МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ

ФГБОУ ВО «Воронежский государственный технический университет»

А. А. Пирогов, А. Б. Буслаев

ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ ФУНКЦИОНАЛЬНЫХ УЗЛОВ НА ОСНОВЕ ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ ИНТЕГРАЛЬНЫХ СХЕМ

Практикум

Утверждено учебно-методическим советом университета
в качестве учебного пособия

Воронеж 2018

УДК 621.382(075.8)
ББК 32.85я7
П334

***Рецензенты**:*
кафедра основ радиотехники и электроники Федеральное казённое образовательное учреждение высшего образования «Воронежский институт Федеральной службы исполнения наказаний»
(начальник кафедры канд. техн. наук, доцент Р.Н. Андреев);
канд. техн. наук, доцент С.М. Федоров

Пирогов, А. А.

П334 **Проектирование цифровых функциональных узлов на основе программируемых логических интегральных схем: практикум** [Электронный ресурс]. – Электрон. текстовые и граф. данные (2,2 Мб) / А. А. Пирогов, А. Б. Буслаев. - Воронеж: ФГБОУ ВО «Воронежский государственный технический университет», 2018. – 1 электрон. опт. диск (CD-ROM): цв. – Систем. требования: ПК 500 и выше; 256 Мб ОЗУ; Windows XP; SVGA с разрешением 1024x768; Adobe Acrobat; CD-ROM дисковод; мышь. – Загл. с экрана.

ISBN 978-5-7731-0649-4

В учебном пособии изложены требования и рекомендации по подготовке и выполнению лабораторных работ, практических занятий и курсового проекта по программе курса «Интегральные устройства радиоэлектроники».

Издание соответствует требованиям Федерального государственного образовательного стандарта высшего образования по направлению 11.03.03 «Конструирование и технология электронных средств» (профиль «Проектирование и технология радиоэлектронных средств»).

Ил. 41. Табл. 7. Библиогр.: 4 назв.

УДК 621.382(075.8)
ББК 32.85я7

ISBN 978-5-7731-0649-4

ВВЕДЕНИЕ

Автоматизированное проектирование – это основное направление развития технологий создания электронной аппаратуры. До недавнего времени основным методом решения подобных задач считались разработки на базе серийно выпускаемых стандартных интегральных микросхем, и большинство систем автоматизированного проектирования (САПР) ориентировались на реализацию именно этой технологии проектирования.

Наиболее востребованным вариантом являются САПР сквозного проектирования, которые позволяют выполнить всю цепочку разработки: от входного описания до создания эскиза.

В последние годы появляются новые технологии проектирования электронной аппаратуры, основанные на современных программных продуктах, интегрированных САПР и программируемых логических интегральных схем (ПЛИС). Одним из представителей данных САПР являются среды проектирования Active-HDLи Xilinx ISE, позволяющие проводить моделирование как на схемной уровне, так и на уровне прямого программирования.

На языке описания аппаратуры (HDL) производится описание поведения или структура разрабатываемого устройства, далее осуществляется моделирование, где выявляются и устраняются ошибки проекта. После чего производится синтез и реализация проекта на базе программируемых логических интегральных схем (ПЛИС).

1. ОСНОВНЫЕ ПРИНЦИПЫ И МЕТОДЫ ПОСТРОЕНИЯ МОДЕЛЕЙ ФУНКЦИОНАЛЬНЫХ УЗЛОВ НА ЛОГИЧЕСКОМ УРОВНЕ

1.1. Проектирование цифровых устройств в среде Active-HDL

1.1.1. Ввод проекта

В начале проектирования разработчику представлены три окна интегрированной среды Active-HDL 7.1:

• **Design Browser –** окно просмотра проекта, в котором отображается все содержимое рабочего пространства.

• **Console –** окно для интерактивного ввода и/или вывода текстовой информации. Все инструментальные средства пакета выдают в это окно сообщения о своей работе, в том числе предупреждения и ошибки.

• В третьем окне активна закладка **Design Flow Manager** («Менеджер маршрута проектирования»). Обычно данное окно занимает большую часть экрана потому, что именно здесь пользователь создает основные документы проекта. На закладке Design Flow Manager изображен доступный для заданных установок маршрут проектирования. Он ограничивает действия тремя способами описания проекта (HDE, FSM, BDE) и последующим функциональным моделированием (functional simulation).

Если необходимо создать проект в текстовом формате на языке VHDL, Verilog или SystemC, то следует нажать кнопку HDE. В качестве рабочего инструмента будет вызван текстовый редактор HDE (от слов Hardware Description language Editor).

Если необходимо представить описание проекта в виде диаграммы состояний конечного автомата (Finite State Machine), то следует использовать редактор FSM.

Для реализация функциональных узлов может быть использован встроенный схемный редактор BDE (Block

Diagram Editor). Вызвав BDE, запускается «мастер» создания нового исходного файла **New Source File Wizard.** Необходимо следовать его командам, далее приступаем к созданию проекта.

Создание проекта. На первом этапе необходимо найти и разместить на чертеже схемы графические изображения нужных нам элементов. Чтобы открыть библиотеку элементов, необходимо воспользоваться инструментом **Show Symbols Toolbox** (см. рис. 1.1.1).

Рис. 1.1.1. Панель инструментов среды проектирования Active-HDL

В режиме, заданном по умолчанию, для построения схемы доступна одна библиотека со встроенными символами **Built-in symbols**. Раскрыв ее содержимое, вы увидите набор простейших логических элементов (см. рис. 1.1.2).

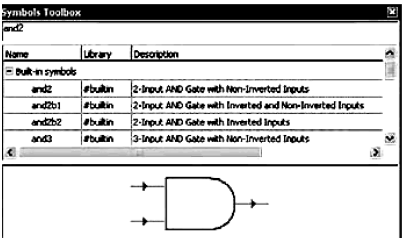

Рис. 1.1.2. Диалоговая панель со встроенными символами

Для построения мультиплексора необходимы элементы (and2, inv, or2) размещаются они в зоне проекта известным методом Drag & Drop. Чтобы сделать видимыми позиционные обозначения символов, исполним команду **View Texts** из

меню Diagram. На открывшейся панели View Texts устанавливается флажок NAME. Позиционные обозначения всех символов на схеме станут видимыми (см. рис. 1.1.3).

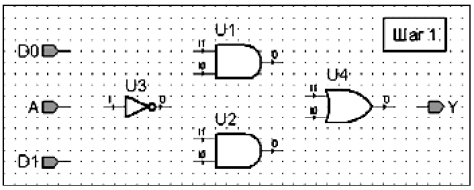

Рис. 1.1.3. Символы с указанными позиционными обозначениями

На втором шаге элементы схемы необходимо соединить проводниками в соответствии со структурой мультиплексора. Нажмем на панели инструментов **Diagram Items** кнопку **Wire** или на клавиатуре клавишу **W.**

Поместите курсор в точку, из которой необходимо начать проводник, и щелкните левой кнопкой мыши (ЛКМ). Переместите курсор в точку назначения и опять щелкните ЛКМ. Если проводник имеет сложную форму, придется фиксировать каждый его сегмент (излом, угол) щелчком мыши. Законченная схема должна выглядеть так, как показано на рис. 1.1.4.

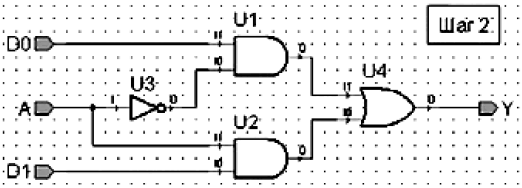

Рис. 1.1.4. Окончательная схема мультиплексоры

На третьем этапе проводникам схемы надо назначить имена. Цепи, подключенные к терминалам (входным и выходным портам), автоматически получают те же самые

названия, что и соответствующий порт. А внутренние проводники система именует по шаблону (NET9, NET13, NET17 и др.), у них есть системные имена. Однако если пользователь собирается контролировать сигналы на внутренних цепях, желательно присвоить им названия.

Чтобы сделать имя видимым на экране монитора, достаточно дважды щелкнуть на проводнике мышью, затем на открывшейся диалоговой панели Wire Properties выбрать закладку View Text и установить флажок NAME. Так же просто заменить системное имя пользовательским. Откройте диалоговую панель Wire Properties и введите в поле Segment желаемое имя.

На четвертом шаге необходимо выполнить электрический контроль схемы (команда **Check Diagram** в выпадающем меню **Diagram**) и компиляцию проекта (команда **Compile** в выпадающем меню **Design**). Система не только проинформирует вас о числе ошибок и предупреждений, но и покажет те места на схеме, где они обнаружены.

На пятом шаге графическое описание проекта должно быть конвертировано в текстовое описание (в формат языка VHDL). Выполняется данная процедура командой **Generate HDL Code**.

1.1.2. Задание диаграмм входных сигналов

Временные диаграммы внешних воздействий задаются в специальном окне редактора сигналов **Waveform Editor**, который вызывается по пиктограмме New Waveform на инструментальной панели Standard. При запуске редактор Waveform Editor открывает новую закладку поверх ранее построенной схемы, которая состоит из двух частей. Левая часть предназначена для ввода сигналов, правая – для показа временных диаграмм.

Необходимо щелкнем правой кнопкой мыши (ПКМ) в левой части окна для выбора в контекстном меню команды Add Signals. Она дублируется также на инструментальной.

Откроется одноименная панель со списком всех доступных сигналов. Нажатием кнопки **Add** осуществляется перенос выбранных сигналов в окно редактора Waveform Editor. Выделив один или несколько программируемых сигналов и щелчком на иконке Stimulators исполним одноименную команду **Stimulators**.

Откроется диалоговая панель Stimulators (см. рис. 1.1.5). В окне Signals выделим сигнал, в окне Type определим для него тип стимулятора **Clock** и в правом окне зададим частоту (Frequency) 10 МГц. Нажмем кнопку Apply, не закрывая панель, необходимо осуществить программирования всех входных сигналов. Кроме стимулятора Clock возможно задавать входные воздействия по средствам стимуляторов **Formula** (задание входных сигналов по указанной закономерности), **Value** (задание фиксированного значения сигнала на весь временной диапазон моделирования), **Hotkey** (переключение состояний входных сигналов с использованием «горячих клавиш»).

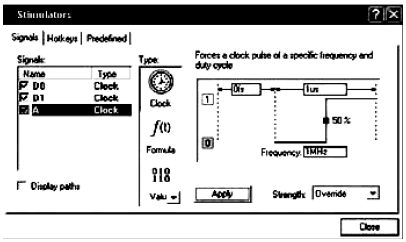

Рис. 1.1.5. Диалоговое окно стимулятора Clock

1.1.3. Моделирование проекта

Есть несколько способов запустить схему на моделирование (см. рис. 1.1.6). Одним из вариантов является пошаговое моделирование командой **Run For** из меню **Simulation**. Имитатор, получив такую команду, продвинет

моделирование на один шаг, величина которого задается в соседнем окне справа от кнопки.

По умолчанию шаг равен 100 ns, но его всегда можно поменять. Повторное исполнение этой команды продвинет время моделирования еще на один шаг и т. д. В паузе перед очередным шагом, возможно, изменить не только его длительность, но и перепрограммировать параметры входных сигналов. Пошаговое моделирование является достаточно эффективным при отладке схемы.

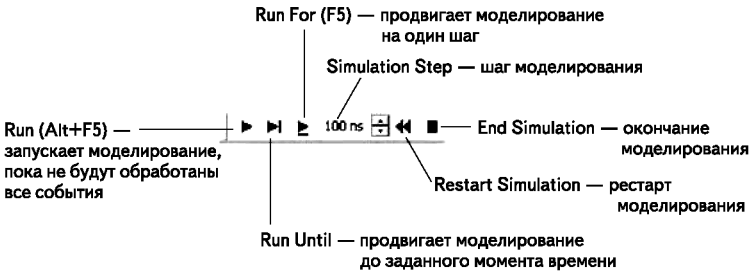

Рис. 1.1.6. Инструменты управления процессом моделирования

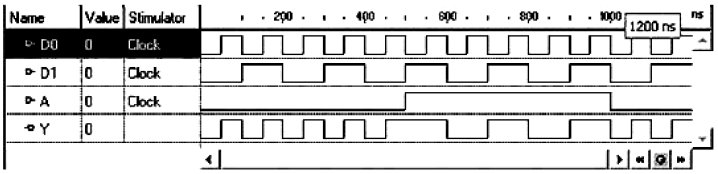

Рис. 1.1.7. Пример результатов моделирования

1.2. Проектирование цифровых устройств в среде Xilinx ISE

Интегрированная система автоматизированного проектирования Integrated Software Environment (САПР ISE) является основным продуктом сквозного проектирования цифровых систем на базе программируемых логических интегральных схем (ПЛИС) фирмы Xilinx. Данный программный продукт поддерживает все выпускаемые ПЛИС и имеет несколько вариантов поставки. В данном разделе представлены основные сведения о порядке работы и способе построения проектов в программном комплексе Xilinx ISE.

1.2.1. Основные этапы создания проекта Xilinx ISE. Программирование отладочной платы ПЛИС

Ознакомимся с основными принципами и методами проектирования с использованием программного комплекса Xilinx ISE для отладочных плат Digilent Basys 2. Необходимо построить проект позволяющий проверить работу базовых булевых операторов *«not, and, or, nand»*. Рассмотрим **основные этапы** создания проекта на примере данного задания.

Запуск среды проектирования Xilinx ISE. На появившемся экране нажать кнопку «New Project», в появившемся диалоге «New Project Wizard» ввести имя проекта «basic_boolean», нажать «Next» (см. рис. 1.2.1).

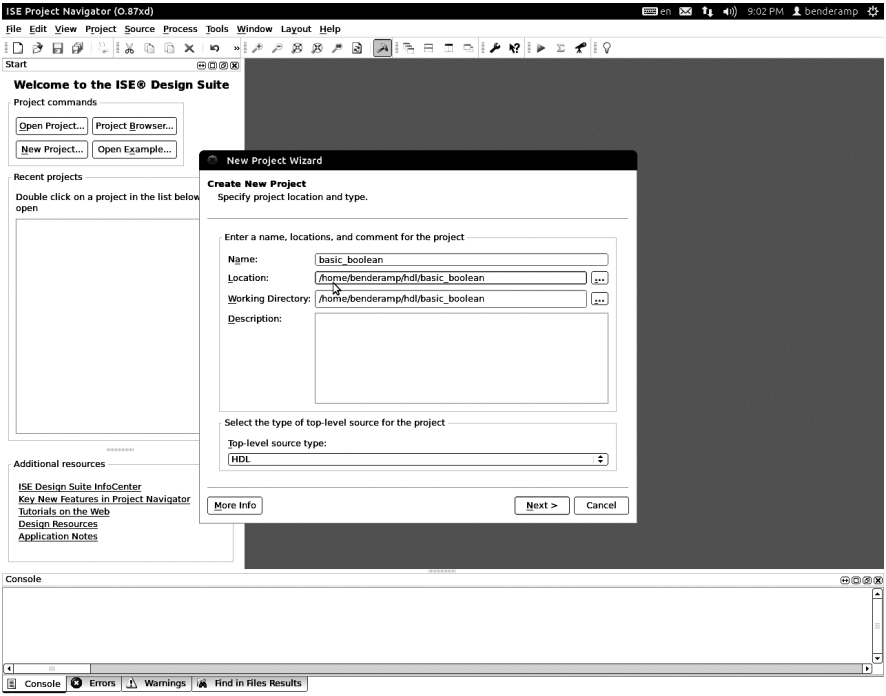

Рис. 1.2.1. Диалоговое окно редактора Xilinx ISE. Создание проекта

На следующем экране «Specify device and project properties» указать следующие настройки для отладочной платы ПЛИС Digilent Basys 2:

Product Category: All
Family: Spartain3E
Device: XC3S250E
Package: CP132
Speed: -5

Далее следует экран со сводкой по новому проекту – нажать «Finish».

1.2.2. Создание файла с исходным кодом на языке описания аппаратуры Verilog

Выбираем меню «Project > New Source...», в открывшемся диалоге «New Source Wizard» выбираем в списке тип файла исходного кода (Source Type) «Verilog Module», вводим имя создаваемого файла в поле «File name»: «basic_boolean», нажимаем кнопку «Next» (см. рис. 1.2.2).

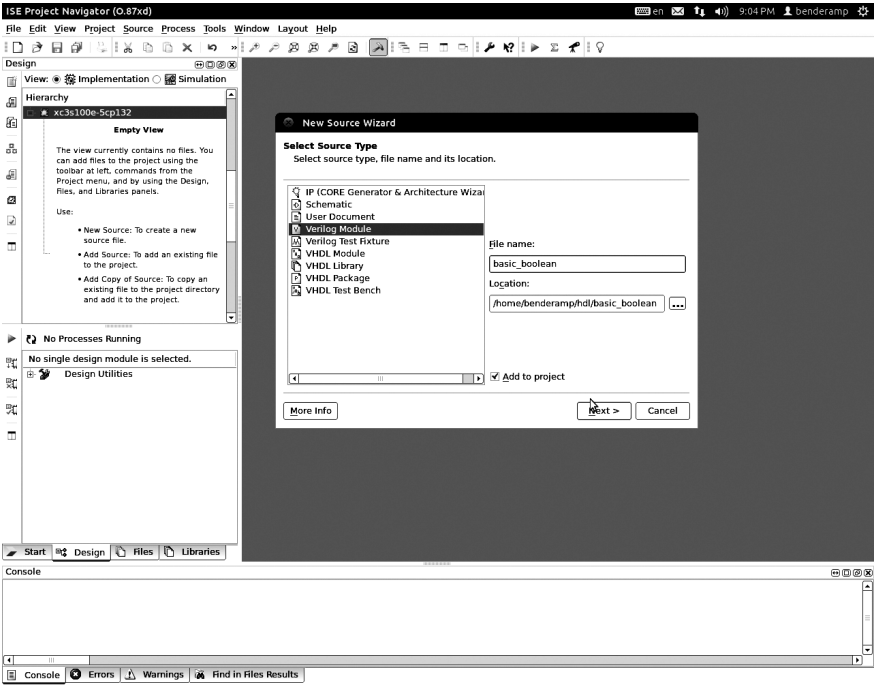

Рис. 1.2.2. Диалоговое окно редактора Xilinx ISE. Создание файла проекта с исходным кодом

Следующее диалоговое окно («Specify ports for module») осуществляет определение портов ввода и вывода для модуля, в данном случае определим их вручную, нажимаем «Next», следующая сводка информации по новому файлу – «Finish».

Получен новый файл с исходными кодами модуля на языке *Verilog*. Часть кода комментариев и директивы «timescale 1ns / 1ps» можно удалить, оставить только главное определение (пока пустого) модуля (см. рис. 1.2.3):

```
module basic_boolean(
);
endmodule.
```

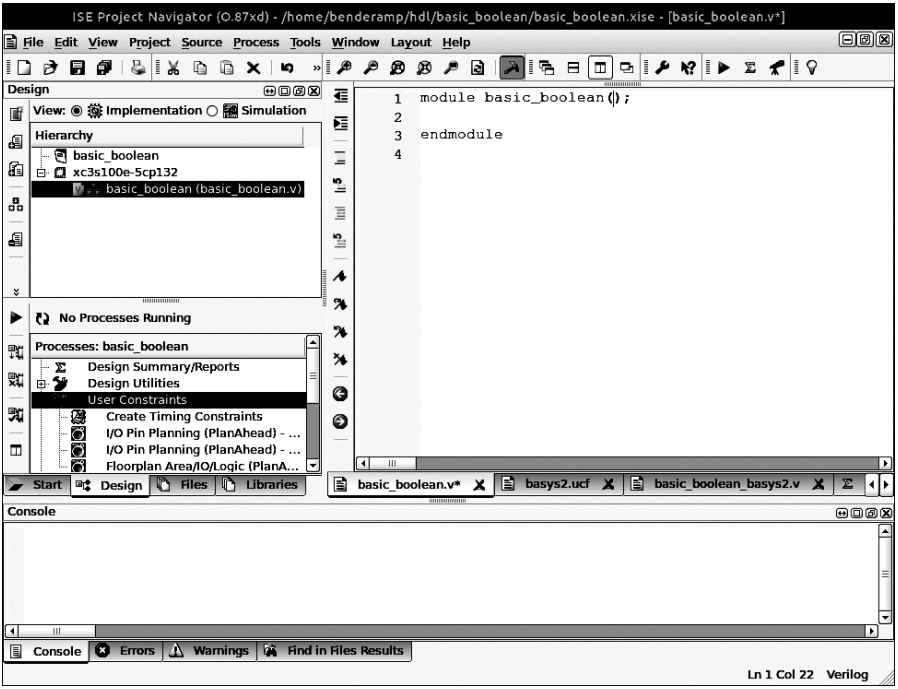

Рис. 1.2.3. Диалоговое окно редактора Xilinx ISE. Файл с исходным кодом

Модуль *Verilog* - это основная структурная единица построения цифровой системы. Определение любого модуля начинается с перечисления его входов (input) и выходов (output).

В модуле *basic_boolean* будет практически реализована работа базовых булевых операторов *NOT, AND, OR, NAND*, поэтому у него будет два входа и несколько выходов - по одному на каждый оператор:

- два входа (input): a и b
- пять выходов (output): not_a, not_b, a_and_b, a_or_b, a_nand_b.

В соотвествии с данным условием добавляем код:

```
module basic_boolean(input a, input b,
output not_a, output not_b, output a_and_b,
output a_or_b, output a_nand_b
);
endmodule
```

В теле модуля входами и выходами можно оперировать – выходам можно присваивать значения при помощи оператора assign. При этом значением может быть результат действия стандартных булевых операторов (или их комбинаций) над входами (или другими внутренними переменными или числовыми константами).

Каждый вход и выход в данном случае несет ровно 1 бит информации *(1=TRUE, 0=FALSE).*

В результате получим следующий код (см. рис. 1.2.4):

```
module basic_boolean(input a, input b,
output not_a, output not_b, output a_and_b,
output a_or_b, output a_nand_b
);
assign not_a = ~a; // NOT
assign not_b = ~b; // NOT
assign a_and_b = a & b; // AND
assign a_or_b = a | b; // OR
assign a_nand_b = ~(a & b); // NAND
endmodule
```

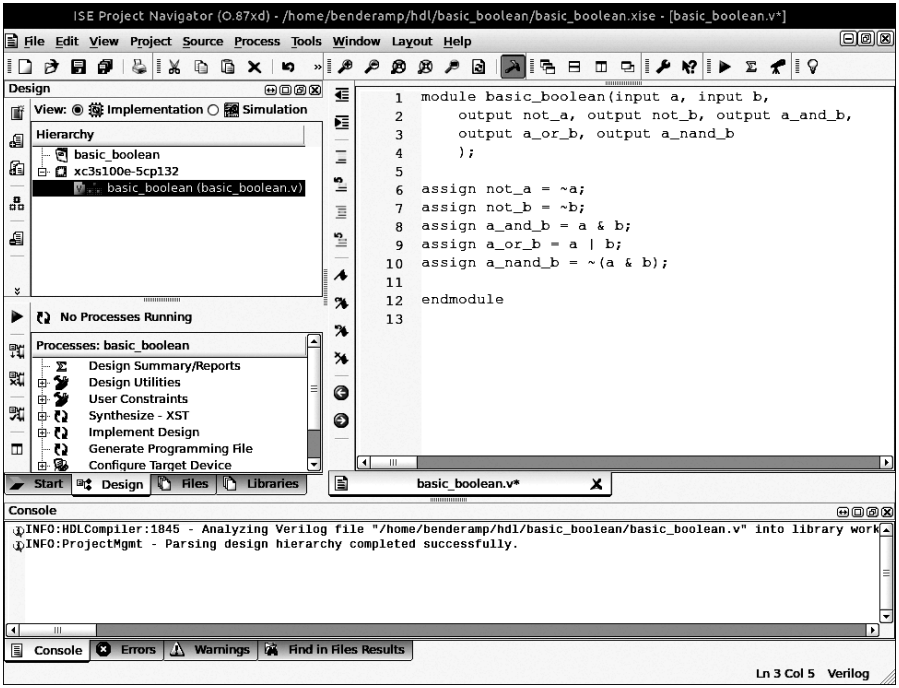

Рис. 1.2.4. Диалоговое окно редактора Xilinx ISE. Описание проекта на языке Verilog

1.2.3. Подключение файла конфигурации

Полученный программный код позволяет осуществить проверку указанных базовых булевых операторов. Данный проект, возможно, реализовать с использованием программных симуляторов или специализированной отладочной плате ПЛИС.

Перед синтезом прошивки для платы ПЛИС отметим, что любая плата ПЛИС представляет из себя «черный ящик», «внутри» которого находится прошивка, синтезированная из программы на языке *Verilog*, а «снаружи» физические устройства ввода/вывода (на отладочной плате Digilent Basys 2 установлены переключатели (*SW*), кнопки (*BTN*) управления

входными портами, светодиоды и дисплей для вывода информации и т.п.). Для того чтобы программа на языке *Verilog* получила возможность связаться с устройствами ввода/вывода определенной платы, нужен соответствующий механизм. Для этого в среде разработки Xilinx ISE используется инструмент специальных файлов конфигурации с расширением **.ucf* (файл *basys2.ucf*).

Все устройства ввода/вывода на плате пронумерованы уникальными идентификаторами. На Digilent Basys 2 эти идентификаторы указаны на самой плате (см. рис. 1.2.5) или же доступны в документации на сайте производителя продукта.

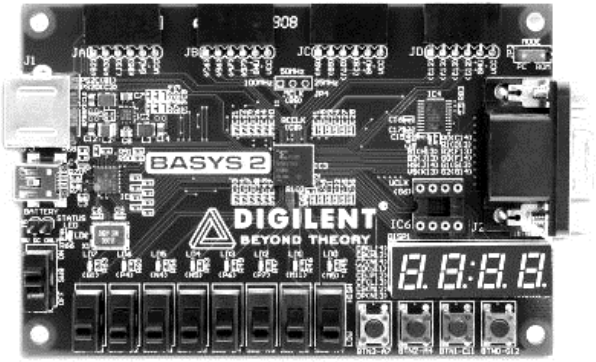

Рис. 1.2.5. Отладочная плата Digilent Basys 2

Файл *basys2.ucf* нужно подключить к проекту, для этого необходимо выбрать меню «Project > Add Copy of Source...» и найти в файловой системе.

Файл должен появиться в дереве проекта, после этого его нужно открыть в редакторе и оставить только те строки, которые соответствуют используемым в проекте устройствам ввода/вывода, а строки для остальных устройств нужно или удалить или закомментировать (символ «#» в начале строки), т.к. иначе система выдаст ошибку при синтезе прошивки (см. рис. 1.2.6).

```
#basys.ucf
NET "ld<4>" LOC = "n5" ;
NET "ld<3>" LOC = "p6" ;
NET "ld<2>" LOC = "p7" ;
NET "ld<1>" LOC = "m11" ;
NET "ld<0>" LOC = "m5" ;
NET "sw<1>" LOC = "l3";
NET "sw<0>" LOC = "p11";
```

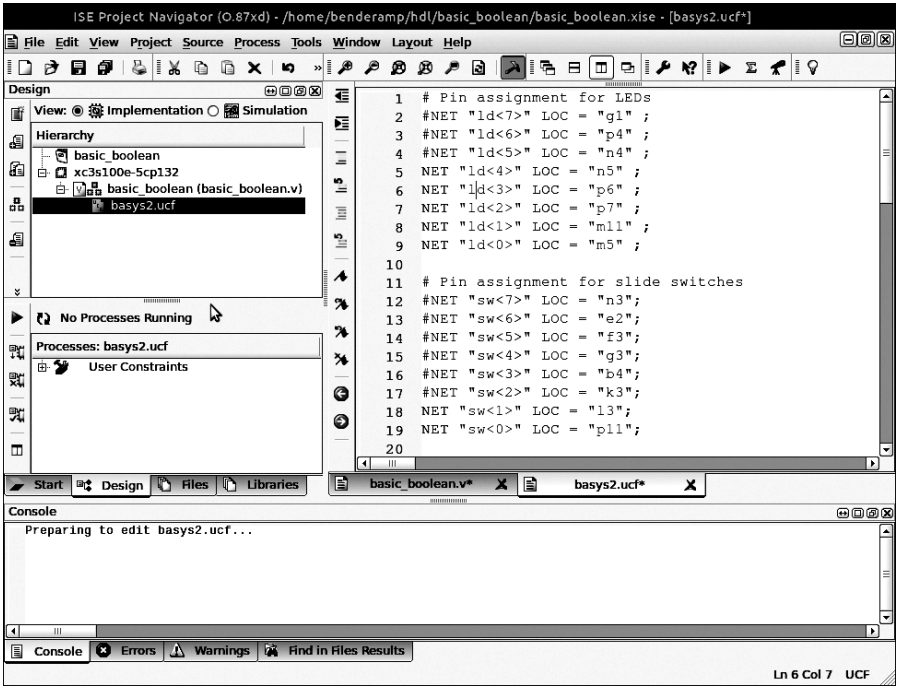

Рис. 1.2.6. Диалоговое окно редактора Xilinx ISE. Структура файла конфигурации

1.2.4. Создание модуля верхнего уровня для связи с устройствами ввода/вывода платы ПЛИС

Как уже было сказано выше, основная структурная единица дизайна на языке *Verilog* - это модуль с набором входов и выходов. В рамках дизайна можно "размещать" модули на одном уровне друг рядом с другом и подключать выходы одного модуля к входам другого. А еще можно поместить несколько модулей внутрь другого модуля более высокого уровня - тогда входы внешнего модуля будут подключены к входам внутренних модулей, а выходы внутренних модулей будут подключены к выходам внешнего модуля.

При данном вложенном размещении модулей должен быть один главный модуль самого верхнего уровня (top module), который будет содержать все остальные модули дизайна вложенные один в другой. При синтезе проекта для ПЛИС в качестве такого модуля верхнего уровня (top module) выбирается модуль, входами и выходами которого являются реальные устройства ввода/вывода, доступные на плате, а внутри этого модуля размещены другие логические модули проекта (в нашем случае один модуль - *basic_boolean*).

Создаем новый модуль *Verilog* как было показано выше («Project > New Source... > Verilog Module») *basic_boolean_basys2*. Определяем входы и выходы с теми именами, которые указаны в файле *basys2.ucf* и направляем их в модуль *basic_boolean*. Получаем следующий код (см. рис. 1.2.7):

```
module basic_boolean_basys2(
input [0:1] sw,
output [0:4] ld);
basic_boolean impl(.a(sw[0]), .b(sw[1]),
.not_a(ld[0]), .not_b(ld[1]),
.a_and_b(ld[2]), .a_or_b(ld[3]),
.a_nand_b(ld[4]));
```

endmodule
basic_boolean - имя модуля

impl - указание внутреннего модуля внутри текущего модуля (может быть использован модуль на языке *Verilog* или *VHDL*), в скобках указывается ассоциирование параметров текущего модуля с параметрами внутреннего модуля.

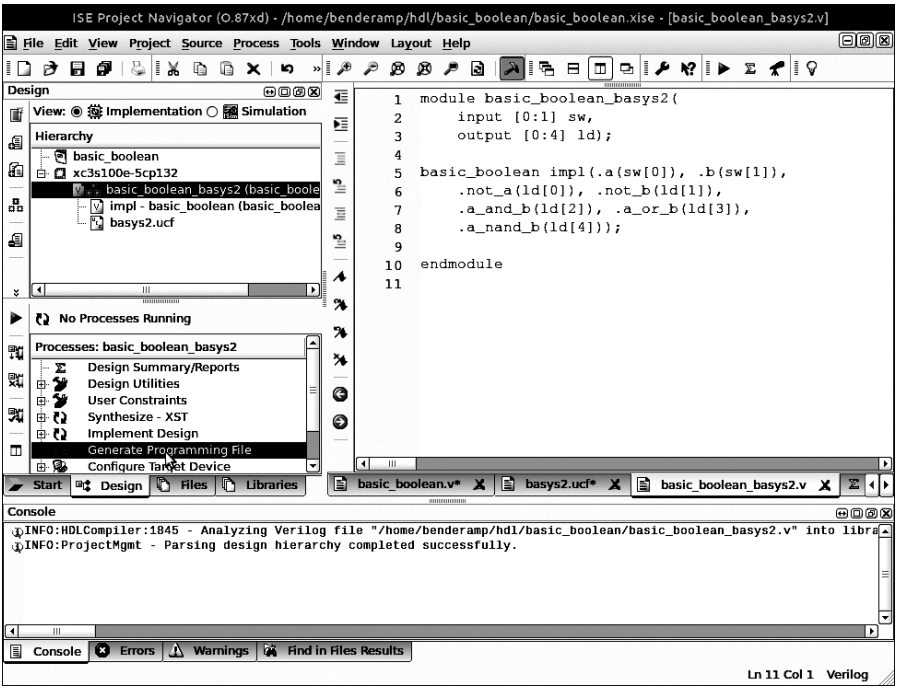

Рис. 1.2.7. Диалоговое окно редактора Xilinx ISE. Проект на языке Verilog (top module)

В результате получаем, что значения переключателей отладочной платы *sw[0]* и *sw[1] (0=FALSE или 1=TRUE)* подаются в качестве значений входных параметров модуля *basic_boolean* a и b, а светодиоды *ld[0]...ld[4]* будут показывать значения результатов работы модуля

basic_boolean not_a, not_b, a_and_b, a_or_b, a_nand_b (0=FALSE=выкл или 1=TRUE=вкл).

1.2.5. Синтез прошивки

Для синтеза прошивки ПЛИС необходимо в списке действий под деревом проекта выбрать элемент Generate Programming File, если синтез прошел без ошибок программный комплекс формирует *bit*-файл (*basic_boolean_basys2.bit*) с прошивкой (см. рис. 1.2.8).

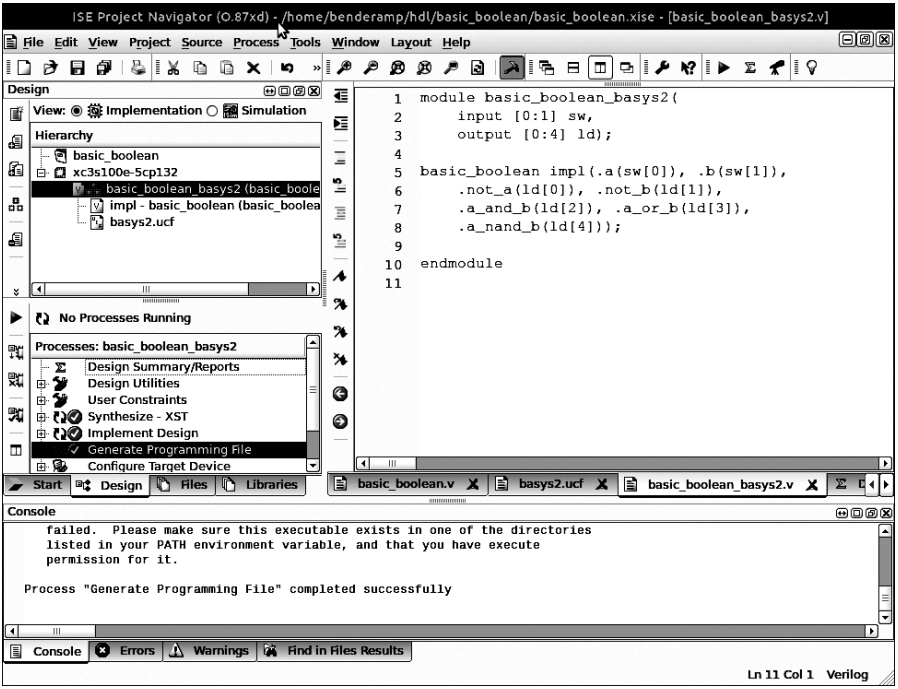

Рис. 1.2.8. Диалоговое окно редактора Xilinx ISE. Синтез прошивки

1.2.6. Программирование платы ПЛИС Digilent Basys 2

Осуществляется по средствам программы Adept компании Digilent, поставляется в комплекте с отладочной платой Digilent Basys 2 (см. рис. 1.2.9). Данная отладочная плата включает в себя две программируемые микросхемы FPGA XC3S250E и PROM XCF02S.

Перед программированием необходимо подключить отладочную плату к ПК, переключить плату в режим «PC», осуществить удаление предыдущей прошивки и произвести программирование. В режиме «ROM» производиться непосредственная работа установленной прошивки (в случае программирования PROM).

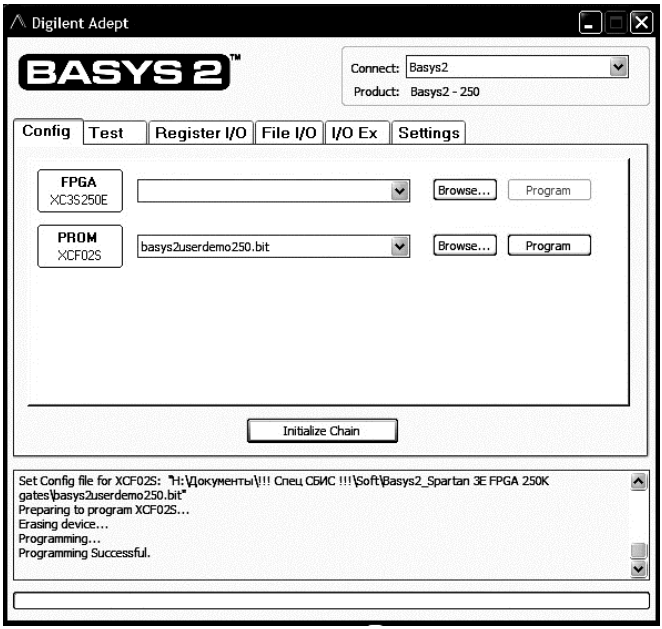

Рис. 1.2.9. Диалоговое окно программы Adept

Подача входных сигналов на данной плате осуществляется при помощи набора переключателей, обозначенных *SW0 – SW7*, четырех не фиксируемых кнопок *BTN0 – BTN3*, а также есть возможность задания входных импульсов с клавиатуры, на плате для этого предусмотрен разъем PS/2. Выходные сигналы поступают на светодиоды *LD0 – LD7*, светодиодную панель или на монитор через разъем *VGA*.

Для проверки работоспособности перед началом программирования необходимо выполнить установку тестовой прошивки для проверки всех органов управления платой (файл *basys2userdemo250.bit*) или использовать вложенную утилиту «Test» программы Adept.

2. ЛАБОРАТОРНЫЕ РАБОТЫ

2.1. Лабораторная работа № 1. Проектирование функциональных узлов комбинационного типа

2.1.1. Общие указания по выполнению лабораторной работы

Целью лабораторной работы является углубление и закрепление знаний студентов в области основных понятий цифровой схемотехники, моделей и параметров логических элементов, а также получение навыков в проектировании программируемых логических интегральных схем (ПЛИС) с использованием систем автоматизированного проектирования (САПР) на языке VHDL.

Лабораторный практикум посвящен проектированию структур функциональных узлов комбинационного и последовательностного типа. В рамках данной лабораторной работы необходимо изучить структуру и основные принципы построения проекта на языке описания аппаратных средств VHDL с использованием программного комплекса Active-HDL, построить модели простейших мультиплексоров и дешифраторов.

Перед выполнением лабораторной работы студент должен самостоятельно выполнить домашнее задание в соответствии с данными методическими указаниями.

Студент, явившийся на занятия, должен иметь методические указания по данной лабораторной работе. В начале занятия преподаватель проверяет выполнение студентом домашнего задания и наличие заготовки отчета по данной лабораторной работе в его рабочей тетради.

К выполненной работе прилагаются необходимые схемы, эскизы, тексты и результаты проектирования, протоколы работы с программным комплексом и другие материалы согласно указаниям по оформлению отчета.

2.1.2. Домашние задания и указания по их выполнению

Задание 1. Ознакомиться с моделями простейших логических элементов, системами их параметров. Изучить основные узлы комбинационного типа и их назначение. Для этого необходимо воспользоваться лекциями по курсу и литературой [1, С. 7-22].

Задание 2. Ознакомиться с основными принципами и методами построения моделей функциональных узлов на логическом уровне. При выполнении домашнего задания студент должен ознакомиться с основными сведениями о языке VHDL и способами описания проекта. Для этого необходимо воспользоваться лекциями по курсу и литературой [1, С. 81-91, С. 637-642, С. 673-680].

2.1.3. Вопросы к домашнему заданию

1. Дать определение идеализованного логического элемента.
2. Перечислить основные логические элементы, представить таблицы истинности.
3. Перечислить основные функциональные узлы комбинационного типа цифровой схемотехники, дать краткое описание.
4. Дать определение двоичного дешифратора, мультиплексора и демультиплексора, представить таблицы состояний, пояснить принцип работы.
5. Дать определение компаратора кода, сумматора и умножителя, представить таблицы состояний, пояснить принцип работы.

2.1.4. Лабораторные задания и указания по их выполнению

Задание 1. Построить и отладить модели мультиплексоров на логическом уровне, опираясь на пример приведенный в домашнем задании, схемы мультиплексоров представлены на рис. 2.1.1 и 2.1.2. Для каждой схемы необходимо построить временные диаграммы, демонстрирующие их работу.

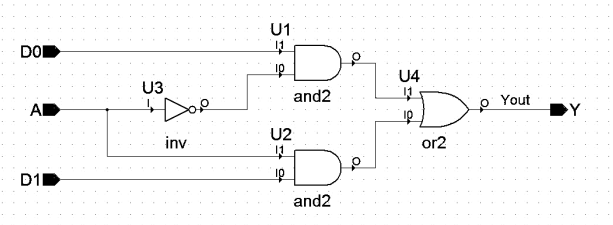

Рис. 2.1.1. Схема мультиплексора «2-1»

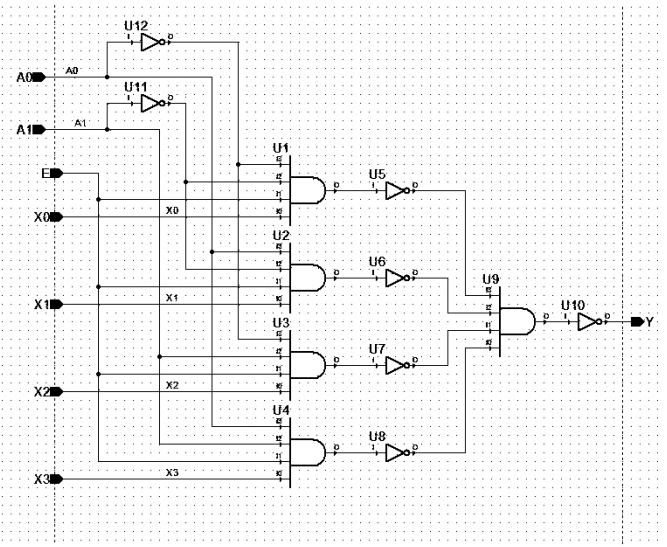

Рис. 2.1.2. Схема мультиплексора «4-1»

Задание 2. Построить и отладить модели дешифраторов на логическом уровне, опираясь на пример приведенный в домашнем задании, схемы дешифраторов представлены на рис. 2.1.3, 2.1.4 и 2.1.5. Для каждой схемы необходимо построить временные диаграммы, демонстрирующие их работу и соответствующие их таблицам состояний.

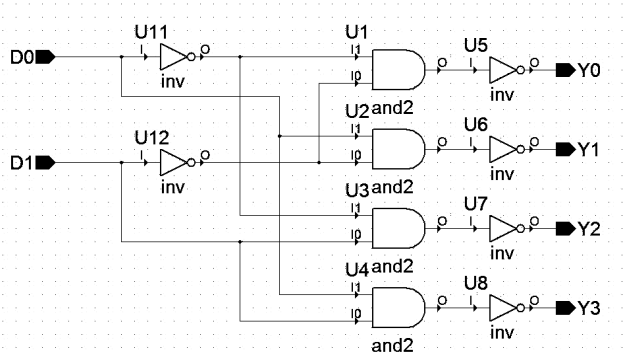

Рис. 2.1.3. Схема дешифратора «2-4» с инверсными выходами

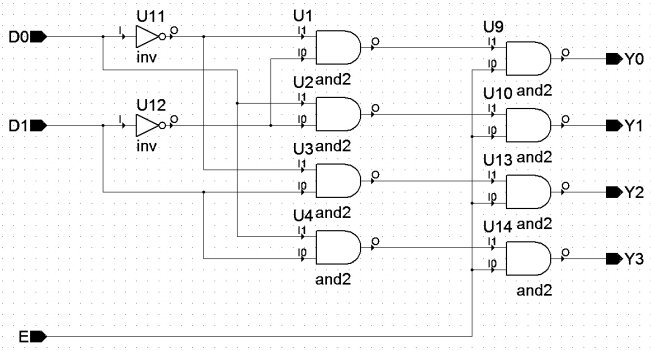

Рис. 2.1.4. Схема дешифратора «2-4» с использованием разрешающего сигнала «E»

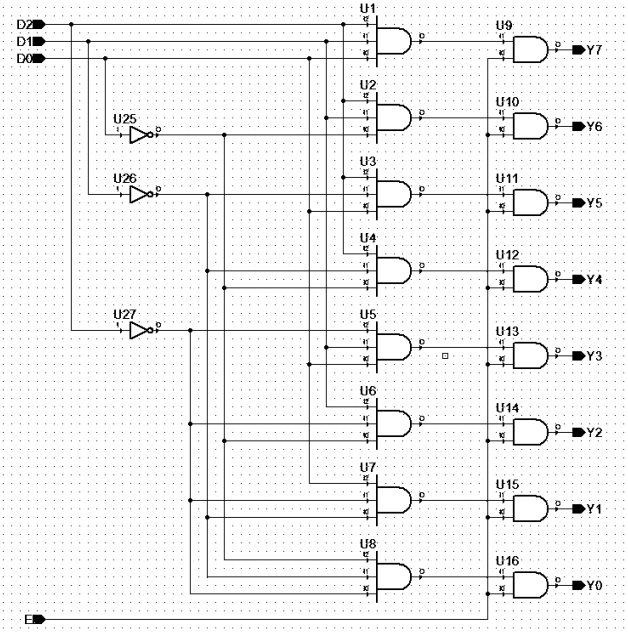

Рис. 2.1.5. Схема дешифратора «3-8» с использованием разрешающего сигнала «E»

Задание 3. Используя материал задания 2 и модель дешифратора «3-8» на языке описания аппаратуры VHDL, полученной на лабораторном практикуме, построить проект данного устройства в среде Xilinx ISE. В качестве внутреннего модуля в проекте использовать код VHDL, сгенерированный в результате моделирования в Active HDL. Подачу разрешающего импульса осуществить при помощи фиксируемого переключателя платы *sw*, задание входной последовательно по средствам кнопок *btn*, для отображения выходных сигналов использовать светодиоды *Ld*. Пример кода представлен ниже:

module DC_38E(input [0:2] btn,
input [0:0] sw,
output [0:7] Ld);

DC38EN impl(.D0(btn [0]), .D1(btn [1]),
.D2(btn [2]), .E(sw [0]),
.Y0(Ld[0]), .Y1(Ld[1]), .Y2(Ld[2]), .Y3(Ld[3]),
.Y4(Ld[4]), .Y5(Ld[5]), .Y6(Ld[6]), .Y7(Ld[7]));

endmodule

DC38EN – имя файла с расширением *.vhd. RTL код проектируемого дешифратора.

Параметры файла конфигурации *basys.ucf*:
NET "ld<7>" LOC = "g1" ;
NET "ld<6>" LOC = "p4" ;
NET "ld<5>" LOC = "n4" ;
NET "ld<4>" LOC = "n5" ;
NET "ld<3>" LOC = "p6" ;
NET "ld<2>" LOC = "p7" ;
NET "ld<1>" LOC = "m11" ;
NET "ld<0>" LOC = "m5" ;
NET "sw<0>" LOC = "p11";
NET "btn<2>" LOC = "m4";
NET "btn<1>" LOC = "c11";
NET "btn<0>" LOC = "g12";

Задание 4. Используя материал задания 2 и модель дешифратора «2-4» на языке описания аппаратуры VHDL, полученной на лабораторном практикуме, построить проект данного устройства в среде Xilinx ISE. В качестве внутреннего модуля в проекте использовать код VHDL, сгенерированный в результате моделирования в Active HDL. Подачу разрешающего импульса осуществить при помощи встроенного тактового генератора *clk* на 25МГц, установку входной последовательно по средствам переключателей *sw*, для отображения выходных сигналов использовать светодиоды *Ld*. Пример кода представлен ниже:

module DC_24E(input [0:1] sw,
input clk,
output [0:3] ld);

DC24EN impl(.D0(sw [0]), .D1(sw [1]), .E(clk),
.Y0(ld[0]), .Y1(ld[1]),
.Y2(ld[2]), .Y3(ld[3]));

endmodule

DC24EN – имя файла с расширением *.vhd. RTL код проектируемого дешифратора.

Параметры файла конфигурации *basys.ucf*:
NET "ld<3>" LOC = "p6" ;
NET "ld<2>" LOC = "p7" ;
NET "ld<1>" LOC = "m11" ;
NET "ld<0>" LOC = "m5" ;
NET "clk" LOC = "b8";
NET "sw<1>" LOC = "l3";
NET "sw<0>" LOC = "p11";

2.1.5. Контрольные вопросы

1. Какова цель лабораторной работы?
2. В чем заключается лабораторное задание? Пояснить ход его выполнения.
3. Какие данные являлись исходными для выполнения работы?
4. Пояснить работу полученных моделей функциональных узлов комбинационного типа.
5. Сформулируйте выводы по данной лабораторной работе.

2.2. Лабораторная работа № 2. Проектирование иерархических модулей

2.2.1. Общие указания по выполнению лабораторной работы

Целью лабораторной работы является углубление и закрепление знаний студентов в области функциональных узлов комбинационного типа цифровой схемотехники. В рамках данной лабораторной работы необходимо построить на логическом уровне иерархическую модель дешифратора адреса м мультиплексора с использованием программного комплекса Active-HDL на языке описания аппаратуры VHDL.

Перед выполнением лабораторной работы студент должен самостоятельно выполнить домашнее задание в соответствии с данными методическими указаниями.

Студент, явившийся на занятия, должен иметь методические указания по данной лабораторной работе. В начале занятия преподаватель проверяет выполнение студентом домашнего задания и наличие заготовки отчета по данной лабораторной работе в его рабочей тетради.

К выполненной работе прилагаются необходимые схемы, эскизы, тексты и результаты проектирования, протоколы работы с программным комплексом и другие материалы согласно указаниям по оформлению отчета.

2.2.2. Домашние задания и указания по их выполнению

Задание 1. Ознакомиться со структурой и принципом работы основных узлов комбинационного типа (двоичные дешифраторы, мультиплексоры и демультиплексоры, компараторы и сумматоры, матричные умножители). Для этого необходимо воспользоваться лекциями по курсу и литературой [1, С. 73-95, С. 104-122].

Задание 2. Изучить методы построения иерархических блоки (ИБ) *FUB (Functional user block)*. ИБ применяются при

проектировании схем в программном комплексе Active-HDL для тех же целей, что и символы, но в отличие от последних они имеют несколько специфических свойств. Есть возможность создавать и редактировать ИБ непосредственно на схеме, сохранять иерархический блок в библиотеке элементов, что позволяет уже построенный и отлаженный блок использовать при создании новых проектов. Такой подход к проектированию логических схем в значительной мере снижает трудоемкость и время разработки.

2.2.3. Вопросы к домашнему заданию

1. Пояснить схемотехническую реализацию дешифраторов.
2. Пояснить схемотехническую реализацию мультиплексоров.
3. Как осуществляется наращивание размерности дешифраторов?
4. Как осуществляется наращивание размерности мультиплексоров?

2.2.4. Лабораторные задания и указания по их выполнению

Задание 1. Построить и отладить иерархическую модель дешифратора «5-32» на логическом уровне с использованием программного комплекса Active-HDL на языке описания аппаратуры VHDL. Обозначение и схема дешифратора представлена на рис. 2.2.1 и 2.2.2 соответственно. Для данной схемы необходимо построить временную диаграмму, демонстрирующую работу, а также полученную модель необходимо сохранить в виде стандартного элемента библиотеки.

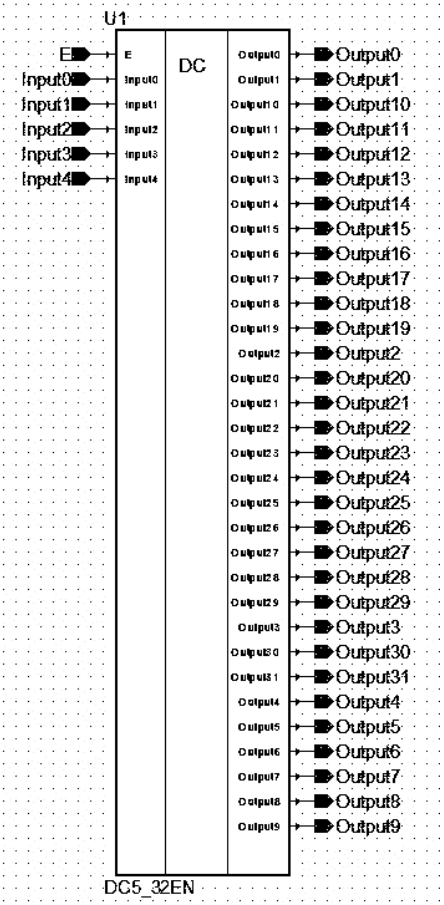

Рис. 2.2.1. Обозначение дешифратора «5-32»

Иерархическая модель дешифратора «5-32» стоится на основе, полученных в первой лабораторной работе моделей простейших дешифраторов «2-4» и «3-8». Данная иерархическая модель должна также содержать разрешающий сигнал «E».

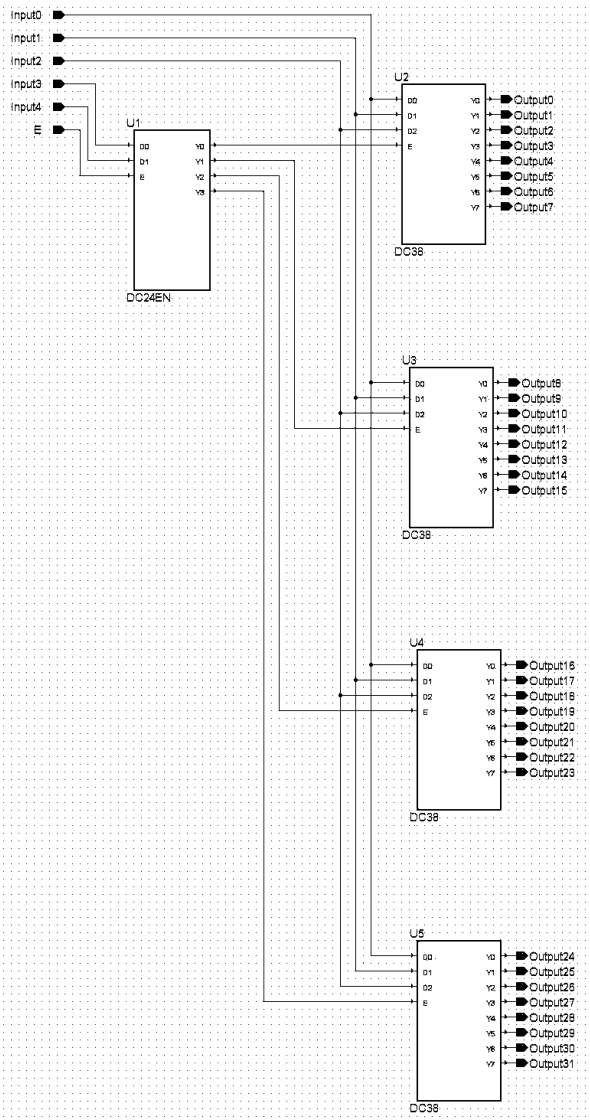

Рис. 2.2.2. Схема иерархической модели дешифратора «5-32»

Задание 2. Построить и отладить иерархическую модель мультиплексора «16-1» на логическом уровне с использованием программного комплекса Active-HDL на языке описания аппаратуры VHDL. Обозначение и схема мультиплексора представлена на рис. 2.2.3 и 2.2.4 соответственно. Для данной схемы необходимо построить временную диаграмму, демонстрирующую работу, а также полученную модель необходимо сохранить в виде стандартного элемента библиотеки.

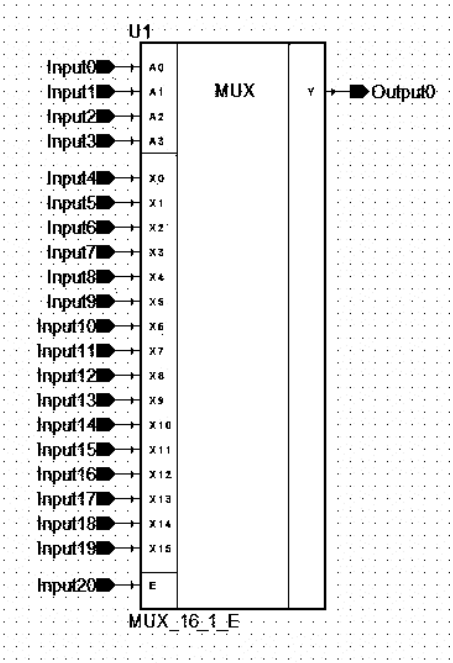

Рис. 2.2.3. Обозначение мультиплексора «16-1»

Иерархическая модель мультиплексора «16-1» стоится на основе, полученной в первой лабораторной работе модели мультиплексора «4-1». Данная иерархическая модель должна также содержать разрешающий сигнал «E».

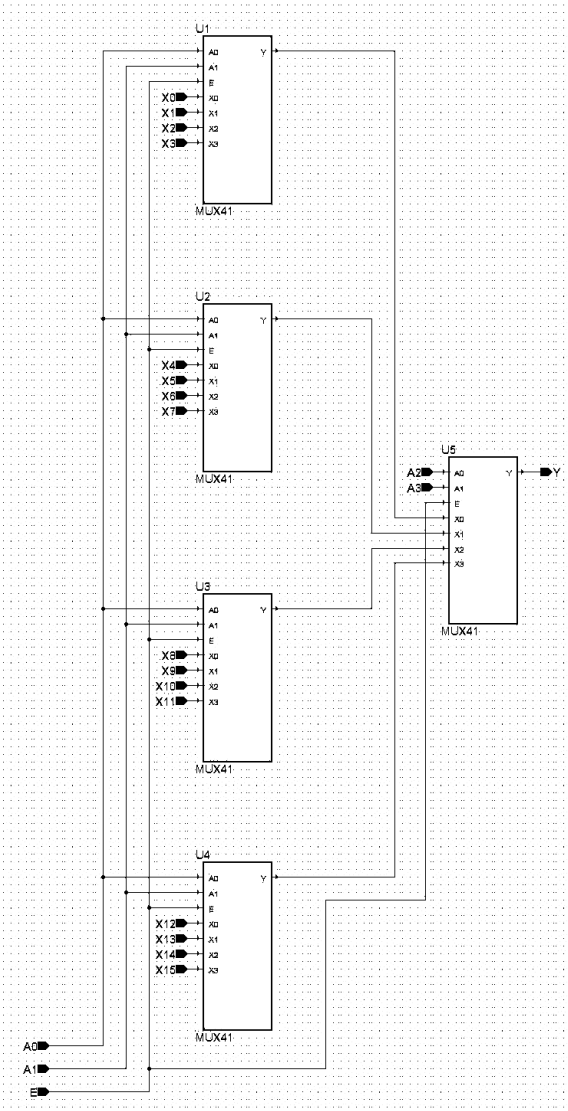

Рис. 2.2.4. Схема иерархической модели мультиплексора «16-1»

2.2.5. Контрольные вопросы

1. Какова цель лабораторной работы?
2. В чем заключается лабораторное задание? Пояснить ход его выполнения.
3. Какие данные являлись исходными для выполнения работы?
4. Пояснить работу полученной иерархической модели дешифратора.
5. Пояснить работу полученной иерархической модели мультиплексора.
6. Сформулируйте выводы по данной лабораторной работе.

2.3. Лабораторная работа № 3. Проектирование компаратора кода с использованием иерархических модулей и многоразрядных шин данных

2.3.1. Общие указания по выполнению лабораторной работы

Целью лабораторной работы является углубление и закрепление знаний студентов в области работы, структуры и систем параметров функциональных узлов комбинационного типа, а также получение навыков в проектировании компараторов кода с использованием языка VHDL. В рамках данной лабораторной работы необходимо построить модели компаратора для двух и четырех разрядных слов с использованием иерархических блоков и многоразрядных шин данных на языке описания аппаратных средств VHDL с применением программного комплекса Active-HDL.

Перед выполнением лабораторной работы студент должен самостоятельно выполнить домашнее задание в соответствии с данными методическими указаниями.

Студент, явившийся на занятия, должен иметь методические указания по данной лабораторной работе. В начале занятия преподаватель проверяет выполнение студентом домашнего задания и наличие заготовки отчета по данной лабораторной работе в его рабочей тетради.

К выполненной работе прилагаются необходимые схемы, эскизы, тексты и результаты проектирования, протоколы работы с программным комплексом и другие материалы согласно указаниям по оформлению отчета.

2.3.2. Домашние задания и указания по их выполнению

Задание 1. Ознакомиться с принципом работы, назначением и схемными конструкциями компараторов. Для этого необходимо воспользоваться лекциями по курсу и литературой [1, С. 100-103].

Компараторы кодов применяются для сравнения двух входных кодов и выдачи на выходы сигналов о результатах этого сравнения (о равенстве или неравенстве кодов). Компараторы (устройства сравнения) определяют отношения между двумя словами. Основными отношениями, через которые можно выразить остальные, можно считать два – «равно» и «больше».

Если используется одиночная микросхема, то для ее правильной работы достаточно подать единицу на вход $A = B$, а состояния входов $A < B$ и $A > B$ не важны, на них можно подать как нуль, так и единицу. Назначение выходов понятно из их названия, а полярность выходных сигналов положительная (активный уровень – единица). Если микросхемы компараторов кодов каскадируются (объединяются) для увеличения числа разрядов сравниваемых кодов, то надо выходные сигналы микросхемы, обрабатывающей младшие разряды кода, подать на одноименные входы микросхемы, обрабатывающей старшие разряды кода (см. рис. 2.3.1).

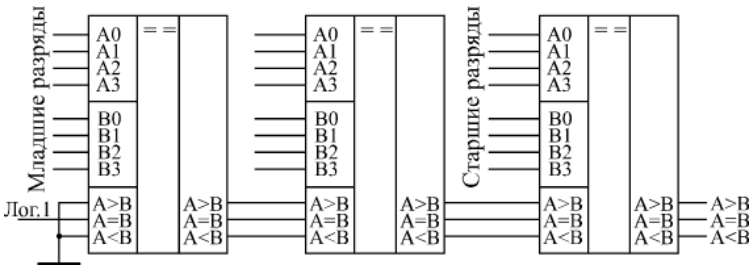

Рис. 2.3.1. Каскадирование компараторов кодов

Задание 2. Ознакомиться с методами трассировки в программном комплексе Active-HDL, с построением проводников и многоразрядных шин данных.

Способы построения проводников. Проводники в пакете Active-HDL 7.1 создаются двумя способами. Первый способ считается классическим, так как реализован во многих

современных САПР. Он носит название «метод последовательных точек». Суть метода в том, что проектировщик щелкает ЛКМ в точках, где начинается проводник, где он меняет свое направление и где заканчивается. В процессе построения создается временная трасса, и есть возможность, двигаясь в обратном направлении по уже проложенному пути, удалять только что созданные сегменты, а для его завершения потребуется двойной щелчок мышью.

Второй метод можно назвать методом «прижатой кнопки»: в процессе построения проводника левая кнопка мыши остается все время в нажатом состоянии. Прижимается ЛКМ в точке, где необходимо начать проводник и, не отпуская ее, ведете указатель мыши в точку назначения. При смене направления, в месте, где надо сделать угол (излом), следует нажать клавишу Space (пробел). Закончив создание проводника, просто отпустите кнопку мыши.

Проектирование шин. Шины размещаются на схеме по тем же правилам, что и проводники. Им требуется задавать кроме имени еще и ширину шины. Например, по умолчанию шины именуются как *BUS3(7:0)*. В круглых скобках указывается, что шина содержит восемь проводников, членов шины. Они имеют имена *BUS3(7)… BUS3(0)*. Не забывайте, что левая граница диапазона индексов, задающих ширину шины, всегда соответствует наиболее значащему разряду. В нашем примере старшим разрядом шины будет *BUS3(7)*.

К шине нельзя подключать проводники, не являющиеся членами шины. Для назначения имен шинам используются те же самые инструменты, что и для проводников. Поэтому мы и здесь обойдемся без особых комментариев.

2.3.3. Вопросы к домашнему заданию

1. Дать определение компаратора кода.
2. Пояснить принцип каскадного построения компараторов кода.

3. Пояснить способы построения проводников и многоразрядных шин данных.

2.3.4. Лабораторные задания и указания по их выполнению

Задание 1. Построить и отладить модель компаратора для двухразрядных слов. Схема данного компаратора представлена на рисунке ниже.

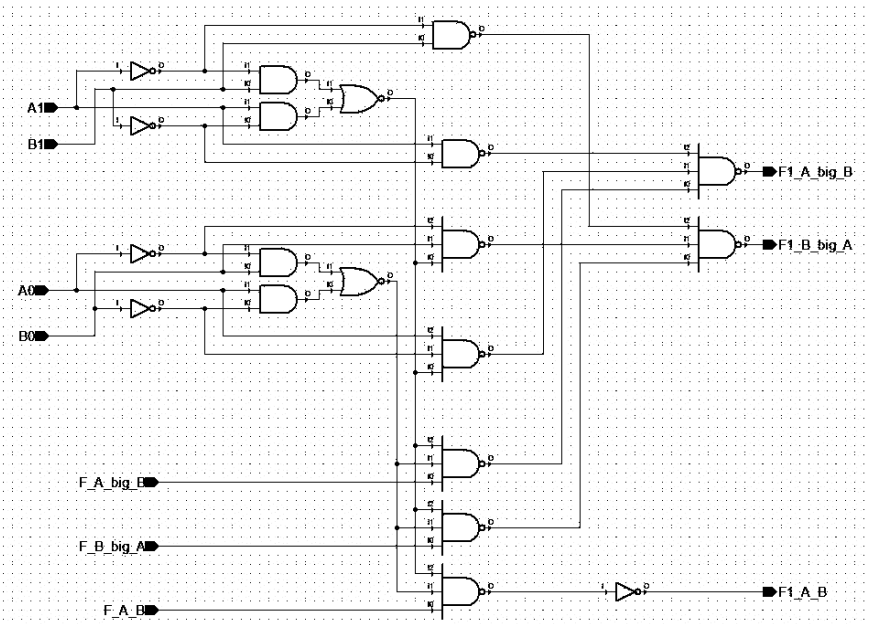

Рис. 2.3.2. Схема компаратора кодов для двухразрядных слов

Задание 2. Построить и отладить иерархическую модель компаратора кода для четырехразрядных слов, опираясь на пример приведенный в домашнем задании, с использованием программного комплекса Active-HDL на языке описания аппаратуры VHDL.

Схема компаратора представлена на рис. 2.3.3. Для данной схемы необходимо построить временную диаграмму, демонстрирующую ее работу.

Данная иерархическая модель компаратора строится на основе, полученной в первом задании модели компаратора для двухразрядных слов. Сравниваемые слова поразрядно объединить шинами данных *A(3:0)* и *B(3:0)*.

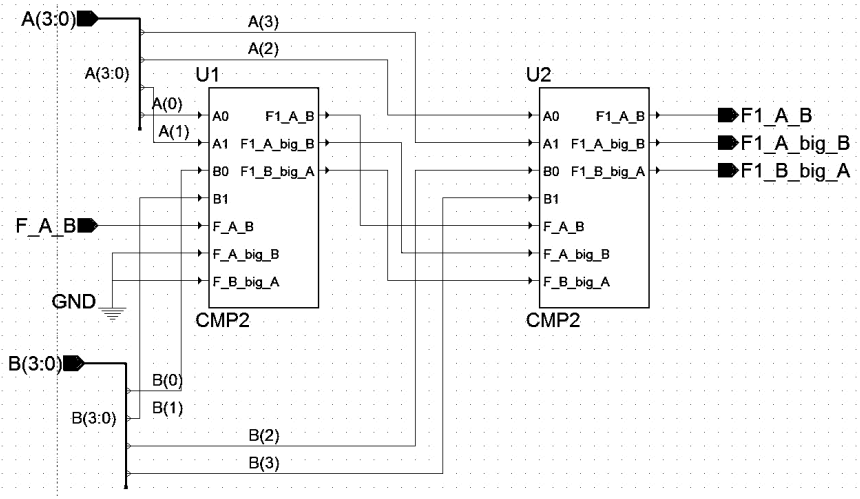

Рис. 2.3.3. Схема каскадной иерархической модели компаратора кодов для четырехразрядных слов

Задание 3. Проектирование модуля компаратора кода CMP в среде Xilinx. Используя материал задания 1 и модель компаратора на языке описания аппаратуры VHDL, полученной на лабораторном практикуме, построить проект данного устройства в среде Xilinx ISE. В качестве внутреннего модуля в проекте использовать код VHDL, сгенерированный в результате моделирования в Active HDL. Подачу управляющих импульсов (A=B, A<B, A>B) осуществить при помощи фиксируемых переключателей платы *sw*, задание входной последовательно по средствам кнопок *btn*, для

отображения результата сравнения слов использовать светодиоды *Ld*. Пример кода представлен ниже:

```
module cmp(input [0:3] btn,
input [0:2] sw,
output [0:2] Ld);

cmp2 impl(.A0(btn [0]), .A1(btn [1]),
.B0(btn [2]), .B1(btn [3]),
.F_A_B(sw[0]), .F_A_big_B(sw[1]),
.F_B_big_A(sw[2]), .F1_A_B(Ld[0]),
.F1_A_big_B(Ld[1]), .F1_B_big_A(Ld[2]));

endmodule
```

CMP2 – имя файла с расширением *.vhd. RTL код проектируемого компаратора кода.

Параметры файла конфигурации *basys2.ucf*:

```
NET "ld<2>" LOC = "p7" ;
NET "ld<1>" LOC = "m11" ;
NET "ld<0>" LOC = "m5" ;
NET "sw<2>" LOC = "k3";
NET "sw<1>" LOC = "l3";
NET "sw<0>" LOC = "p11";
NET "btn<3>" LOC = "a7";
NET "btn<2>" LOC = "m4";
NET "btn<1>" LOC = "c11";
NET "btn<0>" LOC = "g12";
```

2.3.5. Контрольные вопросы

1. Какова цель лабораторной работы?
2. В чем заключается лабораторное задание? Пояснить ход его выполнения.
3. Какие данные являлись исходными для выполнения работы?
4. Пояснить работу полученной модели компаратора.
5. Сформулируйте выводы по данной лабораторной работе.

2.4. Лабораторная работа № 4. Проектирование двоичного сумматора с использованием иерархических модулей

2.4.1. Общие указания по выполнению лабораторной работы

Целью лабораторной работы является углубление и закрепление знаний студентов в области работы, структуры и систем параметров функциональных узлов комбинационного типа, а также получение навыков в проектировании двоичных сумматоров с использованием языка VHDL. В рамках данной лабораторной работы необходимо построить модели одноразрядного и четырехразрядного сумматоров с использованием иерархических блоков на языке описания аппаратных средств VHDL с применением программного комплекса Active-HDL.

Перед выполнением лабораторной работы студент должен самостоятельно выполнить домашнее задание в соответствии с данными методическими указаниями.

Студент, явившийся на занятия, должен иметь методические указания по данной лабораторной работе. В начале занятия преподаватель проверяет выполнение студентом домашнего задания и наличие заготовки отчета по данной лабораторной работе в его рабочей тетради.

К выполненной работе прилагаются необходимые схемы, эскизы, тексты и результаты проектирования, протоколы работы с программным комплексом и другие материалы согласно указаниям по оформлению отчета.

2.4.2. Домашние задания и указания по их выполнению

Задание 1. Ознакомиться с принципом работы, назначением и схемными конструкциями двоичных сумматоров. Для этого необходимо воспользоваться лекциями по курсу и литературой [1, С. 114-129].

Двоичные сумматоры позволяют суммировать два двоичных числа и являются основным блоком процессора. При сравнении процессоров наиболее важной характеристикой является разрядность сумматора, входящего в их состав. Для того чтобы получить многоразрядный сумматор, достаточно соединить входы и выходы переносов соответствующих двоичных разрядов [2]. Схема соединения одноразрядных сумматоров для реализации четырехразрядного сумматора приведена на рис. 2.4.1.

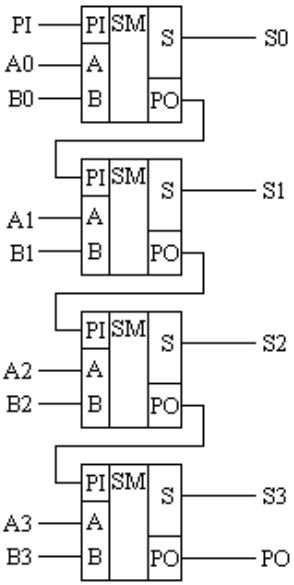

Рис. 2.4.1. Схема многоразрядного двоичного сумматора

Одноразрядные сумматоры практически никогда не использовались, так как почти сразу же были выпущены микросхемы многоразрядных сумматоров. Полный двоичный четырехразрядный сумматор изображается на схемах как показано на рис. 2.4.2.

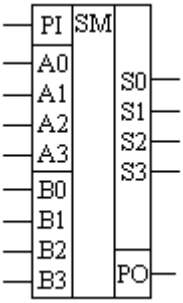

Рис. 2.4.2. Условно-графическое изображение полного двоичного многоразрядного сумматора

В приведенной на рис. 2.4.1 схеме рассматриваются только принципы работы двоичных сумматоров. В реальных схемах никогда не допускают последовательного распространения переноса через все разряды многоразрядного сумматора. Для увеличения скорости работы двоичного сумматора применяется отдельная схема формирования переносов для каждого двоичного разряда. Таблицу истинности для такой схемы легко получить из алгоритма суммирования двоичных чисел, а затем применить принципы построения цифровой схемы по произвольной таблице истинности.

2.4.3. Вопросы к домашнему заданию

1. Дать определение двоичному сумматору.
2. Пояснить принцип увеличения разрядности сумматора.
3. Пояснить работу сумматора по таблице истинности.

2.4.4. Лабораторные задания и указания по их выполнению

Задание 1. Построить и отладить модель одноразрядного двоичного сумматора. Схема данного

компаратора представлена на рисунке ниже. Для данной схемы необходимо построить временную диаграмму, демонстрирующую ее работу согласно таблице истинности, приведенной в домашнем задании.

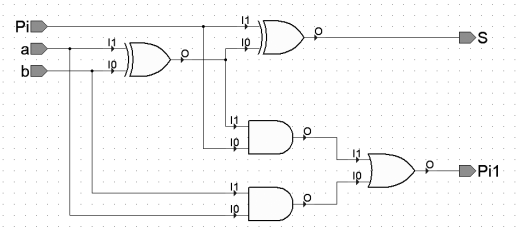

Рис. 2.4.3. Схема одноразрядного двоичного сумматора

Задание 2. Построить и отладить иерархическую модель четырехразрядного двоичного сумматора, опираясь на пример приведенный в домашнем задании, с использованием программного комплекса Active-HDL на языке описания аппаратуры VHDL. Схема сумматора представлена на рис. 2.4.4. Для данной схемы необходимо построить временную диаграмму, демонстрирующую ее работу.

Иерархическая модель сумматора строится на основе модели, полученной в первом задании.

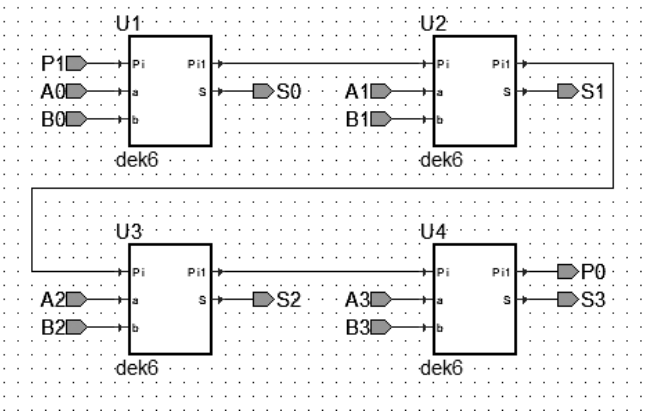

Рис. 2.4.4. Схема четырехразрядного двоичного сумматора

Задание 3. Проектирование модуля сумматора в среде Xilinx. Используя материал задания 2 и модель сумматора на языке описания аппаратуры VHDL, полученной на лабораторном практикуме, построить проект данного устройства в среде Xilinx ISE. В качестве внутреннего модуля в проекте использовать код VHDL, сгенерированный в результате моделирования в Active HDL. Подачу импульсов осуществить при помощи фиксируемых переключателей платы *sw*, перенос из младшего разряда по средствам кнопоки *btn*, для отображения результата суммирования использовать светодиоды *Ld*. Пример кода представлен ниже:

```
module summ (input [0:7] sw,
input [0:0] btn,
 output [0:4] ld);
summ4 impl(.A0(sw [0]),  .A1(sw [1]),
.A2(sw [2]), .A3(sw [3]), .B0(sw [4]),
.B1(sw [5]),  .B2(sw [6]), .B3(sw [7]),
.P1(btn[0]), .S0(ld[0]), .S1(ld[1]),
.S2(ld[2]), .S3(ld[3]),  .P0(ld[4]));

endmodule
```

SUMM4 – имя файла с расширением *.vhd. RTL код проектируемого четырехразрядного двоичного сумматора.

Параметры файла конфигурации *basys2.ucf*:

```
NET "ld<4>" LOC = "n5" ;
NET "ld<3>" LOC = "p6" ;
NET "ld<2>" LOC = "p7" ;
NET "ld<1>" LOC = "m11" ;
NET "ld<0>" LOC = "m5" ;
NET "sw<7>" LOC = "n3";
NET "sw<6>" LOC = "e2";
NET "sw<5>" LOC = "f3";
NET "sw<4>" LOC = "g3";
```

NET "sw<3>" LOC = "b4";
NET "sw<2>" LOC = "k3";
NET "sw<1>" LOC = "l3";
NET "sw<0>" LOC = "p11";
NET "btn<0>" LOC = "g12";

2.4.5. Контрольные вопросы

1. Какова цель лабораторной работы?
2. В чем заключается лабораторное задание? Пояснить ход его выполнения.
3. Какие данные являлись исходными для выполнения работы?
4. Пояснить работу полученной модели сумматора.
5. Сформулируйте выводы по данной лабораторной работе.

2.5. Указания по оформлению отчёта

Отчет по каждой лабораторной работе должен содержать наименование и цель работы, краткие теоретические сведения, ход и результаты выполнения лабораторного задания, где приводятся исходные данные и результаты работы с программным комплексом с необходимыми пояснениями. Отчет завершается кратким перечнем приобретенных навыков и выводами о результатах проделанной работы. Оформление отчета выполняется в соответствии со стандартом ВГТУ.

3. УКАЗАНИЯ ПО ОФОРМЛЕНИЮ КУРСОВОГО ПРОЕКТА

3.1. Общие указания по оформлению курсового проекта

Курсовой проект оформляется в виде пояснительной записки, содержащую теоретическую и практическую части. Темы курсовых работ разделены по вариантам. Номер варианта соответствует порядковому номеру студента в списке группы. Студент самостоятельно планирует выполнение работы в течение всего семестра, с учетом обеспечение равномерности работы. Возникающие трудности можно обсудить на плановых консультациях.

Курсовой проект оформляется по следующим правилам:

- используются чистые белые листы бумаги формата А4 по ГОСТ 9327;
- при наборе курсового проекта с использованием компьютера и принтера в текстовом редакторе Microsoft Word использовать следующие установки: шрифт Times New Roman 14 кегль, цвет шрифта - черный, междустрочный интервал - полуторный, отступ первой строки (абзацный отступ) 1,25 см, выравнивание текста - по ширине, в режиме качественной печати.
- необходимо соблюдать следующие размеры полей: левое - 20 мм, правое - 10 мм, верхнее - 20 мм, нижнее - 20 мм.
- страницы следует нумеровать арабскими цифрами, соблюдая сквозную нумерацию по всему тексту курсового проекта. Номер страницы проставляют в центре нижней части листа без точки, отступив один интервал от текста. Необходимо обратить особое внимание на параметры шрифта номера страницы, он должен совпадать с настройками для осинового текста.
- на все используемые формулы, справочные данные и выдержки из литературы необходимо делать ссылки.
- титульный лист, задание на курсовую работу и лист

замечаний руководителя включают в общую нумерацию страниц курсового проекта, но номера страниц на них не проставляют.

- объем курсового проекта должен составлять 20-25 страниц.

Более детально ознакомиться с правилами оформления текстовой и графической документации можно в СТП ВГТУ 005-2007.

Показателем качества курсового проекта служит не объем пояснительной записки или приложений, а аккуратность и четкость оформления; самостоятельность: степень соответствия заданию и методическим указаниям, правильность, обоснованность и непротиворечивость принимаемых решений: грамотность изложения (при написании работы и ее защите).

Пояснительная записка курсового проекта должна содержать следующие структурные элементы:

- титульный лист;
- задание на курсовую работу;
- лист замечаний руководителя;
- содержание;
- введение;
- основную часть;
- заключение;
- список литературы;
- приложения.

Титульный лист курсовой проект имеет единую форму, представленную в приложении 1. На титульном листе не допускаются исправления и перенос текста. Титульный лист подписывается руководителем курсового проекта, если она допускается к защите. Студентом она подписывается после написания и при сдаче руководителю.

Бланк задания располагается после титульного листа. Задание, содержит тему курсового проекта, исходные данные к работе, перечень подлежащих разработке вопросов, перечень графического материала, дату выдачи задания и срок

сдачи работы с подписями руководителя и исполнителя. Номер варианта совпадает с порядковым номер студента в группе и представлен в приложении 3.

Лист замечаний является третьим листом пояснительной записки курсового проекта. Содержит только название. По результатам проверки курсового проекта руководитель фиксирует в нем, замечания и дополнения. При повторной сдачи пояснительной записки лист замечаний не меняется.

Содержание курсового проекта должно включать в строгом соответствии с текстом пояснительной записки перечень заголовков всех разделов, подразделов работы, список использованных источников и приложения с указанием соответствующих страниц.

Введение - вступительный раздел основного текста курсового проекта. Оно должно содержать обоснование темы, ее актуальность, значение и задачи исследования. В нем определяется объект исследования, приводятся отдельные пояснения к содержанию работы, ее структуре, кругу исследуемых вопросов, указываются источники и фактические материалы, являющиеся методологической основой написания курсового поекта.

Основная часть должна отражать суть курсового проекта. Ее содержание более подробно описано в п.3.

Заключение должно показать практическую значимость выводов. Приводится краткое описание проделанной работы, полученных результатов, степень выполнения поставленной задачи, рекомендации по дальнейшим работам.

Список литературы служит составной частью курсового проекта и показывает степень изученности проблемы студентом. В него должны войти все упомянутые и использованные в тексте работы источники. Нумерация источников производится по мере упоминания о них в тексте.

Приложения. Материал, дополняющий содержание курсового проекта, допускается давать в виде приложений.

3.2. Краткие теоретические сведения

3.2.1. Проектирование модуля оперативного запоминающего устройства (ОЗУ) статического типа в среде Active HDL

В качестве примера рассмотрим структуры ОЗУ статического типа на 16 и 4 одноразрядных слов, построенных по схеме двух и однокоординатной выборки.

ОЗУ на 16 слов реализовано по схеме двухкоординатной выборки. Схема получена при помощи двух дешифраторов адреса выбора столбцов *DCy* и строк *DCx* матрицы запоминающих ячеек (ЗЯ), на пересечении которых и расположена искомая запоминающая ячейка памяти (см. рис. 3.2.1) [3].

Запоминающий элемент (ЗЭ) ОЗУ предназначен для хранения 1 бита информации, а запоминающая ячейка – 1 слова. Слово в данном случае примем одноразрядное, соответственно разрядность ячейки памяти – 1 бит. Запоминающий элемент представляет собой RS-триггер, в данном случае на элементах ИЛИ-НЕ (см. рис. 3.2.2) .

Триггер имеет два входа – установки в единицу (S) и установки в ноль (R) , два выхода, но в данном случае необходим один *Xout*.

Схема запоминающей ячейки на одно слово представлена на рис. 3.2.3. Рассмотрим далее сигналы и блоки данной ячейки. В данной модели на 16 запоминающих ячеек (см. рис. 3.2.5) используются ячейки памяти в соответствии со схемой рис. 3 и имеют два адресных входа.

Ax, Ay – адресные входы соответственно дешифраторов сторок и столбцов матрицы;

CS – сигнал разрешающий или запрещающий работу ЗЯ;

RW – сигнал чтение/запись ЗЯ (при подаче «1» – чтение, при «0»- запись) ;

RESET – сброс, установка ЗЯ в ноль;

Xin, Xout – соответственно вход и выход ячейки памяти;

Запись в ячейку осуществляется при наличии единиц на *Ax, Ay* и *CS*. Слово здесь продолжает храниться до прихода следующей команды. Чтение осуществляется при наличии единицы на *RW*, что приводит к срабатыванию блока *and4* и прохождению слова на выход ячейки памяти *Xout*. Обнуление ЗЯ происходит при подачи на *CS* «нуля», а на *RESET* «единицы».

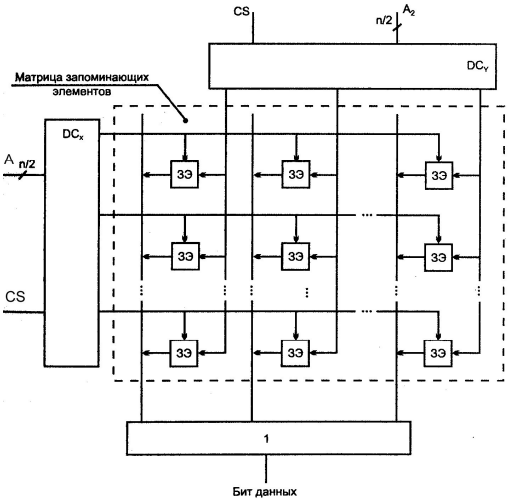

Рис. 3.2.1. Структура двухкоординатной выборки ОЗУ статического типа

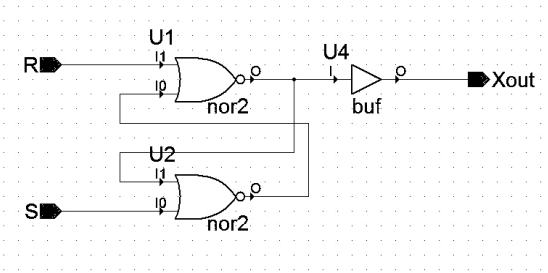

Рис. 3.2.2. Схема запоминающего элемента

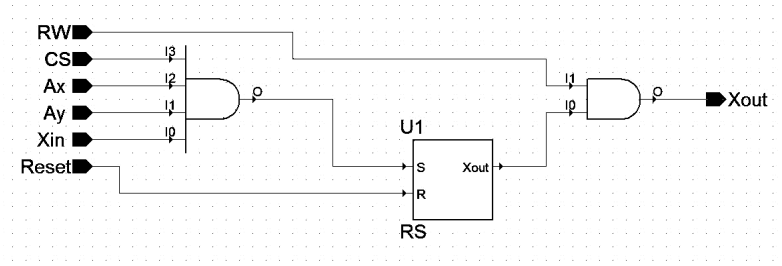

Рис. 3.2.3. Запоминающая ячейка ОЗУ статического типа с двухкоординатной выборкой

В качестве двух дешифраторов адреса используется дешифраторы «2-4» с разрешающим входом «E».

ОЗУ статического типа на 4 сова построенного по схеме однокоординатной выборки (см. рис. 3.2.6). Для решения данной задачи используется запоминающая ячейка с одним адресным входом (см. рис. 3.2.4).

В качестве дешифратора адреса используется дешифратор «2-4» с разрешающим входом «E».

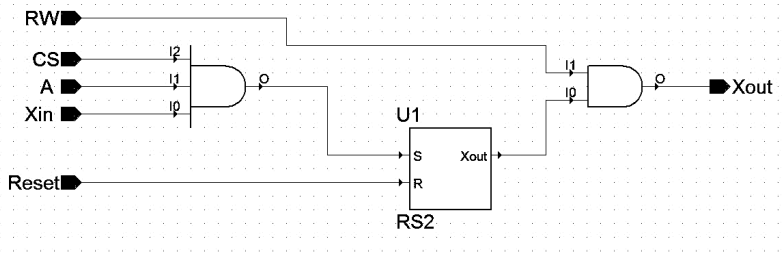

Рис. 3.2.4. Запоминающая ячейка ОЗУ статического типа с однокорр динатной выборкой

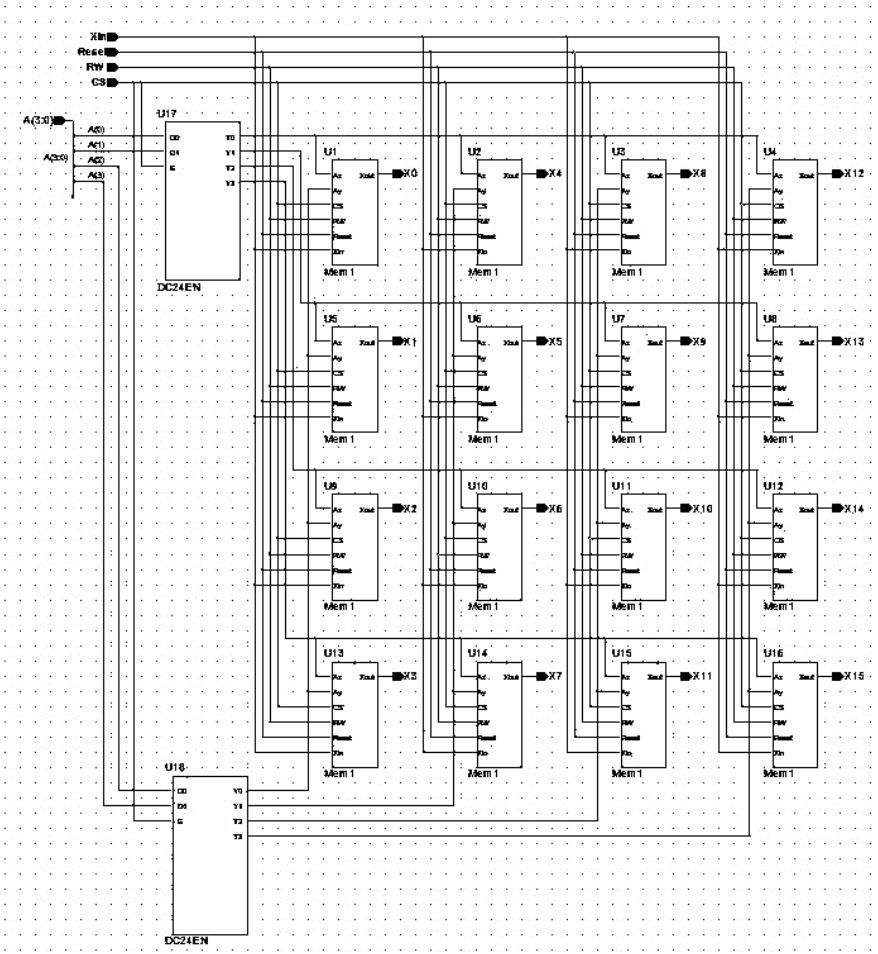

Рис. 3.2.5. Схема ОЗУ статического типа на 16 запоминающих ячеек

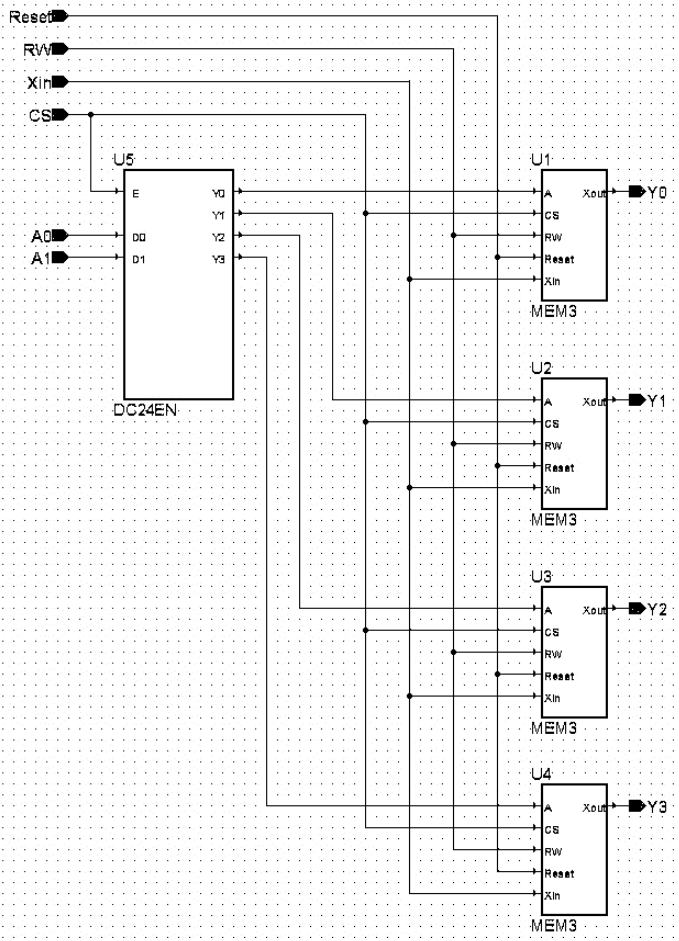

Рис. 3.2.6. Схема ОЗУ статического типа на 4 запоминающие ячейки

3.2.2. Проектирование модуля оперативного запоминающего устройства (ОЗУ) статического типа в среде Xilinx ISE

Используя модель ОЗУ статического типа на 4 ячейки памяти, рассмотрим проект данного устройства в среде Xilinx ISE. В качестве внутренних модулей в данном случае необходимо использовать коды всех элементов иерархической структуры ОЗУ. Данное ОЗУ реализовано по схеме однокоординатной выборки. Схема получена при помощи одного дешифраторы адреса и матрицы запоминающих ячеек. Пример кода представлен ниже:

```
module RAM_4_2(input [0:2] btn,
input [0:2] sw,
output [0:3] Ld);

RAM4_1 impl(.A0(sw [0]), .A1(sw [1]),
 .CS(sw [2]), .Xin(btn [2]),
.Reset (btn [0]), .RW (btn [1]),
 .Y0(Ld[0]), .Y1(Ld[1]),
 .Y2(Ld[2]), .Y3(Ld[3]));

endmodule
```

RAM4_1 – имя файла с расширением *.vhd. RTL код проектируемого ОЗУ, кторый в себя также включает коды ячеек памяти, дешифратора адреса и запоминающих элементов.

Параметры файла конфигурации *basys2.ucf*:

```
NET "ld<3>" LOC = "p6" ;
NET "ld<2>" LOC = "p7" ;
NET "ld<1>" LOC = "m11" ;
NET "ld<0>" LOC = "m5" ;
NET "sw<2>" LOC = "k3";
```

NET "sw<1>" LOC = "l3";
NET "sw<0>" LOC = "p11";
NET "btn<2>" LOC = "m4";
NET "btn<1>" LOC = "c11";
NET "btn<0>" LOC = "g12";

Запись в ячейку памяти осуществляется при наличии импульса на *A0, A1* и «единицы» на *CS*. Слово в запоминающей ячейке продолжает храниться до прихода следующей команды. Чтение осуществляется при наличии «единицы» на *RW*. Обнуление запоминающей ячейки происходит при подаче на *CS* «нуля», а на *RESET* «единицы».

Управление сигналами ОЗУ осуществляется следующим образом:

sw [0], sw [1] – адресные входы (*A0, A1*) дешифратора ОЗУ, формирующие уровень логической единицы на адресном входе *A* одной из четырех запоминающих ячеек;

sw [2] – задание стробирующего (*CS*) импульса ОЗУ;

btn [2] – задание входной информационной последовательности (*Xin*) в один бит;

btn [0] – сброс всех ячеек памяти (*RESET*);

btn [1] – переключение между режимами чтение/запись (*RW*);

Ld[0] ... Ld[3] – отображение результатов записи, выходы ячеек памяти (*Y0 ... Y3*).

3.3. Структура и требования к основной части курсового проекта

Целью курсового проекта является углубление и закрепление знаний студентов в области проектирования цифровых устройств с использованием языков высокого уровня описания аппаратных средств. В рамках данного курсового проекта необходимо построить на вентильном уровне модель оперативного запоминающего устройства (ОЗУ) статического типа с использованием программного комплекса Active-HDL и Xilinx ISE на языке описания аппаратуры VHDL.

В курсовом проекте рекомендуется следующее содержание основной части:

1. Постановка и анализ задачи:
- формулировка условия задачи;
- определение конечных целей решения задачи;
- определение формы выдачи результатов;
- анализ технических и программных средств;
- описание работы и структуры устройств аналогов.

2. Разработка структуры проектируемого устройства:
- проектирование структуры функциональных узлов устройства;
- выбор тестов и метода проверки;
- разработка программы управления отладочной платой.

3. Результаты моделирования:
- временные диаграммы работы устройства;
- программирование отладочной платы, демонстрация работы устройства.

Полученную модель ОЗУ необходимо откомпилировать и проверить на работоспособность. Реализовать циклы запись/чтение запоминающих ячеек в соответствии со следующими итерациями: Сброс (*RESET*), включение схемы (*CS*), адресная запись информации в каждую ячейку памяти, адресное считывание данных, сброс ячеек (*RESET*).

Для этого необходимо в редакторе временных диаграмм программного комплекса Active HDL выбрать необходимые режимы тестирования. Полученные временные диаграммы должны полностью отражать работу устройства.

Продемонстрировать работу полученного ОЗУ на отладочной плате ПЛИС согласно итерациям, указанным в таблице ниже.

Порядок работы ОЗУ

	Сброс ячейки	**Режим чтения**	**Режим записи**	**Режим чтения**	**Сброс ячейки**	**Режим чтения**
Reset	1	0	0	0	1	0
RW	0	1	0	1	0	1
CS	0	0	1	0	0	0
A	0	0	1	0	0	0
Xin	0	0	1	0	0	0

Приложение 3 регламентирует структуру проектируемого ОЗУ согласно варианту. В нем указано требуемое количество запоминающих ячеек и их разрядность, схема реализации матрицы ячеек двухкоординатная (два дешифратора адреса) или однокоординатная (один дешифратор адреса) выборка. Режим считывания информации должен быть единовременный (при поступлении сигнала на вход R\W записанные данные подаются на выходы всех запоминающих ячеек) или с учетом адреса (при поступлении сигнала на вход R\W данные подаются на выходы только тех запоминающих ячеек, которые выбраны дешифраторами адреса).

К пояснительной записке должен прилагаться диск, на котором содержаться построенные модели, а также готовый файл прошивки отладочной платы.

4. ПРАКТИЧЕСКИЕ ЗАНЯТИЯ

4.1. Разработка аналого-цифрового преобразователя (АЦП) на основе отладочной платы AVR

4.1.1. Общие указания по выполнению практических работ

Цель работы построение алгоритма АЦП и разработка программы на языке ассемблера с использованием программного комплекса Almel Studio.

4.1.2. Задания для выполнения практических работ

Задание 1. Ознакомьтесь с исходными данными для подготовки теоретической части отчета (Источник 1). Ознакомьтесь с планом отчета по работе.

Представьте управляющие регистры в формате режима разрешения использования и прерывания по АЦП для нулевого канала с использованием внешнего источника опорного напряжения и второго канала для внутреннего источника опорного напряжения. Оформите отчет.

Условие: частота тактового генератора 1МГц.

План отчета по работе:

1. Структурная схема АЦП, характеристика и назначение функциональных блоков;
2. Описание регистров управления ADCSR и ADMUX, форматы выбора коэффициентов деления тактовой частоты, источника опорного напряжения, канала АЦП;
3. Управляющие регистры ADCSR и ADMUX в формате режима разрешения использования и прерывания по АЦП для нулевого канала с использованием внешнего источника опорного напряжения и второго канала для внутреннего источника опорного напряжения.

Источник 1

Применение аналого-цифрового преобразователя

В своем составе ряд микроконтроллеров имеют аналого-цифровой преобразователь (АЦП). Структурная схема АЦП микроконтроллера приведена на рис. 4.1.1.

АЦП содержит 8-канальный аналоговый мультиплексор входных сигналов, регистр выбора входного сигнала ADMUX, 10-разрядный цифроаналоговый преобразователь, 8-разрядный регистр управления ADCSR, 16 - разрядный регистр данных, схему формирования запроса прерывания ADC CC, схему компаратора и предварительный делитель (ПД) тактовой частоты. Формат регистра ADCSR представлен в табл. 4.1.1.

Таблица 4.1.1

Формат регистра управления ADCSR

№ разряда	7	6	5	4	3	2	1	0
Имя	ADEN	ADSC	ADFR	ADIF	ADIE	ADPS2	ADPS1	ADPS0

Назначение разрядов регистра ADCSR:

– ADPS0 – ADPS2 – выбор коэффициента деления тактовой частоты;

– ADIE – разряд маскирования прерывания от АЦП (1 – по окончанию преобразования разрешено прерывание);

– ADIF – флаг прерывания от АЦП (устанавливается аппаратно по окончанию цикла преобразования);

– ADFR – лог. 1 в этом разряде переводит АЦП в непрерывный режим работы – обычно АЦП работает в режиме прерывания, чтобы процессор каждый раз не ожидал завершения медленно протекающего преобразования, однако в непрерывном режиме АЦП выполняет преобразование постоянно, как можно быстрее (на период такого преобразования должны быть запрещены все прерывания);

– ADSC – флаг начала преобразования;

– ADEN – флаг разрешения использования АЦП.

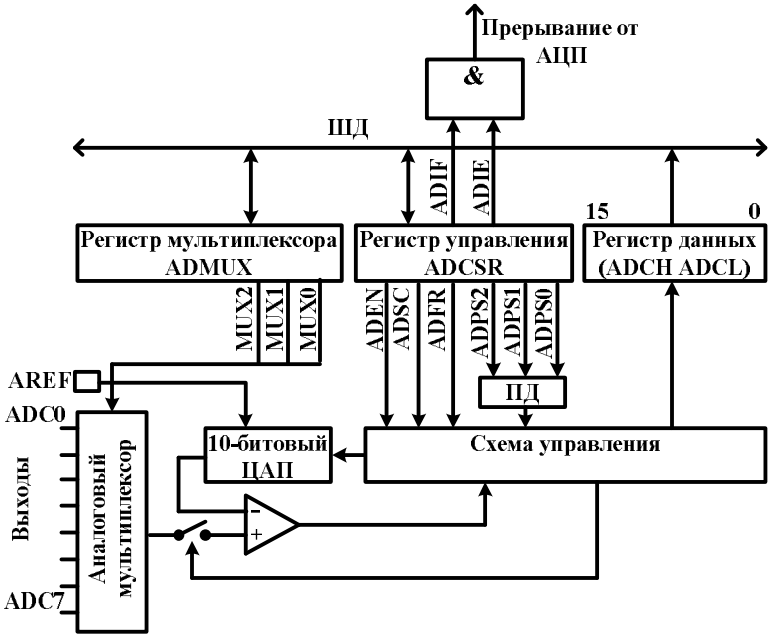

Рис. 4.1.1. Схема аналого-цифрового преобразователя

Преобразователь работает по методу последовательных приближений, формируя 10-разрядный двоичный код, размещаемый в регистре данных. Работа преобразователя выполняется на частоте от 50 до 200 кГц. Для получения этой частоты используют делитель тактовой частоты с заданным коэффициентом деления. АЦП может работать и на более высокой частоте. При этом точность преобразования уменьшается. Например, при частоте преобразования 1 МГц преобразование осуществляется с точностью 8 разрядов.

Значение коэффициента деления тактового генератора задается с помощью трех разрядов ADPS2, ADPS1, ADPS0 регистра управления ADCSR согласно табл. 4.1.2.

Таблица 4.1.2

Таблица коэффициентов деления частоты тактового генератора МК

ADPS2	ADPS1	ADPS0	K	ADPS2	ADPS1	ADPS0	K
0	0	0	1	1	0	0	16
0	0	1	2	1	0	1	32
0	1	0	4	1	1	0	64
0	1	1	8	1	1	1	128

В качестве источника опорного напряжения (ОН) для АЦП может использоваться как напряжение питания микроконтроллера, так и внутренний либо внешний источник опорного напряжения. Выбор источника опорного напряжения осуществляется с помощью 6 и 7 разрядов регистра ADMUX согласно табл. 4.1.3.

Таблица 4.1.3

Выбор источника опорного напряжения

REFS1 (7 разряд)	REFS0 (6 разряд)	Источник опорного напряжения (ОН)
0	0	Внешний источник ОН, подключенный к выводу AREF.
0	1	Внутренний источник. ОН равно напряжению питания МК.

Выбор канала преобразования АЦП осуществляется с помощью разрядов MUX0 (0 разряд), MUX1 (1 разряд), MUX2 (2разряд) регистра мультиплексора ADMUX согласно табл. 4.

Преобразование начинается после установки разряда ADSC (6 разряд) в регистре управления ADCSR. После завершения преобразования устанавливается разряд ADIF (4 разряд) регистра ADCSR. Этот разряд используется для формирования запроса прерывания ADC при разрешающем значении бита ADIE (3 разряд) регистра ADCSR. При переходе к программе обслуживания прерывания от АЦП бит ADIF аппаратно сбрасывается в нулевое состояние. Программно этот бит можно сбросить в 0 путем установки 0 в данный разряд.

Таблица 4.1.4

Выбор канала АЦП

MUX2	MUX1	MUX0	Канал АЦП
0	0	0	ADC0
0	0	1	ADC1
0	1	0	ADC2
0	1	1	ADC3
1	0	0	ADC4
1	0	1	ADC5
1	1	0	ADC6
1	1	1	ADC7

Результат преобразования, представляющий 10-разрядный двоичный код, размещается в старшей и младшей половинах регистра данных ADCH, ADCL. Считывание результата выполняется с помощью двух операций. Сначала считывается младший байт ADCL, затем старший ADCH. Такой порядок обеспечивает принадлежность читаемых данных одному и тому же результату преобразования.

АЦП может работать в одиночном или циклическом режиме. Выбор режима осуществляется с помощью бита ADFR (5 разряд) регистра управления ADCSR: при ADFR=0 выполняется одиночный режим преобразования, при ADFR=1 − циклический. В обоих случаях преобразование начинается после установки бита ADSC (6 разряд). В одиночном режиме для выполнения следующего преобразования необходимо снова установить бит ADSC.

В циклическом режиме следующее преобразование начинается автоматически после завершения предыдущего и прекращается после сброса бита ADFR.

Для уменьшения помех, вызываемых работой процессора, предусмотрена возможность преобразования с переводом микроконтроллера в режим холостого хода. Для этого биты управления устанавливают в состояния: ADEN=1, ADSC = 0, ADFR=0, ADIE=1. Далее контроллер переводится в режим холостого хода, при этом запускается АЦП. После выполнения преобразования формируется запрос прерывания,

контроллер выходит из режима холостого хода и выполняет прерывающую программу.

Задание 2. Составление блок-схемы алгоритма инициализации АЦП и основной программы обработки прерываний по двум каналам.

Ознакомьтесь с исходными данными для подготовки теоретической части отчета (Источник 2). Ознакомьтесь с планом отчета по работе.

Составьте блок-схему алгоритма инициализации АЦП и основной программы обработки прерываний для последовательной работы двух каналов АЦП.

Условие: частота тактового генератора 8МГц; порт В соответствует ADCL, порт С соответствует ADCH; опорное напряжение – внутренний источник; каналы работы – нулевой и второй.

План отчета по работе:

1. Блок-схема алгоритма инициализации АЦП и основной программы обработки прерываний;
2. Наименования и назначение выводов микроконтроллера ATmega16;
3. Блок-схемы алгоритма инициализации АЦП и основной программы обработки прерываний по двум каналам АЦП с указанием в каждом блоке используемых команд и форматов управляющих регистров.

Источник 2

Алгоритм преобразования аналогового сигнала в цифровой

Блок схема алгоритма преобразования аналогового сигнала в цифровой представлена на рис. 4.1.3.

Принципиальная электрическая схема применения МК, для преобразования аналогового сигнала в цифровой представлена на рис. 4.1.2.

Внутренний генератор МК работает с внешним кварцевым резонатором ZQ1 с резонансной частотой 1МГц. Порты B и C используются для вывода информации в двоичном коде на светодиоды VD1 – VD10. Резисторы R2 – R11 ограничивают ток, протекающий через светодиоды. Аналоговый сигнал снимается с потенциометра R1 и подается на нулевой вход аналогового мультиплексора (PA0). Напряжение питания +5В формирует стабилизатор напряжения DA1. Конденсаторы C1, C2 применяют для уменьшения низкочастотных и высокочастотных помех по питанию.

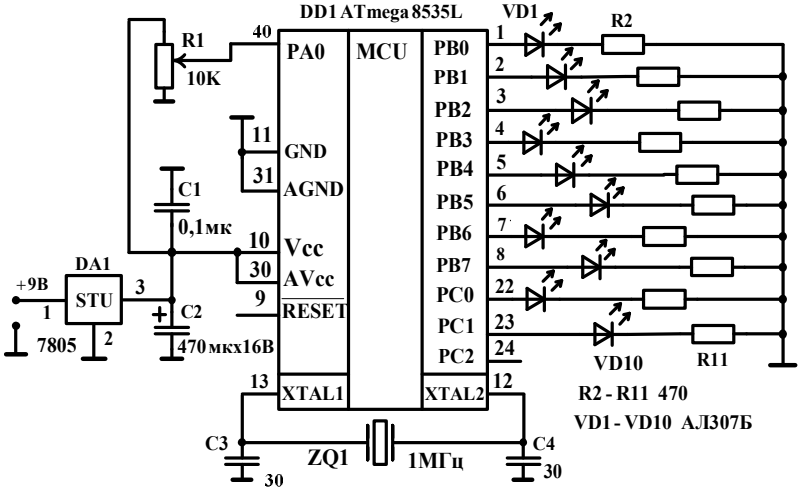

Рис. 4.1.2. Принципиальная схема устройства преобразования аналогового сигнала в цифровой

Текст программы, реализующий поставленную задачу, представлен в табл. 4.1.5. Программа представляет собой последовательный набор директив, команд и комментариев. Комментарии начинаются с «точки с запятой». Они игнорируются в процессе трансляции и необходимы для понятия работы программы.

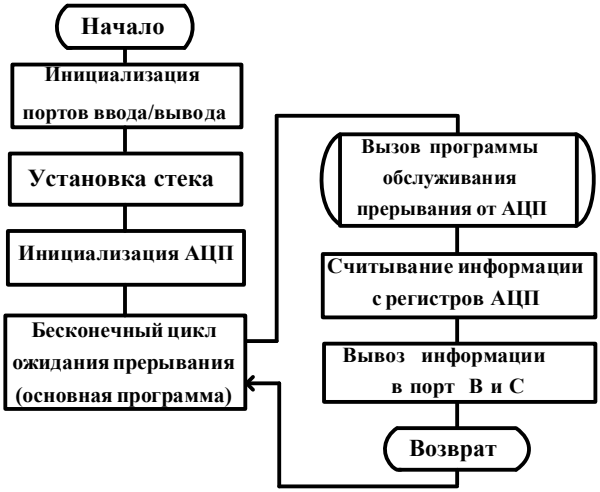

Рис. 4.1.3. Блок схема алгоритма

В случае необходимости перед командой ставят метку**.** Имя метки начинается с латинской буквы или символа подчеркивания. Допускается ставить метку в любой строке программы. В конце метки ставится двоеточие (13 строка программы).

Описание программы

Текст программы начинается с комментариев, где указано название программы и ее назначение.

Строки 4,5 содержат файл присоединения и обозначение РОН r16.

В строках 9-11 записаны команды переопределения вектора прерываний. Переопределение состоит в том, в адрес начала процедуры обработки прерывания помещается команда безусловного перехода, передающая управление на начало подпрограммы обслуживания прерывания.

Вектор прерываний содержит прерывание RESET, возникающее при подаче низкого уровня на вход RESET и

прерывание, которое выдает АЦП, если процесс преобразования закончен. Подпрограмма обслуживания прерывания RESET обозначена меткой init. Подпрограмма обслуживания прерывания от АЦП обозначена меткой adc.

Программа инициализации начинается с метки init. В строках 15-18 производится установка стека. Максимальный адрес ОЗУ выбран в качестве вершины стека.

В строках 20-29 производится настройка портов ввода-вывода. Порты B и C настраиваем на вывод информации. Порт D – на ввод информации и включаем подтягивающие резисторы.

В строках 30-36 производится настройка АЦП на преобразование в одиночном режиме. Значение коэффициента деления тактового генератора выбирается равным 1/8. При тактовой частоте МК 1 МГц частота работы АЦП будет равна 125 КГц. В строке 33-34 устанавливается 0 адрес мультиплексора и выбирается внутренний источник опорного напряжения с фильтрующим конденсатором, подключенным к входу AREF.

Строка 35 разрешает прерывание.

Строка 36 осуществляет запуск АЦП.

Строка 37 организует бесконечный цикл ожидания прерывания.

После окончания преобразования выдается сигнал прерывания от АЦП. Подпрограмма обслуживания прерывания от АЦП расположена в строках 38 - 46. Результат преобразования выводится в порты B и C. Строк 43 разрешает прерывание от АЦП. Строка 45 запускает очередное преобразование.

Таблица 4.1.5

Листинг программы использования АЦП

```
1.  ;Тестовая программа работы АЦП в одиночном режиме с
2.  ;просмотром 10-разрядного выходного кода на светодиодах,
3.  ; подключенных к выводам порта B и C.
4.  .INCLUDE «m16def.inc»       ; файл определений ATmega16
5.  .DEF   temp = r16           ; временный регистр
6.  ;-----------------вектор прерываний---------------------------
```

Окончание табл. 4.1.5

```
7.    .CSEG
8.    .ORG $000
9.    Rjmp init    ; подпрограмма инициализации
10.   .ORG 0x1c
11.   Rjmp k1    ; подпрограмма обслуживания прерывания АЦП
12.   ;----------подпрограмма инициализации------------------
13.   init:
14.   ;--------------установка стека-----------------------------------
15.   ldi      temp, low (RAMEND)   ; установка
16.   out      SPL,  temp                ; указателя стека
17.   ldi      temp, high (RAMEND)  ; на последнюю
18.   out      SPH,   temp               ; ячейкуОЗУ
19.   ------инициализация портов ввода/вывода-----------------
20.   Ldi  temp,   0xFF     ; установка единиц в регистр temp
21.   out  DDRC, temp     ; порт PC на вывод
22.   out  DDRB, temp     ; порт PB на вывод
23.   ldi   temp,    0x00     ; установка нулей  в регистр temp
24.   out  DDRD, temp     ; порт PD наввод
25.   ldi   temp,    0xFF     ; включение подтягивающих резисторов
out  PORTD, temp          ; порта D
26.   ldi   temp,    0x00      ; установка нулей  в регистр temp
27.   out  DDRA,  temp     ; установка порта PA на ввод
28.                              ; аналогового  сигнала
29.   Out  PORTA,  temp  ; подтягивающие резисторы отключены
30.       ;------------Инициализация АЦП-------------------------
31.   ldi       temp,   0b10001011;ADEN=1, ADIE=1, Fadc=Fclk/8
32.   out     ADCSR,  temp          ; Fadc=125 кГцприFclk=1,0 МГц
33.   ldi      temp,   0b11000000   ; Канал 0 АЦП (вход PA0)
34.   out     ADMUX, temp          ; внутреннее опорное напряжение
35.   sei                               ; разрешение прерываний
36.   sbi    ADCSR,   ADSC       ; пуск преобразования
37.   loop: rjmploop                ; цикл ожидания прерываний
38.   ;-----------------Обработка прерывания от АЦП--------------------
39.   k1:
40.   in rl8,   ADCL                ; считывание ADCL
41.   in rl7,   ADCH                ; считывание ADCH
42.   out PORT B, r18               ; вывод младших разрядов
43.   out PORT C, r17               ; вывод старших разрядов
44.   sbi ADCSR,  ADIE          ; разрешение прерывания от АЦП
45.   sbi ADCSR,  ADSC          ; пуск преобразования
reti
```

Задание 3. Разработка программы использования АЦП на языке ассемблера с применением программного комплекса Almel Studio по индивидуальным вариантам.

Изучите перечень директив и команд, которые необходимо использовать для выполнения задания (Источник 3). Ознакомьтесь с исходными данными для вашего варианта (Источник 4).

Настройте работу АЦП по требуемому каналу с заданным источником опорного напряжения и требуемым коэффициентом деления тактовой частоты.

Создайте проект с названием в формате: <Фамилия>_<номер варианты> (Пример: Ivanov_08). Файл с листингом программы должен быть в файле main.asm вашего проекта. С использованием внутрисхемного программатора прошить отладочную плату, продемонстрировать работоспособность.

План отчета по работе:

1. Блок-схема алгоритма инициализации АЦП и основной программы обработки прерываний с использованием данных вашего варианта;
2. Листинг программы инициализации АЦП и основной программы обработки прерываний с использованием данных вашего варианта на языке ассемблера.

Источник 3

Перечень команд, которые необходимо использовать при выполнении работы

Листинг программы использования АЦП (Источник 2) содержит следующие директивы и команды:

.INCLUDE

Присоединение к текущему тексту программы другого фрагмента программы, являющегося общим для многих программ. В 4 строке программы присоединяется файл

m16def.inc. Он содержит описание всех регистров микроконтроллера ATmega 16.

.DEF

Данная директива позволяет присваивать регистрам любые имена. В строке 5 регистру общего назначения (РОН) R16 присвоено имя temp.

.CSEG

Данная директива указывает, что все последующие команды в виде кодов будут записаны в память программ.

.ORG

Данная директива указывает, что все последующие команды будут записаны с указанного адреса. В строке 8 указатель устанавливает нулевой адрес. В строке 10 - на двадцать восьмой адрес 0x1c.

rjmp

Команда безусловного перехода. И строке 9 программы управление передается на строку меченную меткой init (инициализация МК).

ldi

Загрузка в РОН числовой константы. Данная команда работает только с регистрами r16 - r31. В строке 15 в регистр temp записываются младшие разряды числовой константы, указывающей максимальное значение ОЗУ. Эта константа имеет имя RAMEND. Функция low (RAMEND) выделяет младшие разряды константы. Старшие разряды константы выделяет функция high (RAMEND).

sei

Общее разрешение прерываний. Данная команда разрешает прерывание основной программы.

out

Команда вывода содержимого РОН в регистры ввода-вывода. В строке 16 программы содержимое регистра temp записывается в регистр с именем SPL (младший байт регистра стека).

sbi

Установка разряда порта ввода-вывода. В строке 36 эта команда устанавливает «1» и разряд ADSC регистра ADCSR. Это необходимо для запуска АЦП.

Источник 4

Таблица 4.1.5

Исходные данные по вариантам

№	Канал АЦП	Источник опорного напряжения (ОН)	Используемы ПВВ	Заданная тактовая частота, МГц
1	ADC0	ОН равно напряжению питания МК	B (младший), C (старший)	1
2	ADC1	ОН равно напряжению питания МК	C (младший), B (старший)	8
3	ADC2	ОН равно напряжению питания МК	B (младший), D (старший)	2
4	ADC3	ОН равно напряжению питания МК	D (младший), B (старший)	7
5	ADC4	ОН равно напряжению питания МК	B (младший), C (старший)	3
6	ADC5	ОН равно напряжению питания МК	C (младший), B (старший)	6
7	ADC6	ОН равно напряжению питания МК	B (младший), D (старший)	4
8	ADC7	ОН равно напряжению питания МК	D (младший), B (старший)	5
9	ADC0	Внешний источник ОН, подключенный к выводу AREF	B (младший), C (старший)	5
10	ADC1	Внешний источник ОН, подключенный к выводу AREF	C (младший), B (старший)	6
11	ADC2	Внешний источник ОН, подключенный к выводу AREF	B (младший), D (старший)	4
12	ADC3	Внешний источник ОН, подключенный к выводу AREF	D (младший), B (старший)	7
13	ADC4	Внешний источник ОН, подключенный к выводу AREF	B (младший), C (старший)	3
14	ADC5	Внешний источник ОН, подключенный к выводу AREF	C (младший), B (старший)	8
15	ADC6	Внешний источник ОН, подключенный к выводу AREF	B (младший), D (старший)	2
16	ADC7	Внешний источник ОН, подключенный к выводу AREF	D (младший), B (старший)	1

Задание 4. Проведение оценки результатов лабораторной работы на основе анализа полученных результатов преобразования АЦП.

Ознакомьтесь с принципом формирования отчетов работы АЦП отладочной платы AVR (Источник 5). Подайте на вход отладочной платы аналоговый сигнал, изменяя значение входного напряжения, запишите отчеты АЦП и номера уровней квантования, которые выводятся на системах индикации отладочной платы.

План отчета по работе:

1. Определен порог напряжения ΔU для перехода на следующий уровень квантования;
2. Отчеты работы АЦП для входного напряжения в 1В, 2.5В и 5В.

Источник 5

Формирование отчетов АЦП

Подавая на вход отладочной платы различные уровни допустимого напряжения (от 0 до 5В), АЦП осуществляет преобразование аналогового сигнала в 10 разрядный двоичный код. Максимальный номер уровня квантования в данном случае равен 1023, которому соответствует максимальное число, загруженное в регистр ADC.В зависимости от значения опорного напряжения, порог напряжения ΔU для перехода на следующий уровень квантования может быть различным. Уменьшая амплитуду входного аналогового сигнала, изменяется номер уровня и уменьшается число, загруженное в регистр ADC.

ЗАКЛЮЧЕНИЕ

Материал пособия представлен в виде краткого обзора некоторых вопросов проектирования программируемых логических интегральных схем (ПЛИС). Рассмотрены структуры и принципы работы основных цифровых функциональных узлов комбинационного и последовательностного типов.

Особое внимание уделено структурам статических оперативных запоминающих устройств (ОЗУ), так как их проектирование на базе программируемых интегральных схем вынесено в качестве задания на курсовую работу.

Учебное пособие необходимо студентам направления 11.03.03 «Конструирование и технология электронных средств», профиль «Проектирование и технология радиоэлектронных средств» для выполнения заданий лабораторных и практических занятий, курсового проекта по дисциплине «Интегральные устройства радиоэлектроники».

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Угрюмов, Е. П. Цифровая схемотехника : учеб. пособие / Е.П. Угрюмов. – 2-е изд., перераб. и доп. – СПб. : БХВ-Петербург, 2004. – 800 с.

2. Солонина, А.И. Основы цифровой обработки сигналов: учеб. пособие / А.И. Солонина, Д.А. Ухладович, С.М. Арбузов, Е.Б. Соловьева. – СПб. : БХВ-Петербург, 2005. – 768 с.

3. Солонина, А. И. Алгоритмы и процессоры цифровой обработки сигналов: учеб. пособие / А.И. Солонина, Д.А. Улахович, Л.А. Яковлев. – СПб. : БХВ-Петербург, 2002. – 464 с.

ПРИЛОЖЕНИЕ 1

Форма титульного листа курсового проекта

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ” (ФГБОУ ВО “ВГТУ”)

Факультет радиотехники и электроники

Кафедра конструирования и производства радиоаппаратуры

Направление 11.03.03 Конструирование и технология электронных средств
Профиль Проектирование и технология радиоэлектронных средств

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту по дисциплине
«Интегральные устройства радиоэлектроники»
Вариант №

Выполнил студент:
Группа:

Проверил:

Воронеж 2018

ПРИЛОЖЕНИЕ 2

Форма листа задания на курсовой проект

Воронежский государственный технический университет
Кафедра конструирования и производства радиоаппаратуры

ЗАДАНИЕ

на курсовую работу студента группы _____
Направление 11.03.03 Конструирование и технология электронных средств
Профиль Проектирование и технология радиоэлектронных средств

_____
(Ф.И.О.)

1.Тема: «Проектирование оперативного запоминающего устройства статического типа»

2.Техническое задание. Для выбранного варианта задания согласно Приложению В:

– разработать структуру функциональных узлов устройства;
– разработать программу прошивки отладочной платы;
– разработать систему тестирования устройства.

3.Содержание и объём курсового проекта.

Пояснительная записка должна содержать: титульный лист, задание на курсовую работу, лист замечаний руководителя, содержание, введение, основную часть, заключение, список литературы, приложения.

Приложение А. Временные диаграммы работы ОЗУ;
Приложение В. Листинг функциональных узлов ОЗУ;
Приложение В. Листинг программы прошивки отладочной платы;

Объём пояснительной записки: 20-25 страниц включая приложения.

Дата выдачи задания: «____» _____ 20__г.
Срок защиты: до «____» _____ 20__г.

Руководитель проекта: _____
(Ф.И.О.)
Проект защищён с оценкой _____

«____» _____ 20__г.

ПРИЛОЖЕНИЕ 3

Варианты заданий для курсового проекта

№	Количество ячеек ОЗУ	Разрядность ячеек ОЗУ	Количество дешифраторов адреса	Использование генератора тактового сигнала отладочной платы	Режим чтения
1	4	1	1	Нет	Единовременный
2	4	2	1	Да	Единовременный
3	4	1	1	Да	Единовременный
4	4	2	1	Нет	Единовременный
5	8	1	1	Нет	Единовременный
6	8	1	1	Да	Единовременный
7	4	1	2	Нет	Единовременный
8	4	1	2	Да	Единовременный
9	4	2	2	Нет	Единовременный
10	4	2	2	Да	Единовременный
11	8	1	2	Да	Единовременный
12	8	1	2	Нет	Единовременный
13	4	1	1	Нет	С учетом адреса
14	4	2	1	Да	С учетом адреса
15	4	1	1	Да	С учетом адреса
16	4	2	1	Нет	С учетом адреса
17	8	1	1	Нет	С учетом адреса
18	8	1	1	Да	С учетом адреса
19	4	1	2	Нет	С учетом адреса
20	4	1	2	Да	С учетом адреса
21	4	2	2	Нет	С учетом адреса
22	4	2	2	Да	С учетом адреса
23	8	1	2	Да	С учетом адреса
24	8	1	2	Нет	С учетом адреса
25	2	4	1	Нет	Единовременный
26	2	4	1	Да	Единовременный
27	2	4	1	Нет	С учетом адреса
28	2	4	1	Да	С учетом адреса

ПРИЛОЖЕНИЕ 4

Сводная таблица команд микроконтроллера AVR

Группа команд логических операций

Мнемоника	Описание	Операция	Циклы	Флаги
AND Rd,Rr	«Логическое И» двух РОН	Rd=Rd • Rr	1	Z,N,V
ANDI Rd,K	«Логическое И» РОН и константы	Rd = Rd • К	1	Z,N,V
EOR Rd, Rr	«Исключающее ИЛИ» двух РОН	Rd = Rd ®Rr	1	Z,N,V
OR Rd,Rr	«Логическое ИЛИ» двух РОН	Rd = RdvRr	1	Z,N,V
ORI Rd,K	«Логическое ИЛИ» РОН и константы	Rd = Rd v К	1	Z,N,V
COM Rd	Перевод в обратный код	Rd = $FF-Rd	1	Z,C,N,V
NEG Rd	Перевод в дополнительный код	Rd = $00-Rd	1	Z,C,N,V,H
CLR Rd	Сброс всех разрядов РОН	Rd=Rd®Rd	1	Z,N,V
SER Rd	Установка всех разрядов РОН	Rd =$FF	1	
TST Rd	Проверка РОН на отрицательное или нулевое значение	Rd • Rd	1	Z,N,V
SWAP Rd	Обмен местами тетрад в РОН	Rd(3...O) = Rd(7...4), Rd(7...4) = Rd(3...O)	1	

Группа команд арифметических операций

Мнемоника	Описание	Операция	Циклы	Флаги
ADD Rd.Rr	Сложение двух РОН	Rd = Rd + Rr	1	Z,C,N,V,H
DC Rd.Rr	Сложение двух РОН с переносом	Rd = Rd + Rr + C	1	Z,C,N,V,H

ADIW Rd,K	Сложение регистровой пары с константой	Rdh:RdI = Rdh:Rdl + K	2	Z,C,N,V,S
SUB Rd,Rr	Вычитание двух РОН	Rd = Rd-Rr	1	Z,C,N,V,H
SUBI Rd,K	Вычитание константы из РОН	Rd = Rd-K	1	Z,C,N,V,H
SBC Rd,Rr	Вычитание двух РОН с заемом	Rd=Rd-Rr-C	1	Z,C,N,V,H
SBCI Rd,K	Вычитание константы из РОН с заемом	Rd = Rd-K-C	1	Z,C,N,V,H
SBIW Rd,K	Вычитание константы из регистровой пары	Rdh:Rdl = Rdh:Rdt-K	2	Z,C,N,V,S
DEC Rd	Декремент РОН	Rd = Rd - l	1	Z,N,V
INC Rd	Инкремент РОН	Rd = Rd + 1	1	Z,N,V
ASR Rd	Арифметический сдвиг вправо	Rd(n) = Rd(n+ 1), n = 0...6	1	Z,C,N,V
LSL Rd	Логический сдвиг влево	Rd(n+l) = Rd(n), Rd(0) = 0	1	Z,C,N,V
LSR Rd	Логический сдвиг вправо	Rd(n) = Rd(n+l),Rd(7) = 0	1	Z,C,N,V
ROL Rd	Сдвиг влево через перенос	Rd(0) = C, Rd(n+l) = Rd(n), C = Rd(7)	1	Z,C,N,V
ROR Rd	Сдвиг вправо через перенос	Rd(7) = C, Rd(n) = Rd(n+l), C = Rd(0)	1	Z,C,N,V
MUL Rd,Rr	Умножение без знаковых чисел	Rl:RO = RdXRr	2	Z,C
MULS Rd,Rr	Умножение чисел со знаком	Rl:RO = RdXRr	2	Z,C

MULSU Rd,Rr	Умножение без знакового числа на число со знаком	Rl:RO = RdXRr	2	Z,C
FMDL Rd,Rr	Умножение дробных без знаковых чисел	Rl:RO = (RdXRr)«l	2	Z,C
FMULS Rd,Rr	Умножение дробных чисел со знаком	Rl:RO = (RdXRr)«l	2	Z,C
FMULSU Rd,Rr	Умножение дробного без знакового числа и дробного числа со знаком	Rl:RO = (RdXRr)«1	2	Z,C

Группа команд операций с битами

Мнемоника	Описание	Операция	Циклы	Флаги
CBR Rd,K	Сброс разряда(ов) РОН	Rd = Rd»($FF-K)	1	Z,N,V
SBR Rd,K	Установка разрада(ов) РОН	Rd=RdvK	1	Z, N,V
CBI A,b	Сброс разряда РВВ	A.b = 0	2	
SBI A,b	Установка разряда РВВ	A.b=l	2	
BCLR s	Зброс флага	SREG.S - 0	1	SREG.S
BSET s	Установка флага	SREG.S=l	1	SREG.S
BLD Rd,b	Загрузка разряда РОН из флага Т (SREG)	Rd.b = T	1	
BST Rr,b	Запись разряда РОН в флаг Т (SREG)	T = Rr.b	1	T
CLC	Сброс флага переноса	C = 0	1	C
SEC	Установка флага переноса	C=l	1	C

CLN	Сброс флага отрицательного числа	N = 0	1	N
SEN	Установка флага отрицательного числа	N = 1	1	N
CLZ	Сброс флага нуля	Z = 0	1	Z
SEZ	Установка флага нуля	Z = l	1	Z
CLI	Общее запрещение прерываний	1 = 0	1	I
SEI	Общее разрешение прерываний	1 = 1	1	I
CLS	Сброс флага знака	S = 0	1	S
SES	Установка флага знака	S = l	1	S
CLV	Сброс флага переполнения доп. кода	V = 0	1	V
SEV	Установка флага переполнения доп. кода	V = l	1	V
CLT	Сброс флага T	T = 0	1	T
SET	Установка флага T	T = l	1	T
CLH	Сброс флага половинного переноса	H = 0	1	H
SEH	Установка флага половинного переноса	H = l	1	H

Группа команд пересылки данных

Мнемоника	Описание	Операция	Циклы	Флаги
MOV Rd,Rr	Пересылка между РОН	Rd = Rr	1	
MOVW Rd, Rr	Пересылка двухбайтовых значений	Rd + l:Rd = Rr+l:Rr	1	

LDI Rd, K	Загрузка константы в РОН	Rd = K	1	
LD Rd,X	Косвенное чтение	Rd=[X]	2	
LD Rd,X+	Косвенное чтение с постинкрементом	Rd = [X],X = X+I	2	
LD Rd,-X	Косвенное чтение с преддекрементом	X = X-l,Rd = [X]	2	
LD Rd,Y	Косвенное чтение	Rd=[Y]	2	
LD Rd,Y+	Косвенное чтение с постинкрементом	Rd = [Y],Y = Y+l	2	
LD Rd,-Y	Косвенное чтение с преддекрементом	Y=Y-l,Rd=[Y]	2	
LDD Rd,Y+q	Косвенное относительное чтение	Rd = [Y+q]	2	
LD Rd,Z	Косвенное чтение	Rd = [Z]	2	
LD Rd,Z+	Косвенное чтение с постинкрементом	Rd = [Z],Z = Z+l	2	
LD Rd,-Z	Косвенное чтение с преддекрементом	Z = Z - 1, Rd = [Z]	2	
LDD Rd,Z+q	Косвенное относительное чтение	Rd = [Z + q]	2	
LDS Rd,k	Непосредственное чтение из ОЗУ	Rd = [k]	2	
ST X,Rr	Косвенная запись	[X] = Rr	2	
ST X+,Rr	Косвенная запись с постинкрементом	[X] = Rr, X=X+1	2	
ST -X,Rr	Косвенная запись с преддекрементом	X = X-l, [X] = Rr	2	
ST Y,Rr	Косвенная запись	[Y] = Rr	2	
ST Y+,Rr	Косвенная запись с постинкрементом	[Y] = Rr, Y=Y+1	2	
ST –Y, Rr	Косвенная запись с преддекрементом	Y=Y-l, [X] = Rr	2	

STD Y+q,Rr	Косвенная относительная запись	[Y+q] = Rr	2	
ST Z,Rr	Косвенная запись	[Z] = Rr	2	
ST Z+,Rr	Косвенная запись с постинкрементом	[Z] = Rr,Z = Z+l	2	
ST –Z,Rr	Косвенная запись с преддекрементом	Z = Z-l, [Z] = Rr	2	
STD Z+q,Rr	Косвенная запись	[Z + q] = Rr	2	
STS k,Rr	Непосредственная запись в ОЗУ	[k] = Rr	2	
LPM	Загрузка данных из памяти программ	R0 = {Z}	3	
LPM Rd,Z	Загрузка данных из памяти программ	RB={Z}	3	
LPM Rd,Z+	Загрузка данных из памяти программ с постинкрементном	RB = {Z},Z = Z+1	3	
ELPM	Расширенная загрузка данных из памяти программ	RO = {RAMPZ:Z}	3	
ELPM Rd,Z	Расширенная загрузка данных из памяти программ	RB={RAMPZ:Z}	3	
ELPM Rd,Z+	Расширенная загрузка данных из памяти программ с постинкрементном	RB = {RAMPZ:Z}, RAMPZ:Z = RAMPZ:Z+1	3	
SPM	Запись в память программ	{Z} = RI:R0		
IN Rd,A	Пересылка из РВВ в РОН	Rd = A	1	
OUT A,Rr	Пересылка из РОН в РВВ	A=Rr	1	
PUSH Rr	Сохранение байта в стеке	STACK - Rr	2	

POP Rd	Извлечение байта из стека	Rd = STACK	2	

Группа команд передачи управления

Мнемоника	Описание	Операция	Циклы	Флаги
RJMP k	Относительный безусловный переход	PC =PC + к+1	2	
IJMP	Косвенный безусловный переход	PC =Z	2	
JMP к	Переход	PC = к	3	
RCALL к	Относительный вызов подпрограммы	PC = PC + к + 1	3	
ICALL	Косвенный вызов подпрограммы	PC = Z	3	
CALL к	Абсолютный вызов подпрограммы	PC = к	4	
RET	Возврат из подпрограммы	PC = STACK	4	
RETI	Возврат из подпрограммы обработки прерывания	PC = STACK	4	
CP Rd,Rr	Сравнение РОН	Rd-Rr	1	Z,N,V,C,H
CPC Rd, Rr	Сравнение РОН с учетом переноса	Rd-Rr-C	1	Z,N,V,C,H
CPI Rd,K	Сравнение РОН с константой	Rd-K	1	Z,N,V,C,H
CPSE Rd,Rr	Сравнение и пропуск следующей команды при равенстве	Если Rd = Rr, то PC = PC + 2 (3)	1/2/3	

SBRC Rr,b	Пропуск следующей команды, если разряд РОН сброшен	Если Rr.b = 0, то PC = PC + 2 (3)	1/2/3	
SBRS Rr,b	Пропуск следующей команды, если разряд РОН установлен	Если Rr.b=l, то PC = PC + 2 (3)	1/2/3	
SBIC A,b	Пропуск следующей команды, если РВВ сброшен	Если A.b = 0, то PC = PC + 2 (3)	1/2/3	
SBIS A,b	Пропуск следующей команды, если разряд РВВ установлен	Если А.Ь= 1, то PC = PC + 2 (3)	1/2/3	
BRBC s,k	Переход, если флаг s регистра SREG сброшен	Если SREG.s = 0, то PC = PC + к+1	1/2	
BRBS s,k	Переход, если флаг s регистра SREG установлен	Если SREG.s =1, то PC = PC + к+1	1/2	
BRCS к	Переход по переносу	Если С =1, то PC = PC + к+1	1/2	
BRCC к	Переход, если нет переноса	Если С = 0, то PC = PC + к+1	1/2	
BREQ к	Переход по «равно»	Если Z = l, то PC = PC + к+1	1/2	
BRNE к	Переход по «не равно»	Если Z = 0, то PC = PC + к+1	1/2	
BRSH к	Переход по «выше или равно»	Если С = 0, то PC = PC + к+1	1/2	
BRLO к	Переход по «меньше»	Если С =1, то PC = PC + к+1	1/2	

BRMI .	Переход по «отрицательное значение»	ЕслиN = 1, то PC = PC + к +1	1/2	
BRPL	Переход по «положительное значение»	Если N = 0, то PC = PC + к+1	1/2	
BRGE	Переход по «больше или равно» (числа со знаком)	Если (N®V) = 0, то PC = PC + к + 1	1/2	
BRLT	Переход по «меньше нуля» (числа со знаком)	Если (N®V)=l, то PC = PC + к + 1	1/2	
BRHS	Переход по половинному переносу	Если H = 1, то PC = PC + к + 1	1/2	
BRHC	Переход, если нет половинного переноса	Если H = 0, то PC = PC + к + 1	1/2	
BRTS	Переход, если флаг T установлен	Если T= 1, то PC = PC + к+1	1/2	
BRTC	Переход, если флаг T сброшен	ЕслиT = 0, тоPC = PC + к+1	1/2	
BRVS	Переход по переполнению доп. кода	ЕслиУ=1, тоPC = PC + к+1	1/2	
BRVC	Переход, если нет переполнения доп. кода	ЕслиУ = 0, тоPC = PC + к+1	1/2	
BRID	Переход, если прерывания запрещены	Если I = 0, тоPC = PC + к+1	1/2	
BRIE	Переход, если прерывания разрешены	Если1 = 1, тоPC = PC + к+1	1/2	

Группа команд управления системой

Мнемоника	Описание	Операция	Циклы	Флаги
NOP	Нет операции	пустая команда	1	
SLEEP	Переход в «спящий» режим	перевод микроконтроллера в режим пониженного энергопотребления	3	
WDR	Сброс сторожевого таймера	сброс сторожевого таймера	1	

ОГЛАВЛЕНИЕ

Учебное издание

Пирогов Александр Александрович
Буслаев Алексей Борисович

ПРОЕКТИРОВАНИЕ ЦИФРОВЫХ ФУНКЦИОНАЛЬНЫХ УЗЛОВ НА ОСНОВЕ ПРОГРАММИРУЕМЫХ ЛОГИЧЕСКИХ ИНТЕГРАЛЬНЫХ СХЕМ

Практикум

Компьютерная верстка А.А. Пирогова

Подписано к изданию 10.07.2018.
Объем данных 2,2 Мб.

ФГБОУ ВО «Воронежский государственный технический университет»
394026 Воронеж, Московский просп., 14