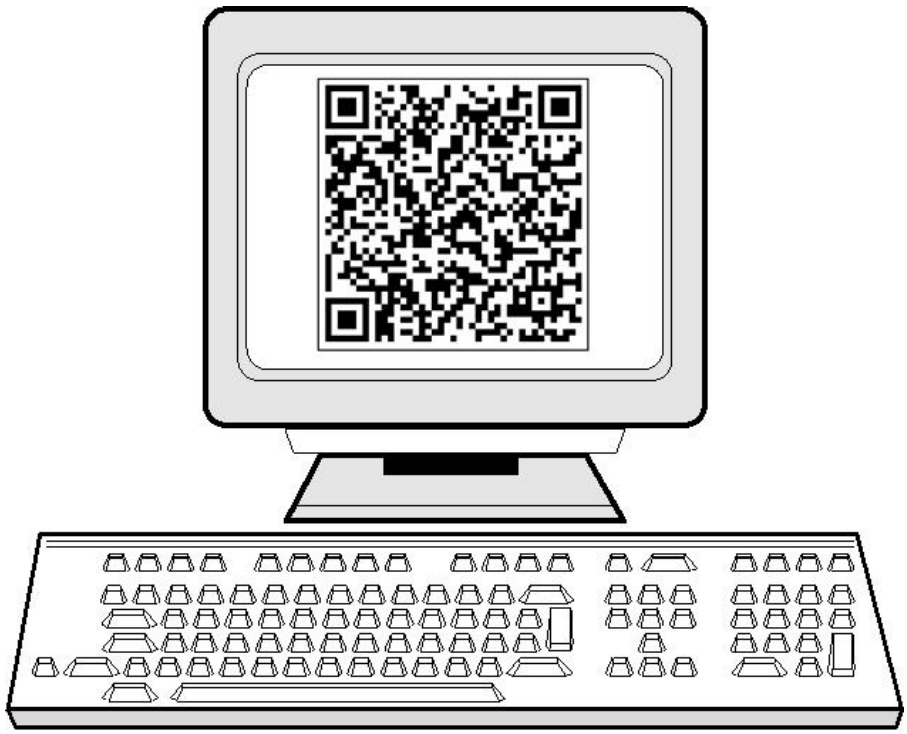
УНИВЕРСИТЕТ ИТМО

В.И. Бойков, Г.И. Болтунов, С.В. Быстров, В.В. Григорьев, Ю.В. Литвинов

ЦИФРОВАЯ ТЕХНИКА СИСТЕМ УПРАВЛЕНИЯ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ



Санкт-Петербург
2019

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

УНИВЕРСИТЕТ ИТМО

В.И. Бойков, Г.И. Болтунов, С.В. Быстров, В.В. Григорьев, Ю.В. Литвинов

ЦИФРОВАЯ ТЕХНИКА СИСТЕМ УПРАВЛЕНИЯ. ЛАБОРАТОРНЫЙ ПРАКТИКУМ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ВЫПОЛНЕНИЮ ЛАБОРАТОРНЫХ РАБОТ

РЕКОМЕНДОВАНО К ИСПОЛЬЗОВАНИЮ В УНИВЕРСИТЕТЕ ИТМО
по направлению подготовки 27.04.03, 27.04.04
в качестве практикума для реализации основных профессиональных образовательных программ высшего образования магистратуры

Санкт-Петербург
2019

Бойков В.И., Болтунов Г.И., Быстров С.В., Григорьев В.В., Литвинов Ю.В., Цифровая техника систем управления. Лабораторный практикум– СПб: Университет ИТМО, 2019. – 58 с.

Рецензент(ы):
Николаев Николай Анатольевич, кандидат технических наук, доцент (квалификационная категория "ординарный доцент") факультета систем управления и робототехники Университета ИТМО.

В пособии изложены методические указания к выполнению лабораторных работ по дисциплине "Цифровая техника систем управления". Приведено краткое описание программной среды выполнения лабораторных работ (Multisim 11,0), приведены примеры реализации учебных заданий.

Пособие предназначено для студентов технических университетов, обучающихся по направлениям магистерской подготовки 27.04.03- «Системный анализ и управление» , 27.04.04 –«Управление в технических системах». Пособие может быть полезно студентам других технических направлений, а также аспирантам, преподавателям вузов и специалистам, научная и практическая деятельность которых связана с решением вопросов проектирования цифровых систем управления.

УНИВЕРСИТЕТ ИТМО

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

Содержание

ВВЕДЕНИЕ

Лабораторный практикум по дисциплине «Цифровая техника систем управления» направлен на выработку у студентов навыков схемотехнического моделирования работы цифровых устройств управления и обработки информации. Наличие таких навыков способствует формированию профессиональных компетенций, связанных со способностью применять современные теоретические и экспериментальные методы разработки математических моделей цифровых систем управления и со способностью разрабатывать экспериментальные макеты цифровых управляющих и информационных модулей систем управления, проводить их исследование с применением современных информационных технологий.

В лабораторных работах №1 – №3 выполняется моделирование динамики цифровых устройств с учетом эффектов квантования времени и уровней сигналов. В работе №1 основное внимание уделяется схемотехнической реализации регистрового процессора. Следует обратить внимание на возможность существования квазиустойчивого установившегося процесса в неустойчивой системе. Работы №2 и №3 следует выполнять по единому варианту. В работе №2 выполняется синтез и исследуется динамика дискретного (импульсного) наблюдателя состояния для непрерывного объекта. В работе №3 исследуется динамика цифрового наблюдателя с учетом эффекта квантования уровней сигналов. Из-за сложности реализации операции умножения при моделировании в пакете Multisim обе эти работы выполняются средствами Matlab Simulink. Предполагается, что с пакетом Matlab студенты знакомятся ранее при выполнении лабораторных работ по дисциплине «Теория автоматического управления».

В лабораторных работах №4 - №6 выполняется синтез и схемотехническое моделирование цифровых управляющих устройств, построенных на базе микропрограммных автоматов. При выполнении этих работ следует уделить особое внимание взаимосвязи кода разработанной микропрограммы и ее аппаратной поддержки.

Общая трудоемкость выполнения лабораторных работ составляет 16 аудиторных часов. Трудоемкость выполнения лабораторных работ №1 и №4 составляет 2 часа, остальных – 3 часа.

МОДЕЛИРОВАНИЕ ЦИФРОВЫХ УСТРОЙСТВ В ПРОГРАММНОЙ СРЕДЕ NI MULTISIM 11.0

1 Запуск программной среды Multisim

Моделирование работы электронных устройств выполняется средствами Multisim фирмы National Instruments [7].

Запуск программы осуществляется по следующему пути: *Пуск -> Все программы -> National Instruments -> Circuit Design Suite 11.0 -> Multisim 11.0.* Если окно нового проекта отсутствует, то необходимо создать новый проект: *File->New->Design (Файл -> Новый -> Создать схему)* или нажать комбинацию клавиш *Ctrl+N* . Окно нового проекта показано на рисунке 1.

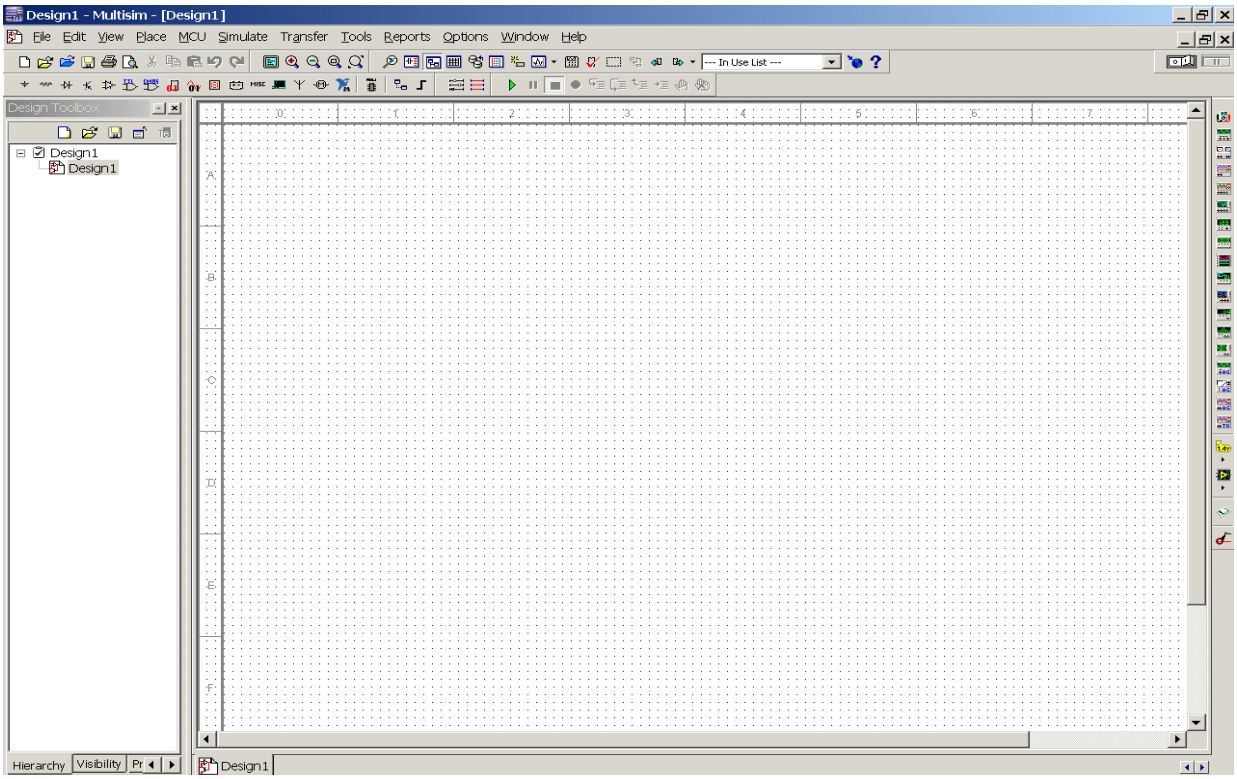


Рисунок 1 – Окно программы Multisim 11.0 с новым проектом Design1

2 Настройка вида отображения компонентов

Необходимо проверить и при необходимости изменить настройку отображения электронных компонентов по следующему пути: *Options -> Global Preferences ->Parts (Установки -> Основные установки -> Компоненты).* В открывшемся окне отметить вид отображения *DIN*. Нажать *OK*.

Отображение элементов по стандарту DIN близко к требованиям ГОСТ, стандарт ANSI существенно отличается от ГОСТ. Окно выбора стандарта отображения электронных компонентов показано на рисунке 2.

Замечание. Использование стандарта DIN в чертежах и документах какого-либо технического проекта запрещено. Отличительным признаком чертежа или документа служит наличие рамки и штампа в соответствии с действующими ГОСТ ЕСКД.

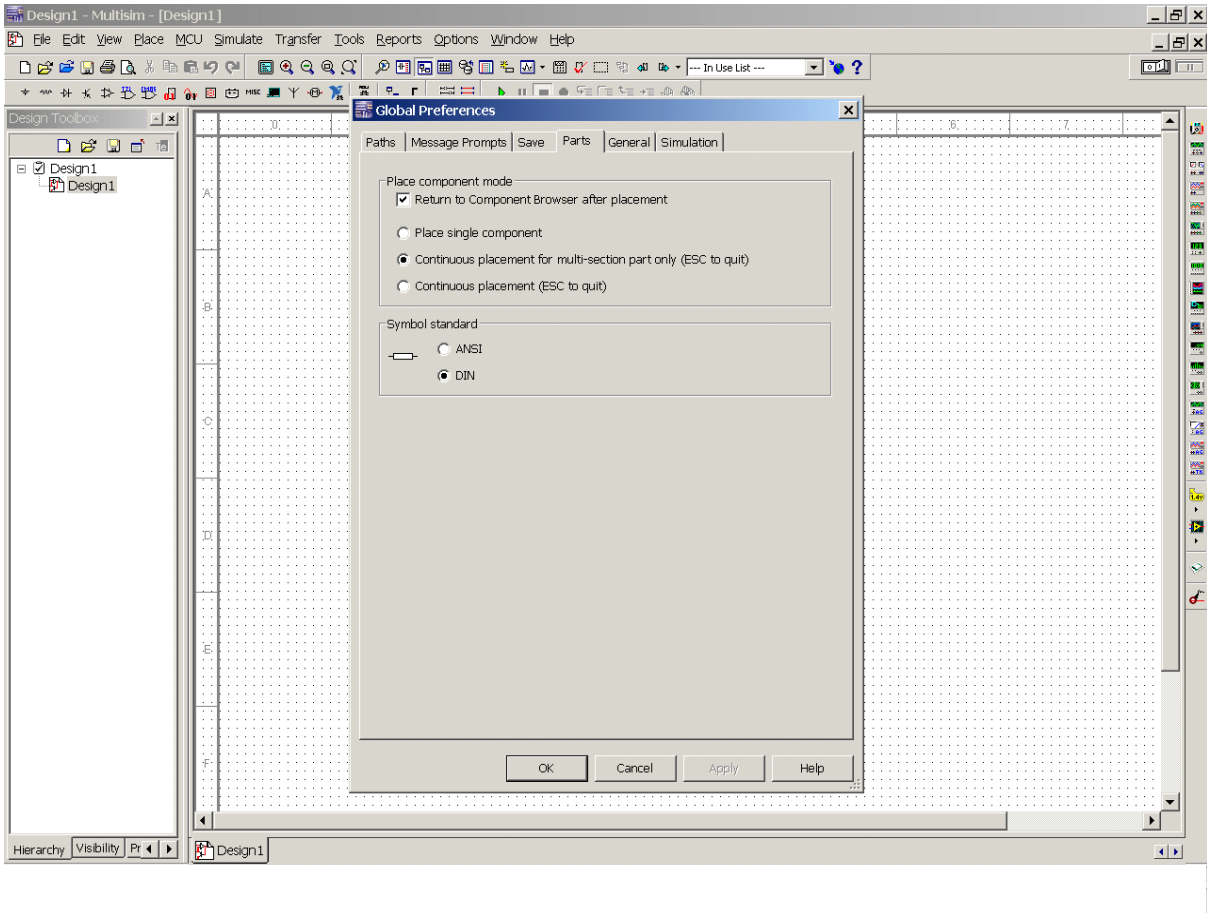


Рисунок 2 – Окно выбора стандарта отображения электронных компонентов

3 Выбор набора электронных компонентов

На панели инструментов *Component* выбрать пиктограмму с необходимым набором электронных компонентов, например *Place TTL* (т.е. выбрать *логические микросхемы ТТЛ)* или иной набор компонентов (Basic, CMOS, MSC и др.). (См. пример на рисунке 3). То же самое можно сделать с помощью команды главного меню *Place->Component … (Вставить -> Компонент …).*

4 Выбор семейства и типа компонента

На появившейся диалоговой панели *выбор компонента* (см. рисунок 4) в поле *семейство* выбираем класс компонента (например, TTL), в поле *компонент*

выбираем тип компонента (например, NAND2 – двухвходовой элемент 2И-НЕ). Нажимаем OK, и этот компонент появится на рабочем поле (он будет следовать за курсором мыши, пока вы не кликните на рабочем поле для его размещения). После размещения компонента на рабочем поле панель *выбор компонента* открывается снова, и можно выбрать и поместить на рабочее поле следующий компонент.

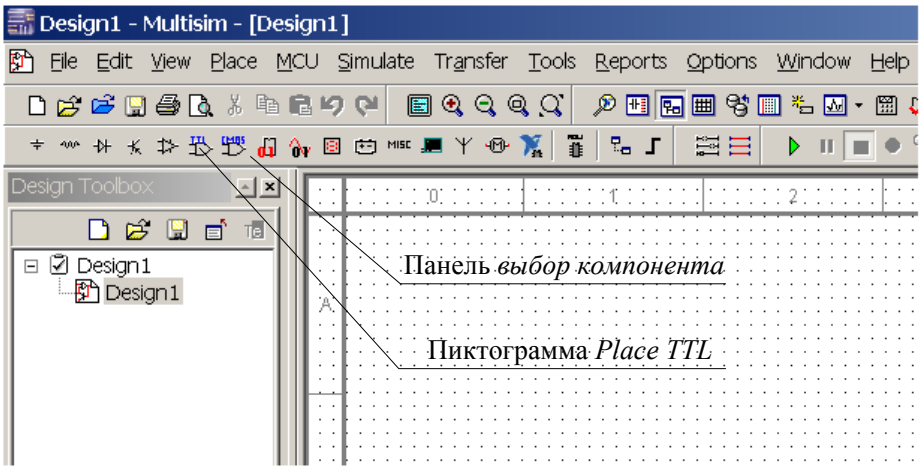


Рисунок 3 – Выбор требуемого набора компонентов

Для выполнения лабораторных заданий потребуются также постоянные запоминающие устройства из набора *MCU*, выбор которых описан в общей части лабораторной работы №4.

Логическая константа «0» реализуется с помощью компонента *DGND*, а логическая константа «1» – с помощью компонента *VCC*. Они находятся в наборе *Источники (Sources)*, семейство *Power Sources*.

5 Работа с компонентом на рабочем поле

Для выделения компонента на рабочем поле следует щелкнуть на его изображении левой кнопкой мыши (ЛКМ). Выделенный компонент можно перетаскивать мышью. Двойной щелчок на компоненте, либо щелчок правой кнопкой мыши (ПКМ), открывает диалоговое окно редактирования свойств компонента. Можно выделить группу компонентов, перемещая мышь с нажатой левой кнопкой. Группу выделенных компонентов можно перетаскивать.

Компонент можно поворачивать с помощью меню редактирования. Для этого компонент нужно предварительно выделить ЛКМ, затем нажать ПКМ и выбрать пункт меню: переворот по вертикали, переворот по горизонтали, поворот на 90 градусов.

Копирование объектов осуществляется нажатием ПКМ и выбором пункта меню *Copy* в меню редактирования. Перед копированием объект нужно

выделить, поместив на него курсор мыши и нажав ЛКМ. Если нужно скопировать несколько объектов, их можно выделить перемещением мышки с нажатой левой клавишей, выделяя прямоугольную область с объектами. При выполнении команды *копировать* выделенный объект копируется в буфер обмена. Для вставки содержимого буфера на рабочее поле нужно нажать правую кнопку мыши на рабочем поле и выбрать пункт меню *Paste*.

Удаление объектов выполняется командой *Delete* из меню редактирования (или клавишей *Delete*).

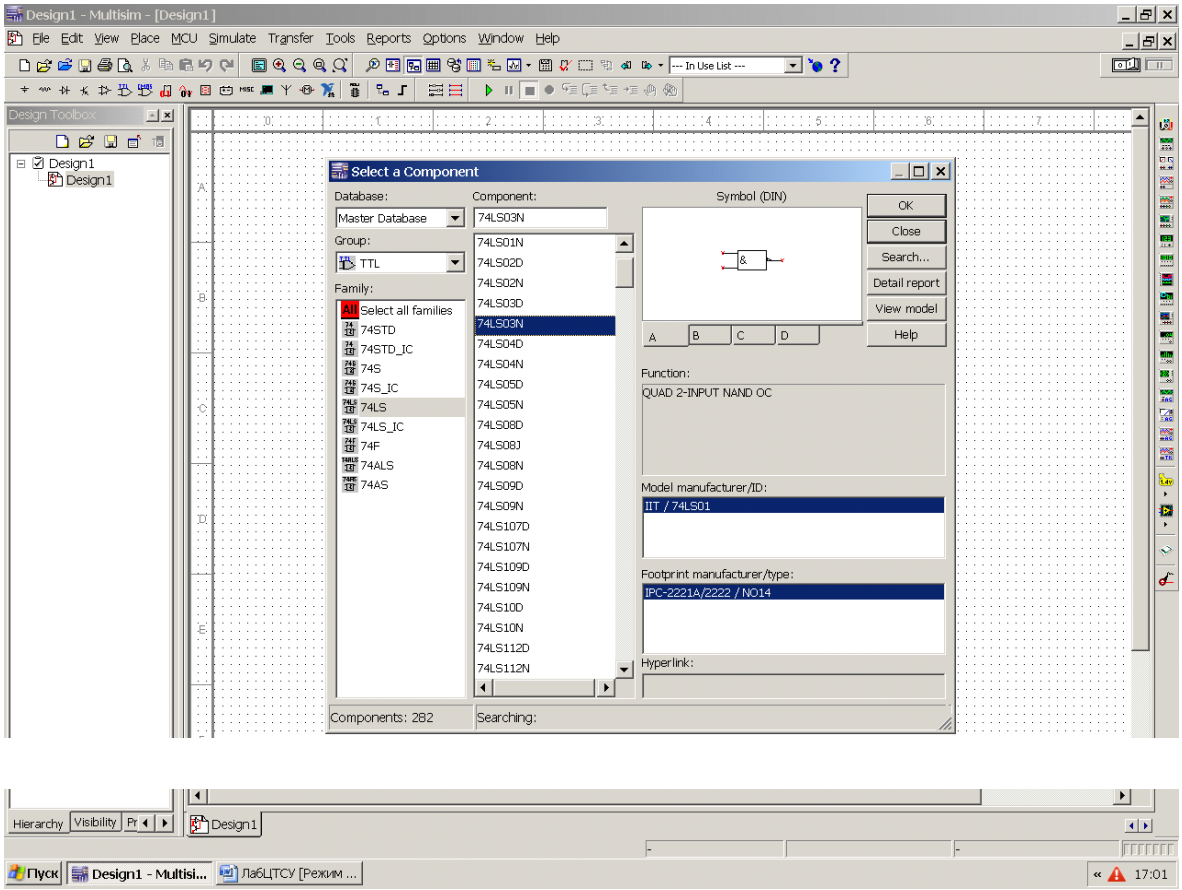


Рисунок 4 – Панель выбора требуемого компонента

6 Соединение компонентов в схему

Для соединения компонентов проводниками в соответствии с принципиальной схемой нужно подвести указатель мыши к выводу компонента (при этом на выводе появится черная точка). Нажмите ЛКМ. Переместите указатель мыши к выводу компонента, с которым нужно соединиться, и снова нажмите ЛКМ. Выводы компонентов соединятся проводником. Аналогично можно соединить вывод компонента с существующим проводником. При прокладке проводника по схеме каждое нажатие на ЛКМ фиксирует текущее положение проводника на

рабочем поле. Эта операция позволяет придать наглядность и повысить читаемость схемы.

Удаление проводника выполняется командой *Delete* из меню редактирования (или клавишей *Delete*). Предварительно проводник следует выделить, щелкнув по нему левой кнопкой мыши.

Каждый проводник в схеме имеет индивидуальное обозначение проводника – его номер, устанавливаемый по умолчанию. Номер можно заменить каким-либо смысловым обозначением. Для этого нужно сделать двойной щелчок ЛКМ на проводнике и в открывшемся окне изменить *Preferred net name (Назначенное имя цепи)* на новое значение.

7 Использование пробников для индикации

Для динамической индикации логического сигнала в процессе моделирования работы электронной схемы можно использовать *пробники.* Для этого необходимо выбрать компонент из раздела *Индикаторы (Indicators) -> Семейство PROBE -> Компонент PROBE_RED (пробник красный* или другого цвета*) -> OK* (см. рисунок 5). Необходимо поместить выбранный компонент на рабочем поле и подключить проводником к контролируемой цепи.

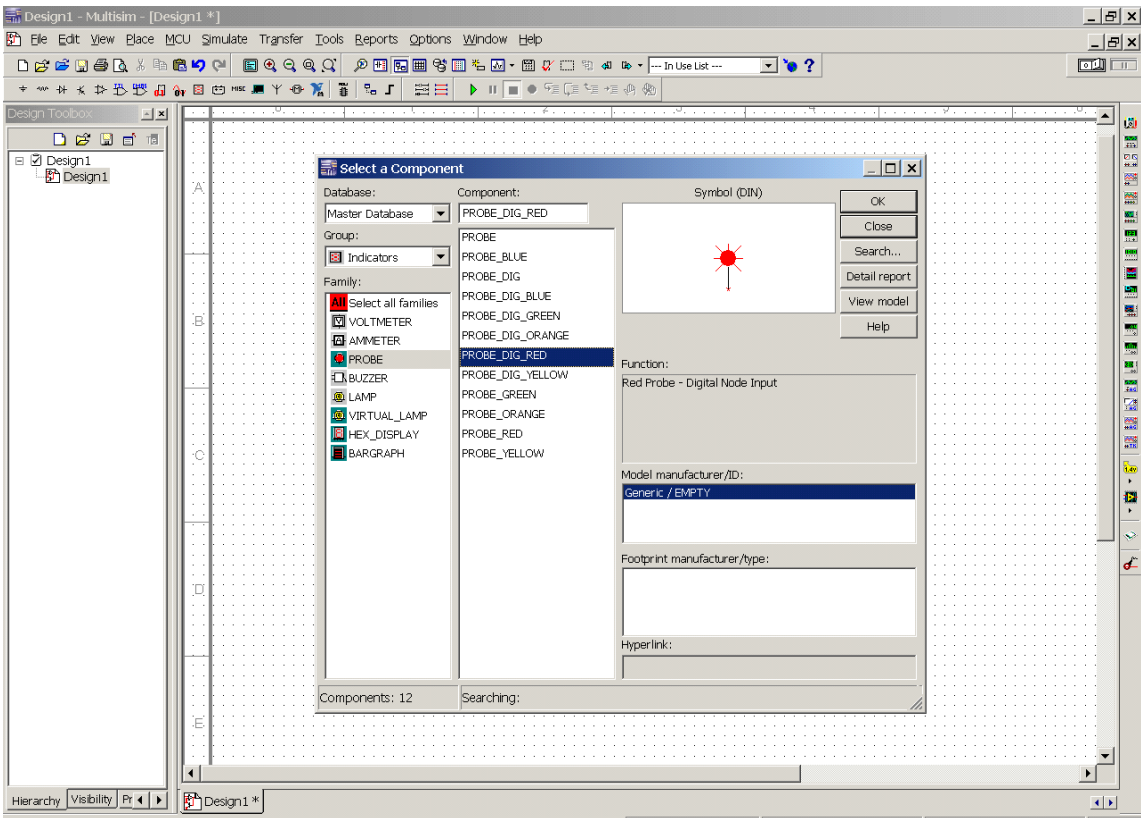


Рисунок 5 – Выбор пробника для индикации сигнала

8 Использование логического анализатора

Логический анализатор – виртуальный прибор, предназначенный для отображения временных диаграмм изменения логических значений сигналов. Его можно выбрать на панели инструментов (на рисунке 1 – правая вертикальная панель в окне программы). Щелкните на иконке *логический анализатор*, а затем поместите его изображение на рабочем поле. Можно выбрать *логический анализатор* и из пункта меню *Simulate ->Instruments ->Logic Analyzer (Моделирование -> Приборы -> Логический анализатор)*, после чего его компактное изображение появляется на рабочем поле. Компактное и развернутое изображения логического анализатора показаны на рисунке 6.

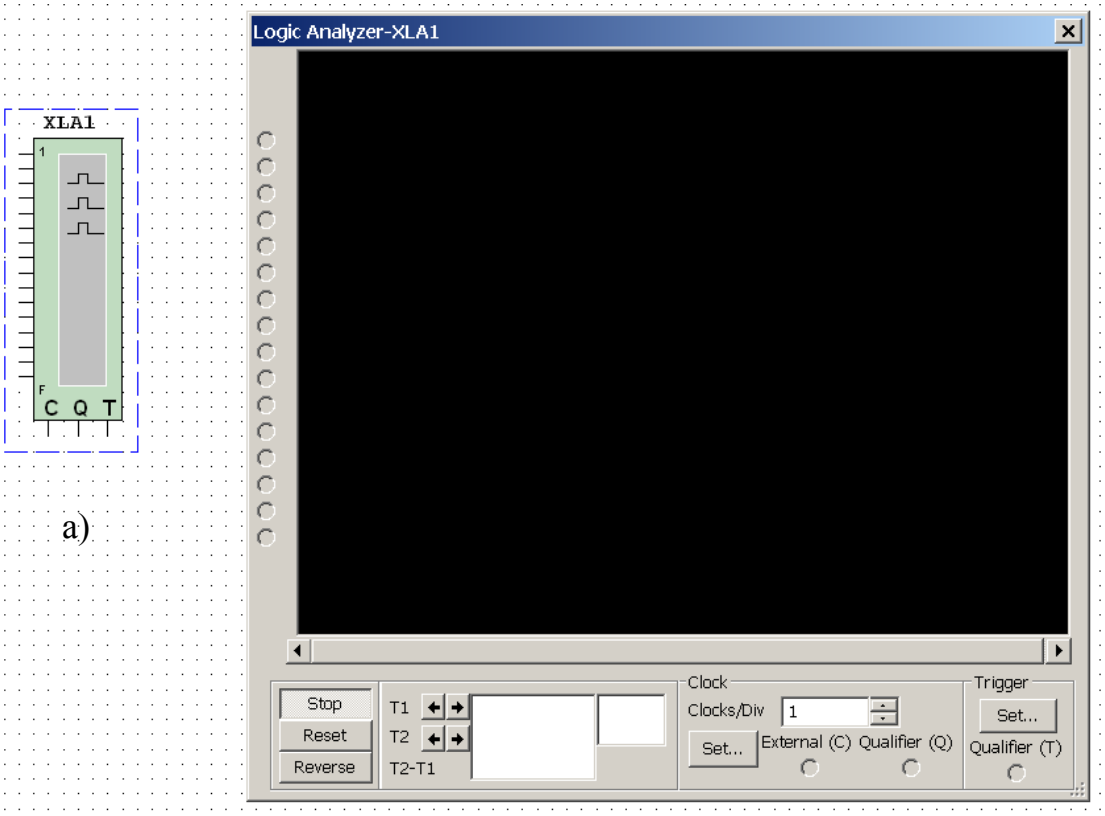
Логический анализатор подключается к схеме с помощью входов в его левой части, обозначенных цифрами 1 и F. Одновременно могут наблюдаться сигналы в 16 точках схемы. Для настройки логического анализатора и просмотра графиков сигналов следует сделать двойной щелчок на его компактном изображении, после чего появляется окно развернутого изображения с панелью настройки (см. рисунок 6). Эту операцию следует делать после того, как все компоненты моделируемой схемы соединены, к схеме подключены генератор слов и логический анализатор. Настройка логического анализатора в основном сводится к установке параметров горизонтальной развертки, для чего следует щелкнуть на кнопке *Set*…. Появляется диалоговое окно *Clock Setup (Установки синхронизации)* (см. рисунок 7).

Выбрать источник синхросигнала *Clock source -> External (*Источник -> Внешний). Если выбран внутренний источник синхронизации *(Internal)*, то значение тактовой частоты нужно установить так, чтобы оно было как минимум в 10 раз больше, чем частота синхронизации в генераторе слов (на рисунке 7 установлено значение 100 Гц). Затем нажать на клавишу *Accept (Применить)*.

Масштаб изображения по горизонтали в окне настройки логического анализатора регулируется параметром *Clock->Clock/Div (*Развертка -> Время/Дел). Кнопка *Reverse* переключает черный/белый цвет фона для диаграмм.

9 Формирование тактовых импульсов

При выполнении лабораторных работ может потребоваться генератор тактовых импульсов. В зависимости от цели исследования возможно формирование тактовых импульсов в ручном или автоматическом режимах.



а)

б)

Рисунок 6 – Компактное (а) и развернутое (б) изображения логического анализатора

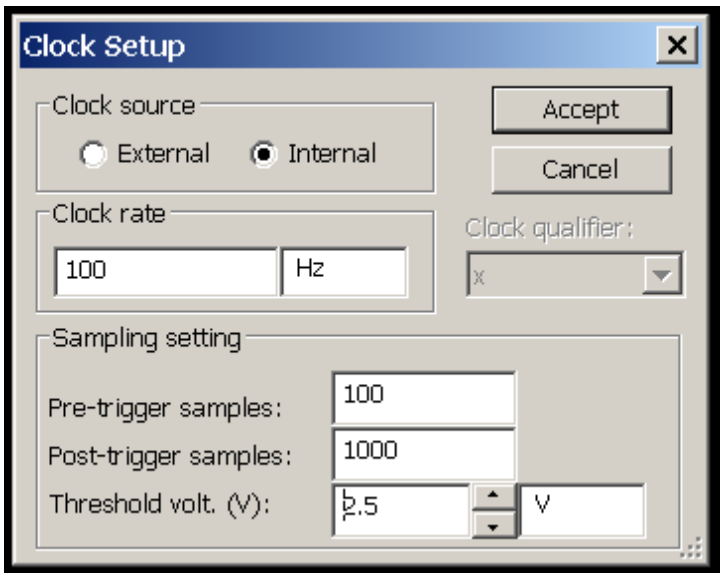


Рисунок 7 – Окно настройки синхронизации логического анализатора

В ручном режиме тактовые импульсы формируют, когда требуется исследовать работу схемы по шагам (тактам). В этом случае целесообразно в качестве генератора тактовых импульсов использовать кнопку, замыкаемую вручную. Такая кнопка имеется в компонентах раздела *Electro_Mechanical-> Momentary_Switches -> PB_NO -> OK* . Кнопку следует включить по схеме, показанной на рисунке 8. В схеме применен источник питания *VCC* из раздела *Source* (Источники) и резистор *R1* из раздела *Basic*. Клавиша, управляющая состоянием кнопки (на рисунке 8 это клавиша A) назначается в свойствах элемента.

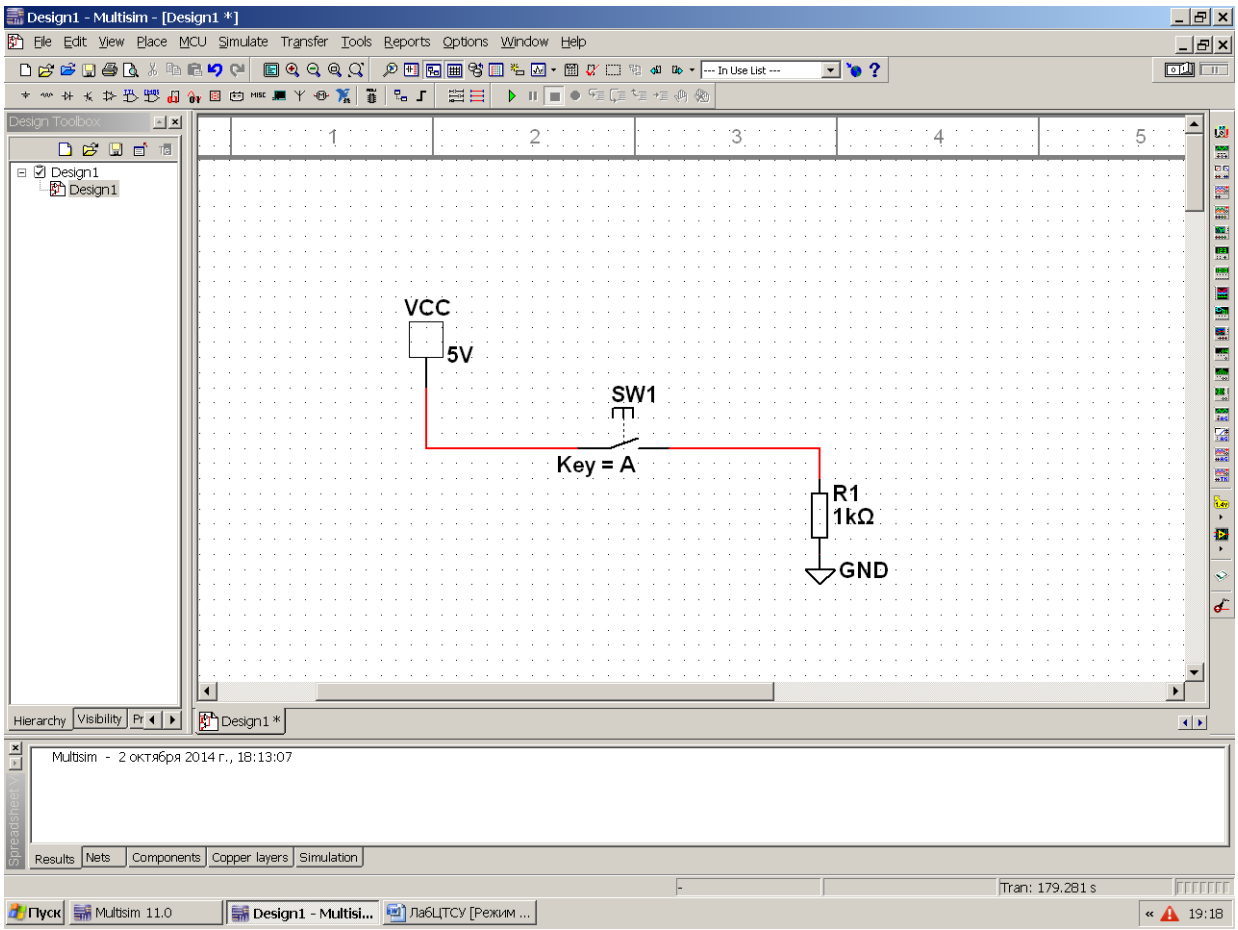


Рисунок 8 – Генератор тактовых импульсов с управляемой кнопкой SW1

Для формирования тактовых импульсов в автоматическом режиме следует использовать функциональный генератор (*Function Generator*). Функциональный генератор – виртуальный прибор, предназначенный для формирования периодических сигналов заданной формы: синусоидальной, треугольной или прямоугольной. Его можно выбрать на панели инструментов (на рисунке 1 –

правая вертикальная панель в окне программы). Щелкните на иконке *Function Generator*, а затем поместите его изображение на рабочем поле. Можно выбрать *функциональный генератор* и из пункта меню *Simulate ->Instruments -> Function Generator (Моделирование -> Приборы -> Функциональный генератор)*, после чего его компактное изображение появляется на рабочем поле. Компактное изображение функционального генератора и панель настройки его параметров показаны на рисунке 9.

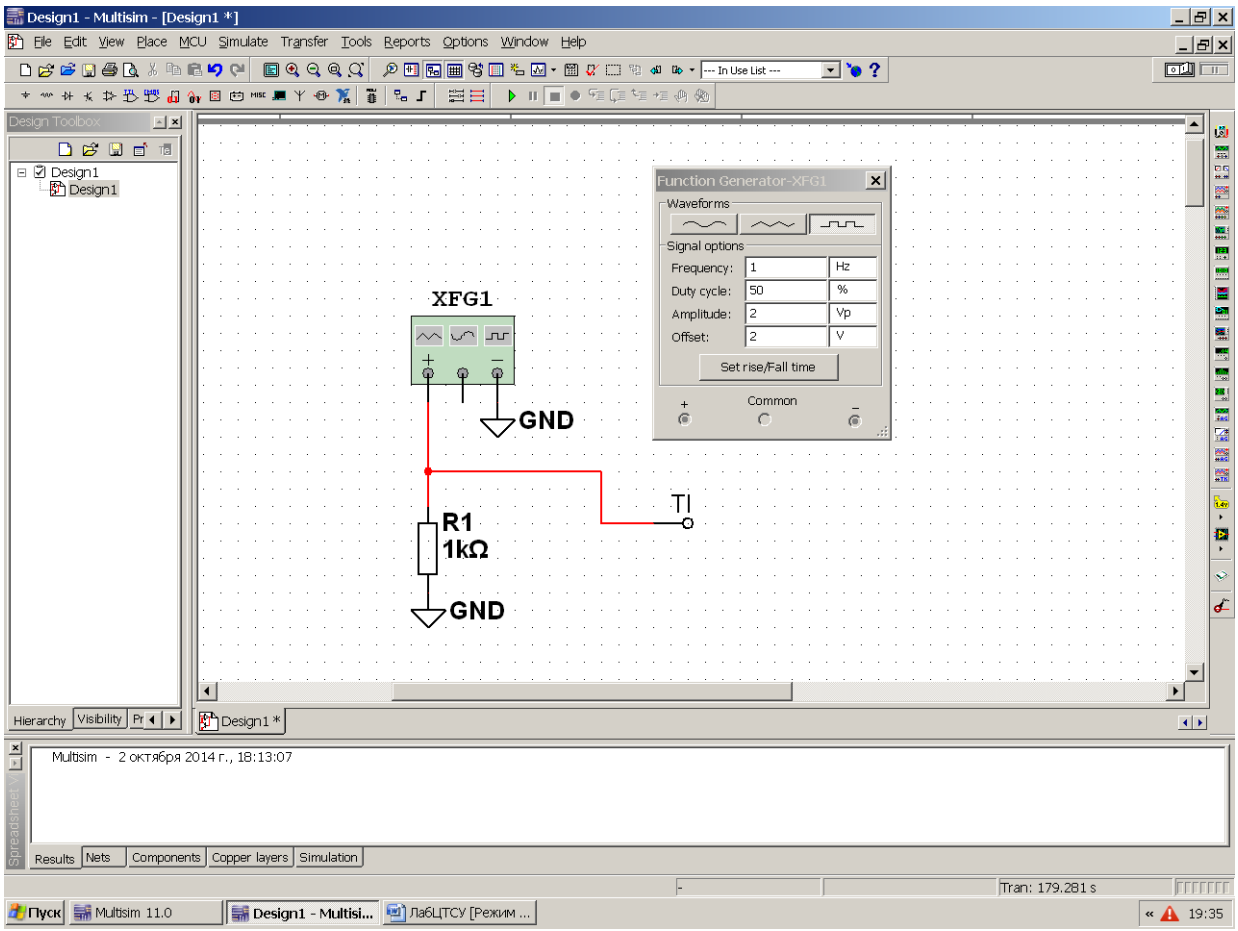


Рисунок 9 – Подключение функционального генератора и панель настройки его параметров

Для настройки параметров функционального генератора следует сделать двойной щелчок на его компактном изображении, после чего появляется окно развернутого изображения с панелью настройки (см. рисунок 9). Настройка функционального генератора сводится к выбору формы генерируемого сигнала (прямоугольная) и установки параметров сигнала. Устанавливаются параметры: частота следования (*Frequency*), скважность (*Duty cycle*), амплитуда сигнала

(*Amplitude*) и величина постоянного смещения уровня (*Offset*). На рисунке 9 показана настройка генератора на формирование прямоугольных импульсов частотой следования 1 Гц, скважностью 50%, изменяющихся от 0 до 4 В (амплитуда 2 В и смещение +2В).

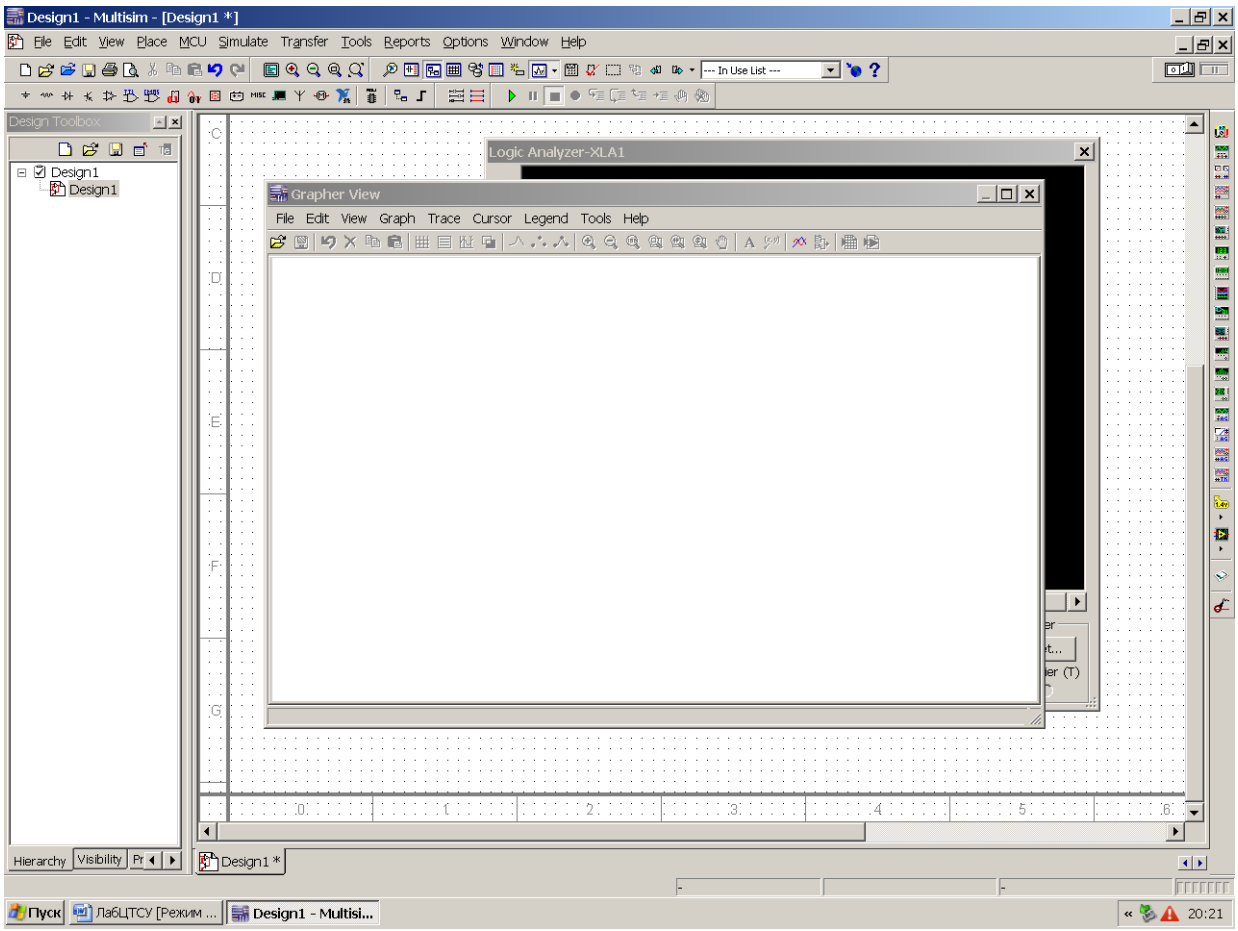


Рисунок 10 – Окно *Просмотр графиков*

10 Запуск и остановка моделирования схемы

Запуск и остановка моделирования схемы могут производиться тремя способами: через пункты меню *Simulate -> Run, Pause, Stop (Моделирование -> Пуск, Пауза, Стоп)*, аналогичными кнопками панели инструментов, выключателем моделирования и кнопкой *Пауза* на панели инструментов в правом верхнем углу экрана.

11 Визуализация результатов

Результаты работы схемы в любой момент времени могут быть просмотрены, если развернуть изображение логического анализатора. При этом в ходе процесса моделирования можно также изменить его настройки. Кроме того, для просмотра всех результатов моделирования можно использовать окно *Grapher View* (*Просмотр графиков)* (см. рисунок 10), которое вызывается через пункт меню *View ->Grapher* (*Вид -> Графопостроитель)*. Команда *Copy* в этом окне позволяет скопировать выбранные диаграммы в буфер обмена.

ЛАБОРАТОРНАЯ РАБОТА №1
РЕГИСТРОВАЯ РЕАЛИЗАЦИЯ ВЫЧИСЛЕНИЙ

ЦЕЛЬ РАБОТЫ: изучение особенностей функционирования вычислительных структур с регистровой организацией.

КРАТКИЕ СВЕДЕНИЯ ИЗ ТЕОРИИ

Регистровые вычислительные структуры представляют собой соединение регистров, сумматоров и блоков умножения. С помощью таких структур выполняется реализация цифровых устройств обработки информации с любыми передаточными функциями [1].

ОПИСАНИЕ МОДЕЛИРУЕМОЙ СХЕМЫ

В пакете NI Multisim 11.0 для выполнения работы используется четырехразрядный параллельный сумматор на микросхеме 74LS83 (или 74LS283) из набора TTL [5]. Российские аналоги этих микросхем - К555ИМ3 и К555ИМ6 соответственно. На таких микросхемах можно собрать сумматор любой разрядности, кратной четырем. Входы *A0-A3* обозначают разряды двоичного числа *A*, младший бит *A0*. Входы *B0-B3* - разряды двоичного числа *B*, младший бит *B0* . Выходы *S0-S3* обозначают разряды суммы:

$$S = A + B + P_n \quad ,$$

причем *S0* - младший бит суммы, а P_n - бит переноса из предыдущей тетрады сложения (если таковая существует, иначе на этот вход подается логический 0). При выполнении операции может появиться единица переноса в старшую тетраду P_{n+1} результата.

Для запоминания значения числа на один такт работы вычислителя используется параллельный регистр [2,4]. В работе рекомендуется использовать микросхему четырехбитового параллельного регистра 74LS296 (аналог

К555ИР16) или восьмибитового параллельного регистра 74LS374 (аналог К555ИР23) из набора TTL.

Для задания входных чисел рекомендуется использовать переключатели с фиксацией. Для выработки тактового сигнала рекомендуется использовать переключатель без фиксации из раздела *Electro_Mechanical->Momentary_Switches*.

Для индикации результата вычислений к каждому выходному проводнику рекомендуется подключить по одному бинарному пробнику.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Проведите исследование схемы параллельного сумматора. Для этого соберите схему в соответствии с рисунком Л1.1.

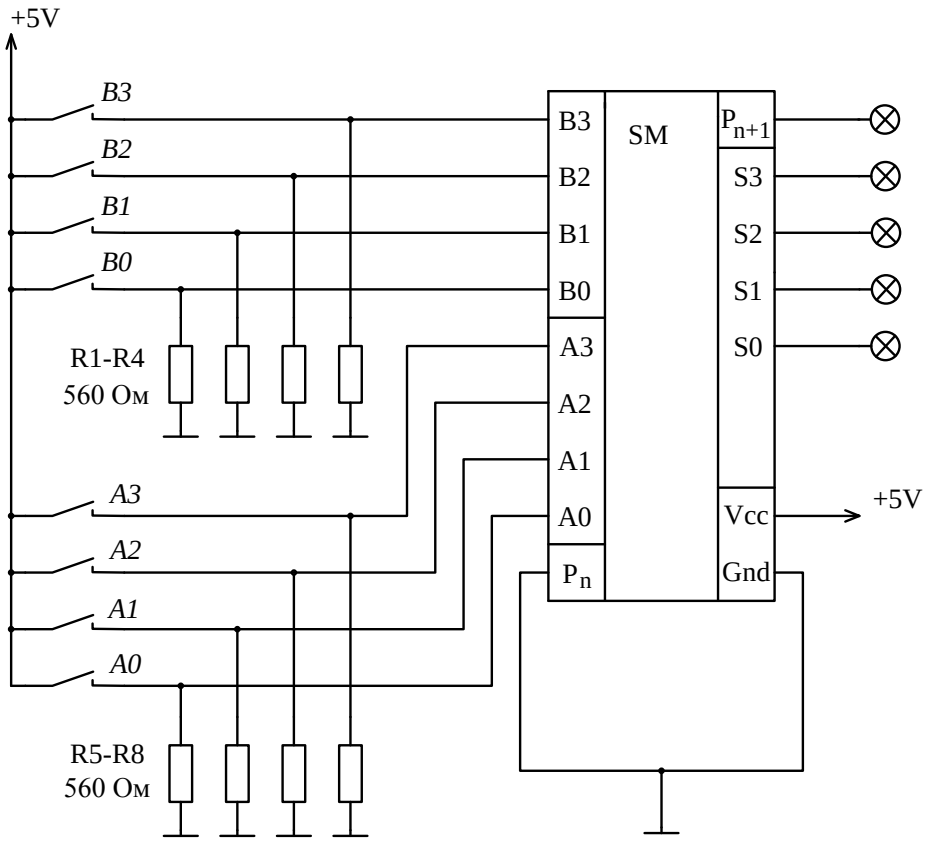


Рисунок Л1.1 – Схема с параллельным сумматором

Подавая десять различных кодовых комбинаций на входы A и B, запишите результаты в таблицу Л1. Вычислите десятичные эквиваленты двоичных чисел, рассматривая их коды сначала как числа без знака (положительные), а затем как числа со старшим знаковым разрядом. Результаты также запишите в таблицу Л1.1. По полученным данным сделайте выводы о работе двоичного сумматора.

Таблица Л1.1 – Результаты исследования сумматора

Коды чисел				Числа без знака			Числа со знаком		
A	B	Σ	P_{n+1}	A	B	Σ	A	B	Σ

2. Проведите исследование схемы регистрового процессора. Для этого используйте схему, показанную укрупненно на рисунке Л1.2.

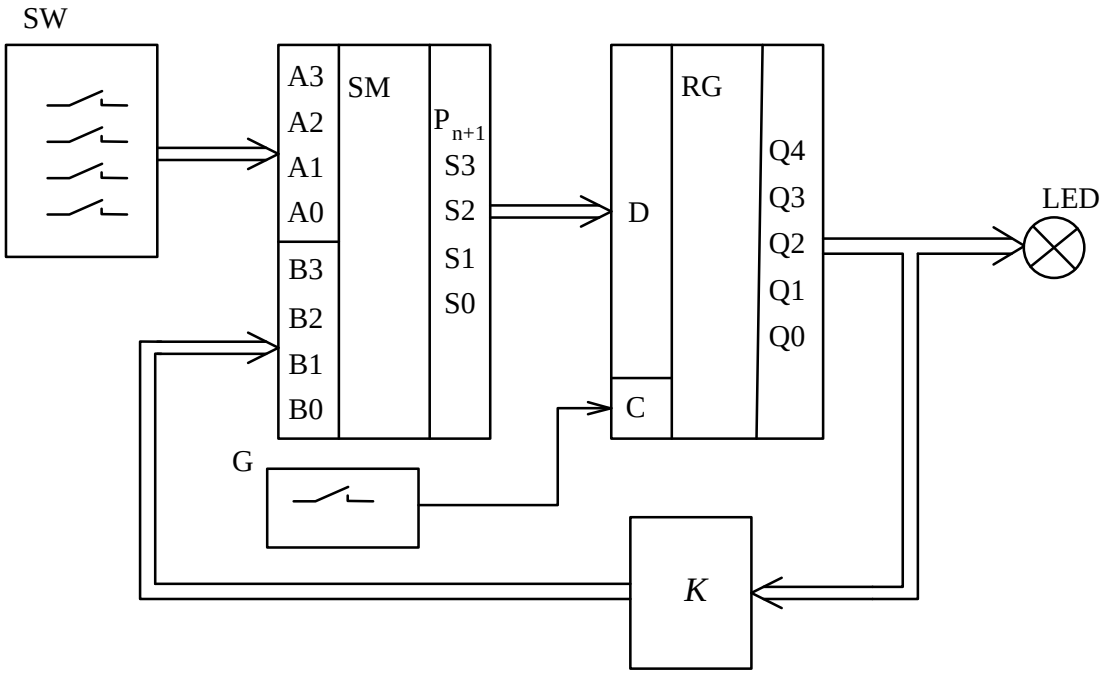


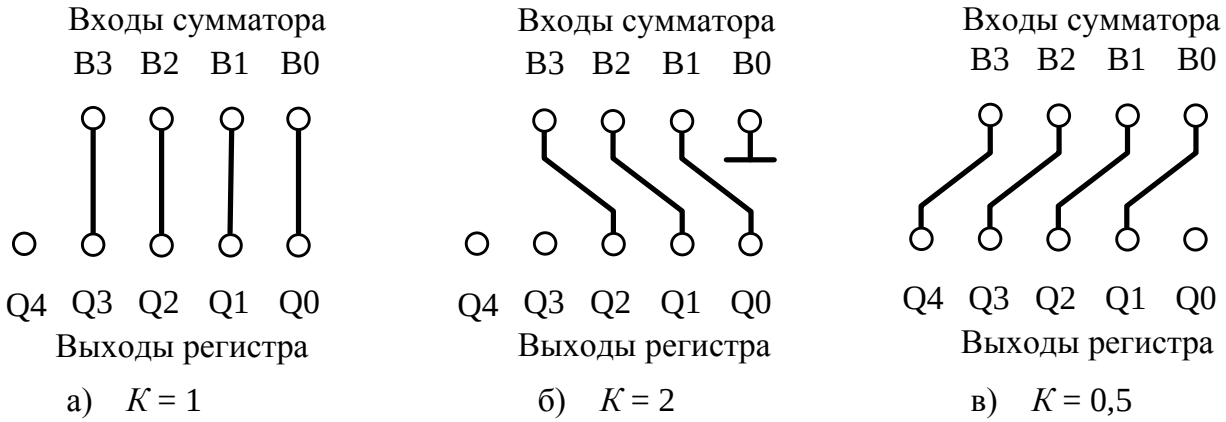
Рисунок Л1.2 – Укрупненная схема регистрового процессора: SW – переключатель для задания входного сигнала, G – генератор тактовых импульсов, LED – индикатор (пробники)

Исследование процессора выполните при трех значениях коэффициента обратной связи K = (0.5, 1, 2). Для каждого значения коэффициента обратной связи уточните разрядности шин передачи данных, гарантируя отсутствие переполнения разрядной сетки процессора. В зависимости от требуемой разрядности шин используйте в схеме либо четырехразрядный регистр 74LS295, либо восьмиразрядный регистр 74LS374. Коэффициент K = (0.5, 1, 2) реализуется сдвигом числа вправо или влево относительно разрядной сетки кода

путем соответствующей коммутации соединительных проводников. Варианты реализации коэффициента обратной связи *K* поясняются на рисунке Л1.3.

Задайте входной код, соответствующий числам *A*=(1, 3, 5). Подавая на вход синхронизации C регистра RG тактовые импульсы от одиночной кнопки без фиксации (генератор G), снимите реакцию схемы на заданный постоянный входной сигнал в зависимости от номера такта (т.е. переходной процесс).

3. По полученным в п.2 данным постройте графики реакций схемы на входные воздействия (графики переходных процессов). Вычислите максимальное и минимальное возможные числа в схеме для каждого значения коэффициента передачи *K*. Вычислите теоретически установившиеся значения выходного сигнала процессора и сравните со снятыми экспериментально. Объясните полученные результаты.



а) *K* = 1 б) *K* = 2 в) *K* = 0,5

Рисунок Л1.3 – Варианты реализации коэффициента передачи цепи обратной связи путем аппаратного сдвига числа

СОДЕРЖАНИЕ ОТЧЕТА

Кроме титульного листа, отчет должен содержать следующие разделы:

1. Задание.
2. Схемы исследуемых устройств.
3. Расчеты разрядности шин и установившихся значений выходного сигнала.
3. Таблицу результатов исследования сумматора.
4. Графики переходных процессов в схеме регистрового процессора.
5. Выводы по работе, комментирующие полученные результаты.

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Какую функцию выполняет регистр в схеме процессора?
2. Как записать математически выполняемую регистром функцию?
3. Каким разностным уравнением описывается работа процессора,

изображенного на рисунке 1.2?

4. Укажите условие сходимости вычислительного алгоритма для процессора, изображенного на рисунке 1.2.
5. Почему у цифрового процессора при одинаковом постоянном входном сигнале могут получаться различные установившиеся значения?

ЛАБОРАТОРНАЯ РАБОТА №2
СИНТЕЗ ДИСКРЕТНОГО НАБЛЮДАТЕЛЯ СОСТОЯНИЯ

ЦЕЛЬ РАБОТЫ: изучение особенностей организации импульсных устройств на примере динамического наблюдателя полного порядка.

КРАТКИЕ СВЕДЕНИЯ ИЗ ТЕОРИИ

При разработке регуляторов цифровых систем управления часто приходится проектировать дискретный наблюдатель состояния непрерывного объекта. В данной лабораторной работе изучаются особенности проектирования и реализации дискретного наблюдателя состояния с использованием встроенных процедур Matlab. Подробно теоретическая часть работы изложена в учебном пособии [3].

Рассмотрим синтез дискретного наблюдателя для непрерывного объекта на примере. Предположим, что непрерывный объект управления (ОУ) задан передаточной функцией

$$W(s) = K\frac{T_0 * s + 1}{T_0^2 * s^2 + 2\xi * T_0 * s + 1} \quad ,$$

где K=5 – коэффициент передачи, T_0=0.2 с – постоянная времени, ξ=0.5 – декремент затухания, s – независимая переменная преобразования Лапласа.

Предположим, что доступны измерению входной и выходной сигналы ОУ в дискретные моменты времени. Задан период квантования времени дискретного наблюдателя T_k = 0.02 с.

Разделим числитель и знаменатель передаточной функции ОУ на T_0^2 и запишем в виде

$$W(s) = 5\frac{5s + 25}{s^2 + 5s + 25} \quad . \qquad (1)$$

По полученной передаточной функции (1) построим структурную схему непрерывного ОУ, которая показана на рисунке Л2.1.

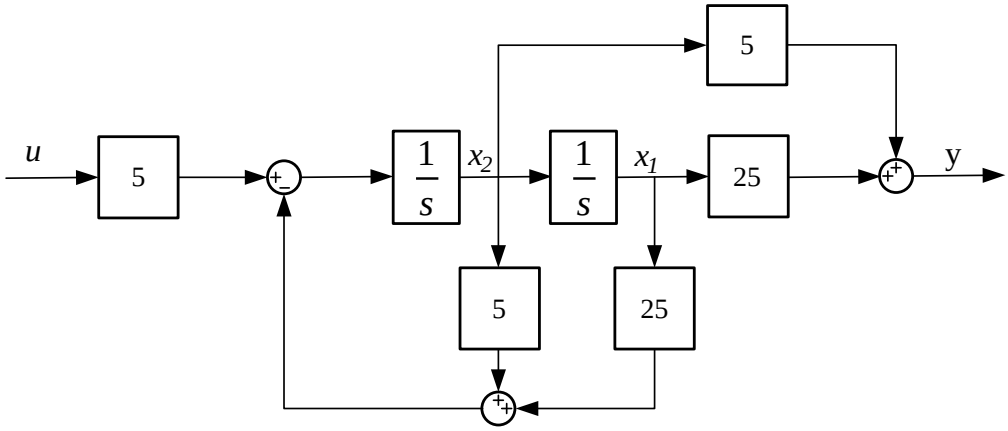


Рисунок Л2.1 – Структурная схема непрерывного объекта управления

Получим модель ОУ в дискретном времени с периодом квантования T_k. Для этого запишем уравнения движения непрерывного ОУ в форме вход-состояние-выход:

$$\begin{aligned} \dot{X}(t) &= A_н * X(t) + B_н * u(t), \\ y(t) &= C_н * X(t). \end{aligned} \qquad (2)$$

В соответствии со структурной схемой непрерывного ОУ, определим параметры матриц, входящих в уравнения (2):

$$A_н = \begin{bmatrix} 0 & 1 \\ -25 & -5 \end{bmatrix},\ B_н = \begin{bmatrix} 0 \\ 5 \end{bmatrix},\ C_н = \begin{bmatrix} 25 & 5 \end{bmatrix}\ .$$

Решение уравнения (2) с постоянными параметрами для произвольного интервала времени $[t,t+T]$ имеет вид [3]

$$\begin{aligned} X(t+T) &= e^{A_н T} * X(t) + \int_0^T e^{A_н (T-\tau)} * B_н * u(\tau) d\tau, \\ y(t) &= C_н * X(t). \end{aligned} \qquad (3)$$

Запишем уравнение (3) на интервале дискретности времени $[nT_k, (n+1)T_k]$, где $n=0,1,2\ \ldots$ – дискретное время. Считая, что значение $u(\tau)$ на интервале дискретности не изменяется, получим разностное уравнение

$$X((n+1)T_k) = e^{A_н T_k} * X(nT_k) + \int_0^{T_k} e^{A_н (T_k - \tau)} d\tau * B_н * u(nT_k),$$
$$y(nT_k) = C_н * X(nT_k). \qquad (4)$$

Уравнение (4) является искомой дискретной моделью ОУ, которое запишем в компактном виде:

$$X(n+1) = A * X(n) + B * u(n),$$
$$y(n) = C * X(n), \qquad (5)$$

где $A = e^{A_н T_k}$, $B = \int_0^{T_k} e^{A_н (T_k - \tau)} d\tau * B_н = A * A_н^{-1}(I - A^{-1}) * B_н$, I – единичная матрица, $C = C_н$.

Параметры матриц дискретной модели ОУ вычислим средствами Matlab. Для вычисления матричной экспоненты воспользуемся функцией *expm()*

$$A = \exp m(A_н * T_k),$$

$$A = \begin{bmatrix} 0.9952 & 0.019 \\ -0.475 & 0.9002 \end{bmatrix}.$$

Матрицу B вычислим непосредственно по формуле

$$B = A * A_н^{-1}(I - A^{-1}) * B_н,$$

$$B = \begin{bmatrix} 0.001 \\ 0.095 \end{bmatrix}.$$

Параметры матрицы C дискретной модели совпадают с параметрами $C_н$ непрерывной модели, т.е.

$$C = [25 \quad 5].$$

На рисунке Л2.2 показана схема моделирования непрерывного ОУ и дискретного наблюдающего устройства, использующего в качестве модели ОУ разностное уравнение (5). Наблюдатель дополнен обратной связью по ошибке оценивания выходной переменной

$$\varepsilon(n) = y(n) - \hat{y}(n),$$

где $y(n), \hat{y}(n)$ – дискретные выходные переменные ОУ и наблюдателя, соответственно. Переменные состояния наблюдателя $\hat{x}_1(n), \hat{x}_2(n)$ являются

оценками переменных состояния $x_1(n), x_2(n)$ дискретной модели ОУ и должны совпадать с переменными состояния непрерывной модели в дискретные моменты времени $t = nT_k$. Обратная связь обеспечивает заданную степень устойчивости наблюдателя, которая должна быть выше степени устойчивости исходной дискретной модели ОУ. Разностное уравнение, описывающее работу наблюдающего устройства, имеет вид

$$\begin{aligned} \hat{X}(n+1) &= A * \hat{X}(n) + K * \varepsilon(n) + B * u(n), \\ \hat{y}(n) &= C * \hat{X}(n), \end{aligned} \qquad (6)$$

где $\hat{X}(n) = \begin{bmatrix} \hat{x}_1(n) \\ \hat{x}_2(n) \end{bmatrix}$ – вектор оценок состояния ОУ, $K = \begin{bmatrix} k_1 \\ k_2 \end{bmatrix}$ – матрица коэффициентов обратной связи по ошибке оценивания выходной переменной.

Качество работы наблюдателя определяется вектором ошибок оценивания, равного $E(n) = X(n) - \hat{X}(n)$. Из уравнений (5) и (6) следует, что изменение вектора ошибок оценивания во времени подчиняется разностному уравнению

$$E(n+1) = (A - K * C)E(n), \quad E(0) = E_0. \qquad (7)$$

Из (7) следует, что если собственные числа матрицы $\phi = A - K * C$ лежат внутри круга единичного радиуса, то при произвольных начальных условиях вектор $E(n) \to 0$ при $n \to \infty$.

Рассчитаем параметры матрицы K методом модального управления. Для этого назначим желаемые собственные числа матрицы ϕ исходя их желаемой скорости сходимости вектора ошибок к нулю. Например, выберем желаемые собственные числа λ матрицы ϕ из условия $|\lambda_m|^5 \le \lambda \le |\lambda_m|^3$, где λ_m – минимальное собственное число матрицы A.

Собственные числа матрицы A вычислим с помощью функции $\lambda_A = eig(A)$, получим

$$\lambda_A = \begin{bmatrix} 0.9477 + 0.08227\,j \\ 0.9477 - 0.08227\,j \end{bmatrix}.$$

Модули собственных чисел матрицы A одинаковые. Значения $|\lambda_m|^3 = 0.86$, $|\lambda_m|^5 = 0.779$.

Назначим желаемые собственные числа матрицы ϕ наблюдателя с обратной связью равными $\lambda = [0.8 \quad 0.8]^T$. Матрицу обратной связи вычислим по формуле Аккермана [9], используя функцию acher():

$$G = \text{acker}(A^T, C^T, \lambda) \ , \quad K = G^T \ ,$$

$$K = \begin{bmatrix} -0.0004 \\ 0.0612 \end{bmatrix}.$$

Проверить правильность результата можно, вычислив собственные числа матрицы $\phi = A - K * C.$

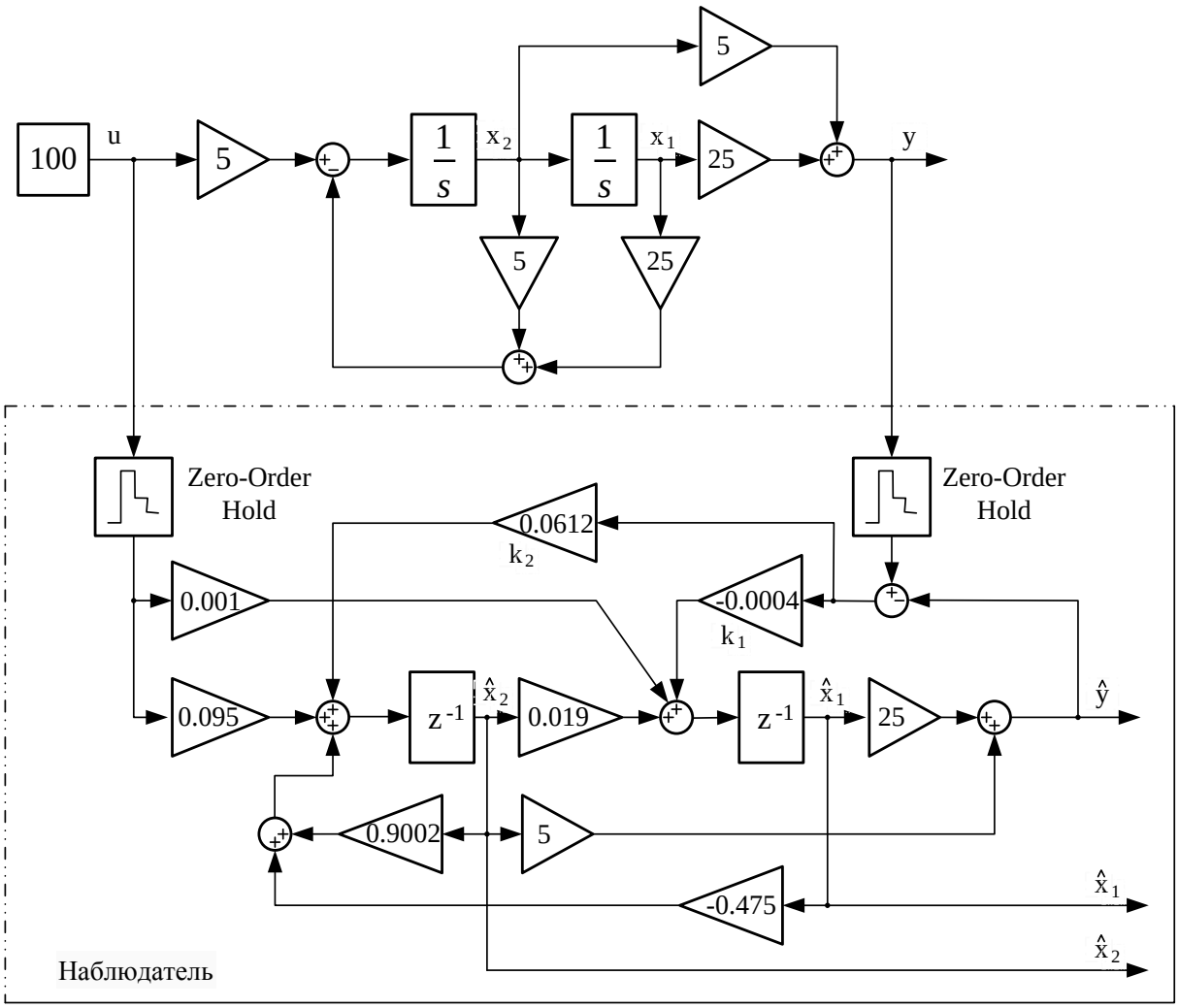


Рисунок Л2.2 – Схема моделирования непрерывного ОУ
и дискретного наблюдателя состояния

ВАРИАНТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

Непрерывный объект управления задан передаточной функцией

$$W(s) = K_0 \frac{T_1 * s + 1_1}{T_0^2 * s^2 + 2\xi T_0 * s + 1} \ .$$

Параметры передаточной функции сведены в таблицу Л2. В таблице Л2.1 также

заданы значение периода квантования времени T_k дискретного наблюдателя и диапазон изменения входного воздействия *u(t).*

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. В соответствии с заданным вариантом разработать схему моделирования непрерывного объекта управления.
2. Используя матричное описание непрерывного ОУ, вычислить параметры матриц уравнений для ОУ в дискретном времени.

Таблица Л2.1 – Варианты выполнения лабораторной работы

№ варианта	K_0	T_1, *с.*	T_0, *с.*	ξ	T_k, *с.*	Диапазон *u(t)*
1	10	0.2	0.5	0.5	0.02	±100
2	3	0.2	0.2	0.7	0.02	±100
3	20	0.2	0.1	0.7	0.01	±100
4	10	0	0.1	0.5	0.01	±250
5	5	0.1	0.2	0.5	0.02	±250
6	15	0.5	0.5	0.7	0.02	±250
7	10	0.1	0.1	0.5	0.01	±200
8	5	0.1	0.2	0.6	0.02	±200
9	15	0.4	0.4	0.4	0.02	±200
10	10	0.2	0.4	0.6	0.02	±200
11	5	0.4	0.2	0.4	0.02	±500
12	7	0.2	0.4	0.5	0.02	±500
13	10	0.05	0.05	0.7	0.005	±500
14	3	0.1	0.05	0.5	0.005	±500
15	5	0.05	0.05	0.4	0.005	±150

3. Сравнить реакции непрерывной и дискретной моделей ОУ на ступенчатое воздействие максимальной амплитуды (при нулевых начальных условиях). В настройках Simulink *Simulation/ Model Configuration Parameters* установить *Type: Fixed-step , Solver: ode4 (Runge-Kutta) , Fixed-step size* – численное значение параметра T_k . Моделированием переходных процессов убедиться, что реакции моделей на ступенчатое воздействие одинаковые как по выходным сигналам, так и по компонентам векторов состояния.
4. Рассчитать коэффициенты матрицы обратной связи дискретного наблюдателя.

5. Моделированием показать, что выходная переменная и компоненты вектора состояния наблюдателя сходятся к соответствующим переменным непрерывной модели ОУ. Моделирование выполнять при ненулевом начальном значении компоненты $x_1(0)$ (например 20 или 50).
6. Методом моделирования определить максимальные значения переменных $\hat{y}(nT)$, $\hat{x}_1(nT)$, $\hat{x}_2(nT)$ дискретного наблюдателя при максимальном значении входного ступенчатого сигнала. Аналогично определить максимальные значения указанных переменных при гармоническом входном сигнале с максимальной амплитудой и частотами $\omega_1 = 1/3T_0$, $\omega_2 = 1/T_0$ и $\omega_3 = 3/T_0$.

   Полученные результаты свести в таблицу. Полученные значения используются при выполнении лабораторной работы №3.

СОДЕРЖАНИЕ ОТЧЕТА

Кроме титульного листа, отчет должен содержать следующие разделы:

1. Задание.
2. Схему моделирования непрерывного объекта.
3. Расчеты и математическую модель дискретного наблюдателя.
4. Графики сравнения реакций непрерывного и дискретного ОУ на ступенчатое воздействие.
5. Расчет параметров матрицы обратной связи дискретного наблюдателя. Графики, иллюстрирующие сходимость наблюдателя при ненулевых начальных условиях.
6. Таблицу максимальных значений переменных дискретного наблюдателя при ступенчатом и гармонических входных воздействиях.
7. Выводы по работе.

КОНТРОЛЬНЫЕ ВОПРОСЫ

6. В каком смысле понимается сходимость значений дискретной последовательности к значениям непрерывного сигнала ?
7. Как рассчитать параметры матриц дискретного аналога непрерывного объекта?
8. Сформулируйте условие сходимости вектора оценок $\hat{X}$ дискретного наблюдателя к вектору X объекта управления.
9. Какова роль обратной связи в наблюдателе?
10. Почему при расчете матрица обратной связи наблюдателя в формуле Аккермана используются транспонированные матрицы модели?

ЛАБОРАТОРНАЯ РАБОТА №3
ВЫБОР РАЗРЯДНОСТИ ПЕРЕМЕННЫХ ЦИФРОВОГО НАБЛЮДАТЕЛЯ СОСТОЯНИЯ

ЦЕЛЬ РАБОТЫ: освоение способа расчета разрядности представления данных в алгоритме цифровой фильтрации и моделирование работы фильтра с учетом эффекта квантования уровней сигналов.

КРАТКИЕ СВЕДЕНИЯ ИЗ ТЕОРИИ

При реализации цифровых устройств возникает проблема, связанная с расчетом разрядности представления данных. Разрядность данных должна быть выбрана так, чтобы не возникало переполнение разрядной сетки при выполнении арифметических операций. Кроме того, необходимо обеспечить вычисление переменных с погрешностью, не превышающей заданную.

Известно несколько способов расчета разрядности цифрового устройства [1]. В настоящей лабораторной работе для заданного цифрового фильтра необходимо рассчитать разрядности представления данных исходя из максимальных значений переменных решаемой задачи. Кроме того, следует выполнить моделирование работы цифрового устройства с учетом рассчитанной разрядности данных. Лабораторная работа выполняется в программе Matlab Simulink с использованием специфических блоков для выполнения вычислений в целочисленном формате представления данных.

Вычислительные блоки, представленные в инструментах Simulink, позволяют легко реализовать необходимые вычислительные операции – суммирование, умножение на постоянный коэффициент и задержку сигнала на 1 такт. Однако эти блоки оперируют с данными, представленными либо в формате с плавающей запятой, либо целочисленном формате общепринятой разрядности (8, 16 и 32 бита). В данной лабораторной работе разрядность данных рассчитывается и может отличаться от общепринятой. Поэтому для выполнения работы требуется создать особый блок, разрядность представления данных которого должна задаваться в виде параметра.

Рассмотрим типовой фрагмент вычислительной структуры цифрового фильтра, показанный на рисунке Л3.1. Он содержит операции умножения переменных *x1(n)* и *x2(n)* на коэффициенты *K1* и *K2*, суммирование результатов, а также элемент задержки сигнала на 1 такт (элемент с передаточной функцией z^{-1}). Как следует из рассматриваемой схемы, вычисление выходной переменной осуществляется в соответствии с уравнением

$$y(n+1) = K_1 * x_1(n) + K_2 * x_2(n), \qquad (8)$$

где n – дискретное время, $n= 0,1,2, \ldots$.

При учете разрядности представления переменных $x_1(n)$, $x_2(n)$ и $y(n)$ учтем следующие особенности реализации целочисленных операций:

- в качестве элемента задержки сигнала на 1 такт используется элемент памяти (регистр), который не изменяет разрядности хранимых данных;
- разрядности переменных $x_1(n)$ и $x_2(n)$ определяются другими фрагментами схемы цифрового фильтра и в рассматриваемом фрагменте являются величинами заданными;
- размерности результатов вычисления произведений $K_1*x_1(n)$ и $K_2*x_2(n)$ ограничиваются разрядностью используемого сумматора и могут быть приняты одинаковыми для обоих слагаемых.

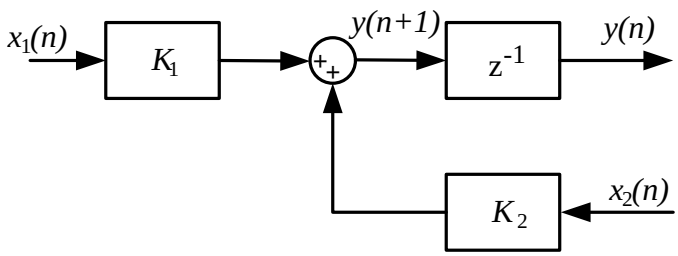


Рисунок Л3.1 – Схема типового фрагмента цифрового фильтра

Если обозначить влияние разрядности на значение переменной некоторой нелинейной функцией $f_m(.)$, где m – разрядность выходной переменной, то с учетом перечисленных выше особенностей реализации вычислений уравнение (8) можем записать в виде

$$y(n+1) = f_m(K_1 * x_1(n)) + f_m(K_2 * x_2(n)). \qquad (9)$$

На рисунке Л3.2 показана схема фрагмента цифрового фильтра, реализующего вычисления в соответствии с уравнением (9). Как следует из рисунка Л3.2, при создании модели фрагмента цифрового фильтра в Simulink потребуется оригинальный элемент – сумматор, оперирующий переменными с заданной разрядностью m. На рисунке Л3.2 структура такого сумматора выделена в виде блока СЗР (сумматор с заданной разрядностью).

Нетрудно показать, что математически функция $f_m(.)$ определяется побитовой логической операцией И между битами операнда и маски, равной 2^m-1. Такую

функцию следует создать в Simulink с помощью блока S-Function [8]. Этот инструментальный блок хранится в Simulink Library в разделе User-Defined Functions и требует задания алгоритма работы в виде кода S-функции.

Создание S-функции удобно выполнить с помощью специального инструмента – S-Function Builder. Этот инструментальный блок также расположен в разделе User-Defined Functions. Порядок создания сумматора с заданной разрядностью переменных следующий.

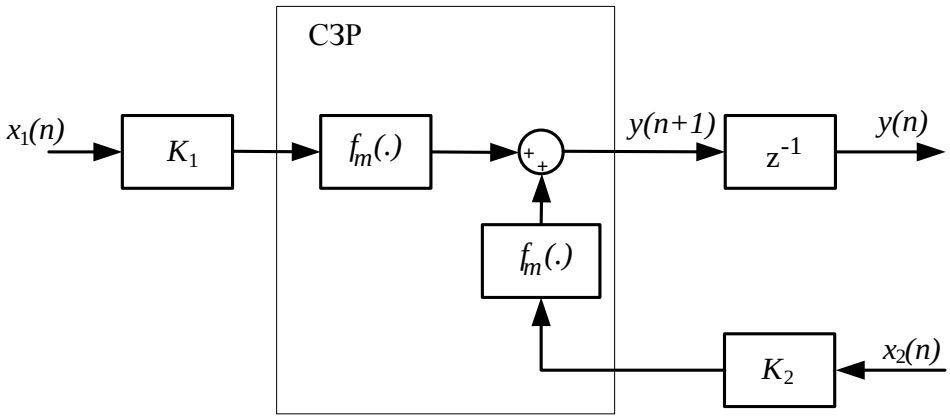


Рисунок Л3.2 – Схема фрагмента цифрового фильтра с учетом разрядности переменных

Шаг 1. Создайте окно для построения новой модели. Мышкой перетащите инструмент S-Function Builder из Simulink Library в окно новой модели. Двойным щелчком левой кнопки мыши запустите инструмент S-Function Builder. В верхней части открывшегося окна в поле S-function name следует задать оригинальное имя функции, например Sum_And. Во вкладке Initialization следует оставить нулевые значения количеств внутренних состояний блока, как показано на рисунке Л3.3.

Шаг 2. Щелчком клавиши мыши на вкладке Date Properties откройте окно задания переменных блока. На вспомогательной вкладке Input Ports следует задать два входных порта с именами *u0* и *u1*. Для добавления очередного порта следует использовать встроенную кнопку  (Add). Следует также установить свойства входных портов, как показано на рисунке Л3.4.

На вспомогательной папке Output Ports следует задать один выходной порт с именем *y0*. На вспомогательной папке Parameters задайте один параметр блока с именем *m*. Выберете формат этой переменной, например int16 (целочисленный 16 бит). Окно задания параметра показано на рисунке Л3.5.

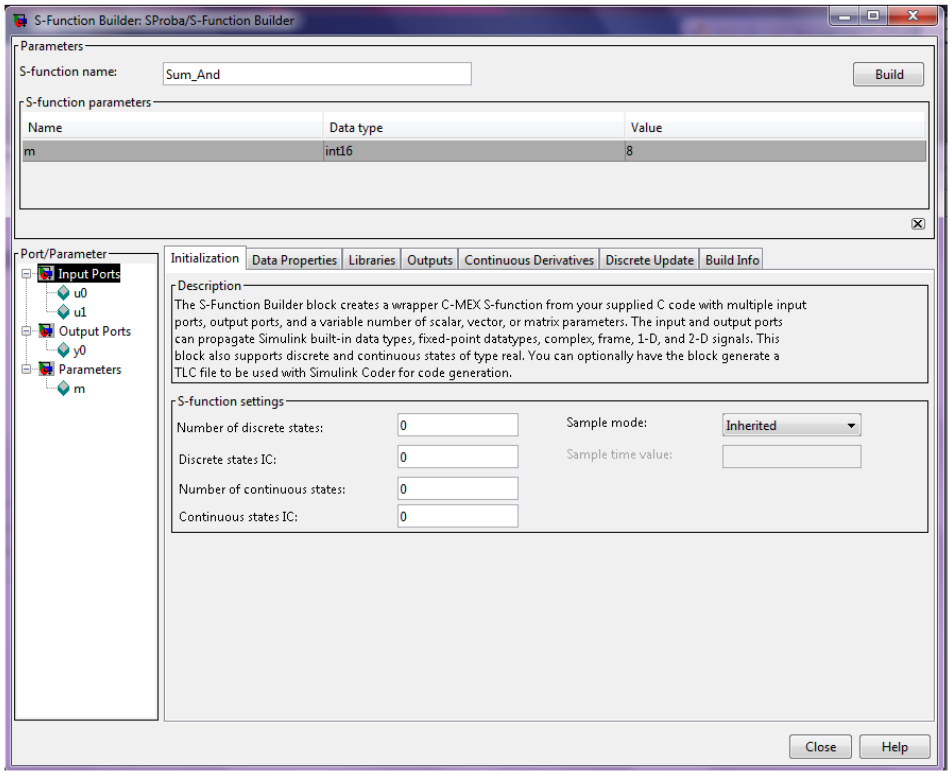


Рисунок Л3.3 – Окно Initialization инструмента S-Function Builder

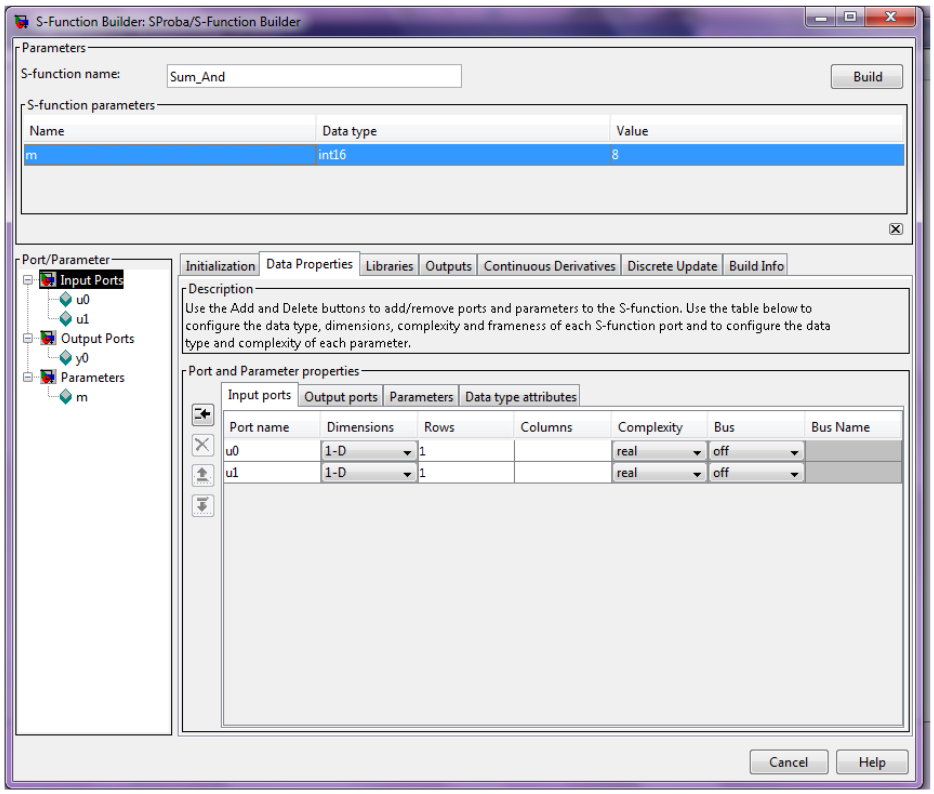


Рисунок Л3.4 – Окно задания входных портов блока

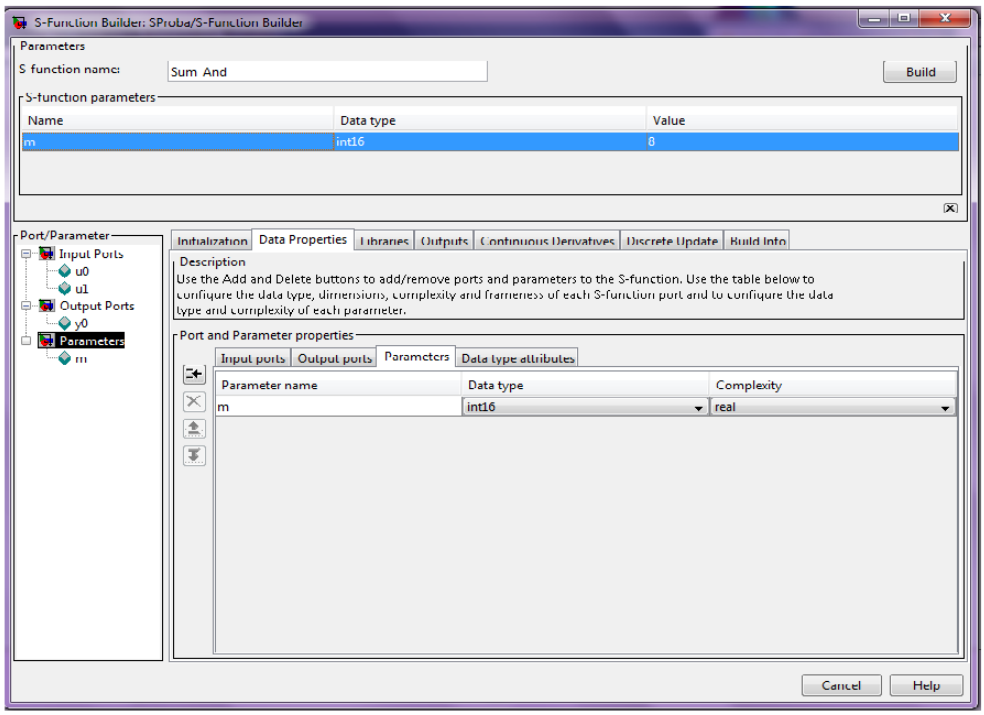


Рисунок Л3.5 – Окно определения параметров блока

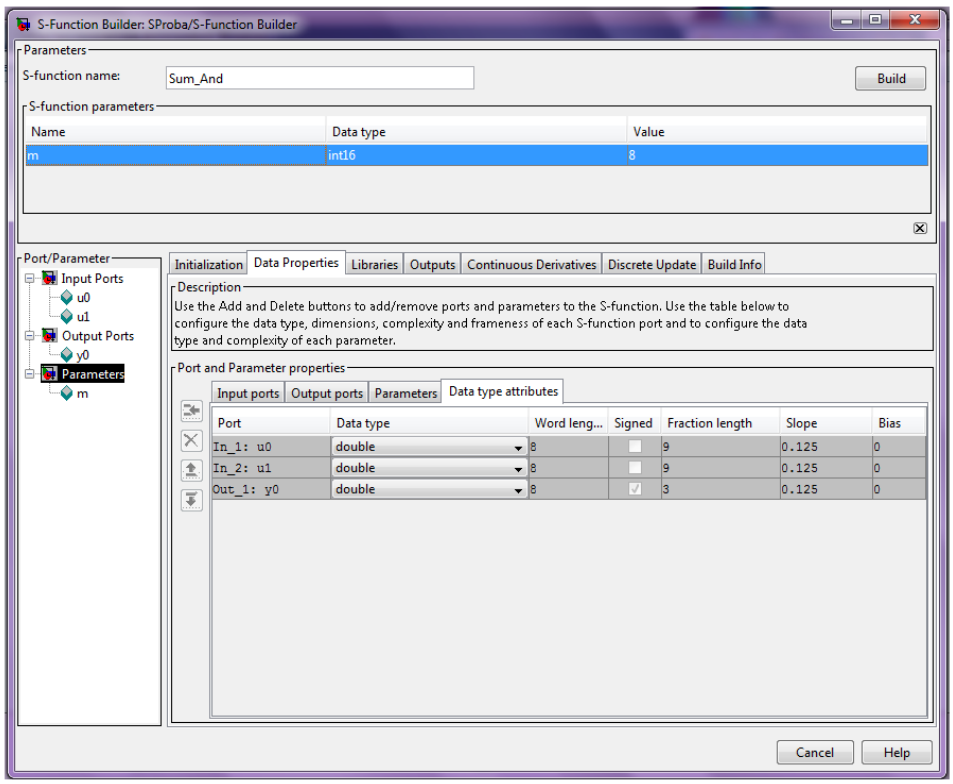


Рисунок Л3.6 – Окно задания свойств переменных

На вспомогательной вкладке Date type attributes задайте форматы представления входных и выходной переменных блока, как показано на рисунке Л3.6.

Шаг 3. Откройте вкладку Outputs. В открывшемся окне на языке С следует написать алгоритм вычисления выходной переменной блока. Для вычисления суммы чисел с заданной разрядностью с учетом сделанных выше настроек блока алгоритм задается следующим кодом:

```
long  x1,x2,M,n1; /* Задаем имена внутренних переменных */
int n;
n=m[0]; /* Считываем заданное значение разрядности переменных */
/* Вычисляем маску M=2^m-1 */
M=1;
while (n>0)
{M=M*2;
 n--;}
M=M-1;
/* Вычисляем сумму с заданной разрядностью */
n1=u0[0];
if(n1>=0)  x1=n1&M;
  else x1=-((-n1)&M);
n1=u1[0];
if(n1>=0)  x2=n1&M;
  else x2=-((-n1)&M);
y0[0]=x1+x2;
```

Вид окна с введенным кодом показан на рисунке Л3.7. Следует помнить, что при сложении двух целых чисел разрядностью *m* результат будет иметь разрядность *m*+1. Дополнительный разряд содержит бит переноса/заема сумматора.

Шаг 4. В правом верхнем углу главного окна S-Function Builder нажмите кнопку Build . Будут созданы файлы SumAnd.c, SumAnd wrapper.c, SumAnd.tlc, необходимые для работы созданной S-функции в среде Simulink. В случае обнаружения каких-либо опечаток или ошибок задания параметров Builder выдаст соответствующие замечания. Обнаруженные ошибки следует исправить и повторить генерацию файлов. При корректном завершении генерации файлов откроется окно, показанное на рисунке Л3.8.

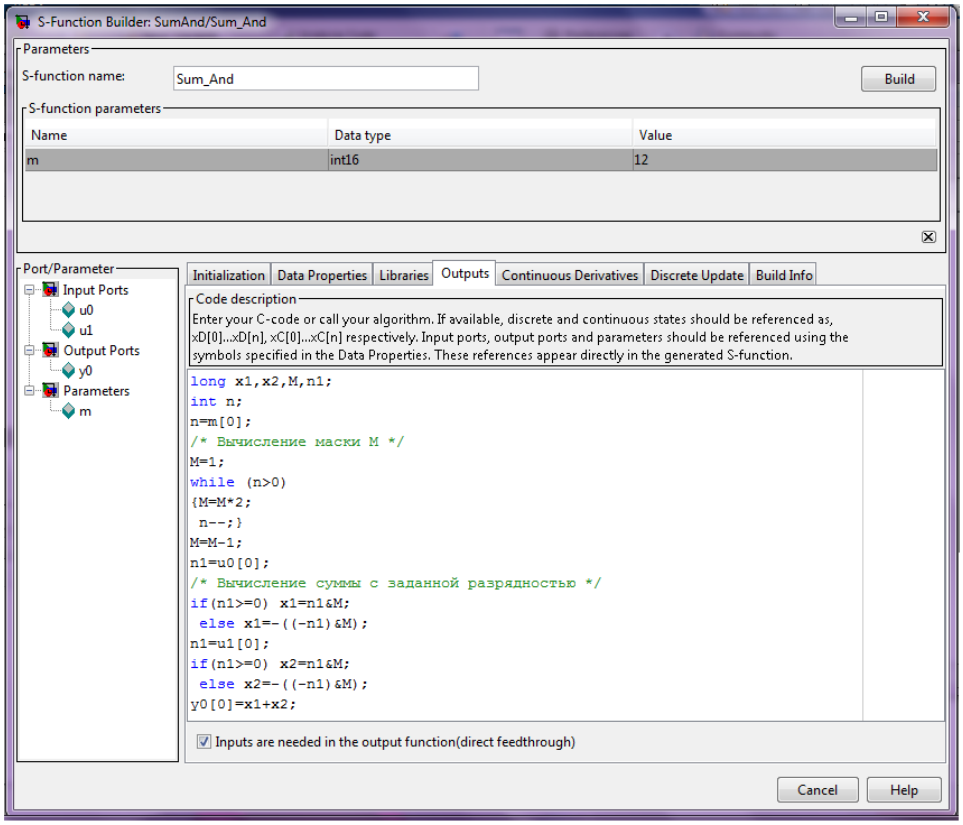


Рисунок Л3.7 – Окно ввода кода S-функции

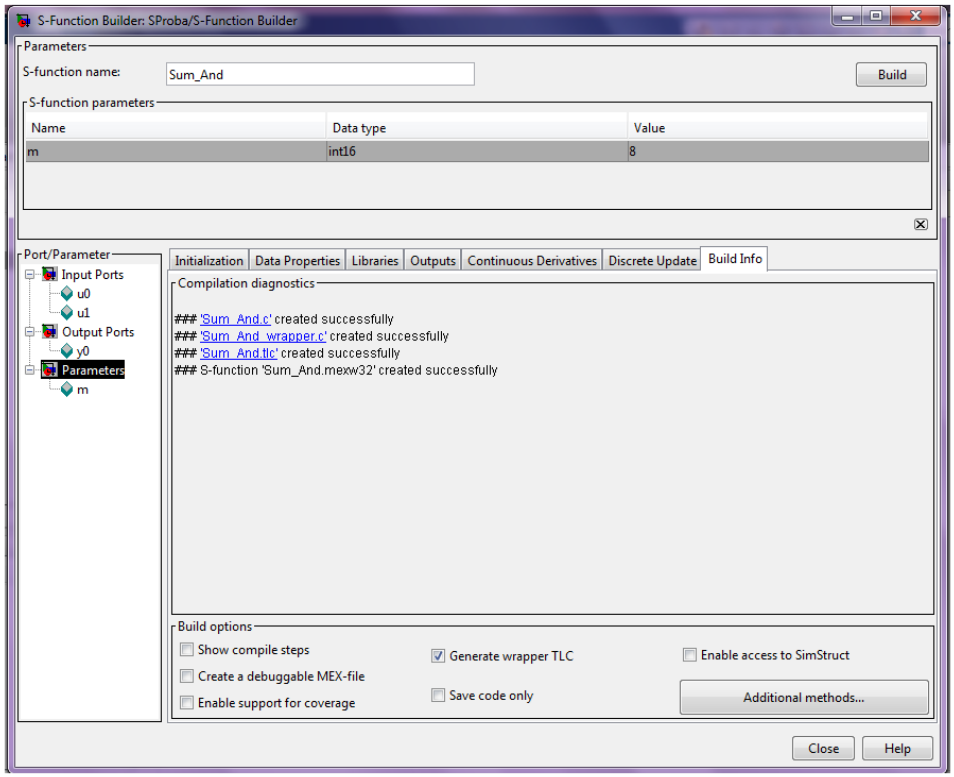


Рисунок Л3.8 – Окно итогового отчета блока S-Function Builder с информацией о корректном завершении работы

Закройте Главное окно S-Function Builder и сохраните модель СЗР (маску блока) в файл Sum_And.slx для последующего использования.

На рисунке Л3.9 показан пример подключения созданного блока Sum_And к фрагменту дискретного устройства при моделировании в среде Simulink.

Целочисленное представление переменных в алгоритме работы наблюдателя приводит к ошибкам оценивания, порожденным квантованием уровней переменных. Теоретически после завершения переходного процесса в установившемся режиме ошибка оценивания $\varepsilon_{уст}(n) = x_{уст}(n) - \hat{x}_{уст}(n)$ должна стремиться к нулю. Наличие квантования уровня переменной $\hat{x}(n)$ приводит к тому, что $\varepsilon_{уст}(n) \neq 0$. Поэтому о качестве назначения разрядности представления переменных можно судить по относительным ошибкам оценивания

$$\varepsilon_{отн} = \max\left(\frac{x_{уст}(n) - \hat{x}_{уст}(n)}{x_{уст}(n)}\right) * 100\% \ .$$

Относительные ошибки оценивания можно уменьшить, введя масштабирование переменных. Для этого в алгоритме наблюдателя используют рабочие переменные, связанные с оцениваемой переменной уравнением

$$\hat{x}_p(n) = M * \hat{x}(n),$$

где $\hat{x}_p(n)$ - рабочая переменная алгоритма, M –масштаб переменной. Непосредственно из уравнения следует, что масштаб M может быть выбран равным отношению диапазона изменения рабочей переменной к диапазону изменения оцениваемой переменной.

После назначения масштабов всех рабочих переменных наблюдателя следует пересчитать коэффициенты всех блоков умножения в алгоритме работы наблюдателя по формуле:

$$K_p = K * \frac{M_{вых}}{M_{вх}} \quad ,$$

где K – коэффициент в исходной модели наблюдателя, K_p – коэффициент рабочей модели, $M_{вых}$ – масштаб выходной переменной блока умножения, $M_{вх}$ – масштаб входной переменной. Более подробно методика масштабирования решаемой задачи изложена в [1].

ВАРИАНТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

Варианты выполнения лабораторной работы приведены в таблице Л2.1. Рекомендуется выполнять лабораторные работы №2 и №3 по единому варианту и результаты выполнения лабораторной работы №2 использовать в качестве исходных данных при выполнении работы №3.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Создайте блок сумматора с заданной разрядностью, как описано выше.
2. Воспользуйтесь дискретной моделью наблюдателя, синтезированной и исследованной в лабораторной работе №2. Замените сумматоры в исходной дискретной модели на сумматоры с заданной разрядностью.
3. Используя полученные в работе №2 максимальные значения переменных наблюдателя, рассчитайте разрядности их цифрового представления. Установите полученные значения разрядностей переменных в полях параметров соответствующих блоков сумматоров.
4. Выполните моделирование работы цифрового наблюдателя при максимальном значении входного ступенчатого сигнала *u(t)*. Определите относительные ошибки оценок переменных состояния и выходной переменной. Полученные данные сведите в таблицу.
5. Последовательно уменьшая значение входного ступенчатого сигнала *u(t)*, определите диапазон значений входного сигнала, при которых максимальная относительная ошибка оценивания не превышает 10% . Для поиска диапазона рекомендуется использовать алгоритм деления пополам.
6. Назначьте масштабы рабочих переменных цифрового наблюдателя, задав желаемые диапазоны их изменения в пределах 1000 – 2000 единиц. Пересчитайте коэффициенты передачи блоков умножения с учетом выбранных масштабов переменных. Масштабы входных переменных *u(t)* и *y(t)* примите равными единице. Установите новые значения разрядностей всех сумматоров цифрового наблюдателя.
7. Повторите п.п. 4 и 5 порядка выполнения работы. Зафиксируйте типовые переходные процессы. При расчете относительных ошибок учтите масштабы рабочих переменных цифрового наблюдателя.

СОДЕРЖАНИЕ ОТЧЕТА

Кроме титульного листа, отчет должен содержать следующие разделы:

1. Задание

2. Схему моделирования непрерывного объекта и цифрового наблюдателя
3. Таблицу результатов моделирования работы цифрового наблюдателя
4. Расчет масштабов переменных и коэффициентов передачи блоков умножения цифрового наблюдателя
5. Графики, иллюстрирующие сходимость к нулю ошибок оценок переменных объекта управления при ненулевых начальных условиях
6. Таблицу результатов моделирования работы цифрового наблюдателя с выбранными масштабами переменных
7. Выводы по работе

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Как математически можно задать разрядность операндов двоичного сумматора?
2. Как вычислить разрядность представления переменной, зная диапазон ее изменения и требуемую погрешность оценивания?
3. Как вычислить абсолютную и относительную ошибки оценивания переменной?
4. Как рассчитать масштаб цифрового представления переменной?
5. Как выполнить масштабирование задачи при заданных значениях масштабов переменных?

ЛАБОРАТОРНАЯ РАБОТА №4
МИКРОПРОГРАММНЫЙ АВТОМАТ С ДВУМЯ АДРЕСНЫМИ ПОЛЯМИ

ЦЕЛЬ РАБОТЫ: изучить особенности программирования микропрограммного автомата с принудительной адресацией и двумя адресными полями.

КРАТКИЕ СВЕДЕНИЯ ИЗ ТЕОРИИ

Микропрограммные автоматы (МПА) предназначены для формирования двоичных кодовых последовательностей. Основные применения МПА: программное управление объектами автоматики, генерирование задающих воздействий для автоматических систем, формирование последовательности

кодов управления ходом вычислений в цифровых устройствах обработки информации [1].

Одна из возможных схем автомата показана на рисунке Л4.1. Микропрограммный автомат состоит из постоянного запоминающего устройства (ПЗУ) и регистра (см. [1,6]), охваченных обратной связью. Из-за наличия в схеме регистра обновление выходной величины происходит в дискретные моменты времени, определяемые поступлением входного тактового импульса. Поведение автомата определяется записанными в ПЗУ данными. В силу 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.

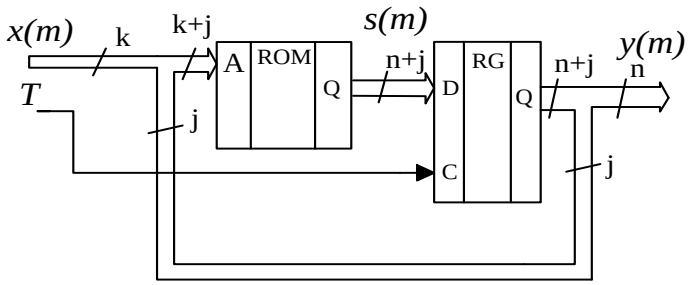


Рисунок Л4.1 – Схема микропрограммного автомата

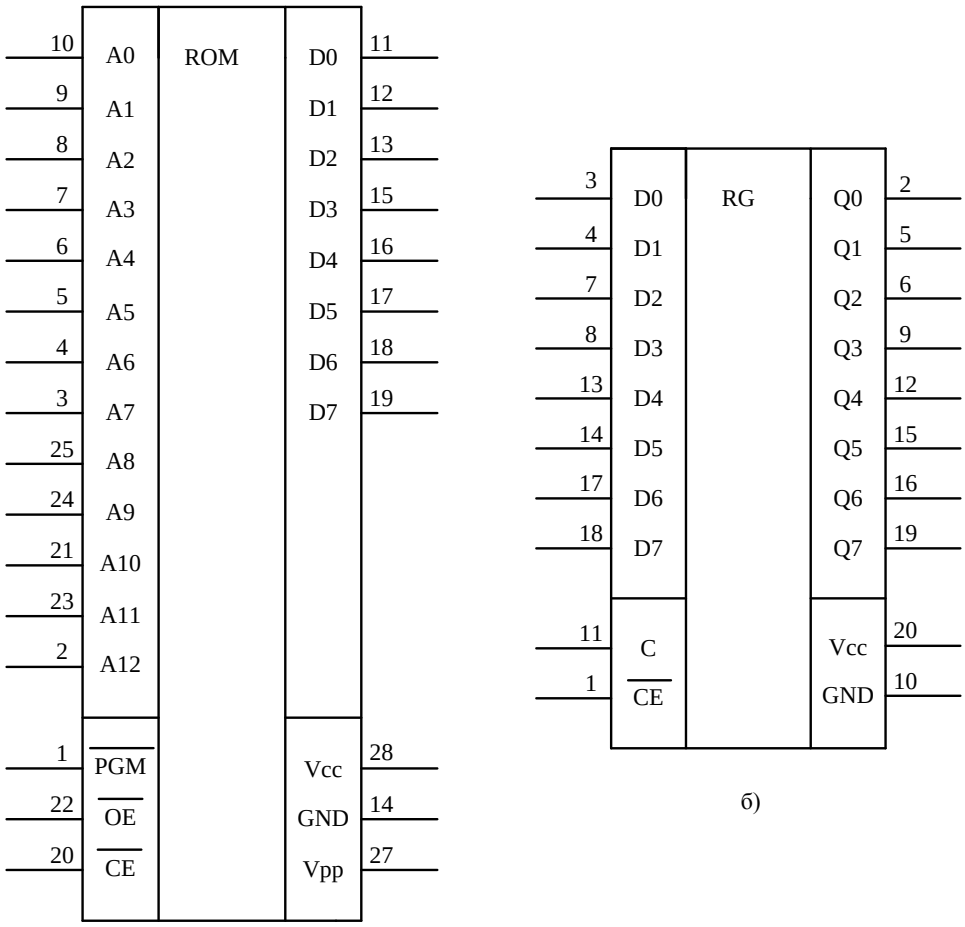
Содержимое постоянного запоминающего устройства называется микропрограммой, а содержимое одной ячейки памяти – микрокомандой. Регистр называют регистром микрокоманд.

Для разработки микропрограммы работы автомата обычно применяется персональный компьютер и специальное программное обеспечение процедуры формального синтеза. Для того чтобы детально разобраться в особенностях программирования автоматов, в данной лабораторной работе применяется неформальная (т.е. “ручная”) разработка микропрограммы. Работу автомата будем изучать методом моделирования в пакете Multisim 11.0.

При моделировании работы автомата в качестве ПЗУ рекомендуется использовать микросхему 27C64-20P из набора компонентов *MSC->ROM*. Эта микросхема ПЗУ имеет объем памяти 64 Кб с организацией 8К ячеек по 8 бит каждая. Таким образом, одна микросхема позволяет создать автомат, разрядность микрокоманды которого не превышает 8 бит. Если исследуемый автомат должен иметь микрокоманду большей разрядности, то следует применить несколько

таких микросхем, включенных параллельно по входам. В качестве регистра рекомендуется использовать микросхему 74LS374N из набора компонентов *TTL*.

Условные графические изображения микросхем 27C64-20P и 74LS374N показаны на рисунке Л4.2. Микросхема ПЗУ имеет входы адреса ячейки памяти A0-A12 (A0 – младший бит адреса), выходы данных D0-D7 (D0 – младший бит данных), а также служебные входы OE, CE и PGM. Активные уровни служебных сигналов – низкие, сигнал CE разрешает работу микросхемы, сигнал OE разрешает работу выходных передатчиков, сигнал PGM разрешает запись информации в микросхему.



а) б)

Рисунок Л4.2 – Условные графические изображения микросхем 27C64 (а) и 74LS374 (б)

Микросхема регистра имеет входы данных D0-D7, выходы данных Q0-Q7, а также служебные входы CE и C. Низкий уровень сигнала на входе CE разрешает

работу микросхемы. Запись байта информации в регистр осуществляется по возрастающему фронту сигнала на входе С.

При моделировании работы устройств в виртуальной среде Multisim напряжение питания на входы Vcc (+5В) и GND (0В) микросхем подавать не требуется.

Для отображения состояния выхода автомата рекомендуется использовать семисегментный индикатор из набора *Indicators->Hex_Display ->DCD_HEX*. Управление таким индикатором осуществляется шестнадцатиричным кодом [5].

СХЕМА МОДЕЛИРОВАНИЯ РАБОТЫ АВТОМАТА

Для облегчения процедуры программирования схему микропрограммного автомата усложняют, запрещая несанкционированные переходы по программе. На рисунке Л4.3 показана обобщенная схема микропрограммного автомата.

В схему автомата введены дополнительные элементы – входной мультиплексор MUX и схема выбора адреса микрокоманды СВА. Кроме того, использовано структуирование цифровой шины передачи данных в обратной связи. Проводники шины сгруппированы в поля – *y(m), Nx, A_1* и *A_0* . Такое усложнение схемы позволяет упростить разработку микропрограммы.

ВАРИАНТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В работе предполагается, что МПА имеет однобитовый управляющий вход *X* и многоразрядный выход *Y*. В зависимости от значения входного сигнала (*X=0* или *X=1*) автомат вырабатывает различные кодовые последовательности. В таблице Л4.1 приведены варианты заданий выходных кодовых последовательностей автомата.

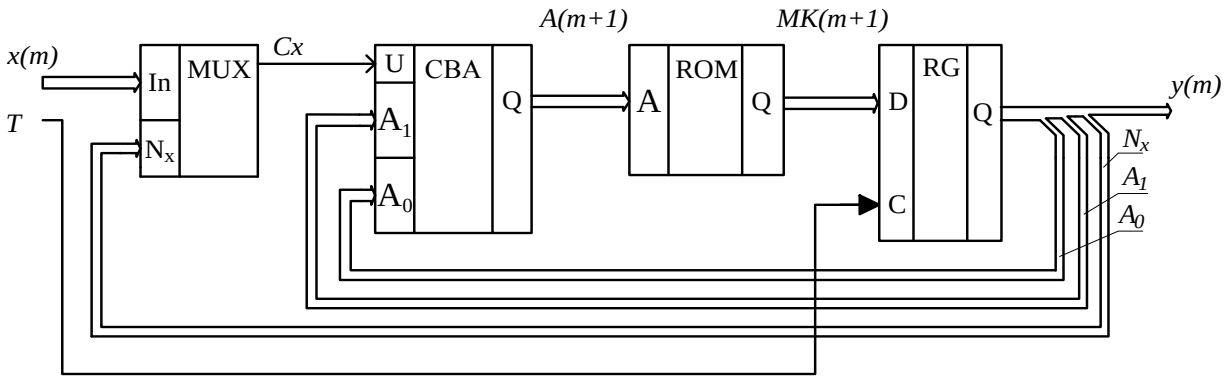


Рисунок Л4.3 – Обобщенная схема микропрограммного автомата

ПРИМЕР РАЗРАБОТКИ МИКРОПРОГРАММЫ РАБОТЫ АВТОМАТА

Для примера рассмотрим вариант программирования автомата для формирования кодовой последовательности: при *X = 0 Y = 0-1-2-3-0-…*, при *X=1 Y = СТОП.*

Таблица Л4.1 – Варианты заданий работы МПА

№ варианта	X = 0	X = 1
1	0-1-0-3-0-6-0-1-…	стоп
2	0-1-3-5-0-1-…	0-5-3-1-0-3…
3	0-1-2-3-2-1-0-1-…	стоп
4	0-2-4-6-0-2-…	0-6-4-2-0-6-…
5	0-2-4-6-0-2-…	0-1-3-6-0-1-…
6	0-1-3-6-0-1-…	0-6-3-1-0-6-…
7	0-4-0-2-0-4-0-2-0-…	0-1-0-3-0-1-0-…
8	0-3-2-1-0-1-…	0-3-0-1-0-3-0-1-…
9	0-1-3-6-0-1-…	0-1-0-3-0-6-0-1-…

На рисунке Л4.4 показан формат используемой микрокоманды (а) и ее графическое изображение (б).

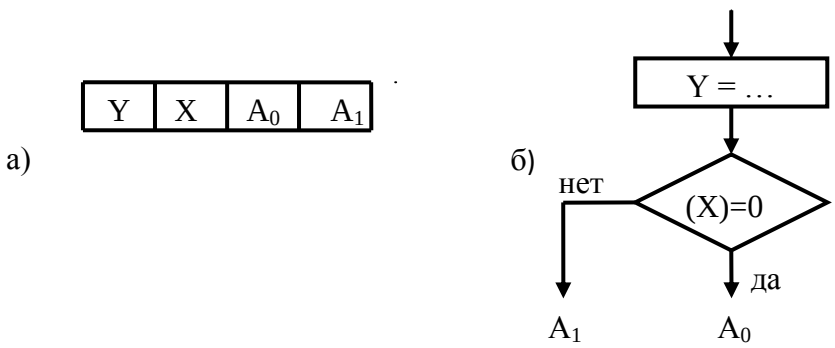


Рисунок 4.4 – МПА с принудительной адресацией и двумя адресными полями: формат микрокоманды (а) и ее графическое изображение (б)

Микрокоманда состоит из 4 полей – поля значения выходной переменной Y (оператор присваивания), поля номера проверяемого входа X (оператор проверки условия) и двух адресных полей A_0 и A_1. Логика исполнения условной части

команды следующая: если на проверяемом входе (X) сигнал 0, то следующая микрокоманда читается из ячейки с адресом A_0 , 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. Учитывая, что адрес следующей микрокоманды выбирается из двух полей (A_0 или A_1), СВА для такого автомата может быть реализована на мультиплексоре.

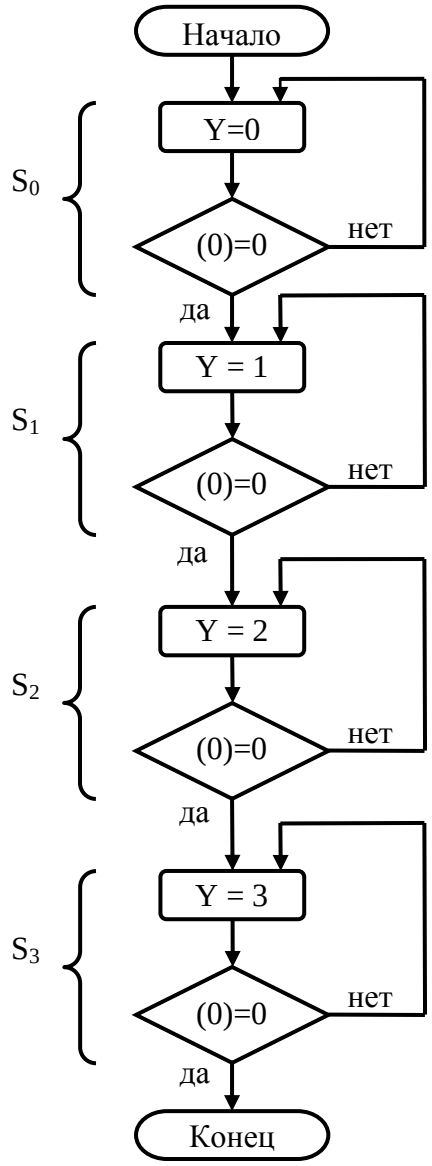


Рисунок 4.5 –Блок-схема программы работы автомата

На рисунке Л4.5 показана блок-схема программы, реализующая кодовую последовательность рассматриваемого примера работы МПА. Она представляет собой 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. У МПА данного типа возможны любые адресные переходы, поэтому команды могут располагаться в любых ячейках памяти программ (постоянном запоминающем устройстве - ПЗУ). При назначении адресов ячеек памяти (индексов i команд) был применен самый простой принцип – команды располагаются последовательно в свободных ячейках по мере появления в блок-схеме программы.

В каждой команде осуществляется проверка состояния входного сигнала и выполняется переход к команде. Если входной сигнал X=0, то читается микрокоманда из ячейки с адресом A_0, если сигнал X = 1, то повторяется выполнение текущей микрокоманды, т.е. реализуется СТОП. Так как расположение команд в памяти

программ определяется номером команды, то легко записать программу работы автомата в виде двоичных кодов.

Программа работы автомата приведена в таблице Л4.2. Как видно из таблицы Л4.2, микрокоманда автомата содержит 7 бит. Всего команд 4. Для кодирования адреса микрокоманды необходимо использовать 2 бита, для кодирования значения выхода необходимо испльзовать тоже 2 бита. Объем ПЗУ, необходимый для записи програмы, составляет M=4*7 = 28 бит.

Таблица Л4.2 – Программа работы автомата

№ ячейки	Данные в ПЗУ			
	Y	X	A_0	A_1
00	00	0	01	00
01	01	0	10	01
10	10	0	11	10
11	11	0	00	11

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Для проверки правильности разработанной программы необходимо собрать схему автомата в пакете Multisim. На рисунке Л4.6 показан пример соединения элементов МПА для микропрограммы, приведенной в таблице Л4.2.

Элементы на рисунке Л4.6 выполняют следующие функции. На вход автомата подается входной сигнал X с переключателя SW1. В процессе моделирования нажатием клавиши *Пробел* можно изменить состояние переключателя и формировать на входе автомата логическую 1 или 0 по желанию пользователя. На логических микросхемах DD1-DD3 собрана схема входного мультиплексора MUX (см. рисунок Л4.3). На микросхеме DD4 (четырехбитовый мультиплексор 2x1, микросхема 74LS257N) реализован элемент выбора адреса следующей команды CBA. Микросхемы DD5 и DD6 представляют ПЗУ и регистр микрокоманд, соответственно. Индикация состояния выхода автомата осуществляется на элементе HL1 – семисегментном индикаторе с входным шестнадцатиричным управляющим кодом. Для тактирования схемы использован функциональный генератор XFG1, настроенный на генерацию прямоугольного сигнала с частотой 1 Гц и скважностью 2. Неиспользуемые входы ПЗУ и регистра подключены к общему проводу (GND).

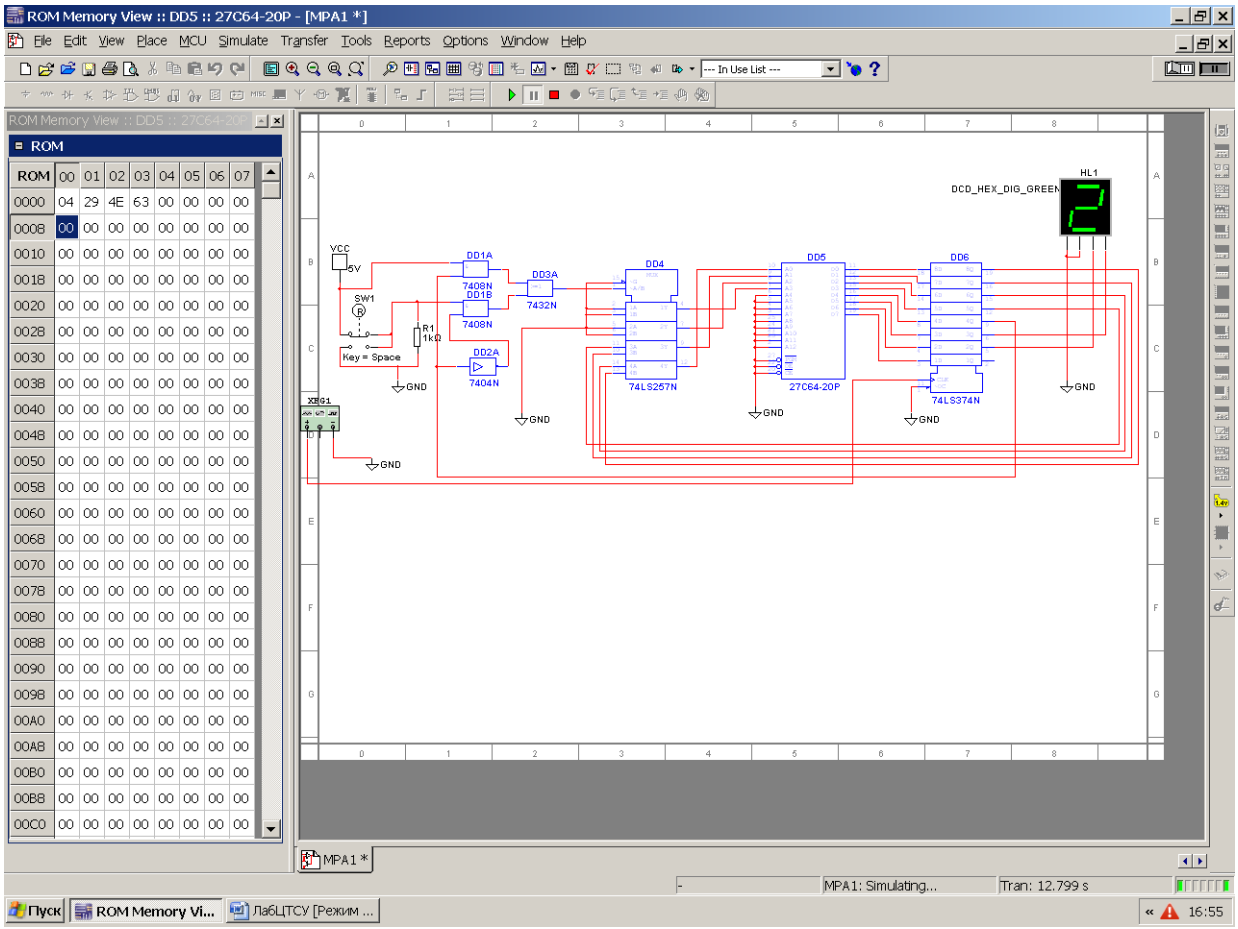


Рисунок Л4.6 – Схема соединения элементов панели для реализации разработанного микропрограммного автомата с принудительной адресацией и двумя адресными полями

2. После сборки схемы необходимо выполнить запись кода программы в ПЗУ. Для этого необходимо выполнить следующие действия:

а) на основной панели инструментов открыть вкладку *MCU -> ROM 27C64-20P DD5->Memory View* . В открывшейся панели отметить пункт *MCU->MCU Windows…->Show windows* - отображать содержимое ПЗУ. При этом откроется окно *ROM Memory View,* в котором будет отображаться содержимое ПЗУ.

б) перевести схему в режим моделирования, нажав на кнопку Run. Поле этого остановить моделирование кнопкой *Pause* (| |). В окне *ROM Memory View* отобразится содержимое ПЗУ – все нули. В соответствии с таблицей Л4.2 занести в ячейки ПЗУ соответствующие данные. Для этого совместить курсор мыши с нужной ячейкой и ЛКМ выделить ее. Заносится код в ячейку посредством клавиатуры. При этом следует помнить, что числа в ПЗУ заносятся в

шестнадцатиричном коде. Так, по таблице Л4.2 в ПЗУ в ячейки с 0 по 3 заносятся коды 04-29-4E-63 соответственно.

в) после занесения кодов в ПЗУ снова перевести схему в режим моделирования кнопкой *RUN*.

1. Проверить работу автомата для входных сигналов $X = 0$ и $X = 1$. Цифровая индикация должна отобразить кодовую последовательность, соответствующую заданию. Продемонстрировать достигнутый результат преподавателю и отметить у него факт выполнение работы.
2. Сохранить схему в файле для оформления отчета по работе.

СОДЕРЖАНИЕ ОТЧЕТА

Кроме титульного листа, отчет должен содержать результаты разработки программы и проверки правильности ее работы. Отчет должен содержать следующие разделы:

1. Задание
2. Блок-схему алгоритма работы МПА
3. Таблицу «прошивки» ПЗУ автомата
4. Расчет требуемых объемов памяти программ и разрядности регистра микрокоманд
5. Схему реализации разработанного микропрограммного автомата
6. Выводы по работе

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что понимается под принудительной адресацией?
2. Какое максимальное число различных значений выходной переменной может содержать один период работы МПА с принудительной адресацией и двумя адресными полями?
3. Как вычислить разрядность поля номера входного сигнала *Nx* в общем случае?
4. Как работает схема выбора адреса следующей микрокоманды автомата?
5. Чему равна длительность одного цикла работы МПА?

ЛАБОРАТОРНАЯ РАБОТА №5
МИКРОПРОГРАММНЫЙ АВТОМАТ С ОДНИМ АДРЕСНЫМ ПОЛЕМ

ЦЕЛЬ РАБОТЫ: изучить особенности программирования автомата с принудительной адресацией и одним адресным полем.

КРАТКИЕ СВЕДЕНИЯ ИЗ ТЕОРИИ

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 выбора адреса СВА.

На рисунке Л5.1 показан формат используемой микрокоманды (а) и ее графическое изображение (б).

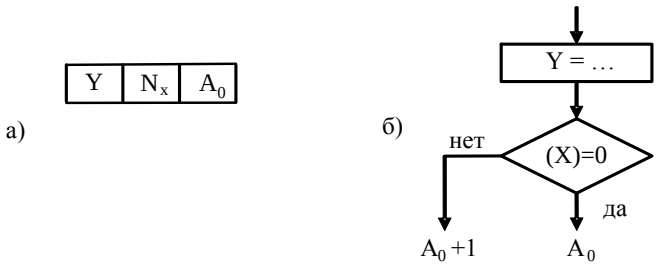


Рисунок Л5.1 – МПА с принудительной адресацией и одним адресным полем: формат микрокоманды (а) и ее графическое изображение (б)

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 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 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 - ветвление программы может быть выполнено только в одну из двух соседних ячеек: A_0 и A_0+1.

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

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

Для реализации данного уравнения можно воспользоваться параллельным сумматором. На рисунке Л5.2 показана реализация схемы выбора адреса

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

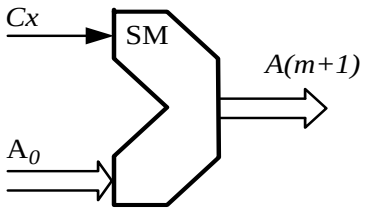


Рисунок Л5.2 – Реализация схемы выбора адреса следующей микрокоманды

ВАРИАНТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В работе предполагается, что МПА имеет одноразрядный управляющий вход *X* и многоразрядный выход *Y*. В зависимости от значения входного сигнала (*X*=0 или *X*=1) вырабатываются различные кодовые последовательности. В таблице Л4.1 приведены варианты заданий кодовых последовательностей автомата.

ПРИМЕР ПРОГРАММИРОВАНИЯ АВТОМАТА

Для примера рассмотрим вариант программирования автомата для формирования кодовой последовательности: при *X* = 0 *Y* = 0-1-2-3-0-…, при *X*=1 *Y* = СТОП.

На рисунке Л5.3 показана блок-схема программы, реализующая кодовую последовательность рассматриваемого примера работы МПА. Она представляет собой 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. У МПА данного типа возможны адресные переходы только на две соседние ячейки, поэтому команды должны располагаться парами в соседних ячейках памяти программ. При назначении адресов ячеек памяти (индексов i команд) эта особенность работы автомата была учтена.

Основная проблема программирования такого автомата связана с невозможностью выполнить некоторые требуемые переходы. Так, например, команда S_0 программы для автомата с двумя адресными полями (см. рисунок Л5.3) содержит переходы к ячейке S_2 и к самой себе, причем при X=0 переход должен быть выполнен к ячейки S_2. 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_2, то при X=1 автомат самостоятельно перейдет к ячейке S_3 (автомат вычислит адрес перехода как S_2 + 1). Основной

способ устранить данное противоречие – продублировать требуемую команду, поместив ее в ячейку памяти, номер которой вычисляет автомат.

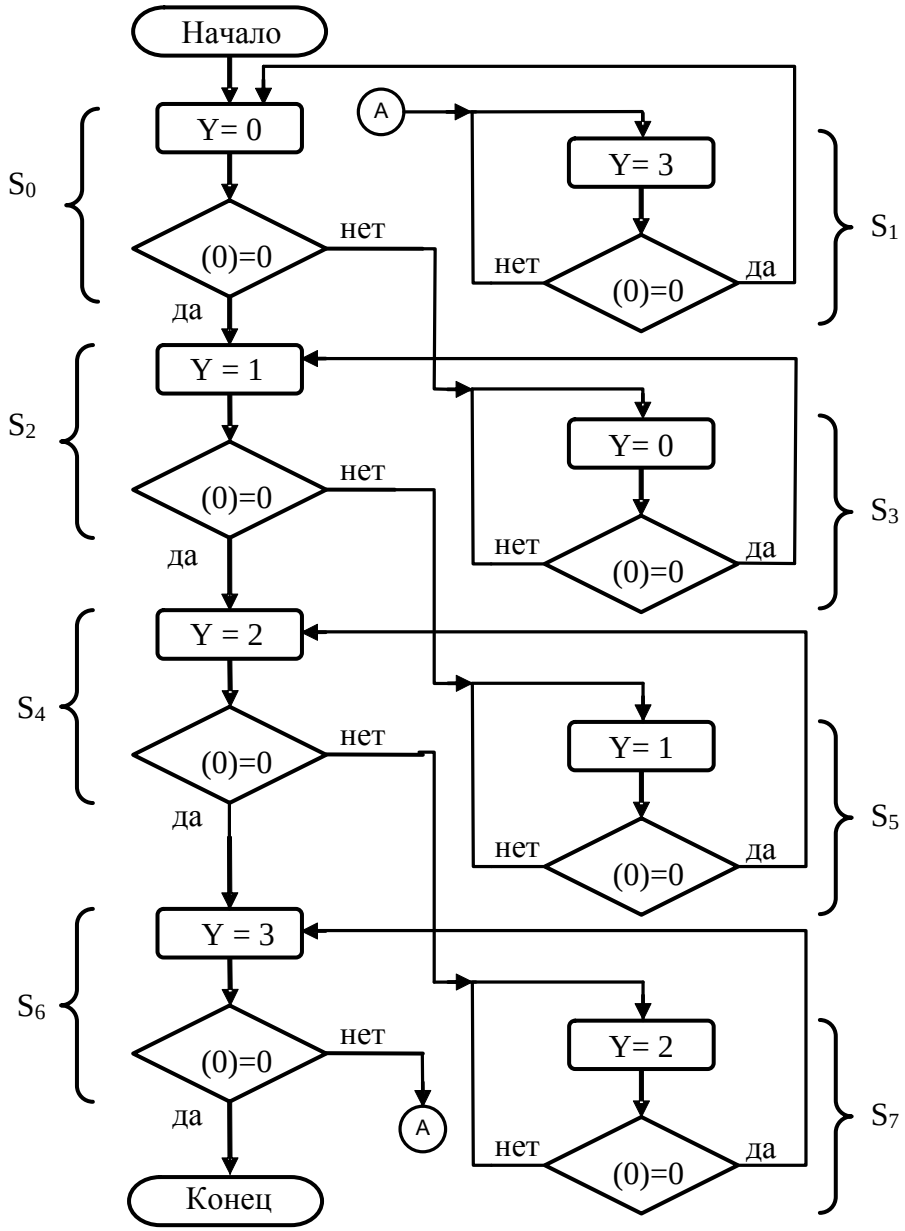


Рисунок Л5.3 – Блок-схема программы работы автомата с принудительной адресацией и одним адресным полем

Для устранения противоречий на рисунке Л5.3 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 дублирующих команд программа для автомата с принудительной адресацией и одним адресным полем оказывается более «длинной», чем

программа для автомата с принудительной адресацией и двумя адресными полями.

Так как расположение команд в памяти программ определяется номером команды, то легко записать программу работы автомата в виде двоичных кодов. Программа работы автомата приведена в таблице Л5.2.

Таблица Л5.2 – Программа работы автомата с принудительной адресацией и одним адресным полем

№ ячейки	Данные в ПЗУ		
	Y	X	A_0
000	00	0	010
001	11	0	000
010	01	0	100
011	00	0	010
100	10	0	110
101	01	0	100
110	11	0	000
111	10	0	110

Как видно из таблицы Л5.2, микрокоманда автомата содержит 6 бит. Всего команд 8. Для кодирования адреса микрокоманды необходимо использовать 3 бита, для кодирования значения выхода – 2 бита. Объем ПЗУ, необходимый для записи программы, составляет M=8*6 = 48 бит.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Для проверки правильности разработанной программы необходимо собрать схему автомата в пакете Multisim. На рисунке Л5.4 показан пример соединения элементов МПА для микропрограммы, приведенной в таблице Л5.2.

Элементы на рисунке Л5.4 выполняют следующие функции. Переключатель SW1 позволяет подавать на вход автомата входной сигнал *X*. В процессе моделирования нажатием клавиши *Пробел* можно изменить состояние переключателя и формировать на входе автомата логическую 1 или 0 по желанию пользователя. На логических микросхемах DD1-DD3 собрана схема входного мультиплексора MUX (см. рисунок Л4.3). На микросхеме DD4 (четырехбитовый параллельный сумматор, микросхема 74LS83N) реализован элемент выбора адреса следующей команды СВА. Микросхемы DD5 и DD6 представляют ПЗУ и

регистр микрокоманд, соответственно. Индикация состояния выхода автомата осуществляется на элементе HL1 – семисегментном индикаторе с входным шестнадцатиричным управляющим кодом. Для тактирования схемы использован функциональный генератор XFG1, настроенный на генерацию прямоугольного сигнала с частотой 1 Гц и скважностью 2. Неиспользуемые входы ПЗУ и регистра подключены к общему проводу (GND).

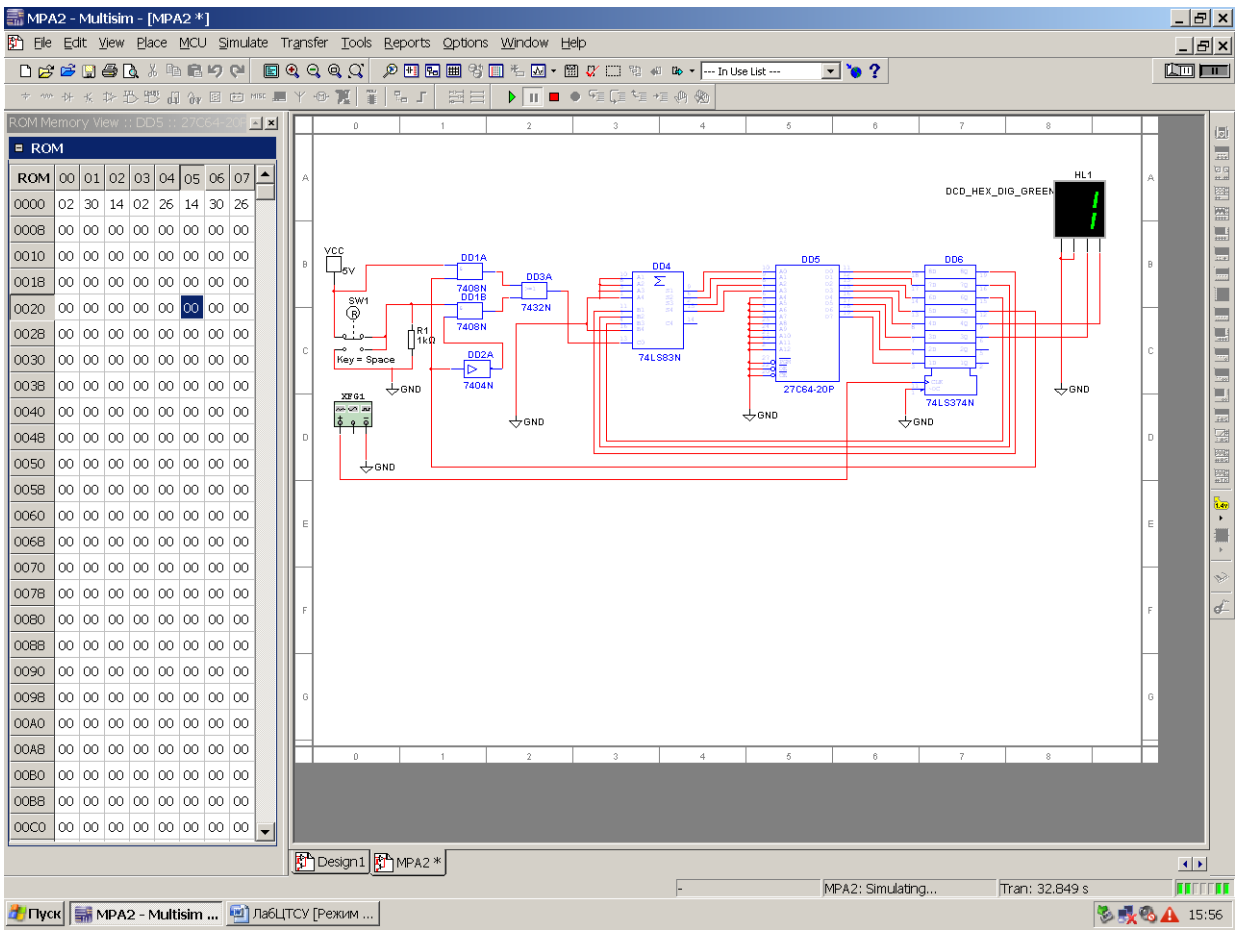


Рисунок Л5.4 – Схема соединения элементов для реализации разработанного микропрограммного автомата с принудительной адресацией и одним адресным полем

2. После сборки схемы необходимо выполнить запись кода программы в ПЗУ. Для этого необходимо выполнить следующие действия:

а) на основной панели инструментов открыть вкладку *MCU -> ROM 27C64-20P DD5->Memory View* . В открывшейся панели отметить пункт *MCU->MCU Windows…->Show windows* - отображать содержимое ПЗУ. При этом откроется окно *ROM Memory View*, в котором будет отображаться содержимое ПЗУ.

б) перевести схему в режим моделирования, нажав на кнопку Run. После этого остановить моделирование кнопкой *Pause* (| |). В окне *ROM Memory View* отобразится содержимое ПЗУ – все нули. В соответствии с таблицей Л5.2 занести в ячейки ПЗУ соответствующие данные. Для этого следует совместить курсор мыши с нужной ячейкой и ЛКМ выделить ее. Заносится код в ячейку посредством клавиатуры. При этом следует помнить, что числа в ПЗУ заносятся в шестнадцатиричном коде. Так, в соответствие с данными таблицы Л5.2 в ПЗУ в ячейки с 0 по 7 заносятся коды 02-30-14-02-26-14-30-26 соответственно.

в) после занесения кодов в ПЗУ снова перевести схему в режим моделирования кнопкой *RUN*.

3. Проверить правильность работы автомата для входных сигналов X=0 и X=1. Выходная последовательность должна соответствовать заданию. Продемонстрировать достигнутый результат преподавателю и отметить у него факт выполнения работы.

Сохранить схему в файле для оформления отчета по работе.

СОДЕРЖАНИЕ ОТЧЕТА

Кроме титульного листа, отчет должен содержать результаты разработки программы и проверки правильности ее работы. Он должен содержать следующие разделы:

1. Задание
2. Блок-схему алгоритма работы МПА
3. Таблицы «прошивки» ПЗУ автомата
4. Расчет требуемых объемов памяти программ и разрядности регистра микрокоманд
5. Схему разработанного микропрограммного автомата
6. Выводы по работе

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что понимается под принудительной адресацией?
2. Какое максимальное число различных значений выходной переменной может содержать один период работы МПА с принудительной адресацией и одним адресным полем?
3. Как оценить максимальный объем ПЗУ до разработки микропрограммы?
4. Как работает схема выбора адреса следующей микрокоманды автомата?

ЛАБОРАТОРНАЯ РАБОТА №6
МИКРОПРОГРАММНЫЙ АВТОМАТ С РАЗДЕЛЕННОЙ ЕСТЕСТВЕННОЙ АДРЕСАЦИЕЙ

ЦЕЛЬ РАБОТЫ: изучить особенности программирования автомата с разделенной естественной адресацией.

КРАТКИЕ СВЕДЕНИЯ ИЗ ТЕОРИИ

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 вычисления адреса A_0 перекладывается на схему выбора 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

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

где $A(m)$ – адрес текущей исполняемой микрокоманды; $A(m+1)$ – адрес следующей исполняемой микрокоманды, m – номер такта.

У автомата с разделенной естественной адресацией микрокоманды разделены на два типа: команды присваивания и команды проверки условия перехода. Для идентификации типа микрокоманды применяется специальный бит – признак микрокоманды p. Примем, что микрокоманде присваивания соответствует значение $p = 0$, микрокоманде проверки условия перехода – значение $p = 1$. На рисунке Л6.1 показан формат используемых микрокоманд (а) и их графическое изображение (б).

Микрокоманда присваивания состоит из 2 полей – поля признака микрокоманды $p = 0$ и поля значения выходной переменной Y. После исполнения команды присваивания выходная величина Y принимает заданное

значение, а из следующей по порядку возрастания адреса ячейки (A_{m+1}) читается очередная микрокоманда.

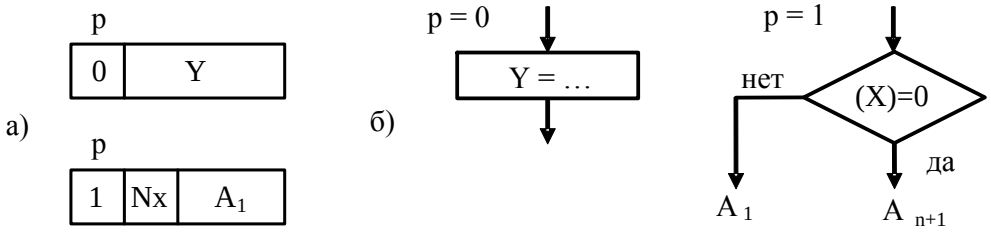


Рисунок Л6.1 – МПА с естественной адресацией и разделенной микрокомандой: формат микрокоманд (а) и их графическое изображение (б)

Микрокоманда проверки условия состоит из 3 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 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_{m+1}, иначе выполняется переход – микрокоманда читается из ячейки с адресом, указанным в поле A_1 . Так как обе микрокоманды читаются из одного и того же ПЗУ, то разрядности микрокоманд присваивания и проверки условия должны быть равными.

Схемы выбора адреса следующей микрокоманды может быть реализована на счетчике импульсов [6]. На рисунке Л6.2 показана реализация схемы выбора адреса следующей микрокоманды автомата с естественной адресацией. Если на счетный вход подать тактовые импульсы, то с приходом очередного тактового импульса содержимое счетчика увеличится на единицу, реализуя естественную последовательность выполнения команд. Для выполнения перехода по адресу A_1 необходимо в счетчик импульсов записывать новое число из поля A_1 микрокоманды. Для этой цели используются входы D предустановки счетчика, которые активизируются при поступлении единицы на управляющий вход U.

Реализация разделенной естественной адресации требует применения специальных элементов для выделения команд присваивания и ветвления микропрограммы. Так, команды ветвления можно запоминать в специальном регистре – защелке данных, который активируется при поступлении сигнала p=0. Выделение команды ветвления может быть выполнено включением элемента «Логическое И» в цепь сигнала Cx. Назначение этого элемента – пропускать сигнал Cx на вход U счетчика микрокоманд при поступлении сигнала p=1.

Детальная информация по этому вопросу приведена ниже в примере реализации автомата.

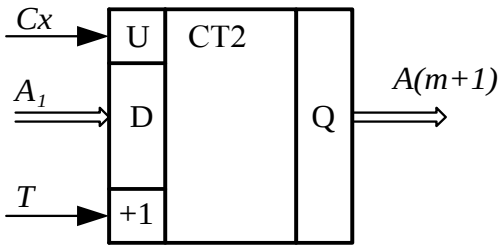


Рисунок Л6.2 – Реализация схемы выбора адреса следующей микрокоманды автомата с естественной адресацией

ВАРИАНТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В работе предполагается, что МПА имеет одноразрядный управляющий вход *X* и многоразрядный выход *Y*. В зависимости от значения входного сигнала (*X*=0 или *X*=1) вырабатываются различные кодовые последовательности. В таблице Л4.1 приведены варианты заданий кодовых последовательностей автомата.

ПРИМЕР ПРОГРАММИРОВАНИЯ АВТОМАТА

Для примера рассмотрим вариант программирования автомата для формирования кодовой последовательности: при *X*=0 *Y* = 0-1-2-3-0-…, при *X*=1 *Y* = СТОП.

На рисунке Л6.3 показана блок-схема программы, реализующая кодовую последовательность рассматриваемого примера работы МПА. Она содержит последовательность отдельных команд, 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 по адресу A=0.

Для разрешения указанной проблемы у автомата можно организовать дополнительный вход. Этот вход аппаратно подключается к единице. В случаях, когда необходимо выполнить принудительный переход к требуемой ячейке памяти, независимо от состояния управляющего входа, следует задать условную команду по дополнительному входу.

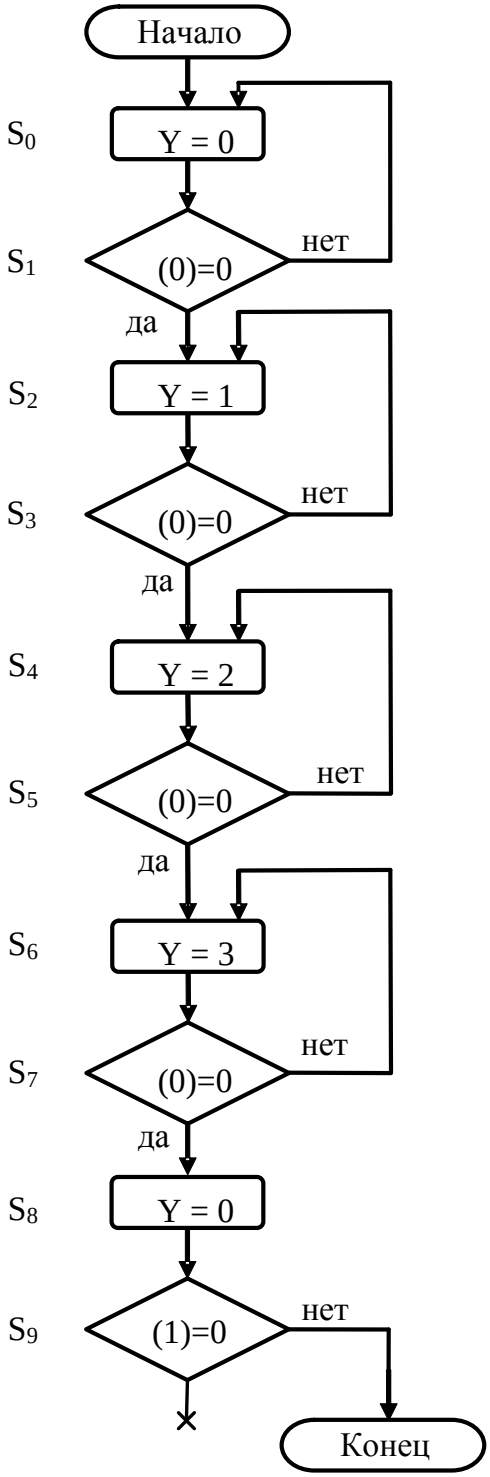


Рисунок Л6.3–Блок-схема программы работы автомата

В рассматриваемом примере организован дополнительный вход №1, на который аппаратно подана единица (т.е. вход №1 аппаратно подключен к линии питания +5 В). Зацикливание программы осуществляется в последней микрокоманде (S_9) алгоритма путем проверки состояния дополнительного входа. Эта микрокоманда дублирует микрокоманду S_0, переход к которой невозможен при исполнении микрокоманды S_8 при управляющем сигнале на входе №0 равном 0. Так как расположение команд в памяти определяется естественным ходом выполнения программы, то микрокоманды нумеруются (номер соответствует адресу микрокоманды в памяти программ) по ходу алгоритма при условии, что сигнал на управляющем входе №0 равен 0. Программа работы автомата приведена в таблице Л6.2.

Как видно из таблицы Л6.2, микрокоманда автомата содержит 6 бит. Всего команд 10. Для кодирования адреса микрокоманды необходимо использовать 4 бита, для кодирования выхода необходимо использовать 2 бита. Объем ПЗУ, необходимый для записи программы работы автомата с разделенной естественной адресацией, составляет M=10*6, т.е. 60 бит.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ

1. Для проверки правильности разработанной программы необходимо собрать схему автомата на панели пакета Multisim. На рисунке Л6.4 показан пример соединения элементов МПА с разделенной естественной адресацией. Элементы на рисунке Л6.4 выполняют следующие функции.

Переключатель SW1 позволяет подавать на вход автомата входной сигнал *X*. В процессе моделирования нажатием клавиши *Пробел* можно изменить состояние переключателя и формировать на входе автомата логическую 1 или 0 по желанию пользователя. На логических микросхемах DD1-DD3 собрана схема входного мультиплексора MUX (см. рисунок Л4.3). На микросхеме DD5 (четырехбитовый счетчик импульсов, микросхема 74LS161N) реализован элемент выбора адреса следующей команды СВА.

Таблица Л6.2 – Программа работы автомата с разделенной естественной адресацией

№ ячейки	Данные в ПЗУ	
	P	Y/ X A_1
0000	0	0000
0001	1	0000
0010	0	0001
0011	1	0010
0100	0	0010
0101	1	0100
0110	0	0011
0111	1	0110
1000	0	0000
1001	1	1000

Схема счетчика с указанием номеров выводов показана на рисунке Л6.5. Счетчик импульсов имеет счетный вход CLK, на который поступают тактовые импульсы. Поступление тактового импульса увеличивает число в счетчике на 1. Двоичный код числа счетчика выводится на выходы Q0-Q3 (Q0 – младший бит). Число в счетчике может изменяться от 0 до 15. С приходом 16-го импульса счетчик обнуляется, а на выходе P2 появляется единица переноса. Для предварительной записи числа в счетчик (предустановка счетчика) используются входы данных D0-D3 (D0- младший бит). При поступлении на управляющий вход V1 логического нуля данные с входов D0-D3 записываются в счетчик. Дополнительные управляющие входы счетчика реализуют следующие функции: R - сброс содержимого счетчика в ноль; V2- разрешение счета импульсов; P1 – разрешение переноса.

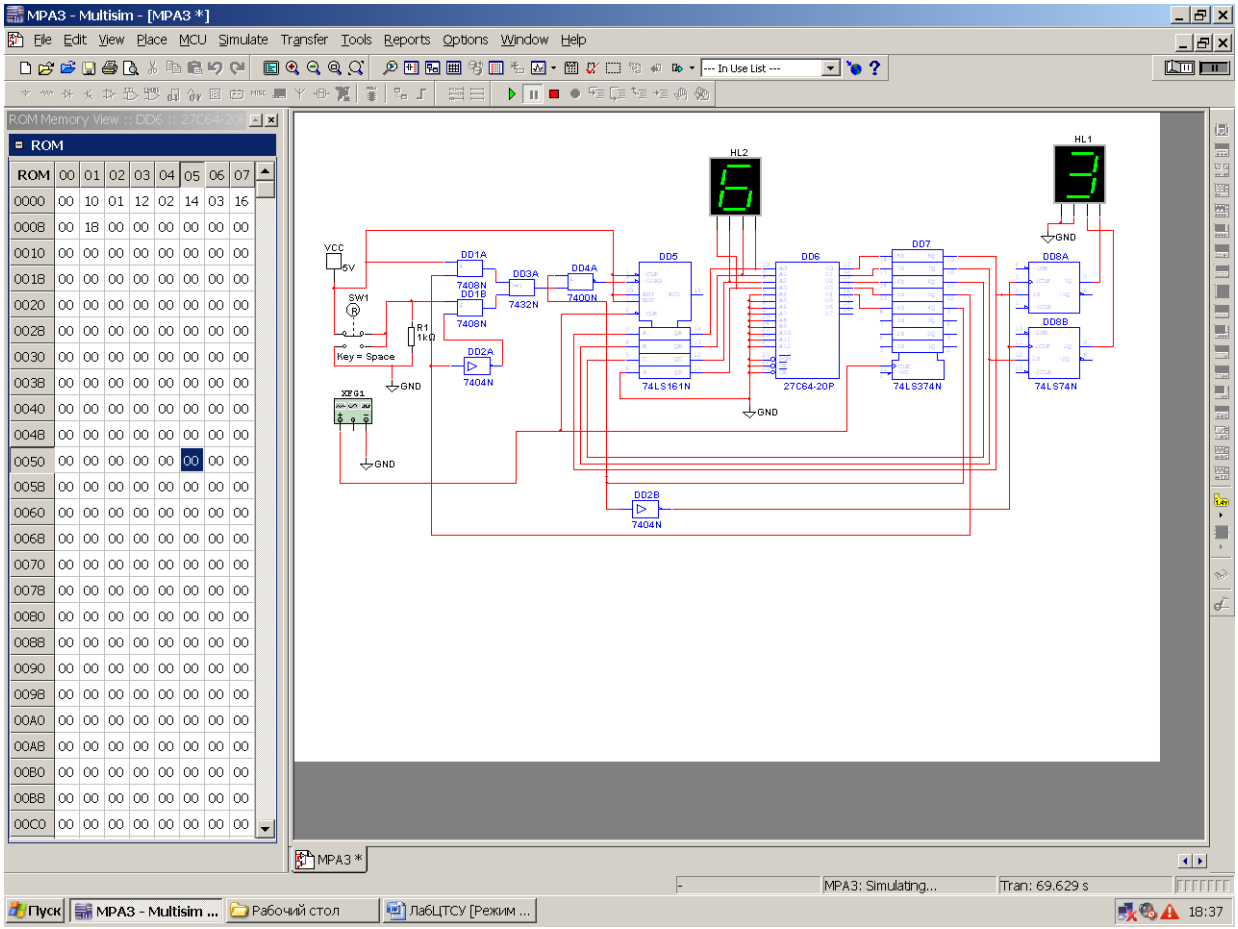


Рисунок Л6.4 - Схема соединения элементов для проверки микропрограммного автомата с разделенной естественной адресацией

Микросхемы DD5 и DD6 представляют ПЗУ и регистр микрокоманд соответственно. Индикация состояния выхода автомата осуществляется на элементе HL1 – семисегментном индикаторе с входным шестнадцатиричным управляющим кодом. Аналогичный индикатор HL2 использован для отображения номера текущей исполняемой микрокоманды. Для тактирования схемы использован функциональный генератор XFG1, настроенный на генерацию прямоугольного сигнала с частотой 1 Гц и скважностью 2. Неиспользуемые входы ПЗУ и регистра подключены к общему проводу (GND).

Элемент DD4A (7400N – элемент 2И-НЕ) служит для разрешения параллельной загрузки счетчика DD5 и реализации микрокоманд ветвления (при p=1). Действительно, при p=1 и Cx=1 на выходе DD4A образуется логический 0, который является активным сигналом на входе V1, стробируя параллельную загрузку счетчика импульсов DD5.

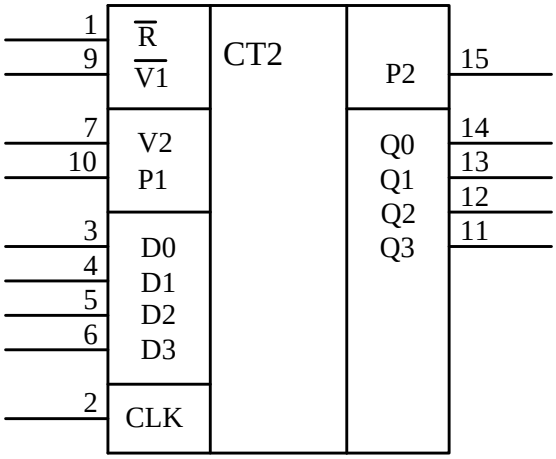


Рисунок Л6.5 – Счетчик импульсов К1555ИЕ10 (74LS161N)

Вывод информации на семисегментный индикатор HL1 должен осуществляться только при выполнении команд присваивания (при *p*=0). Для этого разряды микрокоманды Y0 и Y1 запоминаются в двухбитовом регистре–защелке DD8. Этот регистр реализован на микросхеме К1555ТМ2 (аналог 74LS74N), цоколевка которой показана на рисунке Л6.6. Входы R и S являются асинхронными входами установки триггеров в состояние 0 и 1 соответственно (активный уровень сигнала нулевой). При поступлении на вход синхронизации С активного высоко уровня информация с входа D записывается в триггер. При С=0 выходной сигнал триггера сохраняется неизменным. В схеме автомата на рисунке Л6.4 на вход С триггеров DD8 подается инвертированный элементом DD2B сигнал *p*. За счет этого информация записывается в триггеры при значении *p*=0 (т.е. при выполнении команды присваивания).

2. После набора схемы необходимо выполнить запись кода программы в ПЗУ. Для этого необходимо выполнить следующие действия:

а) на основной панели инструментов открыть вкладку *MCU -> ROM 27C64-20P DD5->Memory View* . В открывшейся панели отметить пункт *MCU->MCU Windows…->Show windows* - отображать содержимое ПЗУ. При этом откроется окно *ROM Memory View,* в котором будет отображаться содержимое ПЗУ.

б) перевести схему в режим моделирования, нажав на кнопку Run. Поле этого остановить моделирование кнопкой *Pause* (| |). В окне *ROM Memory View* отобразится содержимое ПЗУ – все нули. Необходимо занести в ячейки ПЗУ данные в соответствии с таблицей Л6.2. Для этого следует совместить курсор мыши с нужной ячейкой и ЛКМ выделить ее. Заносится код в ячейку

посредством клавиатуры. При этом следует помнить, что числа в ПЗУ заносятся в шестнадцатиричном коде. Так, в соответствии с данными таблицы Л6.2 в ячейки с 0 по 9 заносятся коды 0-10-01-12-02-14-03-16-0-18.

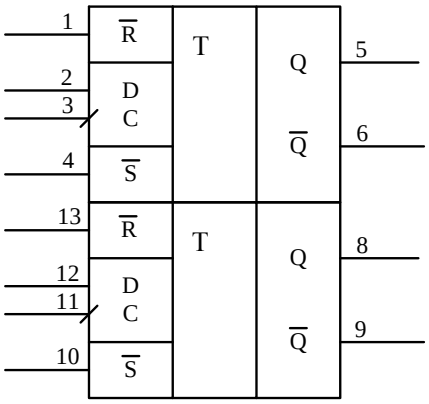


Рисунок Л6.6 – Два D-триггера К1555ТМ2 (74LS74N)

в) после занесения кодов в ПЗУ следует снова перевести схему в режим моделирования кнопкой *RUN*.

3. Проверить правильность работы автомата для входных сигналов X=0 и X =1. Цировая выходная последовательность автомата должна соответствовать заданию. Продемонстрировать достигнутый результат преподавателю и отметить у него факт выполнение работы.

Сохранить схему в файле для оформления отчета по работе.

СОДЕРЖАНИЕ ОТЧЕТА

Кроме титульного листа, отчет должен содержать результаты разработки программы и проверки правильности ее работы. Он должен содержать следующие разделы:

1. Задание
2. Блок-схему алгоритма работы МПА
3. Таблицы «прошивки» ПЗУ автомата
4. Расчет требуемых объемов памяти программ и разрядности регистра микрокоманд
5. Схему разработанного микропрограммного автомата
6. Выводы по работе

КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что понимается под естественной адресацией?

2. Какое максимальное число различных значений выходной переменной может содержать один период работы МПА с разделенной естественной адресацией?
3. Как оценить максимальный объем ПЗУ до разработки микропрограммы?
4. Как работает схема выбора адреса следующей микрокоманды автомата?
5. Как реализуется переход по адресу A_1?
6. Для чего применен инвертор в цепи сигнала p, подающегося на выходной регистр-защелку Y?
7. Сколько различных периодических последовательностей может сформировать разработанный МПА?

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Бойков В.И., Болтунов Г.И., Быстров С.В., Григорьев В.В., Литвинов Ю.В. Цифровая техника систем управления: Учебное пособие. – Университет ИТМО, 2018.- 139 с.
2. Букреев И.Н. Микроэлектронные схемы цифровых устройств / И.Н.Букреев, В.Н.Горячев, Б.М.Мансуров. – М.: Техносфера, 2009. – 708 с.
3. Григорьев В.В., Быстров С.В., Бойков В.И., Болтунов Г.И., Мансурова О.К. Цифровые системы управления: Учебное пособие. – СПб: Университет ИТМО, 2019. – 133 с.
4. Опадчий Ю.Ф., Глудкин О.П., Гуров А.И. Аналоговая и цифровая электроника (полный курс): Учебник для вузов. Под ред. О. П. Глудкина. - М.: Горячая линия - Телеком, 2003. - 768 с. ил.
5. Пряшников В.А. Электроника: Полный курс лекций. СПб.: КОРОНА принт; М.: Бином-Пресс, 2006.
6. Новиков Ю.В. Основы цифровой схемотехники. Базовые элементы и схемы. Методы проектирования - М.: Мир, 2001. - 379с.
7. Марченко А.Л., Освальд С.В. Лабораторный практикум по электротехнике и электронике в среде Multisim. Учебное пособие для вузов. – М.: ДМК Пресс, 2010. – 448 с.: ил.
8. Решение прикладных задач теории управления в МАТЛАБ: Учеб. Пособие / Смирнов Н.В., Смирнов М.Н., Смирнова М.А. – СПб.: Издательство “СОЛО”, 2013. – 186 с.
9. Филлипс Ч., Харбор Р. Системы управления с обратной связью. – М.: Лаборатория Базовых Знаний, 2001 – 616 с., ил. ISBN 0-13-949090-6

Бойков Владимир Иванович
Болтунов Геннадий Иванович
Быстров Сергей Владимирович
Григорьев Валерий Владимирович
Литвинов Юрий Володарович

Цифровая техника систем управления. Лабораторный практикум

В авторской редакции
Редакционно-издательский отдел Университета ИТМО
Зав. РИО Н.Ф. Гусарова
Подписано к печати
Заказ №
Тираж
Отпечатано на ризографе

Редакционно-издательский отдел
Университета ИТМО
197101, Санкт-Петербург, Кронверкский пр., 49