяет использовать общую память для команд и данных: команды выбираются на первом этапе, а чтение или запись данных — на одном из последующих. Кроме того, в многотактном процессоре требуется только один сумматор, который на разных этапах может выполнять разные функции.

Количество этапов для выполнения различных команд варьируется: простые команды могут завершаться быстрее, чем сложные. В таком процессоре используется общая память для хранения команд и данных, что реализуемо благодаря тому, что выбор команды происходит на одном такте, а обращение к

памяти данных — на другом. При этом счетчик команд и регистровый файл остаются без изменений.

На рисунке 3 изображен многотактный процессор с системой управления, связанной с цепью данных.

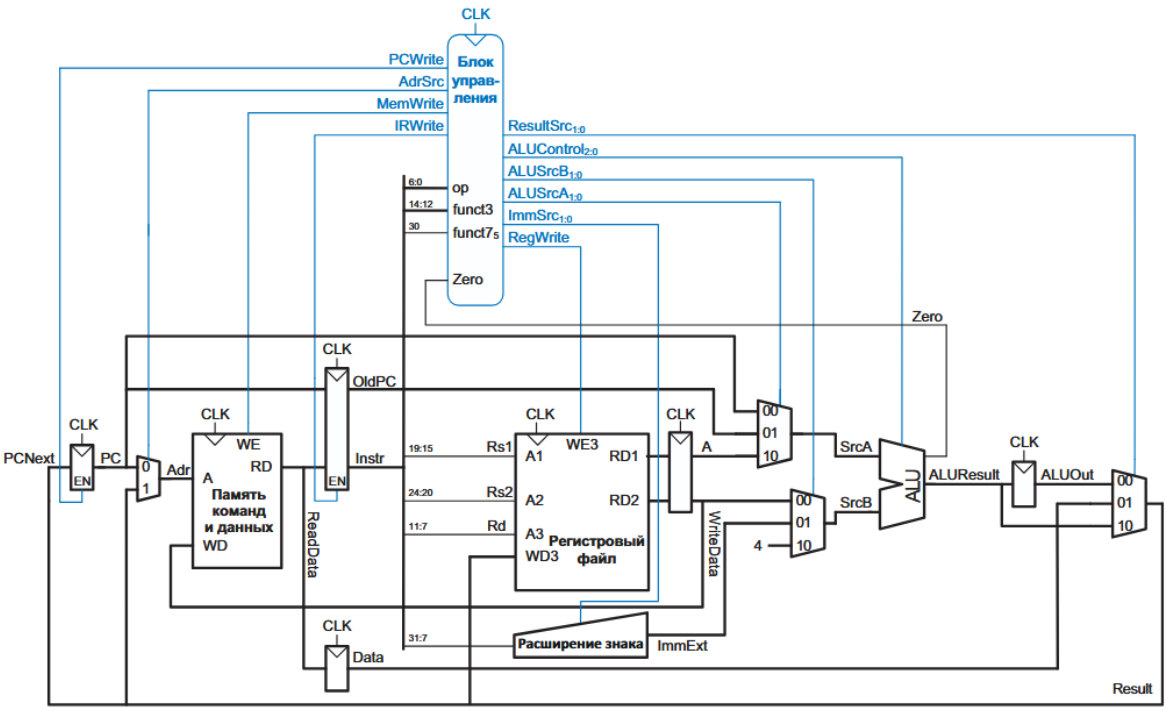

Рисунок 3. Многотактный процессор

Блок управления состоит из дешифратора команд, главного конечного автомата и дешифратора АЛУ. Дешифратор АЛУ такой же, как и в однотактном процессоре (табл. 7.3), но вместо комбинационного основного дешифратора однотактного процессора понадобится основной конечный автомат для генерации последовательности управляющих сигналов при поэтапном выполнении команды.

2.3.3 Конвейерная реализация RISC-V

Обычно микроархитектурная реализация конвейерного процессора на RISC-V состоит из блоков приведенных на рисунке 4.

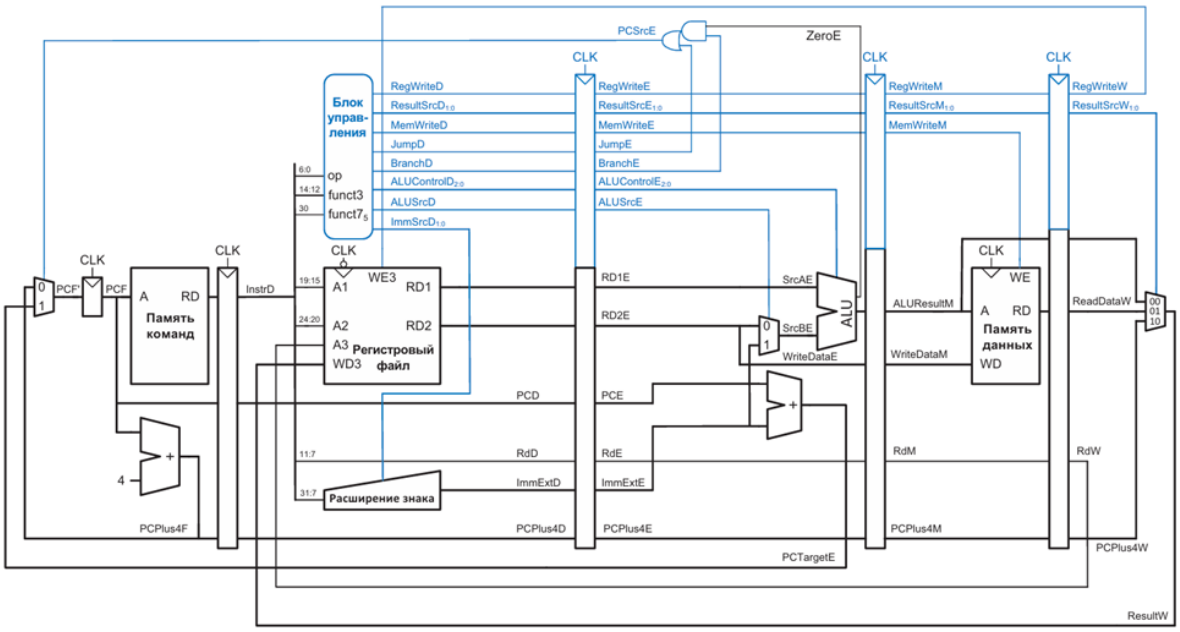

Рисунок 4. Реализация конвейерного процессора на RISC-V

В конвейерном процессоре одновременно выполняется несколько команд. Если одна из них зависит от результатов другой, которая еще не завершена, возникает так называемый конфликт (hazard) в конвейере. Процессор способен за один такт читать и записывать данные в регистровый файл: запись происходит в первой половине такта, а чтение — во второй, поэтому значение в регистре можно записать и затем считать обратно за один такт без возникновения конфликта.

На рисунке 5 показан пример конфликта, который возникает, когда одна команда записывает значение в регистр s8, а следующая команда читает из него. Регистр s8 записывается в регистровый файл на пятом такте, а синие стрелки указывают моменты, когда должна была произойти запись для правильного использования в последующих командах. Такой конфликт называется «чтение после записи» (read after write, RAW). В частности, команда add записывает результат в s8 в первую половину пятого такта. Однако команда sub читает s8 на третьем такте и получает неправильное значение, поскольку запись еще не завершена. Команда or читает s8 на четвертом такте и также получает неверные данные. Только команда and читает s8 во второй половине пятого такта и получает правильное значение, которое было записано ранее. Все последующие

команды также получат правильное значение из s8. Как видно из диаграммы, конфликт возникает именно тогда, когда команда записывает значение в регистр и хотя бы одна из следующих команд его читает. Если не устранить этот конфликт, конвейер даст неправильный результат.

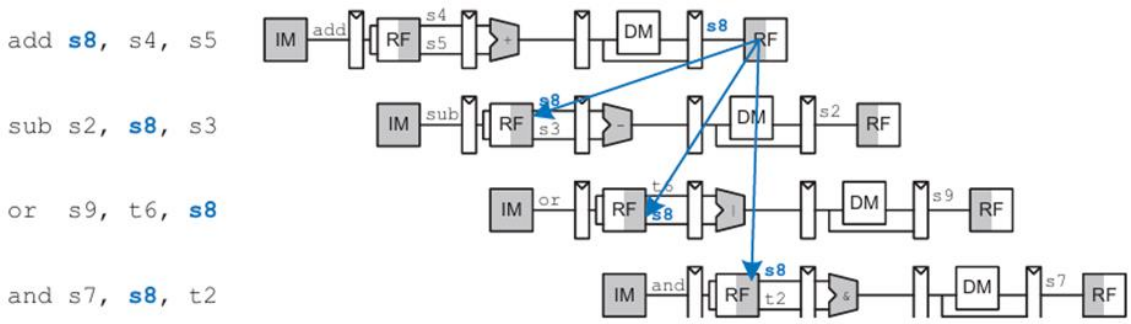

Рисунок 5. Абстрактная схема конвейера, демонстрирующая конфликты.

Для того, чтобы избавиться от конфликта чтения после записи применяют байпасы. Пересылка данных с помощью байпаса требуется в случае, если номер любого из регистров операндов команды, находящейся на стадии выполнения (Execute), совпадает с номером регистра результата команды, находящейся на стадии Memory или Writeback. Каждый такт следует сохранять значение адресов rs1, rs2, rd и сравнивать, используется ли операнд rs1 или rs2 в следующей команде, если условие выполняется, то мы переключаем соответствующий вход АЛУ на выход из АЛУ, тем самым реализовывая механизм байпаса.

3. ПРАКТИЧЕСКАЯ РЕАЛИЗАЦИЯ ЦИФРОВОГО СИНТЕЗАТОРА

3.1 Таблица точек синуса

Была реализована программа генерации таблицы точек синусоиды для загрузки в ПЛИС на языке программирования Python, среда разработки Jupiter Notebook. Количество генерируемых точек равно 32768, количество разрядов каждой точки синуса равно 16, максимальное количество точек равно половине периода синуса. Также программа записывает таблицу в отдельный файл, который можно использовать в среде разработки Quartus, Vivado для загрузки в BRAM память ПЛИС, соответственно количество бит, задействованных для

хранения таблицы синусов равно 524288 бит.

3.2 Реализация алгоритма цифрового синтезатора

Модуль верхнего уровня процессора RISC-V с функцией DDS приведен на рисунке 5.

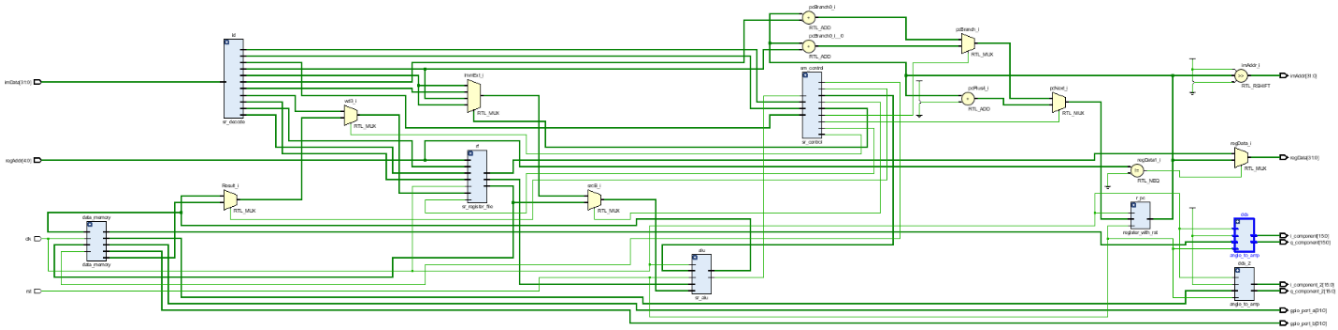

Рисунок 5. Модуль верхнего уровня процессора RISC-V с функцией DDS

Обычно микроархитектурная реализация конвейерного процессора на RISC-V состоит из блоков приведенных на рисунке 4.

В данном модуле объявлены инстансы:

1. Модулей DDS
2. Модуль Декодирования инструкций
3. Регистр хранения текущей инструкции
4. Модуля регистрового файла
5. Модуля АЛУ
6. Модуль Контроль процессора, предназначенный для управления всеми модулями в соответствии с инструкцией

Данный модуль поддерживает работу с памятью, а также инструкцию безусловного перехода jal, выход компоненты синуса и косинуса являются 16 битными кодами амплитуды.

Блок управления (sr_control) можно условно разделить на две основные части: основной дешифратор, который вырабатывает большую часть управляющих сигналов, и дешифратор АЛУ, который решает, какую операцию будет выполнять АЛУ. Основной дешифратор определяет тип инструкции по

коду команды, а затем генерирует соответствующие управляющие сигналы для тракта данных. Основной дешифратор генерирует большинство управляющих сигналов для тракта данных, а также внутренние сигналы Branch и ALUOp для собственных нужд блока управления.

Модуль sr_control представлен на рсиунке 6.

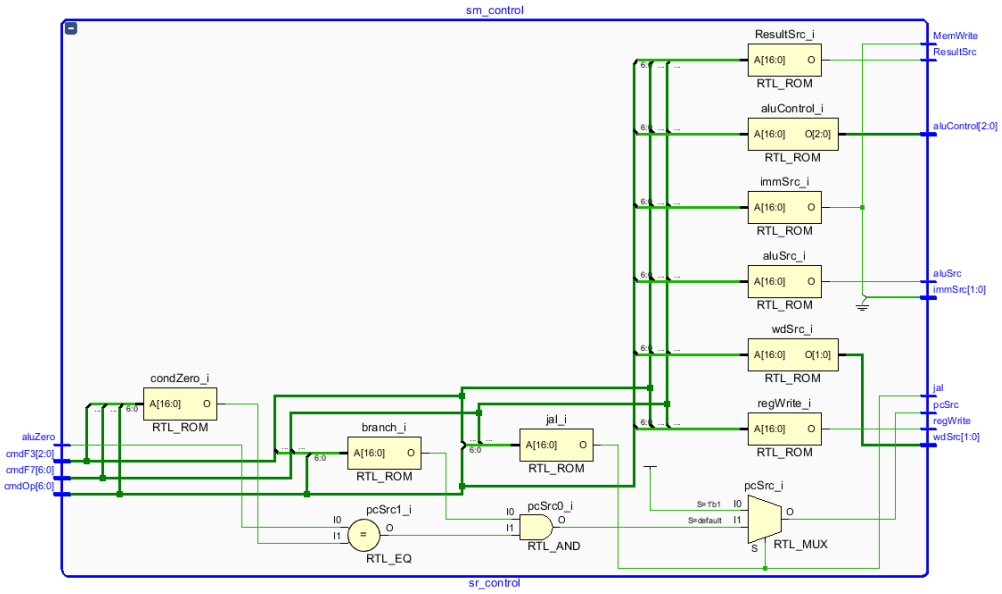

Рисунок 6. Блок управления (sr_control)

Фазовый аккумулятор реализован с помощью накапливающего сумматора и представляет собой регистр, который в каждом такте работы устройства обновляется значением, равным предыдущему значению, увеличенному на код частоты D. Содержимое регистра линейно возрастает со временем, однако приращение не обязательно равно единице — оно зависит от величины кода частоты.

Блок АЛУ выполняет арифметические и логические операции в данной реализации процессора.

Также модуль поддерживает операцию выдачи флага zero, когда выход АЛУ равен нулю. Данный функционал необходим для реализации условных переходов в процессоре. Все операции выполняются за один такт без латентности.

Регистровый файл является памятью для процессора и все инструкции процессора используют именно эту память как основную. В данном модуле

объявлено четыре пятибитных шины адреса, 4 32-разрядных шины данных, и сигнал тактирования с сигналом разрешения записи we3. Внутри модуля объявлено 32 регистра шириной 32 бита. Чтение значений памяти происходит комбинационно, запись по переднему фронту тактового импульса. Соблюдено условие, что при обращении к регистру по нулевому адресу мы выдаем 0. Данный нулевой регистр регламентируется стандартом и используется для выполнения некоторых псевдоинструкций. В блоке always_ff мы реализуем запись по адресу a3 значения на входе wd3 по высокому уровню сигнала we3. Реализация регистрового файла приведена на рисунке 7.

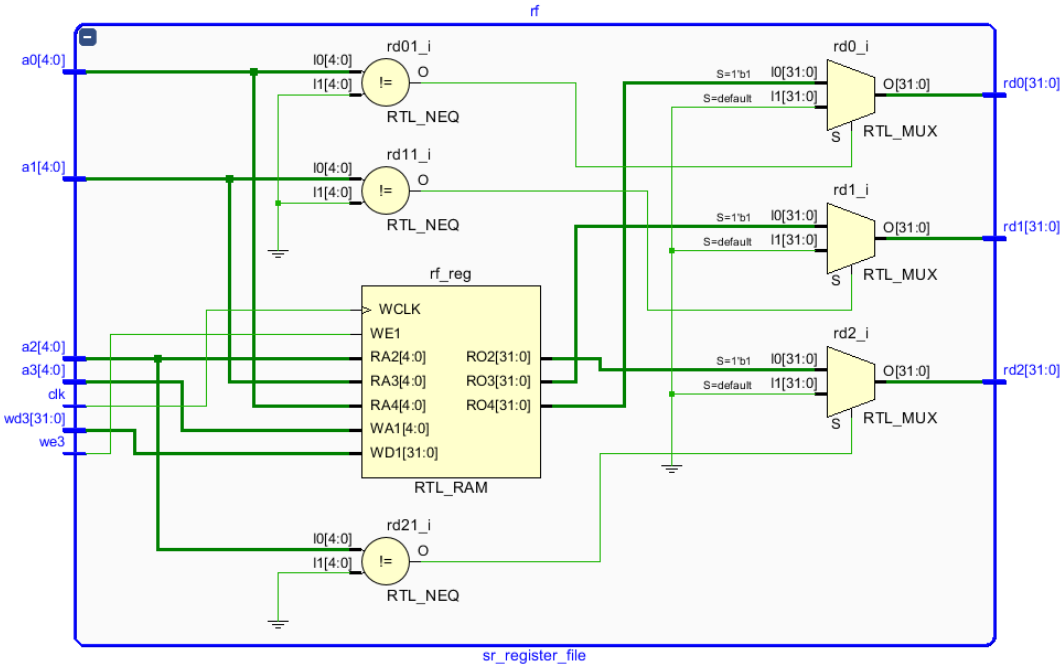

Рисунок 7. Модуль Регистровый файл.

Модуль оперативной памяти RISC-V работает в байтовой адресации при обращении к памяти отрезаются два младших бита. В данном модуле 0x0 адрес ссылается на шаг приращение аккумулятора фазы первого модуля DDS, 0x4 адрес ссылается на шаг приращения аккумулятора фазы второго модуля DDS, 0x8 адрес ссылается на данные блока GPIO порта A, Адрес 0xC ссылается на данные порта B. Глубина памяти задается параметром, что сильно упрощает подстройку глубины памяти на производстве. Данный модуль является законченной версией памяти для процессора RISC-V архитектуры.

Модуль декодирования инструкции предназначен для декодирования пришедшей инструкции в соответствии с полями. Как видно из листинга программного кода процессор поддерживает следующие инструкции:

1. Инструкции типа I
2. Инструкции типа B
3. Инструкции типа S
4. Инструкции типа U

Быстродуйствие дизайна можно улучшить, введя конвейеризацию. Реализация конвейерного АЛУ представлена на рисунке 8.

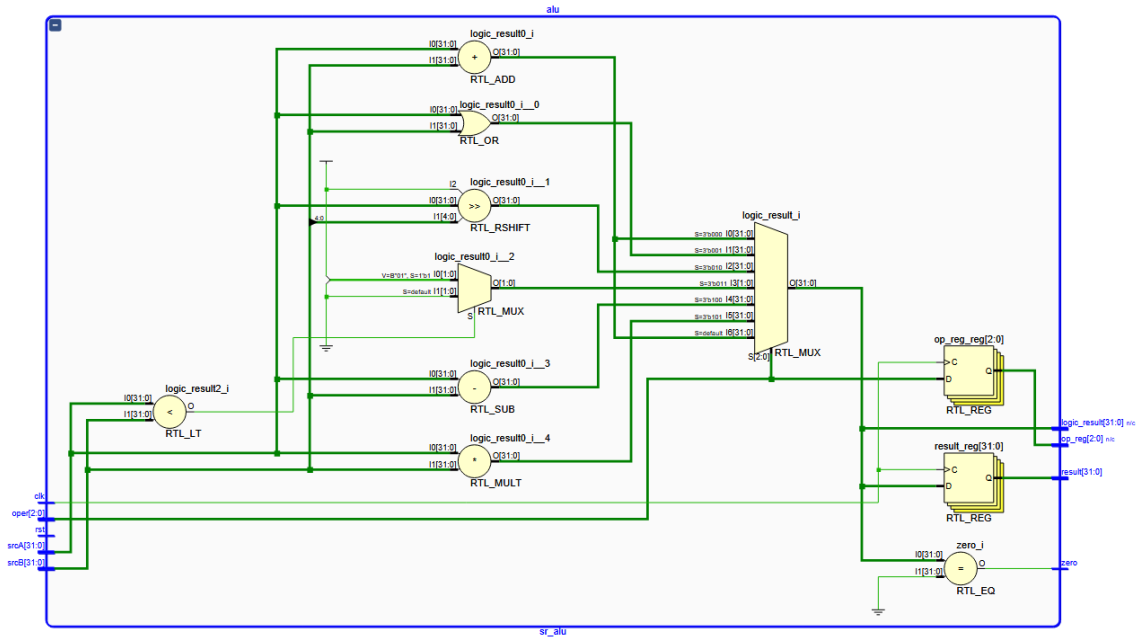

Рисунок 8. Дизайн АЛУ с конвейеризацией

Данный модуль выполняет арифметические и логические операции в данной реализации процессора.

Также модуль поддерживает операцию выдачи флага zero, когда выход АЛУ равен нулю. Данный функционал необходим для реализации условных переходов в процессоре. Реализация конвейеризации необходима для уменьшения комбинационного пути в блоке умножения, что позволит увеличить тактовую частоту дизайна. Соответственно данный модуль обладает латентностью, что необходимо учитывать при разработке дизайна.

3.3 Симуляция проекта цифрового синтезатора

Блок АЛУ выполняет арифметические и логические операции в данной реализации процессора.

Для проверки корректной работы цифрового синтезатора с функцией частотной модуляции было проведено два теста, генерация гармонической частоты и генерация ЛЧМ (линейно-частотной модуляции, где частота нарастает по линейному закону). На рисунке 9 представлен результат тестирования дизайна в режиме генерации ЛЧМ.

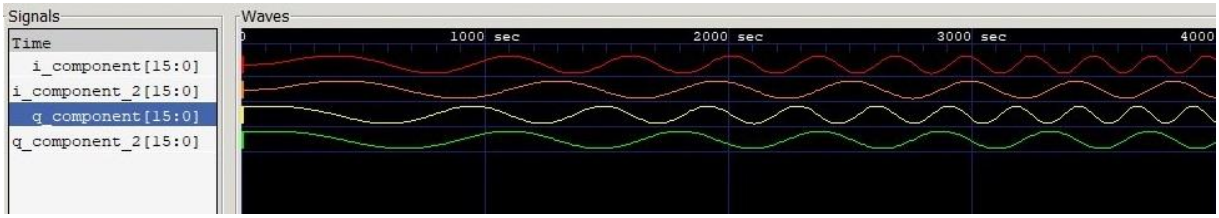

Рисунок 9. Симуляция дизайна в режиме ЛЧМ

Как видно из рисунка 34, частота генерируемого сигнала линейно нарастает, q_component и q_component_2 смещены по фазе относительно i_component и i_component_2 на 90 градусов, что является корректной работой блока цифрового вычислительного синтезатора. На рисунке 10 представлена генерация гармонического сигнала без нарастания частоты.

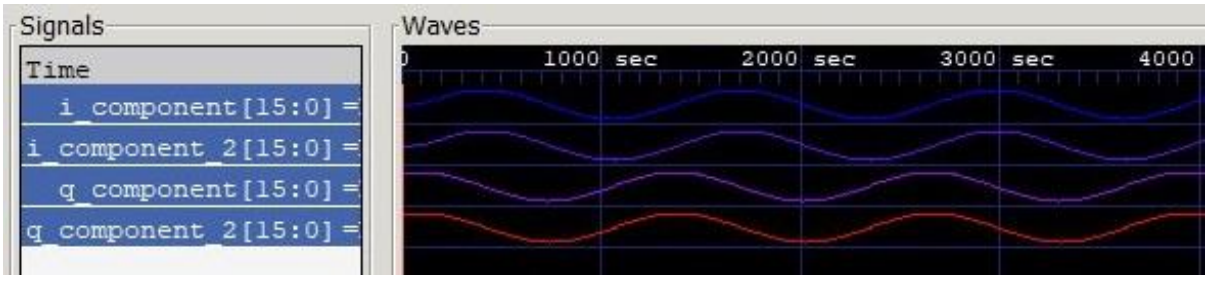

Рисунок 10. Симуляция дизайна в режиме генерации гармонического сигнала

Следующим шагом проверки данного дизайна была разработка программы для вычисления факториала.

Программа вычисляет факториал числа для проверки блока АЛУ и регистрового файла, а также, загружает в память значение аккумулятора фазы

для первой и второй DDS, проверяет работу GPIO блока, а также формирует ЛЧМ модуляцию на первой DDS.

В цикле loop происходит вычисление факториала, выгрузка значения аккумулятора фазы из DDS, инкремент на константу и запись в память. В блок GPIO записывается константа равная 511, и потом записывается 0 для формирования прямоугольных импульсов на блоке GPIO. Результаты симуляции приведены на рисунке 11.

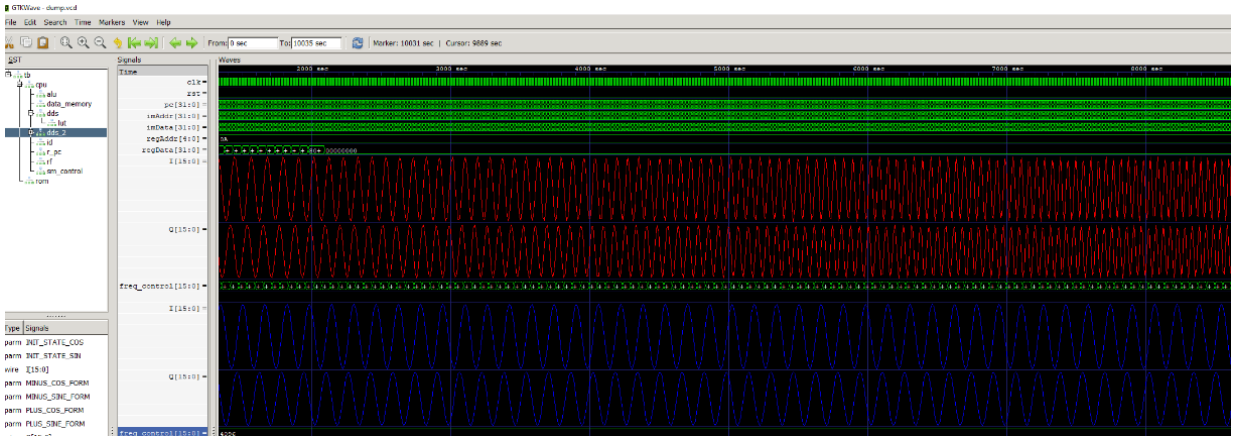

Рисунок 11. Проверка ЛЧМ модуляции на первой DDS

Как видно из рисунка, частота первой и второй DDS в начале симуляции является идентичной, в ходе симуляции частота первой DDS линейно увеличивается за счёт инкремента аккумулятора фазы на 64. В конце симуляции виден сильный прирост частоты относительно второй DDS.

На рисунке 12 приведена симуляция работы блока GPIO.

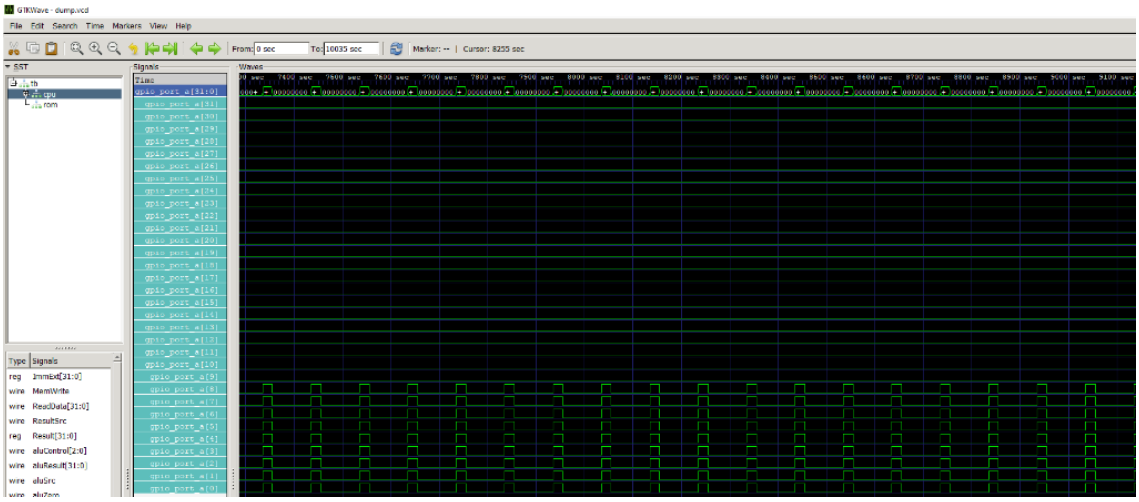

Рисунок 12. Формирование прямоугольных импульсов на блоке GPIO

Как видно из ассемблерной программы и рисунка сначала мы загружаем значение константы в регистр t6 регистрового файла, после загружаем константу в память, значение константы равно 511 которая равна 0b111111111 в двоичной системе счисления, тем самым переводя младшие 9 бит в состояние логической единицы. После этого мы записываем в регистр t6 регистрового фала значение равное нулю, и записываем в память блока GPIO значение регистра t6. Тем самым, мы формируем прямоугольные импульсы для проверки блока GPIO.

В тестовом модуле производится проверка на правильное вычисление значения факториала и на корректное извлечение значения инструкции по адресу.

Заключение. В ходе данной магистерской работы были выполнены следующие задачи:

1. Рассмотрены теоретические основы модуляции сигналов.

2. Изучены архитектурная и микроархитектурная реализации RISC-V процессора.

3. Реализован цифровой синтезатор, управляемый процессорным ядром архитектуры RISC-V, и провести симуляцию проекта.

На основе теоретического расчета, подтвержденного результатами экспериментального исследования, показано, что цифровой синтезатор частоты может быть реализован на языке описания аппаратуры высокого уровня, а также

продемонстрирована возможность управления им с помощью процессорного ядра, также реализованного на языке описания аппаратуры в объёме отдельной БИС программируемой логики.

Также показано, что данная реализация обладает преимуществом в быстродействии за счёт конвейерной реализации блока DDS и процессорного ядра, что позволяет использовать данную реализацию как в микросхемах FPGA, так и при производстве ASIC, что позволит получить наилучшие характеристики, как в перестройке частоты, так и в быстродействии данного дизайна.

Разработанный алгоритм цифрового синтезатора на ПЛИС может быть востребован для создания точных генераторов частоты для промышленности с возможностью оперативной программной перестройки частоты.

Список использованных источников:

1. Стешенко В.Б., Попова Т.В., Малашевич Д.Б. — Основы HDL Verilog как средства проектирования цифровых устройств: Уч. пос. / Под ред. А.И. Сухопарова. - М.: МИЭТ, 2006. -136 с

2. Сара Л. Харрис, Дэвид Харрис. Цифровая схемотехника и архитектура компьютера: RISC-V / пер. с англ. В. С. Яценкова, А.Ю. Романова; под ред. А. Ю. Романова. – М.: ДМК Пресс, 2021. – 810 с.

3. Хоровиц п., Хилл У. Искусство схемотехники: Пер. с англ. - Изд. 2-е. - М.: Издательство БИНОМ . - 2014. - 704 с.

4. Бойко, В.И. Схемотехника электронных систем. Аналоговые и импульсные устройства / В.И. Бойко, А.Н. Гуржий, В.Я. Жуйков [и др.]. - СПб.: БХВ-Петербург, 2004. - 496 с.

5. Балакай В.Г Интегральные схемы аналого-цифровых преобразователей /Балакай В.Г,Крюк И.П.,Лукьянов Л.М.; Под ред.Лукьянова Л.М. .-М: Энергия, 2008 .-257с.: Ил. .-Библиогрс.251-256.

6. Телеконтроль и телеуправление: учебное пособие /С.Н. Ливенцов, Ю.А. Чурсин; Томский политехнический университет – Томск: Изд-во Томского политехнического университета, 2011. – 139 с

7. Фрэнк Бруно. Программирование FPGA для начинающих / пер. с англ. С. Л. Плехановой; под науч. ред. А. Ю. Романова, Ю. В. Ревича. – М.: ДМК Пресс, 2022. – 304 с

8. Patterson D., and Waterman A., The RISC-V Reader: An Open Architecture Atlas, Strawberry Canyon, 2017.

9. Шустов, М.А Цифровая схемотехника. Практика применения / М.А Шустов. - СПб.: Наука и техника, 2018. - 432 с.

10. Китаев Ю. В. Основы цифровой техники. Учебное пособие. - СПб: СПбГУ ИТМО, 2007,- 87 с

11. Ашихмин, А.С. Цифровая схемотехника. Шаг за шагом / А.С. Ашихмин. - М.: Диалог-МИФИ, 2008. - 304 с.

12. Поляков А.К. Языки VHDL и VERILOG в проектировании цифровой аппаратуры. — М.: СОЛОН-Пресс, 2003. — 320 с.

13. Иванов М.Т. Теоретические основы радиотехники: Учеб. пособие для вузов / М.Т. Иванов, А.Б. Сергиенко, В.Н. Ушаков; Под ред. В.Н. Ушакова.-М.: Высш. шк., 2002.-306 с.

14. Хазанова С.В., Нежданов А.В., Николичев Д.Е. ИССЛЕДОВАНИЕ ПРОЦЕССА АМПЛИТУДНОЙ, ЧАСТОТНОЙ И ФАЗОВОЙ МОДУЛЯЦИИ: Практикум. Нижний Новгород: Нижегородский госуниверситет им. Н.И. Лобачевского, 2021. – 24с

15. Баскаков С.И. Радиотехнические цепи и сигналы / С.И. Баскаков. Москва: Высш. школа, 2003. - 462 с.

16. Waterman A.S. Design of the RISC-V instruction set architecture. Electrical Engineering and Computer Sciences, University of California at Berkeley, Technical Report No. UCB/EECS-2016-1, 2016.

17. Крсте Асанович, Александр Редькин и др. Технический симпозиум RISC-V Moscow, 2019

18. Исследование технологии RISC-V / В.А. Фролов, В.А. Галактионов, В.В. Санжаров // Труды ИСП РАН. 2020. Т. 32. Вып. 2. С. 81-98.

19. The RISC-V Instruction Set Manual. URL: https://riscv.org/wpcontent/uploads/2017/05/riscv-spec-v2.2.pdf

20. Угрюмов Е. П., Цифровая схемотехника. – СПб.: БХВ-Петербург, 2004. – 528 с.

21. Попова Т.В. Основы логического проектирования интегральных схем. - М.: МИЭТ, 2002. - 91 с.

22. Clifford E. Cummings. Correct Methods For Adding Delays To Verilog Behavioral Models. Sunburst Design, Inc 2001.

23. Антонов А. А., Барабанов А. В., Данчек Ч. Т., Жельнио С. Л., Иванец С. А., Кудрявцев И. А., Панчул Ю. В., Романов А. Ю., Романова И. И., Телятников А. А., Шуплецов М. С. Цифровой синтез: практический курс / под общ. ред. А. Ю. Романова, Ю. В. Панчула. – М.: ДМК Пресс, 2020. – 556 с

24. SystemVerilog IEEE Standard (IEEE STD 1800). IEEE-стандарт по SysemVerilog HDL; последнее обновление в 2012. URL: ieeexplore.ieee.org

25. Казеннов Г.Г.Основы проектирования интегральных схем и систем : Учеб. пособие / Г.Г. Казеннов. - М. : БИНОМ. Лаборатория знаний, 2005. - 296 с

26. Пухальский Г.И., Новосельцева Т.Я. Проектирование цифровых устройств. Учебное пособие. - СПб. : Лань, 2012. - 896 с.

27. Николаев В.Т. Синтез и расчет электрических схем электронных устройств автоматики. Учеб. пособие / В.Т. Николаев, С.В. Купцов, В.Н. Тикменов; Министерство образования и науки РФ, Национальный исследовательский университет "МИЭТ". - М. : МИЭТ, 2015. - 320 с

28. Степаненко И.П. Основы микроэлектроники. – М.: Сов.радио, 1980 – 424с

29. Кучумов А.И. Электроника и схемотехника. – М.: Гелиос АРВ, 2002. – 304с.

30. Медведев, Б. Л. Практическое пособие по цифровой схемотехнике / Б.Л. Медведев, Л.Г. Пирогов. - М.: Мир, 2019. - 408 с.