МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

В.И. Бойков, Г.И. Болтунов, С.В. Быстров,

В.В. Григорьев, Ю.В. Литвинов

Цифровая техника систем управления

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ В УНИВЕРСИТЕТЕ ИТМО по направлениям подготовки 15.04.06, 27.04.03 , 27.04.04 в качестве учебного пособия для реализации основных образовательных программ высшего образования магистратуры

Санкт-Петербург

2018

Бойков В.И., Болтунов Г.И., Быстров С.В., Григорьев В.В., Литвинов Ю.В. Цифровая техника систем управления: Учебное пособие. – СПБ.: Университет ИТМО, 2018. - 139 с., илл.

Рецензенты:

Шаветов С.В., канд. техн. наук, доцент кафедры интеллектуальных технологий промышленной робототехники Университета ИТМО

В учебном пособии дано общее представление о способах реализации современных цифровых вычислительных средств, их развитии и особенностях применения в задачах управления техническими объектами. Рассмотрены особенности аппаратной реализации регистрового и табличного способов вычислений. На примере задачи управления шаговым двигателем рассмотрены особенности разработки программ и технической реализации микропрограммных автоматов основных типов. В учебных целях при синтезе автоматов использованы только «ручные» методы. Также разобраны особенности применения сигнальных процессоров и микроконтроллеров в задачах обработки информации и управления.

Пособие ориентировано на студентов старших курсов, обучающихся по направлениям подготовки: «Системный анализ и управление», «Управление в технических системах», «Мехатроника и робототехника». При изложении материала авторы исходили из того, что читатель знаком с основами электроники и цифровой техники в рамках дисциплин «Электроника», «Электронные устройства систем управления» и «Микроконтроллерная техника систем управления».

Университет ИТМО–ведущий вуз России в области информационных и фотонных технологий, один из немногих российских вузов, получивших в 2009 г статус национального исследовательского университета. С 2013 г Университет ИТМО – участник программы повышения конкурентоспособности российских университетов среди ведущих мировых научно-образовательных центров, известной как программа «5 в 100». Цель Университета ИТМО–становление исследовательского университета мирового уровня, предпринимательского по типу, ориентированного на интернационализацию всех направлений деятельности.

СОДЕРЖАНИЕ

Введение …… 5
1. Аппаратная реализация устройств цифрового управления
1.1 Классификация цифровых вычислительных устройств 6
1.1.1 ЭНИАК – прорывная технология создания Вычислительной техники XX века …….. 6
1.1.2 Классификация систем по видам сигналов …….. 10
1.1.3 Классификация цифровых вычислительных устройств …….. 12
1.2 Дифференциальные анализаторы и аналоговая вычислительная техника …….. 17
1.3 Метод цифровых дифференциальных анализаторов …….. 19
1.4 Синтез схемы генератора задающего воздействия …….. 22
1.5 Методы цифрового интегрирования. Метод Эйлера …….. 24
1.6 Методы цифрового интегрирования. Метод трапеций 28
1.7 Форматы чисел. Погрешность цифрового представления аналоговой величины …….. 30
1.7.1 Формат целых чисел …….. 31
1.7.2 Формат чисел с фиксированной запятой …….. 33
1.7.3 Формат чисел с плавающей запятой …….. 34
1.8 Простейший регистровый процессор. Особенности математической модели …….. 37
1.9 Выбор разрядности шин передачи данных регистрового процессора …….. 41
1.10 Масштабирование переменных регистрового процессора …... 44
1.11 Аппаратная реализация устройств обработки информации … 47
1.11.1 Цифровой фильтр для сглаживания ступенчатых воздействий …….. 47
1.11.2 Цифровой фильтр для усреднения данных от нескольких датчиков …….. 51
1.11.3 Матричная организация процессора …….. 53
1.12 Применение табличных вычислений в устройствах управления …….. 54
1.13 Табличная реализация вычисления значения нелинейной функции …….. 57
1.14 Кусочно-линейная аппроксимация нелинейной функции. Способ уменьшения требуемого объема ПЗУ …… 60
1.15 Виды постоянных запоминающих устройств …….. 62
1.16 Реконфигурируемые пользователем вычислительные устройства …….. 67

2. Программная и микропрограммная реализации
устройств цифрового управления ……... 72
2.1 Микропрограммный автомат. Базовые термины ……... 72
2.2 Базовая функциональная схема
микропрограммного автомата ……... 74
2.3 Микропрограммный автомат с принудительной
адресацией и двумя адресными полями ……... 76
2.4 Микропрограммный автомат с принудительной
адресацией и одним адресным полем ……... 81
2.5 Микропрограммный автомат с естественной
адресацией ……... 85
2.6 Микропрограммный автомат с естественной
адресацией и разделенной микрокомандой ……... 90
2.7 Микропрограммный автомат с усеченной
естественной адресацией ……... 93
2.8 Микропрограммное управление ходом вычислений.
Архитектура цифрового сигнального процессора ……... 98
2.9 Управления цифровой частью процессора. Реализация
операции умножения на постоянный коэффициент ……... 102
2.10 Управления аналоговой частью процессора. Реализация
аналого-цифрового преобразования ……... 105
2.11 Микропрограммная реализация регистрового способа
вычислений. Распараллеливание вычислений ……... 106
2.12 Программное управление ходом вычислений.
Архитектура микроконтроллера ……... 110
2.13 Программная реализация табличного способа вычислений .. 113
2.13.1 Вычисление значения нелинейной функции …….. 114
2.13.2 Реализация генератора функции времени ……... 117
2.14 Программная реализация регистрового способа
вычислений ……... 121

ЗАКЛЮЧЕНИЕ ……... 125
Список использованных источников ……... 128
ПРИЛОЖЕНИЕ А Терминология ……... 130

ВВЕДЕНИЕ

Настоящее учебное пособие ставит целью дать общее представление о структуре современных электронных вычислительных средств, их развитии и особенностях применения для управления техническими объектами в реальном времени. Выделены два основных способа вычислений (регистровый и табличный) и три способа реализации вычислений (аппаратный, микропрограммный и программный).

В первой главе учебного пособия рассмотрены особенности регистрового и табличного способа вычислений. Рассмотрение ориентировано на аппаратную реализацию устройств, которая, несмотря на ее трудоемкость, находит все большее распространение в современной технике. Связано это в первую очередь с появлением на рынке электроники большого числа различных и весьма мощных микросхем с архитектурой, программируемой пользователем. Большие объемы перепрограммируемой постоянной памяти, технологии ПЛИС и FPGA, сделали реальностью создание пользователем своего собственного вычислительного устройства на базе серийно выпускаемой микросхемы. Появился и специальный термин для обозначения новой технологии – «система на кристалле» или SoC – System on Chip. В пособии рассмотрены базовые принципы аппаратной реализации вычислений, что несомненно поможет заинтересованным лицам овладеть уровнем использования технологии SoC [28].

Во второй главе рассмотрены особенности использования микропрограммного и программного способов реализации вычислений в задачах управления. На примере задачи управления шаговым двигателем рассмотрены особенности разработки программ и технической реализации микропрограммных автоматов базовых типов. Разобраны основные особенности использования сигнальных процессоров и микроконтроллеров в задачах управления.

Пособие ориентировано на студентов старших курсов, обучающихся по программе магистерской подготовки по направлениям «Системный анализ и управление», «Управление в технических системах», «Мехатроника и робототехника». При изложении материала авторы исходили из того, что читатель знаком с основами электроники и цифровой техники в рамках дисциплин «Электроника», «Электронные устройства систем управления» и «Микроконтроллерная техника систем управления», изучаемых при подготовке бакалавров.

Авторы надеются, что учебное пособие будет полезно студентам других специальностей, интересующихся вопросами технической реализации своих идей, а также специалистам, разрабатывающим прикладные управляющие системы и комплексы.

1. Аппаратная реализация устройств цифрового управления

1.1 Классификация цифровых вычислительных устройств

1.1.1 ЭНИАК - прорывная технология создания вычислительной техники XX века

Цифровая вычислительная техника сегодня есть в каждом доме. Эта техника не только позволяет решать математические задачи, но и существенно улучшает качество нашей жизни, обеспечивая ненавязчивый сервис работы бытовой техники.

Типовая система автоматического управления состоит из объекта управления, исполнительных устройств, измерительных устройств и устройства управления, реализующего определенный закон управления объектом. Устройство управления (регулятор) - это специфическое вычислительное устройство, поэтому цифровая вычислительная техника широко используется для реализации современных регуляторов.

На рисунке 1.1.1 показана схема типовой цифровой системы управления. На ход управляемого процесса в объекте управления (ОУ) влияет исполнительное устройство (ИУ). Параметры управляемого процесса измеряются датчиками (Д). Управляет процессом цифровое вычислительное устройство (ЦВУ), которое обрабатывает информацию от датчиков и формирует сигнал управления. ЦВУ соединяется с датчиками и исполнительным устройством посредством специальных устройств

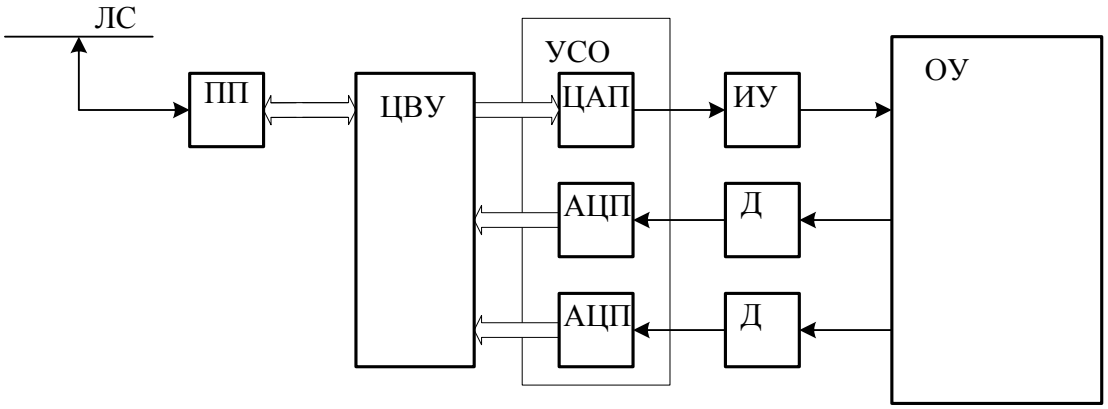

Рисунок 1.1.1 – Схема типовой цифровой системы управления

сопряжения с объектом (УСО). Для преобразования сигналов управления УСО обычно содержат цифро-аналоговые либо широтно-импульсные преобразователи (ЦАП). Для преобразования сигналов датчиков УСО обычно содержат аналого-цифровые преобразователи (АЦП). Задание желаемого значения регулируемого процесса и контроль результата

выполняется устройством более высокого уровня управления, связь с которым осуществляется по цифровой линии связи (ЛС). Линия связи сопрягается с ЦВУ посредством специального модуля согласования – приемо-передатчика (ПП) цифровой информации.

Современная цифровая техника, цифровые вычислительные устройства, позволяет реализовать любые законы управления объектами, обеспечивая заданное качество управляемых процессов и высокий уровень сервисных технологических операций. Такого уровня цифровой техники удалось достигнуть благодаря применению прорывной технологии, использованной в середине двадцатого века. Эта прорывная технология состояла в том, что вместо традиционных электромеханических счетно-решающих устройств начали применять чисто электронные устройства.

Все началось в июне 1943 года, когда артиллерийское управление США заключило договор с Пенсильванским университетом на постройку "Электронной машины для расчета баллистических таблиц". Руководителем работ был назначен Моучли, а главным инженером - Эккерт. В группу разработчиков входили 10 инженеров. Двести техников и большое число рабочих в течение двух с половиной лет трудились над созданием "Электронно цифрового интегратора", сокращенно ENIAC [14].

Основу схемы ENIAC составляли так называемые ячейки "и", действовавшие как переключатели, ячейки "или", предназначавшиеся для объединения на одном выходе импульсов, идущих от разных источников, и триггеры, построенные на электронных лампах. Десять триггеров соединялись в кольцо, образуя десятичный счетчик, который исполнял роль счетного колеса механической машины. Десять таких электронных колец плюс два триггера для представления знака числа составляли запоминающий регистр. Всего ENIAC содержал двадцать запоминающих регистров. Каждый регистр был снабжен схемой передачи десятков и мог быть использован для операций суммирования и вычитания. Другие арифметические операции выполнялись в специализированных электронных блоках. Помимо памяти, на триггерных ячейках в машине имелся блок механических переключателей, на котором вручную могло быть установлено до 300 чисел. Числа передавались из одной части машины в другую посредством проводников, по одному для каждого десятичного разряда и знака числа. Значение передаваемой цифры равнялось числу импульсов, прошедших по данному проводнику. Работой отдельных блоков машины управлял задающий генератор, который определял последовательность тактовых и синхронизирующих импульсов, эти импульсы "включали" и "выключали" соответствующие электронные блоки машины.

Ввод чисел в машину производился с помощью перфокарт, а программирование осуществлялось, как в счетно-аналитических машинах, с помощью штекеров и наборных полей. Хотя такой способ программирования и требовал много времени для подготовки машины к решению конкретной задачи, он позволял гибко менять счетные "способности" ENIAC.

Создание ENIAC было закончено в ноябре 1945 года. Машина представляла собой внушительное сооружение (более 30 м в длину, объем 85 $м^3$, вес 30 т), содержащее более 18000 электронных ламп и 1500 электромагнитных реле. Машина потребляла около 150 кВт электроэнергии.

Использование электронных ламп вместо механических и электромеханических элементов позволило резко увеличить скорость выполнения машинных операций. ENIAC тратил на умножение всего 0,0028 секунды, а на сложение и того меньше - 0,0002 секунды, что было гораздо быстрее, чем у любого известного в то время механического счетного устройства.

С момента ввода с эксплуатацию ENIAC началась бурная история развития электронной вычислительной техники. Уже через десять лет к 1955 году появились компьютеры, работающие по хранимой программе. Были разработаны базовые схемы основных устройств ЭВМ: арифметического устройства, оперативного запоминающего устройства, памяти программ на магнитных сердечниках и другие.

В 1950 году в СССР была введена в эксплуатацию первая российская ЭВМ - МЭСМ или малая электронно-счетная машина. Эта машина, разработанная и построенная под руководством С.А. Лебедева, позволила оценить возможности новой зарождающейся техники. В 1952 г. в Институте точной механики и вычислительной техники Академии наук СССР была создана большая электронно-счетная машина БЭСМ-1. Основная вычислительная часть машины была построена на 4000 электронных ламп, машина выполняла 10000 операций в секунду. В течении нескольких лет различные центры в СССР разработали и ввели в эксплуатацию ЭВМ Минск 1, Урал 1, Урал 2, Урал 4, М1, М3, БЭСМ2, Стрела.

Следующее десятилетие ознаменовалось появлением ЭВМ нового поколения - транзисторных. Применение транзисторов вместо электронных ламп позволило значительно уменьшить энергопотребление машины, ее габаритные размеры, увеличить вычислительную мощность. Появились первые языки программирования высокого уровня - Алгол-60, Кобол, Фортран. В 1957 году появилась первая ЭВМ с микропрограммным управлением. Идея микропрограммного управления, предложенная М. Уилксом и Д. Стринджером, заключается в реализации любой машинной команды в виде последовательности исполняемых микрокоманд. Такой подход позволил существенно упростить процесс программирования работы машин. В 1959 году был создан первый миникомпьютер для управления технологическими процессами. Машину с названием PDP1 разработала и начала выпускать компания DEC (Digital Equipment Corporation). В СССР первой полупроводниковой машиной была "Раздан 2". В 1967 году начат выпуск одной из самых мощных (в то время) машин в мире - БЭСМ 6. Эта машина с быстродействием 1 млн. операций в секунду содержала 60 000 транзисторов и более 200 000 полупроводниковых диодов. В стране был налажен серийный выпуск ЭВМ марок Урал 14, Урал 16, Минск 22, Минск

23, Минск 32, БЭСМ 3, БЭСМ 4, М220, М222, Мир 2, Наири. Вычислительные машины начали широко применяться не только в крупных научных центрах, но и на промышленных предприятиях и в высших учебных заведениях.

В течение следующих десяти лет компьютеры полностью обновились. На смену дискретным транзисторам пришли интегральные микросхемы. Существенно снизилась потребляемая мощность, однако габариты машин существенно не уменьшились. Причина в том, что разработчики резко увеличили вычислительную мощность машин. Появились первые операционные системы и существенно улучшились периферийные устройства. Машины оснастились дисплеями на электронно-лучевых трубках, различными печатающими устройствами, графопостроителями, накопителями информации на магнитных лентах и дисках и т.д. В 1971 году фирма Intel выпустила первый законченный процессор на одной микросхеме – 4-разрядный микропроцессор. Следующая генерация такого устройства появилась уже в 1973 году. Та же фирма выпустила легендарный I8080 - восьмибитовый однокристальный микропроцессор, способный выполнять набор из 75 команд.

Основная масса компьютеров этого времени выпускалась в формате мини-ЭВМ, т.е. основной конструктив - металлический шкаф, начиненный электроникой. Габариты машины диктовались размерами используемых элементов - интегральных микросхем.

В 1969 году в СССР было принято решение о создании Единой Серии ЭВМ (ЕС ЭВМ) и Системы Малых ЭВМ (СМ ЭВМ). В производство на территории СССР начали внедряться клоны машин ведущих мировых производителей. ЕС ЭВМ повторяли универсальную 32-битную мини-ЭВМ IBM360, а СМ ЭВМ повторяли 16-битную управляющую мини-ЭВМ PDP 11 фирмы DEC. Этим решением было положено начало отставания СССР в создании новой вычислительной техники.

Пока в СССР налаживали производство ЭВМ по чужим образцам, ведущие мировые производители вычислительных машин продолжали совершенствовать свою технику. Фирма IBM наращивала мощность своих мини-ЭВМ, продолжая развивать успешную линейку IBM360: IBM370, IBM380, IBM390. Фирма DEC разработала и начала выпуск мощной серверной 32-битной мини-ЭВМ VAX-11 и ее модификаций. Появились мощные мультипроцессорные суперкомпьютеры с общей производительностью более 100 млн. операций в секунду. Появились первые настольные персональные компьютеры.

Параллельно и не так заметно для населения развивалась цифровая управляющая техника. В настоящее время эта техника, использующая программируемые большие интегральные схемы, входит в состав практически всех технических устройств, окружающих человека и облегчающих его деятельность. Базируется вся эта масса цифровых устройств на весьма небольшом числе общих принципов функционирования

и способов реализации. Основная задача управляющего устройства – выполнить расчеты и сформировать сигнал управления в нужный момент времени. Поэтому вычислительная техника с ее возможностями накапливать, обрабатывать и преобразовывать информацию, подходит для реализации требуемых законов управления как нельзя лучше.

1.1.2 Классификация систем по видам сигналов

Современные системы управления представляют собой выделенные технические средства и работающих с ними людей, деятельность которых направлена не решение определенных задач, так называемых целей управления. Среди технических объектов обычно выделяют объект управления, исполнительные устройства, датчики и управляющие устройства (регуляторы). Обмен информацией между ними происходит с помощью сигналов.

В различных системах используются сигналы разных видов. Наибольшее распространение получили системы с непрерывным способом обмена информацией между устройствами. Так, например, регулирование температуры в печи может быть выполнено путем изменения подводимой мощности к нагревателю. В печи установлен датчик, выходной электрический сигнал которого в виде напряжения в каждый момент времени пропорционален измеряемой температуре. Величина подводимой мощности к нагревателю, значение температуры в печи и выходное напряжение датчика определены в каждый момент времени. Кроме того, сигналы непрерывны по времени, т.е. в каждый момент времени существуют производные всех сигналов по времени. Такие системы управления принято называть непрерывными или **аналоговыми.**

В современной технике широко применяются системы управления, у которых некоторые из сигналов не имеют производных в обычном смысле. Так, в наземной радиолокационной системе измерения координат летящего самолета информация о дальности до самолета определена только в определенные, **дискретные** моменты времени. Действительно, радиолокатор излучает в пространство зондирующий электромагнитный импульс. Дальность до самолета рассчитывается по времени прохождения зондирующего импульса до самолета и, после отражения, обратно до антенны радиолокатора. В течении этого времени информация о дальности до самолета полностью отсутствует, т.е. не существует. Информация о дальности до самолета определена только в моменты времени поступления в систему отраженного от самолета зондирующего импульса. Таким образом, в системе следящего радиолокатора присутствует специфический сигнал о дальности до самолета. Величина дальности и соответствующее значение

сигнала может иметь бесконечно малое приращение. Однако информация о дальности, т.е. конкретная величина сигнала, определена в системе только в фиксированные, дискретные моменты времени. Поэтому производная от такого сигнала по времени в обычном смысле не существует. О таких сигналах говорят, что сигнал **дискретен (квантован) по времени.** Системы управления, содержащие специфические сигналы, непрерывные по значению, но дискретные по времени, называют **импульсными** или **дискретными**.

Другой класс составляют системы, содержащие сигналы, не являющиеся непрерывными по уровню. Примером может служить система управления освещением комнаты. Мы управляем освещением с помощью выключателей. Выключатель имеет всего два состояния - "включено" и "выключено". Соответственно и сигнал управления освещением имеет всего два значения, которые можно обозначить " 1 " (т.е. включено) и " 0 " (т.е. выключено). Сигнал на выходе выключателя определен в каждый момент времени, однако он принимает только два значения; изменение значения сигнала может произойти в любой момент времени, но значение сигнала при этом изменяется скачком. Производная по времени от такого сигнала в обычном смысле не определена. Можно обобщить задачу управления освещением - ввести вместо выключателя некий многопозиционный переключатель, изменяющий скачками значение мощности, подводимой к осветителю. Ситуация окажется прежней: в системе управления присутствует сигнал, производная от которого по времени не существует, так как значение сигнала может изменяться скачком в любой момент времени. О сигнале, значение которого может принимать только конечное число вполне определенных значений, говорят, что он **дискретен (квантован) по уровню.** Системы управления, содержащие специфические сигналы, определенные в каждый момент времени, но дискретные по уровню, называют **релейными.**

Наконец, существуют системы, содержащие сигналы дискретные и по уровню, и по времени. К такому классу систем относятся все системы с управляющими компьютерами. Действительно, при вводе в компьютер сигнал обязательно оцифровывается, что эквивалентно квантованию уровня сигнала. При этом компьютер обрабатывает информацию о управляемом процессе по программе, циклически. Получив порцию информации, компьютер ее обрабатывает, затем получает новую информацию, и т.д. Этот процесс эквивалентен квантованию времени. Таким образом, в системе управления присутствует сигналы, производные от которых по времени не существуют, так как значение сигналов могут изменяться только скачками и только в определенные моменты времени. В остальное время значение сигналов либо неизменно, либо не определено. Системы управления,

содержащие сигналы, квантованные и по уровню, и по времени, называют **цифровыми**.

На рисунке 1.1.2 для наглядности приведена графическая интерпретация приведенной выше классификации систем по типам используемых сигналов. На горизонтальной оси условно отложены значения сигналов (уровни), которые могут быть либо непрерывными, либо дискретными. На вертикальной оси условно отложено время, которое также может быть либо непрерывным, либо дискретным. Из возможных комбинаций сигналов получается всего четыре типа систем: аналоговые, импульсные, релейные и цифровые. Из приведений классификации следует, что цифровые вычислительные устройства обязательно обладают двумя особенностями – информация в таких устройствах квантуется как по времени, так и по уровню.

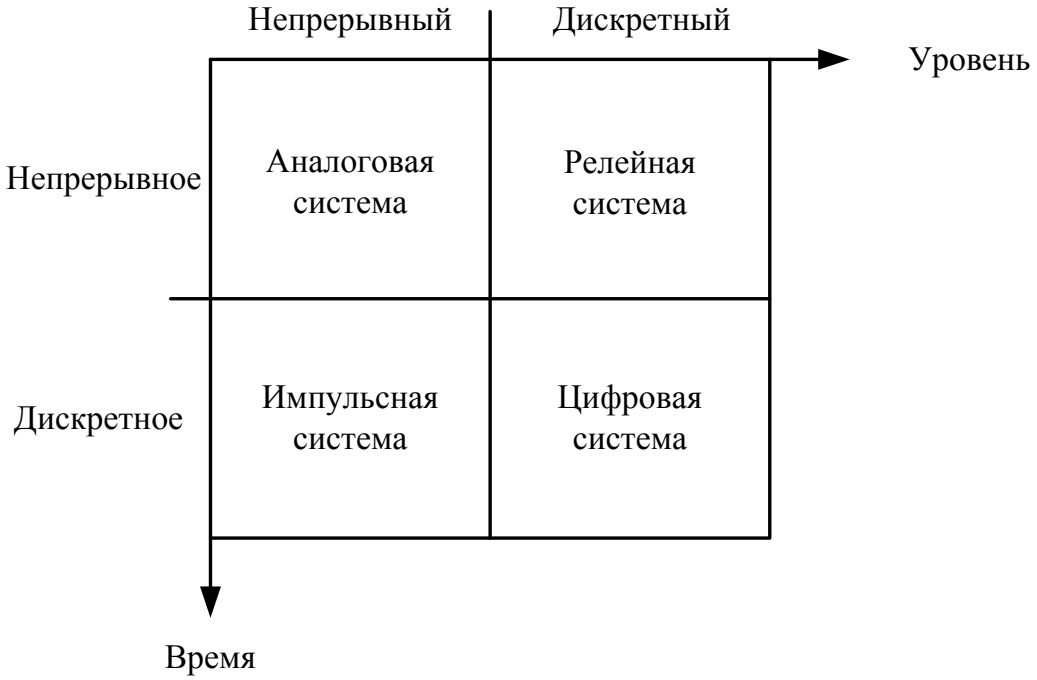

Рисунок 1.1.2 - Классификация систем по типам сигналов

1.1.3 Классификация цифровых вычислительных устройств

Электронные вычислительные машины (ЭВМ) по их целевому назначению можно разделить на два больших класса - **универсальные** и **управляющие.**

Универсальные ЭВМ предназначены для автоматизации интеллектуального труда человека. Это решение задач обработки данных, построения графиков, отчетов, ведение баз данных, финансовые расчеты и т.д. У таких машин нет конкретной специальной решаемой задачи.

Соответственно к машине предъявляются и требования - легкость перехода с задачи на задачу, удобный ввод-вывод информации для человека, наличие накопителя большой емкости для хранения данных, высокая вычислительная производительность и т.д. Темп вычислительного процесса в таких машинах должен соответствовать темпу работы человека.

Управляющие ЭВМ предназначены для решения задач автоматизации технологических процессов, управления физическими объектами, сбором и обработкой информации о ходе физического процесса. Это похожие задача, связанные с обработкой данных, решения математических задач, ведения баз данных. Однако есть и специфические особенности управляющих машин:

- машины должны иметь возможность получать информацию непосредственно от датчиков физических величин и передавать информацию на реальные исполнительные устройства систем управления, т.е. машины должны быть оборудованы специальными **устройствами связи с объектами (УСО);**
- программа работы управляющей машины меняется редко, так как машина используется в конкретной задаче управления длительное время; поэтому управляющая машина должна иметь специфическую память программ, как правило, с постоянно загруженной программой работы;
- темп вычислительного процесса управляющей машины должен быть жестко привязан к темпу технологического процесса, поэтому управляющая машина должна иметь специальные средства, обеспечивающие ее функционирование в режиме **реального времени.**

В то же время сами вычислительные операции (сложение, умножение, деление, логические операции, вычисление значений функций и т.д.) в обоих случаях могут выполняться одинаково. Выделяют три основных способа выполнения операций - **регистровый, табличный и матричный.**

Регистровый способ вычисления можно сравнить с вычислением по формуле, которая задает последовательность выполняемых операций. Промежуточные данные по ходу вычислений хранятся в регистрах оперативного запоминающего устройства. Основное достоинство регистрового способа вычислений - он требует относительно небольшой памяти для хранения данных. Следовательно, вычислительное устройство можно выполнить с использование небольшого количества физических элементов (например, транзисторов) и сделать дешевым. Поэтому исторически все первые ЭВМ использовали именно регистровый способ вычислений и в настоящее время это основной способ вычислений, используемый в процессорах.

Табличный способ вычислений основан на использовании в расчетах специальных таблиц с заранее занесенными данными. Наиболее известны таблицы для вычисления значений тригонометрических функций. Помещенные в долговременную память компьютера таблицы позволяют производить вычисления с наибольшим быстродействием. Однако это достоинство табличного способа сопровождается и существенным

недостатком - для хранения таблиц необходимо использовать относительно большое число физических элементов (тех же транзисторов). Именно поэтому долгое время табличный способ вычислений имел ограниченную область применения в ЭВМ.

Матричный способ вычислений является разновидностью регистрового способа. Он основан на использовании матричной сетки при решении конкретной задачи. Другое название способа - **векторные вычисления.** Основное достоинство способа - он позволяет легко распараллелить вычислительный процесс, т.е. выполнять решение задачи на нескольких одновременно работающих скалярных (обычных) процессорах.

Использование в ЭВМ различных способов выполнения вычислений привело к появлению в техническом языке специальных терминов. Классические процессоры, основанные на использовании регистрового способа вычислений, получили аббревиатуру **CISC - процессоры** (*complex instruction set computing* — компьютер с полным набором команд). Относительно небольшое количество транзисторов, необходимое для реализации каждой функции, позволило разработчикам заложить в компьютер большое число различных исполняемых команд. Плата за такое разнообразие машинных команд - низкая вычислительная эффективность процессора.

Одно из современных направлений проектирования процессоров привело к появлению на компьютерном рынке **RISC – процессоров** (*Restricted (reduced) instruction set computing* — компьютер с сокращённым набором команд). В этих компьютерах машинные команды реализуются табличным способом, что позволяет резко увеличить скорость вычислений. Однако громоздкость вычислительных блоков, реализующих машинные команды, привела к заметному сокращению числа и уменьшению разнообразия команд процессора, что и отразилось в их названии.

Матричный способ вычислений нашел широкое применение при создании **суперкомпьютеров,** т.е. компьютеров, специально создаваемых для решения особо громоздких задач, требующих обработки огромного количества данных за ограниченное время и, соответственно, исключительно высокого быстродействия. К таким задачам обычно относят задачи, связанные с обработкой данных метеоспутников при составлении прогнозов погоды, обработка данных геологических исследований больших территорий, обработкой сейсмических сигналов и др.

Развитие цифровой техники показало, что и реализовать вычисления технически можно различными способами. Большинство современных цифровых вычислительных машин используют **программный** способ реализации вычислений. Этот способ вычислений основан на последовательном выполнении действий, предписанных заданной программой вычислений. При таком способе реализации вычисления в машине разворачиваются последовательно во времени. При замене

программы работы машина перестраивается на другую задачу. Очевидны достоинства такого способа реализации вычислений:

- вычислительная машина имеет типовую архитектуру (универсальность) и может выпускаться большими партиями;
- смена решаемой задачи выполняется простой заменой программы (гибкость);
- от программиста не требуется специальных инженерных знаний, высокой технической квалификации и знания специфики вычислительного устройства (низкие эксплуатационные расходы);
- универсальность архитектуры позволяет приспособить вычислительную машину для рения новой задачи за короткое время (низкая себестоимость новой разработки).

Очевидны также и недостатки программного способа реализации вычислений:

- развертывание вычислений во времени приводит к неизбежным потерям вычислительного времени, т.е. относительно низкому быстродействию системы;
- решение самой примитивной задачи требует использования вычислительных средств типовой архитектуры, т.е. такой же, как и при решении сложной задачи (неэффективность использования технических средств);
- любую задачу универсальное вычислительное устройство, опираясь на универсальные алгоритмы, решает не оптимально (вычислительная неэффективность).

Альтернативой программной реализации вычислений может служить разработка узкоспециализированного вычислителя, архитектура которого отражает специфику решаемой задачи и реализованного на специализированных цифровых блоках. Такой специализированный вычислитель обычно может решать одну конкретную задачу, но наиболее быстро и эффективно. Подобный подход называется **аппаратной** реализация вычислений**.**

Аппаратной реализации вычислений присущи следующие достоинства:

- вычисления в устройстве выполняются различными специализированными блоками зачастую параллельно во времени, т.е. вычисления “разворачиваются в пространстве”, можно создать устройство максимального быстродействия;
- в устройстве используются только те вычислительные блоки, которые необходимы для решения данной задачи, технические вычислительные средства используются максимально эффективно;
- в устройстве не используется какая-либо программа работы, требуемый алгоритм вычислений “зашит” в связях между блоками, между блоками отсутствуют промежуточные пересылки данных.

В то же время аппаратному способу вычислений присущи и существенные недостатки:

- узкая специализация вычислителя делает рынок его потребителей очень узким, следовательно, вычислитель выпускается малыми партиями;

- разработка вычислителя требует привлечения разработчиков высокой квалификации, хорошо знающих специфику машинных вычислений и цифровой электроники, что ведет к удорожанию разработки;

- структура вычислителя не допускает решения каких-либо задач, отличных от заложенной в его архитектуре, т.е. отсутствует вычислительная гибкость;

- стоимость вычислителя получается очень высокой, сроки его разработки большие.

Низкая гибкость и высокая стоимость реализации аппаратного способа вычислений заставляет использовать так называемый **микропрограммный** способ реализации вычислений. Микропрограммный способ по своей идее аналогичен программному способу, однако при его реализации используется управление ходом вычислительного процесса на очень низком уровне. За счет этого удается существенно повысить скорость обработки информации, приближаясь к аппаратному способу вычислений. Именно этот способ реализации вычислений используется в **цифровых сигнальных процессорах** – устройствах для высокоскоростной цифровой обработки сигналов (англ. **DSP** – Digital Signal Processer). При этом сохраняется гибкость и универсальность вычислительной структуры, что характерно для программного способа вычислений. Можно считать, что микропрограммный способ вычислений занимает промежуточное место между программным и аппаратным способами.

К недостаткам микропрограммного способа реализации вычислений можно отнести следующее:

- разработка программного обеспечения требует привлечения разработчиков очень высокой квалификации, отлично знающих цифровую электронику и специфику архитектуры используемого вычислителя;

- разработка программного обеспечения весьма трудоемка и требует больших затрат времени;

- использование универсальных языков программирования высокого уровня зачастую бывает невозможно.

Рассмотренные выше три способа выполнения вычислений и три способа реализации вычислителей допускают любые их сочетания. Например, регистровый способ вычислений может быть использован как при аппаратной, так и при программной или микропрограммной реализациях вычислительного устройства. Сказанное относится и ко всем остальным способам вычислений.

Достигнутый уровень производства современной электроники, выражающийся в ее миниатюризации, определяет основные факторы, определяющие политику проектирования новых вычислительных устройств и систем с их использованием. Объем аппаратных затрат на создание устройства становится малозначительным фактором. На лидирующие

позиции выходят факторы, продолжающие определять себестоимость продукции.

К таким факторам следует отнести в первую очередь время проектирования нового устройства. Руководство фирмы Texas Instruments уверяет, что от начала разработки любого заказного цифрового устройства до начала выпуска опытной партии специалистам фирмы потребуется не более двух недель, [16]. Такие короткие сроки становятся возможными в связи с широким внедрением новых технологий в процесс проектирования устройств. Развитие технологии создания микросхем с программируемыми пользователем связями (**ПЛИС** – программируемые логические интегральные схемы, англ. **PLD**) позволило создать микросхемы с универсальной и гибко изменяющейся архитектурой. Выпускаемые современной промышленностью базовые кристаллы с огромным количеством встроенных логических элементов, переключаемые связи между элементами и специализированные пакеты автоматизации проектирования, все это породило новое направление в создании цифровых устройств – **системы на кристалле** (англ. **SoC** - System on Chip). Технология SoC позволяет пользователю создать на покупной серийной микросхеме свое цифровое устройство, используя любой из перечисленных выше девяти способов построения архитектуры устройства и за счет этого снизить себестоимость проекта.

1.2 Дифференциальные анализаторы и аналоговая вычислительная техника

В современных системах автоматического управления используются регуляторы, действие которых основано на решении дифференциальных или интегральных уравнений в реальном времени. В настоящее время процедуры проектирования таких регуляторов хорошо отработана и описана практически во всех учебниках по теории автоматического управления. Это стало возможно благодаря работам многих ученых, исследовавших основные способы выполнения различных вычислений.

Клод Шеннон в 1941 году опубликовал статью "Математическая теория дифференциального анализатора", [25]. В этой основополагающей статье он рассмотрел возможность использования только трех элементов - **сумматора, блока умножения** и **интегратора** для выполнения вычислений. Было показано, что с помощью конечного числа перечисленных блоков можно:

- точно решить обыкновенное дифференциальное уравнение, если нелинейные функции, входящие в него, не являются гипертрансцендентными;
- вычислить значение любой функции, не являющейся гипертрансцендентной. Такие функции называют **воспроизводимыми.**

Кроме того Шеннон показал, что для любой функции $y = f(x_1, x_2, \ldots, x_n)$, непрерывной относительно всех аргументов, можно найти воспроизводимую функцию этих аргументов $y1 = F(x_1, x_2, \ldots, x_n)$, такую, что $|y - y1| < \varepsilon$, где ε - произвольно малое, наперед заданное число. В итоге Шеннон обосновал возможность воспроизведения любой непрерывной функции с заданной степенью точности.

Из указанной работы Шеннона вытекают следующие базовые положения вычислительной техники:

- всякая аналитическая функция* может быть воспроизведена точно с помощью конечного числа интеграторов, сумматоров и блоков умножения;
- с помощью конечного числа интеграторов, сумматоров и блоков умножения всякая непрерывная функция может быть воспроизведена приближенно, но с любой заранее заданной точностью.

Таким образом, для технической реализации вычислительного устройства требуется в первую очередь создать три базовых элемента – **сумматор, умножитель и интегратор**.

В 50 - 60 годы двадцатого века были разработаны схемы и налажен выпуск **аналоговых вычислительных машин (АВМ)**. В основе работы этих вычислителей лежали электронные блоки для выполнения операций интегрирования по времени, суммирования и умножения электрических сигналов, величины которых моделировали воспроизводимые функции времени. Была отработана концепция операционного усилителя - специального усилителя постоянного тока, обладающего повышенной точностью работы, высокой линейностью статической характеристики, а главное - высоким коэффициентом усиления (100 000 и более). На базе операционного усилителя и строились основные блоки аналоговой вычислительной машины.

На рисунке 1.2.1 показаны электрические схемы построения базовых блоков АВМ. Блок машины содержит один операционный усилитель и пассивные элементы обвязки. На рисунке 1.2.1а показана схема масштабирующего сумматора. Выходное напряжение операционного усилителя в приведенной схеме составит:

$$U_{вых}(t) = - K_1 * U_1(t) - K_2 * U_2(t) - K_3 * U_3(t),$$

где: $K_1 = R_{ос} / R_1$; $K_2 = R_{ос} / R_2$; $K_3 = R_{ос} / R_3$ - постоянные коэффициенты. Блок масштабирующего сумматора позволяет выполнить операции суммирования сигналов и умножения их на постоянные коэффициенты.

На рисунке 1.2.1б показана схема блока интегрирования сигналов по времени. В отличие от предыдущего блока, в обратной связи операционного усилителя интегратора установлен конденсатор С. Выходное напряжение

Аналитическая функция - функция, которая может быть разложена в степенной ряд.

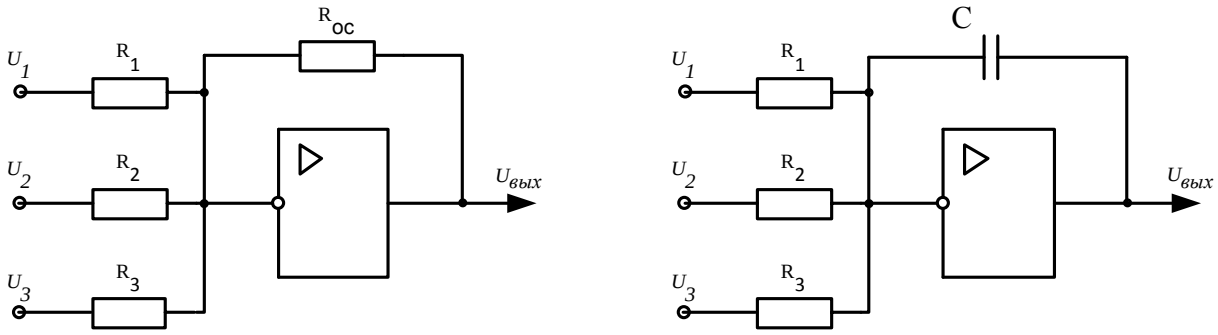

Рисунок 1.2.1 - Базовые элементы аналоговой вычислительной техники: масштабирующий сумматор (а) и интегратор (б)

операционного усилителя в приведенной схеме равно:

$$U_{вых}(t) = \int_{t_0}^{t} (- K_1 * U_1(\tau) - K_2 * U_2(\tau) - K_3 * U_3(\tau)) * d\tau ,$$

где: $K_1 = 1 / R_1 * C$; $K_2 = 1 / R_2 * C$; $K_3 = 1 / R_3 * C$ - постоянные коэффициенты.

Таким образом, аналоговые вычислительные машины, построенные на основе недорогих электронных блоков, позволяли решать значительный круг задач анализа и моделирования систем автоматического управления, синтеза динамических регуляторов. В научных и высших учебных заведениях Советского Союза аналоговые вычислительные машины марки МН-7, портативные аналоговые моделирующие установки ПАМУ и "Экстрема", аналоговые и аналого-цифровые вычислительные комплексы АВК-31, АВК-32 использовались до конца 80-х годов двадцатого столетия. В последствии их полностью заменили цифровые вычислительные машины; однако за многие годы эксплуатации были отработаны основные методики моделирования и разработаны базовые схемные реализации аналоговых промышленных регуляторов систем автоматического управления, получившие обобщающее название - **метод дифференциальных анализаторов**. Накопленный опыт работы с аналоговой вычислительной техникой естественно был использован при работе на цифровой вычислительной технике. Разработанные методики реализации алгоритмов обработки информации на цифровой вычислительной технике, перенесенные с аналоговых вычислительных машин, получили название метод цифровых дифференциальных анализаторов.

1.3 Метод цифровых дифференциальных анализаторов

К концу 60-х годов двадцатого столетия был налажен серийный выпуск не очень дорогих цифровых управляющих машин, известных под названием "**устройство числового программного управления" - УЧПУ.** Широкое применение этих устройств в промышленности породило потребность в

методиках синтеза цифровых регуляторов для систем управления. Была сформулирована задача - найти способы перевода отработанных алгоритмов аналогового управления на новую цифровую базу. Такая методика была создана в конце 60-х, начале 70-х годов и получила название **метод цифровых дифференциальных анализаторов.**

В 1960 году вышла монография Воронова А.А (в последствии академика Академии Наук СССР) " Цифровые аналоги для систем автоматического управления " в которой были изложены основные принципы метода цифровых дифференциальных анализаторов, [7]. В работе показано, что для реализации цифровых вычислительных устройств требуется в первую очередь создать три базовых цифровых устройства – цифровой **сумматор,** цифровой **умножитель** и цифровой **элемент задержки**. В основе метода лежат несколько постулатов:

- во-первых, предполагается, что основная особенность цифрового управления заключается в наличии квантования времени; эффектами квантования уровня сигналов очень часто можно пренебречь;
- во-вторых, предполагается, что известно решение задачи в непрерывном времени, которое служит эталоном;
- в-третьих, предполагается, что эталонное решение задачи получено в форме дифференциального уравнения, либо передаточной функции, либо структурной схемы искомого регулятора автоматической системы.

Ставится задача: преобразовать эталонный аналоговый регулятор в дискретный аналог, используя формальные алгоритмы, так, чтобы сохранить все существенные свойства (показатели качества) исходной замкнутой системы. Так как решение задачи в дискретном времени может быть описано либо разностным уравнением, либо модифицированной передаточной функцией, либо структурной схемой с дискретным временем, то были разработаны соответствующие формальные методики. Некоторые из них изложены в учебных пособиях [9,17,20,23].

На рисунке 1.3.1 изображено структурное представление метода цифровых дифференциальных анализаторов. Исходная задача первоначально решается в непрерывном времени и синтезируется эталонный регулятор в форме дифференциального уравнения, передаточной функции или структурной схемы. Далее необходимо выбрать величину периода квантования времени для цифрового аналога. Обычно это делается на основе теоремы Котельникова-Шеннона, определяющей условия точного восстановления непрерывного сигнала по дискретным отсчетам, [20, 21].

После определения величины периода квантования времени выбирается способ численного интегрирования переменных для цифрового аналога. Так как цифровая машина не может вычислять интеграл точно, разработано достаточно большое количество приближенных способов численного интегрирования функций времени. Каждый из способов обладает определенными особенностями, достижимой точностью и областью применения. Поэтому разработаны специальные методы замены

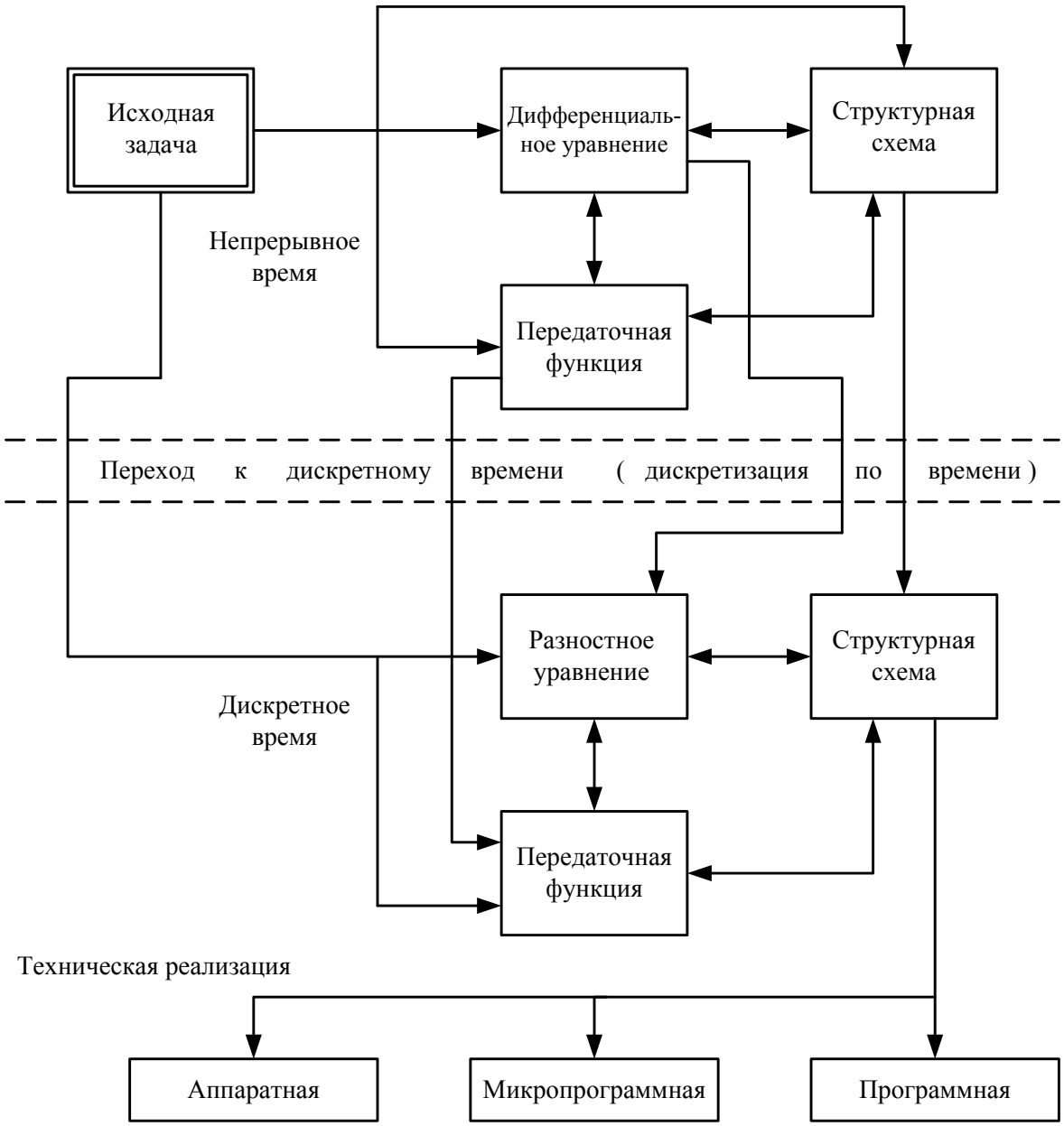

Рисунок 1.3.1 - Структура метода цифровых дифференциальных анализаторов

непрерывного времени дискретным, которые учитывают специфические особенности способов интегрирования, [9, 20].

В результате выполнения перечисленных выше действий получается решение задачи в дискретном времени в виде разностного уравнения с известными параметрами, либо в виде модифицированной передаточной функции, либо в виде структурной схемы регулятора. Полученный результат позволяет перейти непосредственно к технической реализации цифрового аналога - в виде аппаратного, микропрограммного либо программного устройства.

Следует отметить, что к настоящему времени хорошо разработаны методики решения исходной технической задачи непосредственно в дискретном времени по аналогии с подходами, применяемыми в непрерывном времени. Эти методики также стали относить к методу цифровых дифференциальных анализаторов.

Таким образом, отличительная особенность метода цифровых дифференциальных анализаторов заключается в учете эффектов квантования времени при синтезе регулятора системы управления. Эффектами квантования уровня на этом этапе разработки устройств как правило пренебрегают. Влияние квантования уровня на качественные показатели работы системы проверяется методом моделирования после проведения синтеза регулятора.

1.4 Синтез схемы генератора задающего воздействия

Для иллюстрации процедуры использования метода дифференциальных анализаторов, рассмотрим пример синтеза схемы генератора задающего воздействия, [7].

Техническая задача состоит в следующем. Требуется разработать задающее устройство для графопостроителя. Задающее устройство должно вырабатывать сигналы *x(t)* и *y(t)*, *t* – реальное время, которые задают движение пера графопостроителя на плоскости по кривой второго порядка. Другими словами, координаты пера графопостроителя в каждый момент времени должны удовлетворять уравнению кривой второго порядка, которое в общем виде имеет вид:

$$Ax^2+2Bxy+Cy^2+2Dx+2Ey+F=0, \qquad (1.4.1)$$

где: *A, B, C, D, E, F* – заданные постоянные коэффициенты.

Метод дифференциальных анализаторов предполагает использование дифференциальных уравнений. Поэтому от алгебраического уравнения (1.4.1) перейдем к уравнению в дифференциалах:

$$2Axdx+2Bydx+2Bxdy+2Cydy+2Ddx+2Edy=0 .$$

Разделив на 2 и группируя члены, получаем уравнение

$$(Ax+By+D)dx=-(Bx+Cy+E)dy$$

или

$$\frac{dy}{dx}=-\frac{Ax+By+D}{Bx+Cy+E} \quad . \qquad (1.4.2)$$

Для создания генератора задающего воздействия в уравнение (1.4.2) нужно ввести время. Формально разделим числитель и знаменатель дроби в левой части (1.4.2) на *dt*, получим

$$\frac{\frac{dy}{dt}}{\frac{dx}{dt}} = -\frac{Ax(t) + By(t) + D}{Bx(t) + Cy(t) + E} \quad . \qquad (1.4.3)$$

Уравнение (1.4.3) будет выполнено, если движение пера графопостроителя соответствует системе дифференциальных уравнений:

$$\begin{aligned} \frac{dy(t)}{dt} &= -k[Ax(t) + By(t) + D], y(0) = y_0, \\ \frac{dx(t)}{dt} &= k[Bx(t) + Cy(t) + E], x(0) = x_0, \end{aligned} \qquad (1.4.4)$$

где: *k* – некоторый коэффициент; x_0, y_0 – координаты начального положения пера графопостроителя.

Уравнения (1.4.4) описывают требуемое движение пера графопостроителя по траектории, соответствующей заданной кривой второго порядка. По уравнениям легко построить искомую структурную схему задающего генератора. Для этого следует:

-предположив, что все переменные в правой части уравнений (1.4.4) известны, построить схему для вычисления переменных $\frac{dx}{dt}$ и $\frac{dy}{dt}$;

- используя интеграторы вычислить переменные $x(t)$ и $y(t)$;

- ввести в схему обратные связи, обеспечив реализацию вычислений правых частей уравнений.

На рисунке 1.4.1 показана итоговая структурная схема синтезированного генератора задающего воздействия. Как и было показано в работах К. Шеннона, для реализации устройства необходимы элементы только трех типов – сумматоры, умножители и интеграторы. Отметим также, что полученное решение определяет генератор, работающий в непрерывном времени, ориентированный на использование аналоговых элементов. Количество интеграторов в схеме (два) равно порядку дифференциального уравнения (1.4.4), что соответствует теории. Величина коэффициента *k* в уравнениях (1.4.4) и на структурной схеме влияет на скорости изменения переменных *x(t)* и *y(t)*, т.е. определяет скорость движения пера графопостроителя.

Цифровая реализация генератора по методу цифровых дифференциальных анализаторов требует выполнения процедуры дискретизации времени. Это вопрос изложен в следующих параграфах.

1.5 Методы цифрового интегрирования. Метод Эйлера

Рассмотрим возможность реализации схемы задающего генератора (см. рисунок 1.4.1) на цифровом вычислительном устройстве.

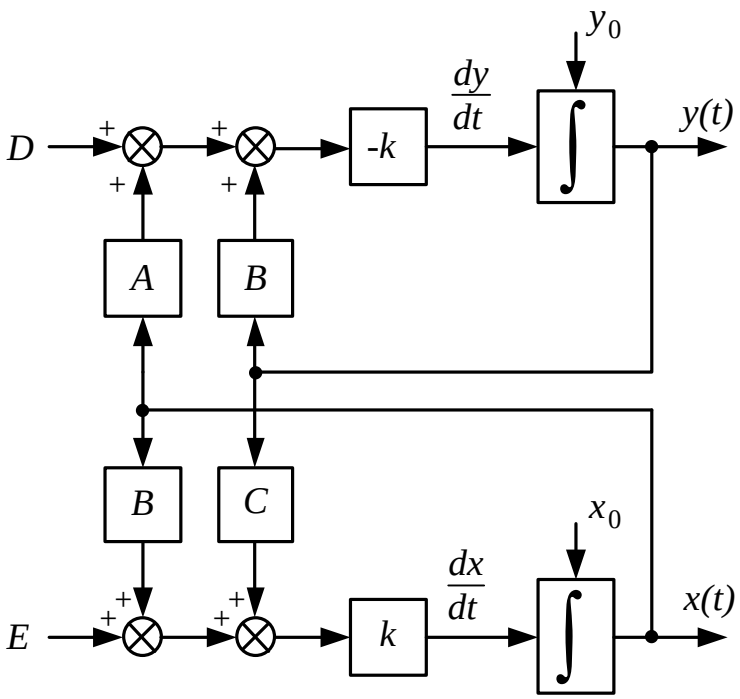

Рисунок 1.4.1 – Структурная схема генератора задающего воздействия

Цифровая реализация алгебраических операций суммирования и умножения, как будет показано ниже, не представляет технических трудностей. Другое дело – реализация интегратора, когда интегрируемая переменная изменяется в темпе реального времени.

Операцию интегрирования переменной *x(t)* по времени можно интерпретировать геометрически, как вычисление площади *S(t)* под кривой *x(t)*:

$$S(t) = \int_{t_0}^{t} x(\tau) * d\tau , \qquad (1.5.1)$$

т.е. интеграл представляется суммой произведений значений *x(t)* на бесконечно малое приращение времени *dt*. Но любое цифровое вычислительное устройство работает по тактам, обрабатывая последовательность значений сигнала *x(0T), x(1T), x(2T), ..., x(nT)*, где *n* – дискретное время (номер такта), *T* – период квантования времени (длительность такта). В результате оказывается принципиально невозможным посредством цифровой техники реализовать бесконечно малое приращение *dt* и вычислить интеграл по формуле (1.5.1). На рисунке 1.5.1 показано представление интегрируемой переменной *x(t)* дискретными отсчетами *x(nT), n*=0,1,2 … с периодом *T* квантования времени.

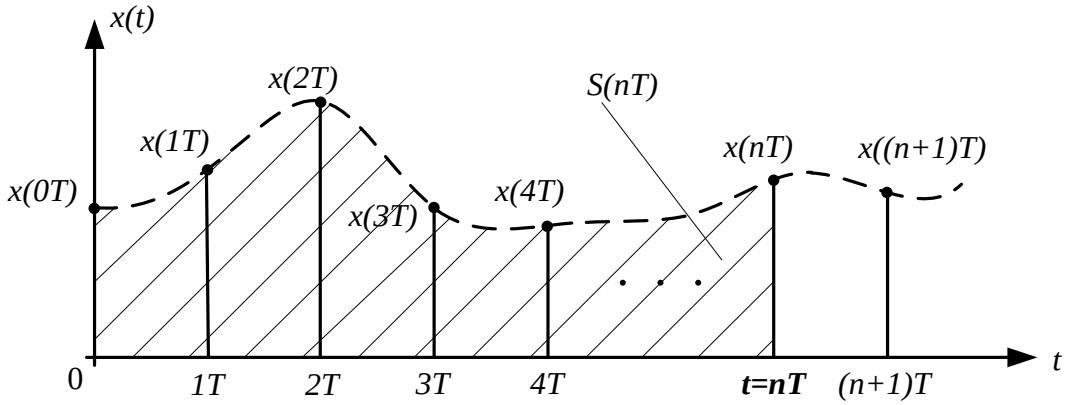

Рисунок 1.5.1 – Представление переменной *x(t)* отсчетами *x(nT)* с периодом квантования *T* при вычислении интеграла *S(t)*

Для преодоления указанной трудности по методу цифровых дифференциальных анализаторов следует заменить точное вычисление интеграла (1.5.1) приближенным. Одним из наиболее распространенных методов приближенного интегрирования является метод Эйлера.

В основе метода Эйлера лежит аппроксимация площади под кривой *x(t)* суммой элементарных прямоугольников, площадь которых равна произведению *T*x(t)*. Другими словами, предполагается, что на интервале дискретности *T* величина сигнала *x(t)* изменяется незначительно. Если положить, что к моменту времени t=nT площадь под кривой x(t) составляла *S(nT)*, то на следующем такте работы цифрового устройства при *t=(n+1)T* площадь под кривой может быть вычислена по формуле

$$S((n+1)T) = S(nT) + T * x(nT), \qquad (1.5.2)$$

где: *x(nT)* – отсчет значения *x(t)* в текущий момент времени *t=nT*. На рисунке 1.5.2 показана геометрическая интерпретация цифрового аналога интегрирования по методу Эйлера.

Уравнение (1.5.2) представляет собой разностное уравнение первого порядка, справедливое для дискретных моментов времени *t=nT*, n=0,1,2 … Отметим, что величина квантования времени *T* входит параметром в уравнение (1.5.2) и должна быть известна. Если величина периода квантования времени постоянная, то уравнение (1.5.2) стационарное и его свойства легко предсказуемы. Если период *T* переменный, то рассматриваемое уравнение становится нестационарным и анализ свойств цифровой системы существенно усложняется. Поэтому при практической реализации технических устройств особое внимание следует уделить вопросу обеспечения заданного постоянного значения величины *T*.

На рисунке 1.5.3 приведена структурная схема цифрового аналога интегратора по методу Эйлера. Для реализации схемы требуется динамический элемент D1 – элемент запоминания значения сигнала на время *T*. Его называют дискретным элементом задержки сигнала на время *T* или

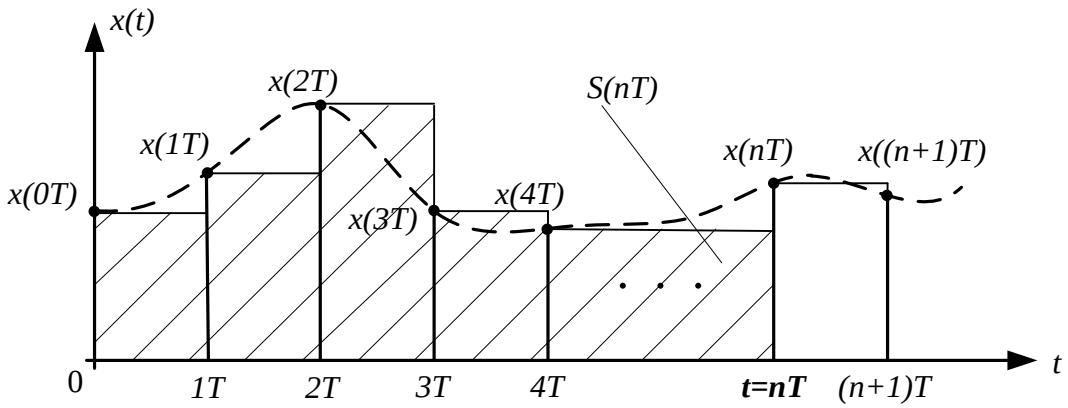

Рисунок 1.5.2 – Вычисление интеграла *S(nT)* по методу Эйлера

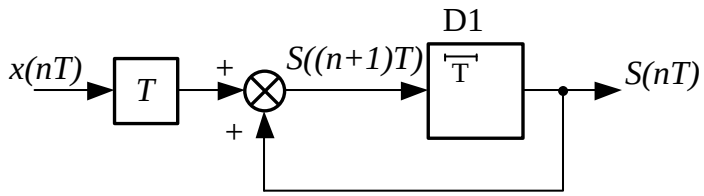

Рисунок 1.5.3- Структурная схема цифрового аналога интегратора по методу Эйлера

просто **элементом задержки.**

Используя цифровой аналог интегратора, нетрудно изобразить структурную схему генератора задающего воздействия, пригодную для реализации на цифровом вычислительном устройстве. Для этого в схеме на рисунке 1.4.1 заменим время $t=nT$ и аналоговые интеграторы поменяем на цифровые. Итоговая схема цифрового аналога генератора задающего воздействия показана на рисунке 1.5.4.

В схеме на рисунке 1.5.4 использованы вычислительные блоки только трех типов – сумматор, блок умножения и элемент задержки. Использование элементов этих трех типов характерно для метода цифровых дифференциальных анализаторов. Заметим также, что количество элементов задержки равно порядку решаемого уравнения, т.е. двум.

Часто при разработке цифрового аналога непрерывное устройство задано своей передаточной функцией *W(s),* где *s* – оператор преобразования Лапласа. Требуется найти соответствующую дискретную передаточную функцию. Эту задачу можно решить следующим простым способом.

Найдем дискретную передаточную функцию цифрового аналога интегратора. Для этого применим к обеим частям уравнения (1.5.2) дискретное Z-преобразование, [9]. Так как по определению передаточная функция вычисляется при нулевых начальных условиях, то в результате получаем

$$zS(z) = S(z) + T * X(z), \qquad (1.5.3)$$

где: *z*- независимая переменная Z-преобразования; *S(z)* – образ переменной *S(nT); X(z)* – образ переменной *x(nT).*

Преобразуем уравнение (1.5.3) к виду

$$(z-1)S(z)=T*X(z),$$

или

$$\frac{S(z)}{X(z)}=W^{*}(z)=\frac{T}{z-1}, \qquad (1.5.4)$$

где: *W*(z)* – искомая передаточная функция цифрового аналога интегратора по методу Эйлера.

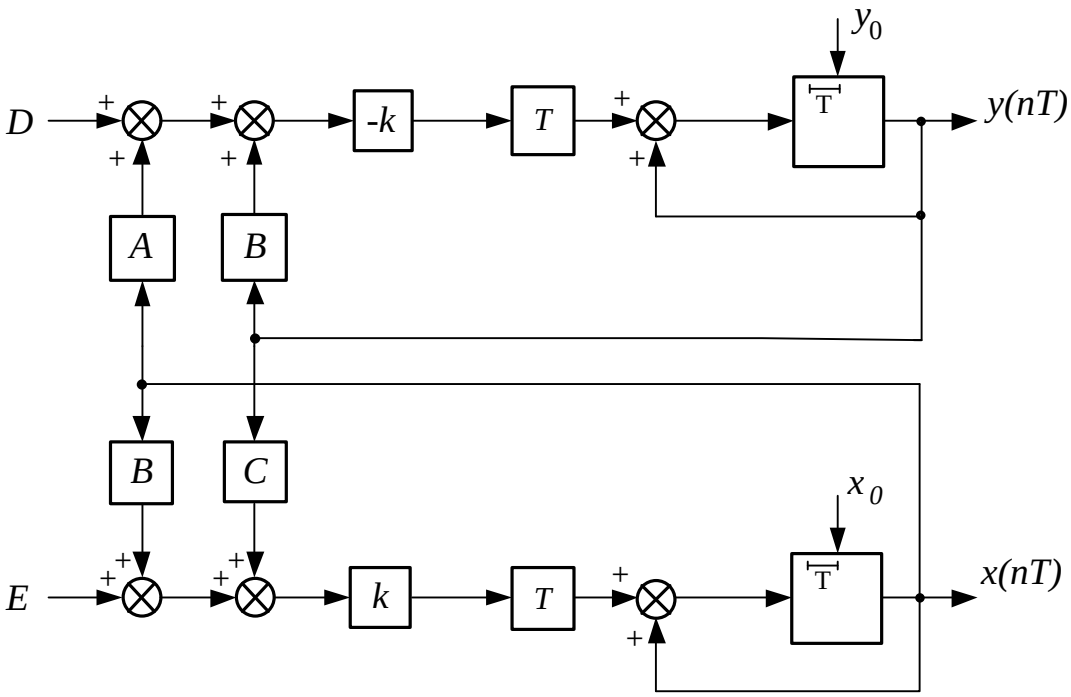

Рисунок 1.5.4- Структурная схема цифрового генератора задающего воздействия

С другой стороны, известно, что передаточная функция непрерывного интегратора равна

$$W(s)=\frac{1}{s}\ .$$

В цифровом устройстве его необходимо заменить на блок с передаточной функцией (1.5.4). При аналитическом вычислении дискретной передаточной функции можем использовать формальную замену

$$\frac{1}{s}=\frac{T}{(z-1)},$$

или

$$s = \frac{(z-1)}{T}. \qquad (1.5.5)$$

Другими словами, передаточную функцию цифрового аналога по методу Эйлера можно получить по известной передаточной функции непрерывного устройства формальной заменой независимой переменной *s*:

$$W^*(z) = W(s)\big|_{s=(z-1)/T} \ . \qquad (1.5.6)$$

Метод Эйлера имеет существенный недостаток. Он не гарантирует сохранения показателей качества и свойства устойчивости цифровой системы, когда исходная непрерывная система устойчивая. Поэтому при применении метода полученную цифровую систему следует проверить на устойчивость и исследовать показатели качества динамических процессов.

1.6 Методы цифрового интегрирования. Метод трапеций

В вычислительной математике разработано большое количество различных методов цифрового интегрирования, улучшающих метод Эйлера. В качестве примера, показывающего пути совершенствования метода Эйлера, рассмотрим метод трапеций, [21].

В основу метода трапеций положена гипотеза о том, что интегрируемая переменная на интервале дискретности изменяется линейно. В этом случае площадь под кривой x(t) аппроксимируется суммой элементарных трапеций, площадь каждой из которых равна

$$\frac{T}{2}[x(nT) + x((n+1)T)].$$

На рисунке 1.6.1 показана геометрическая интерпретация цифрового интегрирования непрерывной переменной x(t), представленной дискретными отсчетами *x(nT), n* =0,1,2, …. Если предположить, что в момент времени *t=nT* площадь под кривой *x(t)* равнялась *S(nT),* то в момент времени *t=(n+1)T* площадь под кривой может быть вычислена, как

$$S((n+1)T) = S(nT) + \frac{T}{2}[x(nT) + x((n+1)T)] \ . \qquad (1.6.1)$$

Уравнение (1.6.1) определяет алгоритм работы цифрового интегратора по методу трапеций. Применив к обеим частям уравнения Z-преобразование при нулевых начальных условиях, получим:

$$zS(z) = S(z) + \frac{T}{2}[X(z) + zX(z)],$$

или

$$(z-1)S(z)=\frac{T}{2}(z+1)X(z), \qquad (1.6.2)$$

где: *S(z)* –образ переменной *S(nT); X(z)* –образ переменной *x(nT); z*-независимая переменная Z-преобразования. Из уравнения (1.6.2) получаем передаточную функцию цифрового интегратора по методу трапеций:

$$W^*(z)=\frac{S(z)}{X(z)}=\frac{T}{2}\frac{(z+1)}{(z-1)}. \qquad (1.6.3)$$

Анализ свойств систем с интегрированием по методу трапеций показывает, что если исходная непрерывная система устойчивая, то получаемая цифровая система также будет устойчивая. Гарантия сохранения

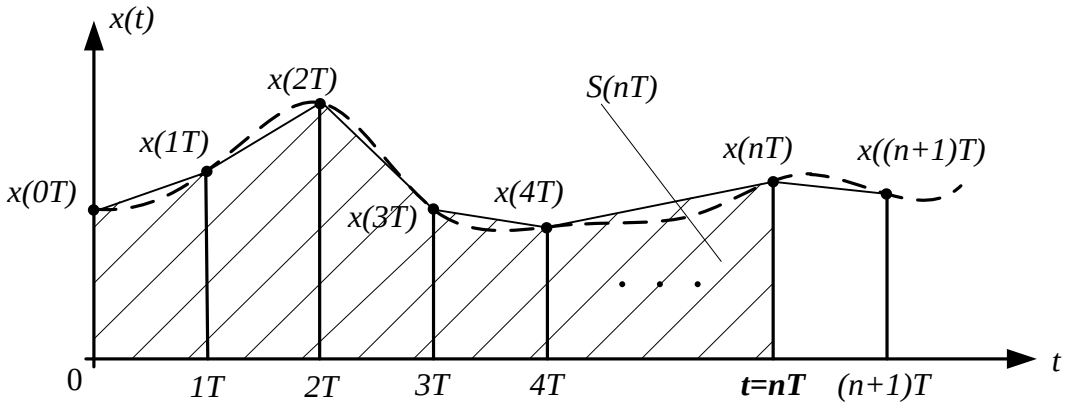

Рисунок 1.6.1 – Вычисление интеграла *S(nT)* по методу трапеций

свойства устойчивости является основным преимуществом метода трапеций по сравнению с методом Эйлера.

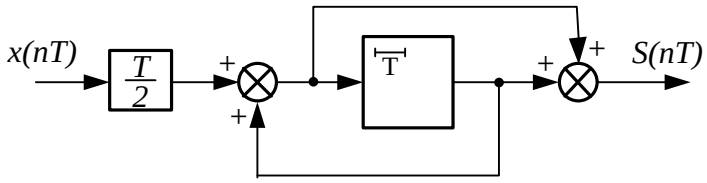

Рисунок 1.6.2 – Структурная схема цифрового интегратора по методу трапеций

Уравнение (1.6.1) цифрового интегратора по методу трапеций является разностным уравнением 1-го порядка. Поэтому для реализации такого интегратора необходим только один элемент задержки. Структурная схема интегратора, построенная по уравнению (1.6.1) показана на рисунке 1.6.2.

На практике метод трапеций применяют аналогично методу Эйлера. При

известной передаточной функции W(s) непрерывной системы для нахождения цифрового аналога можно сделать формальную замену переменной

$$s = \frac{2}{T}\frac{(z-1)}{(z+1)}$$

и получить передаточную функцию дискретного аналога. Приведенная замена переменной получается приравниванием передаточной функции цифрового интегратора (1.6.3) к передаточной функции непрерывного интегратора, равной *1/s*.

В случае, когда непрерывная система задана своей структурной схемой, например, как на рисунке 1.4.1, то для получения структурной схемы цифрового устройства необходимо заменить каждый непрерывный интегратор его цифровым аналогом (показан на рисунке 1.6.2). Кроме того, следует непрерывное время в исходной схеме заменить дискретной подстановкой $t=nT$.

Как уже отмечалось, в вычислительной математике разработано много методов цифрового интегрирования. В частности, в задачах моделирования динамических процессов на ЭВМ применяют методы интегрирования с переменным шагом T. При технической реализации устройств управления реальными объектами такие методы обычно не используют. Это связано с тем, что при моделировании процессов на ЭВМ мы имеем дело с виртуальным управляемым машинным временем. Имеется возможность оперативно изменять период квантования времени, прогнозировать и корректировать результат расчетов по ходу решения задачи. При технической реализации устройств управления мы имеем дело с реальным неуправляемым временем, ход которого от нас не зависит. Разработчику системы управления необходимо «привязаться» к ходу реального времени. Проще всего это сделать, обеспечив в техническом устройстве постоянство периода квантования T и используя соответствующий метод интегрирования. Указанное обстоятельство привело к выделению цифровых устройств управления в особый класс вычислительных систем – систем реального времени.

1.7 Форматы чисел. Погрешность цифрового представления аналоговой величины

В цифровой технике все обрабатываемые переменные представляются в специальном цифровом виде. На заре вычислительной техники разрабатывались машины с различным представлением чисел: десятичным, двоично-десятичным, троичным и двоичным. В настоящее время применяется только двоичное представление переменных, как технически наиболее экономичное. Основная характеристика такого представления - количество разрядов двоичного представления величины (разрядность). В

настоящее время применяются в основном одно, двух, трех и четырех байтовое представления величин.

Для численных переменных используются цифровые форматы, т.е. правила записи чисел в пределах выделенной разрядности. Правил записи чисел может быть предложено очень большое количество. Однако опыт создания ЭВМ позволяет выделить наиболее часто используемые и удобные форматы. В настоящее время в практике разработки новых ЭВМ в основном применяются форматы чисел, определяемые международным стандартом **IEEE754/854.**

Для полного понимания основных проблем, связанных с представлением величин в ЭВМ, рассмотрим три базовых формата записи чисел.

1.7.1 Формат целых чисел

В формате целых чисел переменная представляется коэффициентами разложения величины по положительным степеням числа 2, т.е. переменная *M*, количество единиц в которой равно

$$M = C_{n-1}2^{n-1} + C_{n-2}2^{n-2} + \ldots + C_1 2^1 + C_0 2^0,$$

представляется в машине двоичным кодом

$$K = C_{n-1}C_{n-2}\ldots C_1 C_0,$$

где n - разрядность представления числа, C - коэффициенты разложения, равные 0 или 1.

При необходимости записи чисел со знаком, старший разряд обычно отводят под знаковый, а величину числа задают в дополнительном коде, [23]. Для кодирования знака числа также используют двоичное представление – обычно знак «+» кодируется 0, знак «-» кодируется 1.

Например, двоичный код

$$K = 10110100$$

в формате целых чисел без знака представляет число

$$M_ц = 1*2^7 + 0*2^6 + 1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 0*2^0 = (180)_{10}.$$

В формате целых чисел со знаком, используя правило перевода двоичных чисел из дополнительного в прямой код, можем записать:

$$M_ц = -(0*2^7 + 1*2^6 + 0*2^5 + 0*2^4 + 1*2^3 + 1*2^2 + 0*2^1 + 0*2^0) = -(76)_{10}.$$

Правило перевода двоичных чисел из дополнительного в прямой код весьма

простое - разряды исходного кода инвертируются и к результату прибавляется 1.

Основная особенность двоичного представления числа состоит в том, что в машине число представляется фиксированным числом разрядов, которое определяется физической разрядностью шин передачи данных. Так, в рассмотренном выше примере использовано однобайтовое представление числа. Это означает, что переменные в машине с восьмиразрядным представлением чисел могут изменяться в пределах от 0 до (2^8-1) для случая представления чисел без знака, или от -2^7 до (2^7-1) в случае представления чисел с знаком. Общее количество чисел, с которым будет оперировать машина, определяется количеством сочетаний нулей и единиц в записи числа, что в рассматриваемом примере составит величину 2^8 или 256 различных чисел.

При выполнении любых преобразований изменение величины переменной в формате целых чисел не может быть меньше 1 младшего разряда. Если мы будем иметь дело с некоторой гипотетической величиной *mX, m* - масштаб, изменяющейся непрерывно в диапазоне [-128,127] и имеющей на этом интервале бесконечное число значений, то при ее численном представлении в формате целых восьмиразрядных чисел мы сможем записать только 256 ее различных значений. Такой процесс преобразования непрерывной величины в дискретные значения называют процессом **квантованием уровня.** Обычно в технических устройствах осуществляется равномерное по диапазону квантование уровня сигнала.

Соответствующая цифровая величина может быть представлена как операция взятия целой части переменной *X*:

$$X_N = int(mX),$$

где: X_N - машинное целочисленное представление величины *X, int(.)* - операция выделения целой части числа.

В результате использования операции *int(.)* возникает погрешность представления величины:

$$\varepsilon = | mX - X_N |.$$

Величина этой абсолютной погрешности, в силу определения операции выделения целой части, находится в диапазоне (0, 1) и не зависит от величины переменной *mX*. Иногда в литературе величину погрешности **ε** смещают на 0,5 единицы младшего разряда (е.м.р.) и говорят о симметричной абсолютной погрешности представления переменной в формате целых чисел, равной ± 0,5 е.м.р.

Интересно изменение величины относительной погрешности представления переменной в формате целых чисел, определяемой как

$$\varepsilon_{отн} = \frac{\varepsilon}{mX} * 100\% . \qquad (1.7.1)$$

Непосредственно из определения относительной погрешности следует, что наибольшая погрешность целочисленного представления получается при преобразованиях близких к нулю значений переменой. Величина этой погрешности может составлять десятки процентов. Так как величина погрешности по ходу вычислений имеет тенденцию к накоплению, то вычислительный алгоритм желательно строить так, чтобы в ходе вычислений получались по возможности большие по абсолютной величине значения чисел.

В настоящее время формат целых чисел является основным форматом представления переменных в микроконтроллерах и промышленных компьютерах цифровых систем управления.

1.7.2 Формат чисел с фиксированной запятой

Этот формат представления переменных был очень популярным в цифровой технике 80-х годов прошлого столетия.

В формате чисел с фиксированной запятой переменная представляется коэффициентами разложения величины по отрицательным степеням числа 2, т.е. переменная *M*, количество единиц в которой равно

$$M = C_{n-1}2^{-1} + C_{n-2}2^{-2} + \ldots + C_1 2^{n-2} + C_0 2^{n-1},$$

представляется в машине двоичным кодом

$$K = C_{n-1}C_{n-2}\ldots C_1 C_0,$$

где n - разрядность представления числа, C - коэффициенты разложения, равные 0 или 1.

При необходимости записи чисел со знаком, старший разряд является знаковым, а величину числа задают в дополнительном коде. Для кодирования знака числа также используют двоичное представление - знак «+» кодируется 0, знак «-» кодируется 1.

Непосредственно из определения формата следует, что величина переменной *M* должна быть нормированной на единицу, т.е.

$$| M | < 1 .$$

Так, например, двоичный код K = 10110100 в формате чисел с фиксированной запятой без знака представляет число

$$M_{фз} = 1*2^{-1} + 0*2^{-2} + 1*2^{-3} + 1*2^{-4} + 0*2^{-5} + 1*2^{-6} + 0*2^{-7} + 0*2^{-8}$$

$= (0,5 + 0,125 + 0,0625 + 0,015625)_{10} = (0,703125)_{10}$.

Основное очарование этого формата заключено в разрядности десятичного эквивалента представления числа. Восьмибитовое двоичное число в десятичном эквиваленте представлено 6 разрядами. Возникает иллюзия высокой точности вычислений. При внимательном рассмотрении свойств этого формата оказывается, что общее количество чисел, используемое в формате, составляет 2^n, что совпадает с форматом целых чисел. В рассмотренном примере число сочетаний 0 и 1 в коде числа K составляет 2^8 (256). Значение $M_{фз}$ получается из значения $M_ц$ делением на 256, т.е. на максимальное значение переменной для заданного числа разрядов.

Другой аргумент в пользу использования формата чисел с фиксированной запятой связан с операциями цифровой обработки информации. Так часто приходится выполнять операции умножения переменной X на коэффициент, значение которого по модулю меньше 1, например, $(0,703125)_{10}$. В такой задаче применение формата целых чисел кажется невозможным. Однако, учитывая, что число

$$(0,703125)_{10} = 180/256 \text{ или } 45/64,$$

для вычисления указанного произведения можем выполнить целочисленное умножения X на 45 с последующим делением результата на 64.

Что касается погрешностей представления величины X двоичным числом, то и здесь нет разницы между форматами целых чисел и чисел с фиксированной запятой. Абсолютная погрешность равна 1 е.м.р., что с учетом нормирования переменной находится в диапазоне $(0, 2^{-n})$ и не зависит от величины переменной mX. Величина симметричной абсолютной погрешности ε**,** как и в формате целых чисел, равна **± 0,5 е.м.р.**

Значение относительной погрешности представления переменных вычисляется по формуле (1.7.1). Максимальное значение погрешности получается при преобразованиях значений переменой, близких к нулю. Величина этой погрешности может составлять десятки процентов. Как и в случае использования формата целых чисел, вычислительный алгоритм желательно строить так, чтобы в ходе вычислений получались по возможности большие по абсолютной величине значения чисел.

Таким образом, формат чисел с фиксированной запятой не имеет существенных преимуществ перед форматом целых чисел, как по количеству используемых чисел, так и по точности представления результата. Для программистов этот формат является неудобным, требующим достаточно громоздких записей десятичных эквивалентов используемых чисел. В связи с этим, формат чисел с фиксированной запятой в современных технических средствах управления и обработки информации используется крайне редко.

1.7.3 Формат чисел с плавающей запятой

Иные возможности предоставляет программистам формат записи чисел с плавающей запятой. В этом формате переменная разделяется на две компоненты - мантиссу и порядок числа. Мантисса представляется коэффициентами разложения величины по отрицательным степеням числа 2, т.е. в формате чисел с фиксированной запятой. Мантисса нормирована к единице. Порядок числа представляется коэффициентами разложения величины по положительным степеням числа 2, т.е. в формате целых чисел. Переменная *M*, количество единиц в которой равно

$$M = M_{фз} * 2^{M_ц},$$

где:

$$M_{фз} = C_{k-1}2^{-1} + C_{k-2}2^{-2} + \ldots + C_1 2^{k-2} + C_0 2^{k-1},$$
$$M_ц = P_{m-1}2^{m-1} + C_{m-2}2^{m-2} + \ldots + C_1 2^1 + C_0 2^0,$$

представляется в машине двоичным кодом

$$K = C_{k-1}C_{k-2} \ldots C_1 C_0 P_{m-1} P_{m-2} \ldots P_1 P_0,$$

где $n = (k + m)$ - разрядность представления числа, C - коэффициенты разложения мантиссы, P - коэффициенты разложения порядка числа, равные 0 или 1. При практическом использовании формата выделяют еще по одному биту для записи кодов знаков для мантиссы и порядка.

Непосредственно из определения формата следует, что величина переменной *M* должна быть нормированной на единицу, т.е.

$$| M_{фз} | < 1.$$

Величина мантиссы числа умножается на порядок, что в результате дает число, которое по абсолютной величине может быть значительно больше единицы.

Рассмотрим, например, однобайтовое число без знаков мантиссы и порядка, представленное двоичным кодом $K = 10110100$.

Предположим, что при кодировании принята разрядность мантиссы $k = 6$ бит, и разрядность порядка $m = 2$ бита. Тогда приведенному коду *K* в десятичной системе счисления соответствует число

$$M_{пз} = (0{,}703125)_{10} * 2^0 = (0{,}703125)_{10}$$

Если изменить код числа К, увеличив величину порядка на единицу, т.е.

$$K = 10110101,$$

то десятичный эквивалент величины числа будет равен

$$M_{пз} = (0{,}703125)_{10} * 2^1 = (1{,}40625)_{10}.$$

Дальнейшее увеличение порядка числа дает следующие значения десятичного эквивалента

$$M_{пз} = (0{,}703125)_{10} * 2^2 = (2{,}8125)_{10},$$

и

$$M_{пз} = (0{,}703125)_{10} * 2^3 = (5{,}625)_{10}.$$

Можно сказать, что порядок числа есть двоичный код позиции запятой в двоичной записи мантиссы числа.

Диапазон изменения величины числа в рассматриваемом примере составляет [0, 2^3), т.е. он занимает промежуточное место между диапазоном представления чисел в формате с фиксированной запятой [0, 1) и диапазоном представления чисел в формате целых чисел [0,2^8).

В то же время, количество разрядов числа *К* в наших примерах одинаковое и составляет один байт. Это означает, что и количество возможных чисел в диапазоне изменения у них одинаковое, равное количеству возможных комбинаций 0 и 1 кода, составляющее в нашем случае величину $2^8 = 256$. Наличие порядка в записи кода числа приводит к тому, что в формате чисел с плавающей запятой эти 256 различных значений распределяются по диапазону изменения величины неравномерно. Следовательно, погрешность представления величины числа двоичным кодом также неравномерно распределяется по диапазону изменения.

Действительно, абсолютная погрешность (симметричная) представления мантиссы числа составляет ± 0,5 е.м.р. Указанная погрешность в формате с плавающей запятой умножается на порядок числа. В результате абсолютная погрешность представления величины переменной составит

$$\varepsilon = \pm 0{,}5 \text{ е.м.р.м.} * 2^{M_ц},$$

где: ± 0,5 е.м.р.м. - погрешность представления мантиссы, $M_ц$ - целочисленный порядок числа. Погрешность представления мантиссы числа есть величина постоянная. Но после умножения на величину порядка получаем переменное итоговое значение абсолютной погрешности представления величины числа в формате с плавающей запятой. При этом с ростом абсолютной величины числа растет и величина абсолютной погрешности.

Относительная величина погрешности представления чисел в формате с плавающей запятой составит:

$$\varepsilon_{отн} = \frac{\varepsilon}{mX} * 100\% = \frac{\pm 0{,}5 е.м.р.м. * 2^{M_ц}}{M_{фз} * 2^{M_ц}} * 100\%$$

или

$$\varepsilon_{отн} = \frac{\pm 0{,}5 е.м.р.м.}{M_{фз}} * 100\%. \qquad (1.7.2)$$

Из (1.7.2) следует, что относительная погрешность представления числа не зависит от величины порядка, т.е. значения относительной погрешности одинаковы как для малых, так и для больших по абсолютной величине чисел. Особенно заметно это свойство проявляется при использовании многобайтных чисел.

Формат чисел с плавающей запятой удобен для программистов тем, что позволяет не особенно тщательно контролировать изменение погрешности результата вычислений по ходу выполнения алгоритма программы. Разработка программы упрощается и требует меньше времени на разработку. В то же время, реализация процессоров, работающих в формате с плавающей запятой технически значительно сложнее, чем при использовании других рассмотренных форматов. Так, реализация простейшей операции сложения двух чисел требует предварительного приведения операндов к единому формату, т.е. требует выполнения лишних операций и в итоге снижает быстродействие устройства.

Успехи современной электроники в плане существенного увеличения плотности размещения на кристалле и увеличения быстродействия используемых элементов сделало актуальным создание вычислительных машин с аппаратной реализацией арифметики в формате чисел с плавающей запятой. В настоящее время на рынке появились цифровые сигнальные процессоры, реализующие указанные операции. Примером таких устройств являются сигнальные процессоры семейства TigerSHARC® фирмы Analog Devices.

1.8 Простейший регистровый процессор. Особенности математической модели

На рисунке 1.8.1 показана схема простого цифрового устройства. Она отличается наличием неединичного коэффициента усиления *K* в цепи обратной связи. Вид импульсной характеристики (т.е. свойства вычислителя) зависит от выбора численного значения всего одного параметра – коэффициента усиления *K*. Поэтому такую вычислительную структуру можно считать простейшим регистровым процессором.

Входной сигнал *x(m)*, тактируемый импульсами ТИ, в виде *n*-разрядного параллельного кода поступает на вход сумматора SM. Тактовые импульсы ТИ поступают на вход C регистра RG. Каждый тактовый импульс разрешает запись в регистр новой информации. На выходе регистра формируется

значение выходной переменной *y(m)*, которое по цепи обратной связи образует сигнал *v(m)* и поступает на вход сумматора SM.

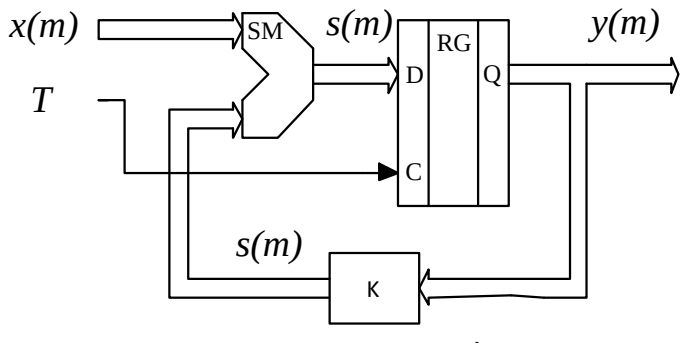

Рисунок 1.8.1 - Схема простейшего регистрового процессора

Пусть в некоторый дискретный момент времени m число на входе схемы равно x*(m)*. В этот же момент времени выходной сигнал имеет значение y*(m),* тогда сигнал на выходе сумматора будет равен

$$s(m) = x(m) + v(m),$$

где $v(m) = K*y(m).$
Тогда

$$s(m) = x(m) + K*y(m). \qquad (1.8.1)$$

При поступлении следующего тактового импульса, т.е. в момент времени (*m+1*), выходной сигнал сумматора запишется в регистр RG, т.е.

$$y(m+1) = s(m). \qquad (1.8.2)$$

Из уравнений (1.8.1) и (1.8.2) следует, что значение выходной величины схемы изменяется во времени в соответствии с уравнением

$$y(m+1) = K*y(m) + x(m). \qquad (1.8.3)$$

Таким образом, мы получили, что ход вычислений в простейшем регистровом процессоре может быть представлен разностным уравнением, широко используемым при описании динамики дискретных систем управления, [9]. Из уравнения (1.8.3) легко получается передаточная функция рассматриваемого вычислителя. Действительно, применяя Z-преобразование к обеим частям уравнения (1.8.3) при нулевом начальном условии *y(0) = 0*, получаем

$$z*Y(z) = K*Y(z) + X(z),$$

откуда

$$W*(z) = 1/(z - K) \qquad (1.8.4)$$

где: X(z), Y(z) - Z-образы входной и выходной переменных, соответственно; W*(z) - передаточная функция рассматриваемого регистрового процессора.

Регистр RG в схеме 1.8.1 выполняет функцию элемента задержки информации на один такт. Действительно, из (1.8.2) и схемы 1.8.1 следует, что входной сигнал регистра равен *y(m+1)*, а выходной сигнал равен *y(m)*. Следовательно, работа регистра подчиняется уравнению

$$y(m+1) = y(m),$$

а его передаточная функция равна

$$W*(z) = 1/z,$$

т.е. соответствует передаточной функции идеального элемента задержки на один такт работы схемы.

Теперь, если пользоваться обычной терминологией дискретных систем управления, то можно нарисовать структурную схему рассматриваемого вычислительного устройства. Структурная схема простейшего регистрового процессора показана на рисунке 1.8.2. В схеме использованы сумматор, блок умножения на постоянный коэффициент и элемент задержки на один такт, соединенные линиями прямой и обратной связей.

Используя известные результаты теории дискретных систем, например, [9], можем утверждать, что вычислительная схема будет устойчивой (т.е. вычислительный процесс будет сходиться к некоторому установившемуся значению при постоянном входном сигнале), если единственный корень

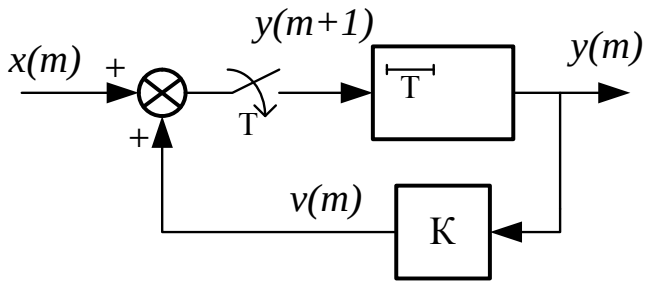

Рисунок 1.8.2 - Структурная схема простейшего регистрового процессора в линейном приближении

характеристического уравнения

$$\lambda - K = 0$$

по модулю будет меньше 1. Таким образом, условие сходимости вычислительного процесса требует, чтобы коэффициент передачи *K* блока умножения был по модулю меньше единицы.

Вид переходной характеристики устройства зависит от значения корня характеристического уравнения, т.е. в рассматриваемом случае от выбора значения коэффициента *K*. При *0<= K<1* переходная характеристика будет иметь сходящийся апериодический характер; при *-1 < K < 0* - сходящийся колебательный характер; при *K >= 1* вычислительный процесс станет неустойчивым с апериодическим изменением; при *K <= - 1* - неустойчивым колебательным. Этот вывод следует из рассмотрения общего решения разностного уравнения (1.8.3), [9]. Кроме того, для устойчивых процессов можно найти установившееся значение выходной переменной $y_{уст}$ при известном постоянном входном воздействии. Напомним, что установившееся значение определяется при $m \rightarrow \infty$ (или по передаточной функции при $z \rightarrow 1$), что в нашем случае дает

$$y_{уст} = x / (1 - K). \qquad (1.8.5)$$

Вычисленное значение установившегося значения единственное для каждого конкретного x и не зависит от начального условия *y(0)*.

Однако, структурная схема, показанная на рисунке 1.8.2, не совсем соответствует функциональной схеме, показанной на рисунке 1.8.1. Действительно, схема 1.8.1 относится к цифровой системе, а схема 1.8.2 - к импульсной. В чем же отличие? Отличие заключено в характере информационных сигналов, передаваемых по линиям связи, и выполнении операций сложения и умножения.

Рассмотрение схемы 1.8.2 показывает, что сигналы *x(m), y(m), s(m)* и *v(m)* могут принимать любые значения на конечном интервале изменения, в том смысле, что на любом конечном интервале изменения величины может быть бесконечное число ее значений. В этом смысле переменные *x(m), y(m), s(m)* и *v(m)* являются непрерывными в области значений, погрешность представления переменных равна нулю. Поэтому операции сложения и умножения с этими сигналами также выполняются с нулевой погрешностью. Те же самые переменные на схеме 1.8.1 являются двоичными сигналами, используется формат целых чисел и, следовательно, на конечном интервале изменения величины располагается конечное число ее значений. За счет этого погрешность представления величины переменной в виде числа не может быть нулевой и составляет величину ±0,5 е.м.р. (см. раздел 1.7). Операции суммирования и умножения, в связи с этим, также не могут быть выполнены с нулевой погрешностью. Еще одна особенность схемы 1.8.1 заключается в том, что значение коэффициента K не всегда может быть реализовано абсолютно точно, он тоже должен быть представлен в некотором числовом формате. Вычисленное произведение будет иметь дополнительную погрешность. Указанные отличия могут быть учтены в

схеме 1.8.2, но для этого необходимо вводить в структуру нелинейные элементы типа «квантование уровня» [21].

Влияние квантования уровней переменных проявляются в том, что возможны отличия в характеристиках системы 1.8.1 и полученной по ее импульсному эквиваленту. В частности, типовыми отличительными эффектами являются:

- появление дополнительной погрешности вычисления значений внутренних переменных;
- появление вычислительной «мертвой зоны», когда при изменении значения входной величины в некоторых пределах выходная величина не изменяется;
- неоднозначность установившегося значения, зависимость установившегося значения выходной переменной не только от значения входной, но и от начальных условий;
- возможность существования устойчивого установившегося решения в теоретически неустойчивой системе;
- возникновение устойчивых автоколебаний в теоретически устойчивой системе;
- возникновение неустойчивого решения в теоретически сходящемся вычислительном процессе.

Несмотря на указанные различия между схемой 1.8.1. и ее моделью 1.8.2, применение импульсной модели очень распространено для представления работы цифровой системы. Это объясняется наличием большого числа хорошо работающих решений, используемых в импульсной технике. Разработчикам цифровых систем оказывается легче синтезировать устройство в рамках линейного импульсного приближения и затем проверить результат синтеза методом моделирования на ЭВМ с учетом эффектов квантования уровня, чем выполнять синтез для исходно нелинейной системы. Мы также по возможности будем использовать для анализа цифровых систем их линейное приближение в виде импульсной системы.

1.9 Выбор разрядности шин передачи данных регистрового процессора

Линеаризованная модель работы регистрового процессора, рассмотренная в разделе 1.8, позволяет оценить динамические свойства проектируемого устройства, выбрать численные значения параметров модели, проверить работу устройства в специфических режимах. Однако модель не дает прямого ответа на очень важный для цифровой техники вопрос - какой должна быть разрядность шин передачи данных в разрабатываемом цифровом устройстве. Сама задача выбора разрядности шины данных неоднозначна и зависит от специфики решаемой задачи. В то же время можно дать некоторые рекомендации по ее решению.

Выбор разрядности шин данных рассмотрим на примере вычислительного устройства, линейная импульсная модель которой в виде структурной схемы показана на рисунке 1.9.1.

Схема состоит из двух последовательно включенных звеньев первого порядка на базе элементов задержки. Коэффициенты передачи К1 - К3 определяют динамические свойства устройства.

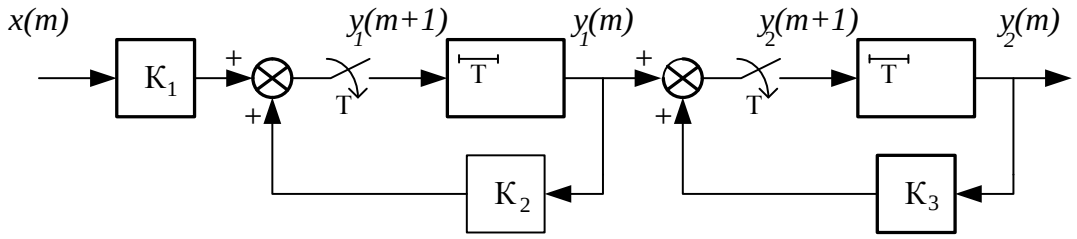

Рисунок 1.9.1 - Структурная схема вычислительного устройства

Предположим для определенности, что численные значения коэффициентов передачи известны и равны: К1 = 5, К2 = 0.95, К3 = 0.5. Тогда рассматриваемое вычислительное устройство описывается разностным уравнением второго порядка:

$$y_1(m+1) = 0{,}95\ y_1(m) + 5\ x(m),$$
$$y_2(m+1) = 0{,}5\ y_2(m) + y_1(m)\ . \qquad (1.9.1)$$

Входная переменная $x(m)$ подается на разрабатываемое устройство от внешнего, обычно известного, источника данных. Поэтому можно считать, что разрядность и формат представления данных переменной $x(m)$ известны. Предположим, что разрядность $x(m)$ равна 8 бит, формат целочисленный с старшим знаковым разрядом. В этом случае можем сразу сказать, что число различных значений $x(m)$ в устройстве равно 256, а диапазон изменения составляет $-128 \le x(m) \le 127$.

Определим разрядности шин данных для внутренних переменных схемы: $y_1(m)$ и $y_2(m)$. Для этого сформулируем наше требование к реализации шин – при максимальных возможных значениях переменных в устройстве не должно возникать переполнение разрядной сетки цифрового представления переменных, т.е.

$$|\,y_1\,|_{max} \le 2^{k-1},$$
$$|\,y_2\,|_{max} \le 2^{l-1}, \qquad (1.9.2)$$

где: k и l – разрядности шин передачи данных $y_1(m)$ и $y_2(m)$, соответственно.

Для вычисления максимальных значений переменных $y_1(m)$ и $y_2(m)$ воспользуемся уравнениями (1.9.1). Так как собственные числа уравнений составляют 0,95 и 0,5 (меньше единицы), то можем утверждать, что при постоянном входном воздействии переменные сходятся к постоянным

установившимся значениям; характер поведения переменных – апериодический. Учитывая, что для постоянного установившегося значения любой переменной выполняется условие

$$y(m+1) = y(m) = y_{уст} ,$$

уравнения (1.9.1) перепишем в виде:

$$y_{1уст} = 0{,}95\, y_{1уст} + 5\, x,$$
$$y_{2уст} = 0{,}5\, y_{2уст} + y_{1уст} ,$$

где x = const.

Из полученной системы уравнений вычисляем установившиеся значения переменных:

$$y_{1уст} = 100\, x,$$
$$y_{2уст} = 200\, x.$$

Так как в нашем примере величина $x(m)$ по абсолютной величине не может превосходить значения 128, легко вычислить и максимальные значения для переменных y_1 и y_2

$$| y_1 |_{\max} \leq 12\,800,$$
$$| y_2 |_{\max} \leq 25\,600.$$

Сравнение полученных значений максимальных значений переменных y_1 и y_2 с (1.9.2) позволяет вычислить искомые разрядности шин передачи данных при условии отсутствия переполнения разрядной сетки:

- для $y_1(m)$ разрядность k = 15 бит,
- для $y_2(m)$ разрядность l = 16 бит.

Для вычисления разрядности шины данных для передачи внутренней переменной 5* $x(m)$ достаточно знать, что

$$| x(m)|_{\max} \leq 128,$$

откуда $$| 5x(m)|_{\max} \leq 640,$$

и требуемая разрядность шины для передачи переменной составляет 11 бит.

На рисунке 1.9.2 показана схема цифрового вычислительного устройства, реализующего вычисления в соответствии с уравнениями (1.9.1). На схеме показаны разрядности шин передачи данных всех внутренних переменных. При разработке рассматриваемой схемы разрядность шин данных переменных $y_i\ (m)$ и $y_i\ (m+1)$ принята одинаковой. Это связано с тем фактом, что число бит кода на входе и выходе регистра всегда одинаковая.

Анализ схемы на рисунке 1.9.2 позволяет сделать следующие выводы:

- выполнение вычислений в регистровом процессоре требует использования шин передачи данных различной разрядности; как следствие, приходится использовать регистры и сумматоры различной разрядности, что приводит к отсутствию унификации элементной базы и увеличению аппаратных затрат;

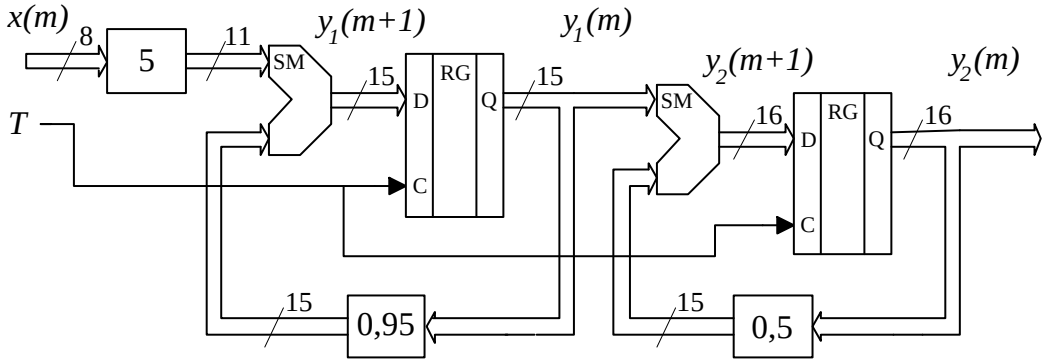

Рисунок 1.9.2 - Схема цифровой реализации регистрового вычислительного устройства

- требование сходимости вычислительного процесса приводит к вычислительным схемам, содержащим в обратных связях коэффициенты передачи, меньшие единицы; в результате по ходу разворачивания вычислительного процесса разрядность шин передачи данных возрастает;

- в рассматриваемой схеме разрядность входной переменной *x(m)* равна 8, т.е. на вход схемы может поступить только 256 различных значений входной переменной; указанному количеству входных переменных в установившемся режиме должно соответствовать 256 различных значений выходной переменной $y_2(m)$; разрядность шины переменной $y_2(m)$ у нас получилась равной 16, что соответствует 65 536 различным значениям переменной; таким образом, по количеству используемых разрядов шин передачи данных рассматриваемая схема весьма избыточна.

Подводя итог выше сказанному, можем отметить, что полученное в настоящем параграфе схемотехническое решение с точки зрения аппаратных затрат является избыточным. Следовательно, можно ставить задачу оптимизации аппаратных затрат за счет снижения избыточности схемы.

1.10 Масштабирование переменных регистрового процессора

Как было показано в разделе 1.9, при выполнении вычислений диапазон изменения внутренних переменных увеличивается “по ходу” задачи. Как следствие приходится применять многоразрядные сетки для записи значений этих переменных. В то же время существует простой прием, позволяющий уменьшить разрядность используемых шин данных. Этот прием называется масштабированием задачи.

Масштабирование задачи рассмотрим на примере, приведенном в разделе 1.9. Мы рассчитали, что максимальные значения переменных в задаче составляют значения $| y_1 |_{max} \leq 12\,800$, $| y_2 |_{max} \leq 25\,600$. Введем в рассмотрение новые переменные ν_1 и ν_2, связанные с переменными y_1 и y_2 масштабными коэффициентами k_1, k_2, т.е.

$$v_1 = k_1 * y_1,$$
$$v_2 = k_2 * y_2. \qquad (1.10.1)$$

Далее, из прикладных особенностей решаемой задачи, либо на основе своего прошлого опыта, необходимо задать максимально возможные значения этих новых переменных. Например, из соображений допустимой погрешности положим $| \nu_1 |_{max} \leq 3\,200$, $| \nu_2 |_{max} \leq 2\,560$. Тогда, подставляя максимальные значения переменных в (1.10.1) можем вычислить масштабные коэффициенты новых переменных:

$$k_1 = \frac{|v_1|_{max}}{|y_1|_{max}}, \qquad k_1 = 0{,}25,$$

$$k_2 = \frac{|v_2|_{max}}{|y_2|_{max}}, \qquad k_2 = 0{,}1.$$

За счет применения масштабов для новых переменных можем использовать 13-разрядную шину для передачи v_1, и 13-разрядную шину для v_2 (напомним, что 12-разрядная шина позволяет передавать числа из диапазона $\pm 2^{12} - 1 = \pm 2047$; 13-разрядная шина позволяет передавать числа из диапазона $\pm 2^{13} - 1 = \pm 4095$).

Для приведения исходной системы уравнений к новым переменным, подставим в уравнения (1.9.1) значения переменных y_1 и y_2 из (1.10.1); получим:

$$v_1(m+1) / k_1 = 0{,}95\ v_1(m) / k_1 + 5\ x(m),$$
$$v_2(m+1) / k_2 = 0{,}5\ v_2(m) / k_2 + v_1(m) / k_1,$$

или, после преобразования, в виде

$$v_1(m+1) = 0{,}95\ v_1(m) + k_1 * 5\ x(m),$$
$$v_2(m+1) = 0{,}5\ v_2(m) + k_2 * v_1(m) / k_1. \qquad (1.10.2)$$

Из (1.10.2) можно вывести простое правило: для перехода к новым переменным коэффициенты передачи блоков умножения в исходной структурной схеме должны быть умножены на отношение масштаба выходной переменной к масштабу входной, т.е.

$$K_i^{нов} = K_i * k_{вых}/k_{вх}.$$

Применение данного правила иллюстрирует рисунок 1.10.1. На рисунке показана структурная схема устройства, работающего с новыми переменными. Элемент задержки на такт является элементом хранения информации и не меняет масштаб переменной, т.е. переменные на его входе и выходе всегда имеют одинаковый масштаб. Сумматор также не меняет масштаба переменной; более того сумматор складывает переменные, имеющие одинаковый масштаб. Результат сложения таких переменных получается в том же масштабе, что и слагаемые. Входная переменная *x(m)* является внешней переменной для данной задачи, поэтому ее масштаб можно принять за единицу. В результате получается, что коэффициенты $К_2$ и $К_3$ схемы не изменяются, так как масштабы входной и выходной переменных

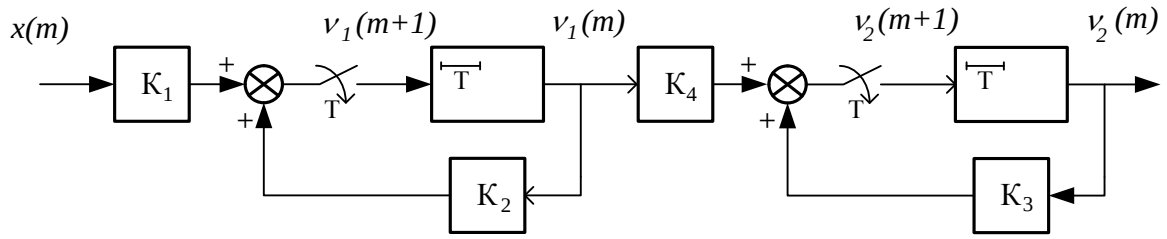

Рисунок 1.10.1 - Структурная схема вычислительного устройства с введенными масштабами для внутренних переменных

этих блоков умножения одинаковый. Коэффициент $К_1$ необходимо пересчитать, приведя масштаб исходной выходной переменной блока умножения (равен 1) к масштабу переменной $\nu_1(m)$, т.е.

$$K_1^{нов} = K_1 * k_1,\ K_1^{нов} = 1{,}25.$$

При таком значении коэффициента для передачи результата умножения на сумматор можно использовать 9ти битовую шину.

Для выполнения операции сложения во втором сумматоре масштаб переменной $\nu_1(m)$ необходимо привести к масштабу переменной $\nu_2(m)$. Для этого в схему введен коэффициент $К_4$. В исходной схеме значение этого коэффициента было равно единице. Поэтому новое значение $К_4$ рассчитываем в соответствии с приведенным выше правилом:

$$K_4^{нов} = K_4 * k_2/k_1,\ K_4^{нов} = 0{,}4.$$

На рисунке 1.10.2 показана функциональная схема модифицированного вычислительного устройства, реализующего уравнения (1. 0.2). На схеме *x(m)* обозначает величину внешней входной переменной в момент времени $t = m*T$; T – период квантования времени в устройстве. В данной схеме, в

отличие от схемы рис. 1.9.2, введен дополнительный блок умножения для реализации коэффициента K_4. При этом разрядности внутренних шин передачи данных и разрядности используемых регистров снижены до 13 разрядов. Таким образом введение масштабирования переменных позволяют снизить аппаратные затраты на реализацию устройства.

Повторно отметим, что снижение разрядности шин передачи данных увеличивает ошибку вычисления внутренних переменных. Так, в рассматриваемом случае, ошибка представления переменных $\nu_1(m)$ и $\nu_2(m)$ составляет ± 0,5 е.м.р. Пересчет к исходным переменным (с учетом примененных масштабов) показывает, что ошибка вычисления в модифицированной схеме для $y_1(m)$ составляет 4 е.м.р., а для $y_2(m)$ - 10 е.м.р.

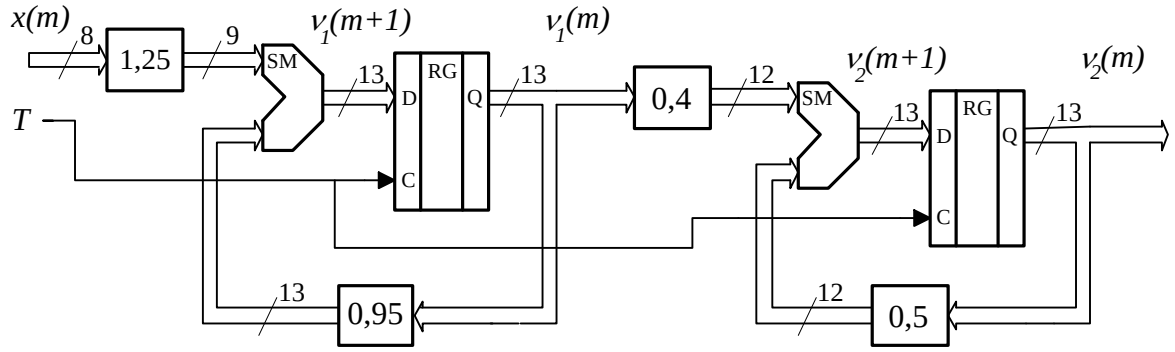

Рисунок 1.10.2 - Функциональная схема модифицированного вычислительного устройства

Таким образом сокращение аппаратных затрат при реализации регистрового способа вычислений возможно за счет увеличения погрешностей расчета внутренних переменных.

Рассмотренный подход к выбору масштабирующих коэффициентов базируется на задании диапазонов изменения внутренних переменных устройства, расчета разрядности по максимальным значениям переменны. Данных подход имеет существенный недостаток – при выборе масштабов не учитывается требуемая погрешность вычислений. В результате можно разработать устройство, вычислительная точность которого будет значительно ниже требуемой. Поэтому при расчете масштабов внутренних переменных применяются и другие подходы, (см. [5]).

1.11 Аппаратная реализация устройств обработки информации

1.11.1 Цифровой фильтр для сглаживания ступенчатых воздействии

На рисунке 1.11.1 показана схема устройства для сглаживания ступенчатых сигналов. Рассматриваемый фильтр подробно описан в [24]. Устройство предназначено для сглаживания бросков управляющих сигналов в электромеханических системах управления. Применение данного

устройства позволяет исключить возникновение ударов в механизмах при резком изменении сигнала управления.

Устройство состоит из цепочки последовательно включенных параллельных регистров RG. Информация в регистры записывается при подачи тактового импульса *T*. Входной сигнал *X*, представленный параллельным 8-разрядным двоичным кодом, подается на вход первого регистра. При поступлении очередного тактового импульса информация в цепочке регистров обновляется: данные *X5* переписываются в *X6*; данные *X4* переписываются в *X5*; данные *X3* переписываются в *X4* и т.д. Новые данные *X* записываются в *X1*. Указанное обновление информации в регистрах может быть описано системой разностных уравнений:

$$X6(n+1) = X5(n)$$
$$X5(n+1) = X4(n)$$
$$X4(n+1) = X3(n)$$
$$X3(n+1) = X2(n)$$
$$X2(n+1) = X1(n)$$
$$X1(n+1) = X(n)$$

где: *n* - момент времени поступления настоящего, *n+1* - следующего тактового импульса.

Выходной сигнал *Y* схемы формируется суммированием значений *X1* - *X6* и последующим делением на 6, т.е.

$$Y(n) = [X1(n)+X2(n)+X3(n)+X4(n)+X5(n)+X6(n)] / 6 . \qquad (1.10.1)$$

Операция суммирования выполняется схемами сумматоров SM, а операция деления - специальным устройством деления на 6.

На схеме устройства (см. рисунок 1.11.1) для примера принято, что входная переменная *X* представлена одним байтом. Это означает, что переменная *X* может принимать не более 256 различных значений (т.е. 2^8). При последовательной перезаписи значения входной переменной из регистра в регистр количество возможных значений переменной не увеличивается, поэтому все регистры выбраны с одинаковой разрядностью - восьмибитовыми. Сумматоры в схеме имеют различную разрядность.
Действительно, при сложении двух однобайтовых чисел количество различных значений выходной переменной составит 512, т.е. выходная переменная должна иметь 9 двоичных разрядов. Формально этот вывод получается из следующей очевидной цепочки равенств для представления количества различных значений переменных:

$$2^8 + 2^8 = 2*2^8 = 2^9.$$

Сложение двух девяти битовых чисел дает 1024 различных значений переменной и требует десяти двоичных разрядов для представления результата. В последнем сумматоре к десяти битовому числу прибавляется девяти битовое; результат может иметь 1024+512 = 1536 различных значений. Для безошибочной записи такого количества различных чисел требуется 11 двоичных разрядов. Таким образом для реализации операции

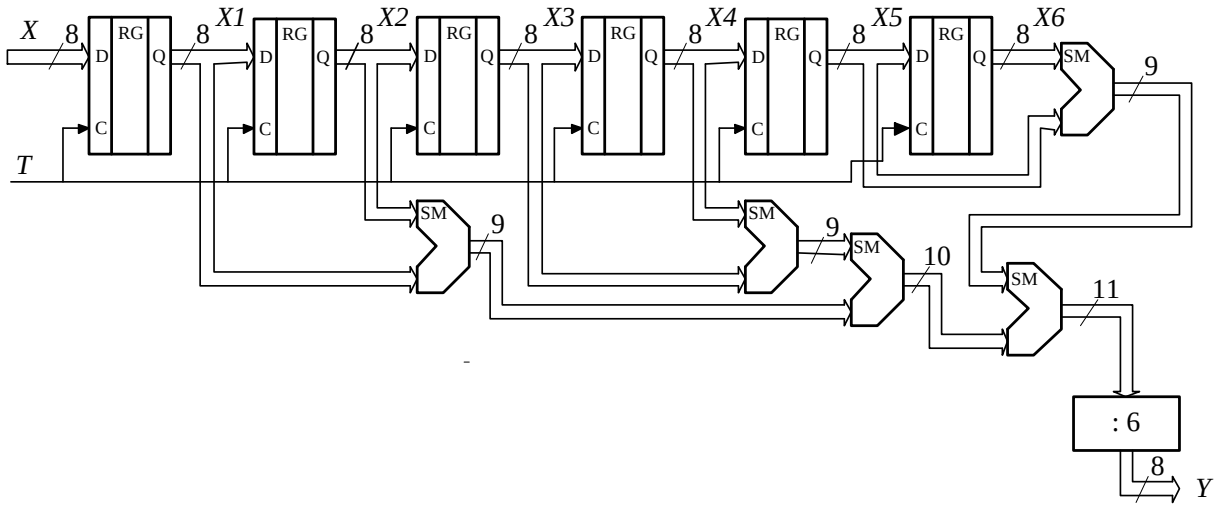

Рисунок 1.11.1 - Устройство для сглаживания ступенчатых воздействий

суммирования необходимо использовать 3 однобайтовых сумматора (считаем по разрядности входных переменных), один девятиразрядный и один десятиразрядный сумматоры.

Выходная переменная вычисляется путем деления полученной суммы на 6.Так как при делении количества возможных значений суммы на 6 получаем

$$1536 / 6 = 256,$$

то разрядность величины *Y* составит восемь бит.

Тот факт, что рассматриваемое устройство сглаживает ступенчатые воздействия, легко показать на примере. На рисунке 1.11.2 показана временная диаграмма работы устройства при определенном входном сигнале *X*. Для удобства моменты времени скачкообразного изменения воздействия *X* обозначены начальными буквами латинского алфавита.

Предположим, что в начальный момент времени в регистрах устройства записаны нули, т.е. $X1=X2=X3=X4=X5=X6 = 0$. Величины изменения воздействия *X* в моменты времени a, b и c составляют по 60 единиц, но имеют разные знаки. Выходная величина *Y* при каждом изменении *X* изменяется в течении 6 тактов, величина изменения составляет 10 единиц. Для подтверждения данного вывода выполним расчет изменений содержимого регистров и выходной переменной для скачка "a" по тактам работы схемы. Указанный расчет сведен в таблицу 1.11.1.

Аналогично схема отрабатывает и другие скачки входного воздействия. Следует отметить, что скачек "d" отрабатывается с изменением *Y* на

величину 5 единиц за такт. При этом скачек "e" происходит раньше, чем схема закончит отработку скачка "d". Расчет изменений содержимого регистров и выходной переменной по тактам приведен в таблице 1.11.2.

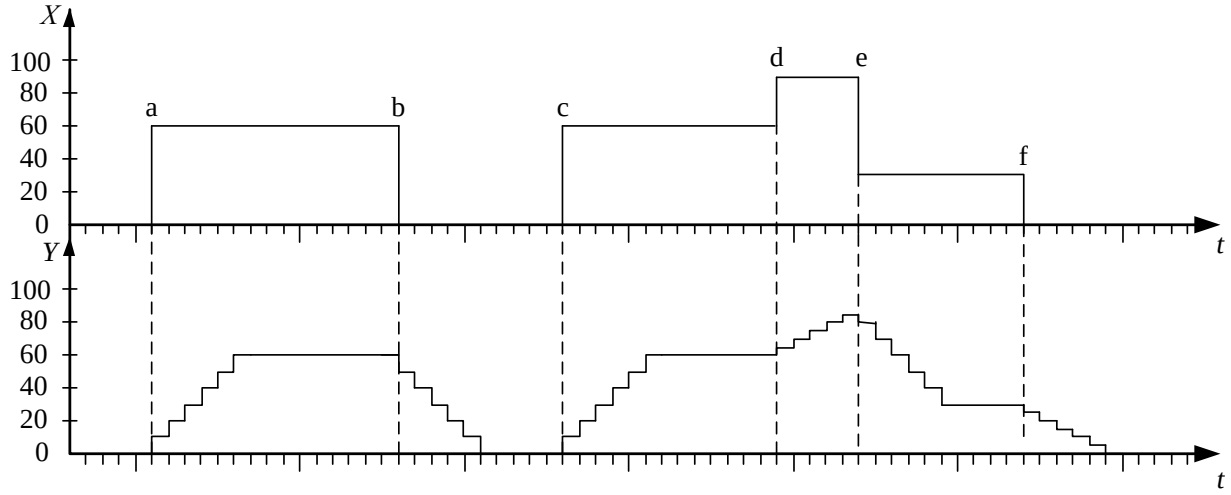

Рисунок 1.11.2 - Временная диаграмма работы устройства

Таблица 1.11.1 Расчет результатов работы схемы для одного скачка входной переменной

Номер Такта	Вход *X*	Содержимое регистров						Сумма	Выход *Y*
0	60	0	0	0	0	0	0	0	0
1	60	60	0	0	0	0	0	60	10
2	60	60	60	0	0	0	0	120	20
3	60	60	60	60	0	0	0	180	30
4	60	60	60	60	60	0	0	240	40
4	60	60	60	60	60	60	0	300	50
6	60	60	60	60	60	60	60	360	60
7	60	60	60	60	60	60	60	360	60

Из приведенных таблиц с результатами вычислений состояний регистров по тактам работы схемы следует, что изменение величины *Y* за один такт может быть вычислено делением величины изменения *X* на 6.

Из уравнения (1.11.1) следует, что ступенчатого воздействия в устройстве осуществляется за счет вычисления среднего значения входной переменной по 6 отсчетам, хранимым в регистрах. С течением времени данные в регистрах обновляются и вычисленное среднее значение изменяется. Такой прием сглаживания данных называется **методом скользящего среднего.** В нашем случае устройство сохраняет только 6 последних значений входной переменной, поэтому говорят "устройство помнит данные на 6 тактах". Временной интервал в 6 тактов называют **размером временного окна** устройства. На практике размер временного окна в зависимости от задачи может иметь самые различные значения.

Таблица 1.11.2 Расчет результатов работы схемы для двух скачков входной переменной

Номер Такта	Вход *X*	Содержимое регистров						Сумма	Выход *Y*
0	60	60	60	60	60	60	60	360	60
1	90	90	60	60	60	60	60	390	65
2	90	90	90	60	60	60	60	420	70
3	90	90	90	90	60	60	60	450	75
4	90	90	90	90	90	60	60	480	80
4	90	90	90	90	90	90	60	510	85
6	30	30	90	90	90	90	90	480	80
7	30	30	30	90	90	90	90	420	70
8	30	30	30	30	90	90	90	360	60
9	30	30	30	30	30	90	90	300	50
10	30	30	30	30	30	30	90	240	40
11	30	30	30	30	30	30	30	180	30
12	30	30	30	30	30	30	30	180	30

1.11.2 Цифровой фильтр для усреднения данных от нескольких датчиков

Рассмотрим задачу обработки измерительных данных, когда входная информация поступает от нескольких одинаковых датчиков. Например, измеряется температура в нескольких местах внутри термокамеры. В результате обработки требуется определить температуру в камере.

Указанную задачу можно решить классическим методом - использовать необходимое число сумматоров для вычисления суммы всех имеющихся данных и далее использовать устройство деления для вычисления среднего значения. Однако возможны и другие решения, одно из которых описано в [24]. Рассмотрим это решение.

На рисунке 1.11.3 показана схема устройства для измерения температуры в термокамере. В разных местах термокамеры установлены пять аналоговых датчиков температуры. Выходные сигналы датчиков оцифровываются аналого-цифровыми преобразователями (АЦП) и записываются в регистры (RG). Для примера, в устройстве использованы восьмиразрядные АЦП и, соответственно, восьмиразрядные регистры.

Работа устройства во времени определяется работой двоичного восьмиразрядного счетчика импульсов CT2. На счетный вход счетчика CT2 поступают импульсы от генератора импульсов G. Число в счетчике CT2 Постоянно увеличивается, пока не достигает максимального значения,

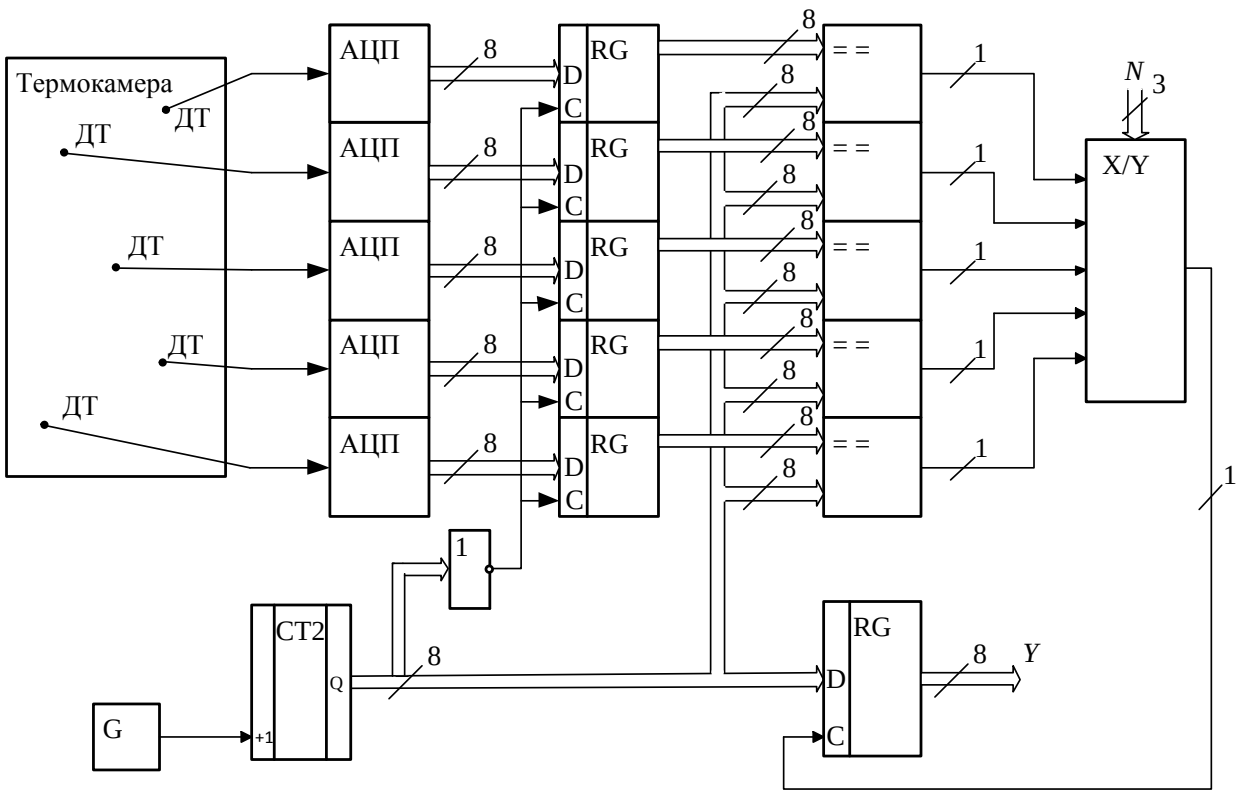

Рисунок 1.11.3 - Устройство для обработки сигналов датчиков температуры

равного 2^8 - 1 = 255. При поступлении следующего импульса счетчик переполняется, число в счетчике становится равным 0, и счет импульсов начинается с начала. При числе в счетчике равном 0 срабатывает элемент ИЛИ-НЕ, который формирует на выходе строб разрешения записи информации в регистры. Таким образом, данные в регистрах обновляются в момент времени, когда число в счетчике равно 0.

Содержимое пяти регистров сравнивается пятью цифровыми компараторами с числом в счетчике. Если число в счетчике становится больше либо равным числу в регистре, то на выходе соответствующего компаратора формируется логическая 1, иначе - логический 0.

Специальная схема порогового элемента X/Y срабатывает, если сумма единиц с выходов компараторов больше либо равна числу N. Число N - это трехразрядное число, которое задает пользователь из диапазона 1-5. При срабатывании порогового элемента формируется строб, разрешающий запись числа из счетчика CT2 в выходной регистр и, соответственно, модификацию данных на выходе Y.

Если пользователь задаст на входе порогового элемента число N=1, то пороговый элемент сработает, как только число в счетчике достигнет минимального из чисел в регистрах. При этом сработает по крайней мере один из компараторов и в выходной регистр схемы будет записан код минимальной температуры в камере. Аналогично, если пользователь задаст число N = 5, то в выходной регистр будет записан код максимальной

температуры в термокамере. Можно ожидать, что при N=3 в выходной регистр будет записан код средней температуры в камере.

Таким образом, аппаратным способом реализован гибкий алгоритм обработки информации от датчиков температуры. Схема позволяет различить 255 различных значений температуры, период квантования информации по времени определяется частотой f_g следования импульсов генератора G и равен $T_{кв} = 256 / f_g$.

1.11.3 Матричная организация процессора

Рекуррентные алгоритмы управления и обработки информации представимы в матричном виде или в виде системы разностных управлений первого порядка. Так, например, линейную систему второго порядка в матричной форме записи

$$X(n+1)=A\cdot X(n)+K\cdot\varepsilon(n);$$
$$u(n)=C\cdot X(n)+d\cdot\varepsilon(n)$$

можно легко привести к эквивалентной системе уравнений

$$x_1(n+1)=a_{11}x_1(n)+a_{12}x_2(n)+k_1\varepsilon(n)$$
$$x_2(n+1)=a_{21}x_1(n)+a_{22}x_2(n)+k_2\varepsilon(n)$$
$$u(n)=c_1x_1(n)+c_2x_2(n)+d_1\,\varepsilon(n)$$

где: $\varepsilon(n)$ - входная переменная; $u(n)$ - выходная переменная; $X(n)$ - вектор состояния с компонентами $x_1(n)$ и $x_2(n)$; A, K, C, d - матрицы с постоянными коэффициентами; n - дискретное время (номер такта синхронизации).

Приведенным уравнениям соответствует структура вычислительного устройства, показанная на рисунке 1.11.4. Вычисление переменной $x_1(n)$ выполняется в блоке Р1, вычисление переменной $x_2(n)$ выполняется в блоке Р2, вычисление выходной величины u(n) выполняется в блоке Р3. При этом блоки Р1 и Р2 структурно идентичные, отличие заключается в численных значениях коэффициентов a_{ij} и k_i. Эта особенность присуща всем матричным вычислительным устройствам.

Таким образом, матричный процессор строится на основе типовых процессоров (однотипных), соединенных по схеме матричной сетки.

Основные достоинства матричных вычислительных структур:

- вычислитель состоит из типовых модулей;
- различные задачи управления и обработки информации реализуются на типовых схемах и отличаются набором коэффициентов a_{ij}, k_i, c_i и d_i;
- быстродействие вычислителя соответствует обычному регистровому способу вычислений.

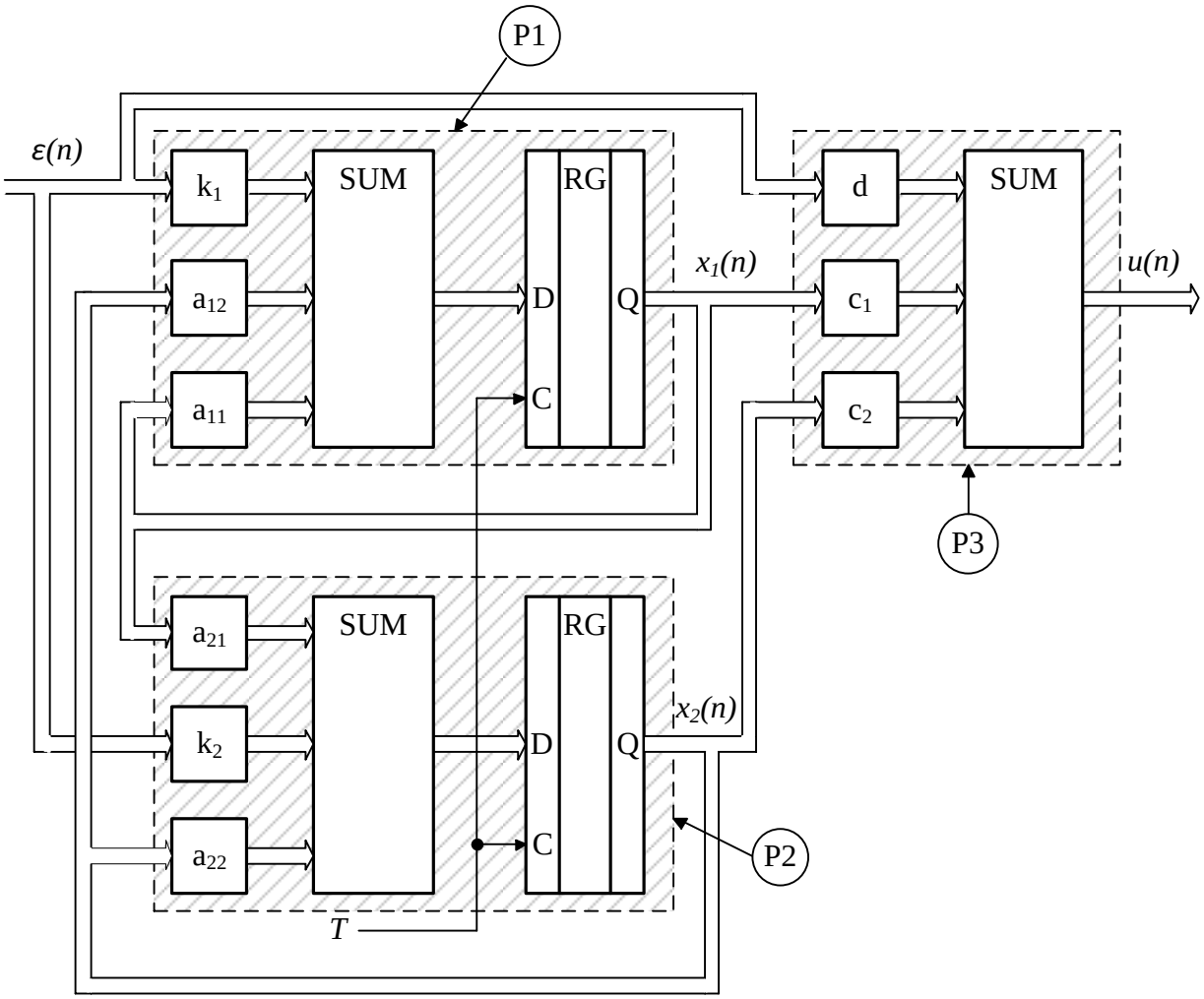

Рисунок 1.11.4 - Матричное вычислительное устройство для решения уравнения второго порядка

1.12 Применение табличных вычислений в устройствах управления

Табличный способ вычислений известен с давних времен. В задачах, где требовалось постоянно производить вычисления по одно и той же формуле, для ускорения процесса получения результата, заранее для всех возможных значений входной переменной рассчитывался результат. Полученные данные сводились в таблицу типа “значение входной переменной – значение результата “ и затем использовались по назначении. Примеры использования табличных вычислений есть у каждого, так как при решении задач по математике в средней школе широко используются таблицы тригонометрических функций.

Табличный способ вычислений можно считать **самым быстрым способом** выполнения расчетов. Это происходит из-за того, что все расчеты по существу выполняются заранее. В процессе выполнения прикладных вычислений из таблицы только выбирается нужное значение.

Однако у табличного способа есть и существенные недостатки. Главный из них – ограниченная точность вычислений. Ограничение точности вычислений происходит от того, что невозможно создать таблицу бесконечных размеров, она обязательно должна содержать ограниченное число данных. Этот недостаток не имеет существенного значения для цифровой вычислительной техники, так как для передачи данных используются информационные шины с ограниченной разрядностью, и следовательно, с ограниченным числом возможных значений переменных.

Другой недостаток табличного способа вычислений заключается в невозможности решения динамических задач, характерных для задач управления объектами. Табличный способ вычислений позволяет решать только статические задачи. Этот вывод следует непосредственно из самого принципа вычислений – в ходе вычислений в таблице ничего не изменяется, все рассчитано заранее и хранится в постоянной (неизменяемой) памяти данных. В противном случае на расчеты уйдет время и главное достоинство табличного способа – быстродействие, будет потеряно.

Несмотря на последний недостаток, табличный способ вычислений очень широко используется при реализации устройств управления. Как правило, табличный способ вычислений используется для:

- реализации типовых арифметических и логических операций - сложения, умножения, деления, возведения в степень и т.п.;
- вычисления значений нелинейных функций, входящих в закон управления – функций ограничения величины, зоны нечувствительности, многозначной релейной характеристики и других **однозначных** нелинейностей;
- вычисления значений функций времени, определяющих требуемую траекторию движения объекта (программное управление);
- построения регуляторов, реализующих логическое управление объектами.

Чаще всего с посредством табличного способа вычисляют значения некоторой однозначной функции. На рисунке 1.12.1 показана типовая функциональная схема табличного вычислителя. Она содержит всего один функциональный блок – статический преобразователь кода X/Y. На вход блока подается значение аргумента в принятом формате двоичного кода разрядности *k*, на выходе формируется значение функции также в двоичном коде, но разрядности n.

Всем обычно хорошо известна табличная реализация логических элементов. У этих элементов *k*-разрядной входной переменной обычно ставится в соответствие однобитовая выходная переменная. Работа элемента задается таблицей преобразования входной переменной *X* в выходную переменную *Y*. Такую таблицу преобразования называют таблицей истинности логического элемента.

В таблице 1.12.1 для примера приведена таблица истинности элемента “ИСКЛЮЧАЮЩЕЕ ИЛИ” для трехразрядного входного аргумента. Этот элемент может быть выполнен в виде комбинационной логической схемы, т.е. является безынерционным элементом устройства управления. Здесь, как

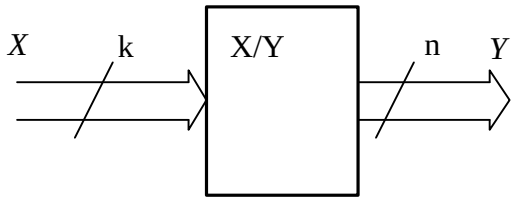

Рисунок 1.12.1 - Функциональная схема табличного вычислителя

и ранее, безинерционность понимается в системном смысле – в смысле отсутствия в реализации какой-либо оперативной памяти (регистров) для временного хранения значений промежуточных вычислений.

Область применения табличных вычислителей не ограничивается реализацией логических элементов и устройств на их основе. В принципе любая однозначная функция одного или нескольких аргументов может быть реализована на табличном процессоре. В первую очередь это устройства умножения переменной на постоянный коэффициент, которые широко

Таблица 1.12.1 Таблица истинности элемента ИСКЛЮЧАЮЩЕЕ ИЛИ

X2	*X1*	*X0*	*Y*
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

используются при выполнении регистровых вычислений. Как функцию нескольких переменных можно рассматривать параллельный сумматор, его тоже можно реализовать табличным способом. Табличный процессор удобно применять для вычисления нелинейных функций, входящих в законы управления современных управляющих систем.

Большой класс регуляторов автоматических систем представляют логические регуляторы. Эти регуляторы формируют управляющий сигнал на основе анализа выполнения определенных логических условий. Например, в бытовом холодильнике может быть использован следующий логический закон управления:

- если температура в камере холодильника выше заданной, то включить хладоагрегат;

- если температура в камере ниже или равна заданной, то выключить хладоагрегат;

- если дверь холодильника открыта, то независимо от температуры, выключить хладоагрегат и включить освещение в камере.

Такой алгоритм представим таблицей типа таблицы истинности 1.12.1 и может быть реализован на табличном процессоре.

Основная характеристика табличного вычислителя - объем таблицы, измеряемый в битах. Так, таблица 1.12.1 содержит 8 однобитовых значений выходной функции, поэтому ее объем равен 8 битам. В общем случае вычислитель, схема которого показана на рисунке 1.12.1, может быть реализован на таблице, которая должна содержать 2^k строк n - битовых значений выходной переменной. Следовательно, объем такой таблицы будет равен

$$M = 2^k * n \text{ [бит]}. \qquad (1.12.1)$$

Кроме объема таблицы, важной характеристикой устройства является его быстродействие, характеризуемое временем распространения сигнала в устройстве. Это время определяется быстродействием используемой элементной базы и должно быть существенно меньше периода квантования информации в системе.

Ограниченность объема таблицы, которую можно реально реализовать, определяет основные ограничения по применимости данного способа в системах управления. Этими ограничениями являются:

- функция y ограничена и представима конечным числом значений (ограниченность по y);

- аргумент функции ограничен или функция периодическая, что позволяет представить аргумент x конечным числом значений (ограниченность по x);

- функция $y = f(x)$ - однозначная, статическая.

Несмотря на указанные ограничения, табличный способ вычислений находит все большее применение в современной электронной технике.

1.13 Табличная реализация вычисления значения нелинейной функции

Рассмотрим особенности аппаратной реализации табличного вычислителя на примере вычисления нелинейной функции.

Предположим, что требуется вычислять значение функции $y = x^3$. Для простоты примем, что входная переменная x поступает в виде 3- битового числа в целочисленном формате с знаком. Тогда искомая функция может быть табулирована, как показано в таблице 1.13.1.

По таблице 1.13.1, используя ее как таблицу истинности, можно построить комбинационную логическую схему, содержащую 3 входных и 7

выходных однобитовых переменных. Такой подход, в настоящее время, интенсивно используется и он будет рассмотрен ниже. В данном разделе мы остановимся на другой реализации.

Правую, двоичную часть таблицы 1.13.1, можем рассматривать как некоторую память, состоящую из восьми ячеек, в каждую из которых записан свой код. Так как в процессе работы устройства содержимое памяти изменяться не будет, то информацию удобно хранить в **постоянном запоминающем устройстве** или **ПЗУ**. Современная промышленность выпускает микросхемы для реализации такой памяти. На принципиальных электрических схемах элементы ПЗУ обозначают как **ROM** от английского названия – **Read Only Memory** (память только для чтения). Входной сигнал микросхемы ПЗУ представляет собой адрес ячейки памяти, содержащей требуемое значение выходной переменной. На рисунке 1.13.1 показана функциональная схема вычислительного устройства для реализации таблицы 1.13.1 на микросхеме ПЗУ.

В схеме на рисунке 1.13.1 входная величина x определяет адрес ячейки памяти ПЗУ, в которой хранится значение вычисляемой функции. По определению адрес задается всегда целым положительным числом. Поэтому для записи в микросхему ПЗУ данные из таблицы 1.13.1 необходимо преобразовать, рассматривая x как целое число без знака. Преобразованная таблица данных для записи в ПЗУ (или, как говорят, таблица прошивки ПЗУ) будет иметь вид, приведенный в таблице 1.13.2.

Таблица 1.13.1 Табуляция функции $y = x^3$

Десятичн. значен. x	Десятичн. значен. y	Двоичный код x	Двоичный код y
0	0	000	0000000
1	1	001	0000001
2	8	010	0001000
3	27	011	0011011
-1	-1	111	1111110
-2	-8	110	1111000
-3	-27	101	1100101
-4	-64	100	1000000

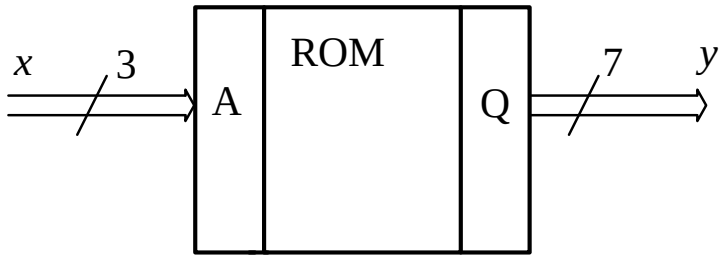

Рисунок 1.13.1 - Функциональная схема вычислителя на ПЗУ

Реализация табличного вычислителя на базе ПЗУ удобна тем, что не требует применения каких-либо специальных процедур синтеза устройства. Требуемый объем ПЗУ вычисляется по формуле (1.12.1) и для рассматриваемого примера составляет

$$M = 2^3 * 7 = 56 \text{ [бит]}.$$

Кроме объема при реализации вычислителя на ПЗУ важна **организация** микросхемы памяти. Дело в том, что объем в 56 бит можно получить по-разному: 2 ячейки по 28 бит, 4 ячейки по 14 бит или 8 ячеек по 7 бит каждая.

Таблица 1.13.2 Таблица прошивки ПЗУ для вычисления функции $y = x^3$

Адрес ячейки	Двоичный код адреса	Двоичный код *y*
0	000	0000000
1	001	0000001
2	010	0001000
3	011	0011011
4	100	1000000
5	101	1100101
6	110	1111000
7	111	1111110

В рассматриваемом примере требуется микросхема ПЗУ объемом 56 бит и организацией 8 ячеек по 7 бит каждая.

Удобство и простота аппаратной реализации табличного способа вычислений обусловили его широкое применение в современной технике цифрового управления и обработки информации. Всего два существенных ограничения имеется у табличного способа вычислений:

- с его помощью решаются только статические задачи, решить разностное уравнение не позволяет отсутствие в структуре вычислителя оперативной памяти;

- в практических задачах переменные обычно многоразрядные, что требует использовать ПЗУ очень большого объема; так, легко видеть, что добавление всего лишь одного дополнительного разряда к шине передачи входной переменной приводит к удвоению требуемого объема ПЗУ.

1.14 Кусочно-линейная аппроксимация нелинейной функции. Способ уменьшения требуемого объема ПЗУ

Рассмотрим еще один пример аппаратной реализации табличного вычисления нелинейной функции, [24].

Пусть требуется вычислять значения функции $Y=f(X)$. Предположим, что X поступает от внешнего устройства в виде 16-разрядного двоичного числа, и значение Y необходимо вычислять 16-разрядным. Обычный табличный процессор требует использования ПЗУ объемом $2^{16}\times16$ бит, т.е. требуется память объемом $M= 2^{10}\times2^{10} = 2^{10}$ Кб = 1 Мб. Схема аппаратной реализации такого табличного процессора показана на рисунке 1.14.1. Аргумент X подается на адресные входы постоянного запоминающего устройства, на выходах Q которого формируется результат Y.

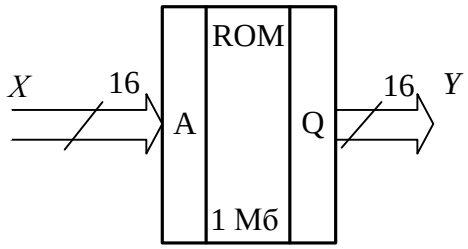

Рисунок 1.14.1 - Типовая реализация табличного процессора

Из приведенного примера видно, что при высокой разрядности операндов необходимо использовать ПЗУ достаточно большого объема, что ограничивает области применения данного способа вычислений.

Уменьшить требуемый объем ПЗУ можно за счет снижения точности вычисления функции. Для этих целей может быть использована ее кусочно - линейная аппроксимация.

При кусочно-линейной аппроксимации область изменения аргумента разделяется на участки, и значение аргумента представляется как

$$X = X_i + \Delta x,$$

где: X_i - координата начала i-го участка, или так называемая **реперная точка**; Δx - смещение значения X относительно ближайшей слева реперной точки.

Если принять гипотезу, что функция в пределах участка изменяется линейно, то значение функции Y при кусочно-линейной аппроксимации может быть вычислено по формулам

$$Y_i = f(X_i),$$
$$Y \approx Y_i + K_i * \Delta x, \qquad (1.14.1)$$

где: Y_i - значение функции в i-ой реперной точке; K_i - коэффициент наклона кривой в реперной точке, равный

$$K_i = (Y_{i+1} - Y_i) / (X_{i+1} - X_i).$$

При двоичном представлении информации код числа X удобно разбить на две части - старшая часть будет представлять реперную точку X_i, а младшая - смещение Δx. Представим 16-разрядный код X прямой суммой 12 - разрядного кода X_i и 4-разрядного кода Δx. При таком разбиении весь диапазон измерения аргумента $[0, x_{max}]$ делится на 2^{12} (4096) реперных точек, а величина Δx принимает одно из 16 возможных значений (2^4).

На рисунке 1.14.2 показана схема устройства для вычисления значения нелинейной функции с использованием кусочно-линейной аппроксимации. В рассматриваемом устройстве использовано два ПЗУ. В первом из них, объемом 2^{12} x 16 бит = 64 Кб, хранятся значения функции в реперных точках (старшие 12 бит) и значения коэффициентов K_i наклонов функции для каждой реперной точки. Во второе ПЗУ, объемом 2^8 x 8 бит = 2 Кб, записаны результаты произведений $K_i * \Delta x$, входящие в формулу (1.14.1). Искомое значение функции Y формируется на выходе сумматора, причем для обеспечения требуемой разрядности результата применен аппаратный сдвиг суммы на 3 бита в сторону старших разрядов.

Реализация рассмотренной схемы требует применения сумматора и двух ПЗУ общим объемом 66 Кбайт. Таким образом, применение кусочно – линейной аппроксимации нелинейной функции позволило более чем в 10 раз

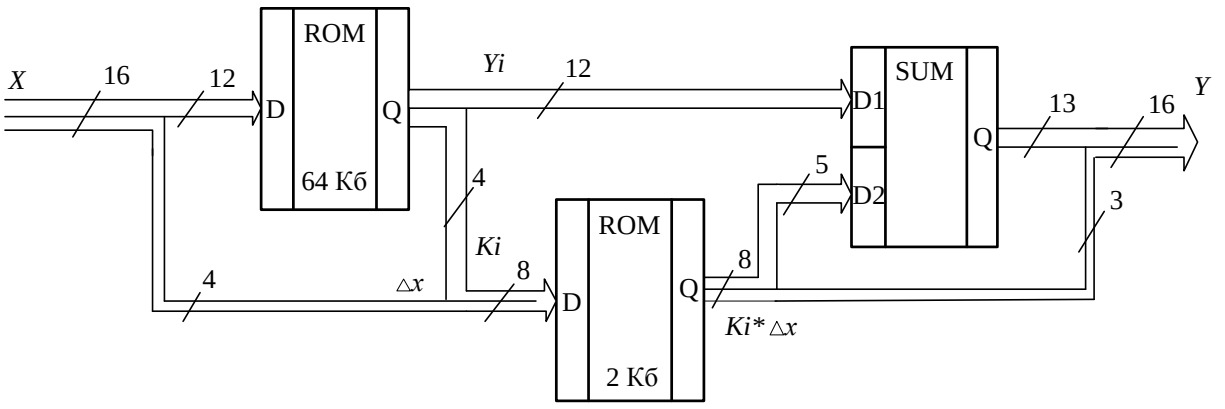

Рисунок 1.14.2 - Аппаратная реализация кусочно-линейной аппроксимации нелинейной функции

снизить объем используемых ПЗУ за счет увеличения допустимой погрешности вычислений.

1.15 Виды постоянных запоминающих устройств

К настоящему времени разработано большое число способов реализации постоянных запоминающих устройств. Из механических устройств наиболее известным является диск с отверстиями и вставленными в них колышками. При вращении диска колышки перемещали механические элементы в заданной последовательности. Такие ПЗУ применялись в музыкальных автоматах XIX века и их можно увидеть в исторических фильмах.

После постройки машины ЭНИАК постоянные запоминающие устройства стали выполнять на электронных компонентах. Были разработаны диодные, диодно – транзисторные, ферритовые, феррит-транзисторные и другие типы ПЗУ. Требование миниатюризации аппаратуры и появление микросхем сократило разнообразие физических принципов, используемых при создании элементов постоянной памяти. Начиная с 70-х годов XX века практически все ПЗУ стали электронными, т.е. стали выпускаться в виде микросхем. С этого времени разработчики ПЗУ занялись поиском таких способов записи информации, которые обеспечивают максимальную емкость устройства при минимальных габаритах.

Сама идея построения ПЗУ на базе электронных компонентов может быть реализована довольно просто. Рассмотрим для примера реализацию ПЗУ для вычисления функции $y = x^3$ из раздела 1.13.

Как уже отмечалось, таблицу 1.13.2 можно рассматривать как таблицу истинности некоторой комбинационной логической схемы, имеющей 3 входных и 7 выходных переменных. Формальный синтез такой схемы приведет к решению задачи. Однако высокая трудоемкость этого способа породила разработки других технологий.

Так, для реализации рассматриваемого ПЗУ необходимо организовать 8 ячеек памяти, каждая из которых хранит 7 бит информации. На рис. 1.15.1 показана схема устройства, реализующего именно такую структуру. Входной трехразрядный код X поступает на дешифратор X/Y, имеющий восемь выходов. Дешифратор X/Y представляет собой комбинационную схему, формирующую 1 только на одном из своих выходов, номер которого задается входным кодом X. Дешифратор имеет одну и туже архитектуру для всех ПЗУ и может быть назван дешифратором адреса ячейки памяти. Сами ячейки памяти - суть горизонтальные проводники, подключенные к выходам дешифратора. В таблице 1.15.2 показана таблица истинности рассматриваемого дешифратора.

Семь вертикальных проводников образуют линии выхода. Сигнал с этих линий через буферные формирователи (одновходовой элемент) поступает на выход устройства.

Если вертикальную линию соединить с горизонтальной линией одной из ячеек памяти, например, 2, то при поступлении на вход устройства кода 2 на выходе 2 дешифратора появится логическая 1 и эта единица через соответствующий буферный формирователь поступит на выход устройства.

Таким образом, для записи определенного числа, например, 0001000, в заданную ячейку памяти, например, 2, необходимо соединить горизонтальный проводник 2 с вертикальным проводником 3. Другими словами, двоичной единице записываемого кода соответствует соединение горизонтального проводника ячейки с вертикальным проводником соответствующего выхода. Оставшиеся вертикальные проводники, т.е. не подключенные к проводнику ячейки памяти, необходимо подключить к 0. Для этого в схему установлены резисторы R, обеспечивающие нулевой потенциал входов шинных формирователей.

Простое соединение горизонтальных и вертикальных проводников приводит к конфликту сигналов различных ячеек памяти. Действительно, каждый вертикальный проводник, имеющий соединение с несколькими горизонтальными проводниками, соединяет между собой выходы дешифратора X/Y соответствующих ячеек памяти. В результате выходные цепи дешифратора оказываются закороченными между собой и сигналы этих ячеек памяти не различаются. Разрешить эту проблему довольно просто –

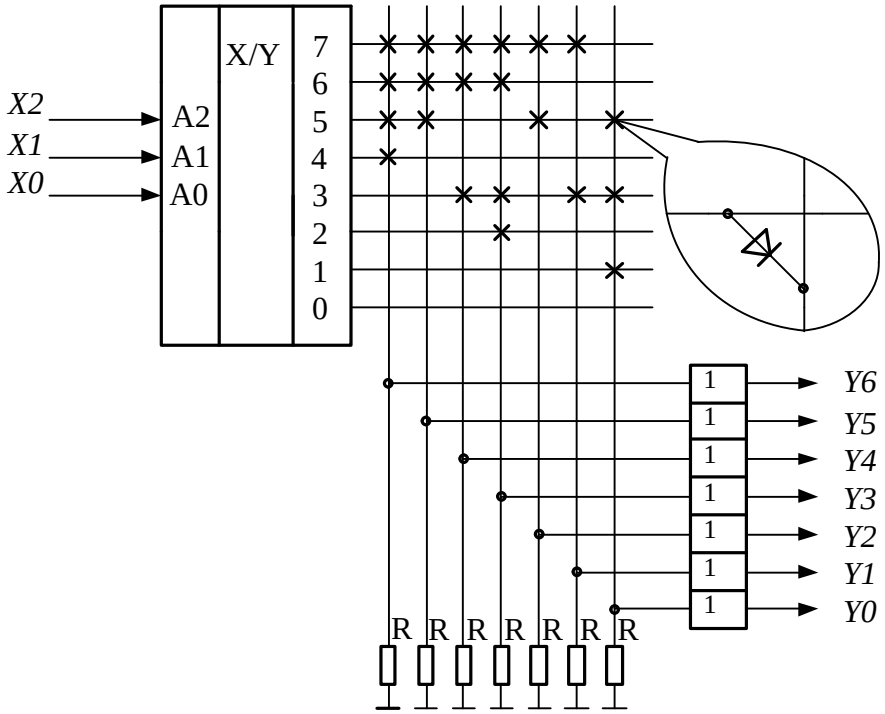

Рисунок 1.15.1 - Схема простейшего постоянного запоминающего устройства

соединение горизонтального и вертикального проводников нужно выполнять через диоды, как показано на рисунке 1.15.1. В этом случае получается, что сигналы разрядов ячеек памяти поступают на вход соответствующего шинного формирователя через диодный элемент ИЛИ.

Такая организация ПЗУ приводит к схеме устройства, содержащей типовые элементы - адресный дешифратор и шинные формирователи.

Таблица 1.15.2. Таблица истинности дешифратора адреса ячейки памяти

Двоичный код адреса	Выходной двоичный код
000	00000001
001	00000010
010	00000100
011	00001000
100	00010000
101	00100000
110	01000000
111	10000000

Различие между схемами ПЗУ состоит только в наличии или отсутствии диодов в цепях чтения сигналов ячеек памяти. Это позволяет наладить серийный выпуск микросхем, содержащих общую базовую часть и переменный слой диодов, конфигурацию которого заказывает пользователь. Выпуск таких микросхем и был налажен в 70-годах двадцатого века. Потребитель заполнял карту заказа, указывая в ней желаемую таблицу прошивки ПЗУ. На заводе - изготовителе по таблице создавали соответствующую маску для формирования слоя диодов и выполняли заказ потребителя. Этот тип ПЗУ получил название «**масочные».**

Недостаток масочных ПЗУ состоит в том, что завод-изготовитель выпускает сразу большую партию заказанных устройств (несколько десятков тысяч штук), только в этом случае цена одного устройства получается небольшой. Потребителю чаще всего такая большая партия была не нужна. Компромиссом в данной ситуации стала разработка микросхем ПЗУ, программируемых потребителем.

Первыми ПЗУ, программируемые пользователями, стали устройства с выжигаемыми элементами. Завод-изготовитель выпускал микросхему с установленными диодами во всех разрядах всех ячеек памяти микросхемы. При чтении информации из всех ячеек такого устройства во всех разрядах читались единицы. Пользователь с помощью специального прибора, называемого программатором, путем подачи на соответствующие биты ячейки памяти мощных импульсов электрического тока, физически уничтожал (выжигал) ненужные диоды. Впоследствии выжигание диодов

было заменено пережиганием специальных тонких перемычек, соединяющих диоды с проводниками микросхемы. Такие микросхемы получили название **однократно программируемые** ПЗУ**.** Действительно, после пережигания перемычки восстановить ее уже было невозможно. Емкость однократно программируемых микросхем ПЗУ достигала несколько сотен килобит.

Однократно программируемые ПЗУ применялись недолго. Пережигание перемычки или диода сопровождалось микровзрывом. Это не позволяло плотно упаковывать компоненты микросхемы, что ограничивало ее информационную емкость. Кроме того, в микросхеме накапливались дефекты, что приводило к низкой надежности работы устройства. В 1971 году пережигаемая перемычка была заменена управляемым ключом на полевом транзисторе с изолированным затвором (МОП транзистор). Теперь информация определялась наличием или отсутствием заряда в области изолированного от подложки затвора транзистора. Это были первые EPROM – электрически программируемые постоянные запоминающие устройства.

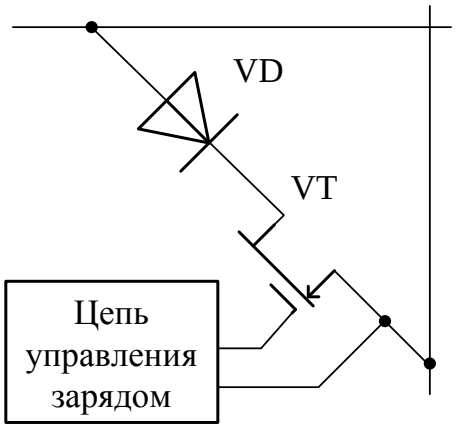

Рисунок 1.15.2 - Схема включения транзистора для управления состоянием одного бита ПЗУ

На рисунке 1.15.2 показана схема элемента хранения одного бита EPROM. Полевой транзистор VT включен последовательно диоду VD и играет роль управляемой перемычки. При отсутствии заряда в области затвора транзистор открыт и «перемычка» замкнута. При чтении информации из ячейки ПЗУ в данном разряде будет читаться единица. Цепь управления зарядом позволяет впрыснуть отрицательный заряд в область затвора транзистора. При этом транзистор закроется и «перемычка» будет «удалена». При чтении информации из ячейки ПЗУ в данном разряде будет читаться ноль.

Основная проблема создания таких ПЗУ заключается в том, что информация при отключенном питании должна сохраняться в виде электрического заряда на емкости структуры затвор-исток полевого транзистора. Существенного времени хранения информации можно добиться, если обеспечить большую величину электрического сопротивления

в цепи управления зарядом между затвором и истоком транзистора VT. Для обеспечения необходимой величины сопротивления цепи разряда в EPROM стали использовать МОП-транзисторы с так называемым плавающим затвором. Рисунок 1.15.3 иллюстрирует конструктивные различия между полевыми транзисторами, используемыми для хранения информации в EPROM.

У обычного полевого транзистора (см. рисунок 1.15.3а) между электродами истока И и стока С сформирован канал для протекания электрического тока. Затвор З изолирован от канала слоем аморфного кремния. Электрическое поле зарядов затвора влияет на величину сопротивления канала электрическому току.

На рисунке 1.15.3 б) показан МОП-транзистор с плавающим затвором. В слое изолятора между основным затвором З и каналом сформирован

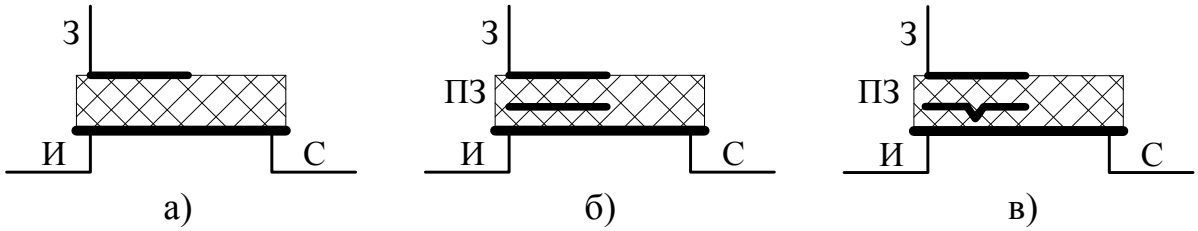

Рисунок 1.15.3 – Конструктивные отличия полевых транзисторов, используемых для хранения информации в EPROM: а) обычный МОП-транзистор; б) МОП-транзистор с плавающим затвором; в) МОП-транзистор с плавающим затвором и туннельным каналом

дополнительный изолированный проводник (обычно из поликристаллического кремния), называемый плавающим затвором ПЗ. В начальном состоянии этот проводник не имеет заряда и не влияет на сопротивление канала. При приложении к электродам затвора и стока относительно высокого напряжения (порядка +9 ÷ +12 В) относительно истока, сопротивление канала резко уменьшается, по нему протекает электрический ток и часть высокоэнергетических или «горячих» электронов из канала попадает на плавающий затвор. Этот процесс называется инжекцией горячих электронов. После снятия высокого напряжения отрицательный заряд остается на плавающем затворе, влияя на величину сопротивления канала. Заряд сохраняется на плавающем электроде более 20 лет. Таки образом производится запись и хранение бита информации в ячейках памяти.

Первые EPROM не имели цепей разряда плавающего затвора транзистора. Такие микросхемы позволяли записать только двоичные нули в любые биты памяти. Запись единиц была невозможна, поэтому перед записью новой информации в ПЗУ старую необходимо было стереть, т.е. записать во все ячейки единицы. Операция стирания информации в микросхеме выполнялась с помощью ультрафиолетового (УФ) излучения.

Микросхему выдерживали в течение 30 минут под излучением кварцевой лампы. Под действием УФ излучения электрическое сопротивление изолятора в области плавающего затвора значительно уменьшалось и заряды с затворов транзисторов стекали на подложку микросхемы. Микросхемы так и назывались – ПЗУ с **ультрафиолетовым стиранием**. Несмотря на наличие дополнительных электронных цепей управления зарядом, за счет отсутствия эффекта микровзрыва, информационная емкость микросхем с ультрафиолетовым стиранием достигла величины нескольких сот мегабит.

Параллельно с усовершенствованием ПЗУ с ультрафиолетовым стиранием шла разработка микросхем с электрической записью и стиранием информации, так называемых **EEPROM.** В этих приборах использовали полевые транзисторы с плавающим затвором и туннельным каналом. Толщина изоляции между плавающим затвором и каналом такого транзистора была существенно уменьшена (см. рисунок 1.15.3 в). При приложении отрицательного напряжения к затвору и стоку относительно истока происходил туннельный переход электронов с плавающего затвора в канал. В результате в ячейку памяти электрическим способом записывалась единица.

Такие полностью электрически программируемые ПЗУ конечно были созданы, но из-за сложности электрических цепей управления зарядом, достаточно длительное время информационная емкость EEPROM была на порядки меньше емкости ПЗУ с УФ стиранием. Это серьезно препятствовало их широкому применению в устройствах управления и обработки информации.

Наконец в 1988 году инженерам фирмы Intel за счет применения технологии блочного стирания информации удалось существенно упростить схемы цепей управления зарядом транзисторов. Получившиеся устройства могли стирать информацию в микросхеме только достаточно большими блоками, время стирания было значительным и составляло несколько миллисекунд, но по информационной емкости они смогли догнать ПЗУ с УФ стиранием. Вместо термина EEPROM микросхемы новой топологии стали называть **Flash памятью.** В дальнейшем информационная емкость Flash-памяти была существенно увеличена за счет применения технологии хранения в одной ячейке нескольких бит информации. В настоящее время эта память практически полностью заменила ПЗУ остальных типов.

Более подробную информацию по микросхемам Flash памяти можно легко найти в Интернете.

1.16 Реконфигурируемые пользователем вычислительные устройства

Исключительно высокая вычислительная эффективность аппаратной реализации специализированных цифровых устройств, а также их востребованность при решении задач обработки информации стимулировали

производство нового типа цифровых микросхем – микросхем с реконфигурируемой пользователем вычислительной архитектурой.

В конце 1970-х годов появились первые программируемые логические интегральные микросхемы (ПЛИС). Решая ту же задачу, что и разработанные ранее ПЗУ, эти микросхемы были организованы с использованием альтернативной идеологии – технологии аппаратной реализации логических уравнений. В России одной из первых ПЛИС была микросхема К556РТ1. На рисунке 1.16.1 показана архитектура логической части этой микросхемы.

Микросхема содержит три слоя логических элементов – слой элементов НЕ, слой элементов И и слой элементов ИЛИ. Эти слои связаны двумя шинами передачи данных – шиной данных слоя И (ШД И) и шиной данных слоя ИЛИ (ШД ИЛИ). Микросхема имеет 16 логических входов In0-In15 и 8

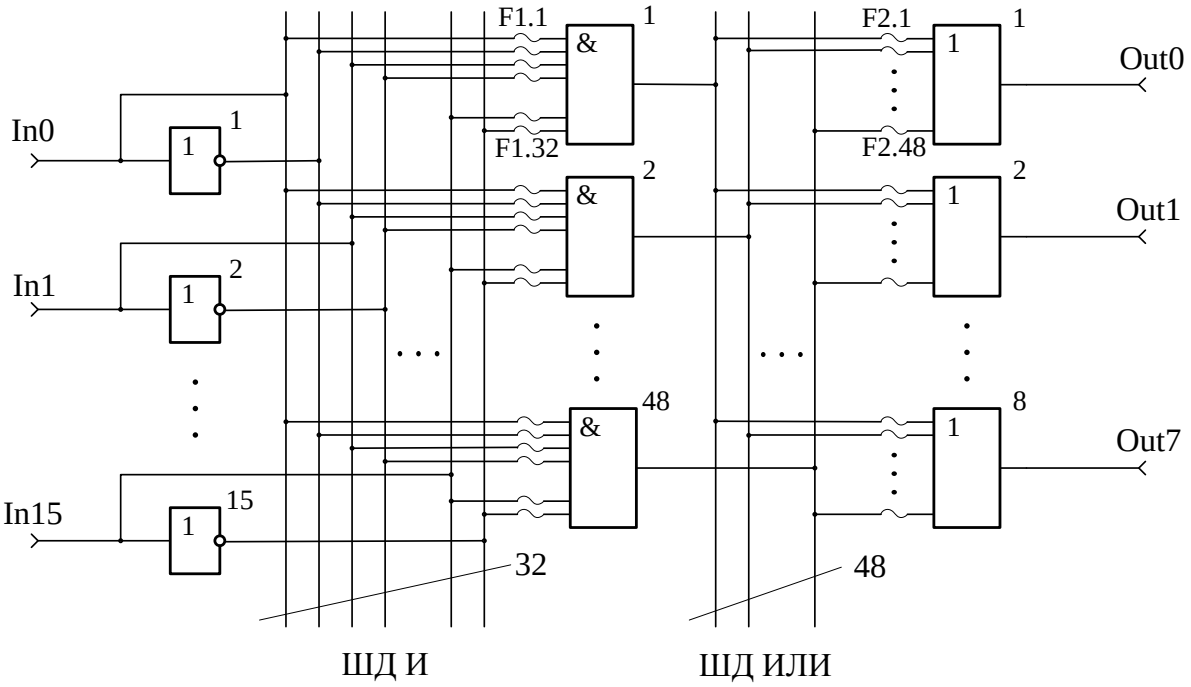

Рисунок 1.16.1 – Архитектура логической части ПЛИС К556РТ1

логических выходов Out0-Out7. Значения входных сигналов инвертируются 15 элементами слоя НЕ. Прямые и инверсные значения сигналов поступают на 32 проводника, образующих ШД И.

Слой И содержит 48 логических элементов И, на 32 входа каждого из которых через пережигаемые перемычки (F1.1 - F1.32) подключены проводники ШД И. Пережигаемые перемычки позволяют пользователю сконфигурировать каждый из 48 элементов И на вычисление произвольной конъюнкции от 16 входных сигналов или 16 их инверсных значений. Выходы 48 элементов И соединены с 48 проводниками, образующими ШД ИЛИ.

Слой ИЛИ состоит их 8 многовходовых элементов ИЛИ. Каждый из 48 входов элементов ИЛИ через пережигаемые перемычки (F2.1 – F2.48) соединены с 48 проводниками шины ШД ИЛИ. Эти пережигаемые

перемычки позволяют пользователю сконфигурировать каждый элемент ИЛИ на вычисление произвольной дизъюнкции от любой комбинации сигналов ШД ИЛИ. Выходные сигналы элементов слоя ИЛИ поступают на 8 выходов микросхемы.

Таким образом, ПЛИС К556РТ1 позволяет пользователю сконфигурировать микросхему на вычисление произвольных 8 логических функций от 16 входных переменных. Для этого требуемые логические функции должны быть представления в дизъюнктивной нормальной форме. Далее в микросхеме с помощью специального устройства (обычно это устройство называют программатором) нужно пережечь «лишние» перемычки.

Как видно из приведенного примера, основное отличие ПЛИС от обычных ПЗУ заключается в том, что пользователь не использует постоянную память с фиксированной архитектурой, а создает свою схему логического устройства. Обычно этот подход позволяет увеличить быстродействие устройства и уменьшить количество используемых элементов. Так, в приведенном примере логическая часть микросхемы К556РТ1 содержит 72 логических элемента и 1920 пережираемых перемычек. Микросхема ПЗУ для реализации тех же функций должна иметь информационную емкость $8*2^{16}$ бит (64К байт). Такая микросхема должна содержать кроме логических элементов дешифратора адреса еще и поле из 524288 (или $8*2^{16}$) пережигаемых перемычек. Учитывая, что пережигаемые перемычки не могут быть плотно размещены на кристалле микросхемы, очевиден вывод – технология ПЛИС имеет явные преимущества перед технологией ПЗУ с точки зрения миниатюризации цифровой электроники. С другой стороны, имеется и недостаток – разработчик устройств на ПЛИС должен хорошо знать цифровую электронику, т.е. иметь высокую инженерную квалификацию.

Последующее развитие ПЛИС было направлено на увеличение количества логических элементов на кристалле микросхемы. Отказ от пережигаемых перемычек и переход на управляемые ключи на основе МОП-транзисторов с “плавающим” затвором позволило создать более емкие программируемые логические матрицы (ПЛМ). Однако желание обеспечить возможность использования всех возможных комбинации сигналов, передаваемых по внутренним информационным шинам ПЛМ, существенно ограничила максимальную информационную емкость этих устройств.

В 1985 году фирма Xilinx выпустила на рынок программируемое логическое устройство нового типа – FPGA (Field Programmable Gate Array). Устройство имеет ячеистую структуру (массив), каждая ячейка которой представляет собой конфигурируемый пользователем логический блок – КЛБ (CLB – Configurable Logic Block). На рисунке 1.16.2 показана типовая ячеистая структура, используемая в FPGA.

КЛБ, расположенные в узлах ячеистой структуры, связаны между собой конфигурируемыми пользователем шинами передачи данных. Каждый КЛБ

состоит из нескольких однотипных логических блоков. Так, типовой логический блок фирмы Xilinx содержит четырехвходовую таблицу соответствия (LUT), мультиплексор (MUX) и триггер (T), схема соединений которых показаны на рисунке 1.16.3.

Таблица соответствия LUT представляет собой 16 однобитовых ячеек статической оперативной памяти, т.е. RAM с организацией 16x1. Входы a-d задают адрес ячейки LUT из которой читается значение. Пользователь может записать в RAM любую свою таблицу и тем самым сконфигурировать логический блок на реализацию требуемой функции. Кроме того, пользователь может использовать LUT просто как оперативную память с организацией 16x1 бит, либо как 16-битный сдвиговый регистр.

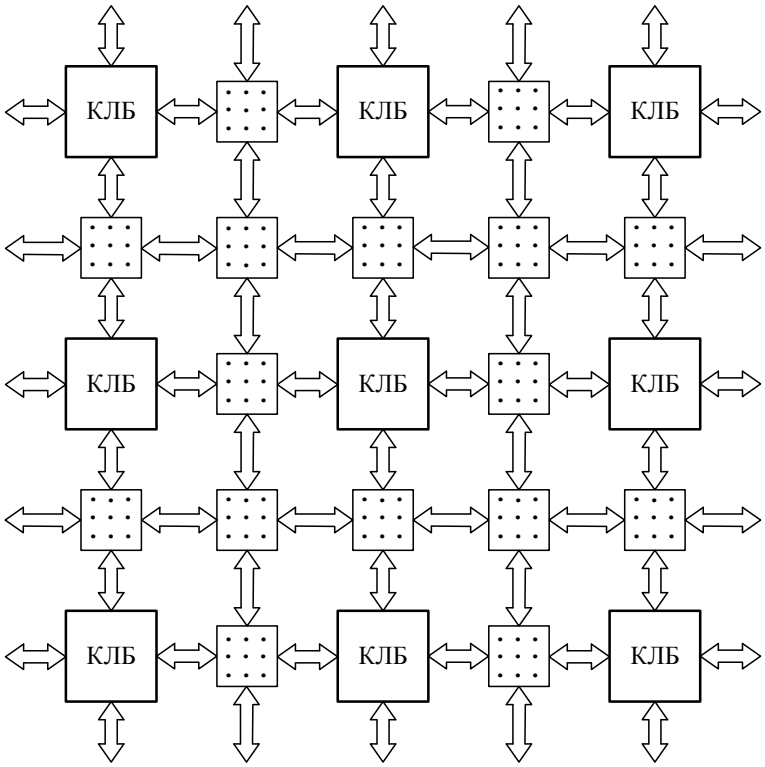

Рисунок 1.16.2 – Типовая ячеистая структура массива из конфигурируемых логических блоков FPGA

Дополнительные элементы логической ячейки – мультиплексор MUX и триггер T еще больше расширяют ее функциональные возможности. Тактирование работы логической ячейки осуществляется специальным сигналом CLK.

Ячеистая архитектура FPGA позволяет достаточно компактно разместить на кристалле микросхемы конфигурируемые пользователем, вычислительные структуры большой информационной емкости.

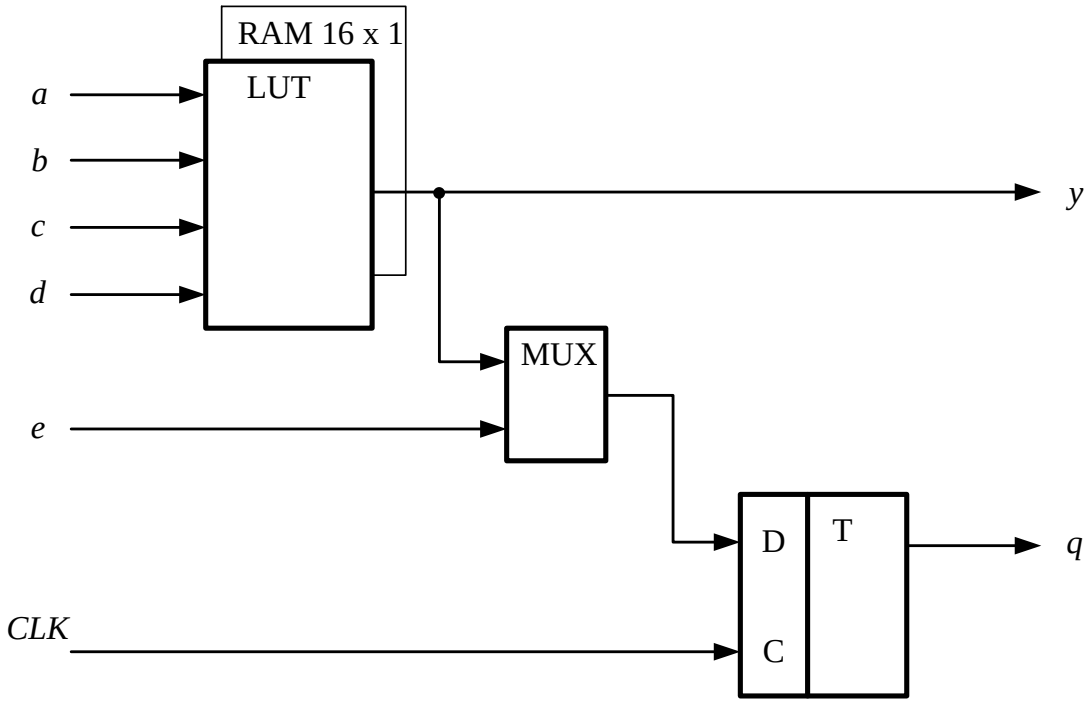

Рисунок 1.16.3 – Пример архитектуры логической ячейки FPGA фирмы Xilinx

Микросхемы FPGA первого поколения сохраняли заданную пользователем конфигурацию в статическом оперативном запоминающем устройстве. Так как память такого типа не сохраняет информацию при выключении питания, то в технические устройства на базе FPGA необходимо было дополнительно устанавливать микросхемы ПЗУ для хранения и загрузки требуемой конфигурации прибора.

В микросхемах FPGA следующих поколений стали использовать встроенную память на базе технологий Flash и EEPROM. Такие устройства получили аббревиатуру CPLD. Кроме того, разработчики FPGA уделяют особое внимание обеспечению возможности изменения конфигурации прибора в процессе его функционирования. Это свойство лучше всего реализуется при использовании внутренней памяти, построенной по технологиям RAM и EEPROM.

Современные FPGA и CPLD позволяют разработчикам осуществить аппаратную реализацию специализированных регистровых вычислительных устройств, обеспечивая высокое быстродействие и относительно невысокую стоимость.

2. Программная и микропрограммная реализации устройств цифрового управления

2.1 Микропрограммные автоматы. Базовые термины

Рассмотренный в разделе 1.11 табличный способ вычислений привлекает разработчиков цифровой техники простотой и компактностью реализации, не зависящей от сложности вычислений. С системной точки зрения дальнейшее развитие цифровой техники, основанной на табличных вычислениях, возможно за счет введения обратной связи. Как известно из теории систем, введение обратной связи позволяет получить новые свойства системы, отсутствующие у ее компонентов.

Охватить обратной связью табличный процессор нельзя из-за его безинерционности (проявится нестабильность и неопределенность в работе). Инерционность в такой системе можно получить только за счет введения в структуру регистра. Поэтому естественным развитием табличного способа вычислений стала вычислительная структура, представляющая собой объединение табличного способа вычислений с регистром, охваченные обратной связь. Такая структура получила название **микропрограммный автомат.**

На рисунке 2.1.1 показана функциональная схема микропрограммного автомата (МПА). На автомат поступает входная переменная *x(m),* код которой объединяется с кодом обратной связи и образует адрес ячейки памяти ROM, хранящей таблицу данных. С приходом тактового импульса считанные данные s(m) записываются в регистр RG. Одна часть выходных данных регистра используется в качестве выходной переменной *y(m)* устройства, другая часть используется для передачи по цепи обратной связи.

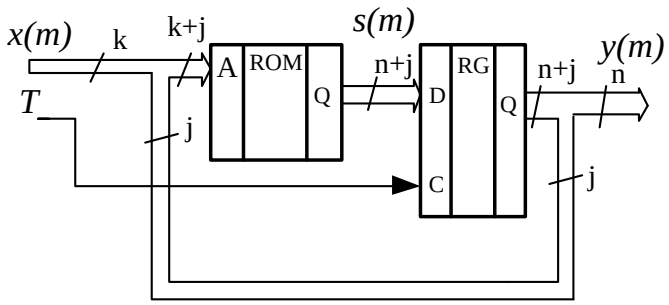

Рисунок 2.1.1 - Функциональная схема микропрограммного автомата

Микропрограммный автомат можно рассматривать, как симбиоз табличного и регистрового способов вычислений. Из-за наличия в схеме регистра обновление выходной переменной происходит в дискретные моменты времени, определяемые поступлением входного тактового импульса. Из-за использования табличного способа, автомат ничего не вычисляет, он только воспроизводит записанные в ПЗУ данные. Однако, в

отличие от табличного способа вычислений, за счет имеющейся обратной связи автомат способен формировать кодовые последовательности при постоянном входном сигнале.

Пусть для примера разрядности переменных *x(m)* и обратной связи равны k=j=2, а разрядность выходной переменной *y(m)* рана 4. Пусть также в начальные ячейки ПЗУ записаны коды, приведенные в таблице 2.1.1. Предположим для определенности, что входной сигнал *x(m)*=0 и сигнал по цепи обратной связи равен 0. Тогда с приходом первого тактового импульса в регистр запишется содержимое нулевой ячейки, т.е. двоичное число 000101. Это число разделится на две составляющие – выходную переменную *y(1)*=0001 и сигнал обратной связи *ос* = 01. Сигнал обратной связи объединяется с входной переменной и образует новый адрес 0001 на входе ПЗУ. С приходом следующего тактового импульса в регистр запишется содержимое первой ячейки ПЗУ, т.е. число 001010. Это число опять делится на две части - выходную переменную *y(2)*=0010 и сигнал обратной связи *ос*=10.

Таблица 2.1.1 Содержимое начальных ячеек ПЗУ

Адрес ячейки		Данные	
x	*ос*	*Y*	*ос*
0000		000101	
0001		001010	
0010		010011	
0011		100000	

С приходом следующего тактового импульса в регистр запишется уже содержимое второй ячейки ПЗУ, изменятся значения выходной переменной *y(3)*=0100 и сигнал обратной связи ос = 11. Далее, с приходом следующего тактового импульса в регистр запишется содержимое третьей ячейки ПЗУ, изменятся значения выходной переменной *y(4)*=1000 и сигнал обратной связи ос = 00. Наконец, с приходом пятого тактового импульса информация снова прочитается из нулевой ячейки и процесс начнет циклически повторяться.

Таким образом, записав в ПЗУ указанные в таблице 2.1.1 коды, мы **запрограммировали** автомат на генерирование периодической кодовой последовательности *y(m)* = 0001, 0010, 0100, 1000, 0001, **…**. Указанная кодовая последовательность будет вырабатываться автоматом при постоянном входном сигнале *x(m)* = 00. Если изменить входной сигнал, то данные будут читаться из другой части ПЗУ. Это позволяет запрограммировать автомат на генерацию другой кодовой последовательности. Таким образом, число на входе *x* можно трактовать, как номер генерируемой автоматом кодовой последовательности.

Разрабатывая таблицу данных для записи в ПЗУ, мы по существу программируем автомат на формирование определенной последовательности кодов. Однако мы программируем не последовательность операций, как вычислительной машине, а непосредственно последовательность выходных кодов и сигналов обратной связи. Поэтому такая процедура называется **микропрограммированием**. Соответственно, содержимое постоянного запоминающего устройства называется **микропрограммой**, а содержимое одной ячейки памяти – **микрокомандой**. ПЗУ называют памятью микропрограммы, а регистр - **регистром микрокоманд.**

Непосредственно из анализа схемы, показанной на рисунке 2.1.1, и рассмотренного примера следует, что микропрограммный автомат имеет следующие особенности:

- автомат вырабатывает на выходе периодическую цифровую кодовую последовательность с тактом *T;*
- автомат ничего не вычисляет, все значения генерируемой последовательности заранее записаны в память автомата;
- период кодовой последовательности при постоянном входном сигнале не может превышать 2^j тактов, где j – разрядность шины обратной связи;
- количество различных кодовых последовательностей автомата при постоянстве входного числа не может превышать 2^k, где k – разрядность входной шины данных;
- количество различных значений выходной переменной не может превышать 2^n, где n – разрядность выходной шины данных.

Микропрограммные автоматы применяются в качестве командных генераторов в цифровых следящих системах, применяются для непосредственного управления объектами типа шаговых двигателей, широко применяются в системах телемеханики и кодирования информации и т.п. Но основная масса автоматов применяется для управления ходом вычислений в счетно-решающих устройствах. Все современные компьютеры, как универсальные, так и управляющие, работают под управлением микропрограммных автоматов, определяющих последовательность действий процессора для реализации команд пользователя, [15].

2.2 Базовая функциональная схема микропрограммного автомата

Микропрограммный автомат, функциональная схема которого показана на рисунке 2.1.1, очень прост и компактен. Однако разработать программу его функционирования бывает довольно трудно, особенно при больших разрядностях шин передачи данных. Это связано с тем, что изменение входного кода *x(m)* может произойти на любом такте времени. Это изменение вызовет переключение адресного пространства в ПЗУ на другую область памяти; автомат начнет работать по другой ветви программы. Разработчику оказывается очень сложно прогнозировать поведение автомата в различных

меняющихся ситуациях. Зачастую ручной синтез автомата оказывается просто невозможным из-за слишком большой трудоемкости.

Выход из такой ситуации – применить компьютер для формального синтеза программы работы автомата и анализа возможных переходов при смене входного кода. В настоящее время все ведущие фирмы, выпускающие соответствующие цифровые микросхемы, предлагают прикладное программное обеспечение, позволяющее формально синтезировать автомат. Поддерживается полный цикл разработки:

- пользователь задает разрядности входной и выходной переменных, тактовую частоту работы устройства и алгоритм работы автомата на специальном языке высокого уровня;
- компьютер определяет требуемые объемы регистровой и долговременной памяти и предлагает пользователю подходящие марки серийно выпускаемых микросхем;
- пользователь выбирает желаемую микросхему и компьютер разрабатывает микропрограмму работы для конкретного изделия;
- пользователь проверяет функционирование автомата с помощью программного симулятора;
- с помощью программатора, подключенного к компьютеру, пользователь физически записывает микропрограмму в память выбранной микросхемы.

После выполнения указанных действий микропрограммный автомат, разработанный пользователем, может быть изготовлен и использован по назначению.

В данном разделе мы будем использовать другой подход. Для того чтобы разобраться в особенностях программирования автоматов и понять их возможности, применим ручную (т.е. неформальную) разработку микропрограмм. Для облегчения процедуры программирования усложним схему микропрограммного автомата, запретив в его функционировании несанкционированные переходы по программе.

На рисунке 2.2.1 показана базовая функциональная схема микропрограммного автомата.

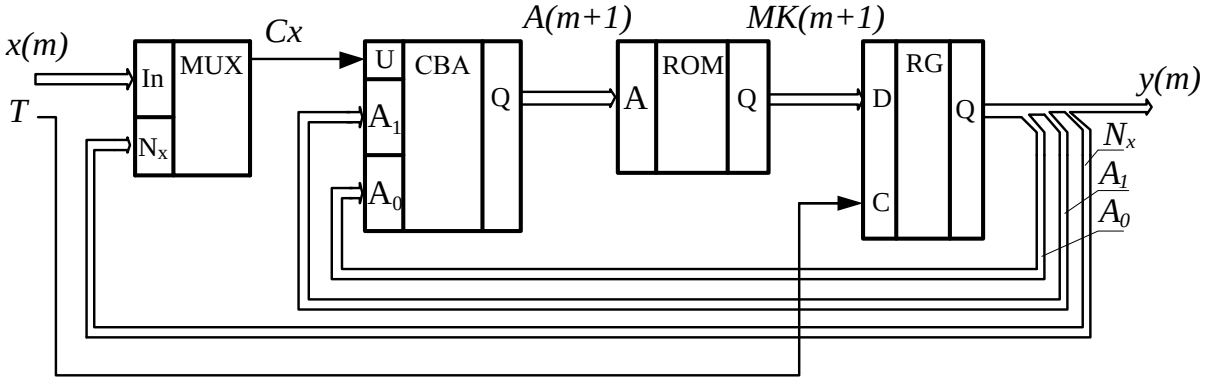

Рисунок 2.2.1 - Базовая функциональная схема микропрограммного автомата

В базовую схему добавлены два новых элемента: MUX – мультиплексор и СВА – схема вычисления адреса следующей микрокоманды. Используя базовую схему, мы рассмотрим основные типы автоматов и особенности их программирования.

2.3 Микропрограммный автомат с принудительной адресацией и двумя адресными полями

Микрокоманда автомата, функциональная схема которого показана на рисунке 2.2.1, имеет определенную структуру. Действительно, выходной код регистра RG разделяется по шинам – для кода выхода *y(m)*, для кода *Nx*, для кодов A_0 и A_1. Принято говорить, что микрокоманда автомата состоит из **полей**: поля *Y*, поля *Nx*, полей A_0 и A_1. Распределение указанных битовых полей в микрокоманде по разрядам кода называется **форматом микрокоманды.**

Микропрограммный автомат с принудительной адресацией и двумя адресными полями использует микрокоманды, содержащие четыре поля. На рисунке 2.3.1 показан формат микрокоманды автомата (а) и ее эквивалентное графическое изображение (б).

Четыре поля микрокоманды – это **поле значения** выходной переменной *Y* (оператор присваивания), **поле номера** разряда проверяемого входа *X* (оператор проверки условия) и два **адресных поля** A_0 и A_1. Логика исполнения условной части команды следующая: если на проверяемом разряде входа (*X*) сигнал равен 0, то следующая микрокоманда читается из ячейки с адресом, указанным в поле A_0, иначе – из ячейки с адресом, указанным в поле A_1. Так как одна микрокоманда исполняется целиком за один такт работы автомата, то операция присваивания и условная операция неразделимы. Этот факт необходимо учитывать при разработке программы, т.е. программу надо строить из блоков по две операции, как показанно на рисунке 2.3.1б.

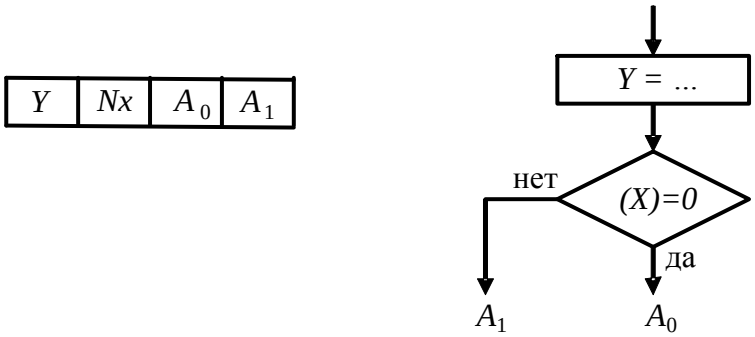

Рисунок 2.3.1 - МПА с принудительной адресацией и двумя адресными полями: формат микрокоманды (а) и ее графическое изображение (б)

Рассмотрим пример. Предположим, что необходимо разработать устройство управления четырехфазным шаговым электродвигателем с активным ротором. Устройство должно иметь один управляющий вход, разрешающий вращения ротора двигателя и формировать четыре бинарных управляющих сигнала. Укрупненная функциональная схема электропривода с шаговым электродвигателем показана на рисунке 2.3.2. Для обеспечения вращения ротора шагового двигателя необходимо сформировать последовательность импульсов напряжения (фазные напряжения) на его обмотках. Эта последовательность определяет направление вращения ротора двигателя и его эксплуатационные характеристики. Требуемую последовательность фазных напряжений сформируем с помощью микропрограммного автомата.

Микропрограммный автомат МПА, см. рисунок 2.3.2, формирует четырехразрядный выход *y(m)*, каждый бит которого определяет включение/выключение одной из четырех обмоток шагового двигателя ШД. Эти четыре бинарных сигнала усиливаются по мощности блоком усилителей БУс и подаются на обмотки шагового двигателя.

Однобитовый входной сигнал *x(m)* задается пользователем посредством переключателя и разрешает вращение ротора шагового двигателя: 0 - ротор вращается, 1 – ротор не вращается. Таким образом, необходимо разработать микропрограммный автомат, вырабатывающий при $x = 0$ циклическую кодовую последовательность *y(m)* = 0001: 0010: 0100; 1000; 0001; **…**, а при $x = 1$ – выполнение команды СТОП.

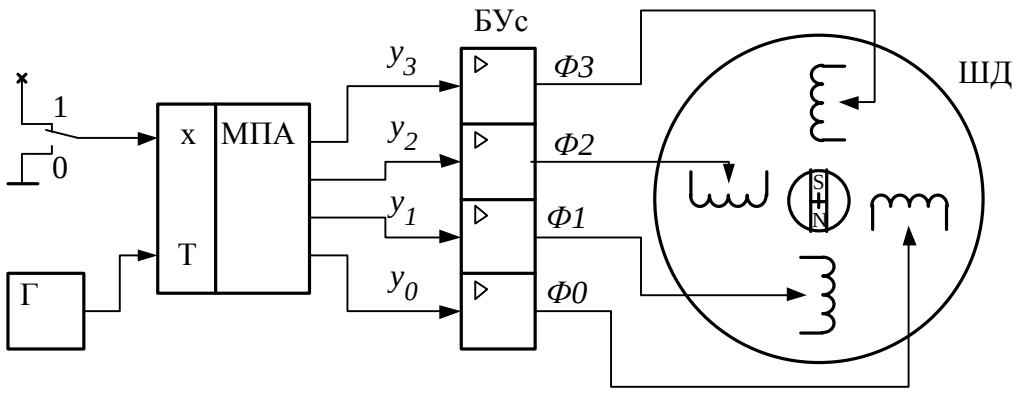

Рисунок 2.3.2 - Укрупненная схема электропривода с микропрограммным управлением

На рисунке 2.3.3 показана блок-схема программы, реализующей принятый алгоритм работы МПА с принудительной адресацией и двумя адресными полями. Она представлена последовательностью микрокоманд, каждая из которых состоит из двух операций – присваивания и проверки условия. Микрокоманды обозначены символами S_i, где i – номер команды, совпадающий с адресом ячейки памяти ПЗУ, в которой хранится команда

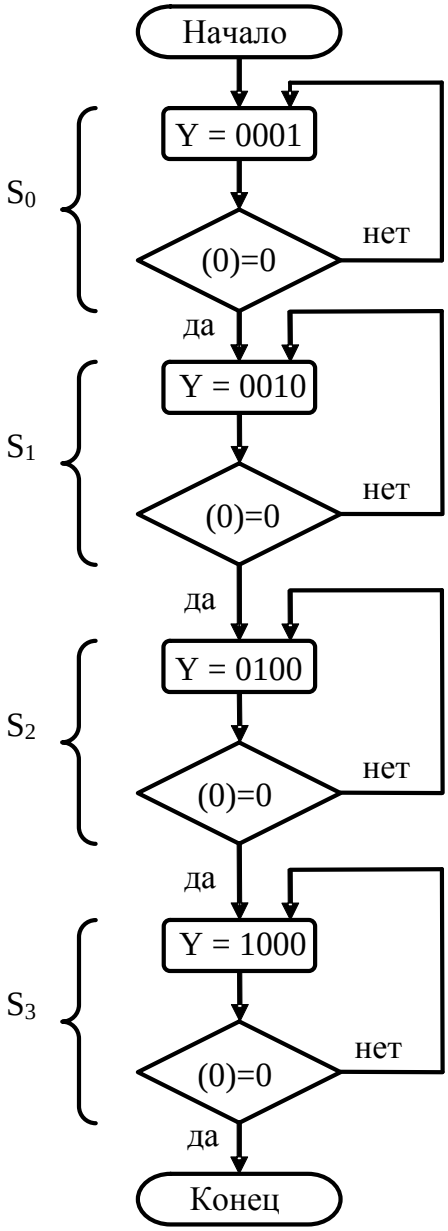

Рисунок 2.3.3 - Блок-схема программы работы микропрограммного автомата

(так мы приняли для удобства). При назначении адресов ячеек памяти (т.е. индексов i команд) был применен самый простой принцип – индекс увеличивается на единицу последовательно по мере появления команды в блок-схеме программы. В каждой команде выполняется проверка состояния входного сигнала и выполняется соответствующий переход к следующей команде. Если входной сигнал $x = 0$, то читается микрокоманда из следующей ячейки; если сигнал $x = 1$, то повторяется выполнение текущей микрокоманды, т.е. реализуется команда СТОП. Согласно принятым правилам изображения блок-схем программ, метки НАЧАЛО и КОНЕЦ обозначают начало и конец **цикла** программы, т.е. представляют одну и ту же точку программы.

Важными численными показателями микропрограммных автоматов являются разрядность микрокоманды и требуемый объем ПЗУ. Разрядность микрокоманды базовой схемы складывается из разрядности поля выходной переменной, поля номера *Nx* и двух адресных полей. Разбивка блок-схемы программы на команды показала, что весь алгоритм работы МПА состоит всего из четырех микрокоманд, для хранения которых требуется четыре ячейки памяти. Следовательно, разрядность полей адресов A_0 и A_1 может быть принята равной 2. Поле *Nx* будет состоять всего из 1 разряда, так как необходимо управлять всего одной входной переменной. Разрядность поля *Y* для удобства реализации примем равным 4. Формат получившейся микрокоманды показан на рисунке 2.3.4., разрядность микрокоманды получилась равной 9 бит.

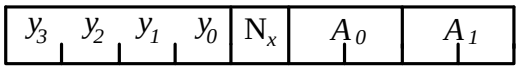

Рисунок 2.3.4 - Формат микрокоманды автомата управления шаговым двигателем

Определив формат микрокоманды автомата управления шаговым электродвигателем, можем разработать микропрограмму работы автомата в виде таблицы двоичных кодов для записи в ПЗУ. Микропрограмма работы автомата приведена в таблице 2.3.1. Объем ПЗУ, необходимый для хранения программы составляет M=4*9 = 36 бит.

Таблица 2. Программа автомата

№ ячейки	Данные в ПЗУ			
	Y	*Nx*	A_0	A_1
00	0001	0	01	00
01	0010	0	10	01
10	0100	0	11	10
11	1000	0	00	11

Последнее, что осталось определить для микропрограммного автомата с двумя адресными полями – определиться с реализацией схемы вычисления адреса следующей микрокоманды. Из алгоритма функционирования микропрограммного автомата с двумя адресными полями следует, что схема вычисления адреса в зависимости от состояния сигнала на управляющем входе *Cx* должна передавать на выход либо код с шины адреса *A0*, либо код с шины адреса *A1*. Обозначения шин выбраны так, чтобы разработчик не путался: при *Cx* = 0 передается код с шины *A0*, при *Cx* = 1 передается код с шины *A1*. Из этого следует, что в качестве схемы вычисления адреса в

автомате с двумя адресными полями целесообразно использовать мультиплексор.

Схема микропрограммного автомата для рассматриваемого примера показана на рисунке 2.3.5. В схеме в качестве СВА используется мультиплексор двух двухразрядных шин на двухразрядную выходную шину. На входе автомата также установлен мультиплексор. Этот мультиплексор переключает две одноразрядные шины на одноразрядный выход. Скорость вращения ротора шагового двигателя определяется частотой поступающих на автомат тактовых импульсов.

Входной мультиплексор в схеме на рисунке 2.3.5 установлен для демонстрации практической реализации базовой функциональной схемы автомата. Если внимательно рассмотреть алгоритм работы автомата, то нетрудно заметить, что входной мультиплексор в схеме практически не используется. Действительно, управляющий сигнал *Nx* во всех микрокомандах равен нулю, т.е мультиплексор постоянно передает входной

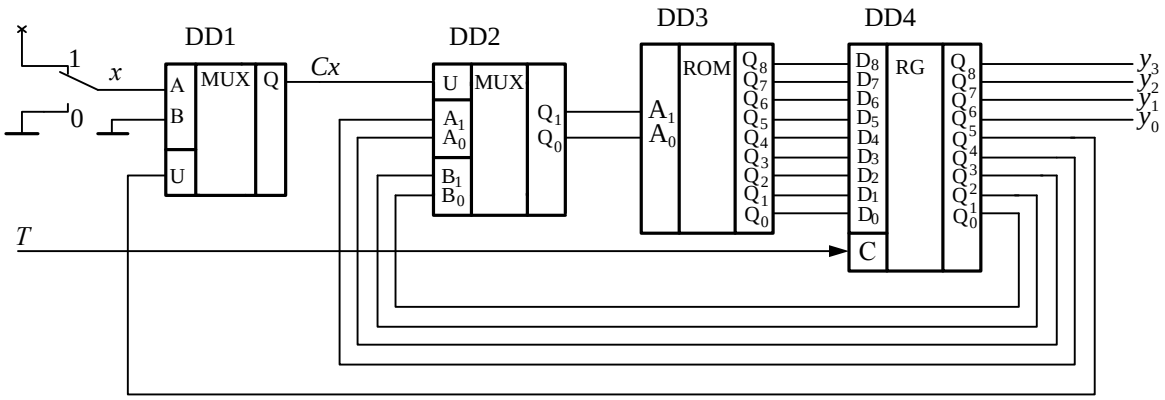

Рисунок 2.3.5 – Схема микропрограммного автомата для управления шаговым двигателем

сигнал на свой выход. Следовательно, схему автомата, показанную на рисунке 2.3.5, можно упростить.

Основное достоинство микропрограммных автоматов с принудительной адресацией и двумя адресными полями заключается в простоте программирования. Действительно, нумеровать команды и располагать их в памяти программ можно в любом порядке, так как в автомате нет ограничений на переходы по программе.

Недостатком такого микропрограммного автомата является относительно большая длина микрокоманды. Это вызвано тем, что команда содержит два поля адреса, которые при большом числе команд могут иметь значительную длину. Попытка убрать этот недостаток микропрограммного автомата с принудительной адресацией и двумя адресными полями привел к созданию автоматов других типов.

2.4 Микропрограммный автомат с принудительной адресацией и одним адресным полем

В микропрограммном автомате с принудительной адресацией и одним адресным полем делается попытка сократить длину микрокоманды. Для этого в микрокоманде ликвидируется поле адреса A_1 и задача вычисления этого адреса перекладывается на схему вычисления адреса СВА.

На рисунке 2.4.1 показан формат используемой микрокоманды (а) и ее графическое изображение (б).

Микрокоманда состоит из 3 полей – **поля значения** выходной переменной *Y* (оператор присваивания), **поля номера** проверяемого входа *X* (оператор проверки условия) и **адресного поля** A_0. Логика исполнения условной части команды следующая: если на проверяемом входе (*X*) сигнал равен 0, то следующая микрокоманда читается из ячейки с адресом A_0, иначе из ячейки с адресом A_0+1. Так как одна микрокоманда исполняется целиком

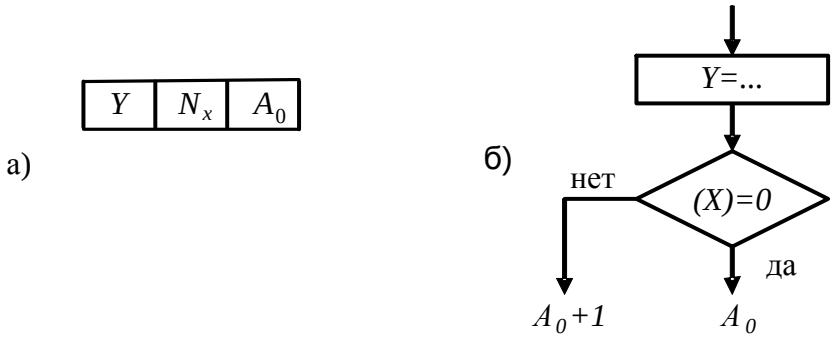

Рисунок 2.4.1 - МПА с принудительной адресацией и одним адресным полем: формат микрокоманды (а) и ее графическое изображение (б).

за один такт работы автомата, то операция присваивания и условная операция неразделимы. Этот факт необходимо учитывать при разработке программы, т.е. программу надо строить из блоков по две операции. Есть и еще одно ограничение – переход по программе может быть выполнен в одну из двух соседних ячеек - с адресами A_0 и A_0+1.

Схема вычисления адреса следующей микрокоманды может быть реализована весьма просто на параллельном сумматоре. Действительно, так как бинарный сигнал *Cx* входного мультиплексора принимает только два значения, то логику исполнения условной части микрокоманды модно представить уравнением:

$$A(m+1) = A_0 + Cx.$$

Для реализации данного уравнения естественно воспользоваться параллельным сумматором. На рисунке 2.4.2 показана реализация схемы

вычисления адреса следующей микрокоманды для автомата с принудительной адресацией и одним адресным полем.

Рассмотрим пример применения микропрограммного автомата с принудительной адресацией и одним адресным полем в устройстве

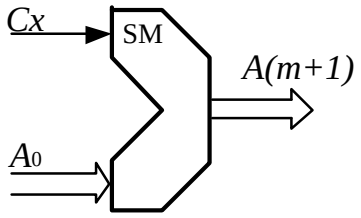

Рисунок 2.4.2 - Реализация схемы вычисления адреса следующей микрокоманды

управления шаговым двигателем. Функциональная схема устройства показана на рисунке 2.3.2.

На рисунке 2.4.3 показана блок-схема программы, реализующая кодовую последовательность рассматриваемого примера работы микропрограммного автомата. Она содержит последовательность команд, каждая из которых состоит из двух операций – присваивания и проверки условия. Команды программы обозначены символами S_i, где i – адрес ячейки памяти, где хранится команда (так принято для удобства кодирования программы). У микропрограммных автоматов данного типа возможны адресные переходы только в две соседние ячейки, поэтому команды программы должны располагаться парами в соседних ячейках памяти программ и их индексы должны различаться на единицу. При назначении адресов ячеек памяти (индексов i команд) эта особенность работы автомата была учтена.

Основная проблема разработки программы для автоматов рассматриваемого типа связана с невозможностью выполнить некоторые требуемые переходы. Так, например, команда S_0 программы для автомата с двумя адресными полями (см. рис.2.3.3) содержит переходы к ячейке S_1 и к самой себе, причем при $X = 0$ переход должен быть выполнен к ячейке S_1. В автомате с одним адресным полем такие переходы организовать невозможно. Действительно, если в команде S_0 при $X = 0$ автомату задать переход к ячейке S_1, то при **X** = 1 автомат **самостоятельно** перейдет к ячейке S_2 (автомат вычислит адрес перехода как $S_1 + 1$). Основной способ разрешить данное противоречие – продублировать требуемую команду, поместив ее в ячейку памяти, номер которой вычисляет автомат.

Для устранения противоречий в блок-схему программы на рисунке 2.4.3 введены дублирующие команды: S_3 дублирует S_0, S_5 дублирует S_2, S_7 дублирует S_4, а S_1 дублирует S_6. За счет использования дублирующих команд программа для микропрограммного автомата с принудительной адресацией и одним адресным полем оказывается более громоздкой, чем

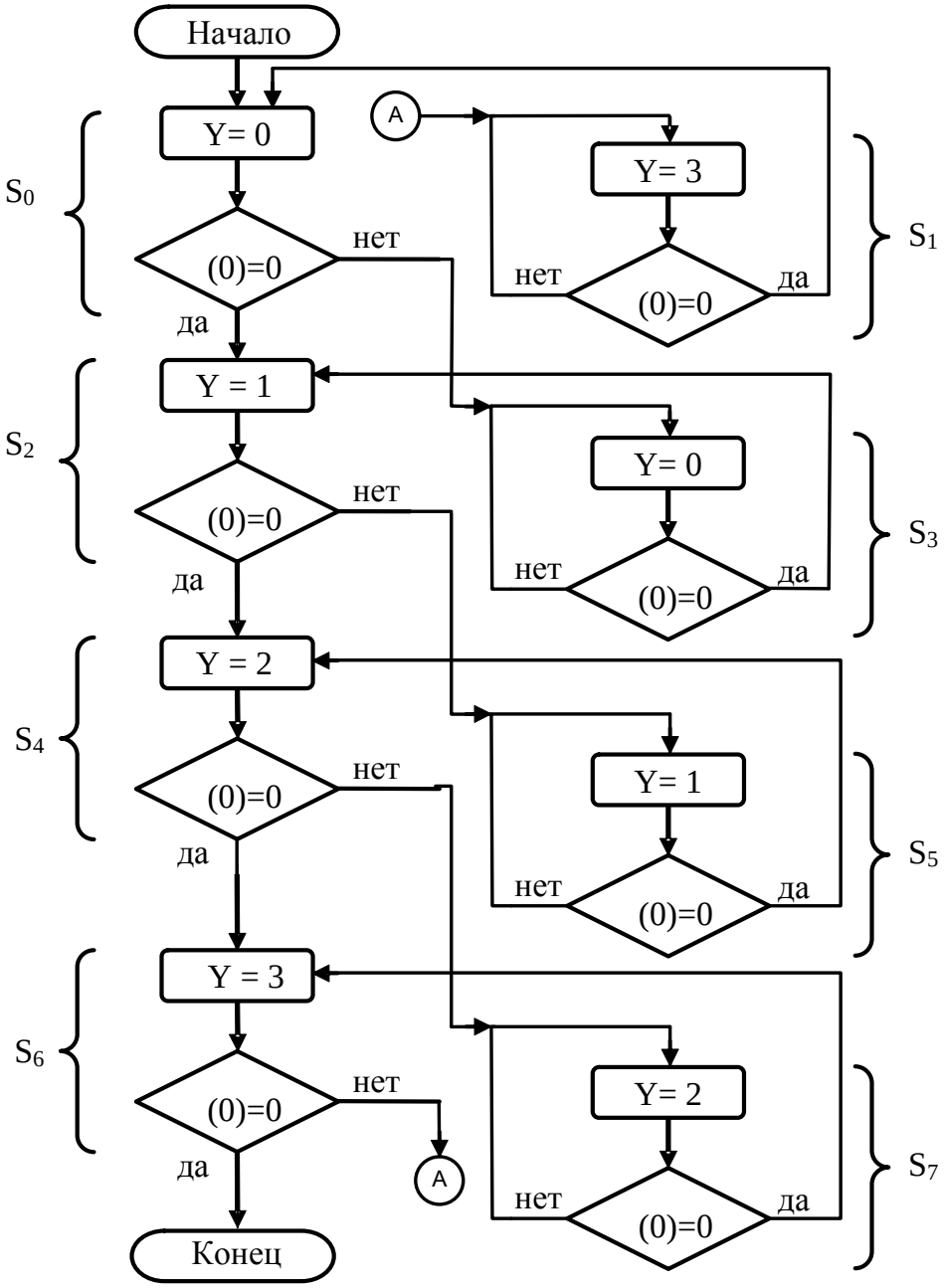

Рисунок 2.4.3 - Блок-схема программы работы автомата с принудительной адресацией и одним адресным полем.

программа для автомата с принудительной адресацией и двумя адресными полями.

Так как адрес команды в памяти микропрограммы определяется номером команды, то по блок-схеме нетрудно написать микропрограмму работы автомата в виде таблицы в двоичных кодах. Микропрограмма работы автомата для данного примера приведена в таблице 2.4.1.

Как видно из таблицы 3, микрокоманда автомата содержит 8 бит. Всего команд 8, занимающих 8 ячеек памяти. Для кодирования адреса микрокоманды необходимо использовать 3 бита, для хранения значения

выходной переменной как и ранее используем 4 бита. Объем ПЗУ, необходимый для записи программы, составляет M=8*8 = 64 бита.

На рисунке 2.4.4 показана итоговая схема микропрограммного автомата с принудительной адресацией и одним адресным полем для задачи управления шаговым двигателем. Скорость вращения ротора шагового двигателя определяется частотой поступающих на автомат тактовых импульсов. Как и в примере реализации автомата с двумя адресными полями, входной мультиплексор из данной схемы можно убрать.

Рассмотренный пример показал, что попытка сократить длину микрокоманды в автомате с принудительной адресацией привела к увеличению длины и сложности микропрограммы. Разрядность микрокоманды в нашем примере, по сравнению с микропрограммным автоматом с принудительной адресацией и двумя адресными полями,

Таблица 2.4.1 Микропрограмма работы автомата с принудительной адресацией и одним адресным полем

№ ячейки	Данные в ПЗУ		
	Y	*X*	A_0
000	0001	0	010
001	1000	0	000
010	0010	0	100
011	0001	0	010
100	0100	0	110
101	0010	0	100
110	1000	0	000
111	0100	0	110

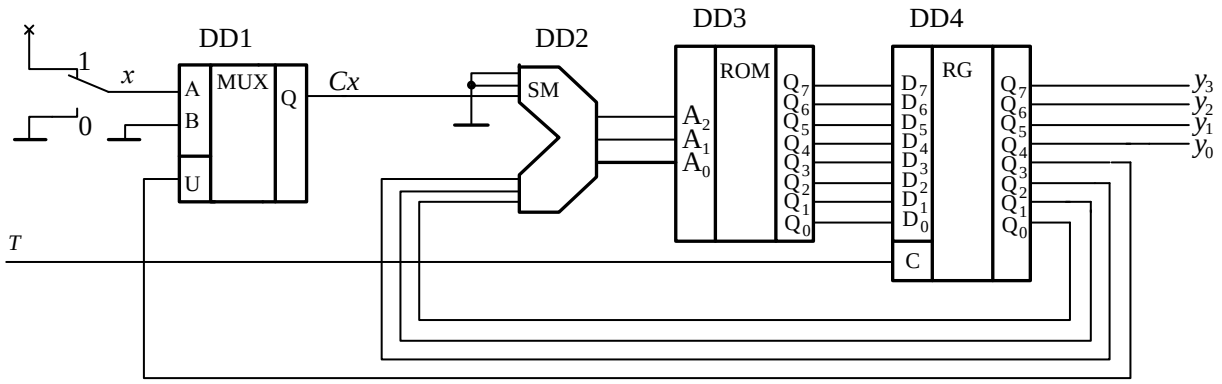

Рисунок 2.4.4 - Схема микропрограммного автомата для управления шаговым двигателем

сократилась всего на 1 бит. При этом общий объем ПЗУ увеличился по сравнению с тем же автоматом с 36 до 64 бит.

Конечно, существуют специфические задачи, в которых микропрограммные автоматы с принудительной адресацией и одним адресным полем дают экономию в аппаратных средствах. Однако сложность разработки программы для таких автоматов существенно ограничивает область их применения.

2.5 Микропрограммный автомат с естественной адресацией

В микропрограммном автомате с естественной адресацией делается попытка сократить длину микрокоманды. Для этого в микрокоманде ликвидируется поле адреса A_0, оставляется поле адреса A_1, а задача определения адреса A_0 перекладывается на схему вычисления адреса СВА.

Понятие естественной адресации связано с понятием естественной последовательности выполнения команд. Последовательность команд называется естественной, если следующая по алгоритму исполняемая команда читается из ячейки памяти, адрес которой на единицу больше адреса предыдущей. Другими словами, естественная последовательность команд располагается в памяти программ в последовательно расположенных ячейках. Для реализации такой программы достаточно последовательно прочитать содержимое памяти, наращивая с каждым тактом адрес на единицу, т.е. реализовать следующую формулу

$$A(m+1) = A(m) + 1.$$

На рисунке 2.5.1 показан формат микрокоманды автомата с естественной адресацией (а) и ее графическое изображение (б).

Микрокоманда состоит из 3 полей – **поля значения** выходной переменной Y (оператор присваивания), **поля номера** проверяемого входа X (оператор проверки условия) и **адресного поля** A_1. Логика исполнения условной части команды следующая: если на проверяемом входе (X) сигнал равен 0, то следующая микрокоманда читается из ячейки с адресом на единицу больше предыдущего, иначе – из ячейки с адресом A_1. . Так как одна микрокоманда исполняется целиком за один такт работы автомата, то операция присваивания и условная операция неразделимы. Этот факт необходимо учитывать при разработке программы, т.е. программу надо строить из блоков по две операции.

Схемы вычисления адреса следующей микрокоманды может быть выполнена весьма просто на счетчике тактовых импульсов. На рисунке 2.5.2 показана реализация схемы вычисления адреса следующей микрокоманды в автомате с естественной адресацией. Действительно, с приходом каждого тактового импульса содержимое счетчика будет увеличиваться на единицу, реализуя естественную последовательность выполнения команд. Переход по

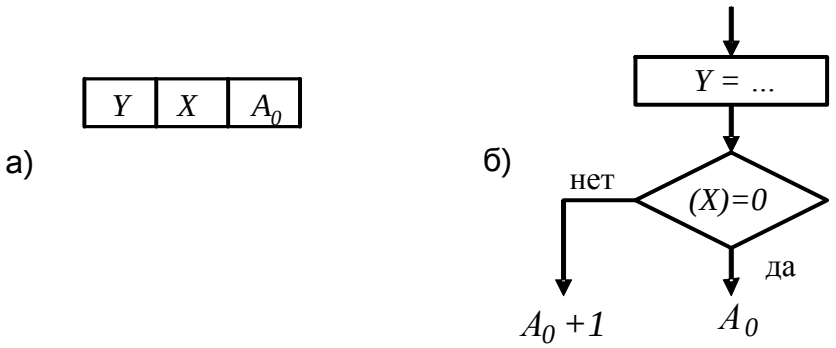

Рисунок 2.5.1 - Микропрограммный автомат с естественной адресацией: формат микрокоманды (а) и ее графическое изображение (б).

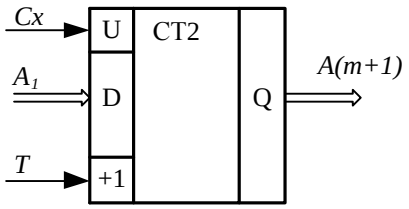

Рисунок 2.5.2 - Реализация схемы вычисления адреса следующей микрокоманды в автомате с естественной адресацией

адресу A_1 осуществляется при записи в счетчик нового числа из поля A_1 микрокоманды. Для этой цели используются входы предустановки счетчика, которые активируются при поступлении единицы на управляющий вход *U*.

Рассмотрим пример применения микропрограммного автомата с естественной адресацией в устройстве управления шаговым двигателем. Функциональная схема устройства показана на рисунке 2.3.2.

На рисунке 2.5.3 показана блок-схема программы для рассматриваемого примера микропрограммного автомата. Она представляет собой последовательность команд, каждая из которых состоит из двух операций – присваивания и проверки условия. Команды микропрограммы обозначены символами S_i, где i – адрес ячейки памяти, где хранится команда (принято удобства). У микропрограммных автоматов данного типа реализуются естественные адресные переходы к следующей ячейке при $Cx = 0$. Поэтому в программе выделена естественная последовательность команд (исполняемых при Cx=0), которым присвоена последовательность возрастающих индексов. При Cx=1 переход может быть выполнен по любому адресу, поэтому такие переходы используются для изменения хода естественной последовательности.

Микропрограмма работы для микропрограммного автомата с естественной адресацией разрабатывается без особых проблем. Единственная особенность - необходимо прерывать ход естественной последовательности выполнения команд в конце программного цикла. Действительно, при

условии $Cx = 0$ после выполнения команды S_4 автомат самостоятельно перейдет к выполнению команды S_5. Так как микрокоманда S_5 в алгоритме не предусмотрена, то это действие автомата необходимо запретить.

Запретить переход к выполнению следующей микрокоманды в конце программного цикла можно несколькими способами. Универсальным является способ, базирующийся на введении дополнительного однобитового входа. В этом случае на дополнительный вход постоянно подается единица (вход подключается к плюсу источника питания). Теперь для запрещения естественного хода выполнения микрокоманд следует выполнить проверку

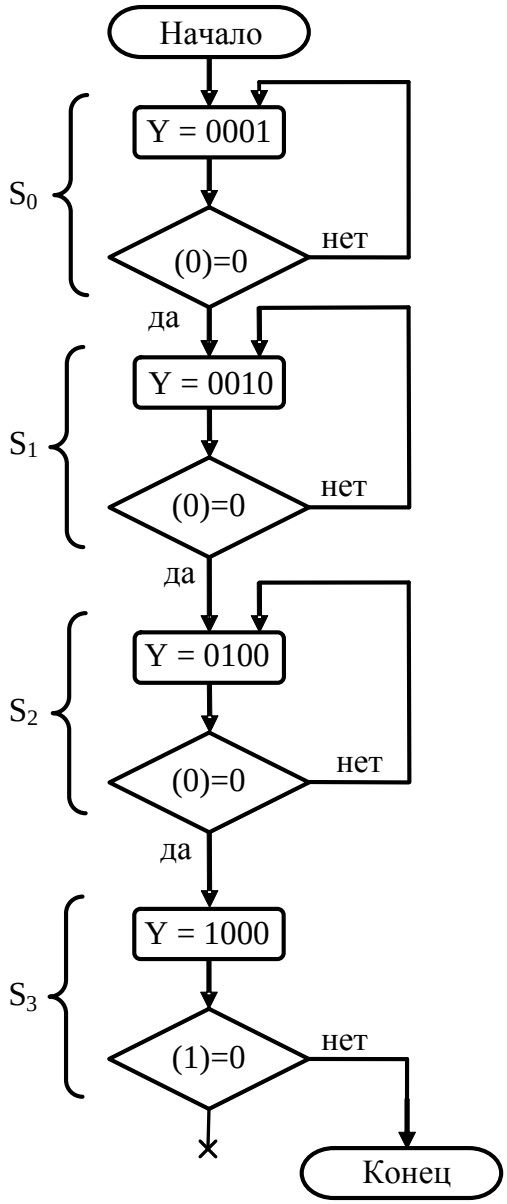

Рисунок 2.5.3 - Блок-схема микропрограммы работы автомата с естественной адресацией.

состояния этого дополнительного входа. Так как при проверке состояние дополнительного входа всегда равно единице, то автомат выполнит переход

по адресу A_1. Именно этот способ запрещения естественного хода выполнения микрокоманд применен в примере реализации программы, представленном на рисунке 2.5.3.

Другой способ запрещения естественного хода выполнения микрокоманд основан на введении дополнительного разряда в микрокоманду. Учитывая, что запрещение естественного хода выполнения микрокоманд в автомате выполняется в конце программного цикла и переход осуществляется в начало цикла, т.е. к ячейке с адресом 0, сигнал с дополнительного разряда микрокоманды подается на вход *R* сброса счетчика в ноль. Однако следует помнить, что рассмотренный способ имеет «подводный камень». У большинства двоичных счетчиков вход сброса *R* асинхронный, т.е. сброс счетчика происходит без привязки к тактовому импульсу. Поэтому для сброса счетчика в ноль необходимо ввести еще одну команду, дублирующую команду S0, но с активизированным дополнительным разрядом. Блок- схему такой программы попробуйте нарисовать самостоятельно.

Третий способ запрещения естественного хода выполнения микрокоманд применим только в частном случае. В случае, когда число микрокоманд программы равно 2^j, где j - целое число, в качестве схемы вычисления адреса следующей микрокоманды используется j - разрядный счетчик микрокоманд. При определенной структуре микропрограммы, когда естественная последовательность микрокоманд начинается с адреса 0 и заканчивается адресом (j-1), переход к следующей ячейке в конце цикла приводит к переполнению счетчика микрокоманд и естественному сбросу в 0 выходного кода. Другими словами, счетчик микрокоманд сам выполнит сброс в ноль своего содержимого. В блок-схеме микропрограммы, показанной на рисунке 2.5.3 можно применить именно такой частный случай. Действительно, число микрокоманд в алгоритме равно 4, естественная последовательность микрокоманд начинается с адреса 0 и заканчивается адресом 3. Если использовать в схеме микропрограммного автомата в качестве СВА двухразрядный двоичный счетчик, то переход от вершины КОНЕЦ к вершине НАЧАЛО будет выполняться без введения в схему дополнительных входов и выходов.

Для пояснения сказанного, рассмотрим пример реализации микропрограммного автомата с применением способа введения дополнительного входа. Как уже говорилось, блок-схема программы работы микропрограммного автомата показана на рисунке 2.5.3. Так как адрес микрокоманды в памяти микропрограммы совпадает с номером микрокоманды, то легко написать микропрограмму работы автомата в двоичных кодах для записи в ПЗУ. Микропрограмма работы автомата приведена в таблице 2.5.1.

Как видно из таблицы 2.5.1, микрокоманда автомата содержит 7 бит. Всего команд 4. Для кодирования адреса микрокоманды необходимо использовать 2 бита, для хранения значения выходной переменной для

удобства реализации будем использовать 4 бита. Объем ПЗУ, необходимый для записи программы, составляет M = 4*7 = 28 бита. Из этого следует, что микропрограмный автомат с естественной адресаций требует для решения задачи управления шаговым двигателем минимального объема ПЗУ из рассмотренных автоматов.

На рисунке 2.5.4 показана итоговая схема микропрограммного автомата с естественной адресацией для задачи управления шаговым двигателем. Скорость вращения ротора шагового двигателя определяется частотой поступающих на автомат тактовых импульсов. В отличие от примера реализации автомата с двумя адресными полями, входной мультиплексор из данной схемы убрать нельзя.

Таблица 2.5.1 Микропрограмма работы автомата с естественной адресацией

Адрес ячейки	Данные в ПЗУ		
	Y	*Nx*	A_0
00	0001	0	00
01	0010	0	01
10	0100	0	10
11	1000	1	11

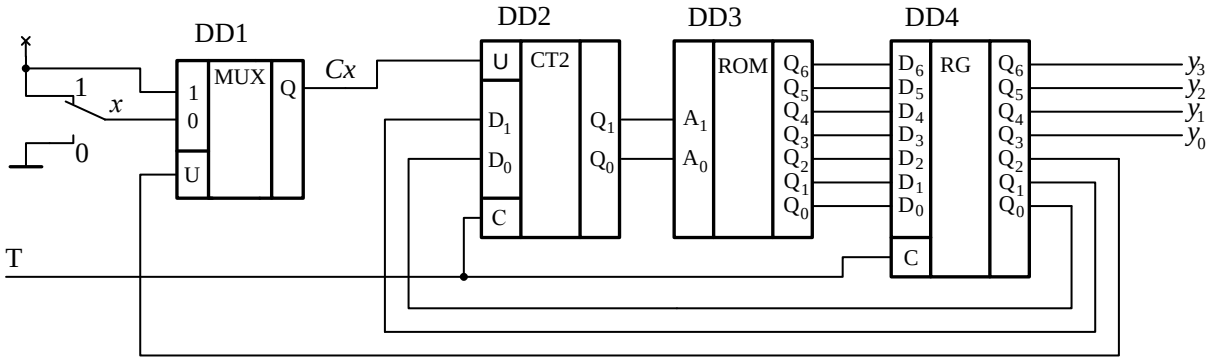

Рисунок 2.5.4 - Схема микропрограммного автомата для управления шаговым двигателем

Микропрограммные автоматы с естественной адресацией очень удобны в программировании и реализации. Некоторое усложнение электрической схемы окупается за счет экономии времени, необходимого на разработку микропрограммы для автомата. Поэтому микропрограммные автоматы данного типа являются наиболее распространенными на практике.

1.2 Микропрограммный автомат с естественной адресацией и разделенной микрокомандой

Сократить разрядность микрокоманды можно за счет разделения ее на микрокоманду присваивания и микрокоманду условного перехода. Рассмотрим особенности организации микропрограммного автомата с естественной адресацией и разделенной микрокомандой.

У автомата микрокоманды разделены на два типа: присваивания и проверки условия перехода. Для идентификации типа микрокоманды применяется специальный бит – **признак микрокоманды р**. Примем, что микрокоманде присваивания соответствует значение p=0, микрокоманде условного перехода – значение p = 1. На рисунке 2.6.1 показан формат используемых микрокоманд (а) и их графическое изображение (б).

Микрокоманда присваивания состоит из двух полей – **поля признака** микрокоманды p и **поля значения** выходной переменной Y. После выполнения команды присваивания выходная величина Y принимает заданное значение, а из следующей по порядку возрастания адреса ячейки (A_{n+1}), читается очередная микрокоманда.

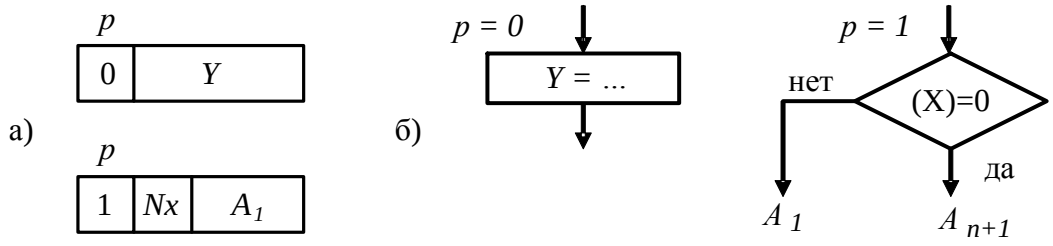

Рисунок 2.6.1 - МПА с естественной адресацией и разделенной микрокомандой: форматы микрокоманд (а) и их графическое изображение (б)

Микрокоманда проверки условия состоит из трех полей: **поля признака** микрокоманды p = 1, **поля номера** проверяемого входа X (оператор проверки условия) и **адресного поля** A_1. Значение Y в этой микрокоманде не определяется. Логика исполнения условной команды следующая: если на проверяемом входе (X) сигнал равен 0, то следующая микрокоманда читается из следующей по порядку возрастания адреса ячейки, т.е. ячейки с адресом A_{n+1}; иначе выполняется переход – микрокоманда читается из ячейки с адресом A_1 . Так как обе микрокоманды читаются из одного и того же ПЗУ, то разрядности микрокоманд присваивания и проверки условия должны быть равными.

Рассмотрим пример применения микропрограммного автомата в задаче управления шаговым двигателем. На рисунке 2.6.2 показана блок-схема микропрограммы, реализующая кодовую последовательность рассматриваемого примера работы МПА. Она представляет собой

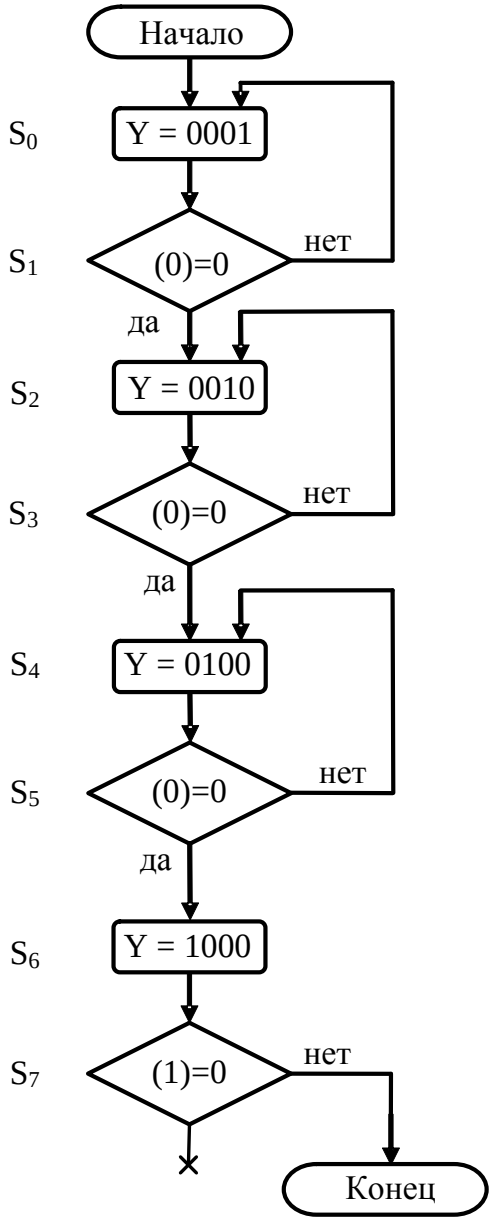

Рисунок 2.6.2 – Блок-схема микропрограммы работы автомата с естественной адресацией и разделенной микрокомандой

последовательность отдельных микрокоманд, представляющих операции присваивания или проверки условия. Команды программы обозначены символами S_i, где i – адрес ячейки памяти, где хранится команда. Основная особенность естественной адресации заключается в чтении следующей микрокоманды из ячейки, адрес которой на единицу больше предыдущей. Поэтому, когда входной сигнал (X) = 0 возникает проблема с завершением цикла работы, т.е. выполнением принудительного перехода к адресу A=0. Для разрешения указанной проблемы применен способ, основанный на использовании дополнительного входа (см. раздел 2.5). Этот дополнительный вход подключен к единице. В результате в случаях, когда необходимо выполнить принудительный переход к требуемой ячейке памяти,

независимо от состояния управляющего входа, следует задать условную команду с проверкой состояния со дополнительного входа. В рассматриваемом примере использован дополнительный вход №1, на который подана единица (т.е. вход №1 подключен к линии питания +5 В).

Зацикливание программы осуществляется в последней микрокоманде (S_7) микропрограммы путем проверки состояния дополнительного входа. Микропрограмма работы автомата приведена в таблице 2.6.1.

Как видно из таблицы 2.6.1, микрокоманда автомата содержит 5 бит. Микропрограмма содержит всего 8 команд. Для кодирования адреса

Таблица 2.6.1. Микропрограмма работы автомата с естественной адресацией и разделенной микрокомандой

№ ячейки	Данные в ПЗУ	
	P	*Y/ Nx* A_1
000	0	0001
001	1	0000
010	0	0010
011	1	0010
100	0	0100
101	1	0100
110	0	1000
111	1	1000

микрокоманды необходимо использовать 3 бита, для кодирования выхода используем 4 бита. Объем ПЗУ, необходимый для записи программы, составляет M=8*5 = 40 бит. В результате при укороченной команде общий объем ПЗУ увеличился.

На рисунке 2.6.3 показана принципиальная схема разработанного микропрограммного автомата с естественной адресацией и разделенной микрокомандой. Схема содержит традиционные для всех автоматов постоянное запоминающее устройство ROM (элемент DD4) для хранения кодов микропрограммы и пятиразрядный регистр микрокоманд RG (элемент DD5). На входе автомата установлен двухвходовой мультиплексор DD1. На вход № 0 мультиплексора подается входной управляющий сигнал *x*, значение которого можно изменять с помощью переключателя; на вход № 1 подается служебный сигнал с фиксированным значением, равным 1. На двоичном счетчике импульсов DD3 реализован счетчик микрокоманд.

Основное отличие в реализации рассматриваемого автомата от микропрограммного автомата с естественной адресацией (см. рисунок 2.5.3) заключается в схемной поддержке разряда *p* – признака микрокоманды. При *p* = 1 должна работать микрокоманда условного перехода. Для этого в цепи сигнала *Cx* установлен логический элемент И (элемент DD2). При значении

p = 1 сигнал Cx «проходит» через элемент И на вход управления U счетчика микрокоманд DD3, т.е. в автомате возможен переход по адресу, указанному в поле A_1. При значении p = 0 на выходе элемента И будет логический 0 и двоичный счетчик DD2 будет считать тактовые импульсы, поступающие на его вход С, осуществляя переход к следующей по порядку номеров микрокоманде.

Выходной сигнал микропрограммного автомата определен только в команде присваивания, т.е. при значении p = 0. Для того, чтобы выходная кодовая последовательность не прерывалась при выполнении микрокоманд условных переходов, в выходную цепь автомата включен четырехразрядный

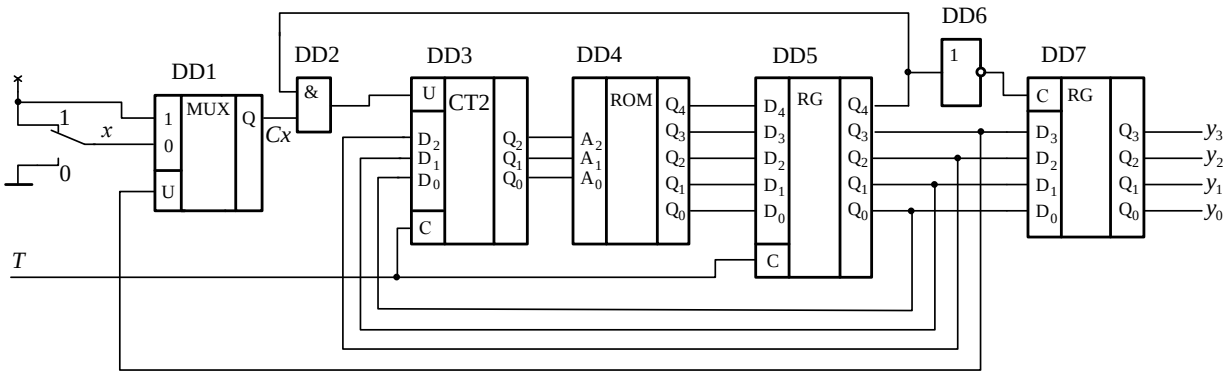

Рисунок 2.6.3 - Схема микропрограммного автомата с естественной адресацией и разделенной микрокомандой

регистр RG (элемент DD7). Сигнал разрешения записи информации в регистр (вход С) формируется инвертированием сигнала признака микрокоманды p = 0 элементом DD6. При этом предполагается, что активное состояние сигнала разрешения записи в регистр равно 1.

Из приведенной на рисунке 2.6.2 блок-схемы микропрограммы видно, что командами присваивания являются каждая вторая команда. Это значит, что частота переключения фаз управления шаговым двигателем в данной схеме в два раза меньше частоты следования тактовых импульсов Т.

Микропрограммные автоматы с естественной адресацией и разделенной микрокомандой довольно удобны для разработчиков систем управления. Небольшая длина микрокоманды позволяет создавать компактные схемы, а разделенные микрокоманды повышают гибкость разработки программного кода, не сильно увеличивая требуемый объем ПЗУ.

2.7 Микропрограммный автомат с усеченной естественной адресацией

В современной технике управления имеется большое число приложений, в которых вообще не нужны переключения режимов работы. Примерами могут служить устройства управления световой рекламой типа «бегущие огни», устройства задания программных траекторий следящих систем,

устройства для задания определенной циклической работы объекта управления. В таких приложениях могут быть применены микропрограммные автоматы, не имеющие входного сигнала *x*. На программном уровне отсутствие входного сигнала приводит к отсутствию микрокоманды условного перехода. За счет этого разрядность используемой микрокоманды может быть снижена (усечена). Такие автоматы получили название **микропрограммный автомат с усеченной естественной адресацией.**

Отсутствие микрокоманд условного перехода приводит к тому, что в микропрограммном автомате используется только естественный ход выполнения микрокоманд. Схема вычисления адреса следующей микрокоманды строится на базе счетчика импульсов, разрядность j которого рассчитывается по количеству k микрокоманд программы по формуле:

$$2^{j} \geq k.$$

На рисунке 2.7.1 показана реализация схемы вычисления адреса следующей микрокоманды в виде счетчика микрокоманд. На счетный

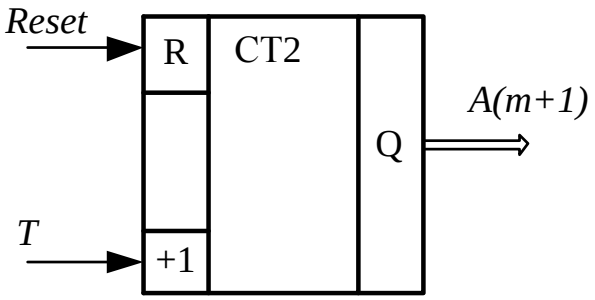

Рисунок 2.7.1 – Реализация схемы вычисления адреса следующей микрокоманды на двоичном счетчике импульсов

вход +1 счетчика подаются тактовые импульсы. С приходом каждого тактового импульса число в счетчике увеличивается на единицу, и в виде двоичного кода поступают на выходы *A(m+1)* счетчика.

Счетчик микрокоманд имеет дополнительный вход R на который подается сигнал Reset. Этот сигнал позволяет обнулить содержимое счетчика и за счет этого зациклить выполнение микропрограммы.

На рисунке 2.7.2 показаны: формат используемой микрокоманды а) и ее графическое представление б) в блок-схеме микропрограммы. Микрокоманда автомата состоит из двух полей: однобитового поля управления сбросом счетчика R и поля значения выходного сигнала *Y*. Фактически длина микрокоманды определяется только разрядностью выходного сигнала *y* автомата. Это обстоятельство позволяет применять в автоматах многоразрядные коды для удобного представления выходной переменной.

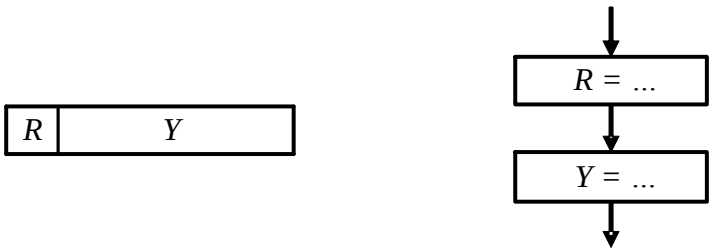

Рисунок 2.7.2 – Формат микрокоманды автомата с усеченной естественной адресацией а) и ее графическое представление б)

Микрокоманда автомата выполняет за один такт две операции присваивания – операцию задания значения сигнала R и операцию задания значения переменной *Y*. Поэтому графически такую микрокоманду на блок-схеме целесообразно изображать двумя операциями присваивания, однако относить к одной микрокоманде.

Для иллюстрации сказанного, рассмотрим простой пример. Предположим, что задача микропрограммного автомата – формировать фазные сигналы для задания вращения ротора шагового двигателя с постоянной скоростью. Внешнее управление вращением ротора отсутствует, останов вращения осуществляется простым выключением электропитания устройства. На рисунке 2.7.3 показана функциональная схема такого устройства. Тактовые импульсы с требуемой частотой следования формируются генератором Г. Выходные сигналы *Y* микропрограммного автомата усиливаются по напряжению и мощности блоком усилителей БУс и подаются на обмотки шагового двигателя. Предполагается, что используется четырехфазный шаговый двигатель с активным ротором. Задача микропрограммного автомата - сформировать сигналы управления последовательностью включения обмоток шагового двигателя. Для простоты будем считать, что необходимо циклически включать напряжение на обмотках двигателя, или формировать на выходе автомата циклическую кодовую последовательность *Y* = 0001, 0010, 0100, 1000, 0001, ...

На рисунке 2.7.4 показана блок-схема микропрограммы работы автомата с усеченной естественной адресацией. Она содержит четыре микрокоманды. В микрокомандах S_0 – S_2 сигнал сброса счетчика не активен, т.е. *R* = 0. В микрокоманде S_3 сигнал *R* = 1; исполнение этой микрокоманды приведет к обнулению счетчика микрокоманд и следующая исполняемая микрокоманда будет прочитана из ячейки 0. Таким образом, установка значения *R* = 1 приводит к зацикливанию микропрограммы.

Следует напомнить, что в данном частном случае число микрокоманд автомата получилось равным 4, что позволяет применить двухразрядный счетчик импульсов и не вводить в микрокоманду дополнительный разряд *R*. Зацикливание микропрограммы в этом случае произойдет естественным образом при переполнении двоичного счетчика (см. раздел 2.5).

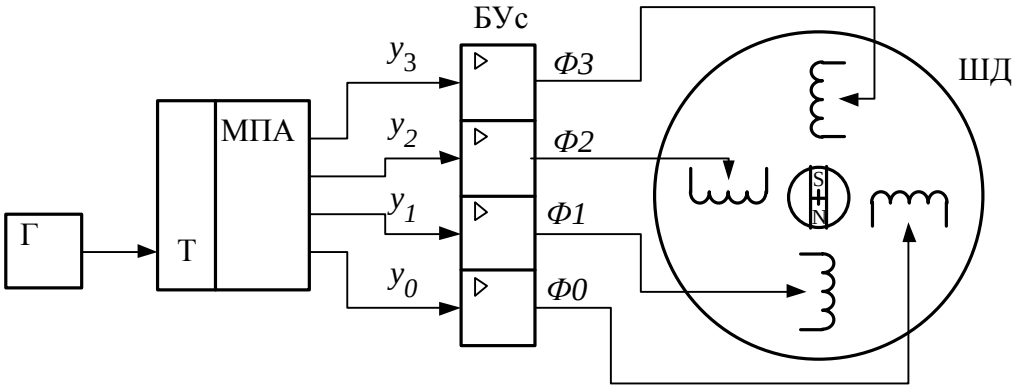

Рисунок 2.7.3 – Функциональная схема устройства управления шаговым двигателем на основе микропрограммного автомата с усеченной естественной адресацией

На рисунке 2.7.5 показана схема микропрограммного автомата с усеченной естественной адресацией. Схема содержит три элемента - ПЗУ для хранения микропрограммы (элемент DD2), регистр микрокоманд RG (элемент DD3) и счетчик микрокоманд СТ2 (элемент DD1). Так как количество микрокоманд в микропрограмме равно 4, то в схеме применена двухразрядная шина адреса следующей микрокоманды. Соответственно, счетчик микрокоманд будет двухразрядным и ПЗУ имеет двухразрядный адресный вход.

Микрокоманда автомата содержит 5 разрядов - однобитовое поле *R* и четырех битовое поле *Y*, причем поле *R* занимает старший бит микрокоманды. Поэтому старший бит выхода регистра RG соединен с входом R двоичного счетчика импульсов, образуя цепь сброса счетчика в ноль. Младшие четыре бита выхода регистра микрокоманд образуют выход микропрограммного автомата. Тактовые импульсы *T* подаются на счетный вход счетчика микрокоманд и вход синхронизации регистра микрокоманд.

Разрядность регистра микрокоманд равна 5, а требуемый объем ПЗУ получается равным M = 4*5 = 20 бит.

В таблице 2.7.1 приведены коды микропрограммы работы автомата. Таблица построена в соответствии с принятым форматом микрокоманды, блок-схемой алгоритма и принятым условием, что номер микрокоманды совпадает с адресом хранения микрокоманды в памяти ПЗУ.

Конечно, современная техника не ограничивается рассмотренными типами микропрограммных автоматов. Разработчики цифровой техники всегда стараются сделать устройство, наиболее эффективно решающее поставленную задачу. Это стремление разработчиков к усовершенствованию аппаратуры отражается в разнообразии применяемых инженерных решений и в постоянном увеличении количества типов микропрограммных автоматов, [15].

Таблица 2.7.1 Микропрограмма работы автомата с усеченной естественной адресацией

Адрес микрокоманды	Микрокоманды	
	R	*Y*
00	00001	
01	00010	
10	00100	
11	11000	

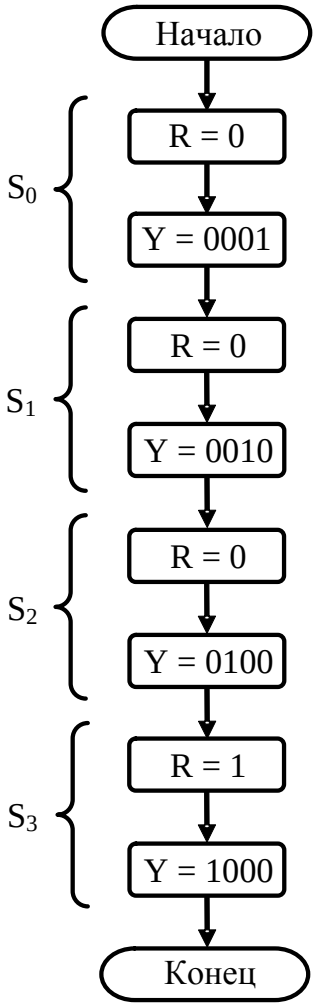

Рисунок 2.7.4 – Блок-схема микропрограммы работы автомата с усеченной естественной адресацией

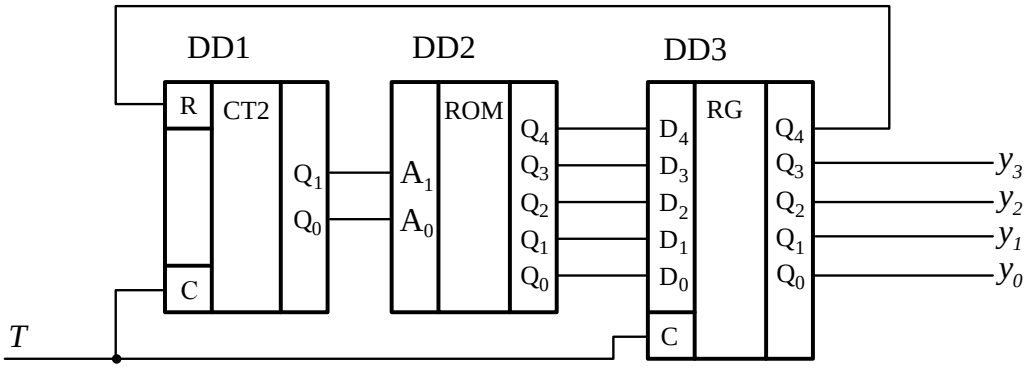

Рисунок 2.7.5 – Схема микропрограммного автомата с усеченной естественной адресацией для управления шаговым двигателем

Подводя итоги рассмотрения свойств и применения микропрограммных автоматов можно сделать следующие общие выводы:

- микропрограммный автомат не вычисляет значения переменных, являясь устройством с табличной организацией;

- микропрограммный автомат функционирует по шагам, темп его работы определяется частотой поступающих тактовых импульсов;

- в минимальной конфигурации микропрограммный автомат должен иметь постоянную память для хранения микропрограммы и оперативную память (регистр) для хранения выбранной микрокоманды.

Стремление снизить затраты времени на разработку конкретных устройств привело к необходимости переложить задачу разработки микропрограммы работы автомата на компьютер с последующей реализацией в виде микросхемы с программируемой архитектурой. В результате, в настоящее время разработаны формальные процедуры синтеза микропрограммных автоматов, микросхемы с программируемой архитектурой (классические ПЛИС и современные FPGA) и программное обеспечение для разработки устройств на их основе [6, 16, 17].

2.8 Микропрограммное управление ходом вычислений. Архитектура цифрового сигнального процессора

Одно из самых распространенных применений микропрограммных автоматов – управление ходом вычислений, выполняемых регистровым арифметико-логическим устройством (АЛУ) вычислительной машины. В зависимости от уровня программирования такие гибридные структуры относятся либо к классу микроконтроллеров, либо к классу цифровых сигнальных процессоров.

Микроконтроллеры будут рассмотрены ниже, а сейчас разберем особенности применения цифровых сигнальных процессоров.

Цифровым сигнальным процессором называют вычислительную структуру, содержащую арифметико-логическое устройство и управляющий микропрограммный автомат, программу работы которого разрабатывает

пользователь. Как будет показано ниже, процесс программирования такого устройства весьма трудоемок, требует высокой квалификации программиста, но позволяет написать очень эффективную в вычислительном смысле программу. Поэтому цифровые сигнальные процессоры применяются в задачах, где требуется обеспечить выполнение больших объемов математических вычислений за очень короткое время, часто в режиме реального времени.

Для выполнения различных сервисных вычислительных функций, связанных с обслуживанием человека, эти процессоры практически не используются. Главное их применение – математическая обработка информационных сигналов, поступающих от высокоскоростных датчиков. Примерами могут служить задачи обработки речи, обработка изображений в системах технического зрения, вычисления спектров сигналов в реальном времени и тому подобные, [5, 19].

Особенности программирования и возможности оптимизации программы сигнального процессора мы рассмотрим на примере кристалла К1813ВЕ1. Это один из первых цифровых сигнальных процессоров, выпускался в СССР в девяностых годах XX века, имеет зарубежный аналог. В настоящее время он конечно сильно устарел, уже снят с производства, а в мире появились значительно более мощные сигнальные процессоры. Однако для учебных целей он хорошо подходит, благодаря простоте своей архитектуры.

На рисунке 2.8.1 показана функциональная схема цифрового процессора К1813ВЕ1. В процессоре можно выделить три крупные части: управляющий микропрограммный автомат, цифровую часть и аналоговую часть. Управляющий микропрограммный автомат состоит из постоянного запоминающего устройства EEPROM с регистром микрокоманд (на схеме не показан), счетчика микрокоманд CT2 и генератора G тактовых импульсов. Генератор G имеет внешние выводы CR1 и CR2 для подключения кварцевого резонатора, определяющего частоту следования тактовых импульсов. По схеме видно, что управляющий автомат относится к микропрограммным автоматам с усеченной естественной адресацией (см. раздел 2.7). Выходной код *Y* управляющего автомата делится на поля: ACOP, AA, AB, K и DCOP. Бинарные значения сигналов этих полей определяют работу остальных частей сигнального процессора. Разрядность микрокоманды 24 бита.

Цифровая часть состоит из двухпортового оперативного запоминающего устройства RAM (ОЗУ), аппаратного сдвигателя SHL и арифметико-логического устройства ALU. Два шестиразрядных числа в полях AA и AB микрокоманды определяют адреса двух ячеек RAM, данные из которых поступают на выходы A и B оперативной памяти, соответственно. Число с выхода A проходит аппаратный сдвигатель SHL, управление сдвигом которого определяется четырехразрядным кодом в поле K микрокоманды. Число с выхода B оперативной памяти и число с выхода сдвигателя в качестве операндов поступают на входы арифметико-логического устройства

ALU. Операция ALU задается трехразрядным кодом в поле DCOP микрокоманды. Результат работы ALU записывается в оперативную память по адресу операнда B. По результату выполнения арифметических операций ALU модифицируется флаг *OF* переполнения разрядной сетки. Все указанные действия выполняются в цифровой части за один такт работы микропрограммного автомата.

Аналоговая часть сигнального процессора содержит входной четырехканальный аналоговый переключатель SW, буферный усилитель AMP с функцией выборки/хранения, компаратор CMP, цифро-аналоговый преобразователь DAC и выходной восьмиканальный переключатель SW. Управление работой аналоговой части задается кодом в поле ACOP микрокоманды, который декодируется преобразователем CU. Для реализации функции выборки/хранения операционный усилитель имеет внешние выводы C1 и C2, к которым подключается внешний конденсатор требуемой емкости. Цифро-аналоговый преобразователь DAC имеет вход U_{ref} для подключения внешнего источника опорного напряжения.

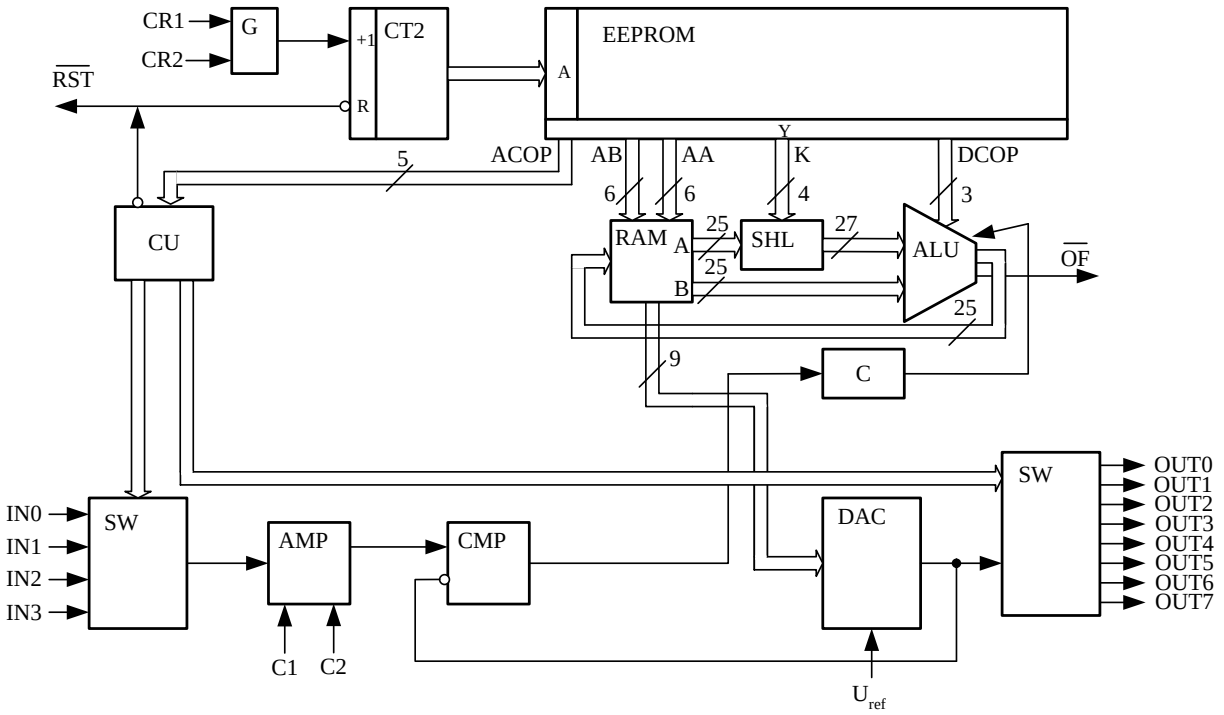

Рисунок 2.8.1 – Архитектура цифрового сигнального процессора К1813ВЕ1

На вход цифро-аналогового преобразователя подаются 9 старших разрядов 40-ой ячейки памяти ОЗУ. Это обеспечивает информационную связь между цифровой и аналоговой частями сигнального процессора.

На элементах аналоговой части сигнального процессора реализуются функции аналого-цифрового и цифро-аналогового преобразований. Следует отметить, что наличие аналоговой части не характерно для современных сигнальных процессоров. Сигнальные процессоры в основном предназначены для выполнения математических операций типа умножения с накоплением

результата в реальном времени. Для них характерно представление данных с повышенной разрешающей способностью, одновременная выборка из оперативной памяти двух операндов, наличие циклических пересылок данных. Все указанные действия определяются микрокомандой управляющего автомата. Поэтому пользователь, программируя работу автомата, имеет возможность управлять **одновременной работой** нескольких блоков цифрового сигнального процессора с целью получения максимальной производительности при решении конкретной задачи цифровой обработки информации.

На рисунке 2.8.2 показана типовая последовательность шагов работы цифрового сигнального процессора в задаче обработки сигнала. Рассматриваемая последовательность шагов ориентирована на процессор К1813ВЕ1, т.е. на обработку аналогового сигнала. Алгоритм построен по типовой циклической схеме преобразования информации - ВВОД - ПРЕОБРАЗОВАНИЕ - ВЫВОД. Операция ввода аналогового сигнала требует задать номер входа, к которому подключен источник сигнала. Далее мгновенное значение сигнала должно быть выбрано и запомнено во входном буферном усилителе (модуль 1). В модуле 2 значение сигнала программно-аппаратным способом преобразуется в число, которое сохраняется в 40-ой ячейке ОЗУ. Теперь цифровой эквивалент аналогового сигнала может быть подвергнут цифровой обработке, что и выполняется в модуле 3. В модуле 4 выполняется обратное преобразование результата обработки в напряжение. Для этой цели используется аппаратный цифро-аналоговый преобразователь DAC (ЦАП). В пятом модуле напряжение подается на вывод процессора, к которому подключен потребитель сигнала. После вывода результата программа работы цифрового сигнального процессора зацикливается.

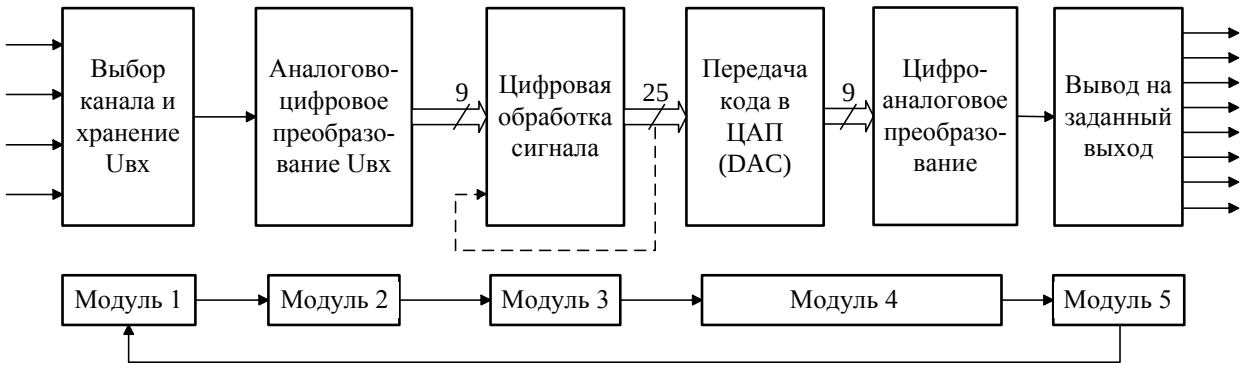

Рисунок 2.8.2 - Типовая последовательность модулей алгоритма цифровой обработки сигнала

Общая характеристика сигнального процессора. КМ1813ВЕ1-однокристальная ЭВМ с встроенным перепрограммируемым ПЗУ программы с УФ-стиранием и аналоговым устройством ввода-вывода. Кристалл выполнен по n-МОП технологии и помещен в металлокерамический 28-выводный корпус.

Основные технические данные сигнального процессора:
- напряжение питания +5В, -5В;
- потребляемый ток +50мА, -170мА;
- тактовая частота 1 - 6 мГц;
- напряжение опорного источника +1,5 ± 0,5В;
- Максимальное входное напряжение (аналоговый сигнал) ± 2В;
- максимальное выходное напряжение (аналоговый сигнал) ± 2В;
- разрядность АЦП и ЦАП - 9 (8+знак);
- число каналов - аналогового ввода – 4, аналогового выхода – 8;
- разрядность АЛУ 25 бит;
- емкость встроенного ОЗУ - 40х25 бит;
- память констант - 16х25 бит;
- аппаратно реализуемые максимальные коэффициенты - от 2^{-13} до 2^{2};
- формат чисел - фиксированная запятая;
- управление - МПА с усеченной естественной адресацией;
- память микропрограммы – 198 слов;
- число циклов перезаписи ПЗУ – 20.

Примером использования сигнального процессора может служить устройство кодирования/декодирования речевого сигнала для повышения качества связи (исправление искажений сигнала в линии связи за счет избыточного кодирования). Мысленно можно представить, что к аналоговому входу №1 подключен микрофон, а к аналоговому выходу №1 подключен радиопередатчик; к аналоговому входу №2 подключен выход радиоприемника, а к аналоговому выходу №2 подключен наушник. Речевой сигнал в реальном времени преобразуется в цифровой, кодируется и после обратного преобразования в напряжение подается на передатчик. Мы можем говорить в микрофон и речь в закодированном виде будет передаваться по радиоканалу нашему абоненту. Ответный речевой сигнал в закодированном виде поступает в сигнальный процессор с выхода радиоприемника, декодируется и после обратного преобразования в напряжение подается на наушники. Мы можем общаться с нашим абонентом, причем качество связи будет значительно выше, чем при использовании приемника и передатчика без кодирования.

2.9 Управление цифровой частью процессора. Реализация операции умножения на постоянный коэффициент

Цифровая часть сигнального процессора К1813ВЕ1 состоит из трех управляемых блоков: двухпортового ОЗУ, аппаратного сдвигателя SHL и арифметико-логического устройства ALU.

Двухпортовое ОЗУ требует задания одновременно адресов двух операндов А и В. Коды адресов в целочисленном формате указываются в полях АА и АВ микрокоманды. При этом на два выхода ОЗУ выдается

содержимое двух указанных ячеек памяти, а входной код записывается в ячейку памяти с адресом B.

Таким образом, каждый порт ОЗУ управляется независимо 6 разрядным кодом адреса. Порт A используется только для чтения. Порт B – для чтения операнда и записи результата из ALU. Объем оперативной памяти - 40 ячеек. Т.к. 2^6=64, то имеются 24 «свободных» адресов ячеек памяти. Эти адреса использованы для хранения констант, заданных производителем кристалла.

При чтении переменной или константы через порт A, число проходит аппаратный сдвигатель SHL, который за такт сдвигает число на заданное число разрядов K. Операция сдвига эквивалентна операции умножения на коэффициент 2^K

В таблице 2.9.1 приведены условные обозначения и коды операции сдвига числа, выполняемых сдвигателем SHL.

Таблица 2.9.1 Коды операций, выполняемых сдвигателем SHL

Обозначение операции	Код операции	Коэффициент умножения
L02	1101	2^2
L01	1110	2^1
R00	1111	2^0
R01	0000	2^{-1}
R02	0001	2^{-2}
…	…	…
R09	1000	2^{-9}
R10	1001	2^{-10}
R11	1010	2^{-11}
R12	1011	2^{-12}
R13	1100	2^{-13}

Операнды $A*2^K$ и B поступают на арифметико-логическое устройство для выполнения математической операции. Принято, что в рассматриваемом сигнальном процессоре числа представляются в формате с фиксированной запятой из диапазона $-1 \leq x \leq 1$. Минимальное приращение переменной определяется ценой младшего разряда числа и составляет 2^{-25}.

Задание конкретной операции определяется кодом в поле DCOP микрокоманды. Трехразрядное поле DCOP позволяет задать восемь различных команд. В таблице 2.9.2 приведены все 8 команд арифметико-логического устройства.

В поле управления аналоговой частью ACOP можно задать признак условной команды **CND(N)** по биту N цифро-аналогового преобразователя. Если этот признак установлен, т.е. в поле ACOP указана условная операция,

то в цифровой части эта условная операция реализуется. Например, в поле АСОР задана условная команда по биту 3 цифро-аналогового преобразователя - команда CND (3), а в цифровой части задана команда сложения ADD. Тогда в цифровой части реализуется условная команда:

if $(DAC_3) = 1$ then $(B):=(B) + (A)*2^k$
else $(B):=(B)$

Применение условных команд позволяет расширить возможности программирования.

Таблица 2.9.2 Команды арифметико-логического устройства

Операция	Код	Мнемоника	Действие
Исключающее ИЛИ	000	XOR	$(B):=(B)$ xor $(A)*2^k$
Логическое И	001	AND	$(B):=(B)$ & $(A)*2^k$
Ограничение	010	LIM	If $(A)\geq 0$ then $(B):=+1$ else $(B):= -1$
Абсолютное значение	011	ABC	$(B):=\|(A)*2^k\|$
Сложение с модулем	100	ABA	$(B):=(B)+\|(A)*2^k\|$
Вычитание	101	SUB	$(B):=(B)-(A)*2^k$
Сложение	110	ADD	$(B):=(B)+(A)*2^k$
Пересылка	111	LDA	$(B):=(A)*2^k$

Рассмотрим пример программирования цифровой части процессора. Типовая операция цифровой части - умножение на константу, т.е. вычисление по формуле

$$y = c* x.$$

Пусть, для примера, значение константы c равно 0,54. Так как операция умножения в ALU отсутствует, но имеется аппаратный сдвигатель, то для выполнения операции умножения разложим константу c по отрицательным степеням 2. В двоичном коде (формат чисел с фиксированной запятой) константа будет представлена рядом $c = 0{,}1000101 = 1*2^{-1} + 1*2^{-5} + 1*2^{-7}$. Далее предположим, что Ах и Ау - адреса ячеек ОЗУ, где хранятся переменные x и y соответственно, тогда можем написать следующий текст программы работы процессора:

AB	A A	K	DCOP	Коментарий
Ay	Ax	R01	LDA	Пересылка $y=2^{-1} * x$
Ay	Ax	R05	ADD	Сложение с накоплением $y = y + 2^{-5} * x$
Ay	Ax	R07	ADD	Сложение с накоплением $y = y + 2^{-7} * x$

При составлении программы использован формат микрокоманды сигнального процессора: в полях AA и AB задаем адреса хранения операндов в ОЗУ, в поле K задаем код операции аппаратного сдвигателя, в поле DCOP задаем код операции арифметико-логического устройства. В результате три команды микропрограммы позволяют реализовать умножение 25-битовой переменной x на константу 0,54.

2.10 Управление аналоговой частью процессора. Реализация аналого-цифрового преобразования

Управление аналоговой частью выполняется заданием кода операции в поле ACOP микрокоманды. Пятиразрядное поле ACOP позволяет задать 32 различные команды.

Команда **IN(Q)**, Q = 0 … 3 – ввод аналогового сигнала с входа №Q и запоминание значения напряжения на операционном усилителе. По этой команде входной аналоговый переключатель подключает к входу операционного усилителя выбранный вход микросхемы. Длительность выполнения операции – один машинный такт. За это время заряжается конденсатор, подключенный к внешним выводам операционного усилителя. Величина заряда на конденсаторе определяется его емкостью и тактовой частотой работы процессора. Из-за переходного процесса заряда конденсатора обычно требуется использования последовательно нескольких (до восьми) микрокоманд IN(Q). В результате выполнения этих команд на выходе операционного усилителя устанавливается значение входного напряжения, которое сохраняется примерно неизменным в течении последующего преобразования его в число.

Аналого-цифровое преобразование напряжения в число выполняется под управлением программы. Преобразователь содержит компаратор CMP и цифроаналоговый преобразователь DAC. Процесс преобразования напряжения в число заключается в подборе такого кода в 40-й ячейке оперативного запоминающего устройства, преобразование которого в напряжение (на выходе DAC) максимально соответствует входному напряжению. Алгоритм аналого-цифрового преобразования – поразрядное уравновешивание напряжений, [1]. Для этого используются команды **CVT(N)** – уравновешивание разряда N числа на входе DAC. Так как число на входе DAC состоит из 9 разрядов, то для выполнения аналого-цифрового преобразования необходимо последовательно выполнить девять команд CND(N), начиная с старшего (восьмого по номеру) разряда: CVT(8), CVT(7), … , CVT(1), CVT(0). В результате выполнения такой последовательности микрокоманд в 40-й ячейке памяти ОЗУ получается код числа, соответствующей преобразуемому напряжению.

Числовой эквивалент входного напряжения, хранящийся в ОЗУ, позволяет применить к входному сигналу методы цифровой обработки. Для

преобразования результата обработки в напряжение используется цифро-аналоговый преобразователь. Результат вычислений просто пересылается в 40-ю ячейку памяти ОЗУ.

Команды **OUT(P),** P = 0 … 7 управляют выходным аналоговым переключателем и позволяют вывести выходное напряжение DAC на выход с требуемым номером P.

Кроме рассмотренных команд, в аналоговой части имеются коды служебных команд. Команды **CND(N),** N = 0 … 8, задают выполнение условной команды в цифровой части по биту N цифро-аналогового преобразователя. Выполнение этих команд рассмотрено в разделе 2.9.

Команда **NOP** задает пропуск выполнения команды в аналоговой части. Команда **END** определяет конец программы. По этой команде выполняется сброс счетчика микрокоманд управляющего автомата в ноль. Выполнение микропрограммы зацикливается.

2.11 Микропрограммная реализация регистрового способа вычислений. Распараллеливание вычислений

В качестве примера использования сигнального процессора, рассмотрим реализацию вычисления сглаженного значения сигнала методом скользящего среднего. Реализация такой задачи аппаратным способом была рассмотрена в разделе 1.11.1.

На рисунке 2.11.1 показана схема регистрового вычисления сглаженного значения сигнала метолом скользящего среднего. Для сглаживания сигнала используется вычисление среднего значения по шести последним отсчетам, т.е.

$$y(m) = [x1(m)+x2(m)+x3(m)+x4(m)+x5(m)+x6(m)]/\,6,$$

где: *x1(m), ..., x6(m)* – шесть последовательных отсчетов входного сигнала.

Для хранения последовательности отсчетов входного сигнала воспользуемся ячейками оперативного запоминающего устройства. Для

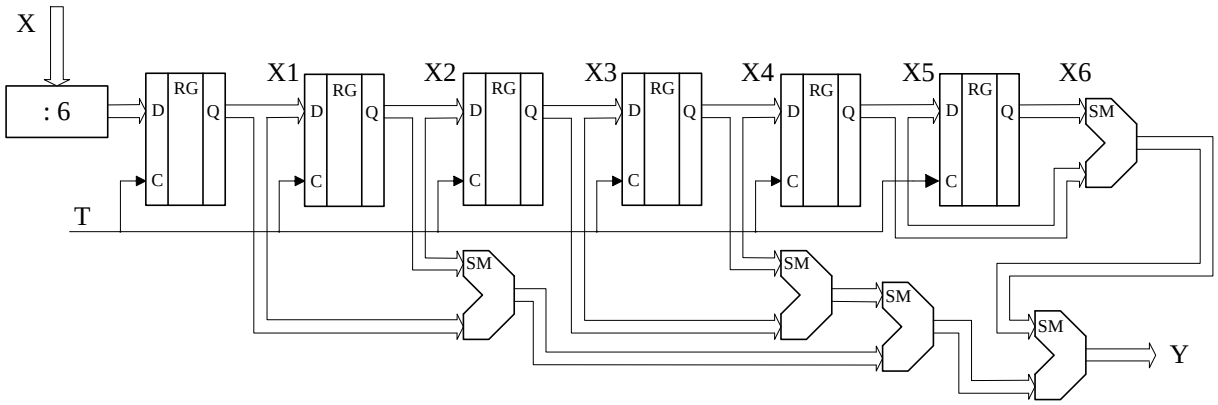

Рисунок 2.11.1 Схема вычисления сглаженного значения сигнала

распределения памяти используем простое правило – номер ячейки памяти равен номеру отсчета сигнала, т.е. отсчет *X1* поместим в ячейку 1, отсчет *X2* в ячейку 2 и т.д.

Ниже приведен текст программы работы цифрового сигнального процессора, реализующей вычислительную схему, показанную на рисунке 2.11.1. Текст программы написан с использованием мнемокода рассматриваемого процессора и сопровождается комментариями. Для удобства в заголовке программы приведен формат микрокоманды управляющего автомата. Программа работы сигнального процессора:

```
№    ACOP AB  AA   K   DCOP ; Комментарий
0    IN(1)  0   0   R00  LDA  ; Ввод аналогового сигнала
1    IN(1)  0   0   R00  LDA  ; Повторение команды для
2    IN(1)  0   0   R00  LDA  ;   увеличения времени заряда
3    IN(1)  0   0   R00  LDA  ;   конденсатора
4    CVT8   0   0   R00  LDA  ; Аналого-цифровое
5    CVT7   0   0   R00  LDA  ;   преобразование входного
6    CVT6   0   0   R00  LDA  ;   сигнала
7    CVT5   0   0   R00  LDA
8    CVT4   0   0   R00  LDA
9    CVT3   0   0   R00  LDA
10   CVT2   0   0   R00  LDA
11   CVT1   0   0   R00  LDA
12   CVT0   0   0   R00  LDA
13   NOP    1   39  R03  ADD  ; Деление цифрового значения
14   NOP    1   39  R05  ADD  ;   сигнала на 6
15   NOP    1   39  R07  ADD
16   NOP    1   39  R09  ADD
17   NOP    1   39  R11  ADD
18   NOP    1   39  R13  ADD
19   NOP    39  1   R00  LDA  ; Вычисление суммы значений
20   NOP    39  2   R00  ADD  ;   сигналов
21   NOP    39  3   R00  ADD
22   NOP    39  4   R00  ADD
23   NOP    39  5   R00  ADD
24   NOP    39  6   R00  ADD
25   OUT(1) 0   0   R00  LDA  ; Вывод результата на выход 1
26   NOP    6   5   R00  LDA  ; Сдвиг данных в цепочке
27   NOP    5   4   R00  LDA  ;     регистров
28   NOP    4   3   R00  LDA
29   NOP    3   2   R00  LDA
30   NOP    2   1   R00  LDA
31   END    0   0   R00  LDA  ; Зацикливание программы
```

Программа вычисления сглаженного значения сигнала начинается с ввода очередного значения входного сигнала и его преобразования в число. Микрокоманды №0 - №3 управляют входным мультиплексором так, что напряжение с аналогового входа №1 микросхемы подается на вход внутреннего операционного усилителя. Микрокоманда ввода IN(1) повторена четыре раза для обеспечения завершения переходного процесса на выходе операционного усилителя. В цифровой части этих микрокоманд выполняется пустая команда – пересылка данных из ячейки ОЗУ №0 в ячейку №0 без сдвига данных. По существу, это реализация операции NOP в цифровой части процессора.

Начиная с микрокоманды №4 по микрокоманду №12 выполняются операции аналого-цифрового преобразования входного сигнала в цифровой код. Результат преобразования (т.е. девяти битовое число) формируется в ячейке №39 ОЗУ. Учитывая, что полученное численное значение сигнала занимает девять старших разрядов 25-разрядного числа в формате с фиксированной запятой, целесообразно сразу разделить его на 6. Деление на 6 выполняется с использованием аппаратного сдвигателя SHL в цифровой части процессора. Двоичный код числа 1/6 равен

$$1/6 = 0,0010101010101$$

с округлением до 2^{-14}, так как SHL сдвигает число вправо максимум на 13 двоичных разрядов. Микрокоманды №№ 13 – 18 реализуют операцию деления на 6 (точнее операцию умножения на 1/6). Результат операции формируется в ячейке №1 оперативного запоминающего устройства. В аналоговой части этих микрокоманд выполняется операция NOP.

Микрокоманды с №19 по №24 реализуют вычисление суммы данных всех регистров вычислительной схемы. Результат суммирования формируется в ячейке №39 ОЗУ. Так как 9 старших бит ячейки 39 физически соединены с входом цифро-аналогового преобразователя, то результат суммирования срезу преобразуется в величину напряжения на выходе DAC.

Микрокоманда №25 выполняет вывод полученного результата на аналоговый выход №1 микросхемы. Следующие за ней микрокоманды №№ 26 – 30 реализуют операцию обновления информации в цепочке регистров вычислительной схемы. Это обновление выполняется операцией пересылки данных между ячейками ОЗУ. Микрокоманда №30 пересылает данные *X1* в *X2*, поэтому в *X1* теперь можно записывать новое значение входной переменной. Зацикливание программы работы сигнального процессора осуществляется микрокомандой №31.

Приведенный пример вычисления среднего значения сигнала показывает, что длительность выполнения одного цикла программы определяется временем выполнения 32 команд и равна 32 тактам работы генератора тактовых импульсов управляющего автомата. Например, при тактовой частоте генератора, равной 5 МГц (длительность такта 0,2 мкс.),

время выполнения одного цикла программы составляет 6,4 мкс. Это значит, что разработанная программа позволяет выполнять сглаживание сигналов, частотный спектр которых простирается до нескольких десятков килогерц. Уменьшить длительность выполнения одного цикла программы можно двумя путями:

- увеличением тактовой частоты работы устройства;
- оптимизацией программы.

В нашем примере тактовую частоту генератора можно увеличить до 6 МГц, которая является предельной частотой работы микросхемы К1813ВЕ1. Этот путь позволит уменьшить длительность цикла программы до 5,3 мкс или на 17% от первоначального значения.

Оптимизация программы может быть выполнена за счет распараллеливания операций аналоговой и цифровой частей процессора. Действительно, регистровый способ вычислений предполагает, что часть необходимой информации читается из памяти данных, которая заполняется на предыдущих циклах работы программы. Ниже приводится текст оптимизированной программы вычисления скользящего среднего.

```
№      ACOP AB  AA   K   DCOP ; Комментарий

0      IN(1)  1   39   R11 ADD   ; Ввод аналогового сигнала
1      IN(1)  1   39   R13 ADD   ; Окончание деления на 6
2      IN(1)  10  1    R00 ADD   ; Вычисление суммы значений
3      IN(1)  39  10   R00 LDA   ; Пересылка из 10 в 39
4      OUT1   10  2    R00 LDA   ; Вывод результата на выход 1
5      CVT8   10  3    R00 ADD   ; Аналого-цифровое
6      CVT7   10  4    R00 ADD   ;    преобразование входного
7      CVT6   10   5   R00  ADD ;   сигнала
8      CVT5   10   6   R00  ADD ; Расчет суммы значений
9      CVT4    6   5   R00  LDA ; Сдвиг данных в цепочке
10     CVT3    5   4   R00  LDA ;       регистров
11     CVT2    4   3   R00  LDA
12     CVT1    3   2   R00  LDA
13     CVT0    2   1   R00  LDA
14     NOP     1   39  R03  ADD; Деление цифрового значения
15     NOP     1   39  R05  ADD;       сигнала на 6
16     NOP     1   39  R07  ADD
17     NOP     1   39  R09  ADD
18     END     0    0  R00  LDA ;Зацикливание программы
```

Основная идея оптимизации программы заключается в возможности вычисления суммы значений переменных *X2 – X6* и сдвига информации в регистрах в то время, когда осуществляется ввод и преобразование нового значения переменной *X1*. Эта сумма вычисляется в дополнительной ячейке

№10 оперативной памяти. Когда преобразование нового значения *X1* завершается, вычисляется окончательное значение суммы всех переменных (микрокоманда №2) и вывод результата вычислений (микрокоманда №4).

В результате длина программы сократилась до 19 микрокоманд, а длительность выполнения цикла - до 19 тактов работы генератора тактовых импульсов. Время выполнения цикла программы сократилась до 3,8 мкс при тактовой частоте 5 МГц, и до 3,2 мкс при тактовой частоте 6 МГц. Таким образом, оптимизация программы работы цифрового сигнального процессора позволило снизить время выполнения цикла программы на 41%.

Рассмотренный пример программирования цифрового сигнального процессора показал, что сигнальные процессоры позволяют реализовать весьма эффективные в вычислительном плане устройства управления и обработки информации. Однако им присущ недостаток – разработка эффективной микропрограммы требует очень хорошего знания особенностей работы процессора. Нам для написания микропрограммы в 19 строк пришлось разобраться в особенностях работы всех блоков машины. Кроме того, микропрограммирование приходится вести на специализированном языке, отражающем архитектуру сигнального процессора, что замедляет разработку устройства. В результате цифровые сигнальные процессоры, даже самые современные, применяются для решения задач, требующих математической обработки данных в условиях большого дефицита вычислительного времени.

2.12 Программное управление ходом вычислений. Архитектура микроконтроллера

Программная реализация вычислений на сегодняшний день является самой востребованной в цифровой технике управления и обработки информации. Идея микропрограммного исполнения программных команд, предложенная М. Уилксом и Д. Стринджером, заключается в реализации любой машинной команды пользователя в виде последовательности исполняемых микрокоманд автомата управления. Такой подход позволил существенно упростить процесс программирования работы машин.

На рисунке 2.12.1 показана архитектура достаточно простых и распространенных микроконтроллеров семейства MCS-51. Этот микроконтроллер, разработанный инженерами фирмы Intel, оказался весьма удачным в плане удобства программирования и имеющегося набора аппаратных средств. В результате микроконтроллер в различных модификациях выпускается многими фирмами и до сегодняшнего дня пользуется популярностью у разработчиков систем управления.

Микроконтроллер представляет собой восьмибитовую вычислительную машину, основная информационная связь у которой выполняется через однобайтовую системную шину, связывающую между собой все устройства. Для хранения программы пользователя имеется специальная память

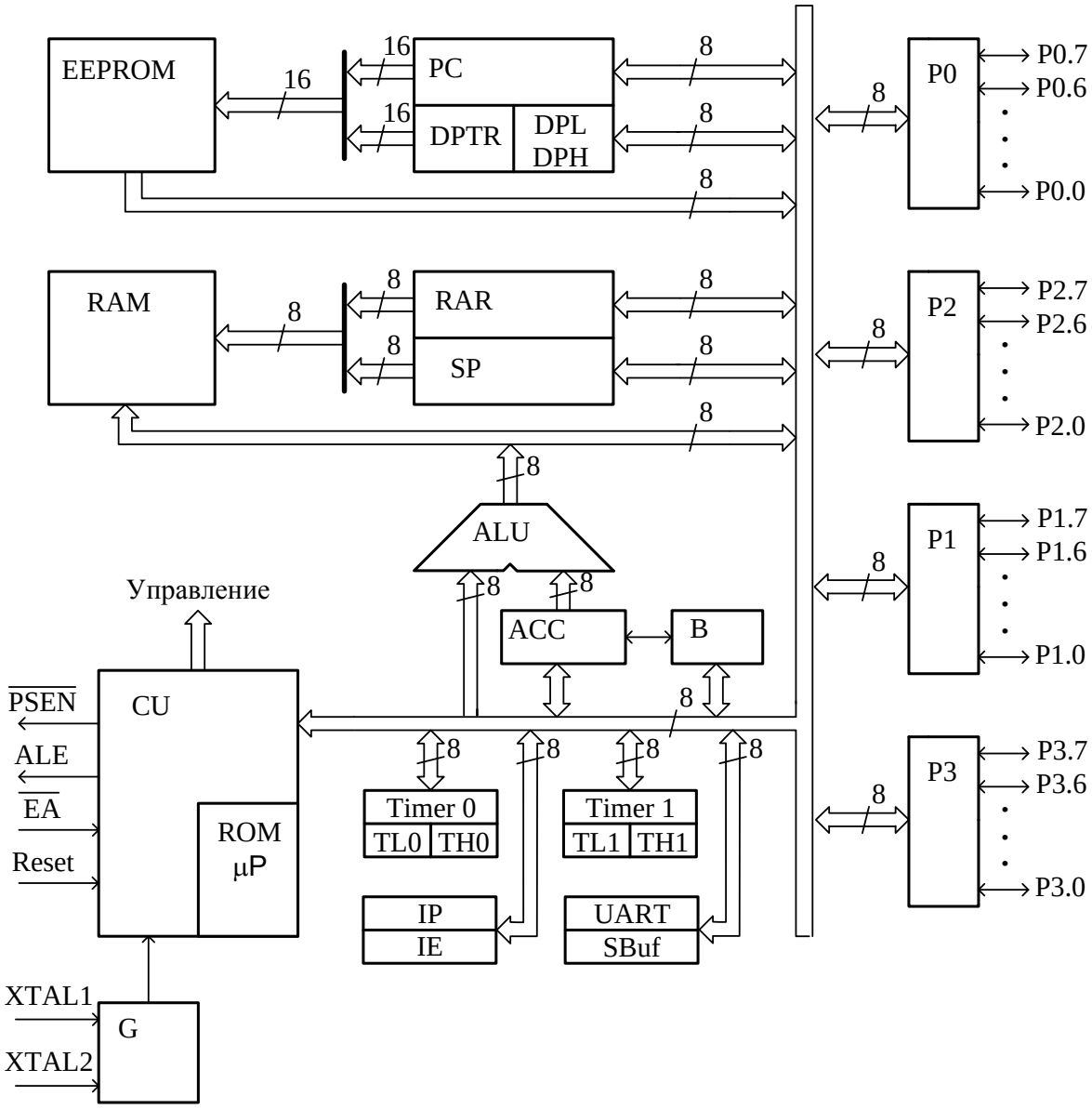

Рисунок 2.12.1 - Архитектура микроконтроллеров семейства MCS-51

программ типа EEPROM (или Flash). Исполнение памяти программ в виде ПЗУ, программируемого пользователем, определяется назначением микроконтроллера – он должен работать в оборудовании по неизменяемой программе.

Адресная шина памяти программ 16-разрядная. Это позволяет использовать память программ объемом до 64 КБ. Следует отметить, что для большинства программ управления хватает 8КБ памяти.

Оперативная память ROM предназначена только для временного хранения данных, используемых при выполнении вычислений. Адресная шина ОЗУ восьмибитовая. Поэтому в базовой архитектуре не может быть большее 256 ячеек оперативной памяти. Микроконтроллер Intel I8051 имел всего 128 ячеек оперативной памяти, но и этого количества обычно достаточно для решения задач управления.

Арифметико-логическое устройство ALU выполняет действия над двумя восьмибитовыми операндами. Операции ALU выполняются с использованием специального регистра, называемого **аккумулятором** ACC. Один из операндов выполняемой операции берется из аккумулятора, результат операции помещается в аккумулятор. При выполнении операций умножения и деления в дополнение к аккумулятору используется вспомогательный регистр B.

Два 16-разрядных таймера Timer0 и Timer 1 позволяют аппаратно задавать временные интервалы требуемой длительности, а также обеспечивают возможность использования микроконтроллера для работы в режиме реального времени.

Связь с «внешним миром» выполняется через четыре двунаправленных байтовых порта P0 – P3. Порты P0 и P2 связывают с «внешним миром» системную шину микроконтроллера. Это позволяет при необходимости добавлять различные модули к архитектуре микроконтроллера за счет их внешнего подключения. Порты P1 и P3 являются портами общего назначения.

Очень полезным элементом микроконтроллера является универсальный асинхронный приемопередатчик UART. Это устройство позволяет передавать и принимать байтовые данные по асинхронной последовательной линии связи. При оснащении микроконтроллера внешним модулем согласования уровней сигналов, появляется возможность включаться в промышленные сети передачи данных, т.е. информационно объединять различное оборудование посредством больших промышленных сетей передачи данных.

Встроенный блок обработки прерываний IP позволяет микроконтроллеру реагировать на определенные события, как внешние, так и внутренние. Механизм обработки прерываний совместно с таймерами делает возможным использование микроконтроллеров семейства MCS-51 для обработки данных и управления в режиме реального времени [2, 22].

Управляет работой микроконтроллера встроенный микропрограммный управляющий автомат CU. Этот микропрограммный автомат с естественной адресацией имеет собственную память ROM для хранения микропрограммы работы. Микропрограмма загружается в память кристалла на этапе его изготовления, пользователю она не доступна. Генератор G с внешним задающим кварцевым резонатором формирует тактовые импульсы, необходимые для работы микропрограммного автомата. Функционируя по своей внутренней микропрограмме, микропрограммный автомат считывает из EEPROM очередную команду пользователя, декодирует ее как входную переменную и, в зависимости от содержания, переходит на одну из ветвей своей микропрограммы для исполнения команды.

Восьмибитовая команда пользователя позволяет закодировать 256 различных команд. В микроконтроллере MCS – 51 использованы все возможные 256 команд. Это означает, что микропрограмма управляющего

автомата должна иметь 256 различных ветвей, отрабатывая которые автомат управляет работой внутренних блоков кристалла.

Микроконтроллеры семейства MCS-51 «тратят» на исполнение команды пользователя 12 микрокоманд автомата (или 12 тактов генератора тактовых импульсов). Поэтому при типовой тактовой частоте 12 или 24 МГц, быстродействие машины составляет 1-2 млн. операций пользователя в секунду.

Двухуровневое управление ходом вычислительного процесса, когда команды пользователя (верхний уровень) реализует микропрограммный управляющий автомат, работающий по программе нижнего уровня, применяется практически во всех современных компьютерах. В таком двухуровневом управлении ходом вычислений и заключается способ **программного управления вычислениями**. Усложнение организации машины приводит к значительным удобствам для пользователя. При программном управлении вычислениями программисту не нужно досконально изучать архитектуру машины и особенности работы отдельных блоков. Достаточно изучить систему команд и результаты, к которым приводит их исполнение. После программирования сигнальных процессоров с их многоразрядными микрокомандами и полями управления отдельными блоками, программирование микроконтроллеров кажется легкой забавой.

И так, программное управление ходом вычислений позволяет существенно упростить процедуру программирования, ускорить процесс разработки устройства и снизить требования к квалификации программиста. Вычислительная эффективность программы, по сравнению с микропрограммным способом, конечно падает. Однако от микроконтроллера и не требуется такого высокого быстродействия, какое получается у сигнального процессора. Его задача – вовремя отреагировать на изменение ситуации, сравнить данные, выполнить их пересылку.

В современной технике применяется огромное количество различных микроконтроллеров. Восьмиразрядный I-8051 был моден в 90-х годах прошлого века. На смену ему пришли 16-разрядные, а затем и 32-разрядные микроконтроллеры. В настоящее время, в связи с развитием сетевых технологий и внедрением распределенных систем управления, все большее число элементов автоматики проектируются с встроенным микроконтроллером. Этот процесс возродил интерес разработчиков к простым и дешевым восьмиразрядным микроконтроллерам, в частности семейства MCS-51, модифицированным под современные требования техники, [3, 4, 22, 27].

2.13. Программная реализация табличного способа вычисления

Как известно, табличные вычисления применяют для вычисления значений нелинейных функций, логического управления объектами, для построения генераторов программных траекторий. Первое применение

требует вычисления значения некоторой функции в темпе текущих вычислений, т.е. в темпе поступления входной переменной X. Генерирование функции времени требует дополнительно учета хода реального времени. Задача логического управления может быть, как асинхронной, в этом случае она сводится к первой задаче, либо синхронной и ее реализация подобна второй задаче. Поэтому подробно рассмотрим особенности программной реализации задач табличного вычисления значения нелинейной функции и генерирования функции времени.

2.13.1. Вычисление значения нелинейной функции

Заранее вычисленные значения функции располагают непосредственно в ПЗУ команд в области памяти вне программы работы контроллера. Пусть необходимо вычислить значение функции *Y = f (X)*, причем *X* и *Y*- байтовые переменные. Для хранения значений функции *f(X)* требуется память, объемом 2^8 * 1 байт =256 байт.

Выделяем в ПЗУ область, объемом 256 байт, начиная с адреса, обозначенного меткой TAB. В ячейку TAB запишем значение *f(0)*, в ячейку TAB+1 запишем *f(1)*, далее *f(2)* и т.д.

Программа работы микроконтроллера должна читать данные *f(X)* из памяти программ. Для этого используется специальная машинная команда

movc A,@A+DPTR,

где: movc - обозначение команды пересылки в аккумулятор A байта из памяти программ, A+DPTR - вычисляемый адрес ячейки памяти программ, DPTR - специальный 16-разрядный регистр. Действие команды состоит в том, что контроллер складывает содержимое регистра DPTR с содержимым аккумулятора A и полученное значение использует как адрес ячейки памяти программ, из которой байт данных пересылается в аккумулятор.

Фрагмент программы вычисления значения нелинейной функции получается следующим:

```
.
.
.
mov DPTR, #TAB   ; Запись адреса начала таблицы в регистр DPTR
mov A, X         ; Пересылка содержимого регистра X в аккумулятор
mov A, @A+DPTR ; Чтение табличного значения функции
mov Y, A         ;  Пересылка результата табличного вычисления в
.                ;       ячейку Y оперативной памяти
.
.
```

Вычисления значения нелинейной функции выполняется с помощью 4 команд и требует 6 мкс времени при реализации программы на типовом

контроллере MCS-51 с частотой кварцевого резонатора 12 МГц. Следует обратить внимание, что ни количество команд, ни время вычисления не зависят от вида нелинейной функции.

Рассмотрим еще пример программной реализации задачи кусочно - линейной аппроксимации нелинейной функции. Предположим, что требуется программно реализовать устройство, схема аппаратной реализации которого показана на рисунке 2.13.1. Работа аналогичного устройства подробно разобрана в разделе 1.14.

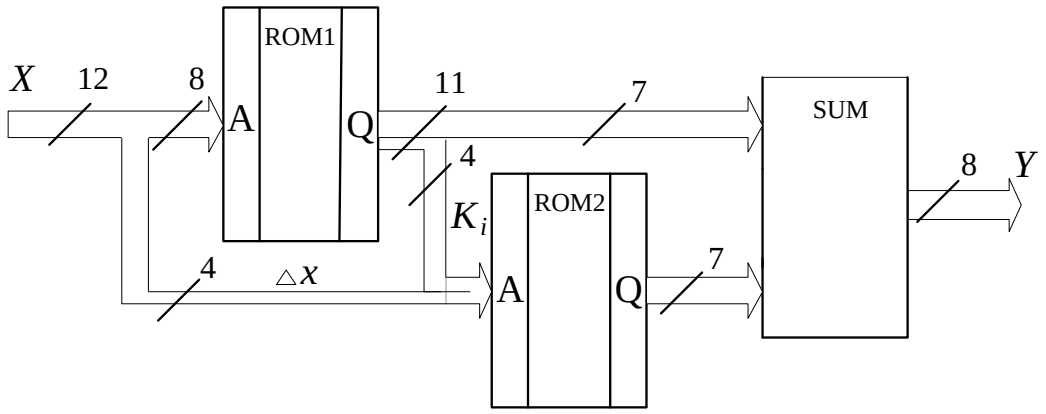

Рисунок 2.13.1 - Схема аппаратной реализации табличного вычислителя

В схеме использованы два постоянных запоминающих устройствах - ROM1 и ROM2. В ПЗУ ROM1 хранятся значения функции Y_i (представляются 7 битовыми кодами) в реперных точках X_i , i = 0… 255, и значения коэффициентов K_i (представляются 4-разрядными кодами) наклона функции в тех же реперных точках. В ПЗУ ROM2 хранятся произведения смещения аргумента Δx (представляется 4-разрядным кодом) на коэффициент наклона K_i. Сумматор вычисляет значение выходной переменной *Y* в виде 8-ми битового кода. Таким образом, в рассматриваемой схеме реализованы вычисления по следующим уравнениям:

$$X = X_i + \Delta x ,$$
$$Y = f(X_i) + K_i * \Delta x ,$$

где: *f(.)* - заданная нелинейная функция.

При программной реализации устройства на восьмибитовом микроконтроллере семейства MCS-51 удобно ПЗУ ROM 1 разделить на две таблицы, объемом по 256 байт каждая. Постоянное запоминающее устройство ROM 2 также реализуем в виде таблицы объем 256 байт. Обозначим начальные адреса (16-разрядные) этих трех таблиц как TAB1Y – для таблицы Y_i, TAB1K – для таблицы коэффициентов K_i , TAB2 – для таблицы произведений.

Будем считать, что аргумент X хранится в 2 соседних ячейках памяти данных: XL-младший (8-разрядный), и XH-старший (4-разрядный). Фрагмент программы работы микроконтроллера MCS-51 приведен ниже:

;Вычисление значений X_i и Δx

```
clr C; Очистим бит переноса
mov A, XL       ; Выделяем у аргумента реперную точку и смещение
rlc A           ;  для этого сдвигаем X в лево через перенос
mov R0,A        ; Сохраняем промежуточный результат в регистре R0
mov A,XH        ; Выполняем то же действие со старшим байтом X
rlc A
mov R1,A        ; Сохраняем промежуточный результат в регистре R1
clr C           ; Повторяем сдвиг в лево еще 3 раза
mov A, R0
rlc A;
mov R0,A
mov A,R1
rlc A
mov R1,A
clr C
mov A, R0
rlc A;
mov R0,A
mov A,R1
rlc A
mov R1,A
clr C
mov A, R0
rlc A;
mov R0,A
mov A,R1
rlc A
mov R1,A        ; В результате в регистре R1 сформирован код Xi
mov A,R0        ;      в R0 хранится код Δx, но он содержится в старшей
swap A          ;      тетраде байта, поэтому переставляем тетрады
mov R0,A
```

; Чтение значения $Y_i = f(X_i)$ из таблицы TAB1Y

```
mov DPTR,#TAB1R; Запись в DPTR адреса таблицы TAB1Y
mov A,R1        ; Запись в аккумулятор смещения в таблице TAB1Y
movc A, @A+DPTR; Пересылка в аккумулятор байта Yi
mov R3,A        ; Временное сохранение результата в регистре R3
```

; Чтение коэффициента K_i из таблицы TAB1K

```
mov DPTR,#TAB1K ; Запись в DPTR адреса начала таблицы TAB1K
mov A,R1        ; Запись в аккумулятор смещения в таблице TAB1K
```

```
movc A, @A+DPTR; Чтение в аккумулятор  байта Ki
swap A          ; Перемещаем коэффициент в старшую тетраду
orl A,R0        ; Добавляем по ИЛИ в младшую тетраду смещение Δx
; Вычисление произведения  Ki * Δx
mov DPTR,#TAB2;  Запись в DPTR адреса начала TAB2
movc A, @A+DPTR; В аккумуляторе  байт Ki * Δx из таблицы TAB2
; Вычисление итогового результата
add A,R3        ; Сложение  Yi  + Ki * Δx
mov Y,A         ; Сохранение результата
```

Как следует из приведенного текста фрагмента программы, вычисление значения нелинейной функции методом кусочно-линейной аппроксимации требует 47 команд и его выполнение будет продолжаться 50 мкс.

2.13.2. Реализация генератора функции времени

В разделе 2.7 был рассмотрен микропрограммный автомат с усеченной естественной адресацией, который можно использовать в качестве генератора периодической функции времени. Схема такого устройства показана на рисунке 2.13.2. Она содержит генератор импульсов G, двоичный счетчик CT2 и постоянное запоминающее устройство ROM. Рассмотрим вариант его программного аналога.

Пусть, для примера, нам необходим генератор задающего воздействия

$$y = sin\ (\omega * t)\ ,$$

с частотой 10 Гц, т.е. ω = 62,8 1/с. Будем считать, что за период синусоиды должно вырабатываться 100 значений функции. Следовательно, период квантования времени должен быть равен 1/1000 с., т.е. частота квантования равна 1000 Гц. Однозначное задание номера отсчета N требует использования 7-разрядного счетчика адреса микрокоманды.

Для выбора разрядности выходной переменной вспомним, что минимальное изменение выходная переменная претерпевает в точках $\omega * t = (0{,}25 + 0{,}5 * i) * 6.28$, $i = 0, 1, 2, \ldots$. При количестве отсчетов функции на одном периоде, равном 100, это приращение составит:

$$(y_{i+1} - y_i)_{min} = sin(6.28*25/100) - sin(6{,}28*24/100) = 0{,}002.$$

Так как диапазон изменения функции от -1 до +1, то для разрешения всех возможных значений синусоиды нам потребуется разрешать 1000 различных ее значений на диапазоне. Однозначное кодирование такого количества значений требует использования 10-разрядного кода.

Для целочисленного задания Y введем для нее масштаб m = 1024 (т.е. 2^{10}). Кроме того, необходим сигнал сброса счетчика СТ2 в ноль, когда число в счетчике $N \geq 100$.

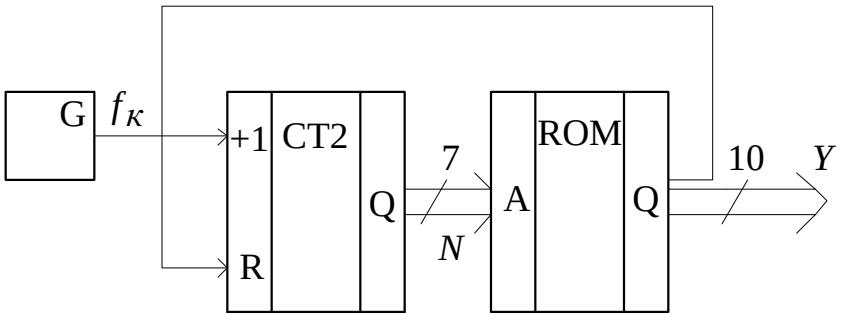

Рисунок 2.13.2 - Генератор функции времени на основе микропрограммного автомата с усеченной естественной адресацией

Как было показано в разделе 2.13.1. программная реализация постоянного запоминающего устройства не представляет сложностей. Микроконтроллер MCS51 имеет специальную команду movc A,@A+DPTR. Счетчик импульсов также легко реализуется программно. Остается задача реализации заданного периода квантования выходного сигнала Y, равного в нашем примере 1/1000 с.

Для реализации заданного периода квантования времени микроконтроллеры содержат специальные устройства - таймеры/счетчики. Микроконтроллер MCS51 содержит два таймера/счетчика - Timer0 и Timer1. Оба счетчика имеют одинаковую архитектуру и способны формировать сигналы прерывания программы при переполнении. Поэтому для задания периода квантования сигналов используем механизм обработки прерываний от таймера, а программу генерации значений функции запишем, как программу обработки прерывания. Такая организация работы микроконтроллера позволяет наиболее эффективно использовать его архитектурные и вычислительные возможности.

На рисунке 2.13.3 показана укрупненная блок-схема программы работы микроконтроллера. Она состоит из трех основных частей: инициализации, основного цикла и подпрограммы обработки прерывания. В части инициализации обычно определяются конфигурация микроконтроллера, режимы работы его устройств, в нашем примере - Timer0. Здесь задается период отсчетов генерируемой функции. Так как в этой части производится настройка устройств, то обычно прерывания от любых устройств запрещены. Часть инициализации исполняется один раз при включении питания (или при перезапуске) микроконтроллера.

В части основного цикла пишутся коды действий микроконтроллера, не требующие работы в реальном времени. Это может быть опрос клавиатуры, анализ команд пользователя, вывод информации на дисплей, выполнение некоторого большого расчета и т.д. Главное, что эти действия выполняются

циклически все время при работе микроконтроллера. Может показаться, что приведенные выше примеры действий являются действиями в реальном времени. Однако это не совсем так. Например, вывод информации на

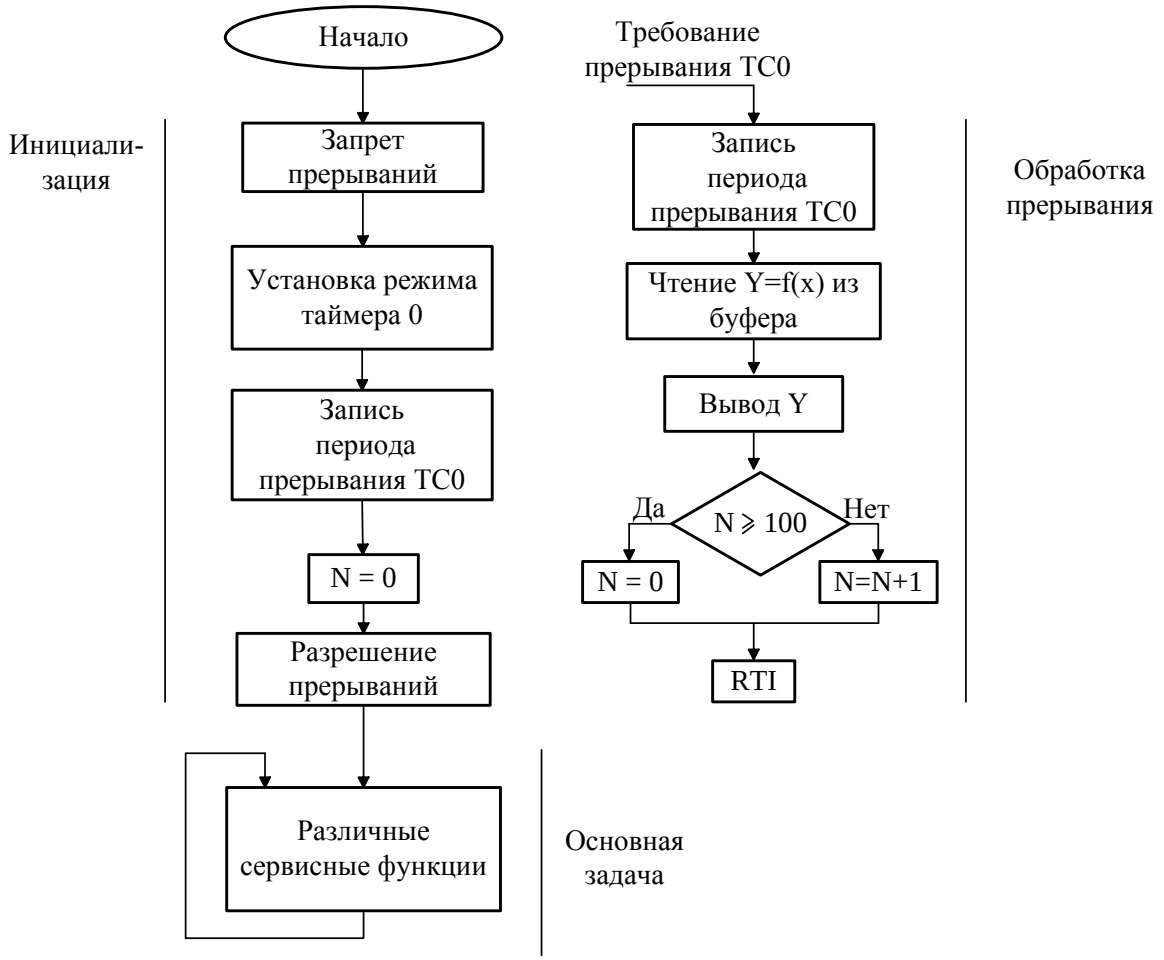

Рисунок 2.13.3 - Блок-схема программы генератора функции времени

цифровой индикатор для чтения человеком производится циклически с темпом порядка 0.5 с. (500 мс). Это значение выбирается исходя из темпа восприятия информации человеком. Если в ходе расчетов микроконтроллер "задержится" и выведет очередное число на индикатор через 511 мс (например), человек этого даже не заметит. Следовательно, указанный выше период вывода информации не является жестко заданным.

Другое дело - расчет значений функции синуса в генераторе синусоидального воздействия. Микроконтроллер обязан выдавать значения синусоиды с тактом 0,5 мс, задержка с выдачей информации в этой задаче недопустима. Поэтому задачи, операции ввода-вывода информации которых должны быть жестко привязаны к ходу реального времени, отклонения от заданных значений периодов квантования информации у которых недопустимы, являются **задачами реального времени.** На современном

техническом языке такие задачи стали обозначать усиленным термином - **задачи жесткого реального времени**. Программная реализация генератора функции времени относится к этому классу задач.

Определим начальную загрузку Timer0 такую, чтобы прерывания от него имели период 1 мс. Так как на таймер в контроллере MCS51 при частоте кварцевого резонатора 12 МГц поступают импульсы с частотой 1 МГц, то до наступления прерывания таймер должен подсчитать 1000 импульсов. При 16 -разрядном режиме работы прерывание наступает при переполнении таймера, т.е. при превышении числа 65 535. Поэтому при начальной загрузке или перезагрузке таймера в него необходимо записывать число 65536 - 1000 = 64536, что в шестнадцатиричном коде равно 0FC18h. Таким образом, инициирующая часть программы должна содержать следующий код инициализации таймера/счетчика Timer0:

```
.
.
.
mov TMOD,#31h ; Timer0 работает в 16-разрядном режиме
mov TH0,#0FCh  ; Запись числа 64536 в старший
mov TL0,#18h     ;   и младший регистры Timer0
mov TCON,#10h ; Пуск Timer0
mov N,#0           ; Обнуление счетчика тиков времени
mov IE,#82h       ; Разрешение прерываний от Timer0
.
.
.
```

Генератор функции времени теперь пишем, как обработчик прерываний от Timer0:

```
Timer0: mov TH0,#0FCh; Запись числа 64536 в старший
        mov TL0,#18h   ;   и младший регистры таймера 0
        push PSW        ; Сохранение состояния процессора в стеке
        push ACC        ; Сохранение значения аккумулятора в стеке
        mov DPTR,#TABY; Запись адреса начала таблицы
        mov A,N          ; Удвоение номера такта N,
        rl A                ;    так как работаем с двухбайтовым Y
        movc A,@A+DPTR; Чтение младшего байта Y
        mov P1,A         ; Вывод младшего байта Y в порт P1
        mov A,N          ; Восстановление A
        rl A
        inc A              ; Увеличение смещения в таблице на 1
        movc A,@A+DPTR; Чтение старшего байта Y
        mov P2,A        ; Вывод старшего байта Y в порт P2
        inc N             ; Увеличение счетчика номера тика времени на 1
        mov A,N          ; Проверка окончания периода
```

```
        cjne A,#100,Tim1  ; Переход, если не равно 100
        mov N,#0          ; При N ≥ 100,  обнуляем N
Tim1:   pop A             ; Восстанавливаем значение в аккумуляторе
        pop PSW           ; Восстанавливаем слово состояния процессора
        reti              ; Завершение обработки прерывания. Возврат к
                          ;    прерванной программе
```

Как видно из приведенного текста, программная реализация генератора функции времени получается достаточно компактной и несложной. При этом быстродействие микроконтроллера не играет особой роли, темп обновления информации на выходах микроконтроллера определяется работой таймера.

2.14 Программная реализация регистрового способа вычисления

Регистровый способ вычислений реализуется на микроконтроллерах чаще всего, так как требует минимальных затрат оперативной памяти.

Особенности программной реализации регистрового способа вычислений рассмотрим на простом примере. Предположим, что необходимо реализовать устройство, вычисляющее среднее значение по четырем последовательно измеряемым значениям входной переменной *x(n)*:

$$y(nT) = [x(nT-T) + x(nT-2T) + x(nT-3T) + x(nT-4T)]/ 4,$$

где: *n* - дискретное время, *T* - заданный период квантования сигнала во времени.

Структурная схема устройства показана на рисунке 2.14.1. На рисунке приняты следующие обозначения: $X1(nT) = x(nT-T)$, $X2(nT) = x(nT-2T)$, $X3(nT) = x(nT-3T)$, $X4(nT) = x(nT-4T)$. Сразу отметим, что деление на 4 можно выполнять как в конце вычислений, так и в начале. Действительно, входную переменную *x(nT)* можно делить на 4 сразу после получения. Однако для обеспечения наименьшей погрешности работы устройства деление целесообразно выполнять в конце вычислений. Этот вывод следует из того, что в микроконтроллере величина *x(nT)* обычно представляется в целочисленном формате с конечным числом разрядов. В рассматриваемом примере разрядность величины *x(nT)* принята равной 8.

Период квантования времени *T* вычислительного процесса может быть реализован по крайней мере двумя способами. Первый способ состоит в использовании аппаратного таймера микроконтроллера. Таймер нужно настроить таким образом, чтобы вырабатываемые им прерывания происходили с заданным периодом, равным *T*. Ход вычислительного процесса в этом случае тактируется внутренними средствами микроконтроллера. Пример использования прерываний таймера для задания периода квантования времени *T* рассмотрен в разделе 2.13.2.

Второй способ реализации периода квантования времени T состоит в использовании внешних по отношению к микроконтроллеру сигналов синхронизации передаваемой цифровой информации. В настоящем параграфе рассмотрим именно этот способ.

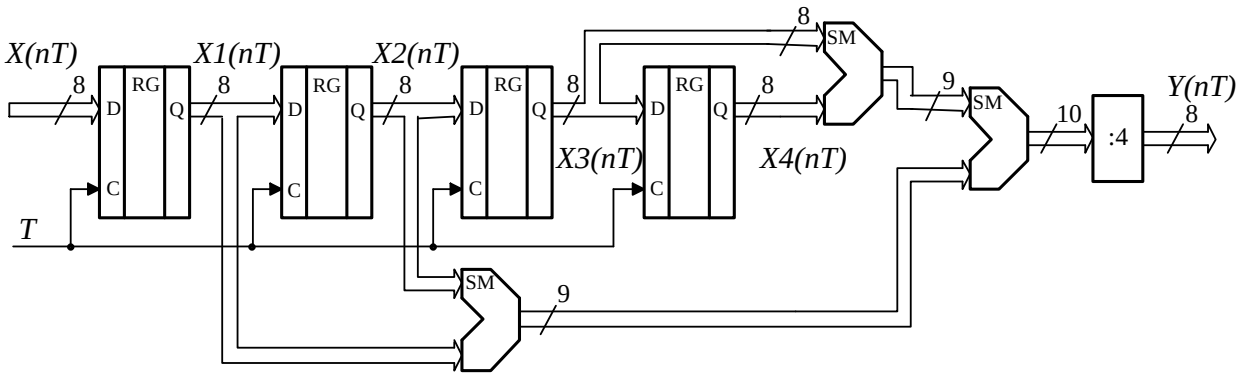

Рисунок 2.14.1 - Схема устройства, аппаратно реализующего регистровый способ вычислений

Предположим, что входной сигнал *x(nT)* поступает от некоторого цифрового датчика на порт P1 микроконтроллера в формате однобайтового целого числа без знака. Поступление сигнала *x(nT)* сопровождается синхроимпульсом, формируемым датчиком. Так как первичное квантование информации выполняется датчиком, то именно он задает величину периода T квантование сигнала *x(nT)*, формируя сопровождающие синхроимпульсы.

Для реализации синхронной работы с датчиком воспользуемся режимом прерывания программы. Для этого синхроимпульс от датчика подадим на вход внешнего прерывания Int0 микроконтроллера. При поступлении очередного отсчета *x(nT)* на вход P1, наличие синхроимпульса будет вызывать прерывание выполнения текущего фрагмента программы и переход на подпрограмму обслуживания прерывания от Int0. В нашем примере подпрограмма обслуживания прерывания от Int0 и будет реализовывать требуемый алгоритм вычисления среднего значения входного сигнала.

На рисунке 2.14.2 показана блок-схема программы работы микроконтроллера. Программа содержит части: инициализации, основного цикла и подпрограммы обработки прерывания по сигналу на линии Int0. В блоке инициализации, запускаемом один раз после подачи питания, выполняется настройка микроконтроллера под специфику решаемой задачи. Поле процедуры инициализации запускаются процессы, не требующие строгой синхронизации с ходом реального времени – чаще всего различные сервисные действия устройства.

Поступление сигнала на вход Int0 означает обновление входных данных. Поэтому при возникновении прерывания по линии Int0 обработчик прерывания выполняет ввод этих данных, расчет среднего значения Y и вывод его через порт P2, обновление информации во внутренних регистрах вычислительной схемы (оперативной памяти).

Предположим, что для хранения внутренней информации вычислительной схемы использованы ячейки оперативной памяти, имеющие символические имена *X1, X2, X3* и *X4*. Кроме того, для вычисления суммы

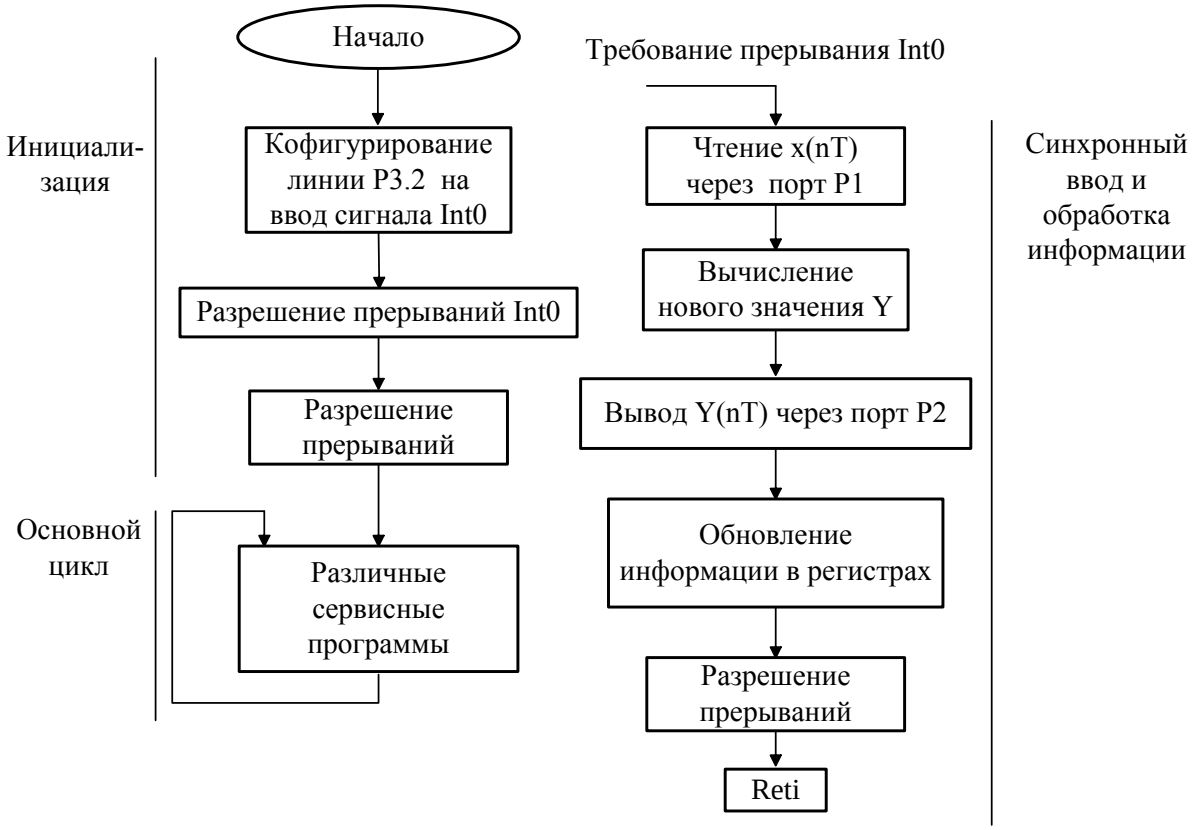

Рисунок 2.14.2 - Блок-схема программы, реализующей регистровый способ вычислений.

четырех переменных используется вспомогательная двухбайтовая переменная SUM, старший байт которой обозначим SUMH, а младший SUML. Ниже приводится текст программы работы устройства на языке ассемблер MCS-51 (без распределения переменных в оперативной памяти):

```
    Org 0        ; Задан абсолютный адрес в памяти программ
    jmp Start    ; По сигналу Сброс – переход на метку Start

    Org 3
    jmp NewX     ; В векторе прерывания Int0 записан переход на
                 ;метку NewX
     Org 100     ; Адрес начала программы

    ; Инициализация.
Start: mov IE,#0  ;Запрет всех прерываний
       setb P3.2  ; Линия порта P3.2 сконфигурирована на ввод  Int0
```

```
        setb TCON.0; Прерывание Int0 вызывается по срезу сигнала
        mov IE,#81h; Разрешение прерывания Int0 и прерываний вообще

; Основной цикл
Main:  nop    ; Здесь просто пустая операция
         jmp Main

; Обработчик прерывания Int0
NewX:      push PSW  ; Сохранение с стеке слова состояния процессора
           push ACC  ; Сохранение с стеке содержимого аккумулятора
           mov A, P1 ; Чтение X[nT]
           mov X1, A ; Сохранение значения в X1
           add A, SUML; Сложение с частичной суммой
           mov SUML, A
           mov A, SUMH
           addc A, #0
           clr C          ;  Вычисление Y делением суммы на 4
           rlc A          ;          Деление  на 2 путем сдвига вправо
           mov SUMH, A
           mov A, SUML
           rlc A
           mov SUML, A
           clr C          ; Повторное деление на 2 путем сдвига вправо
           mov A, SUMH
           rlc A
           mov A,SUML
           rlc A
           mov P2, A  ;  Вывод Y в порт P2
           mov A, X2 ;  Вычисление частичной суммы
           add A, X3
           mov SUML, A
           mov A, #0
           addc A, #0
           mov SUMH, A
           mov A, SUML
           add A, X4
           mov SUML, A
           mov A, SUMH
           addc A, #0
           mov SUMH, A
           mov X4, X3; Обновление информации в ячейках ОЗУ
           mov X3, X2
           mov X2, X1
           pop ACC; Восстановление прежнего значения аккумулятора
```

```
pop PSW; Восстановление слова состояния процессора
reti;  Возврат к выполнению прерванной программы
```

В приведенном тексте программы, в части обработчика прерывания, используется двухбайтовая переменная SUM. Это связано с тем, что при суммировании четырех однобайтовых переменных разрядность результата равна 10 бит. Поэтому в операциях суммирования переменных и сдвига переменной SUM вправо (для деления результата суммирования на 4) используется бит переноса C, хранящийся в слове состояния процессора PSW. В результате из общих ресурсов контроллера подпрограмма обработки прерывания использует аккумулятор и слово состояния процессора. Чтобы не нарушать работу других подпрограмм, значения этих двух регистров сохраняются в стеке в начале обработки прерывания, и восстанавливаются из стека в конце обработки.

Если принять, что тактовая частота микроконтроллера MCS-51 равна 12 МГц, то время выполнения одного цикла обработки прерывания составит 70 мкс, не считая времени реакции микроконтроллера на внешний сигнал Int0. В разделе 2.11 мы рассчитали время цикла обработки сигнала в аналогичной задаче для сигнального процессора. Это время составило 3.2 мкс при тактовой частоте процессора 6 МГц. Этот несложный пример показывает вычислительную эффективность сигнальных процессоров.

С другой стороны, для написания программы обработчика прерывания микроконтроллера MCS-51, нам не потребовалось детального знания архитектуры и особенностей функционирования отдельных блоков микроконтроллера. Потребовались в основном общие знания двоичного представления чисел, информация о переносе и общее правило сложения многоразрядных двоичных чисел. Поэтому можно утверждать, что для микроконтроллеров писать программы существенно проще, чем для сигнальных процессоров. Хотя, если использовать ассемблер, программы для микроконтроллеров тоже получаются довольно громоздкими.

Заключение

Создание современных систем управления немыслимо без использования цифровых вычислительных устройств. Имеется в виду не применение персональных компьютеров для моделирования в программной среде, а использование вычислительных устройств для реализации систем управления объектами в режиме реального времени. Разнообразие задач, решаемых такими вычислителями, постоянно расширяется. Соответственно, постоянно появляются новые и более совершенные средства обработки информации, ее накопления и передачи.

На смену классическим микроконтроллерам, построенным на регистровых структурах с микропрограммным управлением, приходят микроконтроллеры с табличной организацией вычислительного ядра.

Появились микросхемы с архитектурой, программируемой пользователем. При этом плотность размещения полупроводниковых элементов на кристалле увеличилась на порядки, увеличилось быстродействие, снизилась потребляемая мощность. Другими словами, вычислительные средства также значительно изменились.

В этой связи интересно исследование, какие технические средства вычислительной техники применяют разработчики в повседневной работе. На сайте магазина электронных компонентов «Элтех» в 2006 году проводился подсчет голосов разработчиков, отдающих предпочтение тому или иному вычислительному средству. На 1 июня 2006 года распределение голосов разработчиков было следующее:

AVR - 1185;
MCS-51 - 1031;
PIC - 906;
z80 - 633;
i8080 - 609;
x86 - 601;
MSP430 - 415;
ARM - 396;
MCS-196 - 145;
C166 - 114;
68K - 95;
F2MC - 66;
M16C - 63;
H85 - 44;
ColdFire - 31;
SuperH - 23;
78K - 17;
v850 - 12;
Остальные - 204;

Среднее количество разработчиков на одно ядро контроллера – 347 человек. Как видно из приведенного перечня лидерами разработок были микроконтроллеры AVR фирмы Atmel, MCS-51 различных фирм и PIC фирмы Microchip. Характерно, что все три лидирующие платформы в своей массе восьмибитовые. Лидер продаж, микроконтроллеры AVR, базируются на табличном ядре (имеют RISC архитектуру), имеют сокращенную систему команд и повышенное быстродействие.

В следующие годы ситуация несколько изменилась. В 2010 году цена на восьмиразрядный микроконтроллер ADuC841 (ядро MCS-51) с встроенным 12-разрядным аналого-цифровым преобразователем стала примерно равной цене на микроконтроллер ADuC7026 (32 битовое ядро ARM-7) с более скоростным АЦП и более мощной периферией. В продаже появились микроконтроллеры с ядрами ARM-9, Cortex-M3, Cortex-M4 и другими 32-битовыми ядрами. Конечно, это не могло не сказаться на предпочтениях

разработчиков. К сожалению магазин «Элтех» прекратил опрос разработчиков и мы не располагаем новыми статистическими данными.

Разработчики современной цифровой техники управления и обработки информации располагают для реализации проектов мощной технической элементной базой, имеющей относительно невысокую стоимость. Поэтому основная тенденция разработок - снижение затрат времени на их выполнение. С этой целью, при выборе технического средства, предпочтение отдаются таким элементам, которые имеют мощную программную поддержку в виде программных сред разработки прикладного программного обеспечения и средства для его отладки.

Список использованных источников

1 Аналого-цифровое преобразование / под ред. **У. Кестера.** – М.: Техносфера, 2007. – 1016 с., ил.; ISBN 978-5-94836-146-8

2 Аксенов В.П. Сигнальные процессоры: учебное пособие.- Владивосток: ДВПИ им. Куйбышева, 2006. – 135 с., ил.

3 **Белов А.В.** Микроконтроллеры AVR: от азов программирования до создания практических устройств. Книга + CD с видеокурсами, листингами, программами, драйверами, справочниками. – СПб.: Наука и Техника, 2016. – 544 с., ил. + CD; ISBN 978-5–94387-363-8

4 **Белодедов М.В.** Методы проектирования цифровых фильтров: Учебное пособие. – Волгоград: Издательство Волгоградского государственного университета, 2004. – 64 с., ил.; ISBN 5-85534-929-2

5 **Бондарев В.Н., Трестер Г., Чернега В.С.** Цифровая обработка сигналов: методы и средства. Учеб. пособие для вузов. 2-е изд. – Х.:Конус, 2001.- 398 с.: ил.; ISBN 966-7836-14-3

6 **Бородин В.Б., Калинин А.В.** Системы на микроконтроллерах и БИС программируемой логики. – М.: Издательство ЭКОМ, 2002. – 400 с., ил.; ISBN 5 – 7163- 0089 -8

7 **Воронов А.А.** и др. Цифровые аналоги для систем автоматического управления. – М.: Наука, 1960.- 196 с., ил.

8 **Глазков В.В.** Программируемые логические интегральные схемы фирмы Altera: учеб. пособие по дисциплине «Технология и схемотехника средств управления в технических системах». [Электронный ресурс] : учеб. пособие — Электрон. дан. — М. : МГТУ им. Н.Э. Баумана, 2014. — 133 с. — Режим доступа: http://e.lanbook.com /book/58395, ISBN 978-5-7038-3839-6

9 **Григорьев В.В. и др.** Цифровые системы управления: учебное пособие. – СПб.: Универитет ИТМО, 2018. - 133 с.: ил.

10 **Дударенко Н.А., Слита О.В., Ушаков А.В.** Математические основы современной теории управления: аппарат метода пространства состояний: учебное пособие./ Под ред. Ушакова А.В. – СПб.: СПб ГУ ИТМО, 2009. – 325 с., ил.

11 **Катупития Я., Бентли К.** Управление электронными устройствами на C++. Разработка практических приложений. / Перевод с англ. Бакомчев И. В. – М.: ДМК Пресс, 2016. – 442 с., ил.; ISBN 978-5-97060-175-4

12 **Кнышев, Д.А.** ПЛИС фирмы «Xilinx»: описание структуры основных семейств [Текст] / Кнышев Д. А., Кузелин М.О. – М. : ДМК Пресс, 2017. - 238 с. : ил.; ISBN 978-5-97060-546-2

13 **Круг П.Г.** Процессоры цифровой обработки сигналов: учебное пособие.- М.: Издательство МЭИ, 2001. – 128 с., ил.; ISBN 5-7046-0778-0

14 **Ланина Э.П.** История развития вычислительной техники. – Иркутск.: Изд. ИрГТУ, 2001.-166 с.

15 **Лазарев В.Г., Пийль Е.И.** Синтез управляющих автоматов. – 3-е изд., перераб. и доп. – М.: Энергоатомиздат, 1989. – 328 с., ил. ISBN 5-283-01494-0
16 **Максфилд К.** Проектирование на ПЛИС. Курс молодого бойца. – М.: Издательский дом «Додэка - XXI», 2007 – 408 с., илл. ISBN 978 – 5-94120-147-1
17 **Мельников А.А., Ушаков А.В.** Двоичные динамические системы дискретной автоматики / Под ред. А.В. Ушакова. – СПб.: СПбГУ ИТМО, 2005.-220 с., ил. ISBN 5-7577-0253-2
18 **Наваби З.** Проектирование встраиваемых систем на ПЛИС.- М.: ДМК-Пресс, 2016. - 464 с., ил.; ISBN 978-5-97060-174-7
19 Основы цифровой обработки сигналов: Курс лекций / Авторы: **Слонина А.И., Улахович Д.А., Арбузов С.М., Соловьева Е.Б.** - С-Пб.: БХВ-Петербург, 2005. – 768 с.: ил.
20 **Поляков К.Ю.** Основы теории цифровых систем управления: учебное пособие. - СПб.: СПб ГМТУ 2006.- 161 с.: ил.
21 **Сергиенко Александр Борисович**. Цифровая обработка сигналов [Текст]: доп. М-вом образования Рос. Федерации в качестве учебного пособия для студентов высших учебных заведений, обучающихся по направлению подготовки дипломированных специалистов "Информатика и вычислительная техника"/А. Б. Сергиенко.— СПб.: Питер, 2005.— 604 с.: ил .: ISBN 5-318-00666-3
22 **Трамперт В.** Измерение, управление и регулирование с помощью AVR микроконтроллеров.: Пер. с нем. – К.: «МК-Пресс», 2006 – 208 с., илл. ISBN 966-8806-14-X (рус.)
23 **Ушаков А.В.** Прикладная теория информации: элементы теории и практикум: Учебное пособие для вузов. - СПб.: СПб НИУ ИТМО, 2012. - 326 с., ил.
24 **Шевкопляс Б.В.** Микропроцессорные структуры. Инженерные решения. Справочник. - М.:Радио и Связь, 1990г.
25 **Шеннон К.Э.** Математическая теория дифференциального анализатора в кн.: Работы по теории информации и кибернетике.: Пер. с англ. – М.: Издательство иностранной литературы, 1963. – 824 с., ил.
26 **Шишкин Г.Г.** Наноэлектроника. Элементы, приборы, устройства [Электронный ресурс]: Учебное пособие/Г.Г.Шишкин, И.М. Агеев.-2е изд.(эл.). - М.: БИНОМ. Лаборатория знаний, 2012.-408 с.; ил.; ISBN 978-5-9963-1443-0
27 **Яценков В.С.** Микроконтроллеры Microchip с аппаратной поддержкой USB. – М.: Горячая линия – Телеком, 2008 – 400 с., ил.; ISBN 978 – 5-9912-0030-1

Приложение А

Терминология

Antifuse технология - способ создания PROM-памяти. Довольно часто применяется для создания ПЛИС-микросхем, предназначенных для применения в тяжелых природных условиях - при больших перепадах температур или радиации.
Argus - ассемблер реконфигурируемых вычислительных систем. Разработка НИИ МВС ЮФУ.
ASIC (Application-Specific Integrated Circuit) - интегральная схема, выполненная на заказ для решения конкретной задачи. Способна выполнять ограниченный набор функций, однако эффективность реализации этих функций обычно очень высока. Является своего рода конкурентом ПЛИС-микросхем, поскольку и те, и другие обычно используются для высокоэффективного решения небольшого круга задач.
CLB (Configurable Logic Blocks) - программируемый логический блок, часть FPGA-устройства, предназначенная для программирования некоторой функции или её части. Может быть использован для других целей, например, в качестве памяти.
Colamo - язык структурно-процедурного программирования высокого уровня реконфигурируемых вычислительных систем. Разработка НИИ МВС ЮФУ.
CPLD/SPLD (Complex/Simple Programmable Logic Device) - разновидность ПЛИС, содержащая относительно крупные программируемые логические блоки - макроячейки (англ. macrocells), соединённые с внешними выводами и внутренними шинами. Функциональность CPLD кодируется в энергонезависимой памяти, поэтому нет необходимости их перепрограммировать при включении.
DSP (Digital Signal Processing) - цифровая обработка сигналов (ЦОС), также может обозначать цифровой сигнальный процессор, который выполняет ЦОС. ЦОС является одной из наиболее распространенных задач, для решения которых применяются ПЛИС-микросхемы.
EEPROM (Electrically Erasable Programmable Read-Only Memory) - тип полупроводниковой энергонезависимой памяти, содержимое которой может быть многократно электрически запрограммировано пользователем. Стирание содержимого производится электрически. Значительная часть современных ПЛИС-микросхем построена с использованием технологии EEPROM, а её модификация лежит в основе Flash-памяти.
EPROM (Erasable Programmable Read-Only Memory) - тип полупроводниковой энергонезависимой памяти, содержимое которых может быть многократно электрически запрограммировано пользователем. Содержимое может быть стёрто с помощью внешнего ультрафиолетового излучения, попадающего на кристалл через кварцевое окно в корпусе микросхемы. В среднем для стирания информации требовалось порядка 20-30 минут.

Embedded processing - вычисления, проводимые в рамках встраиваемой системы. Наряду с ЦОС, для таких вычислений часто применяют ПЛИС-микросхемы. Встраиваемой системой (ВС) называется компьютерная система, которая является частью некоторого устройства или другой системы. Характерными чертами ВС являются небольшие размеры и энергопотребление. Обычно ВС предназначена для решения узкого класса задач. ВС используются во многих мобильных устройствах, таких как автомобили, спутники, мобильные телефоны, часы и т.д.

Flash –память –тип полупроводниковой энергонезависимой памяти, содержимое которых может быть многократно электрически запрограммировано пользователем, разновидность EEPROM. Основное отличие – использование технологии блочного стирания информации, за счет чего увеличена емкость микросхем.

FPGA (Field Programmable Gate Array) - разновидность ПЛИС, содержащая логические элементы и блоки коммутации. Программа для FPGA хранится в распределённой оперативной памяти микросхемы, поэтому требуется начальный загрузчик.

FPSC (Field Programmable System Chip) - устройство, представляющее собой объединение на одном кристалле FPGA и встроенного ASIC-ядра. На данный момент основным производителем устройств данного типа является компания Lattice Semiconductor.

GAL (Generic Array Logic) - патентованное название устройств PAL компании Lattice Semiconductor, которые представляют собой их более сложные электрически стираемые КМОП-разновидности.

IP-ядро (Intellectual Property core) - логический блок или блок данных, используемый в ПЛИС- и ASIC-микросхемах, который реализует некоторую законченную функциональность. IP-ядро представляет собой программную или аппаратную реализацию некоторого компонента системы, например, центрального процессорного устройства, Ethernet-контроллера или PCI-интерфейса.

LUT (LookUp Table, LUT-таблица) - таблица соответствия, с помощью которой реализуется логический блок в FPGA-микросхемах. Группа поступающих сигналов (соответствующих входным переменным функции, реализуемой данной таблицей) используется в качестве индекса в таблице соответствия и однозначным образом определяют некоторое выходное значение.

LVDS (Low Voltage Differential Signaling) - передача информации дифференциальными сигналами малых напряжений. Это направление передачи данных использует очень малые перепады дифференциального напряжения (до 350 мВ) на двух линиях печатной платы или сбалансированного кабеля. Малые перепады уровня и токовый режим выхода передатчика обеспечивают малый уровень шума и очень малую потребляемую мощность во всём диапазоне скоростей передачи. Технология LVDS отражена в двух стандартах: ANSI/TIA/EIA(Telecommunications

Industry Association/Electronic Industries Association)-644 (LVDS) и IEEE (Institute for Electrical and Electronics Engineering) 1596.3. Стандарт ANSI/TIA/EIA рекомендует максимальную пропускную способность в 655 Мб/сек. и оговаривает теоретический максимум в 1.923 Гб/сек. В настоящее время издание стандарта 644-ой версии пересмотрено и дополнено информацией о работе на множество приёмников. Пересмотренный стандарт, известный как TIA-644-A, утверждён в 2000 г.

PAL (Programmable Array Logic) - программируемые логические устройства наподобие ППЗУ. Однако, в отличие от последних, в данных устройствах массив элементов И является программируемым, а массив ИЛИ - нет.

PLA (Programmable Logic Array) - программируемые логические устройства наподобие ППЗУ. Однако, в отличие от последних, в данных устройствах оба массива элементов И и ИЛИ являются программируемыми.

PLD (Programmable Logic Device) - программируемые логические устройства, которые реализуют функции, необходимые для решения поставленных задач, в виде совершенной дизъюнктивной нормальной формы (совершенной ДНФ).

PROM (Programmable Read-Only Memory) - тип полупроводниковой энергонезависимой памяти, содержимое которой может быть однократно электрически запрограммировано пользователем. Такое устройство, поступившее от производителя, изначально находится в незапрограммированном состоянии, и перед его использованием пользователь может один раз провести его настройку под нужную ему задачу. Однако после этого архитектура микросхемы не может быть изменена. Память такого типа иногда применяется в ПЛИС-микросхемах.

SRAM (Static Random Access Memory) - полупроводниковое устройство оперативной памяти с произвольным доступом. SRAM-память является энергозависимой, поскольку при отключении питания данные теряются. Однако при наличии питания значение, записанное в ячейку такой памяти, будет находиться там до тех пор, пока на его место не будут записаны другие данные, поэтому такая память называется статической. SRAM-память часто применяется в FPGA-микросхемах.

ULA (Uncommitted Logic Array) - нескоммутированная вентильная матрица. Такое название носили первые вентильные матрицы на КМОП-технологии, которые стали появляться в середине 70-х годов.

Verilog - это язык описания аппаратуры, используемый для описания и моделирования электронных систем. Этот язык (также известный как Verilog HDL) позволяет осуществить проектирование, верификацию и реализацию (например, в виде СБИС) аналоговых, цифровых и смешанных электронных систем на различных уровнях абстракции. Разработчики Verilog сделали его синтаксис очень похожим на синтаксис языка C, что упрощает его освоение. Verilog имеет препроцессор, очень похожий на препроцессор языка C, и основные управляющие конструкции "if", "while" также подобны

одноимённым конструкциям языка С. Соглашения по форматированию вывода также очень похожи.

VHDL (Very high speed integrated circuits Hardware Description Language) - язык описания аппаратуры высокоскоростных интегральных схем. VHDL является базовым языком при разработке аппаратуры современных вычислительных систем.

Базовый модуль - структурная единица многопроцессорной системы, которая конструктивно представляет собой многослойную печатную плату, содержащую несколько программируемых логических интегральных схем (ПЛИС). Наращивание вычислительной мощности системы осуществляется за счет соединения нескольких базовых модулей между собой.

Вентильная матрица - заказная специализированная микросхема (ASIC), которая представляет собой массив изготовленных заводским способом несоединенных между собой компонентов (транзисторов и резисторов), организованных в группы, называемые базисными ячейками. Разработчику предоставляется набор доступных базисных ячеек, с помощью которого он описывает функции устройства, указывая соединение ячеек между собой. Затем на основе полученных результатов изготавливаются фотошаблоны для реализации связей в вентильных матрицах.

Встраиваемая система (ВС) - компьютерная система, которая является частью некоторого устройства или другой системы. Характерными чертами ВС являются небольшие размеры и энергопотребление.

КРП (контроллер распределённой памяти) - управляющий элемент секции макропамяти базового модуля ПЛИС. Система команд КРП в сочетании с возможностями коммутационного элемента секции макропамяти обеспечивает возможность организации процедур доступа чипов ПЛИС к информационным массивам, расположенным в распределенной памяти базового модуля ПЛИС. Как видно по наличию системы команд, КРП должен иметь свою управляющую программу, согласованную с управляющей программой чипов ПЛИС базового модуля.

МНМС (Модульно-Наращиваемые Многопроцессорные Системы) - вычислительные системы, обеспечивающие пользователю возможности динамического программирования архитектуры под структуру решаемой задачи. МНМС включает в себя реконфигурируемый вычислитель, состоящий из нескольких базовых модулей, и хост-машину. Каждый базовый модуль представляет собой многопроцессорную вычислительную систему со структурно-процедурной организацией вычислений. Базовые модули строятся на основе ПЛИС-технологий и по тем же архитектурным принципам, что и система в целом.

ПАИС (Программируемая Аналоговая Интегральная Схема) - электронный компонент, используемый для создания аналоговых и аналого-цифровых устройств. В отличие от обычных аналоговых микросхем, соединения элементов ПАИС не определяется при изготовлении, а задаётся посредством программирования.

ПЛИС (Программируемая Логическая Интегральная Схема) - электронный компонент, используемый для создания цифровых устройств. В отличие от обычных цифровых микросхем, логика работы ПЛИС не определяется при изготовлении, а задаётся посредством программирования.
ПЛМ (Программируемая Логическая Матрица) - устройства, которые реализуют функции из И и ИЛИ массивов, оба из которых программируемы.
ПЗУ (Постоянное Запоминающее Устройство) - цифровое устройство, организованное как память данных, которые не могут быть изменены пользователем, и сохраняющее информацию при выключении питания.
ППЗУ (Программируемое Постоянное Запоминающее Устройство) - программируемое логическое устройство, в котором любая функция реализуется в виде ДНФ (Дизъюнктивной Нормальной Формы). ППЗУ состоит из фиксированного массива логических функций И, подсоединенного к программируемому массиву логических функций ИЛИ. Входы устройства подаются сразу на вход массиву И, а выходы с массива ИЛИ являются выходами самого устройства. Обычно термин ППЗУ относится к устройствам PROM-памяти.
РВС (Реконфигурируемые Вычислительные Системы) - вычислители, обладающие возможностью изменения (реконфигурации) архитектуры информационных связей между модулями в процессе функционирования для реализации алгоритмов с различной топологией.
Трансивер - устройство для приёма и передачи сигнала, часто используемое для соединения ПЛИС-микросхем с внешними устройствами.
Элементарный процессор (ЭП) - функциональное устройство, обладающее системой команд и выполняющее операции над данными. Фактически ЭП - это АЛУ с входными и выходными буферными устройствами, небольшой внутренней памятью для хранения констант, системами управления и коммутации. ЭП выполняет арифметические операции над потоками приходящих на его входы данных, результат которых передается на один или несколько выходов ЭП.

Миссия университета – генерация передовых знаний, внедрение инновационных разработок и подготовка элитных кадров, способных действовать в условиях быстро меняющегося мира и обеспечивать опережающее развитие науки, технологий и других областей для содействия решению актуальных задач.

КАФЕДРА СИСТЕМ УПРАВЛЕНИЯ И ИНФОРМАТИКИ

Кафедра систем управления и информатики (до 2001 года автоматики и телемеханики) была образована в 1945 году как подразделение основанного в тот же год факультета Электроприборостроения ЛИТМО и именовалась кафедрой Электроприборостроения (№80). Основание кафедры связано с именем ее первого заведующего и первого декана факультета Электроприборостроения профессора Марка Львовича Цуккермана. Профессор М.Л. Цуккерман в 1913–м году закончил электромеханический факультет Санкт-Петербургского политехнического института им. Петра Великого, в двадцатые годы организовал в Ленинграде отраслевую лабораторию электроизмерений (ОЛИЗ) и был известен в стране как крупный специалист в области систем телеизмерений. С 1933–го по 1935–й год профессор М.Л. Цуккерман руководит кафедрой « Автоматизации и телемеханизации» ЛЭТИ им. В.И. Ульянова (Ленина). В 1935–м году профессор М.Л. Цуккерман вплоть до начала Великой отечественной войны находится в научной командировке в Европе.

В отличие от существовавших к тому моменту кафедр аналогичного профиля в ЛПИ им. М.И. Калинина и ЛЭТИ им. В.И. Ульянова (Ленина), на кафедру автоматики и телемеханики ЛИТМО была возложена задача подготовки специалистов по автоматизации приборостроительной, оптической и оборонной промышленности, автоматических систем управления, систем телемеханики и телеизмерений. Осенью 1945 года кафедра провела первый набор студентов по специальности электроприборостроение. В 1947–м году кафедра претерпевает первое изменение своего названия, после которого называется кафедрой Автоматики и телемеханики (№80 вплоть до XX–го съезда КПСС). Первый выпуск инженеров–электромехаников по специальности «приборы автоматики и телемеханики» состоялся уже в 1948 году и составил 17 человек. По временной хронологии это событие совпало в выходом в свет на английском языке известной книги Норберта Винера "Кибернетика или наука об управлении и связи в машинах, живом организме и обществах", в которой дается обоснование кибернетического подхода, выдвигающего на передний план информационное содержание природных, социальных и технических

процессов и рассматривающего проблемы автоматического управления с точки зрения преобразования, передачи и использования информации. Советская научная общественность познакомится с этой книгой в переводе на русский язык только в 1958–м году.

Профессор М.Л. Цуккерман руководил кафедрой с 1945 по 1959 год. К своей работе кафедра приступила, имея преподавательский состав, включавший профессора Д.И. Зорина, доцентов Е.А. Танского и Р.И. Юргенсона и заведующего лабораторией А.А. Мезерина. В пятидесятые годы в преподавательский состав кафедры вошли профессор А.А. Кампе-Немм, доцент Г.А. Тацитов, старшие преподаватели В.А. Борисов, В.Г. Новиков и В.В. Соколов. К концу пятидесятых годов преподавательский состав пополнился выпускниками ЛИТМО доцентом Н.М. Яковлевым, старшими преподавателями Л.Т. Никифоровой, Н.М. Перевозчиковым, Ю.Б. Ганту и ассистентом А.М. Шпаковым, а также доцентом Б.А. Арефьевым.

В 1955 году при кафедре образована научно-исследовательская лаборатории (НИЛ). В этот период основные направления научно-исследовательских работ представляли задачи автоматизации измерения и регистрации параметров кораблей во время их мореходных испытаний, а также стабилизации скорости и фазирования двигателей. Под научным руководством проф. М.Л. Цуккермана была налажена подготовка научных кадров высшей квалификации через систему аспирантуры.

С 1959 года по 1970 кафедру возглавлял ученик М.Л. Цуккермана, выпускник кафедры Автоматики и телемеханики ЛЭТИ им. В.И. Ульянова (Ленина) 1936 года, доцент Евфимий Аполлонович Танский. За время его руководства профессорско–преподавательский состав пополнился старшим преподавателем Л.Л. Бориной, доцентами А.И. Новоселовым и И.П. Пальтовым, пришедшими из промышленности и высших военных учебных заведений, а также выпускниками кафедры, успешно закончившими обучение в ее аспирантуре, доцентами В.Н. Дроздовым, А.В. Ушаковым, В.А. Власенко, и ассистентом И.Н. Богоявленской. В этот период защитили диссертации на соискание ученой степени доктора технических наук доценты Б.А. Арефьев и Р.И. Юргенсон. В научно-исследовательской работе на кафедре произошел заметный поворот к проблемам автоматизации оптико–механического приборостроения, что привело к длительному научно – техническому сотрудничеству кафедры с ЛОМО им. В.И. Ленина, в рамках которого для нужд оборонной техники была разработана целая гамма прецизионных фотоэлектрических следящих систем. В рамках научно-технического сотрудничества с НИИЭТУ кафедра приняла участие в разработке автоматической фототелеграфной аппаратуры, реализованной в виде комплекса "Газета-2".

С 1970 по 1990 год кафедрой руководил известный в стране специалист в области автоматизированного электропривода и фотоэлектрических следящих систем доктор технических наук, профессор Юрий Алексеевич Сабинин. В эти годы заметно изменилась структура дисциплин и курсов,

читаемых студентам кафедры. К традиционным курсам "Теория автоматического регулирования и следящие системы", "Теория автоматического управления, экстремальные и адаптивные системы", "Элементы автоматики" и "Телемеханика" были добавлены дисциплины: "Теоретические основы кибернетики", "Локальные системы управления", "САПР систем управления" и другие. Коллектив преподавателей пополнился новым отрядом выпускников ее аспирантуры: доцентами Ю.Л. Тихоновым, В.В. Лаврентьевым, В.В. Григорьевым, В.В. Хабаловым, Л.С. Громовой, В.И. Бойковым, С.В. Быстровым, А.Б. Бушуевым, А.Н. Коровьяковым, И.В. Мирошником, Ю.П. Котельниковым, Г.И. Болтуновым, старшим преподавателем И.П. Салмыгиным. Из промышленности и других подразделений института пришли на кафедру доценты И.Ю. Рогинский, П.В. Николаев, И.П. Болтунов. Приобрела устойчивый характер система подготовки кадров высшей квалификации. В период с 1970–го по 1990–й защитили диссертации на соискание ученой степени доктора технических наук доценты И.П. Пальтов, В.В. Григорьев и В.Н. Дроздов. Более 40 человек успешно завершили обучение в аспирантуре.

Прикладные разработки кафедры были связаны с задачами адаптивной оптики для многоэлементных зеркал оптических телескопов и коррекции волнового фронта технологических лазеров; с задачами адаптивной радиооптики применительно к проблеме управления большими полноповоротными радиотелескопами; с задачами автоматизации обработки снимков в пузырьковых камерах; гребного электропривода и робототехнических систем, автоматического управления процессом мягкой посадки летательных аппаратов. Новый облик теории управления 1970 годов, внедрение метода пространства состояний и вычислительной техники, повышение математического уровня научных исследований нашли отражение в научных разработках кафедры, многочисленных трудах и монографиях. В эти годы интенсивно разрабатываются проблемы теории многомерных динамических систем, качественная теория устойчивости, методы согласованного и многорежимного управления, положено начало теоретическим работам в области робототехники. Научное руководство перечисленными работами осуществляли профессора кафедры Ю.А. Сабинин, В.Н. Дроздов, А.В. Ушаков, В.В. Григорьев и И.В. Мирошник .

С 1990 года по 1995–й год кафедра переживает «смутное время» на уровне руководства ею, но не на уровне интеллектуальной обстановке в ее коллективе. Известно высказывание ректора НИУ ИТМО: «Интересно, на кафедре автоматики нет номинального заведующего вот уже столько лет и ни одного скандала». Лучшего комплимента кафедре не придумаешь. С 1990 года по 1992–й обязанности заведующего кафедрой исполнял профессор В.В.Григорьев, в 1992–м году в результате проведенного конкурса заведующим кафедрой автоматики и телемеханики становится профессор Таганрогского радиотехнического института Анатолий Аркадьевич Колесников, известный специалист в области синергетики. К сожалению, по

причинам личного характера он так и не покинул Таганрог и не приступил к руководству кафедрой автоматики и телемеханики ЛИТМО. В 1994–м году его заведование руководством института приостанавливается, объявляется новый конкурс, в результате которого с 1995–го года по 2010–й кафедрой руководил ее воспитанник доктор технических наук, профессор Валерий Владимирович Григорьев, по инициативе которого в 2001–м году кафедра получила название кафедры «Систем управления и информатики». В эти годы профессорско-преподавательский состав пополнился профессором Е.Ф. Очиным (1993-1996 годы), а также выпускниками аспирантуры ИТМО В.В. Черноусовым, А.П. Баевым, В.О. Никифоровым, М.С. Чежиным, А.В. Ляминым, А.А. Бобцовым и К.А. Сергеевым. Продолжала эффективно работать система подготовки кадров высшей квалификации, диссертации на соискание ученой степени доктора технических наук защитили И.В. Мирошник, Р.О. Оморов, А.В. Ушаков, А.И. Скалон, В.О. Никифоров, А.А. Бобцов.

Помимо традиционной подготовки инженеров–электриков была начата подготовка бакалавров по направлению "Управление и автоматизация". С введением локальной сети и подключением к Интернет проведена модернизация компьютерного класса и учебных лабораторий. Научно - исследовательская работа ведется по целевым программам и конкурсным проектам РФФИ, Минобразования и Администрации Санкт-Петербурга. Завершилось формирование научной школы кафедры и ее основных направлений, возглавляемых профессорами В.В. Григорьевым, А.В. Ушаковым, И.В. Мирошником, В.О. Никифоровым и доцентом В.И. Бойковым. С целью расширения исследований, проводимых по теории нелинейных и адаптивным систем, роботов и микропроцессорной техники, а также активизации подготовки кадров в 1994 году образована научная лаборатория Кибернетики и Систем управления (руководитель проф. И.В. Мирошник). С 1994 года существенно расширились международные контакты кафедры, участие в международных научных мероприятиях, организации конференций и симпозиумов. Профессора кафедры Григорьев В.В., Мирошник И.В, Ушаков А.В. , а позднее и Никифоров В.О. становятся действительными членами (академиками) Международной Академии нелинейных наук.

В феврале 2010 года заведующим кафедрой Систем управления и информатики был избран выпускник кафедры 1996–го года декан факультета компьютерных технологий и управления, доктор технических наук, профессор Алексей Алексеевич Бобцов, А.А. Бобцов является также председателем Совета молодых ученых и специалистов при Правительстве Санкт–Петербурга, действительным членом академии Навигации и управления движением и членом научного совета РАН по теории управляемых процессов и автоматизации.

В последние годы профессорско-преподавательский состав кафедры пополнился молодыми кадрами: доцентами Кремлевым А.С., Чепинским

С.А. (выпуск кафедры 2002-го года), Дударенко Н.А., Нуйей (Осипцевой) О.С., Николаевым Н.А., Слитой О.В. (выпуск кафедры 2003-го года), Герасимовым Д.Н. (выпуск кафедры 2005-го года), Арановским С.В., Блинниковым А.А. (выпуск кафедры 2006-го года), Сержантовой (Поляковой) М.В. (выпуск кафедры 2007-го года), Пыркиным А.А. (выпуск кафедры 2008-го года), Колюбиным С.А. (выпуск кафедры 2010-го года). К участию в подготовке магистров подключились профессора из санкт-петербургских университетов Фрадков А.Л., Андриевский Б.Р., Тертычный В.Ю. и Фуртат И.Б.

В настоящее время кафедра является одним из ведущих российских научных и образовательных центров, ориентированным на фундаментальные и прикладные исследования в области автоматических систем и прикладной информатики, подготовку высококвалифицированных специалистов XXI–го столетия. На кафедре функционируют четыре научно-исследовательские группы: «Технической кибернетики» (основатель профессор И.В. Мирошник, научный руководитель профессор А.А. Бобцов), «Автоматизированного оптоэлектронного мониторинга технических объектов и комплексов» (основатели профессор Ю.А. Сабинин и доцент П.В. Николаев, научные руководители – доцент В.И. Бойков и профессор А.В. Ушаков) и «Технической информатики и телемеханики (основатель профессор М.Л. Цуккерман, научный руководитель профессор А.В. Ушаков), «Интеллектуальной робототехники» (основатель и научный руководитель профессор А.А.Бобцов). Усилиями ученых кафедры на кафедре создана научная школа « Управление в условиях системных неопределенностей», при кафедре вот уже второе десятилетие проводятся ежегодные «Крещенские научные чтения», имеющие статус городского семинара по теории управления.

Ученые кафедры издают монографии, печатаются в журналах академий наук РФ и стран бывшего СССР, отраслевых журналах, известиях высших учебных заведений, а также зарубежных журналах и трудах международных конференций. Сотрудниками кафедры опубликовано более 120 монографий и учебников, 250 методических и учебных пособий, 3500 статей, из них более 380 в журналах академий наук, около 300 статей и докладов в зарубежных научных изданиях. Ученые кафедры являются авторами более 600 изобретений, постоянно принимают участие в работе российских и зарубежных семинаров, конференций и конгрессов. Кафедра поддерживает контакты с 20 техническими зарубежными университетами.

На восьмом десятке своего существования кафедра систем управления и информатики представляет собой работоспособный коллектив, полный новых идей и творческих планов.