Оглавление

Введение

Человек движется в сторону автоматизации рутинной работы. Некоторые математические операции могут сводиться к выполнению однотипных действий, которые будут решаться человеком монотонно с большим риском совершить ошибку в вычислениях. Поэтому учёные математики придумали ЭВМ (электронную вычислительную машину): она позволяет автоматизировать вычисления, сведя их к определённому алгоритму, который будет выполняться машиной.

Калькулятор представляет собой небольшое устройство, которое позволяет производить простые арифметические операции (сложение, вычитание, умножение, деление). Калькулятор может производить вычисления быстро и с высокой точностью, что облегчает прикладную работу человека.

Все вычислительные устройства создаются на базе микросхем различной степени интеграции. Каждая микросхема может выполнять определённую функцию. Из простых схем собирается более сложная схема, которая выполняет уже более сложную функцию.

В этом курсовом проекте будет разработано схемотехническое устройство, выполняющее функции калькулятора. Калькулятор сможет выполнять одну алгебраическую и одну логическую операцию.

Синтез устройства может выполняться «снизу-вверх», т. е. одно большое устройство будет разбито (декомпозировано) на несколько более простых устройств, связанных друг с другом определёнными связями.

Перед нами будут представлены задачи:

- Разбиения вычислительного устройства «калькулятора» на более простые структурные блоки, из которых будет собрана структурная схема устройства
- Формирование связи между блоками устройства и синтез функциональной схемы
- Выбор серии микросхем для синтеза устройства, которых удовлетворял бы заданным ограничениям
- Построение принципиальной схемы устройства

1. Теоретическая часть

Схемотехника – это техническая дисциплина, которая занимается синтезом и анализом различных электронных схем. В схемах идёт передача различных сигналов между электронными компонентами. Сигналы могут быть во времени непрерывными величинами (аналоговыми), или они могут принимать строго определённые значения из заранее известного набора (дискретные).

В цифровой схемотехнике в качестве сигналов приняты импульсы, значения которых условно принимают как логический «0» или логическая «1». Импульс – это кратковременный всплеск электрического напряжения или тока. В цифровой схемотехнике обычно применяют прямоугольные импульсы. Активный уровень сигнала – это логический уровень, который соответствует приходу сигнала.

У каждого цифрового сигнала есть следующие параметры:

- Положительный фронт сигнала – это переход сигнала из «0» в «1»
- Отрицательный фронт сигнала – переход из «1» в «0»
- Передний фронт сигнала – переход из пассивного уровня в активный
- Задний фронт сигнала – переход из активного уровня в пассивный

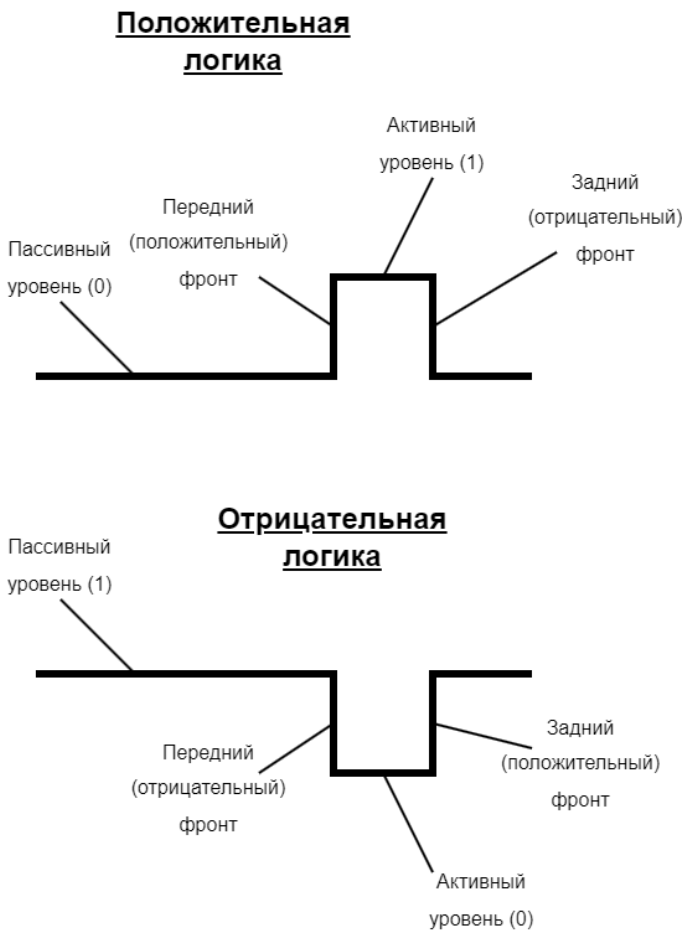

Рисунок 1. Описание цифрового сигнала

Управление электронным компонентом по средствам импульсов может происходить по нескольким сценариям.

Потенциальный способ управления заключается в том, что значения логического «0» и «1» соответствуют определённому значению напряжения или силы тока в импульсе. Следовательно, различают положительную и отрицательную логику, где высокому уровню напряжения соответствует логическая «1» (положительная логика) или логический «0» (отрицательная).

Импульсный способ управления заключается в том, что электронный компонент реагирует на изменение значения напряжения или силы тока, т. е. на переход с одного логического уровня на другой (управление по фронту).

Любой сигнал передаётся в различных средах с различными шумами и помехами, которые мешают анализу сигнала. Благодаря тому, что цифровые сигналы представлены импульсами, значение амплитуд которых соответствуют «0» или «1», цифровые сигналы устойчивей в помехам, в сравнение с аналоговыми. Защи-

та сигналов от различных помех, шумов и наводок идёт за рамки данного курсового проекта, поэтому они не будут рассмотрены.

Цифровые электронные устройства можно рассматривать как 3-и модели: логическая, электрическая и модель с временной задержкой.

Логическая модель описывает алгоритм работы цифрового устройства и таблицу истинности его. Таблица истинности – это специальная таблица, которая показывает, как меняется выходной сигнал устройства, если на его вход подавать различные логические сигналы. В правой части таблицы составляются наборы булевой (логических) переменных, от изменения которых зависит значение выходного сигнала. В правой части таблицы показывается выходной сигнал устройства.

J	K	Q
0	0	Q
0	1	0
1	0	1
1	1	$\overline{Q}$

Рисунок 2. Таблица истинности JK-триггера

Модель с временной задержкой показывает, сколько времени нужно устройству, чтобы на выходе получить тот или иной сигнал. Это связано с тем, что сигнал перемещается по своему носителю (проводам, трассам печатной платы и подобное) не мгновенно: сигнал, так или иначе, затрачивает некоторое время для перехода из точки «А» в «В» и для своего преобразования. Стоит также учесть то, что прямоугольные импульсы не совсем прямоугольные: хотя фронт переходов с одного логического уровня на другой выглядит как вертикальная прямая линия, она, на самом деле, немного наклонена, что свидетельствует о не мгновенном переходе между уровнями.

Электрическая модель показывает электронные характеристики устройства, такие как напряжение питания, входное напряжение для логического «0» и «1»,

выходное напряжение для логического «0» и «1», а также другие электрические характеристики.

Инженеры и учёные создавали цифровые электронные устройства на определённой элементной базе. Эта элементная база менялась со временем в ходе развития технологий. Главная характеристика любых электронных схем – это плотность упаковки (количество элементов на $см^3$).

На основании плотности упаковки можно определить следующие поколения электроники:

- Ламповая электроника
- Транзисторная электроника
- Интегральные схемы (Малые, средние и большие интегральные схемы)
- Современные электронные схемы (СБИС, ССБИС, УБИС)

В каждом поколением плотность упаковки увеличивалась, т. е. увеличивалось количество элементов на единицу объёма.

В зависимости от элементной базы электронных схем можно выделить их семейства. Каждое из семейств обладает своими положительными сторонами и отрицательными. Какие из семейств в наши дни практический не используются, так как они считаются устаревшими, какие-то из семестр практически повсеместно распространены с современных электронных устройства, а какие-то семейства обладают своеобразными характеристиками и применяются в узкоспециализированных областях.

Основные семейства цифровых электронных схем:

- ТЛНС (Транзисторная логика с непосредственными связами)
- ДЛ (Диодная логика)
- ЭСЛ (Эмиттерно-связанная логика на биполярных транзисторах)
- ТТЛ (Транзисторно-транзисторная логика)
- ТТЛШ (ТТЛ с диодами Шоттки)
- МОП-логика (логика на однотипных МОП-транзисторах)

- КМОП-логика (логика на взаимодополняющих МОП-транзисторах)
- БиКМОП-логика (логика на биполярных и КМОП транзисторах)
- И другие

Существую ещё различные семейства микросхем, но они перечислены не будут в связи с их меньшим распространением относительно самых основных семестр. В пояснительной записке будут рассмотрены основные семейства микросхем, которые достаточно широко применялись в прошлом и применяются в наши дни: ТТЛ, ТТЛШ и КМОП.

Перед рассмотрением основных семестр микросхем стоит рассказать, что было до логики на базе транзисторов и что представляют собой транзисторы.

Первоначально в электронике применялись схемы на базе диодной логики, которые были сконструированы на базе диодов, электрических элементов, которые пропускают ток в одну сторону и не пропускают в другую. Это семейство широко применялось в ранние годы развитие цифровой техники. К недостаткам этой технологии можно отнести низкое быстродействие, высокое напряжение питания на выходе (напряжение на выходе и входе значительно отличались) и то, что в одном элементе могло быть максимум 8-ем входов (по одному на каждый диод). Со временем эта технология была заменена на различные технологии на базе различных транзисторов.

Первоначально стоит объяснить, что есть транзистор. Транзистор – это активный электронный компонент, сигналы первичной цепи которой управляют вторичной цепью за счёт поступления энергии из внешнего источника питания. Говоря упрощённо, с точки зрения аналоговых сигналов: транзистор – это управляющий резистор, величина тока которого зависит от величины тока/напряжения на базе/затворе, - а с цифровой точки зрения: транзистор – это ключ (переключатель), который в зависимости от тока на базе/затворе либо проводит ток, либо не проводит (соответствует логическому «0» или «1»). Транзисторы изготавливаются из полупроводников, и их принцип работы основан на зонной теории. Подробнее устройство транзистора рассмотрено не будет.

Транзисторы бывают биполярными и униполярными (полевыми).

Биполярный транзисторы состоят из трёх слоёв полупроводников разных типов проводимости (либо n-проводимость, либо p). Эти слои называют эмиттером, коллектором и базой. Различают pnp- и npn-транзисторы. Такие транзисторы применяют, в первую очередь, в таких технологиях изготовления микросхем как ТТЛ и ТТЛШ.

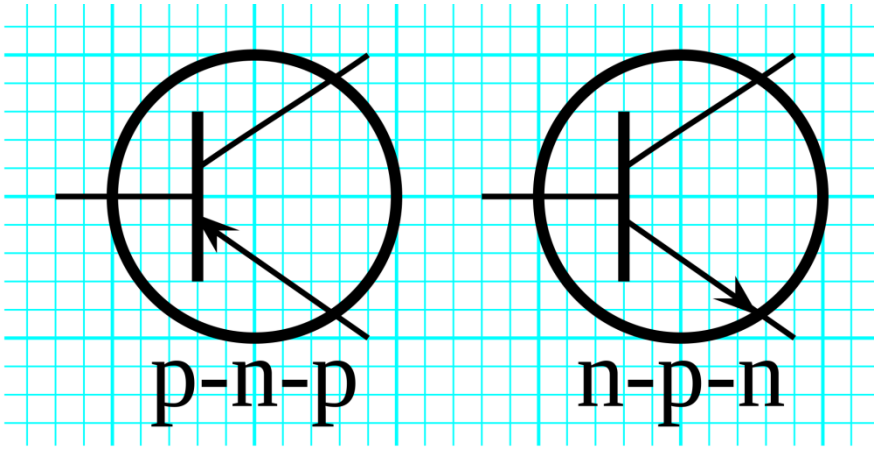

Рисунок 3. Графическое обозначение биполярных транзисторов

Технология ТТЛ (Транзисторно-транзисторная логика) создана на базе многоэмиттерных биполярных транзисторов. Входом считаются эмиттеры многоэмиттерного транзистора (VT1 на схеме). Коллектор многоэмиттерного транзистора подключён к обычному биполярному транзистору (VT2), который выполняет функцию инвертирующего усилителя.

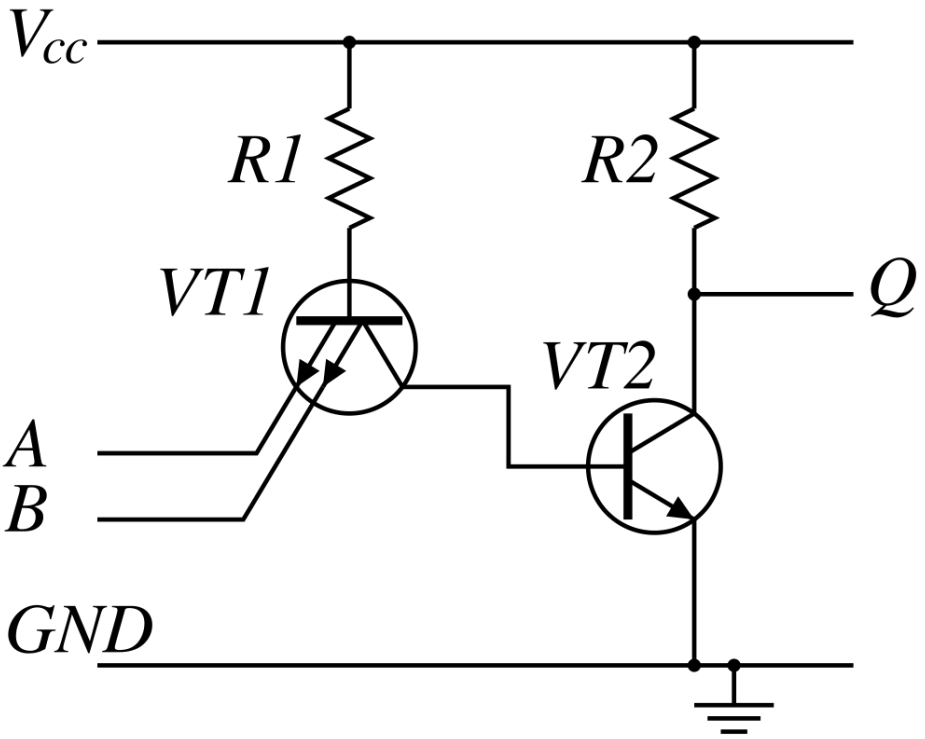

Рисунок 4. Логический элемент «2И-НЕ» на базе технологии ТТЛ.

Достоинства технологии ТТЛ:

- Простота и дешевизна
- Относительно высокое быстродействие
- Среднее энергопотребление

Недостатки технологии ТТЛ:

- Малый коэффициент разветвления
- Более низкая помехоустойчивость по сравнению с более старыми технологиями

Микросхемы этой технологии обладают высоким быстродействием, но также повышенным энергопотреблением. Несмотря на это, было разработано много микросхем этой технологии, и они достаточно долго применялись в микроэлектронике. Данная технология была заменена на ТТЛШ.

ТТЛШ – это технология ТТЛ модифицированная диодами и транзисторами Шоттки. Транзисторы Шоттки – это обычные биполярные транзисторы, между базой и коллектором которых подключён диод Шоттки. Диоды Шоттки используют переход «металл-полупроводник», в отличие от обычных диодов с перехо-

дом «полупроводник-полупроводник», что обеспечивает малое падение напряжения на переходах и высокое быстродействие.

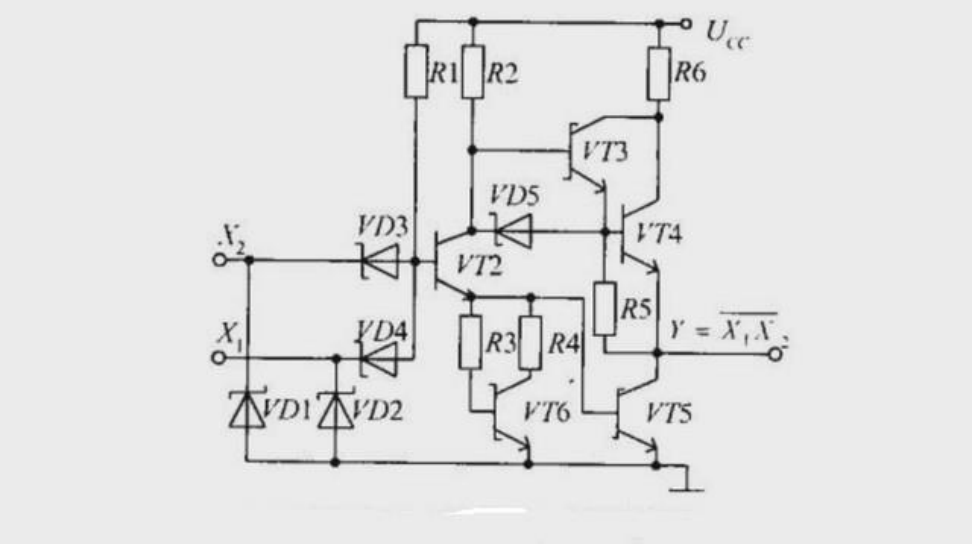

Рисунок 5. Логический элемент «2И-НЕ»на базе технологии ТТЛШ

ТТЛШ быстрее ТТШ в 3-5 раз, при более низком энергопотреблении (до 4 раз). Это обеспечило последующее замену элементной базы в микроэлектронике с ТТЛ на ТТЛШ. Несмотря на это, ТТЛШ микросхемы обладают более низкой помехоустойчивостью.

Полевые (униполярные) транзисторы – это транзисторы с одним переходом (либо n-, либо p-). Принцип работы таких транзисторов заключается в том, что управляющий ток управляет поперечным сечением «канала»: чем больше сечение, тем больших ток проходит по транзистору. В отличие от биполярных транзисторов контакты полевых транзисторов носят другие имена: затвор (база в биполярных транзисторах) – вывод, управляющий током, исток (эмиттер) – вывод, откуда поступает ток, и сток (коллектор) – вывод, куда идёт ток.

Полевые транзисторы могут отличаться по конструкции. Самые простые полевые транзисторы имеют один переход (JFET, англ. junction-gate field-effect transistor).

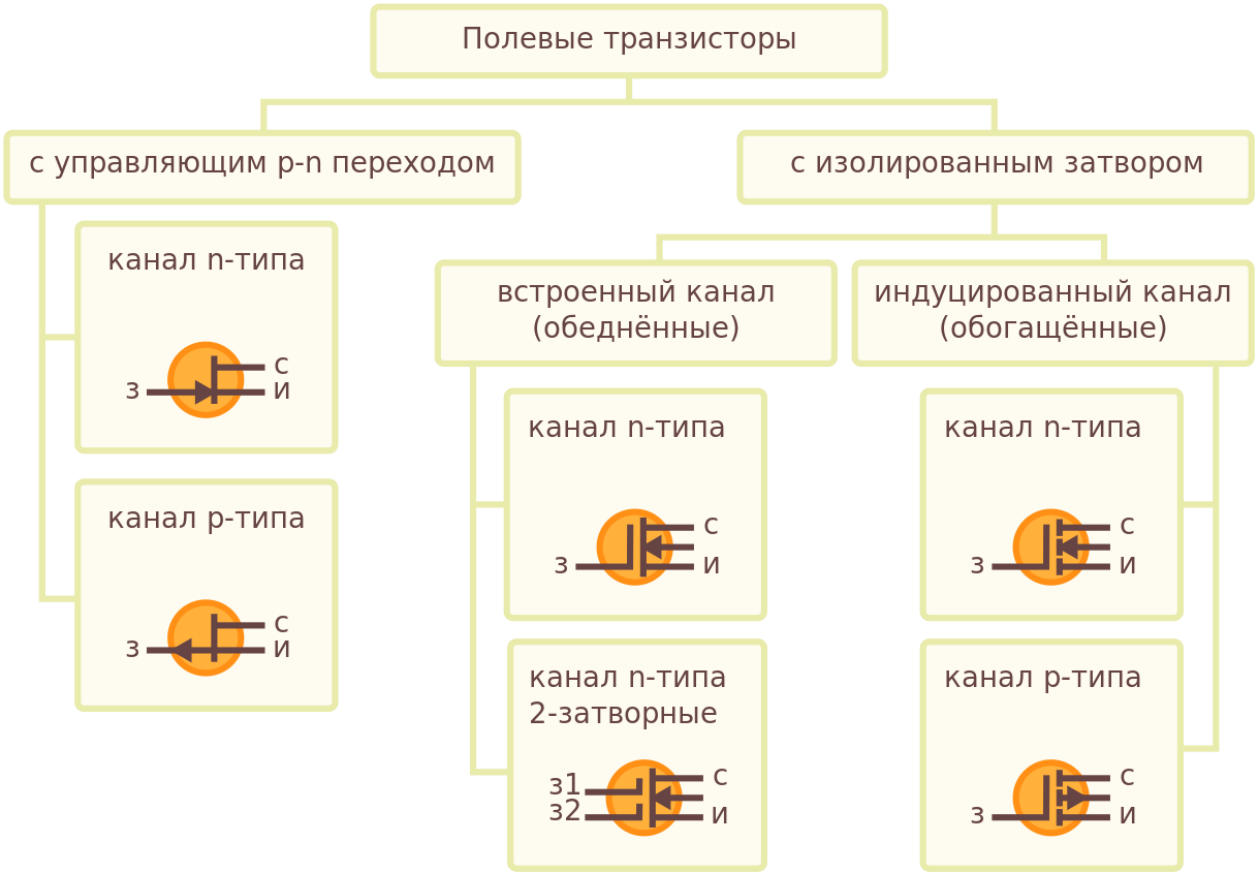

Рисунок 6. Классификация полевых транзисторов

MOSFET (metal-oxide-semiconductor field-effect transistor) или транзисторы с изолированным затвором – это полевые транзисторы, где затвор изолирован от остальных выводом специальным материалом. Такие транзисторы называют либо МДП (металл-диэлектрик-полупроводник), либо МОП (металл-оксид-полупроводник, так как в качестве диэлектрика использую диоксид кремния SiO2).

Полевые транзисторы с изолированным затвором делятся на транзисторы со встроенным каналом, где между истоком и стоком есть канал, по которому проходят электроны, и транзисторы с индуцированным каналом (индицирование – возбуждение в объекте какого-либо свойства под определённым воздействием), где канал между стоком и истоком возникает под воздействием электрического поля затвора.

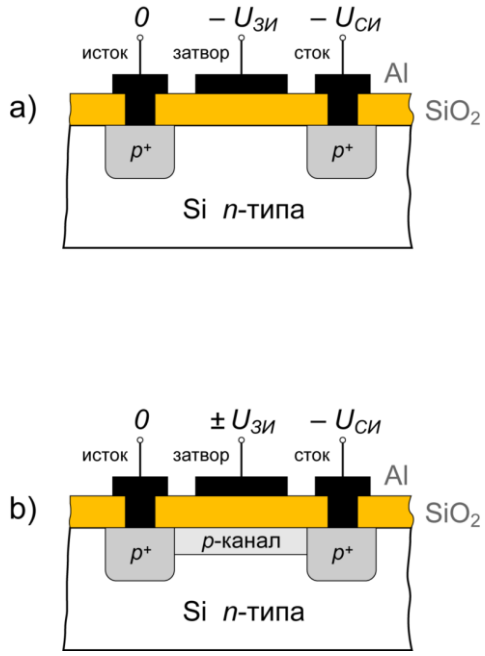

Рисунок 7. Конструкция MOSFET с индуцированным (a) и со встроенным (b) каналом

КМОП технология микросхем – это технология изготовления микросхем с использованием комплементарных МОП-транзисторов, что значит, что МОП-транзисторы p- и n-переходов взаимно дополняют друг друга (свойство комплементарности). К особенностям этой технологии можно отнести:

- Малый ток потребления (в мкА, следовательно мощность потребления в мкВт)
- Малый размер транзисторов на кристалле
- Большое входное сопротивление
- Малое падение напряжения на отрытых транзисторах (в районе мвВ, следовательно можно использовать источники питания в 1 В)

Но эта технология не лишена и недостатков:

- Малое усиление по напряжению
- Быстродействие может быть слегка ниже, чем у технологий на базе биполярных транзисторах
- Трудность в эксплуатации в экстремальных условиях

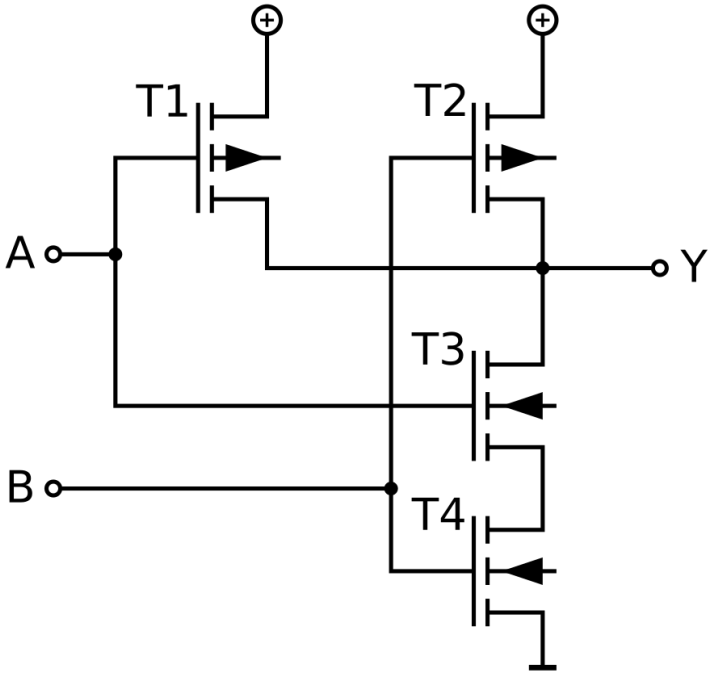

Рисунок 8. Логический элемент «2И-НЕ» на базе технологии КМОП.

Для того чтобы лучше ориентироваться в многообразии микросхем, инженеры разработали специальные буквенно-числовые обозначения для интегральных схем – серии микросхем.

Серия микросхем состоит из нескольких частей, которые обозначают те или иные особенности и характеристики схемы. В основном будут рассмотрены серии микросхем, разработанные американской компанией Texas Instruments, так как разработка принципиальной схемы вычислительного устройства будет происходить в американском программном обеспечении «Proteus». Российские аналоги американских схем в американской программе не предусмотрены, но, в принципе, это не является особой проблемой, так как можно сопоставить используемые американские схемы с российскими аналогами.

Таблица 1. Обозначение серий микросхем фирмы Texas Instrumets

Обозначение серии микросхем фирмы Texas Instruments										
Обозначение	SN	74	LVC	H	16	2	244	A	DGG	R
№	1	2	3	4	5	6	7	8	9	10

Рассмотрим обозначение:

1) Префикс производителя

2) Температурный режим (военная или коммерческая схема)

3) Серия микросхем (технология изготовления)

4) Специальная функция

5) Кол-во обрабатываемых бит

6) Дополнительная опция

7) Функциональное назначение схемы

8) Версия прибора

9) Используемый корпус

10) Нумерация

Одним из важных блоков любого вычислительного устройства является устройство управления (УУ), которое связывает все остальные структурные блоки устройства между собой. Формирование устройства управления тесно связано с теорией автоматов и идеей управления автоматами.

Один большой сложный автомат (устройство) можно представить в виде двух автоматов: операционный автомат и управляющий автомата. Операционный автомат (ОА) – это автомат, который обрабатывает входные сигналы, образуя выходы и признаки. Каждая операция в ОА является неделимой. Из-за этого её называют микрооперацией. Каждый такт времени ОА выполняет одну микрооперацию, переходя в другое состояние. Управляющий автомата (УА) – это автомата, который каждый такт времени образует определённый выходной сигнал, который соответствует микрооперации, которую должен выполнить ОА. Последовательность микроопераций, которые должен выполнить ОА, называется микропрограмма.

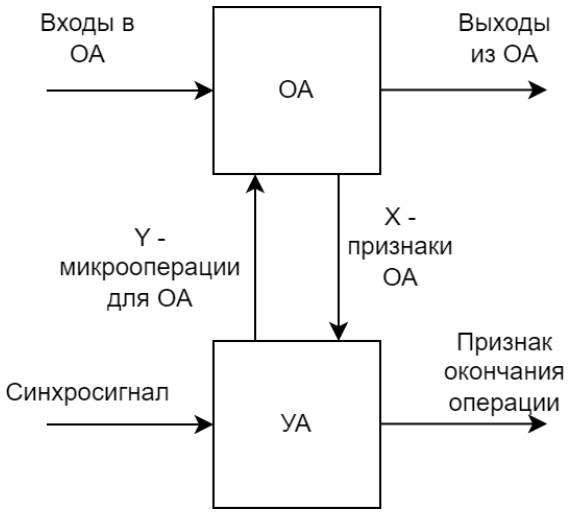

Рисунок 9. Структурная схема взаимодействия операционного автомата (ОА) и управляющего автомата (УА)

С помощью разделения устройства на операционную часть и устройство управления можно реализовать в ЭВМ операции деления и умножения, как последовательность определённых микроопераций, некоторые из которых выполняются при соблюдении определённых условий.

Так, например, в качестве операционного автомата можно взять АЛУ, которое может выполнять определённые микрооперации и образовывать выходные признаки. Можно создать устройство управления, которое будет анализировать признаки ОА и на их основе создавать такие выходные сигналы, которые будут приводить к выполнению определённой микрооперации. Таким образом будет реализовываться одна большая сложная операция.

Есть два подхода к созданию управляющего автомата:

1) Управляющий автомат с жёсткой логикой, где выходные сигналы создаются с помощью комбинационных схем, что позволяет создать быстрый автомат, но такое УУ сложно менять, так как это требует изменения структуры автомата.

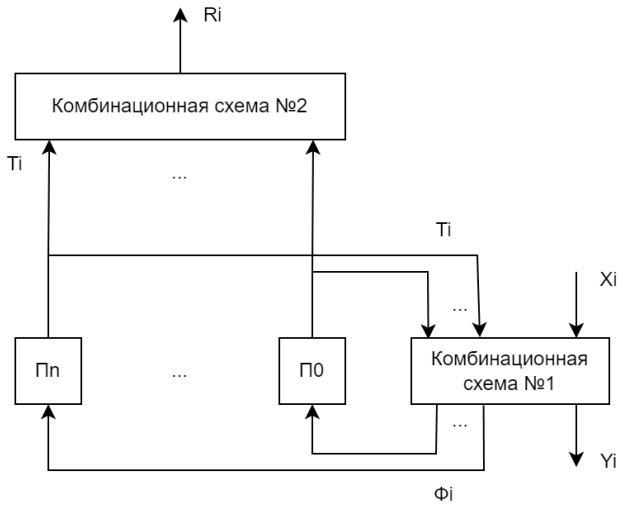

Рисунок 10. Функциональная схема совмещённого автомата.

2) Управляющий автомата с программируемой логикой, где используется специальное запоминающее устройство (ЗУ), которое содержит микропрограмму, и регистр, который хранит состояние автомата; состояние автомата и признаки ОА подаются на вход ЗУ; на выходе такого автомата выводится содержимое ячейки памяти ЗУ, в котором хранится следующее состояние автомата и выходы УА, предназначенные для ОА. Такой автомат работает медленнее, но он является более гибким, так как изменить микропрограмму в ЗУ проще, чем изменить комбинационную схему автомата.

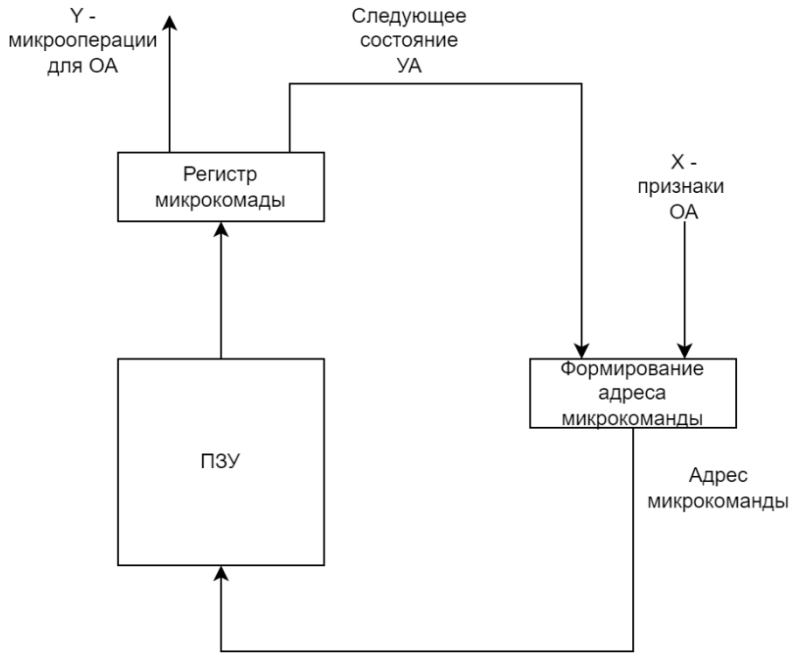

Рисунок 11. Структурная схема управляющего автомата с программируемой логикой.

2. Специальная часть.

2.1. Задание на курсовой проект.

Спроектировать вычислительное устройство для выполнения заданных функций в соответствии с техническим заданием.

Вариант **25.**

- Разрядность операндов: 24
- Логическая операция: $X \vee Y$
- Арифметическая операция: $X + Y$
- Кодировка ввода чисел: Десятичная
- Принцип управление устройством: Микропрограммный
- Быстродействие устройства: $t_{\text{задержки}} = 400$ нс
- Мощность потребления устройства: $P_{\text{потребления}} = 300$ мВт

Т. е. задание на курсовой проект звучит так:

Разработать цифровое устройство для выполнения логической операции $X \vee Y$ и арифметической операции $X + Y$ над 24-разрядными операндами X и Y. Ввод чисел с клавиатуры выполняется в десятичной кодировке (клавиатура имеет кнопки «0», «1», …, «8», «9»). Принцип управления устройством – микропрограммный. Быстродействие устройства определяется исходя из максимально допустимой задержки $T_{зад} < 400$ нс. Максимальная мощность потребления устройства $P_{потр} < 300$ мВт.

2.2. Обоснование выбора схемы технологий микросхем

В данном курсовом проекте будут использовании микросхемы, произведённые по КМОП-технологии («Комплементарная структура Металл-Оксид-Полупроводник», англ. CMOS). Причиной этого является то, что КМОП-технология на базе униполярных транзисторах обладает лучшими характеристиками, чем технологии на базе биполярных транзисторах (ТТЛ и ТТЛШ).

В проекте необходимо разработать принципиальную схему цифрового арифметико-логического устройства, уложившись в ограничение по мощности потребления электроэнергии и быстродействию схемы. При этом в данном варианте задания необходимо разработать устройства, которое будет управляться с помощью автомата с программируемой логикой (микропрограммное управление), что значит наличие в схеме запоминающего устройства, которое будет хранить микропрограмму для управления разрабатываемым устройством. Обычно запоминающие устройства потребляют достаточно много электроэнергии. Также автоматы с микропрограммным управлением работают медленней, чем аппараты с аппаратным управлением, где управляющие сигналы создаются с помощью преобразований в логических схемах. Всё это создаёт очень серьёзные ограничения в плане оптимизации схемы: необходимо минимизировать и скорость работы схемы (быстродействие), и мощность потребление схемы при наличии микропрограммного управления.

Сравним потребляемую мощность и время задержки для серий КМОП и ТТЛ(Ш) микросхем фирмы Texas Instruments:

Таблица 2. Сравнение параметров серий микросхем фирмы Texas Instruments

Параметры	Серии															
	CMOS									TTL			TTLS			
	C	AC	ACT	AHC	AHCT	HC	HCT	ALVC	LVC	-	H	L	S	LS	ALS	AS
	CMOS	Advanced		Advanced High-Speed		High-Speed		Advanced Low-Voltage	Low-Voltage	TTL	High-Speed	Low-Power	TTLS	Low-Power	Advanced Low-Power	Advanced
t_{pd}, нс	60	8		5,5	6,9	15		3	6	22	10	60	5	15	11	4,5
V_{cc}, В	3-15	2-6	5	2-5,5	5	2-6	5	1,65-3,6	2-3,6	5	5	5	5	5	5	5
I_{OL}, мА	0,36	24		8		4	4,8	24	24	16	20	3,6	20	8		20
I_{HL}, мА	-0,36	-24		-8		-4	-4,8	-24	-24	-0,4	-0,5	-0,2	-1	-0,4		-2

Где t_{pd} – максимальное время задержки, V_{cc} – напряжение питания, I_{OL} – сила тока на выходе при «0», I_{HL} – сила тока на выходе при «1»

На основании характеристик серий микросхем можно сказать, что у серий на базе технологии КМОП в среднем более быстрые (3-8 нс), и они имеют более низкое напряжение питания (2-6 В) и, следовательно, более низкое энергопотребление, чем у серий на базе ТТЛ/ТТЛШ (питание: 5 В; скорость: 4,5-15 нс).

Все эти ограничения являются причиной того, что в данном курсовом проекте будут использоваться схемы на базе КМОП технологии. Эта технология обеспечивает более низкое энергопотребление и более высокое быстродействие, чем микросхемы на базе технологии ТТЛ или ТТЛШ. Хорошо подобранные серии КМОП микросхем позволят разработать такую схему устройства, которая сможет уложиться в пределы по энергопотреблению и быстродействию

2.3. Выбор серии микросхем.

В данном курсовом проекте будут использоваться серии микросхем фирмы Texas Instruments, так как разработка принципиальной схемы вычислительного устройства будет осуществляться в американском программном обеспечении Proteus, где отсутствуют российские аналоги этих микросхем.

Будут использованы следующие серии фирмы Texas Instruments:

- SN74AC (коммерческая усовершенствованная КМОП)
- SN74ACT
- SN74AHC (коммерческая усовершенствованная высокоскоростная КМОП; в 3-е быстрее и энергопотребление в 2-а раза меньше, чем у SN74HC)
- SN74AHCT (совместимая с ТТЛ SN74AHC)
- SN74ALVC (коммерческая усовершенствованная низковольтная КМОП)
- SN74HC (коммерческая высокоскоростная КМОП; низкое энергопотребление, скорость как у LS)
- SN74HCT (совместимая с ТТЛ SN74HC)
- SN74LVC (коммерческая низковольтная КМОП)

Несколько серий схем были выбраны для того, чтобы обеспечить схему лучшими характеристиками в плане энергопотребления и быстродействия проектируемого устройства. В определённых блоках устройства будут использоваться определённые микросхемы определённой серии. Благодаря этому схема должна уложиться в пределы по потреблению энергии и быстродействию.

2.4. Описание работы устройства

Устройство «калькулятор» состоит и 2-х клавиатур и 3-х семисегментных индикаторов.

Индикаторы показывают числа, которые храниться в 2-х регистрах операндов и в регистре результата.

Ввод чисел в устройство осуществляется с помощью десятичной клавиатуры. При каждом нажатии клавиатура подаёт сигнал на устройство управления и на регистры. Управление устройством выполняется с помощью клавиатуры управления с кнопками выбора операций («+», «ИЛИ»), получения результата («=») и очистки устройства («С»). По умолчанию выбрана операция сложения.

Перед тем, как начать пользоваться устройством, нужно нажать на любую цифру для ввода её в операнд А, иначе какие-либо другие действия с устройством не имеют смысла.

Регистры построены на базе универсальных 4-х разрядных универсальных регистров. Один регистр хранит одно десятичное число. При каждом нажатии на числовую клавиатуру на регистры операндов А и В подаётся синхросигнал, с помощью которого регистры запоминают введённое число. Введённое число поступает на вход последнего регистра (соответствует младшему разряду в десятичной разрядной сетке), и вводе новой цифры младшее число переходит в следующий регистр (в следующий разряд).

Устройство управления определяет, в какой регистр происходит ввод числа. Чтобы начать ввод цифр в операнд В, нужно выбрать операцию, нажатием на соответствующую кнопку. При этом, если есть необходимость выбрать другую операцию, то можно просто нажать на нужную кнопку и продолжить вводить операнд В.

Регистры операндов А и В парраре льно передают свои числа в АЛУ (арифметико-логическое устройство). АЛУ, в свою очередь, парраре льно передаёт результат операции на мультиплексоры. Выбор результата операций определяется тем, какая операция выбрана пользователем.

После того, как ввод операндов в устройство был завершён, можно нажать на клавишу получения результата («=»). Тогда в регистр результата С парраре льно будет записано число, которое соответствует результату операции. После того, как результат был получен, устройство блокирует свою работу и ни на что не реагирует до тех пор, пока не будет нажата кнопка очистки устройства («С»).

С помощью кнопки очистки устройства («С») производиться асинхронная очистка регистров операндов и результата, а также сброс устройства управления в состояние ввода операнда А. При этом устройство запоминает, какая была выбрана операция на момент очистка. После того, как результат операции был получен, устройство будет ожидать очистки.

2.5. Программная среда проектирования устройства

Proteus – это специальное программное обеспечение, разработанная британской компанией Labcenter Electronics Ltd и относящееся к категории систем автоматического проектирования (САПР). ПО позволяет проектировать принципиальные электронные схемы и моделировать их.

Proteus позволяет разрабатывать принципиальные схемы электрических устройств и моделировать их работу. Программа также умеет моделировать работу схем высокой степени интеграции, как микроконтроллеры и процессоры различных архитектур. При этом пользователь может использовать в микроконтроллерах свою заранее разработанную прошивку. В Proteus 8 уже встроенная своя среда разработки для программирования микроконтроллеров – VSM Studio.

Программа также позволяет проектировать печатные платы по разработанной принципиальной схеме устройства и, даже, просмотреть 3D-модель платы.

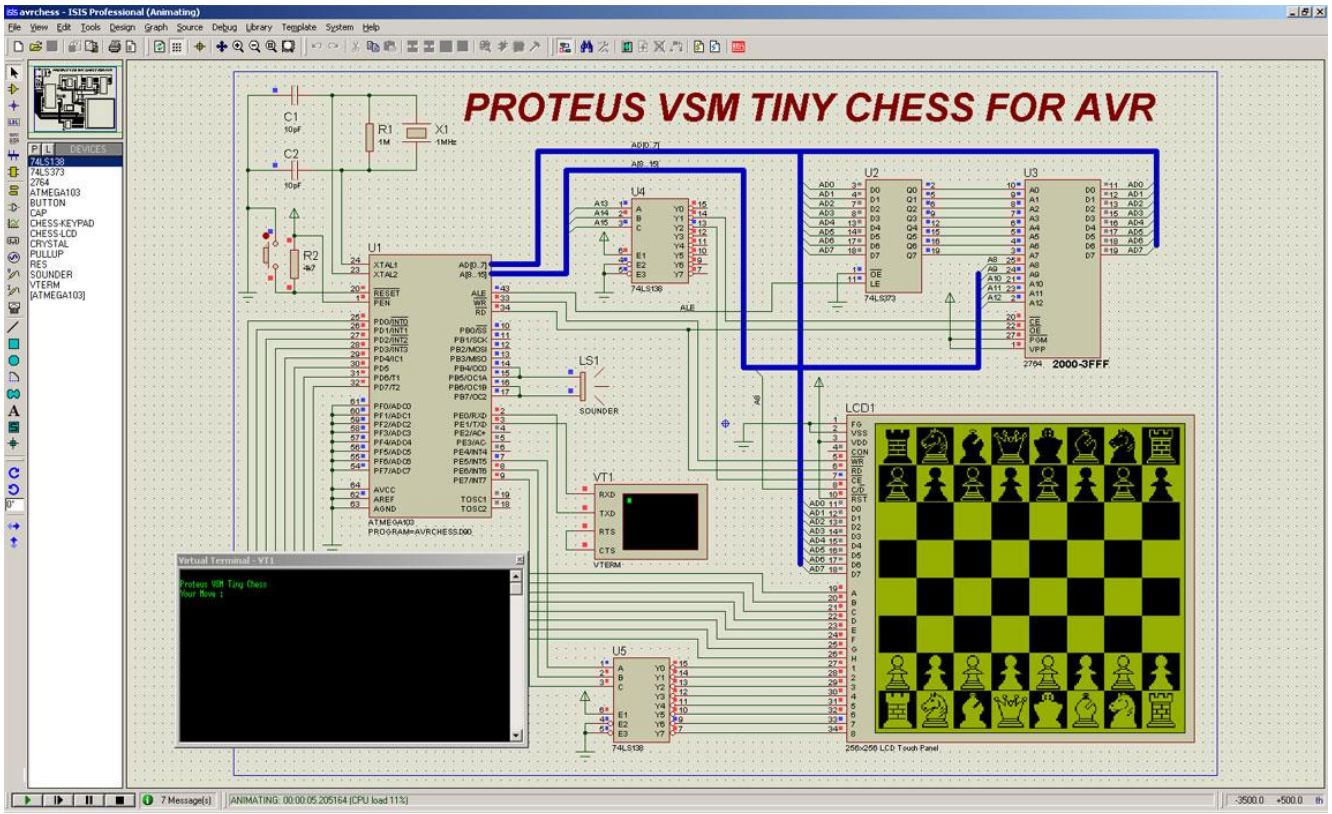

Рисунок 12. Пример интерфейса и принципиальной схемы устройства в ПО Proteus

2.6. Реализация логической операции «ИЛИ»

Реализация логической операции «ИЛИ» в АЛУ устройства является достаточно тривиальной задачей, так как операция логической «ИЛИ» является одной из типовых операций в схемотехнике.

Операция «ИЛИ» в АЛУ реализована с помощью 6-х схем 74HC32. Одна такая схема, в свою очередь, представляет собой 4-ре логических элемента «2ИЛИ». Таким образом, операция «ИЛИ» реализуется с помощью 24-х логических элементов «2ИЛИ». На вход каждого элемента парралельно подаются разряды операндов A и B из соответствующих регистров одного. Выход элемента представляет собой один разряд результата (обозначен как «orN», где N – номер разряда результата). Выход логической операции парралельно подаётся на мультиплексоры выбора результата из АЛУ, а оттуда в регистр результата C.

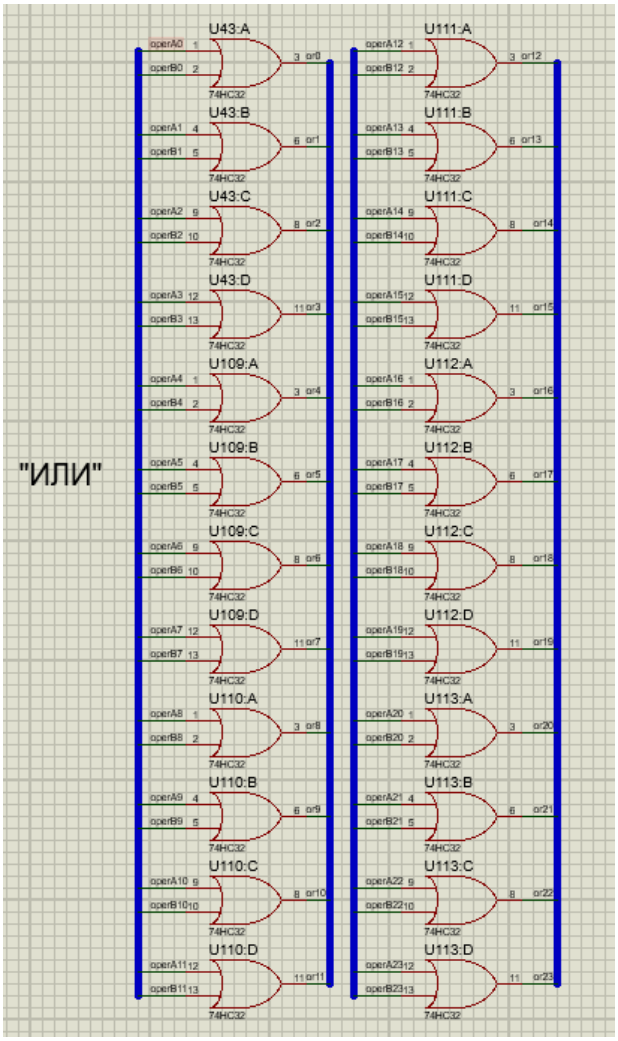

Рисунок 13. Принципиальная схема реализации логической операции «ИЛИ»

2.7. Реализация арифметической операции «СЛОЖЕНИЕ»

Проблема реализации операции арифметического сложения является то, что операнды состоят из нескольких 4-х разрядных регистров, каждый из которых хранит десятичных разряд числа. Но с помощью 4-х разрядов кодируются числа от 0_{16} до F_{16} (или 0_{10} до 15_{10}, или 0000_2 до 1111_2). Это значит, что при реализации арифметической операции следует вводить дополнительную логику для коррекции результата.

Таким образом, операция «СЛОЖЕНИЕ» реализуется с помощью 12-и 4-х разрядных сумматоров 74HC283, 6-и компараторов 74HC85 и нескольких комбинаторных логических элементов.

Один десятичных разряд операндов (4-х разрядное число одного регистра) поступает парраре льно на сумматор №1. Результат сложения передаётся на компаратор, который сравнивает результат с числом 10_{10} (1010_2), и на сумматор №2. Если результат сложения меньше 10_{10}-и, то результат сумматора №1 складывается с 0_{10} (т. е. не изменяется). В противном случае (если результат сложения сумматора №1 больше или равен 10_{10}) сумматор №2 складывает результат сумматора №1 с числом 6_{10} (0110_2). Это приводит к коррекции результата.

Признак переноса в следующий десятичный разряд определяется в обоих сумматорах. Вывода для переноса обоих сумматоров идут на логическое «ИЛИ» и затем переходят в следующий разряд.

Для получения результата для одного десятичного разряда нужно 2-а сумматора, 1-н компаратор и 1-н элемент «2ИЛИ».

Стоит также учесть то, что разрядность операндов в устройстве равна 24. Это значит, что максимальное число операндов может быть 999.999_{10}. При суммировании двух таких операндов в результате будет $1.999.998_{10}$. Из-за этого в составе регистра результата есть ещё 1-н дополнительный триггер, который хранит признак переноса из последнего (23-его) разряда.

Пример: $9_{10}+6_{10}$. 4-х разрядный сумматор выдаст результат F_{16} (15_{10}), тогда как один 4-х разрядный регистр хранит одно десятичное число (от 0_{10} до 9_{10}).

Прибавляем ещё число 6_{10} и получаем в результате число 21_{10}. $21_{10} = 1.0101_2$. Старший разряд числа 21_{10} является признаком переноса в следующий сумматор, остальные числа десятичным результатом сложения двух десятичных чисел ($0101_2 = 5_{10}$). Таким образом, при $9_{10} + 6_{10}$ в разряд этих операндов будет записан 5_{10} и признак переноса перейдёт в следующий сумматор.

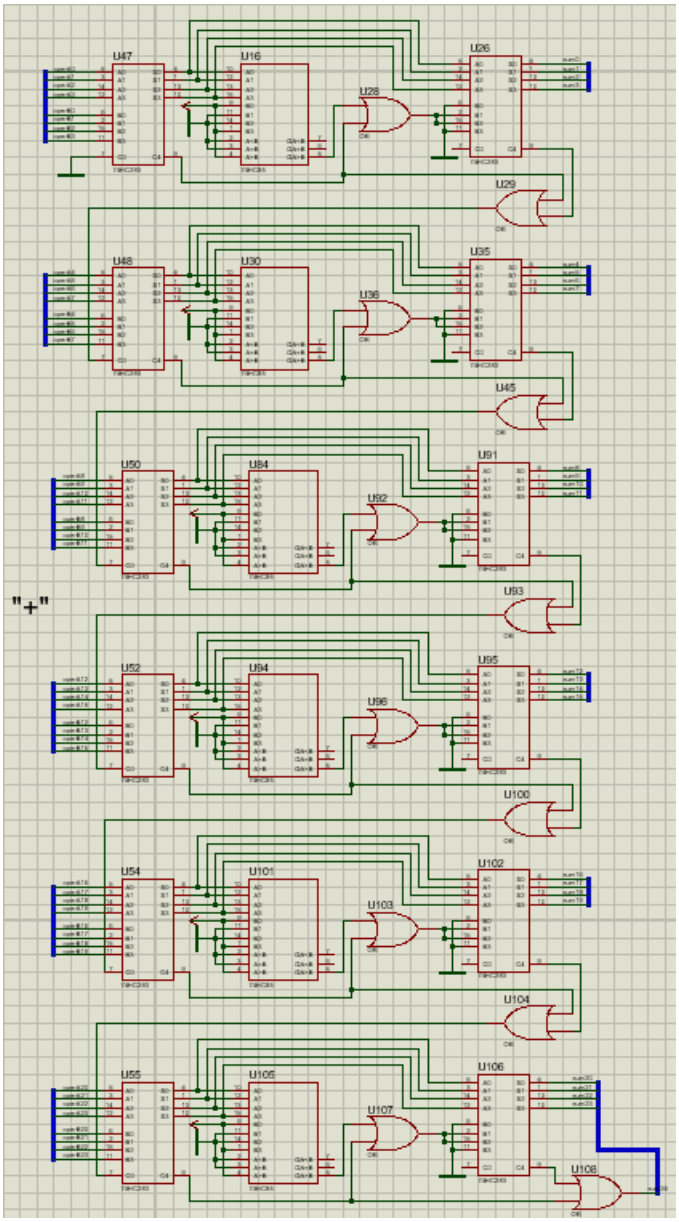

Рисунок 14. Принципиальная схема реализации арифметической операции «СЛОЖЕНИЕ»

Стоит учесть то, что такая реализация увеличивает кол-во компонентов, а значит, что повышается потребляемая мощность и увеличивает время задержки распространения сигналов.

2.8. Реализация управления устройством

2.8.1. Алгоритм управления устройством

Устройство будет работать по следующему алгоритму:

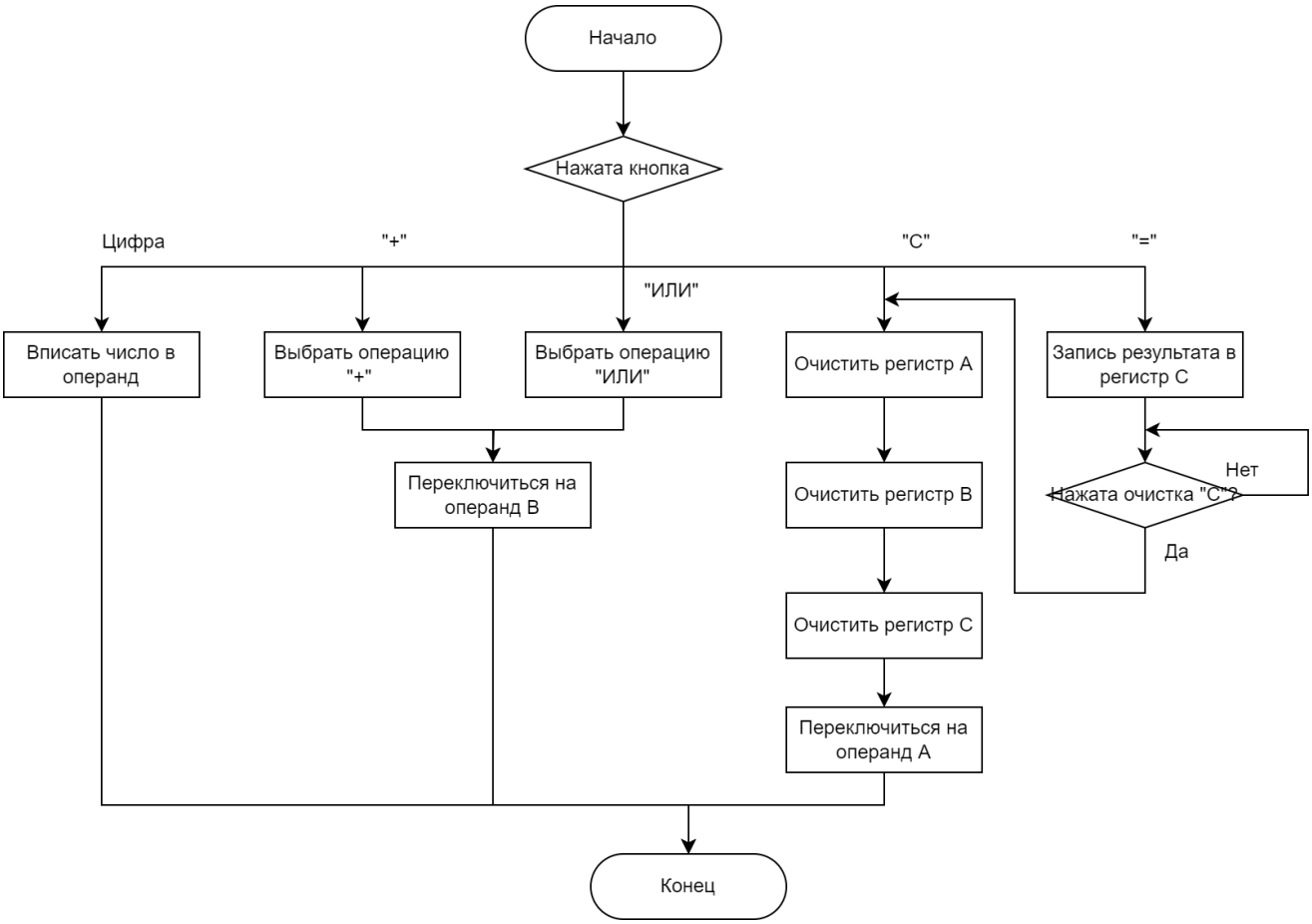

Рисунок 15. Общий алгоритм функционирования устройства

Сначала пользователь должен ввести в устройство число. В противном случае устройство будет игнорировать ввод пользователя.

При нажатии на цифры они будут записываться в последний десятичный разряд выбранного операнда.

При нажатии на кнопки «+» и «ИЛИ» происходит смена текущей операции и переключение на регистр B для ввода второго операнда.

При нажатии на кнопку «C» происходит очистка всех регистров в устройстве и переключение на регистр A.

При нажатии на кнопку «=» в регистр результата C записывается результат текущей операции из АЛУ. При этом устройство блокируется и игнорирует ввод

пользователя до тех пор, пока не будет нажата кнопка «C», т. е. устройство требует обязательной очистки для дальнейшей работы.

Стоит заметить, что устройство запоминает предысторию, т. е. оно запоминает в какой регистр сейчас производиться ввод и какая операция была выбрана пользователем. Это приводит к большому числу состояний автомата. Из-за этого было решено выбрать в качестве управляющего автомата автомат Мили, где выходные сигналы зависят от состояния автомата и его входных сигналов.

2.8.2. Перечень микроопераций и микрокоманд

Определим перечень микроопераций, которые должен выполнять операционный автомат (разрабатываемое устройство без дополнительной схемы управления).

Таблица 3. Перечень микроопераций управляющего автомата

Микрооперация	Комментарий
Y1	Сигнал сохранения цифры
Y2	Выбранная операция: «0» - «+», «1» - «ИЛИ»
Y3	Текущий регистр: «0» - операнд A, «1» - операнд B
Y4	Сигнал сброс
Y5	Сигнал получение результата

2.8.3. Разработка микропрограммы

Определим состояния автомата:

- A0 – недостижимые начальное состояние простоя автомата
- A1 – ввод операнда A для операции «+»
- A2 – ввод операнда A для операции «ИЛИ»
- A3 – ввод операнда B для операции «+»
- A4 – ввод операнда B для операции «ИЛИ»
- A5 – получение результата для операции «+»
- A6 – получение результата для операции «ИЛИ»

Определим входные сигналы:

- X1 – нажата цифра
- X2 – нажата кнопка «+»
- X3 – нажата кнопка «ИЛИ»
- X4 – нажата кнопка «C»
- X5 – нажата кнопка «=»

Запируем состояния автомата:

- A0 – 000
- A1 – 001
- A2 – 010
- A3 – 011
- A4 – 100
- A5 – 101
- A6 – 110

Спроектируем граф-переходов и таблицу переходов автомата.

Таблица 4. Прямая таблица переходов управляющего автомата

Исходное состояние	Состояние перехода	Входной сигнал	Выходной сигнал
A0	A1	X1	Y1
	A0	X2	0
		X3	0
		X4	0
		X5	0
A1	A1	X1	Y1
	A1	X4	Y4
	A3	X2	!Y2 Y3
	A4	X3	Y2 Y3
	A1	X5	0
A2	A2	X1	Y1
	A2	X4	Y4
	A3	X2	!Y2 Y3
	A4	X3	Y2 Y3
	A2	X5	0
A3	A3	X1	Y1
	A4	X3	Y2 Y3
	A1	X4	!Y3 Y4
	A5	X5	Y5
	A3	X2	0

Продолжение таблицы 4. Прямая таблица переходов управляющего автомата

Исходное состояние	Состояние перехода	Входной сигнал	Выходной сигнал
A4	A4	X1	Y1
	A3	X2	!Y2 Y3
	A4	X3	0
	A2	X4	!Y3 Y4
	A6	X5	Y5
A5	A1	X4	!Y3 Y4
	A5	X1	0
		X3	0
		X4	0
		X5	0
A6	A2	X4	!Y3 Y4
	A6	X1	0
		X3	0
		X4	0
		X5	0

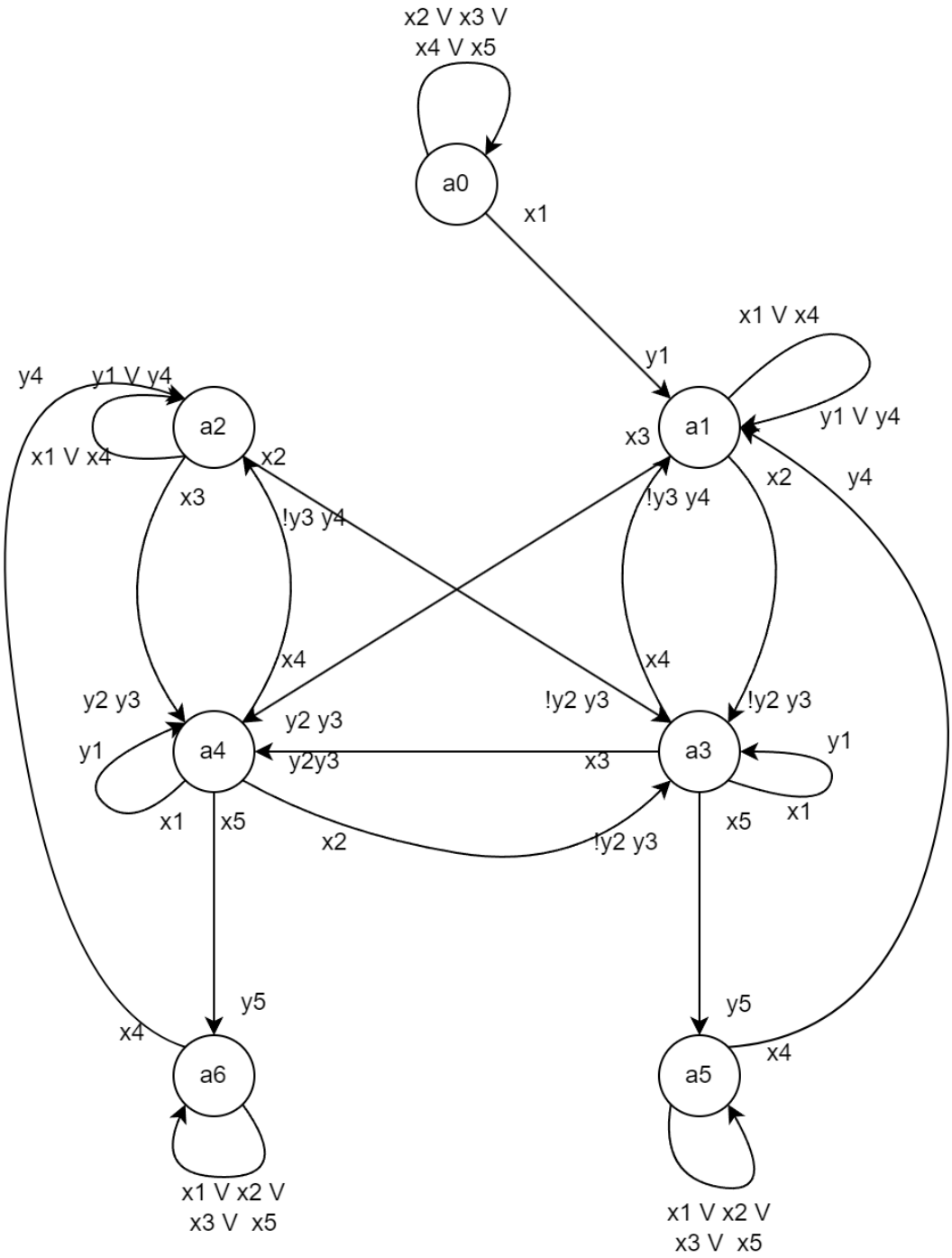

Рисунок 16. Граф-схема автомата управления устройством

На основе таблица переходов и граф-схемы автомата составим таблицу микропрограммы управляющего автомата.

Таблица 5. Таблица микропрограммы ПЗУ

Адрес ПЗУ								Содержимое ПЗУ							
Код исходного состояния			Код входа					Код выхода					Код следующего состояния		
Q2	Q1	Q3	X1	X2	X3	X4	X5	Y1	Y2	Y3	Y4	Y5	Q2	Q1	Q0
0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1
			x	x	x	x	x	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	1	0	0	0	0	0	0	1
			x	1	0	0	0	0	0	1	0	0	0	1	1
			x	x	1	0	0	0	1	1	0	0	1	0	0
			x	x	x	1	0	0	0	0	1	0	0	0	1
			x	x	x	x	1	0	0	0	0	0	0	0	1
0	1	0	1	0	0	0	0	1	1	0	0	0	0	1	0
			x	1	0	0	0	0	0	1	0	0	0	1	1
			x	x	1	0	0	0	1	1	0	0	1	0	0
			x	x	x	1	0	0	1	0	1	0	0	1	0
			x	x	x	x	1	0	0	0	0	0	0	1	0
0	1	1	1	0	0	0	0	1	0	1	0	0	0	1	1
			x	1	0	0	0	0	0	1	0	0	0	1	1
			x	x	1	0	0	0	1	1	0	0	1	0	0
			x	x	x	1	0	0	0	0	1	0	0	0	1
			x	x	x	x	1	0	0	1	0	1	1	0	1
1	0	0	1	0	0	0	0	1	1	1	0	0	1	0	0
			x	1	0	0	0	0	0	1	0	0	0	1	1
			x	x	1	0	0	0	1	1	0	0	1	0	0
			x	x	x	1	0	0	1	0	1	0	0	1	0
			x	x	x	x	1	0	1	1	0	1	1	1	0
1	0	1	x	x	x	1	0	0	0	0	1	0	0	0	1
			x	x	x	x	x	0	0	0	0	0	1	0	1
1	1	0	x	x	x	1	0	0	1	0	1	0	0	1	0
			x	x	x	x	x	0	0	0	0	0	1	1	0

Входные сигналы автомата будут иметь приоритет, т. е. при подаче на вход сигнала X2 сигнал X1 будет игнорироваться. Это позволит упростить построение автомата.

Построение файла микропрограммы для ПЗУ является достаточно сложной задачей. Это можно сделать либо вводя машинный код в файл с помощью специальных программ, таких как HxD, либо написав программу, которая сама сформирует файл согласно определённому пользователем алгоритму.

Стоит заметить, что многие программы для удобства работы с двоичными файлами представляют их содержимое в виде шестнадцатеричных чисел, а не двоичных. Поэтому при ручном вводе микропрограммы необходимо будет переводить двоичное содержимое таблицы в шестнадцатеричный вид.

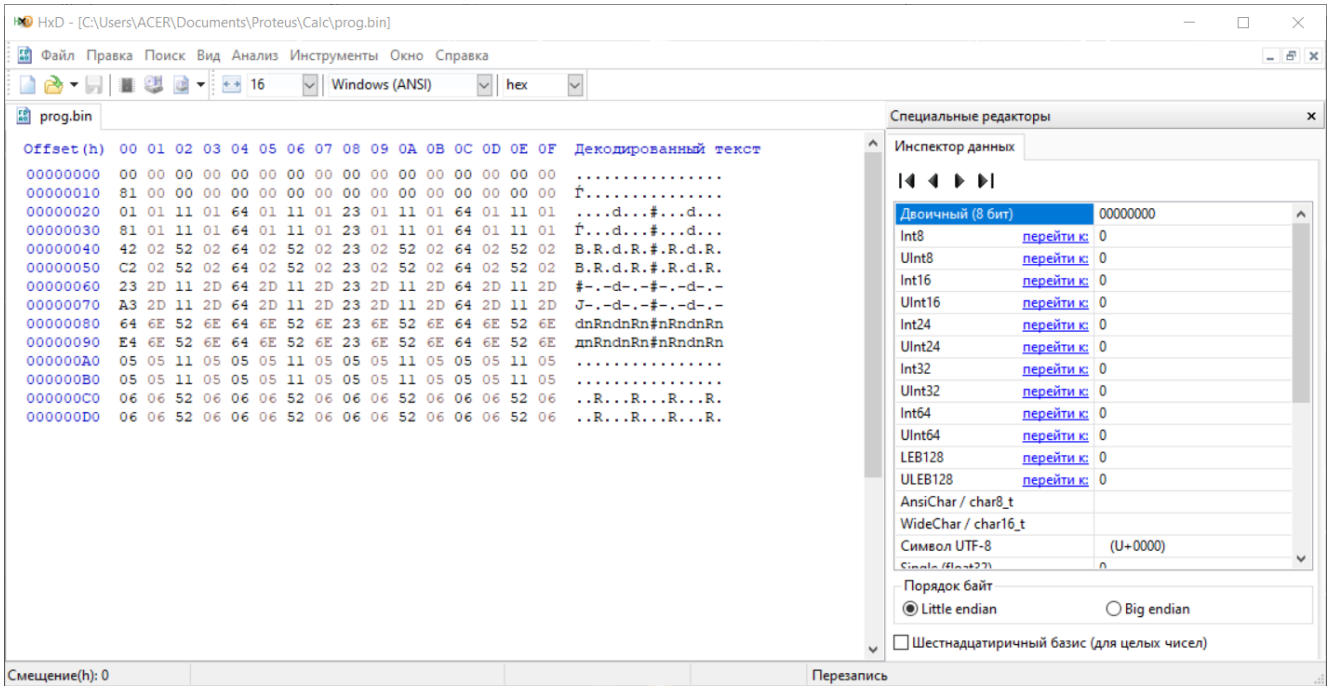

Рисунок 17. Файл микропрограммы устройства, открытый с помощью редактора HxD.

Формирование микропрограммы осуществлялось с помощью небольшой консольной программе на языке программирования C# на базе технологии .Net Framework 4.8. Программу можно просмотреть в приложение.

2.8.4. Выбор запоминающего устройства для микропрограммы

Определим объём ПЗУ, требуемый для записи микропрограммы. Объём ПЗУ будет завесить от исходного состояния и входных сигналов: их в сумме 8. Это значит, что требуемый объём ПЗУ равен $2^8 = 256$ адресов (ячеек памяти) по 8 бит (1 байт) каждый, т. е. 256 * 8 = 2048 бит = 256 байт.

Таким объёмом обладает память 27C256. Это ПЗУ сделано на базе КМОП технологии, благодаря чему потребляемый ток достаточно мал. Быстродействие схемы зависит от производителя ПЗУ: задержка «Адрес-Вывод» варьируется в интервале от 45 до 255 нс (производитель AMD).

2.9 Описание характеристик использованных микросхем

2.9.1 Элемент «2И» 74HC09

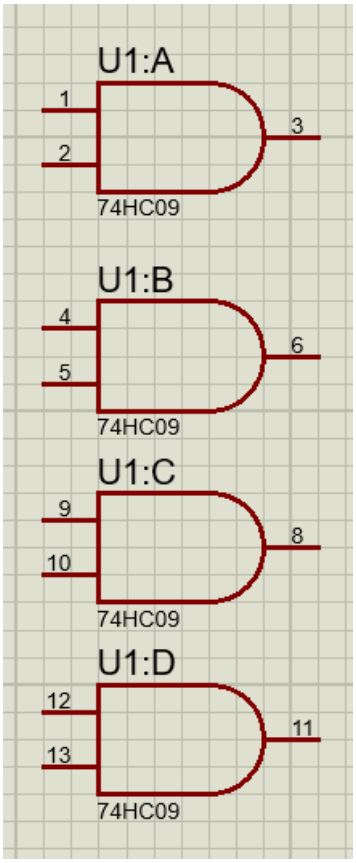

Рисунок 18. Микросхема «2И» 74HC09

Микросхема 74HC09 – это 6-сть логических элементов «2И», сделанных по технологии КМОП.

Характеристики:

- Напряжение питания: U_{cc} = 2-6 В
- Входное/выходное напряжение: U_{output}/U_{input} = 0-U_{cc} В
- Диапазон температур: Ta = от -40 до +85 °C
- Ток питания при U_{cc} = 6 В и Ta = -40 - +85 °C: I_{cc} = 10 мкА
- Задержка «0-1»: t_{pd} = 20 нс при U_{cc} = 6 В и Ta = -40 - +85 °C
- Задержка «1-0»: t_{pd} = 18 нс при U_{cc} = 6 В и Ta = -40 - +85 °C

2.9.2 Элемент «НЕ» 74HC14

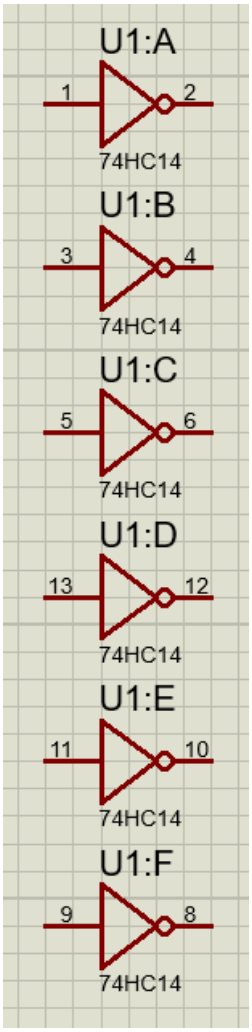

Рисунок 19. Микросхема «НЕ» 74HC14

Микросхема 74HC14 – это 6-сть логических элементов «НЕ» (инверторов) с триггерами Шмитта, сделанных по технологии КМОП. Имеется совместимость с серией ТТЛ по выходным уровням.

Характеристики:

- Напряжение питания: U_{cc} = 2-6 В
- Входное/выходное напряжение :U_{output}/U_{input} = 0-U_{cc} В
- Диапазон температур: Ta = от -40 до +125 °C
- Ток питания при U_{cc} = 6 В и Ta = -40 - +85 °C: I_{cc} = 20 мкА
- Типовая задержка при T = 25°C и U_{cc} = 6 В: t_{pd} = 12 нс
- Максимально задержка при T = -40 - +85°C и U_{cc} = 6 В: t_{pd} = 26 нс

2.9.3 Элемент «4И» 74HC21

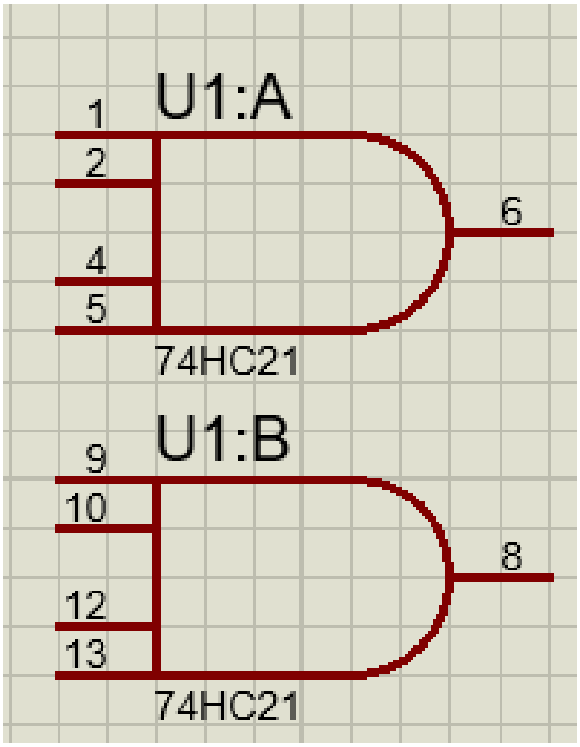

Рисунок 20. Микросхема «4И» 74HC21

Микросхема 74HC21 – это 2-а логических элементов «4И», сделанных по технологии КМОП. Имеется совместимость с серией маломощной серией ТТЛШ (74LS*) по выходным уровням.

Характеристики:

- Напряжение питания: U_{cc} = 2-6 В
- Входное/выходное напряжение: U_{output}/U_{input} = 0-U_{cc} В
- Диапазон температур: Ta = от -40 до +125 °C
- Ток питания при U_{cc} = 6 В и Ta = -40 - +85 °C: I_{cc} = 20 мкА
- Типовая задержка при T = 25°C и U_{cc} = 6 В: t_{pd} = 10 нс
- Максимально задержка при T = от -40 до +85°C и U_{cc} = 6 В: t_{pd} = 24 нс

2.9.4 Элемент «2ИЛИ» 74HC32

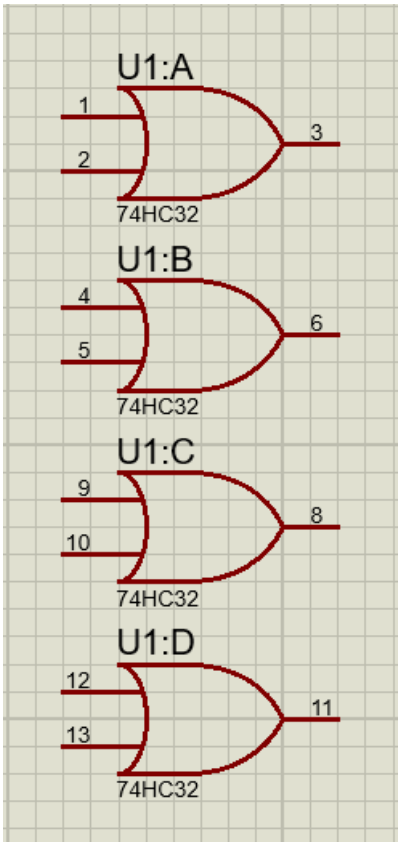

Рисунок 21. Микросхема «2ИЛИ» 74HC32

Микросхема 74HC32 – это 4-ре логических элементов «2ИЛИ», сделанных по технологии КМОП.

Характеристики:

- Напряжение питания: U_{cc} = 2-6 В
- Входное/выходное напряжение: U_{output}/U_{input} = 0-U_{cc} В
- Диапазон температур: Ta = от -40 до +125 °C
- Ток питания при U_{cc} = 6 В и Ta = -40 - +85 °C: I_{cc} = 20 мкА
- Типовая задержка при T = 25°C и U_{cc} = 6 В: t_{pd} = 7 нс
- Максимально задержка при T = от -40 до +85°C и U_{cc} = 6 В: t_{pd} = 20 нс

2.9.5 D-триггер 74HC74

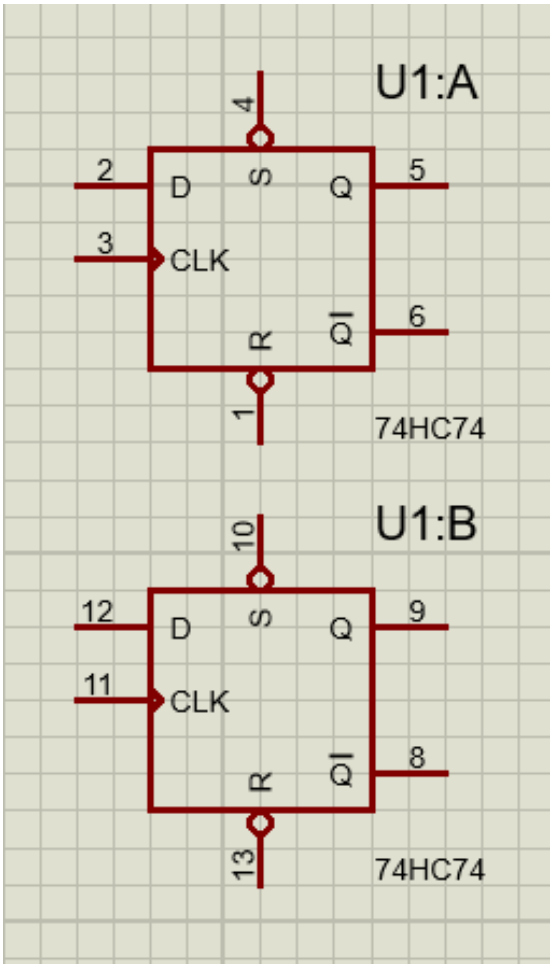

Рисунок 22. Микросхема «D-триггер» 74HC74

Микросхема 74HC74 – это 2-а D-триггера, сделанных по технологии КМОП. Каждый триггер имеет входы данных D, тактирования по положительному фронту C, установки S и сброса R, а также информационные выходы Q и !Q. Входы R и S выполняют функцию асинхронного RS-триггера.

Характеристики:

- Напряжение питания: U_{cc} = 2-6 В
- Входное/выходное напряжение: U_{output}/U_{input} = 0-U_{cc} В
- Диапазон температур: Ta = от -40 до +125 °C
- Ток питания при U_{cc} = 6 В и Ta = -40 - +85 °C: I_{cc} = 40 мкА
- Типовая задержка «Такт-Q», «S-Q», «R-Q» при T = 25°C и U_{cc} = 6 В: t_{pd} = 14 нс

2.9.6 Компаратор 74HC85

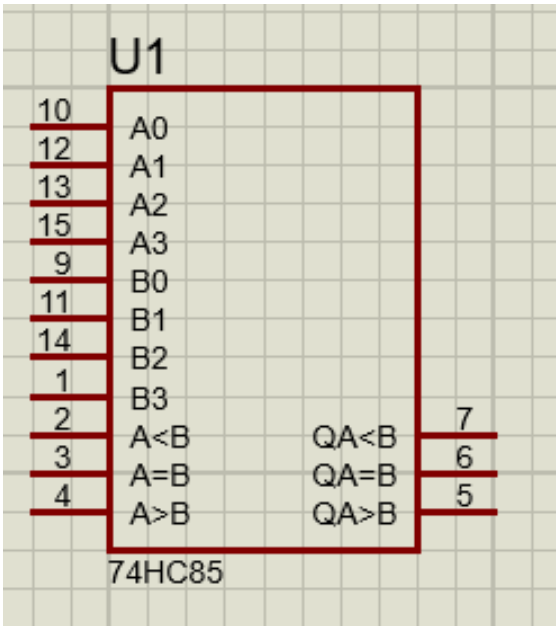

Рисунок 23. Микросхема «4-х разрядный компаратор» 74HC85

Микросхема 74HC85 – это 4-х разрядный компаратор, которые сравнивает числа A и B. Результатами сравнения могут быть сценарии, когда A=B, A>B и A<B. Дополнительные входы регулирует строгость сравнения сигналов A и B. Микросхема сделана по КМОП технологии.

Характеристики:

- Напряжение питания: U_{cc} = 2-6 В
- Входное/выходное напряжение: U_{output}/U_{input} = 0-U_{cc} В
- Диапазон температур: Ta = от -40 до +125 °C
- Ток питания при U_{cc} = 6 В и Ta = -40 - +85 °C: I_{cc} = 80 мкА
- Типовая задержка «A и B-$Q_{A>B}$ и $Q_{A<B}$» при T = 25°C и U_{cc} = 6 В: t_{pd} = 18 нс
- Типовая задержка «A и B-$Q_{A=B}$» при T = 25°C и U_{cc} = 6 В: t_{pd} = 17 нс
- Типовая задержка «A=B и A>B-$Q_{A<B}$» и «A=B и A<B-$Q_{A>B}$» при T = 25°C и U_{cc} = 6 В: t_{pd} = 14 нс
- Типовая задержка «A=B-$Q_{A=B}$» при T = 25°C и U_{cc} = 6 В: t_{pd} = 11 нс

2.9.7 Элемент «2И-НЕ» 74HC132

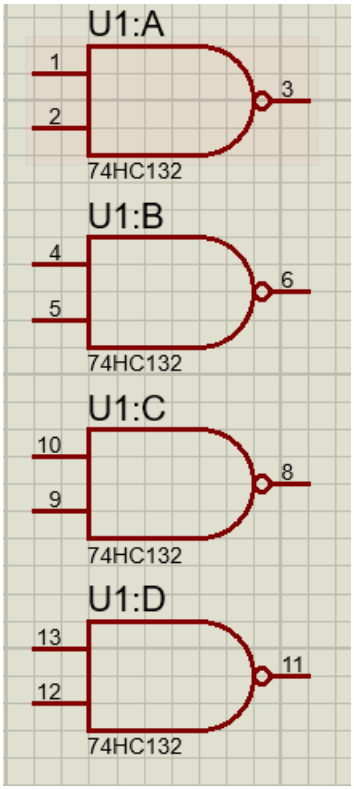

Рисунок 24. Микросхема «2И-НЕ» 74HC132

Микросхема 74HC132 – это 4-ре логических элемента «2И-НЕ» с триггерами Шмитта. Микросхема сделана на базе КМОП технологии.

Характеристики:

- Напряжение питания: U_{cc} = 2-6 В
- Входное/выходное напряжение: U_{output}/U_{input} = 0-U_{cc} В
- Диапазон температур: Ta = от -40 до +125 °C
- Ток питания при U_{cc} = 6 В и Ta = -40 - +85 °C: I_{cc} = 20 мкА
- Типовая задержка при T = 25°C и U_{cc} = 6 В: t_{pd} = 10 нс

2.9.8 Дешифратор 74HC148

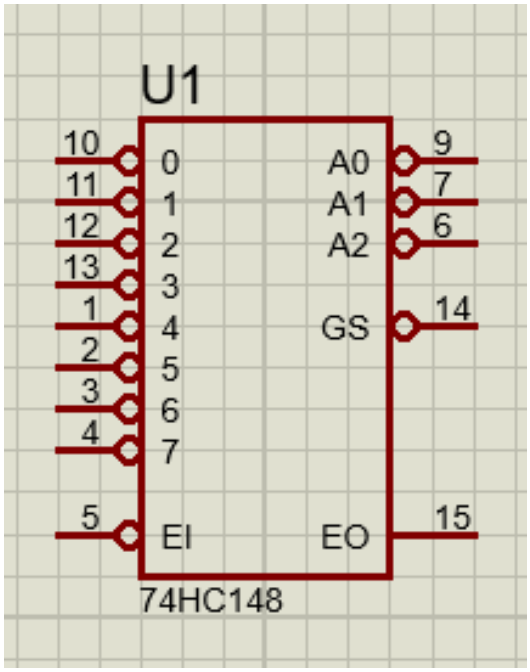

Рисунок 25. Микросхема «Приоритетный дешифратор “8-к-3”» 74HC148

Микросхема 74HC148– это приоритетный дешифратор «8-3» (8-ем входов расшифровываются в 3-х разрядное двоичное число), сделанный по КМОП технологии. Все выводы схемы инвертированы. Имеются дополнительные выходы GS (активен хотя бы один вход) и EO (для связывания нескольких дешифраторов в одну группу).

Характеристики:

- Напряжение питания: U_{cc} = 2-6 В
- Входное/выходное напряжение: U_{output}/U_{input} = 0-U_{cc} В
- Диапазон температур: Ta = от -40 до +125 °C
- Ток питания при U_{cc} = 6 В и Ta = -40 - +85 °C: I_{cc} = 40 мкА
- Типовая задержка «Вход-Выход» и «Вход-GS и EO» при T = 25°C и U_{cc} = 6 В: t_{pd} = 16 нс
- Типовая задержка «EI-EO», «EI-GS» и «EI-Выход» при T = 25°C и U_{cc} = 6 В: t_{pd} = 12 нс

2.9.9 Универсальный регистр 74HC195

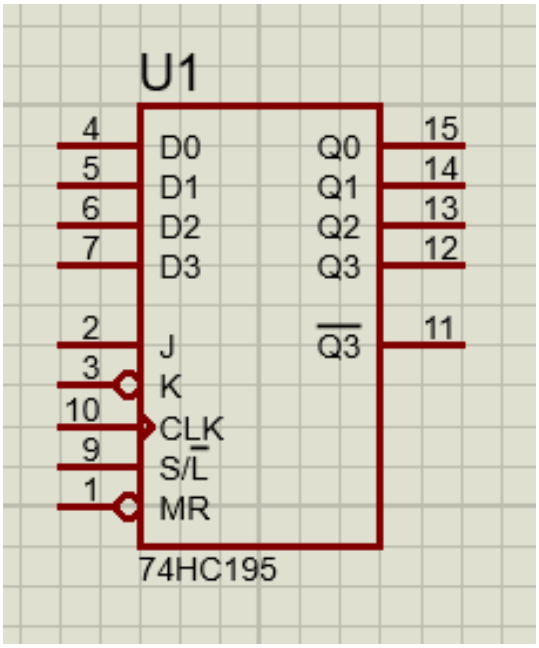

Рисунок 26. Микросхема «Универсальный 4-х разрядный регистр» 74HC195

Микросхема 74HC195– это 4-х разрядный универсальный регистр, сделанный по КМОП технологии. Регистр имеет по 4-ре информационный входа и выхода и один инвертированный информационный выход для Q3, а также вход тактирования по положительному фронту C, асинхронный инвертированный сброс регистра MR и входы для сдвига значений регистра. За сдвиг регистра овечают входы S/!L (парраральный регистр при «0» или регистр сдвига при «1») и входы J и K (работают как JK-триггер для младшего информационного входа Q0).

Характеристики:

- Напряжение питания: U_{cc} = 2-6 В
- Входное/выходное напряжение: U_{output}/U_{input} = 0-U_{cc} В
- Диапазон температур: Ta = от -40 до +125 °C
- Ток питания при U_{cc} = 6 В и Ta = -40 - +85 °C: I_{cc} = 80 мкА
- Максимальная частота при U_{cc}=6 В и Ta=-40 - +85°C: f_{max}=29 МГц
- Типовая задержка «Такт-Выход» при T = 25°C и U_{cc} = 6 В: t_{pd} = 30 нс
- Типовая задержка «Сброс MR-Выход» при T = 25°C и U_{cc} = 6 В: t_{pd} = 26 нс

2.9.10 Мультиплексор 74HC257

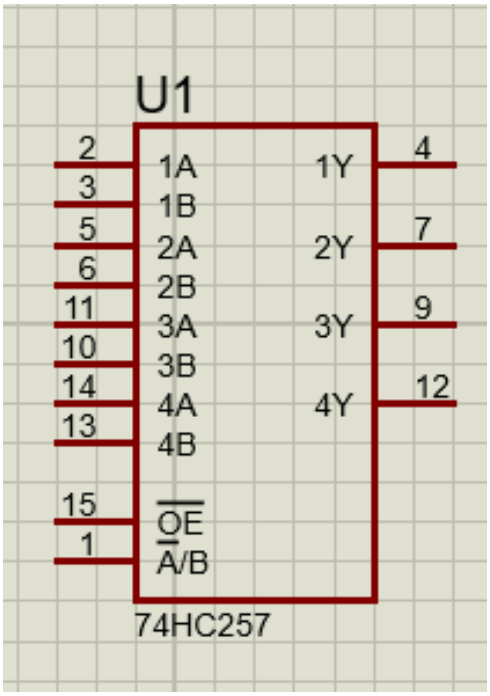

Рисунок 27. Микросхема «Мультиплексор для 2-х 4-х разрядных слов» 74HC257

Микросхема 74HC257 – это мультиплексор «2-1» для 2-х 4-х разрядных чисел с 3-мя состояниями. Имеется инвертированный вход активации вывода EO и вход мультиплексирования входных сигналов !A/B. Микросхема сделана на базе КМОП технологии.

Характеристики:

- Напряжение питания: U_{cc} = 2-6 В
- Входное/выходное напряжение: U_{output}/U_{input} = 0-U_{cc} В
- Диапазон температур: Ta = от -40 до +125 °C
- Ток питания при U_{cc} = 6 В и Ta = -40 - +85 °C: I_{cc} = 80 мкА
- Типовая задержка «Вход-Выход» при T = 25°C и U_{cc} = 6 В: t_{pd} = 10 нс
- Типовая задержка «!A/B-Выход» при T = 25°C и U_{cc} = 6 В: t_{pd} = 14 нс
- Типовая задержка «EO-Выход» при T = 25°C и U_{cc} = 6 В: t_{pd} = 10 или 12 нс (при EO=1 и EO=0 соответственно)

2.9.11 Сумматор 74HC283

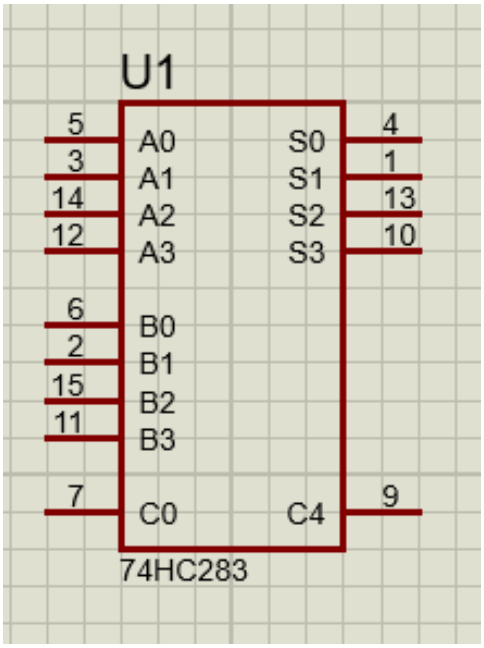

Рисунок 28. Микросхема «4-х разрядный сумматор с быстрым переносом» 74HC283

Микросхема 74HC283 – это 4-х разрядный сумматор с быстрым переносом, выполненный на базе технологии КМОП. Выводы соотносимы с маломощными схемами серии ТТЛШ (74LS*).

Характеристики:

- Напряжение питания: U_{cc} = 2-6 В
- Входное/выходное напряжение: U_{output}/U_{input} = 0-U_{cc} В
- Диапазон температур: Ta = от -40 до +125 °C
- Ток питания при U_{cc} = 6 В и Ta = -40 - +85 °C: I_{cc} = 8 мкА
- Типовая задержка «Cin-S0» при T = 25°C и U_{cc} = 6 В: t_{pd} = 15 нс
- Типовая задержка «Cin-S1» при T = 25°C и U_{cc} = 6 В: t_{pd} = 17 нс
- Типовая задержка «Cin-S2» при T = 25°C и U_{cc} = 6 В: t_{pd} = 18 нс
- Типовая задержка «Cin-S3» при T = 25°C и U_{cc} = 6 В: t_{pd} = 22 нс
- Типовая задержка «A и B-Sn» при T = 25°C и U_{cc} = 6 В: t_{pd} = 20 нс
- Типовая задержка «Cin-Cout» при T = 25°C и U_{cc} = 6 В: t_{pd} = 18 нс
- Типовая задержка «A и B-Cout» при T = 25°C и U_{cc} = 6 В: t_{pd} = 18 нс

2.9.12 Элемент «4ИЛИ-НЕ» 74HC4002

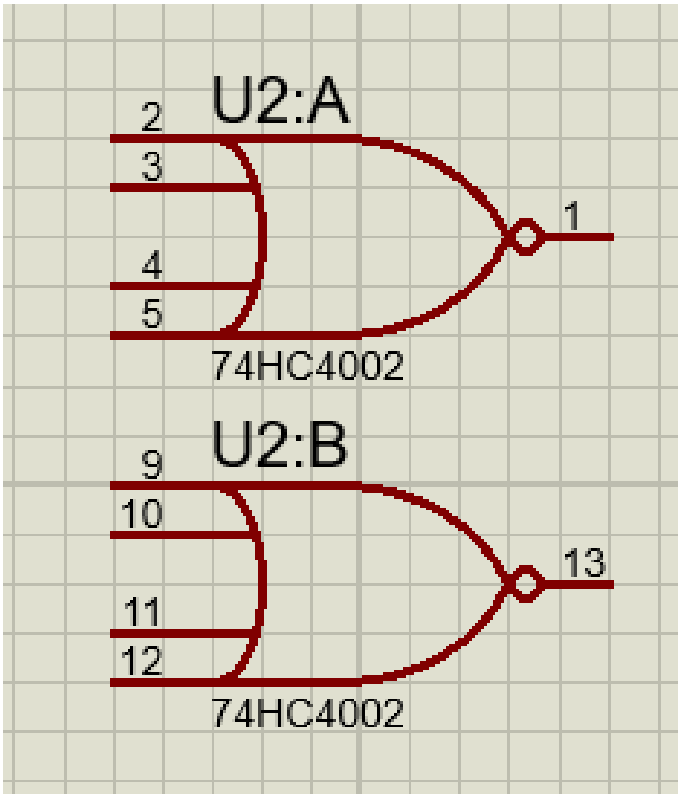

Рисунок 29. Микросхема «4ИЛИ-НЕ» 74HC4002

Микросхема 74HC4002– это 2-а логических элемента «4ИЛИ-НЕ», сделанный на технологии КМОП.

Характеристики:

- Напряжение питания: U_{cc} = 2-6 В
- Входное/выходное напряжение: U_{output}/U_{input} = 0-U_{cc} В
- Диапазон температур: Ta = от -40 до +125 °C
- Ток питания при U_{cc} = 6 В и Ta = -40 - +85 °C: I_{cc} = 20 мкА
- Типовая задержка при T = 25°C и U_{cc} = 6 В: t_{pd} = 9 нс

2.9.13 Элемент «4ИЛИ» 74HC4072

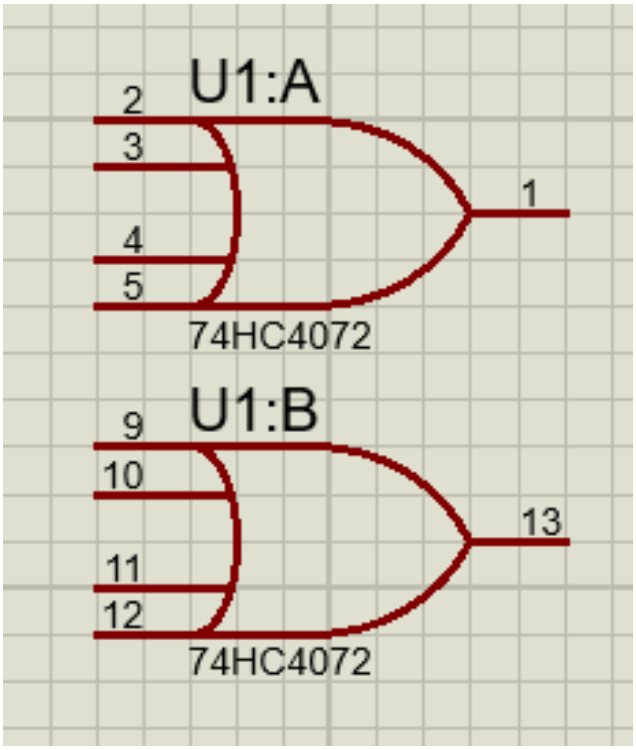

Рисунок 30. Микросхема «4ИЛИ» 74HC4072

Микросхема 74HC4002– это 2-а логических элемента «4ИЛИ» с 3-мя состоянии. Микросхема сделана на базе технологии КМОП.

Характеристики:

- Напряжение питания: U_{cc} = 2-6 В
- Входное/выходное напряжение: U_{output}/U_{input} = 0-U_{cc} В
- Диапазон температур: Ta = от -40 до +125 °C
- Ток питания при U_{cc} = 6 В и Ta = -40 - +85 °C: I_{cc} = 10 мкА
- Типовая задержка при T = 25°C и U_{cc} = 6 В: t_{pd} = 7 нс

2.9.14 ПЗУ 27C256

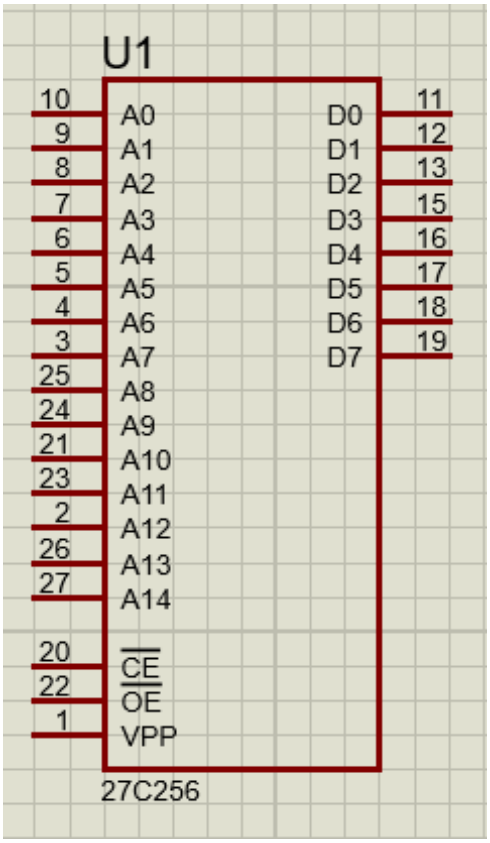

Рисунок 31. Микросхема «32 КБ ППЗУ» 27C256

Микросхема 27C256– это 256 Килобитное (32 КБ) ППЗУ (программируемое постоянное запоминающее устройство), выполненное по КМОП технологии. Имеет дополнительные входы для разрешения выхода памяти (EO), разрешения программирования (CE) и для напряжения программирования (Vpp).

Многие характеристики зависят от производителя памяти и от подтипа микросхемы. Будем рассматривать ПЗУ производителя AMD (Am27C256-XXX).

Характеристики:

- Напряжение питания: U_{cc} = 5 В
- Входное/выходное напряжение: U_{output}/U_{input} = 0-U_{cc} В
- Диапазон температур (для промышленной схемы I): Ta = от -40 до +125 °C
- Ток питания в активном режиме (считывание): I_{cc} = 25 мА
- Ток питания в активном простоя (для КМОП уровней): I_{cc} = 100 мкА
- Типовая задержка «Адрес-Выход»: зависит от типа схемы

- Типовая задержка «CE-Выход»: зависит от типа схемы
- Типовая задержка «EO-Выход»: зависит от типа схемы

Таблица 6. Подтипы микросхемы ПЗУ

Am27C256-								
Подтип	45	55	70	90	120	150	200	255
Max скорость доступа, нс	45	55	70	90	120	150	200	250
CE, нс	45	55	70	90	120	150	200	250
EO, нс	35	35	40	40	50	50	50	50

2.10. Расчёт мощности потребления устройства

Расчёт мощности потребления устройства заключается в суммировании мощностей потребления всех компонентов устройства.

$$P_{\text{потребления схемы}} = \sum P_{\text{микросхемы}}$$

В сведеньях о микросхеме обычно указывается диапазон напряжения питания и ток потребления, с помощью которых можно рассчитать мощность потребления одной микросхемы.

$$P_{\text{потребления}} = I_{\text{потребления}} U_{\text{питания}}$$

Таблица 7. Мощность потребления микросхем в статическом режиме

Микросхема	Назначение схема	$U_{пит}$, [В]	$I_{пит}$, [мкА]	$P_{пит}$, [мкВт]	Кол-во микросхем	$P_{пит}$ всех схем, [мкВт]
74HC148	Дешифратор	6	40	240	2	480
27C256	ПЗУ	6	25.000	150.000	1	150.000
74HC195	Регистр	6	80	480	19	9.120
74HC32	«2ИЛИ»	6	20	120	9	1.080
74HC283	Сумматор	6	80	480	12	5.760
74HC85	Компаратор	6	80	480	6	2.880
74HC257	Мультиплексор	6	80	480	6	2.880
74HC14	«НЕ»	6	20	120	2	240
74HC74	D-триггер	6	40	480	1	480
74HC132	«2И-НЕ»	6	20	120	1	120
74HC21	«4И»	6	20	120	1	120
74HC09	«2И»»	6	10	60	1	60
74HC4072	«4ИЛИ»	6	10	60	1	60
74HC4002	«4ИЛИ-НЕ»	6	20	120	1	120
					$P_{пот}$, [мкВт]	173.400
					$P_{пот}$, [мВт]	173,4

$P_{\text{потр. схемы}} = 173{,}4$ мВт < 300 мВт, что удовлетворяет заданному ограничению.

Зная, что $P_{\text{дим}} \geq P_{\text{ном}}(1 + 0{,}21f)$, можно определить минимальную частоту

$\frac{(\frac{P_{\text{дим}}}{P_{\text{ном}}} - 1)}{0{,}21} \geq f = 3{,}48$ МГц.

2.11. Расчёт быстродействия устройства

Так как расчёт быстродействия всех схемы является достаточно трудоёмким процессом, то быстродействие схемы будет рассмотрено, как время, за которое пользователь устройства получает результат после нажатия на кнопку «=» (равно).

На процесс получения результата С в схеме влияет устройство управления (УУ) и АЛУ, а также некоторые дополнительные схемы и регистры, где храниться результата операции С. Так как логические операции является достаточно простыми с точки зрения схемотехнической реализации (их можно построить с помощью простых комбинационных схем), то влияние логической операции на быстродействие схемы рассмотрено не будет.

Рассчитаем быстродействие схема для аритмической операции сложения.

Таблица 8. Значение времени задержек для микросхем

Микросхема	Обозначение задержки	Комментарий к задержки	Значение времени, [нс]
27C256 (ПЗУ)	t_{acc}	Задержка от адреса к выводу	45
74HC195 (Регистры)	t_{PHL}	C => Qn	30
74HC32 (ИЛИ)	t_{pd}	Xn => Yn	7
74HC283 (Сумматор)	t_{pd}	C0 => C4	18
		An или Bn => Sn	20
		An или Bn => C4	18
		C0 => S3	22
74HC85 (Компаратор)	t_{pd}	An или Bn => A>B или A<B	18
74HC257 (Мультиплексор)	t_{pd}	An или Bn => Yn	10
74HC74 (D-триггер)	t_{pd}	C => Q	14

Сумматоры и регистры являются 4-х разрядными. Так как числа в устройстве являются 24-и разрядными, то нужно минимум 6-сть регистров для результата и 1-н триггер для хранения переноса для операции сложения.

$$t_{\substack{\text{задержка в} \\ 1-\text{ом сумматоре}}} = 20 * 2 + 18 * 2 + 7 + 18 = 101 \text{ нс}$$

$$t_{\substack{\text{задержка в} \\ \text{остальных сумматоре}}} = 22 + 7 + 18 = 47 \text{ нс}$$

$$t_{\substack{\text{задержка в} \\ \text{сумматорах}}} = 101 + 47 * 5 = 336 \text{ нс}$$

$$t_{\text{задержки}} = 336 + 10 + 30 + 45 = 421 \text{ нс}$$

$t_{\text{задержкт}} = 421 \text{ нс} < 400 \text{ нс}$, т. е. ограничение по времени нарушено на 5,25%, что является незначительным.

В целом, в связи с отсутствием буферов, результат операции высчитывается до нажатия на кнопку «=» (равно): регистры результата С ждут получения сигнала на запись значения.

В теории, можно улучшить АЛУ, заменив 4-х разрядные сумматоры серии КМОП сумматорами другой, более быстрой серии, или разбив операцию сложения на алгоритм, где каждый такт будут суммироваться один или несколько десятичных разрядов операндов.

2.12. Структурная схема устройства

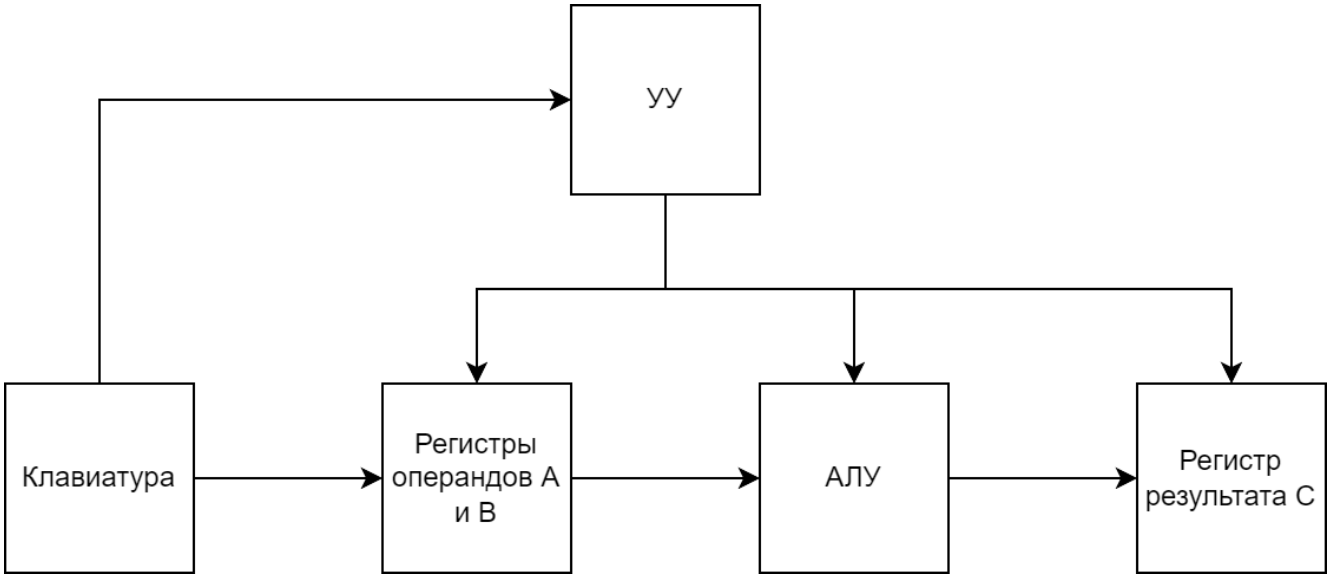

Рисунок 32. Структурная схема устройства «калькулятор»

Устройство состоит из 5-и блоков:

- Блок клавиатуры – блок, с помощью которого пользователь вводит в устройство числа и управляет им
- Блок регистров операндов A и B – блок, где хранятся операнды, с которыми производятся операции
- АЛУ (арифметико-логическое устройство) – блок, где будут происходить операции, а также выбор результата операции (т. е. определение, какая операция выбрана)
- Блок регистра результата C – блок, где будет храниться результат операции
- УУ (устройство управления) – блок, с помощью которого происходит управление устройством (операционным автоматом)

Определим связи между блоками и составим структурную схему устройства.

2.13. Функциональная схема устройства

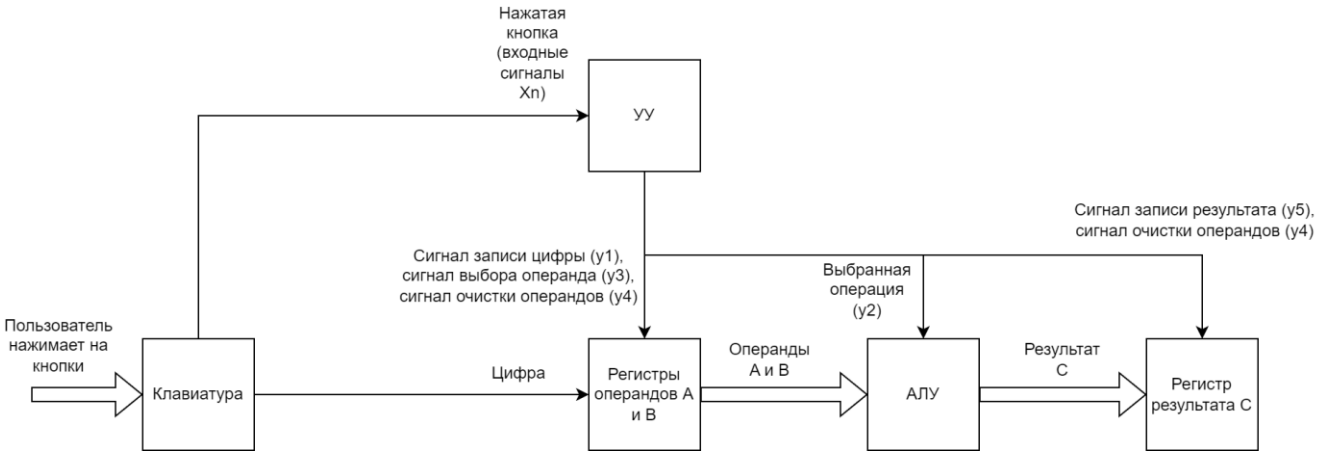

Рисунок 33. Функциональная схема устройства «калькулятор»

Пользователь будет взаимодействовать с устройством с помощью клавиатуры. Клавиатура при нажатии на цифру отправляет на блок операндов десятичную цифру, которую необходимо записать в выбранный регистр. При нажатии на любую из кнопок клавиатуры она посылает сигнал на устройство управления. Этот сигнал обрабатывается, и определяются управляющие сигналы для остальных блоков.

УУ определяет сигналы для остальных блоков устройства. Блок операндов A и B должен знать, в какой регистр сейчас происходит запись цифр клавиатуры. Также блок операндов должен получать сигналы того, что необходимо записать в регистр операнда цифру и что необходимо произвести очистку регистров операндов.

АЛУ должен парраре льно получать значения операндов A и B и знать, какая операция в данный момент времени выбрана, чтобы получить результат той операции, которая требуется.

Регистр результата C должен парраре льно получать результат операции из АЛУ и сигналы на сохранение результата и на очистку регистра.

Рассмотрим некоторые особые блоки устройства.

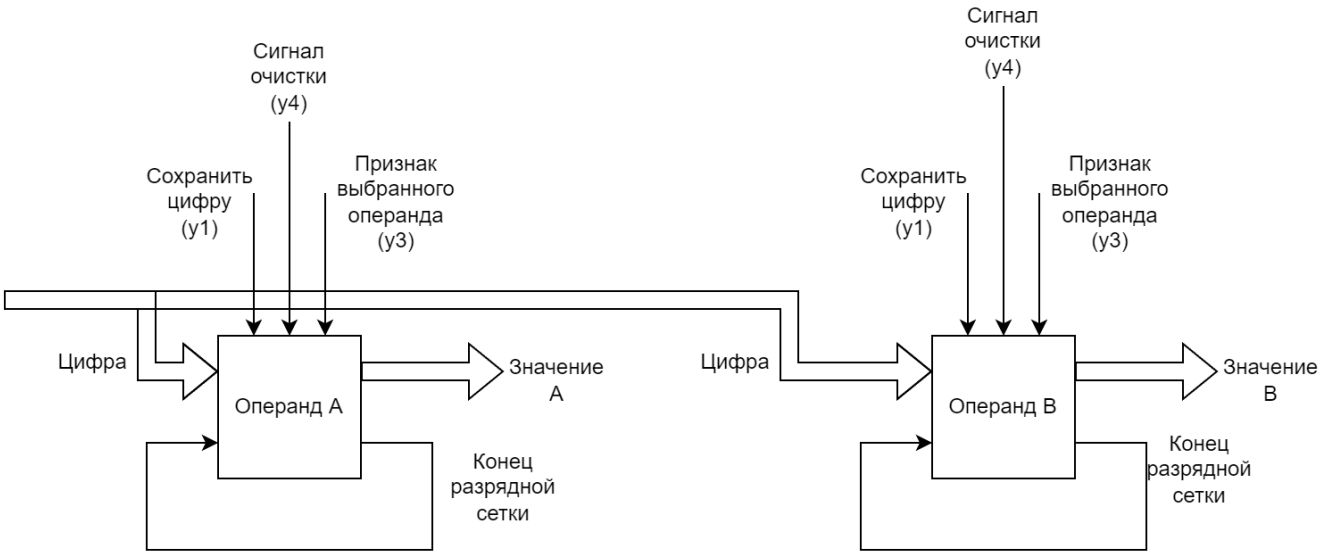

Рисунок 34. Функциональная схема блока регистров операндов

Подробнее рассмотрим блок операндов. Этот блок будет состоять из двух подблоков, каждый из которых представляет собой регистры, где хранятся операнды A и B. На входы регистров поступают цифры с клавиатуры. Эти цифры записываются в конец десятичной разрядной сетки операнда. При этом остальные десятичные разряды сдвигаются влево. Таким образом реализуется удобный и привычный для пользователя ввод чисел в калькулятор. Значения операндов из регистров выводятся парраре льно.

У каждого подблока операндов есть специальный сигнал достижения конца десятичной разрядной стеки. Количество десятичных разрядов ограничено, поэтому этот признак блокирует ввод цифр при достижении конца разрядной сетки.

Каждый подблок операндов принимает от УУ микрокоманды. Таким образом происходит сохранение цифр в регистрах операндов, выборка операнда, в который производиться ввод цифр, и очитка регистров при нажатии на кнопку очистки «C».

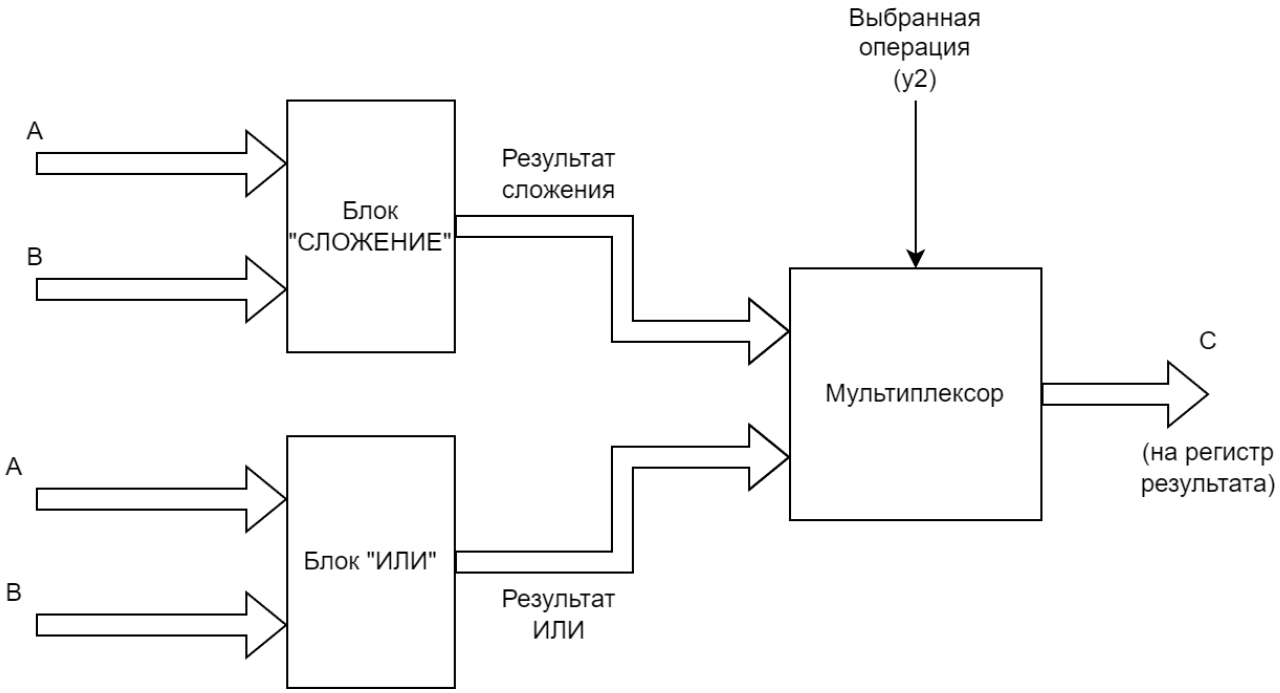

Рисунок 35. Функциональная схема блока АЛУ устройства

Подробнее рассмотрим блок АЛУ. Этот блок состоит из 3-х подблоков. Подблоки для операций «СЛОЖЕНИЕ» и «ИЛИ» выполняют арифметикою и логическую операцию соответственно. Они парралельно принимают значения операндов A и B из блока операндов.

Из подблоков операций парралельно выводиться результат этих операций. Он поступает на подблок мультиплексора (подблок выбора результата), которых осуществляет выбор результата, чтобы сохранить в регистр результата C результат выбранной операции. Блок мультиплексора принимает от УУ сигнал того, какая операция выбрана в данный момент времени. Из подблока мультиплексора результата парралельно поступает на блок регистра результата C.

2.14. Принципиальная схема устройства

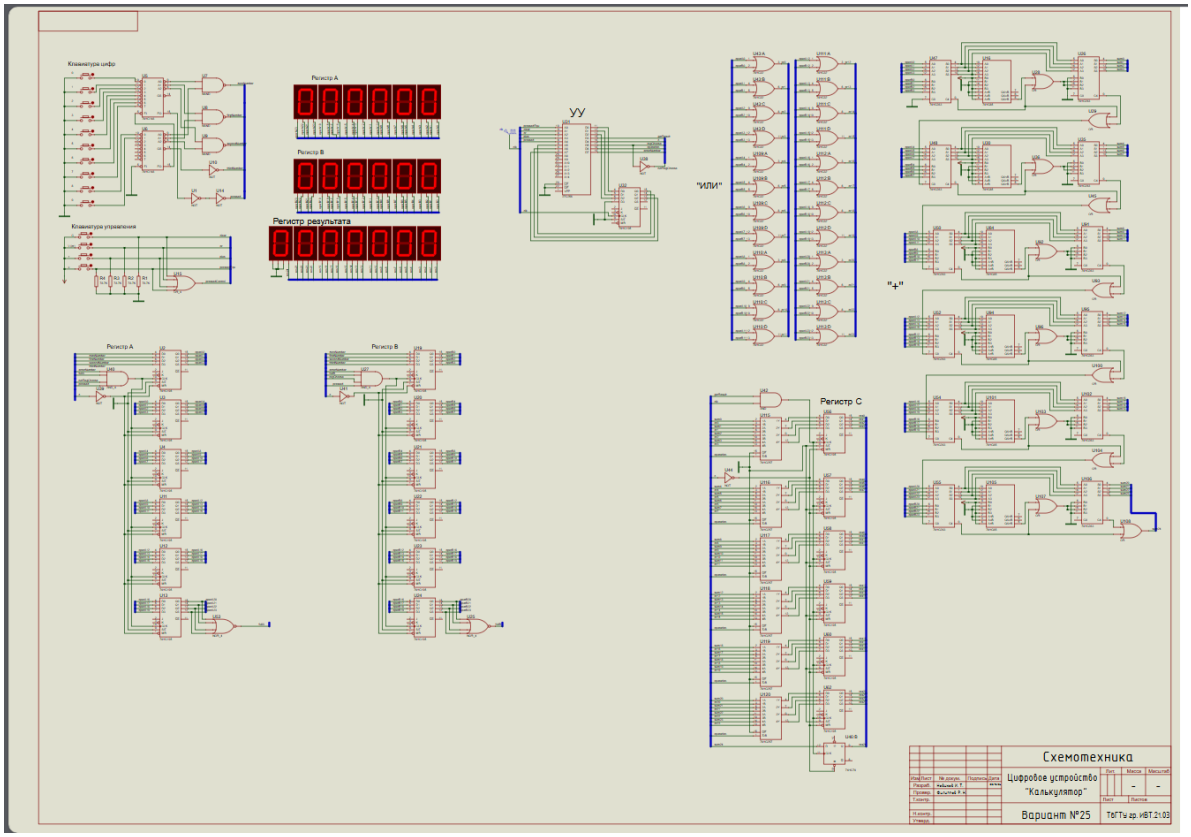

Рисунок 36. Уменьшенное изображение чертежа принципиальной схемы

Так, как принципиальная схема является чрезмерно большой, то его можно будет найти в приложении.

Заключение

В ходе выполнения данного курсового проекта было разработано устройство типа «калькулятор», которое выполняет операции логического «ИЛИ» и арифметического «СЛОЖЕНИЕ», с учётом заданных ограничений.

Устройство выполняет операции корректно. Ограничение устройства по мощности потребления было полностью удовлетворено, но ограничение устройства по быстродействию схемы было незначительно превышено. Это можно исправить заменой микросхем на более быстрые или разбиением операции сложения на алгоритм сложения каждого разряда операндов или полной сменой структуры операции арифметического сложения.

В ходе курсового проекта были выполнены задачи разработки структурной, функциональной и принципиальной схемы устройства. Принципиальная схема была спроектирована и промоделирована в САПР Proteus 8.

Список использованной литературы

1. Постников, А. И. Схемотехника ЭВМ : учебное пособие / А. И. Постников, В. И. Иванов, О. В. Непомнящий. — Красноярск : СФУ, 2018. — 284 с. — ISBN 978-5-7638-3701-8. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/117783 (дата обращения: 7.12.2024). — Режим доступа: для авториз. пользователей.

2. Чеботарев, А. Л. Электроника : учебное пособие / А. Л. Чеботарев. — Кемерово : КемГУ, 2022. — 106 с. — ISBN 978-5-8353-2925-0. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/332342 (дата обращения: 17.12.2024). — Режим доступа: для авториз. пользователей.

3. Новожилов, О. П. Электроника и схемотехника в 2 ч. Часть 1 : учебник для вузов / О. П. Новожилов. — Москва : Издательство Юрайт, 2024. — 382 с. — (Высшее образование). — ISBN 978-5-534-03513-1. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/537682 (дата обращения: 23.11.2024).

4. Новожилов, О. П. Электроника и схемотехника в 2 ч. Часть 2 : учебник для вузов / О. П. Новожилов. — Москва : Издательство Юрайт, 2024. — 421 с. — (Высшее образование). — ISBN 978-5-534-03515-5. — Текст : электронный // Образовательная платформа Юрайт [сайт]. — URL: https://urait.ru/bcode/537683 (дата обращения: 01.12.2024).

5. Гильванов, Р. Г. Схемотехника : учебное пособие / Р. Г. Гильванов. — Санкт-Петербург : ПГУПС, 2021. — 59 с. — ISBN 978-5-7641-1646-4. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/222521 (дата обращения: 10.10.2024). — Режим доступа: для авториз. пользователей.

6. Теория автоматов : учебное пособие / В. В. Лозовский, Е. Н. Штрекер, А. С. Боронников, Л. В. Казанцева. — Москва : РТУ МИРЭА, 2024. — 454 с. — ISBN 978-5-7339-2221-8. — Текст : электронный // Лань : электронно-

библиотечная система. — URL: https://e.lanbook.com/book/421109 (дата обращения: 15.12.2024). — Режим доступа: для авториз. пользователей.

7. Каширская, Е. Н. Теория конечных автоматов : учебное пособие / Е. Н. Каширская, М. М. Клягин, В. А. Серебрянкин. — Москва : РТУ МИРЭА, 2021. — 100 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/226538 (дата обращения: 28.11.2024). — Режим доступа: для авториз. пользователей.

8. Овчаренко, А. Ю. Дискретная математика: теория автоматов : учебно-методическое пособие / А. Ю. Овчаренко ; RU. — Новосибирск : СибГУТИ, 2021. — 24 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/257294 (дата обращения: 28.11.2024). — Режим доступа: для авториз. пользователей.

9. Львова, Е. Ю. Разработка модели цифрового автомата, управляющего работой калькулятора // Science Time — 2014 — с. 123-129

10. Гончаров С., Николаев Д., Никитин В., Писецкий В. Схемотехническая реализация автомата // Компоненты и технологии. 2013. No 2.

11. Сафронов В. Практика математического синтеза микропрограммных управляющих автоматов на основе ПЗУ и ПЛМ [Электронный ресурс] / В. Сафронов, — Компоненты и Технологии, 2014. — Режим доступа: https://cyberleninka.ru/article/n/praktika-matematicheskogo-sinteza-mikroprogrammnyh-upravlyayuschih-avtomatov-na-osnove-pzu-i-plm (дата обращения: 15.11.2024)

12. Ожиганов, А. А. Теория автоматов : учебное пособие / А. А. Ожиганов. — Санкт-Петербург : НИУ ИТМО, 2013. — 84 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/40714 (дата обращения: 15.12.2024). — Режим доступа: для авториз. пользователей.

13. Proteus | Симуляция работы микроконтроллера. – Текст: электронный // dividov.net Электроника для начинающих : [сайт]. URL:

https://diodov.net/proteus-simulyatsiya-raboty-mikrokontrollerov/ (дата обращения: 26.09.2024).

14. Обозначение микросхем Texas Instruments – Текст: электронный // Учебно-методические материалы: [сайт] URL: https://kaf403.rloc.ru/CSMP/ICdesignat.html (дата обращения: 14.10.2024).

15. Сощенко С.В., Штерн М.И. ЭЛЕКТРОНИКА. От азов до создания практических устройств. — СПб.: Издательство Наука и Техника, 2022. — 608 с., илл.

16. Шустов М.А.. ЦИФРОВАЯ СХЕМОТЕХНИКА. Основы построения. — СПб.: Издательство Наука и Техника, 2018. — 320 с., илл.

17. Угрюмов, Е. Цифровая схемотехника / Е. Угрюмов. - СПб.: BHV, 2010. - 816 с. — Текст : электронный // Лань : электронно-библиотечная система. — URL: https://e.lanbook.com/book/257294 (дата обращения: 28.11.2024). — Режим доступа: для авториз. пользователей.

18. Миленина, С.А. Электротехника, электроника и схемотехника: Учебник и практикум для академического бакалавриата / С.А. Миленина, Н.К. Миленин. - Люберцы: Юрайт, 2016. - 399 с., илл.

Приложение

Листинг программы формирования микропрограммы управляющего автомата:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Mikroprogramma
{
    internal class Program
    {
        static void Main(string[] args)
        {
            Work();
            Console.ReadLine();
        }

        static void Work()
        {
            const int a0 = 0b000;
            const int a1 = 0b001;
            const int a2 = 0b010;
            const int a3 = 0b011;
            const int a4 = 0b100;
            const int a5 = 0b101;
            const int a6 = 0b110;
            //const int a7 = 0b111;

            const int y0 = 0b0;
            const int y1 = 0b10000;
            const int y2 = 0b1000;
            const int y3 = 0b100;
            const int y4 = 0b10;
            const int y5 = 0b1;


            List<int> sourse = new List<int>();
            List<int> target = new List<int>();
            List<int> enter = new List<int>();
            List<int> exit = new List<int>();


            #region n1
            //A0
```

```
//Простой
for (int forA = 0; forA < 32; forA++)
{
    sourse.Add(a0);
    enter.Add(forA);

    IEnumerable<char> val = Convert.ToString(forA, 2).Reverse();

    while (val.Count() < 5)
    {
        val = val.Append( '0' );
    }

    if (val.ElementAt<char>(0) == '1')
    {
        target.Add(a0);
        exit.Add(y0);
    }
    else if (val.ElementAt<char>(1) == '1')
    {
        target.Add(a0);
        exit.Add(y0);
    }
    else if (val.ElementAt<char>(2) == '1')
    {
        target.Add(a0);
        exit.Add(y0);
    }
    else if (val.ElementAt<char>(3) == '1')
    {
        target.Add(a0);
        exit.Add(y0);
    }
    else if (val.ElementAt<char>(4) == '1')
    {
        target.Add(a1);
        exit.Add(y1);
    }
    else
    {
        target.Add(a0);
        exit.Add(y0);
    }
}

//A1
//Ввод A для +
for (int forA = 0; forA < 32; forA++)
{
```

```
        sourse.Add(a1);
        enter.Add(forA);

        IEnumerable<char> val = Convert.ToString(forA, 2).Reverse();

        while (val.Count() < 5)
        {
            val = val.Append('0');
        }

        if (val.ElementAt<char>(0) == '1')
        {
            target.Add(a1);
            exit.Add(y0);
        }
        else if (val.ElementAt<char>(1) == '1')
        {
            target.Add(a1);
            exit.Add(y4);
        }
        else if (val.ElementAt<char>(2) == '1')
        {
            target.Add(a4);
            exit.Add(y2+y3);
        }
        else if (val.ElementAt<char>(3) == '1')
        {
            target.Add(a3);
            exit.Add(y3);
        }
        else if (val.ElementAt<char>(4) == '1')
        {
            target.Add(a1);
            exit.Add(y1);
        }
        else
        {
            target.Add(a1);
            exit.Add(y0);
        }
    }

    //A2
    //Ввод A для ИЛИ
    for (int forA = 0; forA < 32; forA++)
    {
        sourse.Add(a2);
        enter.Add(forA);
```

```
    IEnumerable<char> val = Convert.ToString(forA, 2).Reverse();

    while (val.Count() < 5)
    {
        val = val.Append('0');
    }

    if (val.ElementAt<char>(0) == '1')
    {
        target.Add(a2);
        exit.Add(y0);
    }
    else if (val.ElementAt<char>(1) == '1')
    {
        target.Add(a2);
        exit.Add(y2+y4);
    }
    else if (val.ElementAt<char>(2) == '1')
    {
        target.Add(a4);
        exit.Add(y2 + y3);
    }
    else if (val.ElementAt<char>(3) == '1')
    {
        target.Add(a3);
        exit.Add(y3);
    }
    else if (val.ElementAt<char>(4) == '1')
    {
        target.Add(a2);
        exit.Add(y1+y2);
    }
    else
    {
        target.Add(a2);
        exit.Add(y2);
    }
}

//A3
//Ввод B для +
for (int forA = 0; forA < 32; forA++)
{
    sourse.Add(a3);
    enter.Add(forA);

    IEnumerable<char> val = Convert.ToString(forA, 2).Reverse();

    while (val.Count() < 5)
```

```
    {
        val = val.Append('0');
    }

    if (val.ElementAt<char>(0) == '1')
    {
        target.Add(a5);
        exit.Add(y3+y5);
    }
    else if (val.ElementAt<char>(1) == '1')
    {
        target.Add(a1);
        exit.Add(y4);
    }
    else if (val.ElementAt<char>(2) == '1')
    {
        target.Add(a4);
        exit.Add(y2 + y3);
    }
    else if (val.ElementAt<char>(3) == '1')
    {
        target.Add(a3);
        exit.Add(y3);
    }
    else if (val.ElementAt<char>(4) == '1')
    {
        target.Add(a3);
        exit.Add(y1+y3);
    }
    else
    {
        target.Add(a3);
        exit.Add(y3);
    }
}

//A4
//Ввод B для ИЛИ
for (int forA = 0; forA < 32; forA++)
{
    sourse.Add(a4);
    enter.Add(forA);

    IEnumerable<char> val = Convert.ToString(forA, 2).Reverse();

    while (val.Count() < 5)
    {
        val = val.Append('0');
    }
```

```
    if (val.ElementAt<char>(0) == '1')
    {
        target.Add(a6);
        exit.Add(y2+y3+y5);
    }
    else if (val.ElementAt<char>(1) == '1')
    {
        target.Add(a2);
        exit.Add(y2+y4);
    }
    else if (val.ElementAt<char>(2) == '1')
    {
        target.Add(a4);
        exit.Add(y2 + y3);
    }
    else if (val.ElementAt<char>(3) == '1')
    {
        target.Add(a3);
        exit.Add(y3);
    }
    else if (val.ElementAt<char>(4) == '1')
    {
        target.Add(a4);
        exit.Add(y1+y2+y3);
    }
    else
    {
        target.Add(a4);
        exit.Add(y3+y2);
    }
}

//A5
//Получить res для +
for (int forA = 0; forA < 32; forA++)
{
    sourse.Add(a5);
    enter.Add(forA);

    IEnumerable<char> val = Convert.ToString(forA, 2).Reverse();

    while (val.Count() < 5)
    {
        val = val.Append('0');
    }

    if (val.ElementAt<char>(0) == '1')
    {
```

```
            target.Add(a5);
            exit.Add(y0);
        }
        else if (val.ElementAt<char>(1) == '1')
        {
            target.Add(a1);
            exit.Add(y4);
        }
        else if (val.ElementAt<char>(2) == '1')
        {
            target.Add(a5);
            exit.Add(y0);
        }
        else if (val.ElementAt<char>(3) == '1')
        {
            target.Add(a5);
            exit.Add(y0);
        }
        else if (val.ElementAt<char>(4) == '1')
        {
            target.Add(a5);
            exit.Add(y0);
        }
        else
        {
            target.Add(a5);
            exit.Add(y0);
        }
    }

    //A6
    //ПОлучить res для ИЛИ
    for (int forA = 0; forA < 32; forA++)
    {
        sourse.Add(a6);
        enter.Add(forA);

        IEnumerable<char> val = Convert.ToString(forA, 2).Reverse();

        while (val.Count() < 5)
        {
            val = val.Append('0');
        }

        if (val.ElementAt<char>(0) == '1')
        {
            target.Add(a6);
            exit.Add(y0);
        }
```

```
                else if (val.ElementAt<char>(1) == '1')
                {
                    target.Add(a2);
                    exit.Add(y2+y4);
                }
                else if (val.ElementAt<char>(2) == '1')
                {
                    target.Add(a6);
                    exit.Add(y0);
                }
                else if (val.ElementAt<char>(3) == '1')
                {
                    target.Add(a6);
                    exit.Add(y0);
                }
                else if (val.ElementAt<char>(4) == '1')
                {
                    target.Add(a6);
                    exit.Add(y0);
                }
                else
                {
                    target.Add(a6);
                    exit.Add(y0);
                }
            }
            #endregion

            List<int> addresses = new List<int>(sourse.Count);
            List<int> output = new List<int>(sourse.Count);

            StringBuilder sb = new StringBuilder();
            StringBuilder sb2 = new StringBuilder();

            sb.AppendLine("A;X;Y;A;");
            sb2.AppendLine("Adress;Value;");
            for (int i = 0; i < sourse.Count; i++)
            {
                sb.AppendLine($"{Convert.ToString(sourse[i],
2)};{Convert.ToString(enter[i], 2)};{Convert.ToString(exit[i],
2)};{Convert.ToString(target[i], 2)};");

                addresses.Add( (sourse[i] << 5) + enter[i] );
                output.Add( (exit[i] << 3) + target[i] );

                sb2.AppendLine($"{Convert.ToString(addresses[i],
16)};{Convert.ToString(output[i], 16)};");
            }
```

```
            using (StreamWriter tw = new StreamWriter("table.txt"))
            {
                tw.Write(sb.ToString());
            }

            using (StreamWriter tw = new StreamWriter("adresses.txt"))
            {
                tw.Write(sb2.ToString());
            }

            using (StreamWriter sw = new StreamWriter("prog"))
            {
                BinaryWriter bw = new BinaryWriter(sw.BaseStream);

                foreach (int item in output)
                {
                    bw.Write((byte)item);
                }

            }
        }
    }
}
```