Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт
Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ _____
подпись инициалы, фамилия
« _____» _____ 20 ___ г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Специализированный процессор на ПЛИС
тема

Руководитель _____ доцент, к.т.н. А.И. Постников
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник _____ Д.Н. Щукина
подпись, дата инициалы, фамилия

Нормоконтролер _____ доцент, к.т.н. А.И. Постников
подпись, дата должность, ученая степень инициалы, фамилия

Красноярск 2020

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 4
1 Анализ задания на ВКР 5
1.1 Анализ предметной области 6
1.1.1 Принципы построения процессоров ЭВМ 6
1.1.2 Архитектура процессоров 7
1.1.3 Структура процессора 9
1.1.4 Операционный автомат 11
1.1.5 Управляющий автомат 17
1.2 Код и формат операндов 21
1.3 Система команд спецпроцессора 25
1.4 Выбор ПЛИС для реализации спецпроцессора 29
1.5 Результаты анализа 31
2 Проектирование структурной схемы спецпроцессора 32
2.1 Условное графическое обозначение спецпроцессора 32
2.2 Разработка операционного автомата спецпроцессора 33
2.2.1 Функциональная схема ОА 33
2.2.2 Блок формирования флажков 35
2.2.3 Элементы и узлы ОА 38
2.2.4 Осведомительные и управляющие сигналы 43
2.3 Разработка управляющего автомата 46
2.3.1 Определение требований к компонентам 46
2.3.2 Элементы и узлы УА 48
3 Реализация на ПЛИС 51
3.1 Операционный автомат 51
3.2 Управляющий автомат 57
3.3 Главный модуль 60
4 Проверка работы спецпроцессора 60
4.1 Разработка граф-схемы микропрограммы 61
4.2 Разработка микропрограммы 66

4.3 Результаты выполнения 68
4.4 Тестирование 70
4.5 Выводы по главе 72
ЗАКЛЮЧЕНИЕ 73
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 74

ВВЕДЕНИЕ

При изучении дисциплины «Прикладная теория цифровых автоматов», в качестве курсового проекта студентам необходимо разработать функциональную схему специализированного процессора, который предназначен для выполнения заданных арифметических операций [1].

В настоящее время проверка правильности разработанных в курсовом проекте функциональной схемы и микропрограммы проверяется преподавателем «вручную». Использование заявленной разработки в учебном процессе позволит сократить трудозатраты преподавателя на проверку, что должно привести к повышению качества обучения.

Целью данной работы является разработка на ПЛИС специализированного процессора, состоящего из операционного и управляющего автоматов, предназначенного для выполнения арифметических операций. Для достижения цели в работе решаются следующие **задачи**:

1. Выполнить анализ предметной области

    1.1. Рассмотреть принципы построения процессоров ЭВМ как сочетания операционного и управляющего автоматов.

    1.2. Выполнить анализ вариантов задания на курсовое проектирование с целью составления списка необходимых условий (флажков), в соответствии с которыми требуется выполнить ту или иную арифметическую операцию.

    1.3. Выполнить анализ вариантов задания на курсовое проектирование с целью составления системы команд спецпроцессора.

2. Разработать функциональную схему спецпроцессора.

3. Выполнить проектирование специализированного процессора на ПЛИС.

4. Реализовать спецпроцессор.

5. Выполнить тестирование разработки.

6. Разработать инструкцию пользователя.

1 Анализ задания на ВКР

В соответствии с заданием на ВКР необходимо разработать специализированный процессор на ПЛИС, состоящий из операционного и управляющего автоматов, для проверки специализированных процессоров, разрабатываемых студентами в рамках курсового проектирования по дисциплине «Прикладная теория цифровых автоматов».

Исходные данные:

1. Арифметические операции и условия, при которых они должны выполняться представлены в методических указаниях по курсовому проектированию [1].

2. Код чисел, поступающих в спецпроцессор – модифицированный дополнительный.

3. Код обработки – модифицированный дополнительный.

4. Код выдачи результата – модифицированный дополнительный.

5. Разрядность операндов – 8.

6. Формат представления чисел – с фиксированной точкой.

7. Масштабный коэффициент $К_А = 2^6$.

8. Тип управляющего автомата – микропрограммный автомат с программируемой логикой и принудительной адресацией.

Требуемые особенности реализации:

Для реализации спецпроцессора на ПЛИС использовать структурный метод для чего разработать библиотеку модулей элементов, узлов и устройств спецпроцессора.

1.1 Анализ предметной области

1.1.1 Принципы построения процессоров ЭВМ

Процессор – устройство, которое осуществляет обработку информации и управляет этим процессом. Процессор включает в себя арифметико-логическое устройство, блок управления и синхронизации, запоминающее устройство, регистры и другие блоки, необходимые для выполнения вычислительного процесса. Реализует функции выборки, декодирования, управления выполнением команд, а также выполнения операций тестирования и преобразования данных. В итоге, он организует и отчасти выполняет заданную последовательность действий – процесс, откуда и название – процессор [2].

Классификация процессоров по наиболее существенным характеристикам служит основой для выбора эффективной области применения того или иного типа процессора. Классифицировать процессоры можно по разрядности, по способу управления, по назначению и т.д.

По назначению процессоры делятся на три группы: универсальные, проблемно-ориентированные и специализированные.

К универсальным относят процессоры, имеющие широкое применение в различных областях при выполнении самых разных задач. При этом их эффективная производительность слабо зависит от проблемной специфики решаемых задач.

Проблемно-ориентированные процессоры служат для решения более узкого круга задач.

Специализированные предназначены для решения узкого круга задач, а также для реализации строго определенной группы функций. Узкая ориентация позволяет четко специализировать их структуру и существенно снизить время выполнения некоторых операций.

1.1.2 Архитектура процессоров

Существуют процессоры различной архитектуры.

CISC (*Complicated Instruction Set Computer* – компьютер с полным набором команд) – архитектура процессора с нефиксированной длиной команд, с кодированием арифметических действий в одной команде и небольшим числом регистров, многие из которых выполняют строго определенную функцию [3]. Внутри процессора содержится автомат, который для выполнения команды заставляет все внутренние схемы осуществлять действия в определённой последовательности. Эта последовательность действий называется микропрограммой, а автомат, её реализующий – микропрограммным. Реализация большого количества команд приводит к усложнению устройства управления – микропрограммного автомата, а также для реализации некоторых команд используются очень сложные микропрограммы, выполняющиеся за большое количество тактов.

Для концепции *CISC* характерны свойства:

- нефиксированное значение длины команды;
- арифметические действия кодируются в одной команде;
- широкий набор способов адресации;
- небольшое число регистров, каждый из которых выполняет строго определённую функцию.

Архитектура *CISC* процессоров стимулирует создание большой универсальной системы команд, многочисленных режимов адресации памяти и эффективных механизмов вызова подпрограмм, что позволяет повысить плотность кода и гибкость программирования.

Недостатками архитектуры *CISC* являются:

- высокая сложность и, следовательно, стоимость аппаратной части;
- неоднородность структуры машинных инструкций и сильно отличающаяся длина;

- расшифровка сложных инструкций часто работает медленнее, чем точно такие же цепочки команд, встречающиеся в основной программе;
- сложности с распараллеливанием вычислений [2].

В результате все *CISC*-процессоры оказались очень трудоемкими в проектировании и изготовлении, оказалось трудно наращивать их тактовую частоту, а зашитые в память процессора расшифровки сложных инструкций часто работают медленнее. Итак, стало очевидным, что *CISC*-процессоры нужно упрощать.

С точки зрения экономии затрат целесообразно создание процессора, реализующего небольшой набор наиболее часто используемых команд с максимально возможной скоростью. А выполнение сложных действий может быть реализовано на программном уровне в виде подпрограмм.

RISC (*Reduced Instruction Set Computer*– компьютер с сокращенным набором команд) – архитектура процессора, где быстродействие увеличивается за счет упрощения инструкций с константной длиной команды.

В набор команд *RISC*-архитектуры вошли только основные элементарные микрооперации, что позволило унифицировать формат команд вычислительного ядра, упростить конструкцию и снизить стоимость изготовления вычислительных ядер [3].

Уменьшение набора машинных команд в *RISC*-архитектуре позволило разместить на кристалле вычислительного ядра большое количество регистров общего назначения, за счет чего были минимизированы обращения к медленной оперативной памяти, оставив для работы с *RAM* только операции чтения данных из оперативной памяти в регистр и запись данных из регистра в оперативную память, остальные машинные команды используют в качестве операндов регистры общего назначения.

В *RISC*-процессорах такой архитектуры, в которой выполнение любой команды занимало бы не более одного такта синхронизации, в этом случае для выполнения команд не требуется сложное устройство управления. Для реализации этой идеи *RISC*-процессоры строятся по трёхадресной схеме и

поэтому в своём составе содержат большое количество регистров общего назначения, несколько исполнительных устройств, которые реализуют большинство функций на аппаратном уровне или за минимальное количество тактов.

Особенности, характерные для *RISC*-процессоров:

- фиксированная длина машинных инструкций, простой формат команды;
- равное время выполнения всех команд;
- специализированные команды для операций с памятью – чтения или записи;
- большое количество регистров общего назначения. Регистры – основное достоинство *RISC*;
- отсутствие поддержки операций вида "изменить" над укороченными типами данных – байт, 16-битное слово;
- отсутствие микропрограмм внутри самого процессора;
- спекулятивное исполнение. При встрече с командой условного перехода процессор исполняет сразу обе ветви до тех пор, пока не окончится вычисление управляющего выражения перехода;
- переименование регистров. Каждый регистр процессора на самом деле представляет собой несколько параллельных регистров, хранящих несколько версий значения [2].

1.1.3 Структура процессора

В структурно-функциональном отношении процессор можно разделить на две части – операционный и управляющий автоматы (рис. 1.1). Операционный и управляющий автоматы могут быть заданы своими функциями или перечнем выполняемых ими действий, на основании которых строятся схемы автоматов.

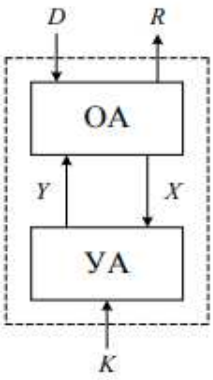

Рисунок 1.1 – Структура процессора

Обычный цикл работы процессора выглядит так: он читает первую команду из памяти, декодирует ее для определения ее типа и операндов, выполняет команду, затем считывает, декодирует и выполняет последующие команды. Таким образом, осуществляется выполнение программ.

Построение процессоров основано на принципе микропрограммного управления, согласно которому:

– каждая операция, которая реализуется процессором, рассматривается как сложное действие. Это действие делится на последовательные элементарные действия – микрооперации;

– для управления порядком их следования используются логические условия, с помощью которых определяется состояние процессора после выполненных микроопераций в виде логического 0 или 1;

– процесс выполнения операций описывается в виде алгоритма и называется микропрограммой;

– микропрограмма используется как форма представления функции процессора, на основе которой определяются его структура и порядок функционирования во времени.

1.1.4 Операционный автомат

Операционный автомат – набор функциональных элементов, таких как АЛУ, которые выполняют обработку данных. Любая команда, операция, процедура, выполняемая в операционном автомате, описывается микропрограммой и реализуется за несколько тактов, в каждом из которых выполняется одна или несколько микроопераций [4].

Как и любой структурный автомат, операционный автомат представляется в виде композиции комбинационных схем и памяти (рис. 1.2).

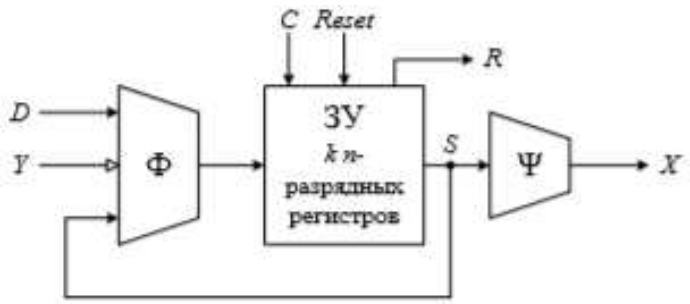

Рисунок 1.2 – Структура операционного автомата

На входы комбинационной схемы *Ф* поступают входные слова и управляющие сигналы, формируемые управляющим автоматом и инициирующие соответствующие микрооперации обработки информации *D*. Начальные значения обрабатываемых слов, помещаются в память операционного автомата, состоящую из *k* *n*-разрядных регистров. Выходы регистров формируют множество промежуточных результатов *S*, поступающих в *Ф* в качестве сигналов обратной связи. Под воздействием микроопераций и значений результатов схема *Ф* формирует новые значения слов, которые заносятся в память по синхросигналу *C*. В начальный момент времени $t = 0$ все регистры памяти переводятся в нулевое состояние сигналом *Reset*. Содержимое памяти поступает в комбинационную схему *Ψ*, формирующую значения логических условий, являющихся осведомительными сигналами.

Таким образом комбинационная схема Φ предназначена для выработки промежуточных (S) и окончательных (R) результатов работы ОА, а схема Ψ – для формирования логических осведомительных сигналов, которые позволяют УА определить адрес следующей микрокоманды.

В процессе разработки был определен набор элементов, используемый при проектировании операционных автоматов. Каждый элемент выполняет определенную обработку информации. Сигналы, которые инициируют ее стандартными элементами, называются типовыми микрооперациями, которые можно поделить на классы:

– микрооперация установки, в результате выполнения которой регистру присваивается значение константы;

– микрооперация передачи, которая инициирует передачу информации между регистрами;

– микрооперация инвертирования, переводящая 0 в 1 и 1 в 0, после которой значение слова меняется на инверсное;

– микрооперации сдвига, которые предназначены для изменения положений разряда слова по отношению к первоначальному;

– микрооперации счёта, в результате выполнения которых происходит изменение значения слова на единицу;

– логические микрооперации – присваивают слову значение, получаемое при поразрядном выполнении булевых операций дизъюнкции, конъюнкции и сложения по модулю 2;

– микрооперация сложения, присваивающая слову значение суммы слагаемых;

– комбинированные микрооперации. Каждая из комбинированных микроопераций содержит несколько действий, присущих микрооперациям классов, перечисленных выше.

Микрооперации, которые используются в микропрограмме, могут выполнятся последовательно и параллельно. Микрооперации, которые могут

выполнятся параллельно, называются совместимыми. Функциональная совместимость – совместимость, обусловленная содержанием операторов, структурная совместимость обусловлена ограничениями структуры ОА [5].

В микропрограммах встречаются микрооперации вычисляющие значения слов с использованием одной функции, применяемой к различным наборам аргументов.

Например, микрооперации

$$RgD := RgA + RgB \tag{1.1}$$

$$RgE := RgA + \overline{RgC} + 1 \tag{1.2}$$

считаются эквивалентными, т.к. в них содержится одна и та же функция сложения над различными словами. Эквивалентность микроопераций означает, что для вычисления результатов, соответствующих, микрооперациям, может использоваться одна и та же комбинационная схема (рис.1.3).

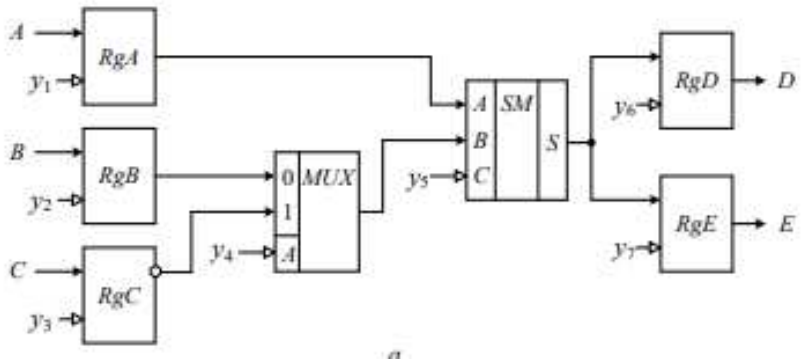

Рисунок 1.3 – Реализация эквивалентных микроопераций на одном сумматоре

Использование одной комбинационной схемы для выполнения нескольких микроопераций исключает совместимость этих микроопераций. Так, микрооперации функционально совместимые могут выполняться только в разных тактах. Таким образом, время выполнения этой пары операций

увеличится из-за структурных ограничений, т. е. экономия оборудования, как правило, влечёт за собой увеличение времени выполнения микропрограммы.

Для построения структуры, реализующей совокупность эквивалентных микроопераций, используется обобщённый оператор.

Например, эквивалентным микрооперациям (1.1) и (1.2) соответствует обобщённый оператор

$$RgS = RgA + Q_1 + Q_2 \tag{1.3}$$

где

$$Q_1 = \begin{cases} RgB \text{ при } y_4 = 0; \\ \overline{RgC} \text{ при } y_4 = 1; \end{cases} \quad Q_2 = \begin{cases} 0 \text{ при } y_5 = 0; \\ 1 \text{ при } y_5 = 1. \end{cases}$$

Обобщённому оператору (1.3) соответствует структура, представленная на рисунке 1.4.

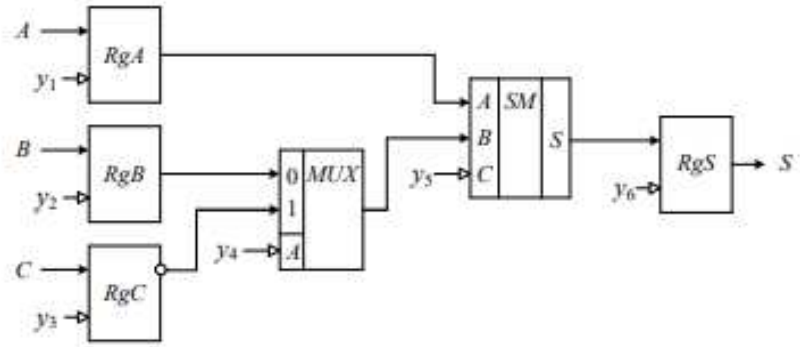

Рисунок 1.4 – Структура, реализующая обобщенный оператор

Аргументы функции интерпретируются входами комбинационной схемы. Q1 – имя шины, по которой содержимое регистра RgB, либо инвертированное содержимое регистра RgC поступает на вход комбинационной схемы сумматора, а Q2 определяет значение разряда, прибавляемого к младшему разряду суммы.

Для выбора наиболее подходящего операционного автомата, используются некоторые характеристики: производительность, быстродействие, затраты оборудования, регулярность, универсальность.

Структуру операционного автомата можно синтезировать непосредственно по его функции заданной:

– множеством слов;

– множеством микроопераций над словами;

– множеством логических условий.

Структуры операционных автоматов бывают:

- Каноническая. Каноническая структура ОА – структура, полученная путём замены каждого элемента функции соответствующими элементами структурного базиса (регистрами, комбинационными схемами, шинами).

- I – автомат. I – "Individual" – для вычислений значений каждого слова используется индивидуальная комбинационная схема, средствами которой параллельно реализуются совместимые микрооперации, вычисляющие новые значения слов. Затраты оборудования можно уменьшить, если в одной комбинационной схеме использовать только один вычисляющий элемент – сумматор, сдвигатель и т.д. используемый для выполнения эквивалентных микроопераций. Это приводит к появлению мультиплексоров, подсоединяющих к одному операционному элементу различные слова.

- M – автомат. M – означает "Mutual" – для вычисления значения любого слова используется одна общая комбинационная схема равнодоступная по отношению к регистрам. В структуре I-автомата могут содержаться эквивалентные по своим функциям комбинационные схемы, используемые для обслуживания разных регистров. Если операционные ресурсы (сумматоры, сдвигатели и т.д.) сделать общими для всех регистров, то можно свести до минимума число операционных элементов. Таким образом комбинационная схема будет включать один сумматор, один сдвигатель и т.д.

- IM – автомат. Автомат, характеристики которого занимают промежуточное значение по сравнению I- и M-автоматами. Структура IM-автоматов вносит ограничения на совместимость микроопераций и одновременно с этим обеспечивает выполнение за один такт более одной операции функциональной микропрограммы. Это позволяет повысить производительность и сохранить низкие аппаратурные затраты.

IM – автоматы можно разделить по способу организации комбинационной схемы.

IM – автоматы с параллельной комбинационной частью (IMp – автоматы) включают комбинационные схемы Ф1 для выполнения двухместных микроопераций (конъюнкция, дизъюнкция, сумма по модулю 2 и др.) и Ф2 для выполнения одноместных, унарных микроопераций (передача, инверсия, сдвиг, формирование констант и т.п.), а также схемы Ψ1 и Ψ2 для вычисления логических условий.

IM – автоматы с последовательной комбинационной частью (IMs – автоматы) позволяют выполнять последовательно за один такт три микрооперации, включают три комбинационные схемы и одну схему для вычисления логических условий. Комбинационная схема Ф1 – формирователь кодов, используется для формирования констант и кодов чисел, Ф2 – сумматор, выполняет бинарные операции, Ф3 – сдвигатель, выполняет функцию сдвига.

- S – автомат. ОА, у которого в качестве памяти используется ЗУ с произвольным доступом, называется S-автоматом. В некоторых операционных устройствах однотипные операции выполняются над большим числом слов. Примером такого устройства является специализированный процессор ввода-вывода. Для уменьшения аппаратурных затрат таких устройств регистровая память ОА заменяется запоминающим устройством.

1.1.5 Управляющий автомат

Управляющий автомат – часть микропроцессора, выполняющая управляющие функции над данными. Управляющий автомат вырабатывает сигналы управления по заданной программе и с учетом значений внутренних и внешних логических условий, которые для него являются входными переменными [6].

Построение УА производится на основе принципа программного управления, который сводится к упорядоченной выработке сигналов-команд, выполнение которых приводит к достижению заданной цели. Микрокоманда – управляющий сигнал, осуществляющий выполнение одного элементарного шага. Она является основным элементом программного управления. Микрокоманда содержит информацию о микрооперациях, которые должны выполняться в данном такте работы устройства, а также информацию об адресе следующей микрокоманды. Совокупность микрокоманд образует массив – микропрограмму.

В зависимости от способа определения адреса очередной микрокоманды различают УА с принудительным и естественным порядком следования микрокоманд.

Принудительная адресация микрокоманд заключается в том, что в каждой микрокоманде указываются все возможные адреса следующих микрокоманд. Адрес следующей микрокоманды может задаваться безусловно (независимо от значений осведомительных сигналов), или выбираться в зависимости от условия, определяемого текущими значениями осведомительных сигналов.

В разветвляющейся ГСМ содержатся условные вершины, и в зависимости от значений опрашиваемых в них осведомительных сигналов X адрес следующей микрокоманды выбирается из двух или более возможных адресов.

Если, например, в каждой микрокоманде опрашивается только один осведомительный сигнал x_i, формат микрокоманды будет выглядеть, как представлено на рисунке 1.5.

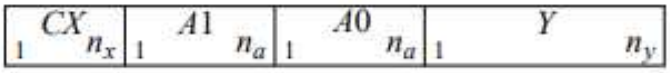

Рисунок 1.5 – Формат микрокоманды УА с принудительной адресацией для разветвляющейся ГСМ.

Микрокоманда содержит в себе четыре поля: СХ – поле управления выбором опрашиваемого входного осведомительного сигнала x_i; А1 – поле адреса следующей микрокоманды, если опрашиваемый осведомительный сигнал $x_i = 1$; А0 – поле адреса следующей микрокоманды, если опрашиваемый осведомительный сигнал $x_i = 0$; Y – поле выходных управляющих сигналов [2].

Структура УА с принудительной адресацией для разветвляющейся ГСМ приведена на рисунке 1.6.

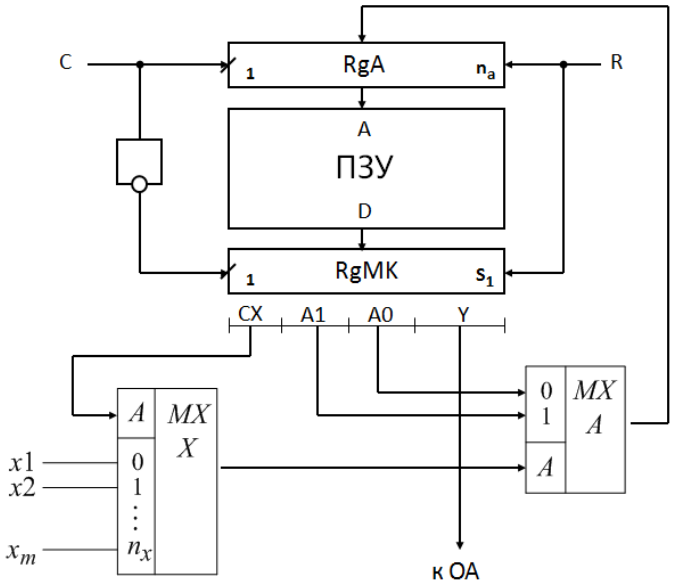

Рисунок 1.6 – Структурная схема УА с принудительной адресацией для разветвляющейся ГСМ

Кроме регистра адреса RgA разрядности n_a, регистра микрокоманды RgMK разрядности $S_1 = n_x + 2 * n_a + n_y$ и ЗУ организации $2^{na} \times S_1$ структура УА содержит два мультиплексора: MXX – мультиплексор выбора, опрашиваемого

входного осведомительного сигнала. МХА – мультиплексор выбора поля адреса следующей микрокоманды.

Запись информации в регистры RgA и RgMK осуществляется по нарастающему фронту синхросигнала. При подаче сигнала сброса R осуществляется принудительная установка УА в начальное состояние.

При естественной адресации адрес следующей микрокоманды принимается равным адресу на 1 больше адреса текущей микрокоманды. Если микрокоманды следуют в естественном порядке, то процесс адресации может выполнять обычный счетчик адреса, состояние которого увеличивается на единицу после чтения очередной микрокоманды. При наличии в ГСМ условных вершин возникает необходимость в переходе к микрокоманде с адресом $A1 \neq A_i + 1$. Переход может быть либо безусловным, либо зависеть от текущего значения опрашиваемого осведомительного сигнала x_i. Для реализации условных переходов микрокоманды кроме поля выходных сигналов Y имеются еще два поля – поле управления выбором осведомительного сигнала x_i (поле CX) и поле адреса следующей микрокоманды (поле A1). Формат микрокоманды представлен на рисунке 1.7.

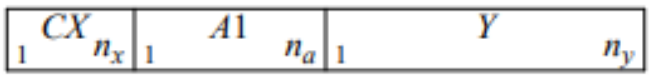

Рисунок 1.7 – Формат микрокоманды УА с естественной адресацией для разветвляющейся ГСМ.

Структура УА с естественной адресацией для разветвляющейся граф-схемы микропрограммы представлена на рисунке 1.8.

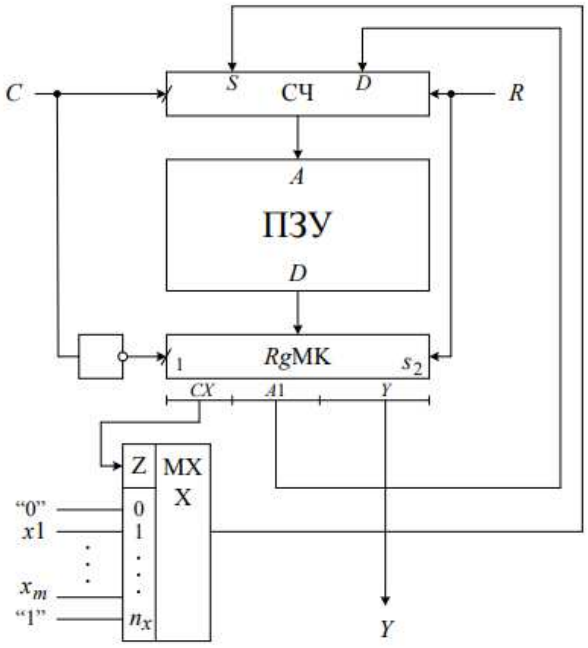

Рисунок 1.8 – Структурная схема УА с естественной адресацией для разветвляющейся ГСМ

Схема управляющего автомата содержит регистр микрокоманд RgMK, запоминающее устройство ЗУ, регистр-счетчик адреса микрокоманд СЧ, который допускает параллельную запись информации, и мультиплексор выбора входного сигнала MXX. При поступлении единицы на вход S счетчика в него записывается адрес, находящийся на параллельных входах D. Увеличение адреса, содержащегося в СЧ, производится по синхросигналу C. На информационные входы мультиплексора входных сигналов MXX кроме опрашиваемых осведомительных сигналов X поступают жестко заданные сигналы – 0 и 1. Если на MXX выбран 0, то с выхода MXX он поступает на вход S счетчика и, следовательно, будет осуществлен безусловный переход по счетчику на следующую микрокоманду. Если выбрана 1, то она поступит на вход S счетчика и в него будет записан адрес следующей микрокоманды из поля A1 текущей микрокоманды. Таким образом будет осуществлен безусловный переход на адрес из поля A1 микрокоманды [2].

Применяемые в микрокомандах варианты кодирования сигналов управления можно свести к трем группам: горизонтальное, вертикальное и смешанное.

При горизонтальном каждой микрооперации ставится в соответствие разряд поля микроопераций микрокомандного слова. В этом случае количество разрядов поля микроопераций равно числу различных микроопераций, вырабатываемых УА. При вертикальном кодировании в поле микроопераций помещается номер выполняемой микрооперации. При этом количество разрядов N, которое следует предусмотреть в поле микроопераций, определяется выражением: $N=k \geq \log_2 n$.

Для того, чтобы исключить дешифровку микрокоманды выбираем горизонтальное кодирование.

1.2 Код и формат операндов

Чем больше различных операций выполняет операционный автомат, тем больше у него различных элементов. Применение специальных кодов дает возможность заменить операции вычитания, умножения и деления одним действием – сложением. Дополнительный код является наиболее распространенным способом представления отрицательных целых чисел. Он позволяет заменить операцию вычитания операцией сложения.

Дополнительный код отрицательного числа можно получить инвертированием разрядов двоичного положительного числа и прибавлением к результату единицы. Положительное число из отрицательного в дополнительном коде получается точно таким же образом. В данной работе используются операции сложения, вычитания, умножения и деления чисел. Пользуясь возможностями дополнительного кода, операция вычитания заменяется операцией сложения с отрицательным числом. Анализ операций (табл. 1 [10]), которые должен выполнять спецпроцессор, показал, что операции

умножения и деления целесообразно выполнять с помощью сдвига числа на один разряд в сторону старших и младших разрядов соответственно.

Модифицированный код числа служит для упрощения процесса обнаружения переполнения разрядной сетки. От отличается от простого кода тем, что в него вводится вспомогательный разряд в знаковую часть, который называют разрядом переполнения.

Для модифицированного кода переполнение определяется следующим образом:

– если $Sg1 \oplus Sg2 = 1$ – произошло переполнение разрядной сетки;

– если $Sg1 \oplus Sg2 = 0$ – переполнения разрядной сетки нет.

Другими словами, при переполнении $Sg1 \neq Sg2$ ($Sg1 = 0$, а $Sg2 = 1$ или $Sg1 = 1$, а $Sg2 = 0$).

Представление чисел с фиксированной точкой (ФТ) характеризуется тем, что положение разрядов в машинном изображении остается всегда постоянным независимо от величины самого числа. Формат представления числа с ФТ имеет два поля: поле числа и поле знака. Если число положительное или равно нулю – в поле знака записывается 0, если число отрицательное – в поле знака записывается 1. Преимущества формата с ФТ – быстродействие ЭВМ при корректном проектировании и простота проектирования.

Таблица 1 – Варианты заданий

Операции	Условия
0,25A + 2B 1,5A A - 0,5B	A < 0, B < 0, A – кратно 4, |B|< 0,5 A < 0, B > 0 A > 0, B – четное
4A – B (2A + B) / 2 3B	A < 0, B > 0, $\|A\|$< 0,25 A < 0, B < 0, $\|A\|$< 0,5 A > 0, $\|B\|$< 0,5
0,5A + B 2,5A 4B – A	A > 0, B < 0, A – четное $SgA = SgB$, $\|A\|$< 0,5, A – четное A < 0, B > 0, $\|B\|$< 0,25
A + 0,25B 3,5B 2A – B	A > 0, B > 0, A – кратно 4 A > 0, B < 0, |B|< 0,5, B – четное A < 0, $\|A\|$< 0,5

Продолжение таблицы 1 – Варианты заданий

Операции	Условия
$\lvert A + B \rvert$ A + 0,5B 2A – 0,5B	A > 0, B < 0 A < 0, B > 0, B – четное $SgA = SgB$, $\lvert A \rvert < 0{,}5$, B – четное
2A – 0,25B 1,25A A + 4B	A> 0, B> 0, $\lvert A \rvert < 0{,}5$, B– кратно 4 A > 0, B < 0 A < 0, $\lvert B \rvert < 0{,}25$
0,75B (A + B) / 2 2A – B	A < 0, B– кратно 4 A > 0, B > 0 A > 0, B < 0, $\lvert A \rvert < 0{,}5$
2A + 2B 1,75A $\lvert A - B \rvert$	A < 0, B < 0, $\lvert A \rvert < 0{,}5$,$\lvert B \rvert < 0{,}5$ A > 0, B > 0,A – кратно 4 $SgA \neq SgB$
A – 2B 0,5B + $\lvert A \rvert$ 2,25A	A > 0, B < 0 A > 0, B > 0, B – четное A < 0, $\lvert A \rvert < 0{,}5$, A – кратно 4
$\lvert 2A - 0{,}5B \rvert$ 0,5A + 2B 0,75A	A < 0, B > 0, $\lvert A \rvert < 0{,}5$, B – четное A > 0, B > 0, $\lvert B \rvert < 0{,}5$, A – четное B < 0, A – кратно 4
0,25A + 2B 4A – B 2,5A	A > 0, B < 0, A – кратно 4, $\lvert B \rvert < 0{,}5$ A < 0, B > 0, $\lvert A \rvert < 0{,}25$ $SgA = SgB$, $\lvert A \rvert < 0{,}5$, A – четное
0,5A + B 1,25A – 0,5B 3B	A > 0, B < 0, A – четное A > 0, B > 0, B – четное A < 0, $\lvert B \rvert < 0{,}5$
(2A + B) / 2 4B – A 1,5A	A < 0, B < 0, $\lvert A \rvert < 0{,}5$ A < 0, B > 0, $\lvert B \rvert < 0{,}25$ A > 0
A + 2B 2A – 0,25B 2,25A	A < 0, B > 0, $\lvert B \rvert < 0{,}5$ A> 0, B< 0, $\lvert A \rvert < 0{,}5$, B– кратно 4 $SgA = SgB$, $\lvert A \rvert < 0{,}5$, B– кратно 4
A + 4B 1,25A – 0,5B 3,5B	A < 0, B < 0, $\lvert B \rvert < 0{,}25$ A < 0, B > 0, B – четное A > 0, $\lvert B \rvert < 0{,}5$, B – четное
2A + B $\lvert A + 0{,}5B \rvert$ 1,25A	A > 0, $\lvert A \rvert < 0{,}5$ A < 0, B > 0, B – четное A < 0, B < 0, A – кратно 4
1,25A + 1,75B 0,5A – B 2,25A	A < 0, B < 0, $\lvert A \rvert < 0{,}5$, $\lvert B \rvert < 0{,}5$ A > 0, A – четное A < 0, B > 0, $\lvert A \rvert < 0{,}5$, A – кратно 4
0,5B + 0,25A -0,5*$\lvert 2A - B \rvert$ 0,75B	A> 0, B> 0,A– кратно 4, B – четное $SgA \neq SgB$, $\lvert A \rvert < 0{,}5$ A < 0, B < 0, B– кратно 4
(0,5A + B) / 2 1,75A A – 1,5B	A> 0, B> 0, A– четное A< 0, A– четное A> 0, B< 0
2(A + 1,25B) 1,5A + 0,5B $\lvert A \rvert$ - $\lvert B \rvert$	A < 0, B < 0 A> 0, A – четное, B – четное A < 0, B > 0

Окончание таблицы 1 – Варианты заданий

Операции	Условия
2,25A – 0,5B 1,75B 2,5A – B	A < 0, $\lvert A\rvert$< 0,5, B – четное A> 0, B< 0, B– кратно 4 A> 0, B> 0,$\lvert A\rvert$< 0,5, A– четное
1,75A + 1,5B A – 0,5B 2,25A	A> 0, B> 0, B – четное A> 0, B< 0, B – нечетное A < 0, $\lvert A\rvert$< 0,5, A– кратно 4
0,5A + B 2,75A – 0,25B 1,5A	A> 0, B< 0, A– четное A> 0, B> 0, $\lvert A\rvert$< 0,5, B– кратно 4 A < 0
2A + 1,25B 4B – A 0,75B	A < 0, B < 0 A < 0, B > 0, $\lvert B\rvert$< 0,25 A > 0, B– кратно 4
2A – 0,5B 1,25A 2A + 2B	$SgA = SgB$, $\lvert A\rvert$< 0,5, B – четное A> 0, B< 0 A< 0, B> 0,$\lvert A\rvert$< 0,5,$\lvert B\rvert$< 0,5
0,5B + |A| A + 0,5B 3B	A< 0, B> 0, B – четное A> 0, B – четное A< 0, B< 0,$\lvert B\rvert$< 0,5
A + 4B 2A – B 3,5B	A< 0, B< 0, $\lvert B\rvert$< 0,25 A< 0, B> 0, $\lvert A\rvert$< 0,5 A > 0, $\lvert B\rvert$< 0,5, B – четное
0,5A + 2B 1,75A + 0,25B 1,75B	A> 0, B> 0, A– четное, $\lvert B\rvert$< 0,5 A < 0 A> 0, B< 0, B– кратно 4
1,5A + 0,5B 2A – B 0,75A	A< 0, $\lvert A\rvert$< 0,5, $\lvert B\rvert$ – четное A> 0, B< 0, $\lvert A\rvert$< 0,5,A– четное A > 0, B > 0
|2A| – 0,5B 0,25A – B 2,5A	A < 0, B > 0, $\lvert A\rvert$< 0,5, B – четное A > 0, A– кратно 4 A < 0, B < 0, $\lvert A\rvert$< 0,5

Проанализировав варианты задания, представленные в таблице 1, был составлен список необходимых условий (флажков), в соответствии с которыми требуется выполнить ту или иную арифметическую операцию.

Список флажков для числа A:

1. Знак: A < 0, A > 0
2. Четность
3. Кратность 4
4. $\lvert A\rvert$< 0,5
5. $\lvert A\rvert$< 0,25

Список флажков для числа B:

1. Знак: B <0, B > 0
2. Четность
3. Кратность 4
4. $\lvert B\rvert$< 0,5
5. $\lvert B\rvert$< 0,25

Таблица 2 – Проверка условий выполнения операций

Условия	Условия разрядов
A<0 B<0	11.******
A > 0 B> 0	00.******, а также есть хотя бы одна единица
A – четное B– четное	**.*****0
A –кратно 4 B–кратно 4	**.****00
$\lvert A\rvert < 0{,}5$ $\lvert B\rvert < 0{,}5$	11.1*****, а также есть хотя бы одна единица или 00.0*****
$\lvert A\rvert < 0{,}25$ $\lvert B\rvert < 0{,}25$	11.11****, а также есть хотя бы одна единица или 00.00****
$SgA=SgB$	Знаковые разряды числа A равны знаковым разрядам числа B
$SgA \neq SgB$	Знаковые разряды числа A неравны знаковым разрядам числа B

Общее количество флажков равно 10, из чего следует, что потребуется десятиразрядный регистр флагов.

Также необходимы флажки: знак результата, который будет храниться в знаковом разряде RgS1 (рис. 2.5) и признак переполнения разрядной сетки, формирующийся на основе разрядов RgS2 (рис. 2.5).

1.3 Система команд спецпроцессора

Набор команд для универсальных процессоров можно разделить на несколько групп:

1. Команды пересылки данных, осуществляющие перенос данных между регистрами или между регистрами и памятью.

2. Арифметические команды, включающие команды сложения, вычитания, увеличения или уменьшения на единицу данных в регистрах или памяти.

3. Логические команды, позволяющие осуществить логические операции, поразрядное сложение по модулю 2, сравнение и сдвиг содержимого регистров или памяти.

4. Команды передачи управления, обеспечивающие безусловную или передачу управления по условию, а также вызов или возврат из подпрограммы.

5. Команды управления и работы со стеком, организующие ввод-вывод данных из МПС, доступ к стеку и внутреннему регистру признаков МП [6].

Любая команда состоит из двух частей – операционной и адресной. Операционная часть показывает действия из списка допустимых, которые нужно выполнить с данными. Адресная часть хранит адрес ячейки памяти следующей микрокоманды.

В зависимости от количества операндов, команды бывают:

1. Одноадресные. При выполнении одноадресной команды используется один адрес, по которому до выполнения операции находится операнд, а после выполнения операции записывается результат (например, увеличить или уменьшить значение операнда на 1).

2. Двухадресные. При выполнении операции с двумя операндами до выполнения операции по первому адресу находится первый операнд, а после выполнения операции записывается результат (например, сложить два операнда и записать результат на место одного из операндов, вычесть из первого операнда второй, сравнить значения двух операндов).

3. Трехадресные. С помощью трехадресных команд выполняются арифметические и логические операции с двумя операндами. При этом используется три адреса. По первому адресу записывается результат операции, по второму и третьему - первый и второй операнд соответственно (например, сложить два операнда, а результат записать в третий операнд) [7].

Выполнив анализ вариантов задания на курсовое проектирование, был сделан вывод, что необходимо использовать одноадресные и трехадресные команды, так как в задании присутствуют операции и над одним операндом, и над двумя.

Одноадресные команды, предназначены для выполнения унарных операций, выполняющихся над одним операндом: сдвиг, поразрядное инвертирование.

Трехадресные команды, предназначены для выполнения бинарных операций, выполняющихся над двумя операндами: сложение, вычитание, умножение, деление.

При реализации операции сдвига необходимо вдвигать на место освободившегося разряда двоичный разряд. Рассмотрим сдвиги числа в дополнительном коде.

Если осуществляется сдвиг в сторону младших разрядов, то вдвигается старший разряд (рис. 1.9.1).

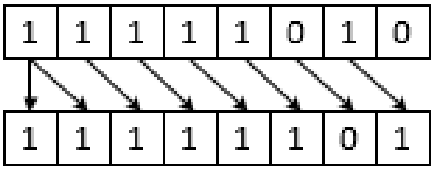

Рисунок 1.9.1 – Арифметический сдвиг числа в дополнительном коде в сторону младших разрядов

Допустим у нас есть число 11.111010 = -6 (в двоичной системе в дополнительном коде.) После сдвига вправо на 1 бит, получим число 11.111101 = -3.

Если осуществляется сдвиг в сторону старших разрядов, то вдвигается ноль (рис. 1.9.2).

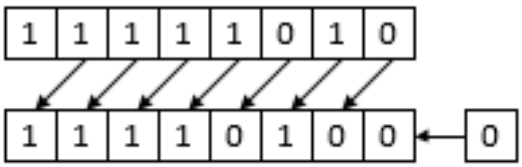

Рисунок 1.9.2 – Арифметический сдвиг числа в дополнительном коде в сторону старших разрядов

После сдвига числа 11.111010 = -6 влево на 1 бит, получим число 11.110100 = -12.

Таким образом на входы IL, IR сдвиговых регистров подаем 0 и старший разряд соответственно.

Описание некоторых команд спецпроцессора:

0,25A+2B

Операция деления, реализовывается с помощью сдвига в сторону младших разрядов. Эта операция необходима для получения числа 0,25A. Для реализации операции сдвига используется регистр, предназначенный для сдвига в сторону младших разрядов и хранения числа A.

Операция умножения реализуется с помощью сдвига числа на один разряд в сторону старших разрядов. Регистр, предназначенный для сдвига в сторону старших разрядов и хранения числа B, используется для получения числа 2B, путем сдвига в сторону старших разрядов.

Используя сумматор, производится операция сложения чисел 0,25A и 2B, тем самым реализуя операцию 0,25A+2B.

A-0,5B

В регистре, предназначенном для сдвига в сторону старших разрядов и хранения числа, хранится число A. Регистр, предназначенный для сдвига в сторону младших разрядов и хранения числа B, используется для получения числа 0,5B, путем сдвига в сторону младших разрядов.

Для выбора передаваемого значения с регистра, используется мультиплексор. Отрицательное число 0,5B заменяет операцию вычитания операцией сложения.

С помощью сумматора, выполняется операция сложения содержимого регистра, хранящего число A и содержимого мультиплексора, используемого для выбора отрицательного числа 0,5B, тем самым реализуется операция A-0,5B.

1.4 Выбор ПЛИС для реализации спецпроцессора

ПЛИС (программируемые логические интегральные схемы) – электронный компонент, логика работы которого не определяется при изготовлении, а задаётся посредством программирования связей между элементами. В зависимости от способа изготовления ПЛИС могут программироваться либо один раз, либо многократно. Устройства, которые могут программироваться только один раз, называются однократно программируемыми. В настоящее время ПЛИС заполняют четыре крупных сегмента рынка: заказные интегральные схемы, цифровая обработка сигналов, системы на основе встраиваемых микроконтроллеров и микросхемы, обеспечивающие физический уровень передачи данных [7].

Достаточно много компаний в мире занято производством цифровых устройств на основе ПЛИС и использованием их в своих системах. Некоторые из них представлены ниже.

1. Intel (Altera). Компания концентрируется на разработке схем и модулей на основе таких языков описания аппаратуры, как VHDL, Verilog и AHDL. Основными изделиями являются программируемые микросхемы, а также услуги по преобразованию проектов под ПЛИС в ASIC для массового производства. Компания также выпускает программы для разработки встроенного программного обеспечения для ПЛИС и компиляторы под ядро процессора собственной разработки.

2. Xilinx. Основная продукция – микросхемы FPGA, перепрограммируемые микросхемы с традиционной архитектурой PAL, – а также средства их проектирования и отладки, выпускаемые фирмой Xilinx. Используются в устройствах цифровой обработки информации, например, в системах телекоммуникации и связи, вычислительной технике, периферийном и тестовом оборудовании, электробытовых приборах.

3. Microchip (Microsemi/Actel). Особенностью данных ПЛИС является применение Antifuse технологии, которая обеспечивает высокую надежность и

гибкие ресурсы трассировки и не требует конфигурационного ПЗУ. Продукция в виде радиационно-стойких программируемых матриц применяется в военной и аэрокосмической отраслях. Также, компания представляет семейства ПЛИС для области портативной и энергозависимой электроники и предоставляет возможности для построения систем на кристалле (СнК), таким образом обеспечивая большую гибкость по сравнению с традиционными решениями на аппаратных микроконтроллерах.

4. Achronix Semiconductor. Компания делает ставку на высокую производительность и надежность, а не на низкую стоимость и энергопотребление. Выпускается две серии микросхем: Achronix-Ultra с частотой до 2.2 ГГц, что является рекордным значением для ПЛИС, а также Achronix-Xtreme с частотой около 1 ГГц, устойчивых к радиационному излучению и предназначенных для работы в большом температурном диапазоне - от -260°C до +130°C. Причиной появления столь высокой тактовой частоты является использование асинхронной технологии - элементы микросхемы не синхронизированы между собой [8].

Продукты Xilinx и Altera первыми появились на российском рынке, и их применение сегодня во многом обусловлено наличием опыта работы как с самими микросхемами, так и со средствами разработки, традициями применения ПЛИС одного производителя в рамках одного предприятия и действительными и мнимыми рисками, связанными с переходом на новую элементную базу.

Для разработки спецпроцессора была выбрана ПЛИС фирмы Altera, их среда проектирования стабильно обновляется. Это одна из простых и недорогих FPGA. Отладочная плата DE2-115 с целевым кристаллом семейства: Cyclone IV E (EP4CE115F29C7N)). Преимуществом семейства является то, что это недорогие FPGA с высокоскоростными приемопередатчиками и низким энергопотреблением. Cyclone IV необходимо только два источника питания, поэтому экономиться затраты и время благодаря более упрощенной сети распределения питания. Семейство Cyclone IV автоматически калибрует

интерфейсы памяти для различных процессов и подстраивается под изменения напряжения и температуры, что минимизирует временное рассогласование и увеличивает надежность.

Необходимое для разработки ПО: Quartus Prime Light Edition версия 18.0.

1.5 Результаты анализа

Был выполнен обзор предметной области и проведен анализ задания на ВКР. На основе анализа, были сделаны выводы:

1. Сформулированы требования к разрабатываемому спецпроцессору:

   1.1 В качестве архитектуры процессора предпочтительнее выбрать RISC–архитектуру процессора с сокращённым набором инструкций, так как благодаря использованию простых команд и минимума их форматов сокращается время разработки. Каждая команда RISC-процессора должна выполняться за один машинный такт, что облегчает повышение тактовой частоты. Также, в таких процессорах, как правило, повышается производительность.

   1.2 Структура операционного автомата смешанная, включает в себя элементы канонической структуры, I-автомата, M-автомата, т.к. используются эквивалентные операции и обобщенные операторы.

   1.3 Составив список флажков, в соответствии с которыми требуется выполнить ту или иную арифметическую операцию, был сделан вывод, что потребуется десятиразрядный регистр флагов.

2. Проанализирован список необходимых условий, в соответствии с которыми выполняются арифметические операции. Команды выбраны одноадресные, где используется один адрес, по которому до выполнения операции находится операнд, сама операция производится в этом регистре и после выполнения операции, по этому адресу находится результат выполнения. А также трехадресные, где используется три адреса: по первому адресу

находится результат операции, по второму и третьему - первый и второй операнд соответственно.

3. ПЛИС для реализации спецпроцессора выбрана фирмы Altera.

2 Проектирование структурной схемы спецпроцессора

2.1 Условное графическое обозначение спецпроцессора

Для обозначения спецпроцессора будет использоваться следующее условное графическое обозначение (рис. 2.1).

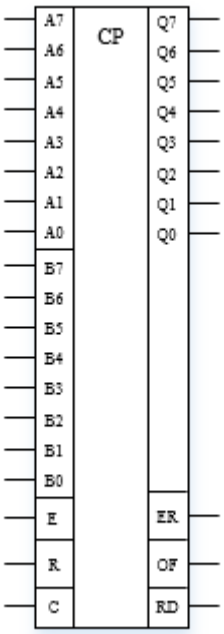

Рисунок 2.1 – УГО спецпроцессора

Микросхема разработанного специализированного спецпроцессора имеет 30 информационных выводов.

Назначение входов спецпроцессора:

– A0-A7 – число A (A7 – знаковый разряд, A6 – разряд переполнения, A5-A0 – разряды от старшего к младшему);

– B0-B7 – число B (B7 – знаковый разряд, B6 – разряд переполнения, B5-B0 – разряды от старшего к младшему);

– E – сигнал о наличии чисел A и B на входах A0 – A7 и B0 – B7 соответственно;

– C – вход синхросигнала;

– R – сброс.

Назначение выходов спецпроцессора:

– Q0-Q7 – результат операций (Q7, Q6 – знаковые разряды, Q5-Q0 – разряды от старшего к младшему);

– ER – выход сигнала об ошибке – не соответствие операндов A и B ни одному из условий выполнения операции;

– OF – сигнал о возникновении переполнения разрядной сетки;

– RD– сигнал о наличии на выходах результата.

2.2 Разработка операционного автомата спецпроцессора

2.2.1 Функциональная схема ОА

После анализа операций, выполняемых спецпроцессором, а также сведений, связанных с представлением числа в модифицированном дополнительном коде, предлагается следующая структура операционного автомата (рис. 2.2).

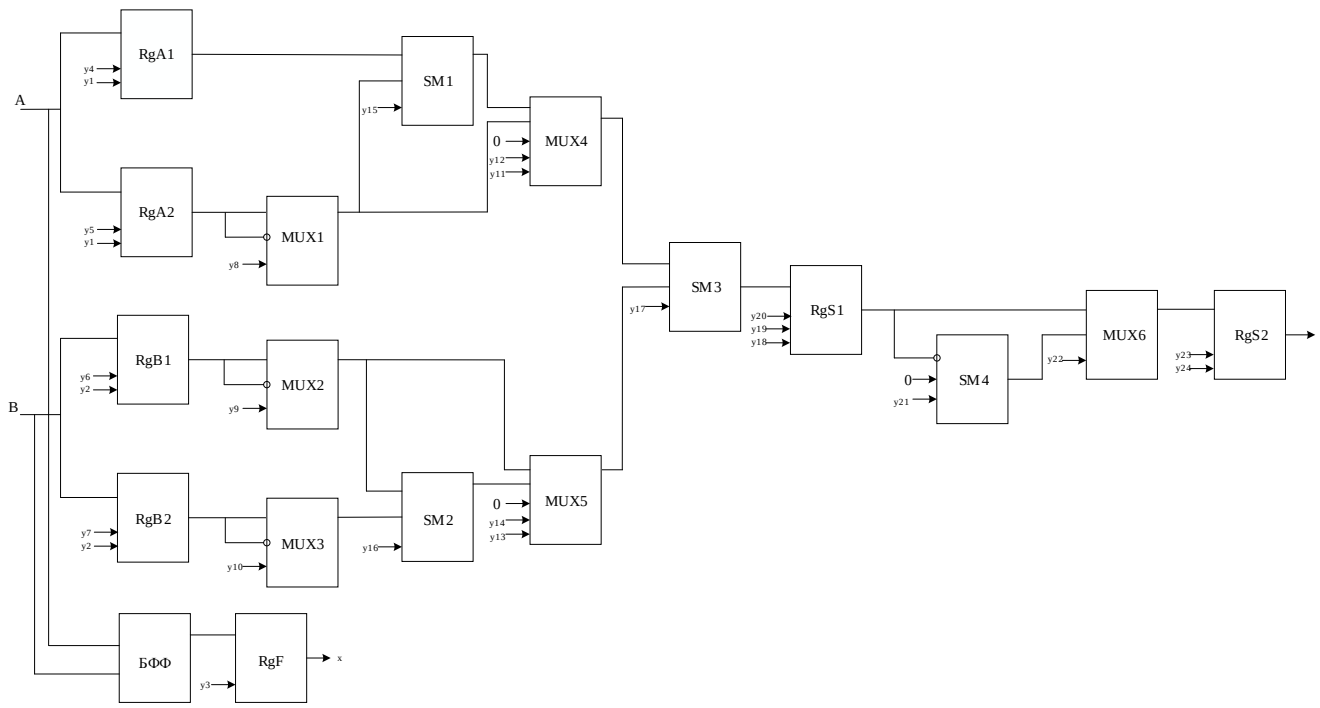

Рисунок 2.2 – Функциональная схема ОА

Для разработки данной схемы был произведен анализ и сделаны следующие выводы

– для представления чисел A и B всего потребуется 4 регистра. Регистры RgA1 и RgB1 предназначены для хранения чисел A и B соответственно, а также для сдвига двоичных дополнительных кодов этих чисел в сторону старших разрядов, для представления чисел 2A, 4A, 2B, 4B. Регистры RgA2 и RgB2 также выполняют функцию хранения чисел A и B и осуществляют сдвиг в сторону младших разрядов, для представления чисел 0,5A, 0,25A, 0,5B, 0,25B.

– мультиплексор MUX1 предназначен для передачи числа A или –A. Мультиплексоры MUX2 и MUX3 для передачи чисел B или –B, 2B или -2B, 0,5B или -0,5B, 0,25B или -0,25B.

– для выбора одного значения от мультиплексора и сумматора используются два мультиплексора MUX4 и MUX5, выходы которых идут на сумматор SUM3.

– мультиплексор MUX6 предназначен для выбора нужного результата.

– необходимо использовать четыре сумматора.

Сумматор SM1 предназначен для сложения содержимого регистров RgA1 и RgA2. Пример: в регистре RgA1 хранится число A, регистр RgA2 используется для получения числа 0,25A, путем сдвига в сторону младших разрядов. В сумматоре SM1 происходит сложение содержимого этих регистров, для выполнения операции 1,25A.

Сумматор SM2 – для сложения RgB1 и RgB2. Пример: регистр RgB1 используется для получения числа 2B, путем сдвига в сторону старших разрядов, регистр RgB2 используется для получения числа 0,25B, путем сдвига в сторону младших разрядов. С помощью мультиплексора MUX3 получаем инверсное значение числа 0,25B. В сумматоре SM2 происходит сложение содержимого этих регистров, для выполнения операции 1,75B.

SM3 нужен для получения промежуточного результата, передает значение на регистр RgS1. Пример: на выходе мультиплексора MUX4 число 1,25A, на выходе мультиплексора MUX5 число 1,75B. В сумматоре SM3 происходит сложение содержимого этих регистров, для выполнения операции 1,25A + 1,75B.

SM4 – для сложения с единицей. На сумматор подается управляющий сигнал, для прибавления единицы к младшему разряду во время преобразования в дополнительный код, если требуется выполнить операцию вычитания.

2.2.2 Блок формирования флажков

Регистр флагов предназначен для фиксации и хранения флагов (признаков), характеризующих результат выполненной арифметической или логической операции. Такие признаки могут информировать о равенстве результата нулю, о знаке результата, о возникновении переноса из старшего разряда, переполнении разрядной сетки и т. д. Содержимое регистра обычно используется устройством управления АЛУ и процессора для реализации

условных переходов по результатам операций АЛУ. Под каждый из возможных признаков отводится один разряд регистра флагов.

Формирование признаков осуществляется блоком формирования состояний регистра признаков, который может входить в состав операционного блока либо реализуется в виде внешней схемы, располагаемой между операционным блоком и регистром флагов.

a[0] - a[7] – число A.

b[0] - b[7] – число B.

s[0] - s[9] – флажки, подаваемые на входы регистра флагов.

Флажки числа A:

Рисунок 2.3.1 – Знак

Рисунок 2.3.2 – Четность

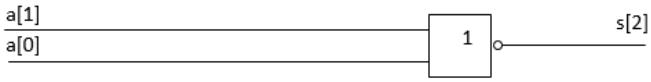

Рисунок 2.3.3 – Кратность 4

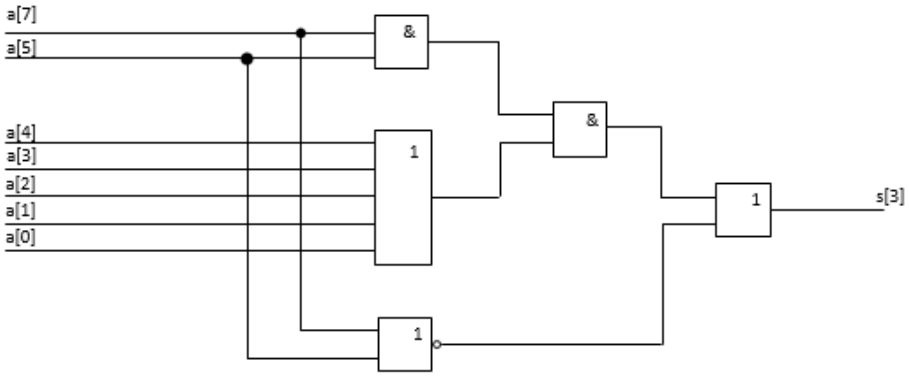

Рисунок 2.3.4 – $|A| < 0{,}5$

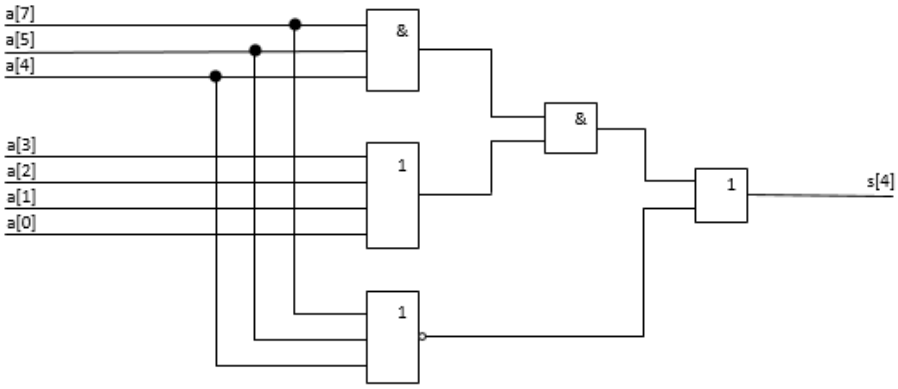

Рисунок 2.3.5 – $|A| < 0{,}25$

Флажки числа B:

Рисунок 2.3.6 – Знак

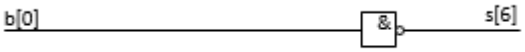

Рисунок 2.3.7 – Четность

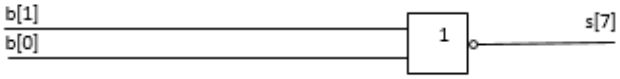

Рисунок 2.3.8 – Кратность 4

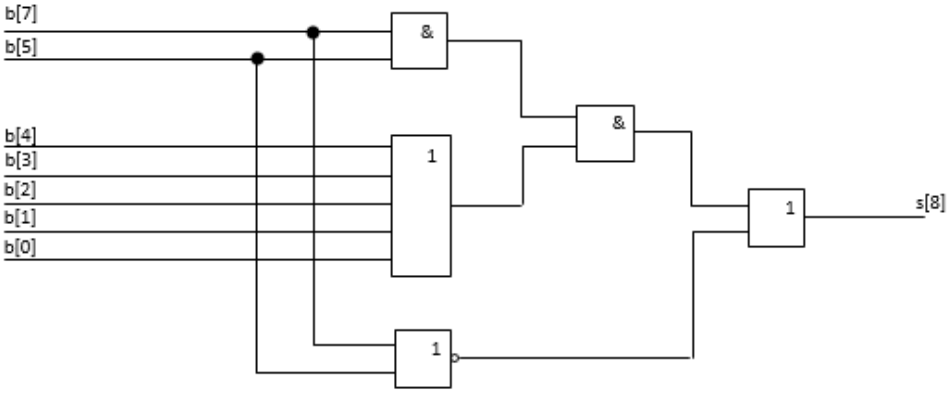

Рисунок 2.3.9 – $|B| < 0{,}5$

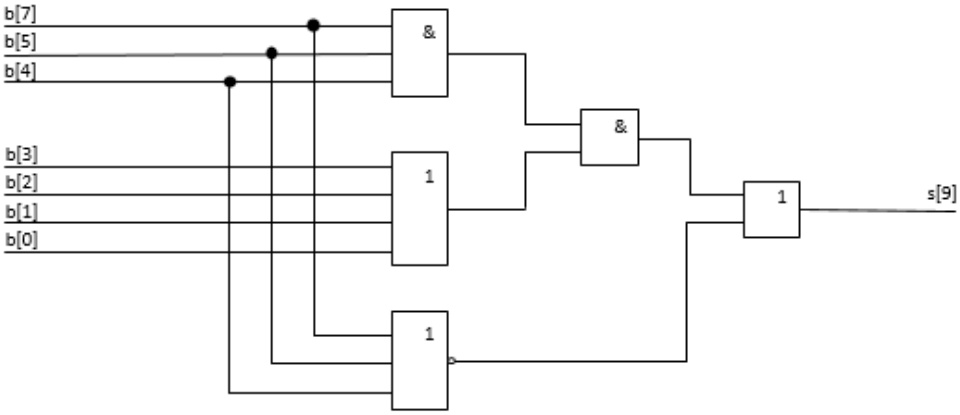

Рисунок 2.3.10 – $|B| < 0{,}25$

Запись полученных флажков производится в десятиразрядный регистр флагов по сигналу y3 = 1 (рис 2.4). Регистр RgF имеет: D0-D9 –прямые входы регистра; Q0-Q9 –прямые выходы регистра; вход LD, при поступлении на вход логической единицы производится запись с входов D0 – D9, информация появляется на выходах Q0-Q9.

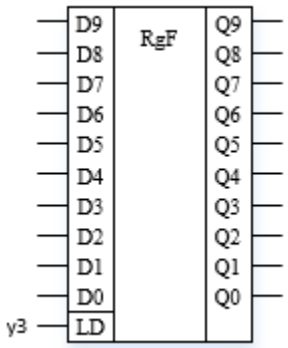

Рисунок 2.4 – Десятиразрядный регистр флагов

2.2.3 Элементы и узлы ОА

В функциональной схеме операционного автомата используются следующие элементы и узлы.

RgA1, RgB1 – восьмиразрядные универсальные регистры сдвига, предназначенные для сдвига в сторону старших разрядов и хранения чисел A и B соответственно (рис.2.5). При подаче управляющего сигнала y4 = 1 (для

регистра RgA1) или y6 = 1 (для регистра RgB1), который поступает на управляющий вход ← происходит сдвиг числа в сторону старших разрядов. При сдвиге в освободившийся разряд будет записываться значение со входа IL = 0.

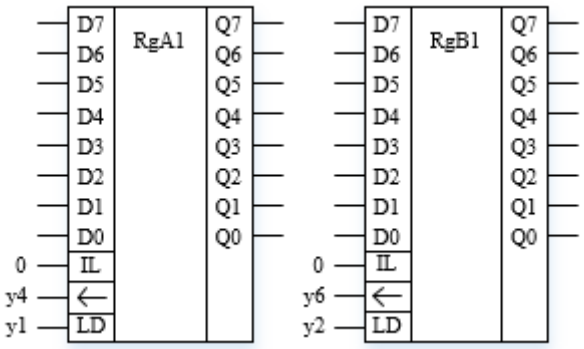

Рисунок 2.5 – Регистры RgA1, RgB1

RgA2, RgB2 – восьмиразрядные универсальные регистры сдвига, предназначенные для сдвига в сторону младших разрядов и хранения чисел A и B (рис.2.6).

При подаче управляющего сигнала y5 = 1 (для регистра RgA2) или y7 = 1 (для регистра RgB2), который поступает на вход → происходит сдвиг числа в сторону младших разрядов. При сдвиге, в освободившийся разряд будет записываться значение со входа IR.

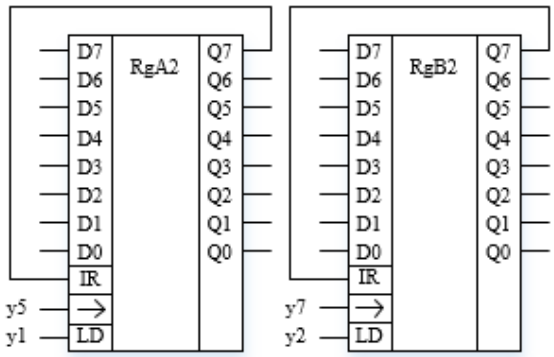

Рисунок 2.6 – Регистры RgA2, RgB2

Восьмиразрядный универсальный реверсивный регистр RgS1 служит для хранения промежуточного результата и деления или умножения его на 2 (рис. 2.7). Регистр RgS1 имеет входы ← и →, IL и IR. При подаче управляющего сигнала y19 = 1 на управляющий вход ← происходит сдвиг числа в сторону старших разрядов. При сдвиге в освободившийся разряд будет записываться значение со входа IL = 0. При подаче управляющего сигнала y20 = 1 на вход → происходит сдвиг числа в сторону младших разрядов. При сдвиге, в освободившийся разряд будет записываться значение со входа IR.

Для запоминания результата – параллельный регистр RgS2 (рис. 2.7). Регистр RgS2 имеет входы → и IR. При подаче управляющего сигнала y23 = 1, который поступает на вход → происходит сдвиг числа в сторону младших разрядов. При сдвиге, в освободившийся разряд будет записываться значение со входа IR.

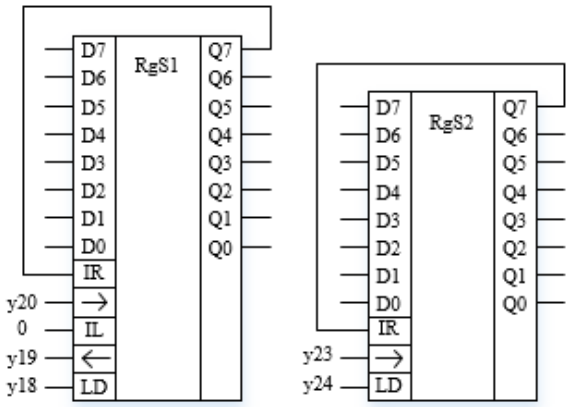

Рисунок 2.7 – Регистры RgS1 и RgS2

Все регистры ОА (RgA1, RgB1, RgA2, RgB2, RgS1, RgS2) имеют:

– D0-D7 –прямые входы регистров (D6, D7 – знаковые разряды, D0-D5– значащие).

– Q0-Q7 –прямые выходы регистров (Q6, Q7 –знаковые разряды, Q0-Q5– значащие).

– LD. При поступлении на вход LD логической единицы производится запись с входов D0 – D7, информация появляется на выходах Q0-Q7.

На рисунке 2.8 представлен мультиплексор MUX1. При подаче управляющего сигнала y8 = 1, который поступает на вход Z на выходы Q0 – Q7 поступают данные с входов 1.0 – 1.7, если y8 = 0, подаются данные с входов 0.0 – 0.7. Мультиплексоры MUX2, MUX3 и MUX6 работают аналогично.

Мультиплексоры MUX1 и MUX3 используются для выбора передаваемого значения с регистров RgA2 и RgB2. Отрицательные числа A и B заменяют операцию вычитания операцией сложения. Мультиплексор MUX2 используется для выбора передаваемого значения с регистра RgB1. Для выбора записываемого значения в регистр RgS2 используется мультиплексор MUX6.

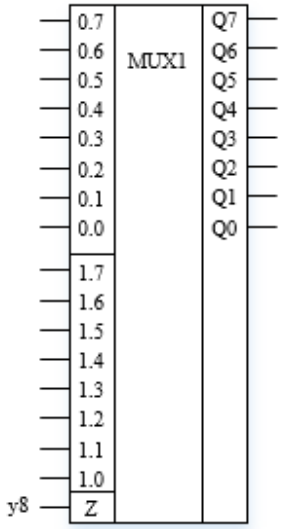

Рисунок 2.8 – Мультиплексор MUX1

Выбор пары чисел, которые будут поданы на сумматор SM3 происходит с помощью MUX4 и MUX5 – мультиплексоров на три группы информационных входов. На рисунке 2.9 представлен мультиплексор MUX4. Таблица истинности этого мультиплексора представлена в таблице 3 [10]. Мультиплексор MUX5 работает аналогично.

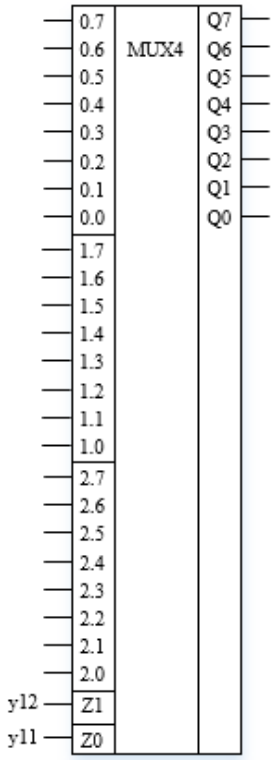

Рисунок 2.9 – Мультиплексор MUX4

Таблица 3 – Таблица истинности MUX4

Z1	Z0	Выходы
0	0	0.0 – 0.7
0	1	1.0 – 1.7
1	1	2.0 – 2.7

Для выполнения операции суммирования используются сумматоры. Сумматор SM1 представлен на рисунке 2.10. В данном случае используется четыре восьмиразрядных сумматора: SM1, SM2, SM3, SM4.

A0 – A7 входы числа A.

B0 – B7 входы числа B.

На входы подается два числа для выполнения операции суммирования, результат суммирования будет на выходах S0 – S7.

C – вход входного переноса.

P – выходной перенос в следующий разряд.

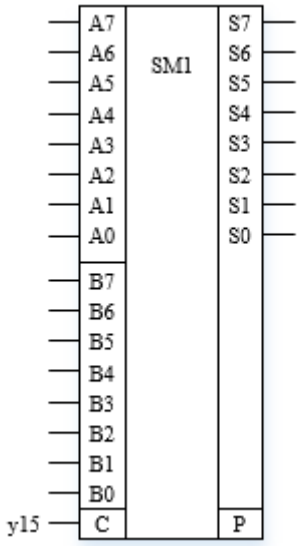

Рисунок 2.10 – Сумматор SM0

Работа всех сумматоров аналогична. Для замены операции вычитания операцией сложения на вход C подается 1. На выходах S0-S7 получается число в дополнительном коде.

2.2.4 Осведомительные и управляющие сигналы

Определение осведомительных и управляющих сигналов.

Осведомительные сигналы:

x1 – сигнал, представляющий собой флажок s[0] (знак числа A), который находится в разряде Q9 регистра флагов RgF;

x2 – сигнал, представляющий собой флажок s[1] (четность числа A), который находится в разряде Q8 регистра флагов RgF;

x3 – сигнал, представляющий собой флажок s[2] (кратность 4 для числа A), который находится в разряде Q7 регистра флагов RgF;

x4 – сигнал, представляющий собой флажок s[3] ($|A| < 0{,}5$), который находится в разряде Q6 регистра флагов RgF;

x5 – сигнал, представляющий собой флажок s[4] ($|A| < 0{,}25$), который находится в разряде Q5 регистра флагов RgF;

x6 – сигнал, представляющий собой флажок s[5] (знак числа B), который находится в разряде Q4 регистра флагов RgF;

x7 – сигнал, представляющий собой флажок s[6] (четность числа B), который находится в разряде Q3 регистра флагов RgF;

x8 – сигнал, представляющий собой флажок s[7] (кратность 4 для числа B), который находится в разряде Q2 регистра флагов RgF;

x9 – сигнал, представляющий собой флажок s[8] ($|B| < 0,5$), который находится в разряде Q1 регистра флагов RgF;

x10 – сигнал, представляющий собой флажок s[9] ($|B| < 0,25$), который находится в разряде Q0 регистра флагов RgF;

P – сигнал проверки переполнения разрядной сетки;

E – сигнал о наличии на входах чисел A и B.

Управляющие сигналы:

y1 – запись числа с входов A7-A0 спецпроцессора в регистры RgA1 и RgA2;

y2 – запись числа с входов B7-B0 спецпроцессора в регистры RgB1 и RgB2;

y3 – запись флажков в регистр флагов RgF;

y4 – RgA1 ← 2*RgA1 – сдвиг содержимого регистра RgA1 на один разряд в сторону старших разрядов (умножение числа, содержащегося в RgA1 на 2);

y5 – RgA2 ← RgA2/2 сдвиг содержимого регистра RgA2 на один разряд в сторону младших разрядов (деление числа, содержащегося в RgA2 на 2);

y6 – RgB1 ← 2*RgB1 – сдвиг содержимого регистра RgB1 на один разряд в сторону старших разрядов (умножение числа, содержащегося в RgB1 на 2);

y7 – RgB2 ← RgB2/2 сдвиг содержимого регистра RgB2 на один разряд в сторону младших разрядов (деление числа, содержащегося в RgB2 на 2);

y8 – инверсия содержимого регистра RgA2 с помощью мультиплексора MUX1;

y9 – инверсия содержимого регистра RgB1 с помощью мультиплексора MUX2;

y10 – инверсия содержимого регистра RgB2 с помощью мультиплексора MUX3;

y11 – выбор адреса в мультиплексоре MUX4 (вход Z0);

y12 – выбор адреса в мультиплексоре MUX4 (вход Z1);

y13 – выбор адреса в мультиплексоре MUX5 (вход Z0);

y14 – выбор адреса в мультиплексоре MUX5 (вход Z1);

y15 – прибавление «1» к младшему разряду с помощью сумматора SM1;

y16 – прибавление «1» к младшему разряду с помощью сумматора SM2;

y17 – прибавление «1» к младшему разряду с помощью сумматора SM3;

y18 – запись результата операции в регистр RgS1;

y19 – RgS1 ← 2*RgS1 – сдвиг содержимого регистра RgS1 на один разряд в сторону старших разрядов (умножение числа, содержащегося в RgS1 на 2);

y20 – RgS1 ← RgS1/2 – сдвиг содержимого регистра RgS1 на один разряд в сторону младших разрядов (деление числа, содержащегося в RgS1 на 2);

y21 – $Rgs2 \leftarrow \overline{RgS1}+1$ – инверсия содержимого RgS1, прибавление «1» к младшему разряду с помощью сумматора SM4;

y22 – выбор адреса в мультиплексоре MUX6;

y23 – RgS2 ← RgS2/2 – сдвиг содержимого регистра RgS2 на один разряд в сторону младших разрядов (деление числа, содержащегося в RgS2 на 2);

y24 – запись результата операции в регистр RgS2;

y25 (RD) – сообщение о готовности результата;

y26 (OF) – сообщение о переполнении разрядной сетки;

y27 (ER) – сообщение об ошибке.

Некоторые команды являются совместимыми, т.е. могут выполняться параллельно. Например, запись чисел A и B в регистры RgA1, RgA2 и RgB1, RgB2 соответственно; сдвиг содержимого регистров RgA1, RgA2, RgB1, RgB2.

2.3 Разработка управляющего автомата

2.3.1 Определение требований к компонентам

Микрокоманда в автомате с принудительной адресацией включает в себя 4 поля:

– CX – поле управления выбором опрашиваемого входного осведомительного сигнала x_i;

– A1 – поле адреса следующей микрокоманды в случае, если опрашиваемый осведомительный сигнал принимает значение 1.

– A0 – поле адреса следующей микрокоманды, если опрашиваемый осведомительный сигнал будет равен 0.

– Y – поле выходных управляющих сигналов.

Процесс разработки функциональной схемы заключается в определении разрядностей полей микрокоманды и выборе разрядности регистра адреса, регистра микрокоманд, разрядности мультиплексоров и организации ЗУ.

Определяем разрядность отдельных полей микрокоманды и самой микрокоманды в целом [10].

Разрядность поля управления выбором опрашиваемого входного осведомительного сигнала:

$$n_x = \lceil \log_2 n_{\text{ос}} \rceil = \lceil \log_2 14 \rceil = 4 \tag{2.1}$$

Проанализировав варианты задания на курсовой проект, был сделан вывод, что 32 ячеек ЗУ хватит для любого варианта, из чего следует, что разрядность полей адреса следующей микрокоманды равна 5

Пример для 54 варианта:

$$n_a = \lceil \log_2 [n_{оп} + n_{ув}] \rceil = \lceil \log_2 (13 + 8) \rceil = 5 \tag{2.2}$$

Число разрядов поля Y:

$$n_y = k = 27 \tag{2.3}$$

$n_{\text{ос}}$ – 14 (12 внутренних флажков, сигнал о наличии чисел A и B на входах A0-A7 и B0-B7, сигнал о переполнении);

$n_{\text{оп}}$ – количество операторных вершин, равное 13;

$n_{\text{ув}}$ – количество условных вершин, равное 8;

k – число управляющих сигналов, равное 27.

Разрядность микрокоманды в целом составляет:

$$S = n_x + 2 * n_a + n_y = 4 + 2 * 5 + 27 = 41 \tag{2.4}$$

Таким образом, для реализации спецпроцессора потребуется ЗУ организации $2^5 \times 41$. Это значит, что ЗУ должно иметь 5 адресных входов, что позволит обратиться к 32 ячейкам памяти. Также ЗУ будет иметь 41 выходов для вывода микрокоманды.

Выполнив анализ вариантов задания на курсовое проектирование и разработав несколько граф-схем, был сделан вывод, что для реализации управляющего автомата спецпроцессора потребуется ЗУ организации $2^5 \times 41$. Условное графическое обозначение такого ЗУ представлено на рисунке 2.11.

2.3.2 Элементы и узлы УА

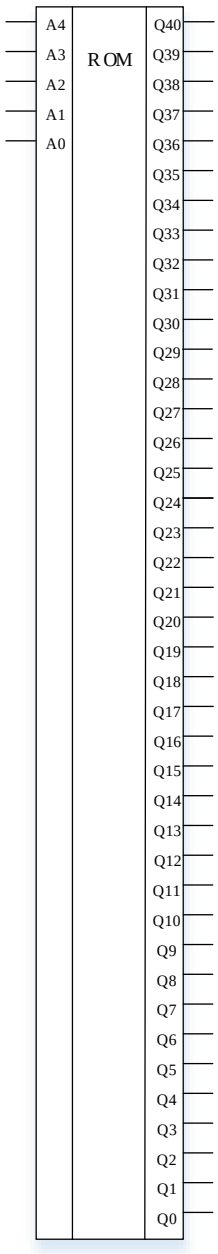

Рисунок 2.11 – ЗУ, предназначенное для хранения микропрограммы

При поступлении на адресные входы нового адреса происходит выдача информации из ячейки на выход.

В качестве регистра адреса выбран 5-разрядный параллельный регистр с возможностью асинхронного сброса (рис. 2.12). В каждом такте работы спецпроцессора в нем хранится адрес текущей микрокоманды.

Вход R – обнуление регистра.

Вход C – синхронизация. При поступлении синхроимпульса происходит запись адреса следующей микрокоманды в регистр.

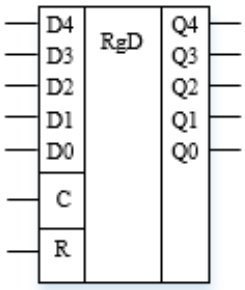

Рисунок 2.12 – Регистр, отвечающий за хранение адреса текущей микрокоманды

На рисунке 2.13 представлен мультиплексор MUXA, предназначенный для выбора адреса следующей микрокоманды.

Вход Z – выбор входов A0.0-A0.4 или A1.0-A1.4 и коммутация их на выходы Q0-Q4.

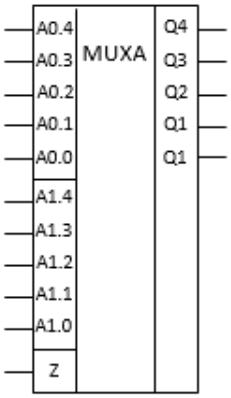

Рисунок 2.13 – Мультиплексор выбора адреса следующей микрокоманды

Мультиплексор MUXX отвечает за распределение выбора осведомительного сигнала (рис. 2.14). На входы A0 – A2 подается адрес осведомительного сигнала, на выходе появляется значение данного сигнала.

0 – 7 – входы осведомительных сигналов.

Назначение адресных входов мультиплексора MUXX приведено в таблице 4 [10].

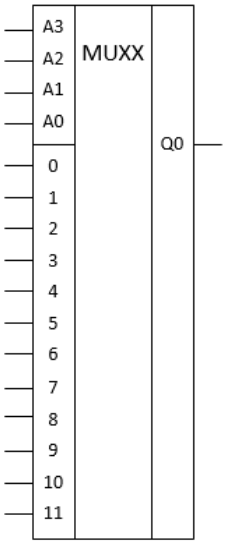

Рисунок 2.14 – Мультиплексор выбора осведомительного сигнала

Таблица 4 – Назначение адресных входов мультиплексора MUXX

A3	A2	A1	A0	Сигнал
0	0	0	0	E
0	0	0	1	x1
0	0	1	0	x2
0	0	1	1	x3
0	1	0	0	x4
0	1	0	1	x5
0	1	1	0	x6
0	1	1	1	x7
1	0	0	0	x8
1	0	0	1	x9
1	0	1	0	x10
1	0	1	1	P

Регистр текущей микрокоманды (рис. 2.15) представляет собой параллельный регистр. Запись текущей микрокоманды производится по сигналу C. Вход R – обнуление (сброс) регистра.

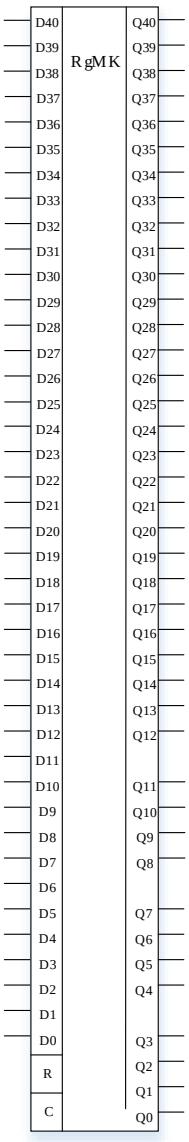

Рисунок 2.15 – Регистр текущей микрокоманды

3 Реализация на ПЛИС

3.1 Операционный автомат

Для реализации операционного автомата спецпроцессора был создан модуль ОА, представленный на рисунке 3.2. Разработанная программа для модуля ОА приведена на CD-диске прилагающемся к ВКР. Модуль ОА включает в себя девять модулей, представленных на рисунках 3.1.1 – 3.1.9, которые получены на основе описания алгоритма работы модулей:

– Reg_left – модуль, предназначенный для реализации восьмиразрядного регистра сдвига, с помощью которого выполняется хранение и сдвиг числа в сторону старших разрядов (рис 3.1.1);

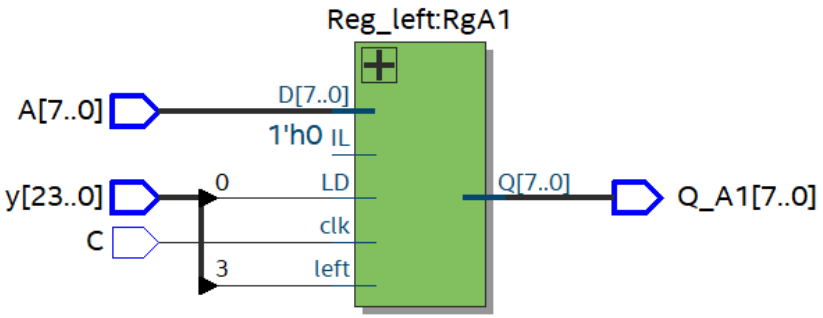

Рисунок 3.1.1 – Регистр сдвига в сторону старших разрядов

– Reg_right – модуль, предназначенный для реализации восьмиразрядного регистра сдвига, с помощью которого выполняется хранение и сдвиг числа в сторону младших разрядов (рис 3.1.2);

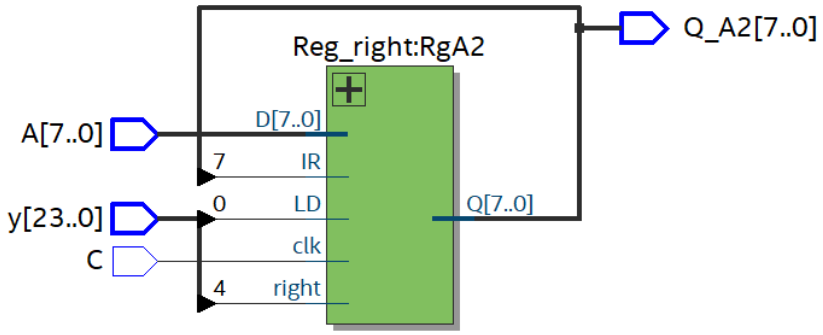

Рисунок 3.1.2 – Регистр сдвига в сторону младших разрядов

– FlagsBlock – модуль, предназначенный для формирования флажков, в соответствии с которыми требуется выполнить ту или иную арифметическую операцию (рис 3.1.3);

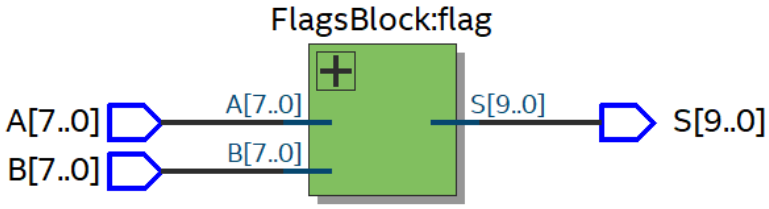

Рисунок 3.1.3 – Блок формирования флажков

– RegN – модуль, предназначенный для реализации десятиразрядного регистра флагов, в который записываются сформированные флажки (рис 3.1.4);

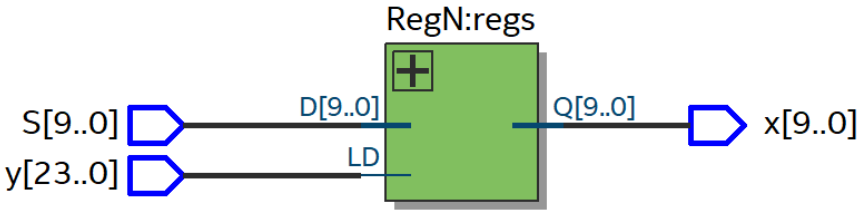

Рисунок 3.1.4 – Регистр флагов

– Reg_revers – модуль, предназначенный для реализации восьмиразрядного реверсивного регистра, с помощью которого выполняется хранение и сдвиг числа в сторону младших или старших разрядов (рис 3.1.5);

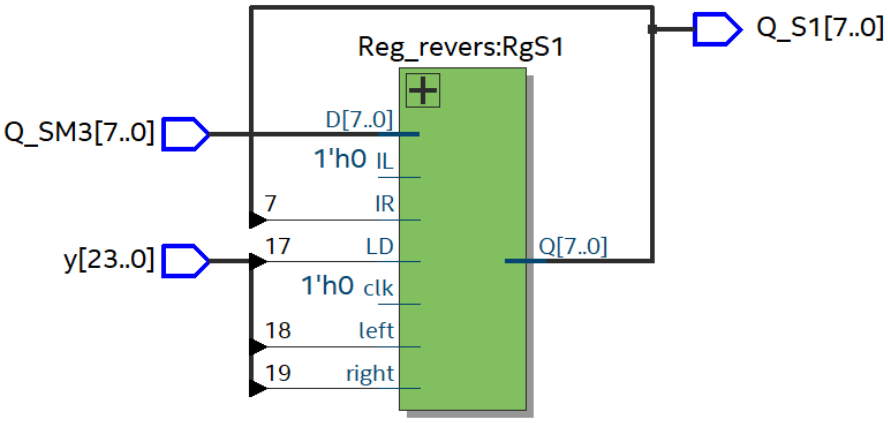

Рисунок 3.1.5 – Реверсивный регистр сдвига

Модули Reg_left, Reg_right, RegN и Reg_revers включают в себя модуль REG, который представляет собой одноразрядный регистр.

– Sum8 – модуль, предназначенный для реализации восьмиразрядного сумматора, с помощью которого выполняется сложение двух чисел (рис 3.1.6);

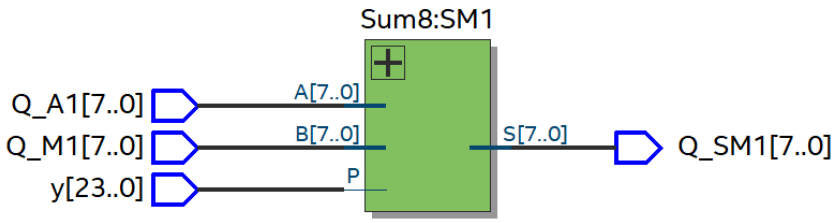

Рисунок 3.1.6 – Восьмиразрядный сумматор

– Sum4 – модуль, реализующий восьмиразрядный сумматор, предназначенный для получения чисел в дополнительном коде (рис 3.1.7);

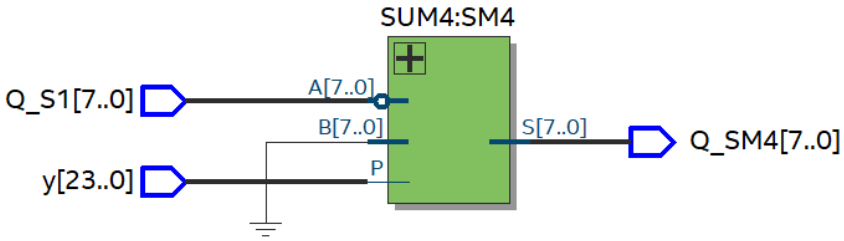

Рисунок 3.1.7 – Восьмиразрядный сумматор для получения чисел в дополнительном коде

Модули Sum8 и Sum4 включают в себя модуль SUM, который представляет собой одноразрядный сумматор.

– Mux2 – модуль, реализующий мультиплексор на две группы информационных входов, предназначенный для выбора одного из двух, поступающих на входы, значений (рис 3.1.8);

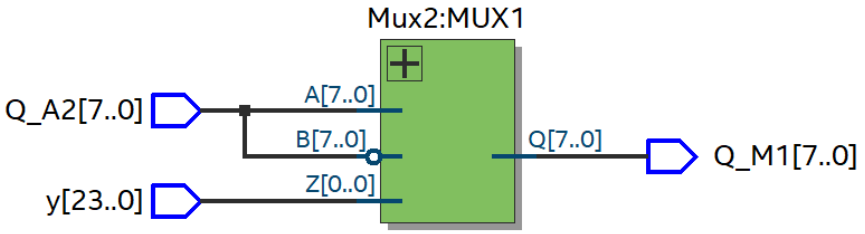

Рисунок 3.1.8 – Мультиплексор на две группы информационных входов

– Mux3 – модуль, реализующий мультиплексор на три группы информационных входов, предназначенный для выбора одного из трех, поступающих на входы, значений (рис 3.1.9);

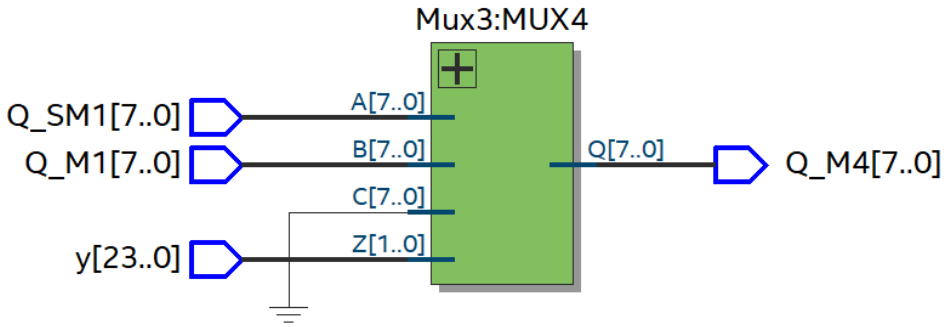

Рисунок 3.1.9 – Мультиплексор на три группы информационных входов

Модули Mux2 и Mux3 включают в себя модуль Decoder, который представляет собой дешифратор для мультиплексора.

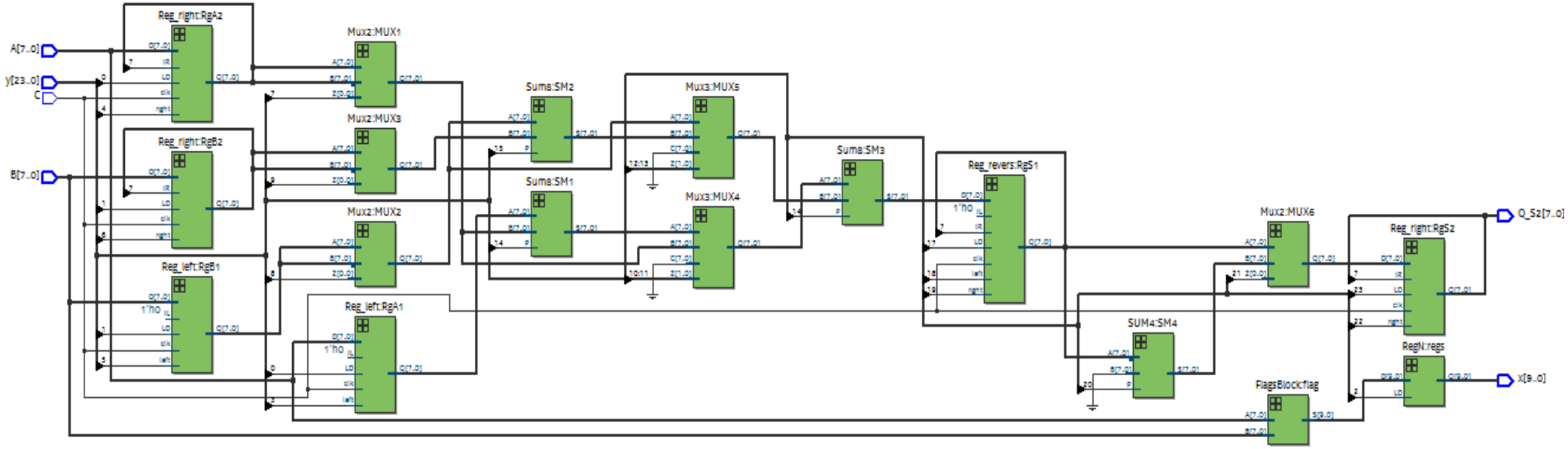

Рисунок 3.2 – Операционный автомат спецпроцессора

3.2 Управляющий автомат

Для реализации управляющего автомата спецпроцессора был создан модуль YA, представленный на рисунке 3.4. Разработанная программа для модуля YA приведена на CD-диске прилагающемся к ВКР.

В случае реализации разрабатываемого специализированного процессора на ПЛИС вместо ПЗУ необходимо использовать ОЗУ, в которое перед началом работы из файла будет записываться одна микропрограмма, соответствующая определенному варианту.

Модуль YA включает в себя пять модулей, представленных на рисунках 3.3.1 – 3.3.5, которые получены на основе описания алгоритма работы модулей:

– RAM – модуль, предназначенный для реализации ЗУ организации $2^5 \times 41$ (рис. 3.3.1);

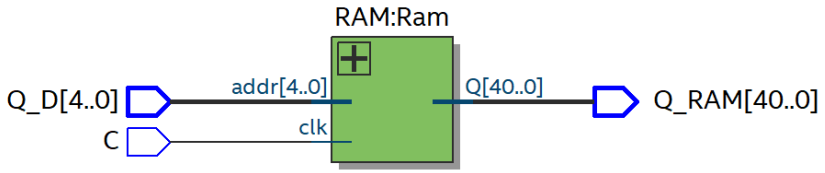

Рисунок 3.3.1 – Запоминающее устройство

– RegD – модуль, предназначенный для реализации пятиразрядного регистра адреса в котором хранится адрес текущей микрокоманды (рис. 3.3.2);

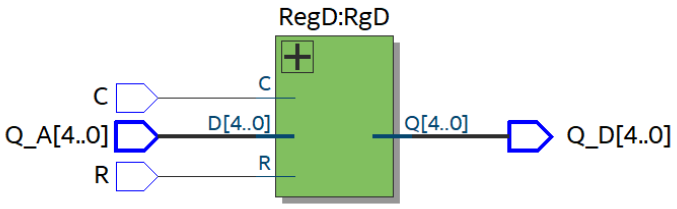

Рисунок 3.3.2 – Регистр адреса

– RegMK – модуль, предназначенный для реализации параллельного 41-разрядного регистра текущей микрокоманды (рис. 3.3.3);

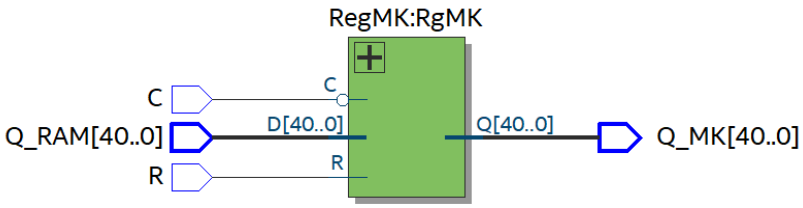

Рисунок 3.3.3 – Регистр текущей микрокоманды

Модули RegD и RegMK включают в себя модуль REG_D, который представляет собой одноразрядный регистр.

– MuxX – модуль, предназначенный для реализации мультиплексора, отвечающего за распределение выбора осведомительного сигнала (рис. 3.3.4);

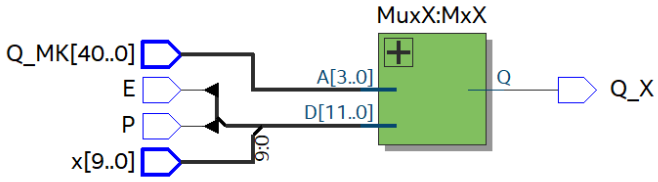

Рисунок 3.3.4 – Мультиплексор выбора осведомительного сигнала

– MuxA – модуль, предназначенный для реализации мультиплексора, с помощью которого происходит выбор адреса следующей микрокоманды (рис. 3.3.5);

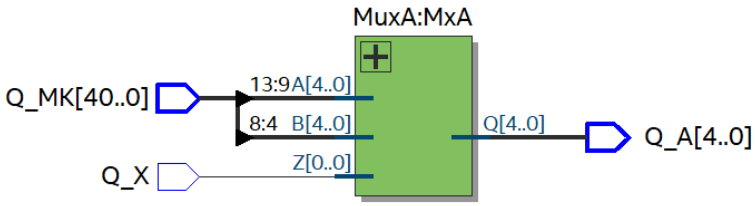

Рисунок 3.3.5 – Мультиплексор выбора адреса следующей микрокоманды

Модули MuxX и MuxA включают в себя модуль Decoder, который представляет собой дешифратор для мультиплексора.

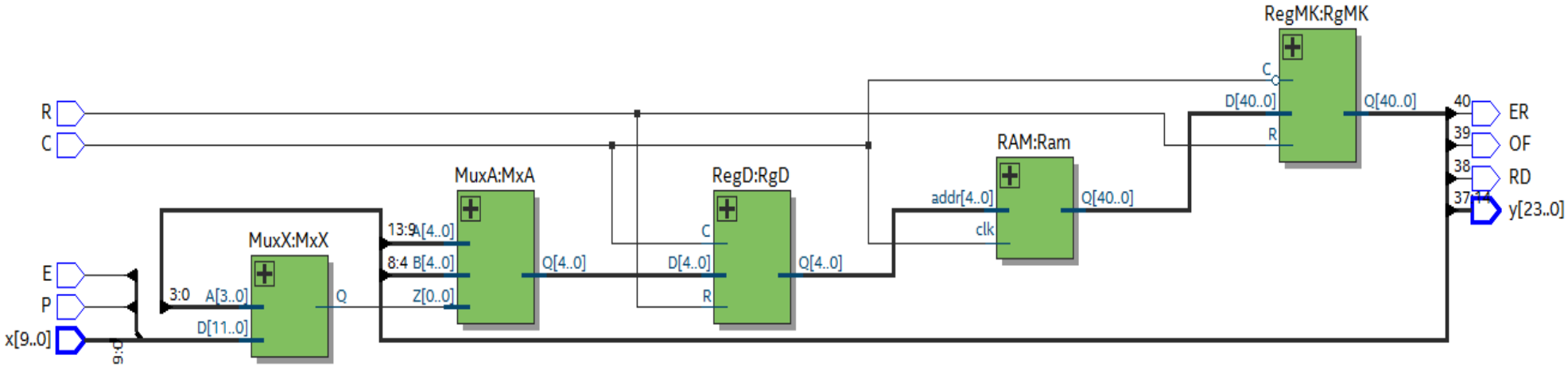

Рисунок 3.4 – Управляющий автомат спецпроцессора

3.3 Главный модуль

Для реализации спецпроцессора был создан главный модуль MyWork, представленный на рисунке 3.5. Разработанная программа для модуля MyWork приведена на CD-диске прилагающемся к ВКР.

Модуль MyWork включает в себя:

– OA – модуль, предназначенный для реализации операционного автомата;

– YA – модуль, предназначенный для реализации управляющего автомата.

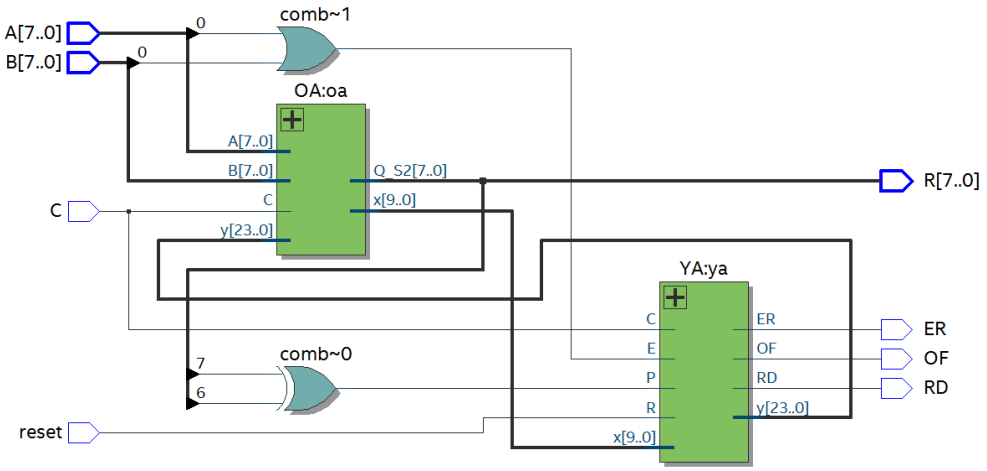

Рисунок 3.5 – Главный модуль спецпроцессора

4 Проверка работы спецпроцессора

Для проверки работы спецпроцессора был выбран один из предложенных вариантов задания на курсовое проектирование по ПТЦА, представленный в таблице 5 [10].

Таблица 5 – Задание для 54 варианта

Операция	Условие
A + 4B	A < 0, B < 0, $\lvert B \rvert$ < 0,25
2A – B	A < 0, B > 0, $\lvert A \rvert$ < 0,5
3,5 B	A > 0, $\lvert B \rvert$ < 0,5, B – четное

Чтобы проверить работу преподаватель осуществляет ввод чисел A и B, которые записываются в регистры.

При поступлении чисел A и B, представленных в двоичной системе счисления в модифицированном дополнительном коде, в блок формирования флажков, происходит формирование флажков, в соответствии с которыми требуется выполнить ту или иную арифметическую операцию. Также происходит формирование осведомительных сигналов, представляющих собой соответствующие флажки.

Перед началом работы в ОЗУ из файла записывается микропрограмма, соответствующая заданному варианту. В управляющем автомате происходит формирование управляющих сигналов, которые поступают на входы элементов операционного автомата, где, с учетом условий поставленной задачи, выполняется необходимая операция.

Результат записывается в регистр результата и выводится на экран.

4.1 Разработка граф-схемы микропрограммы

Рассмотрим пример граф-схемы, составленной на основе задания для 54 варианта.

Определение осведомительных и управляющих сигналов для варианта 54.

Осведомительные сигналы:

x1 – сигнал, представляющий собой флажок s[0] (знак числа A), который находится в разряде Q9 регистра флагов RgF;

x4 – сигнал, представляющий собой флажок s[3] ($|A| < 0{,}5$), который находится в разряде Q6 регистра флагов RgF;

x6 – сигнал, представляющий собой флажок s[5] (знак числа B), который находится в разряде Q4 регистра флагов RgF;

x7 – сигнал, представляющий собой флажок s[6] (четность числа B), который находится в разряде Q3 регистра флагов RgF;

x9 – сигнал, представляющий собой флажок s[8] (|B| < 0,5), который находится в разряде Q1 регистра флагов RgF;

x10 – сигнал, представляющий собой флажок s[9] (|B| < 0,25), который находится в разряде Q0 регистра флагов RgF;

P – сигнал проверки переполнения разрядной сетки;

E – сигнал о наличии на входах чисел A и B.

Управляющие сигналы:

y1 – запись числа с входов A7-A0 спецпроцессора в регистры RgA1 и RgA2;

y2 – запись числа с входов B7-B0 спецпроцессора в регистры RgB1 и RgB2;

y3– запись флажков в регистр флагов RgF;

y4 – RgA1 ← 2*RgA1 – сдвиг содержимого регистра RgA1 на один разряд в сторону старших разрядов (умножение числа, содержащегося в RgA1 на 2);

y5 – RgA2 ← RgA2/2 сдвиг содержимого регистра RgA2 на один разряд в сторону младших разрядов (деление числа, содержащегося в RgA2 на 2);

y6 – RgB1 ← 2*RgB1 – сдвиг содержимого регистра RgB1 на один разряд в сторону старших разрядов (умножение числа, содержащегося в RgB1 на 2);

y7 – RgB2 ← RgB2/2 сдвиг содержимого регистра RgB2 на один разряд в сторону младших разрядов (деление числа, содержащегося в RgB2 на 2);

y9 – инверсия содержимого регистра RgB1 с помощью мультиплексора MUX2;

y10 – инверсия содержимого регистра RgB2 с помощью мультиплексора MUX3;

y11 – выбор адреса в мультиплексоре MUX4 (вход Z0);

y12 – выбор адреса в мультиплексоре MUX4 (вход Z1);

y14 – выбор адреса в мультиплексоре MUX5 (вход Z1);

y16 – прибавление «1» к младшему разряду с помощью сумматора SM2;

y17 – прибавление «1» к младшему разряду с помощью сумматора SM3;

y18 – запись результата операции в регистр RgS1;

y24 – запись результата операции в регистр RgS2;

y25 (RD) – сообщение о готовности результата;

y26 (OF) – сообщение о переполнении разрядной сетки;

y27 (ER) – сообщение об ошибке.

Граф-схема микропрограммы (ГСМ) строится с использованием вершин четырех типов (начальная, операторная, условная, конечная) и линий связи. Каждому действию, осуществляемому алгоритмом, придаётся значение микрооперации. Любая микрооперация изменяет состояние функциональных узлов разрабатываемого устройства.

На рисунке 4.1 представлена граф-схема микропрограммы, составленная для примера на основе задания 54 варианта [9].

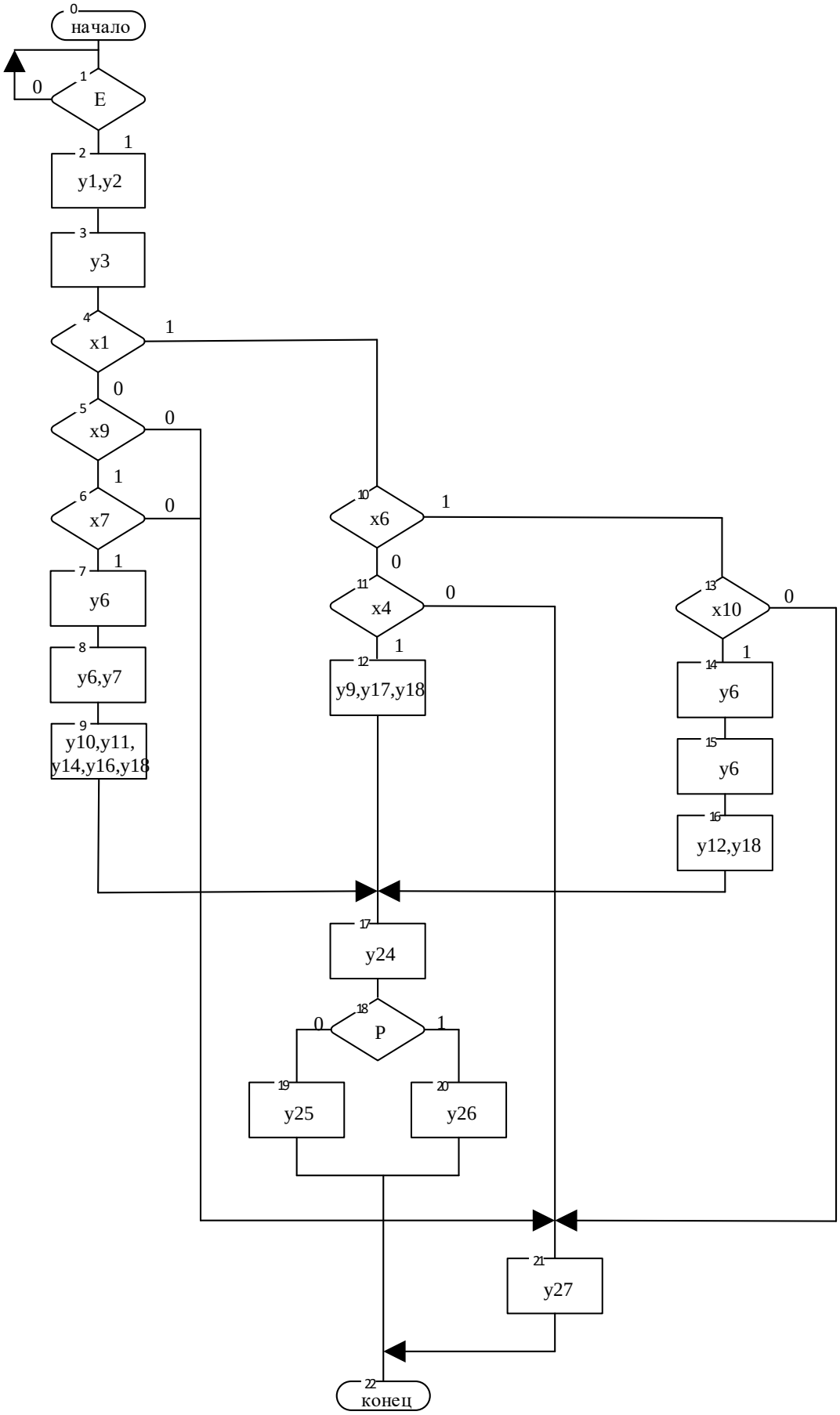

Рисунок 4.1 – Граф-схема микропрограммы для варианта 54

Словесное описание граф-схемы:

Вершина 0: Начало микропрограммы.

Вершина 1: проверка осведомительного сигнала E (наличие чисел A и B на входах специализированного процессора). При E = 1 – переход на вершину 2, иначе – переход к вершине 1;

Вершина 2: подача на входы LD регистров RgA1 и RgA2 сигнала y1, на входы LD регистров RgB1 и RgB2 сигнала y2 (запись чисел A и B в регистры RgA1, RgA2, RgB1 и RgB2). Переход к вершине 3;

Вершина 3: подача на вход LD регистра RgF сигнала y3 (запись флажков в регистр RgF). Переход к вершине 4;

Вершина 4: проверка осведомительного сигнала x1, при x1 = 1 (A < 0) – переход к вершине 10, иначе переход к вершине 5;

Вершина 5: проверка осведомительного сигнала x9, при x9 = 1 (|B| < 0,5) – переход к вершине 6, иначе переход к вершине 21;

Вершина 6: проверка осведомительного сигнала x7, при x7 = 1 (B – четное) – переход к вершине 7, иначе переход к вершине 21;

Вершина 7: выполняется сдвиг числа B в сторону старших разрядов по сигналу y6, переход к вершине 8;

Вершина 8: выполняется сдвиг числа B в сторону старших разрядов по сигналу y6 и сдвиг числа B в сторону младших разрядов по сигналу y7, переход к вершине 9;

Вершина 9: выполняется инверсия содержимого регистра RgB2 по сигналу y10, сложение с помощью сумматора SM2 содержимого регистра RgB1 и содержимого мультиплексора MUX3 по сигналу y16, в мультиплексоре MUX4 выбор нуля по сигналу y11, выбор содержимого сумматора SM2 в мультиплексоре MUX5 по сигналу y14, запись промежуточного результата в RgS1 по сигналу y18, переход к вершине 17;

Вершина 10: проверка осведомительного сигнала x6, при x6 = 1 (B < 0) – переход к вершине 13, иначе переход к вершине 11;

Вершина 11: проверка осведомительного сигнала x4, при x4 = 1 (|A| < 0,5) – переход к вершине 12, иначе переход к вершине 21;

Вершина 12: выполняется инверсия содержимого регистра RgB1 по сигналу y9 и сложение с помощью сумматора SM3 по сигналу y17, запись промежуточного результата в RgS1 по сигналу y18, переход к вершине 17;

Вершина 13: проверка осведомительного сигнала x10, при x10 = 1 (|B| < 0,25) – переход к вершине 14, иначе переход к вершине 21;

Вершина 14: выполняется сдвиг числа B в сторону старших разрядов по сигналу y6, переход к вершине 15;

Вершина 15: выполняется сдвиг числа B в сторону старших разрядов по сигналу y6, переход к вершине 16;

Вершина 16: в мультиплексоре MUX4 выбор содержимого мультиплексора MUX2 по сигналу y12, запись промежуточного результата в RgS1 по сигналу y18, переход к вершине 17;

Вершина 17: производится запись результата в регистр RgS2 по сигналу y24, переход к вершине 18;

Вершина 18: проверка осведомительного сигнала P, при P = 1 – переход к вершине 20, иначе переход к вершине 19;

Вершина 19: выдается сообщение о готовности результата по сигналу y25, переход к вершине 22;

Вершина 20: выдается сообщение о переполнении разрядной сетки по сигналу y26, переход к вершине 22;

Вершина 21: выдается сообщение об ошибке по сигналу y27, переход к вершине 22;

Вершина 22: конец программы.

4.2 Разработка микропрограммы

В соответствии с граф-схемой, представленной на рисунке 4.1, была разработана микропрограмма выполнения арифметических операций (табл. 6).

Таблица 6 – Микропрограмма для 54 варианта

Адрес ПЗУ	СХ	A1	A0	Y
0 0 0 0 0	0 0 0 0	0 0 0 0 1	0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 1	* * * *	0 0 0 1 0	0 0 0 1 0	1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 0	* * * *	0 0 0 1 1	0 0 0 1 1	0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 1 1	0 0 0 1	0 1 0 0 1	0 0 1 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 0	1 0 0 1	0 0 1 0 1	1 0 1 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 0 1	0 1 1 1	0 0 1 1 0	1 0 1 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 0	* * * *	0 0 1 1 1	0 0 1 1 1	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 1 1 1	* * * *	0 1 0 0 0	0 1 0 0 0	0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 0	* * * *	1 0 0 0 0	1 0 0 0 0	0 0 0 0 0 0 0 0 0 1 1 0 0 1 0 1 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 0 1	0 1 1 0	0 1 1 0 0	0 1 0 1 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 0	0 1 0 0	0 1 0 1 1	1 0 1 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 0 1 1	* * * *	1 0 0 0 0	1 0 0 0 0	0 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 0	1 0 1 0	0 1 1 0 1	1 0 1 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 0 1	* * * *	0 1 1 1 0	0 1 1 1 0	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 0	* * * *	0 1 1 1 1	0 1 1 1 1	0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 1 1 1 1	* * * *	1 0 0 0 0	1 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 0 0	* * * *	1 0 0 0 1	1 0 0 0 1	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 0
1 0 0 0 1	1 0 1 1	1 0 0 1 1	1 0 0 1 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
1 0 0 1 0	* * * *	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0
1 0 0 1 1	* * * *	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0
1 0 1 0 0	* * * *	0 0 0 0 0	0 0 0 0 0	0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1

Где * - это 1 или 0.

Разработанная микропрограмма заносится в ЗУ. В каждой ячейке ЗУ записана отдельная микрокоманда. Выполнение микропрограммы начинается с микрокоманды, размещенной в нулевой ячейке.

4.3 Результаты выполнения

На входы управляющего автомата поступают осведомительные сигналы, представляющие собой флажки, сформированные в операционном автомате. На выходе – управляющие сигналы, сформированные в следствие выполнения микропрограммы. Результат моделирования отображен на рисунках 4.1.1 и 4.1.2.

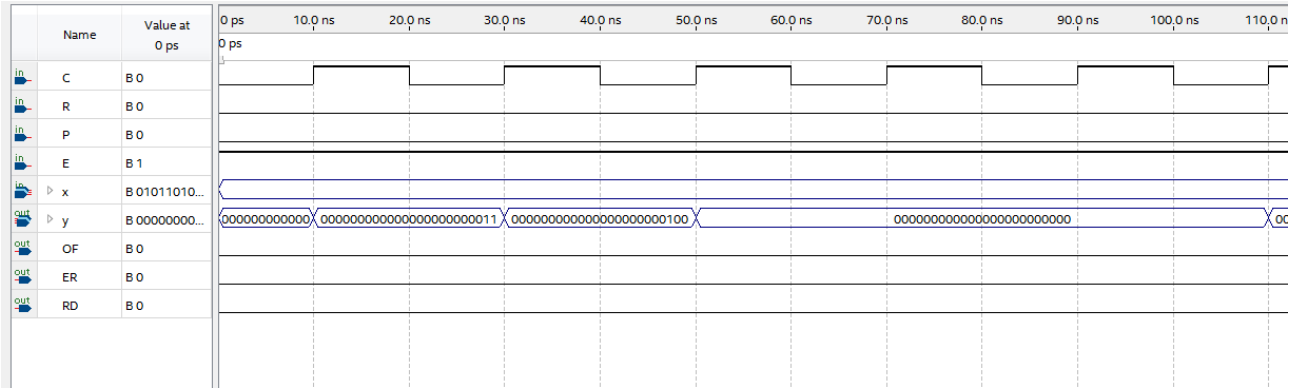

Рисунок 4.1.1 – Результаты моделирования управляющего автомата

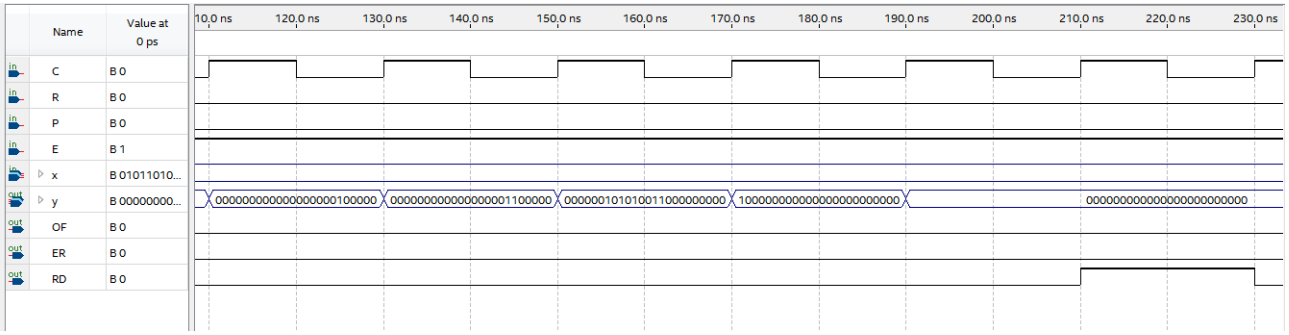

Рисунок 4.1.2 – Результаты моделирования управляющего автомата

На входы операционного автомата поступают числа A и B и управляющие сигналы, сформированные в управляющем автомате. На выходе

формируются флажки и записываются в регистр флагов RgF, а также формируется результат выполнения микропрограммы. Результат моделирования отображен на рисунке 4.2.

Исходная пара чисел *A* и *B* для проверки работы спецпроцессора:

A = 25, *B* = -26

В модифицированном дополнительном коде:

A = 00.011001, *B* = 11.100110

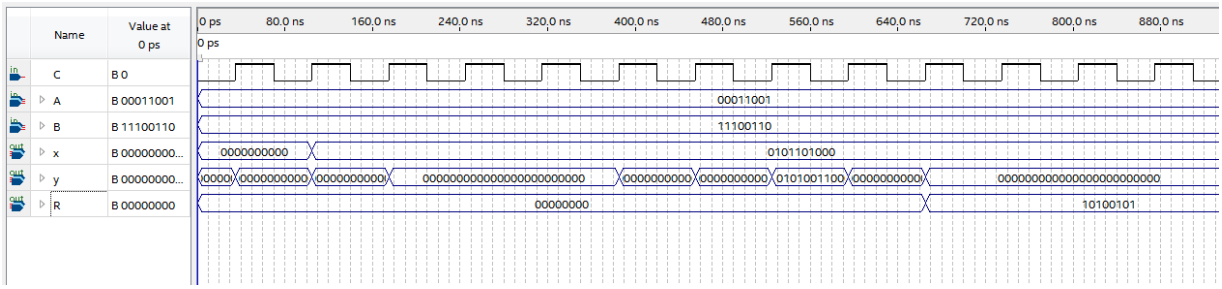

Рисунок 4.2 – Результаты моделирования операционного автомата

По результатам моделирования можно сделать выводы:

Для числа A:

Q[0] = 0, A > 0;

Q[3] = 1, $|A| < 0{,}5$;

Для числа B:

Q[5] = 1, B < 0;

Q[6] = 1, B – четное;

Q[8] = 1, $|B| < 0{,}5$.

Учитывая условия задачи, программа должна была выполнить операцию 3,5B. Если A = 25, B = –26, то

3,5B = 3,5 · (–26) = –91

В дополнительном двоичном коде: $-91_{\text{доп.}} = 10.100101$

Результат моделирования отображен на рисунке 4.3.

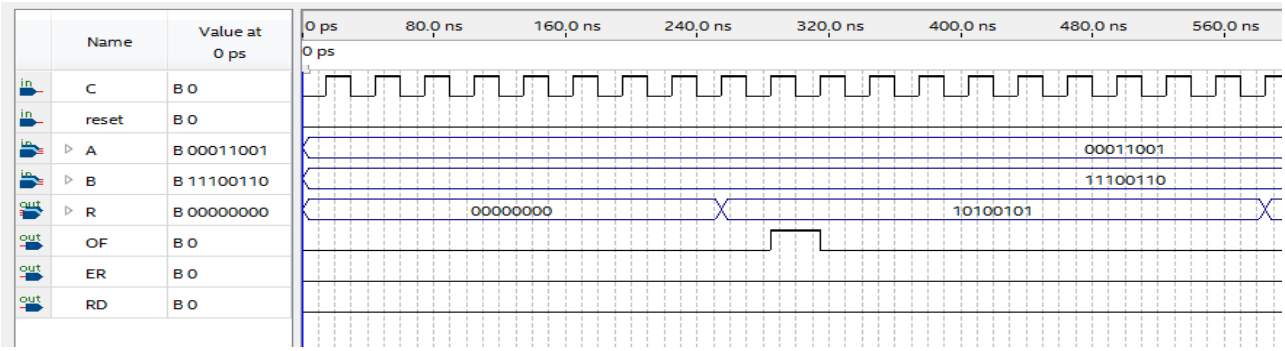

Рисунок 4.3 – Результаты моделирования главного модуля

Так как разрядность автомата – 8 разрядов, то произошло переполнение разрядной сетки, о чем свидетельствует выходной сигнал OF.

4.4 Тестирование

Также было проведено тестирование на других числах.

Для выполнения операции A + 4B необходимо соблюдение условий:

A < 0, B < 0, $|\boldsymbol{B}| < 0{,}25$

Пара чисел *A* и *B* для проверки работы:

A = -5, *B* = -11

В модифицированном дополнительном коде:

A = 11.111011, *B* = 11.110101

Программа должна была выполнить операцию A + 4B.

A + 4B = -5 + (-11*4) = – 49

В дополнительном двоичном коде: $-49_{доп.} = 11.001111$

Результат моделирования отображен на рисунке 4.4. Сигнал RD свидетельствует о готовности результата.

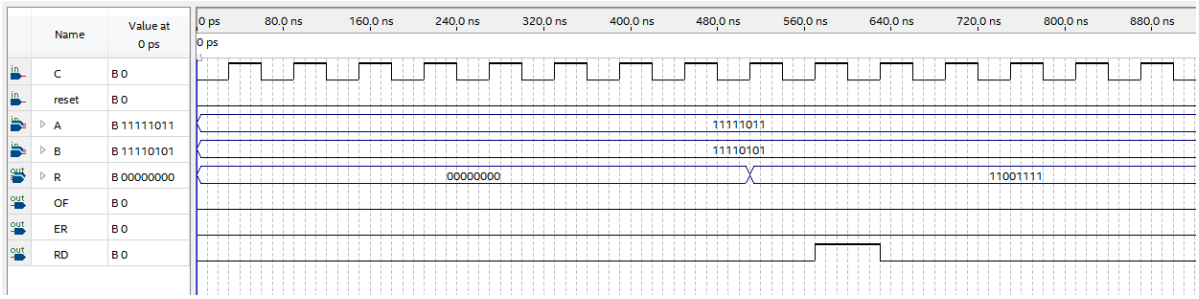

Рисунок 4.4 – Результаты моделирования для операции A + 4B

Для выполнения операции 2A - B необходимо соблюдение условий:

A < 0, B > 0, |A| < 0,5

Пара чисел *A* и *B* для проверки работы:

A = -20, *B* = 5

В модифицированном дополнительном коде:

A = 11.101100, *B* = 00.000101

Программа должна была выполнить операцию 2A - B.

2A - B = -20*2 - 5 = – 45

В дополнительном двоичном коде: $-45_{доп.} = 11.010011$

Результат моделирования отображен на рисунке 4.5. Сигнал RD свидетельствует о готовности результата.

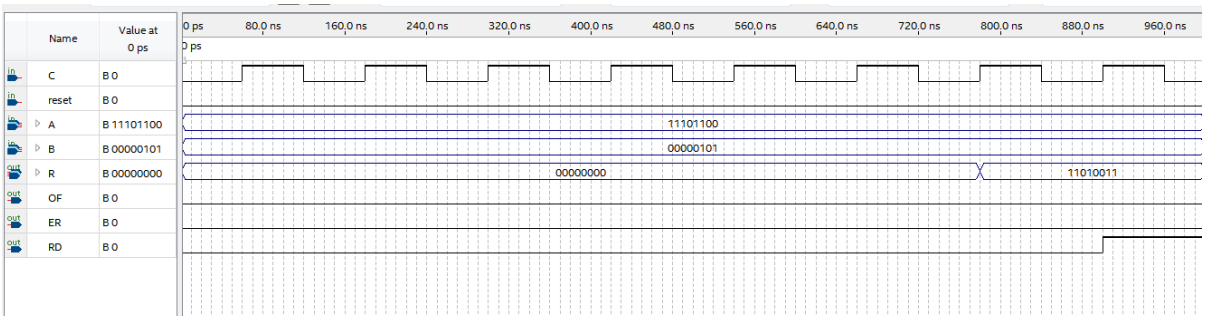

Рисунок 4.5 – Результаты моделирования для операции 2A – B

Для появления ошибки необходимо, чтобы ни одно из представленных условий не выполнилось.

Пара чисел *A* и *B* для проверки работы:

A = 10, *B* = 35

В модифицированном дополнительном коде:

A = 00.001010, B = 00.100011

Результат моделирования отображен на рисунке 4.6.

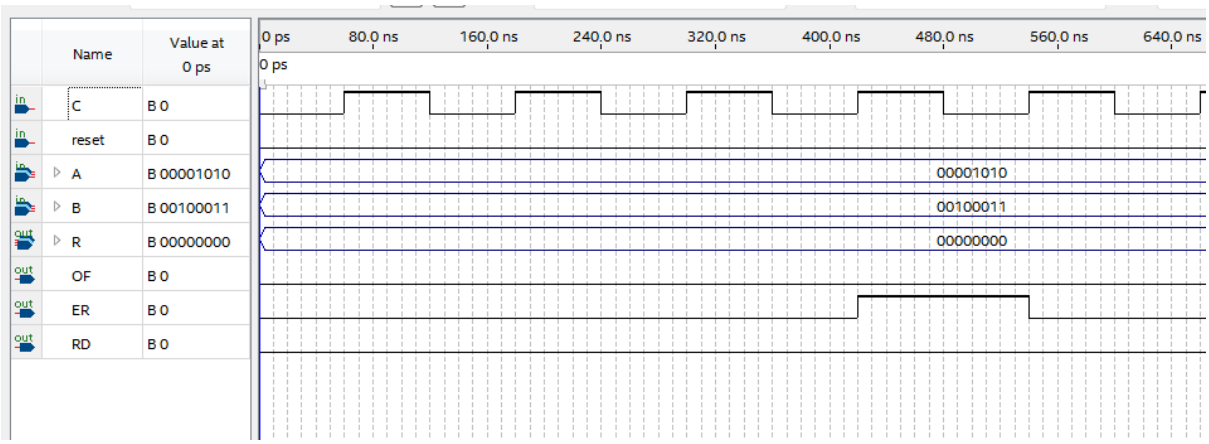

Рисунок 4.6 – Результаты моделирования для генерации ошибки

Так как ни одно из условий не выполнилось, произошла ошибка выполнения, о чем свидетельствует сигнал ER.

4.5 Выводы по главе

1. Реализован специализированный процессор, предназначенный для выполнения арифметических операций

2. Проведено тестирование выполнения микропрограммы, разработанной для проверки работы спецпроцессора.

ЗАКЛЮЧЕНИЕ

Результатом выполнения данной работы является разработанный специализированный процессор, состоящий из двух автоматов – операционного и управляющего, и выполняющего заданную арифметическую операцию (операцию арифметического сложения в дополнительном коде двоичных чисел представленных в модифицированном коде с плавающей точкой).

В процессе выполнения были решены следующие задачи:

1. Рассмотрены принципы построения процессоров ЭВМ как сочетания операционного и управляющего автоматов.
2. Выполнен обзор предметной области и проведен анализ вариантов задания на курсовое проектирование, в ходе которого составлен список необходимых флажков, в соответствии с которыми требуется выполнить ту или иную арифметическую операцию, а также составлена система команд спецпроцессора.
3. Разработана функциональная схема спецпроцессора.
4. Выполнено проектирование специализированного процессора на ПЛИС.
5. Реализован спецпроцессор.
6. Выполнено тестирование разработки.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Прикладная теория цифровых автоматов [Электронный ресурс]: – Режим доступа: https://e.sfu-kras.ru/mod/resource/view.php?id=500093
2. Кузьменко, Н.Г.Аппаратные средства вычислительной техники. Микропроцессоры: учеб. пособие / Н.Г. Кузьменко, А.И. Постников, Н.Г. Кузьменко. – Красноярск: Сиб. федер. ун-т, 2012. – 416 с.
3. Виды популярных архитектур процессоров [Электронный ресурс]: – Режим доступа: https://tproger.ru/articles/processors-architectures-review/
4. Операционный автомат [Электронный ресурс]:– Режим доступа:http://dic.academic.ru/dic.nsf/ruwiki/1426933
5. Типовые микрооперации [Электронный ресурс]: – Режим доступа: https://studfile.net/preview/1695238/
6. Управляющий автомат [Электронный ресурс]: – Режим доступа: http://dic.academic.ru/dic.nsf/ruwiki/1425805
7. Максфилд К, Проектирование на ПЛИС. Курс молодого бойца. — М.: Издательский дом «ДодэкаXXI». — 408 с.:
8. Обзор основных производителей ПЛИС [Электронный ресурс]: – Режим доступа: http://www.zolshar.ru/tech/93
9. ГОСТ 2.743-91 "Обозначения условные графические в схемах. Элементы цифровой техники"
10. СТО 4.2–07–2014 "Система менеджмента качества. Общие требования к построению, изложению и оформлению документов учебной деятельности"

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Вычислительная техника
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой
_____ _____
подпись инициалы, фамилия
« _____» _____ 20 ___ г.

БАКАЛАВРСКАЯ РАБОТА

09.03.01 Информатика и вычислительная техника
код и наименование направления

Специализированный процессор на ПЛИС
тема

Руководитель	_____ подпись, дата	доцент, к.т.н. должность, ученая степень	А.И. Постников инициалы, фамилия
Выпускник	_____ подпись, дата		Д.Н. Щукина инициалы, фамилия
Нормоконтролер	_____ подпись, дата	доцент, к.т.н. должность, ученая степень	А.И. Постников инициалы, фамилия

Красноярск 2020