МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ

Государственное образовательное учреждение
высшего профессионального образования
«Оренбургский государственный университет»

Кафедра промышленной электроники и
информационно-измерительной техники

А.В. ХЛУДЕНЕВ

ЭЛЕКТРОННЫЕ ПРОМЫШЛЕННЫЕ УСТРОЙСТВА

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

Рекомендовано к изданию Редакционно-издательским советом
государственного образовательного учреждения высшего профессионального
образования «Оренбургский государственный университет»

Оренбург 2008

УДК 621.396.6
ББК 32.844-02
Х60

Рецензент
доктор технических наук, профессор В.Н. Булатов

Хлуденев, А.В.
Х60 **Электронные промышленные устройства: методические указания/ А.В.Хлуденев. – Оренбург: ИПК ГОУ ОГУ, 2008. - 50 с.**

Методические указания содержат рекомендации по выполнению лабораторных работ по дисциплине «Электронные промышленные устройства».

Методические указания предназначены для студентов, обучающихся по программам высшего профессионального образования по специальности 210106 «Промышленная электроника», а также могут быть использованы студентами других специальностей, связанных с использованием и разработкой цифровых средств управления и обработки информации.

ББК 32.844-02

Содержание

1 Исследование комбинационных схем в системе Quartus II

1.1 Описание работы с кратким изложением теории

Quartus II – инструментальная система разработки цифровых устройств и систем на базе интегральных схем программируемой логики (ПЛИС) фирмы ALTERA. Quartus II обеспечивает выполнение всех этапов, необходимых для выпуска готовых изделий:

- создание проектов устройств;
- синтез структур и трассировку внутренних связей ПЛИС;
- подготовку наборов данных для программирования или конфигурирования ПЛИС;
- верификацию проектов (функциональное моделирование и временной анализ);
- программирование или конфигурирование ПЛИС.

В состав системы Quartus II входят следующие основные компоненты:

- графический редактор схем *(Graphic Editor)*;
- символьный редактор *(Symbol Editor)* для работы с библиотеками символов элементов схем;
- редактор сигналов *(Waveform Editor)* обеспечивает ввод диаграмм тестовых входных сигналов и задание перечня тестируемых выходов;
- текстовый редактор *(Text Editor)* для создания текстовых файлов проекта на языках описания аппаратуры (HDL);
- редактор топологии *(Floorplan Editor)* позволяет на плане расположения основных логических элементов закреплять выводы ПЛИС за входными и выходными сигнальными линиями проектов и перераспределять внутренние ресурсы ПЛИС;
- компилятор *(Compiler)*, выполняющий преобразование исходного описания проекта в набор данных для программирования или конфигурирования ПЛИС;
- симулятор *(Simulator)* предназначен для функционального моделирования проекта с целью проверки правильности логики его функционирования;
- временной анализатор *Timing Analyzer* обеспечивает расчет временных задержек от каждого входа до каждого логически связанного с ним выхода;
- программатор *(Programmer)* для программирования или конфигурирования ПЛИС.

Цель работы - получить начальные практические навыки работы в среде системы Quartus II с инструментами *Graphic Editor, Waveform Editor , Simulator* при исследовании простой комбинационной схемы.

Рассмотрим работу с графическим редактором *Graphic Editor* системы Quartus II на примере схемы цифрового узла комбинационного типа, представленной на рисунке 1.1.

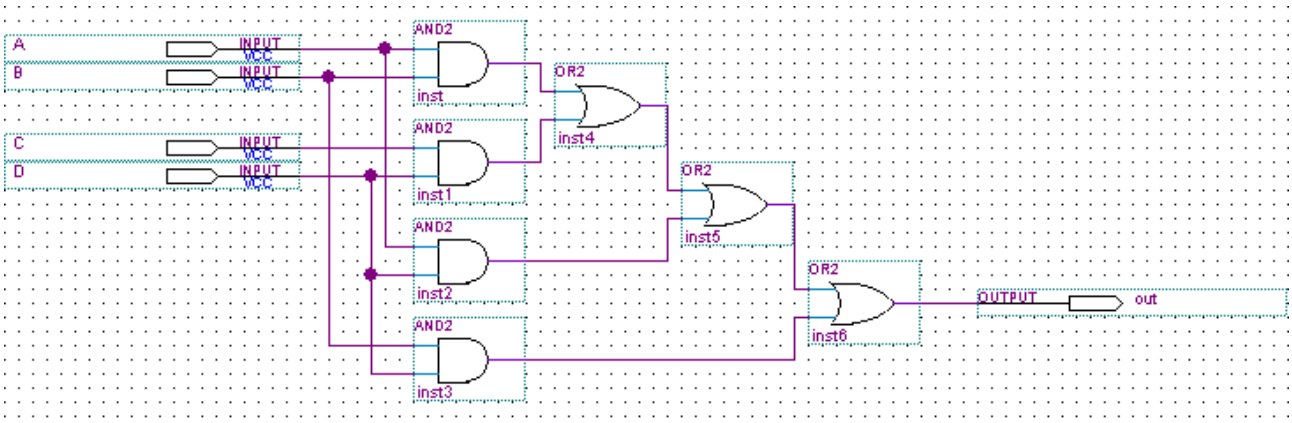


Рисунок 1.1 – Схема цифрового узла

Схема состоит из элементов и связей между ними. Условные графические обозначения (УГО) элементов содержатся в библиотеках. В левой части экрана находится меню инструментов, с помощью которых удобно строить схему. Для ввода элементов схемы используется инструмент *Symbol Tool*, который также вызывается двойным щелчком левой кнопки мыши в окне графического редактора. В открывшемся окне *Symbol* необходимо выбрать библиотеку логических элементов *Libraries>Primitives>Logic*, выбрать нужный элемент and2 и нажать ОК. В окне графического редактора выбранный элемент следует разместить в необходимых позициях нажатием левой кнопки мыши. Прекратить размещение элементов выбранного типа можно нажатием правой кнопки мыши и выбором команды *Cansel*. Аналогично выполняется размещение элементов другого типа (or2). Кроме логических элементов схема содержит входные (*input*) и выходной (*output*) выводы, которые содержатся в библиотеке *Libraries>Primitives>Pin*. Размещаются выводы схемы аналогично элементам. Каждый вывод должен иметь имя. При размещении выводам автоматически присваиваются имена pin_name, pin_name2, …., которые удобнее заменить именами, отражающими функциональное назначение вывода. Редактирование имен выводов схемы можно выполнить двойным щелчком левой кнопки мыши.

Связи между выводами элементов можно выполнить с помощью инструмента *Orthogonal Node Tool*. Начало линии связи формируется при нажатии на левую кнопку мыши, окончание линии связи – при ее отпускании (при перемещении мыши кнопка должна быть нажата). Если необходимо построить линию связи со сложной топологией, то ее можно представить как систему, состоящую из ортогональных отрезков, каждый из которых строится вышеуказанным образом. При этом в месте Т-образных соединений автоматически устанавливаются точки соединения.

В цифровых узлах комбинационного типа значения выходных сигналов в любой момент времени однозначно определяются текущими значениями входных сигналов.

Исследовать работу схемы можно с помощью встроенного в систему Quartus II инструмента моделирования – симулятора. Выполняется имитационное моделирование работы схемы, в ходе которого рассчитываются значения

выходных сигналов для заданных значений входных сигналов. Для выполнения моделирования необходимо создать тест, включающий последовательность двоичных наборов входных сигналов. Так как комбинационная схема имеет четыре входных сигнала, то всего на вход схемы могут быть поданы 2^4=16 различных наборов:

```
A:0101010101010101
B:0011001100110011
C:0000111100001111
D:0000000011111111
```

Рассчитанные симулятором системы значения выходного сигнала представляются в окне сигнального редактора в форме временной диаграммы. Сравнивая полученные значения выходного сигнала с ожидаемыми, можно сделать вывод о корректности сформированной схемы.

1.3 Задание к выполнению лабораторной работы

1.3.1 Ввести заданную схему в графическом редакторе Quartus II .

1.3.2 Сформировать функциональный тест.

1.3.3 Произвести проверку работы схемы. Зарисовать временные диаграммы сигналов и заполнить таблицу истинности схемы.

1.3.4 Ответить на контрольные вопросы, оформить отчет о выполненной работе.

1.4 Порядок выполнения работы

1.4.1 Запустить на выполнение Quartus II, выбрать из меню *File* пункт *New Project Wizard…* - мастер создания новых проектов. В открывшемся окне нажать кнопку Next и перейти в окно для задания текущей директории проекта. Заполнить три строки:

C:/altera/qdesigns/lr3

Lr3

Lr3

нажать кнопку *Finish* и подтвердить создание проекта. Текущий проект будет назван Lr3 и будет создана директория по адресу C:/altera/qdesigns/Lr3.

1.4.2 При помощи меню *File/New…* на закладке *Device Design File* выбрать *Block Diagram/Schematic File* и нажать ОК. В открывшемся окне графического редактора *Graphic Editor* ввести заданную схему.

1.4.3 После окончания ввода сохранить схему в файле Lr3.bdf (расширение присваивается автоматически) через меню *File/Save As…*

1.4.4 Открыть сигнальный редактор и создать в нем файл с расширением .scf. При помощи меню *Edit/End Time…* задать время моделирования 80 нс, а в меню *Edit/Grid Size…* - шаг сетки моделирования для периода 10 нс. Далее в

окне сигнального редактора двойным щелчком правой кнопки мыши на поле *Name:* вызвать меню *Insert Node or Bus*, при помощи которого задать очередной вход схемы. Для входов схемы задать значения сигналов на протяжении необходимого времени моделирования – 80 нс. В данном случае для формирования значений сигналов рекомендуется воспользоваться инструментом *OverWrite Clock*, который активируется после выбора сигнала в поле *Name*. Используя инструмент *OverWrite Clock*, задайте последовательно для сигналов A, B, C, D значения периода формируемой импульсной последовательности соответственно 10, 20, 40, 80 нс. В окне редактора входные сигналы представляются в форме временных диаграмм. Убедитесь, что сформированные сигналы полностью соответствуют замыслу модельного эксперимента (сформированы необходимые 16 двоичных наборов). В противном случае необходимо выполнить редактирование теста. После этого сохранить тест в файле Lr3.vwf (расширение присваивается автоматически) через меню *File/Save As…*

1.4.5 Для выполнения моделирования вызвать инструмент *Simulator Tool*. В окне инструмента в поле *Simulation Mode* выбрать режим функционального моделирования *Functional* (без учета задержек). Для создания модели схемы нажать кнопку *Generate Functional Simulation Netlist*. В поле *Simulation Input* выбрать и задать имя созданного файла тестовых сигналов, после чего нажать кнопку *Start*. После завершения расчетов нажатием кнопки *Open* открыть окно сигнального редактора, в котором будут отражены временные диаграммы входных и выходного сигналов схемы.

1.4.6 По результатам моделирования необходимо сформировать таблицу истинности, которая описывает функционирование схемы. В этой таблице каждому входному набору необходимо сопоставить значение выходного сигнала.

1.4 Содержание отчета

Отчет должен содержать:
- схему цифрового узла;
- временные диаграммы сигналов;
- таблицу истинности, выводы.

1.5 Контрольные вопросы

1.5.1 Каким образом в системе Quartus II выполняется ввод описаний цифровых устройств в схемном виде?

1.5.2 Каким образом в системе Quartus II выполняется моделирование цифровых устройств?

1.5.3 Объясните полученные результаты моделирования работы схемы. Как они могут быть использованы?

2 Исследование комбинационных схем на стенде SDK6.1

2.1 Описание работы с кратким изложением теории

Цель работы - исследовать работу простой комбинационной схемы на учебном стенде SDK6.1.

Учебный лабораторный комплекс SDK-6.1 предназначен для обучения основам проектирования современных электронных модулей на ПЛИС. В SDK-6.1 использованы ПЛИС типа EP1C3T144-8 семейства CYCLONE фирмы ALTERA. Путем конфигурирования ПЛИС можно создавать и затем экспериментально исследовать разнообразные цифровые устройства.

Стенд оснащен разнообразными периферийными модулями и интерфейсными каналами. Элементы управления общего назначения стенда включают:

- «SW0»-«SW7» движковые переключатели;
- «D0»-«D7» управляемые светодиоды;
- «C» управляемый светодиод;
- «CONTROL» кнопка тактовая;
- IND1 жидкокристаллический индикатор (ЖКИ).

Системные элементы управления стенда включают:

- «RESET» кнопка полного рестарта системы;
- «PAGE» переключатели номера конфигурации ПЛИС;
- «CONFIG» индикатор состояния конфигурации ПЛИС;
- «POWER» индикатор питания.

На рисунках 2.1 и 2.2 приведены схемы включения на стенде движковых переключателей «SW0»-«SW7» и управляемых светодиодов.

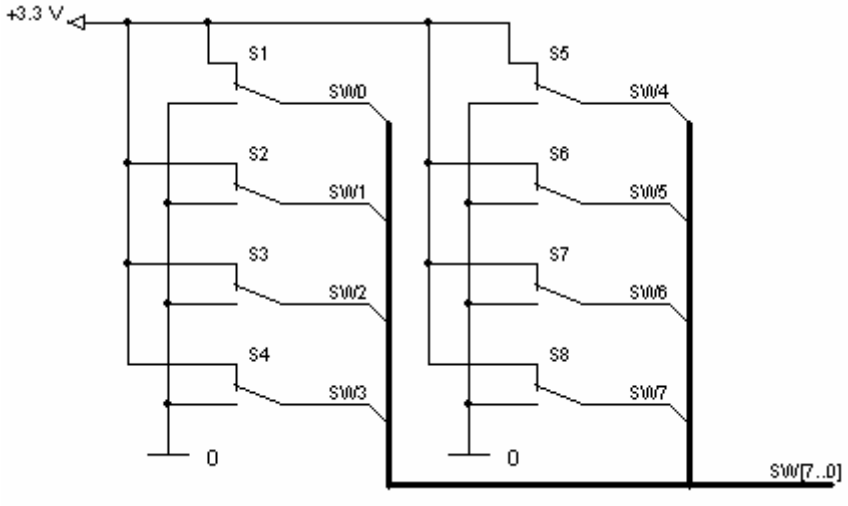


Рисунок 2.1

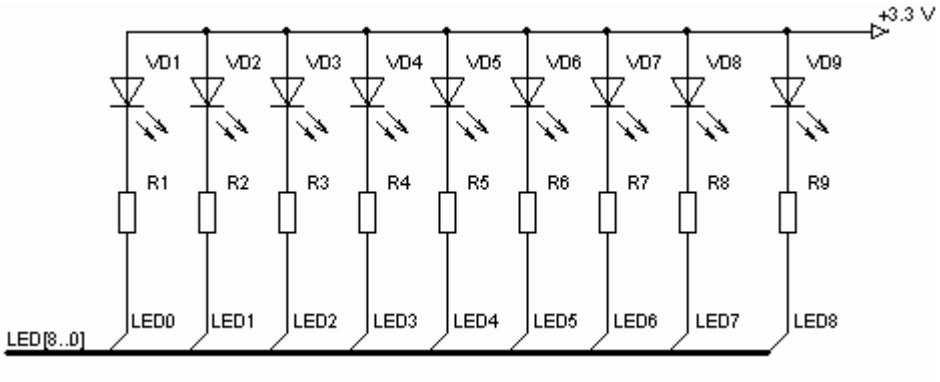


Рисунок 2.2

Рассмотрим работу со стендом SDK-6.1 на примере схемы цифрового узла комбинационного типа (рисунок 1.1).

Примем, что для исследования схемы на стенде SDK-6.1 к ее входам будут подключены движковые переключатели «SW0»-«SW3», а выходной сигнал будет управлять светодиодом «С» (сигнальная линия LED8). Для управления светодиодом, который включен по схеме на рисунке 2.2, необходим буферный элемент с открытым стоком (Opndrn). При этом светодиод будет излучать, когда на выходе буфера Opndrn будет низкий потенциал логического нуля. Будем полагать, что светодиод должен излучать, когда на выходе исследуемой схемы комбинационного узла формируется логическая единица. Тогда между выходом схемы и буфером Opndrn необходимо включить логический инвертор (not).

2.2 Задание к выполнению лабораторной работы

2.2.1 Выполнить редактирование схемы в среде графического редактора системы Quartus II.

2.2.2 Выполнить компиляцию проекта и сформировать конфигурационный файл.

2.2.3 Выполнить конфигурирование ПЛИС.

2.2.4 Выполнить проверку работы комбинационного узла на стенде SDK6.1, заполнить таблицу истинности.

2.2.5 Ответить на контрольные вопросы, оформить отчет о выполненной работе.

2.3 Порядок выполнения работы

2.3.1 Запустите на выполнение систему Quartus II, выберите из меню *File* пункт *Open Project* и откройте проект Lr1.qpf. При этом открывается окно графического редактора с сохраненной схемой цифрового узла комбинационного типа. Выполните редактирование схемы узла – включите между выходным элементом типа or2 и выводом output логический инвертор not и буфер Opndrn из библиотеки *Libraries>Primitives>Buffers*.

2.3.2 После окончания редактирования схемы сохраните схему в файле Lr2.bdf (расширение присваивается автоматически) через меню *File/Save As…* в директории C:\altera\qdesigns\Lr2 и закройте проект Lr1.bdf (меню *File/Close Project*).

2.3.3 Выберите из меню *File* пункт *New Project Wizard…* - мастер создания новых проектов. В открывшемся окне нажмите кнопку *Next* и перейдите в окно для задания текущей директории проекта. Заполните три строки:

C:\altera\qdesigns\lr2

Lr2

Lr2

нажмите кнопку *Finish* и подтвердите создание проекта. Текущий проект будет назван Lr2 в директории по адресу C:\altera\qdesigns\Lr2.

2.3.4 Выполните анализ схемы и логический синтез проекта из меню *Processing/Start/Start Analysis & Synthesis*.

2.3.5 Через меню *Assignment/Device* активируйте окно назначения типа ПЛИС *Settings*. В окне *Family* выберите семейство Cyclone и в окне *Available devices* тип ПЛИС EP1C3T144-8. Далее необходимо нажать кнопку *Device & Pin Options*, в открывшемся окне выбрать вкладку *Programming Files* и установить метку на пункте *Raw Binary File (.rbf)*, затем выбрать вкладку *Unused Pins* и выбрать в группе переключателя *Reserve all unused pins* позицию *As inputs, tri-stated*. Тем самым задается формат файла конфигурации ПЛИС и режим включения ее неиспользуемых в проекте выводов. **Невыполнение или неправильное выполнение последнего требования может привести в выходу из строя ПЛИС и периферийных модулей стенда!** Закройте окна *Device & Pin Options u Settings*.

2.3.6 Через меню *Assignment/Pins* активируйте окно назначения используемых в проекте выводов ПЛИС. В окне *Assignment Editor* содержится электронная таблица. В ее столбце *To* выберите в каждой строке имена выводов на схеме, а в столбце *Location* - необходимые выводы ПЛИС в соответствии со схемой включения ПЛИС на стенде. Необходимые для данной работы данные приведены в таблице 2.1. Выбор имен выполняется из списка, который активируется после двойного нажатия левой кнопки мыши по соответствующей ячейке электронной таблицы.

Таблица 2.1

Вывод исследуемой схемы	Сигнальная линия стенда	Вывод ПЛИС
A	SW0	Pin_112
B	SW1	Pin_120
C	SW2	Pin_113
D	SW3	Pin_121
out	LED8	Pin_103

2.3.7 Для запуска процесса компиляции выберите пункт *Start Compilation* из меню *Processing*. В процессе компиляции в нижней части экрана в окне сообщений на вкладке *Processing* выводится информация о результатах компиляции и количестве ошибок и предупреждений. Если сообщения об ошибках (Error) отсутствуют, то файл конфигурации сформирован.

По результатам компиляции система Quartus II формирует отчеты, с которыми можно ознакомиться через меню *Processing/Compilation Report*. Левое поле окна *Compilation Report* содержит список отчетов, а правое поле – их содержимое. Отчет *Flow Summary* содержит общие сведения о проекте и использованных ресурсах ПЛИС. Отчет *tpd* (time propogation delay) в папке *Timing Analyzer* содержит значения задержек распространения сигналов от входов к выходу схемы.

2.3.8 Подключите разъем J7 стенда SDK-6.1 к COM-порту ПК при помощи интерфейсного кабеля, ***только затем*** подключите к SDK-6.1 блок питания.

2.3.9 Переключите стенд в режим загрузки конфигурационных файлов для EP1C3 по последовательному каналу. Для этого нажмите кнопку «RESET» при нажатой и удерживаемой кнопке «CONTROL» (это приведет к тому, что будет загружен файл из нулевой страницы, независимо от того, в каком положении находится переключатель «PAGE»). После этого на ЖКИ стенда будет выведено сообщение «SDK6.1 Flash Writer», светодиоды «D0»-«D7» будут по очереди зажигаться и гаснуть, а светодиод «CONFIG» будет погашен.

2.3.10 Запустите на Host-компьютере программатор sdk61fw для того, чтобы произвести запись конфигурационного файла во flash-память стенда. Чтобы записать файл Lr2.rbf в страницу номер 2 через com-порт 1, нужно вызвать программатор со следующими параметрами:

sdk61fw write page2 Lr2.rbf com1

где: sdk61fw - имя программы;
write – режим записи во flash-память;
page1 - номер страницы во flash-памяти стенда;
Lr2.rbf – имя записываемого конфигурационного файла;
com1 – имя последовательного порта.

2.3.11 После того, как программатор завершит работу, путем нажатия кнопки «RESET» произведите рестарт системы, при этом в ПЛИС EP1C3 будет загружен конфигурационный файл из той страницы flash-памяти, номер которой указан на переключателе «PAGE» в соответствии с таблицей 2.2 (нижнее положение переключателя - 1, верхнее положение - 0).

2.3.12 После конфигурирования ПЛИС должна выполнять функцию заданного цифрового устройства – в данном случае узла комбинационного типа. Исследовать работу комбинационного узла можно путем снятия таблицы истинности. Для этого необходимо последовательно устанавливать комбинации положений переключателей «SW0»-«SW3» и фиксировать состояние светодиода

«C» (нижнее положение переключателя - 0, верхнее положение - 1). Для полной проверки необходимо использовать наборы:

«SW0» (A):0101010101010101
«SW1» (B):0011001100110011
«SW2» (C):0000111100001111
«SW3» (D):0000000011111111

Таблица 2.2

Номер страницы	Положение «Page» 1234	Адрес Страницы Flash-памяти	Номер страницы	Положение «Page» 1234	Адрес страницы Flash-памяти
0	0000	0x000000	8	0001	0x100000
1	1000	0x020000	9	1001	0x120000
2	0100	0x040000	10	0101	0x140000
3	1100	0x060000	11	1101	0x160000
4	0010	0x080000	12	0011	0x180000
5	1010	0x0A0000	13	1011	0x1A0000
6	0110	0x0C0000	14	0111	0x1C0000
7	1110	0x0E0000	15	1111	0x1E0000

2.4 Содержание отчета

Отчет должен содержать:
- значения задержек распространения сигналов.
- таблицу истинности;
- выводы.

2.5 Контрольные вопросы

2.5.1 Каким образом в системе Quartus II выполняется формирование конфигурационного файла для ПЛИС?

2.5.2 Каким образом выполняется конфигурирование ПЛИС на стенде SDK6.1?

2.5.3 Какие параметры характеризуют быстродействие устройств комбинационного типа? Поясните, как они определяются.

3 Исследование преобразователя кода

3.1 Описание работы с кратким изложением теории

Цель работы – экспериментально исследовать работу преобразователя кода символов на учебном стенде SDK6.1.

Двоичное кодирование используется в цифровых устройствах для представления не только чисел, но и символьной информации. При использовании кода ASCII можно представлять цифры, прописные и строчные буквы латинского алфавита, другие символы. Расширенный восьмиразрядный код ASCII позволяет также представлять буквы национальных алфавитов. В стенде SDK6.1 установлен двухстрочный жидко-кристаллический индикатор (ЖКИ) типа PC1602. Каждая строка содержит 16 знакомест. Набор отображаемых символов и их восьмиразрядные ASCII коды приведены на рисунке 3.1.

Для многих прикладных задач бывает достаточно отображать ограниченный набор символов. В этих случаях для кодирования символов обычно используют другие коды с меньшим количеством разрядов. Например, для представления символов шестнадцатеричного алфавита достаточно использовать четырехразрядный код. Тогда для отображения этих символов на ЖКИ типа PC1602 необходим преобразователь четырехразрядного Hex-кода в код ASCII.

Преобразователи кода относятся к цифровым узлам комбинационного типа. Алгоритм их функционирования можно представить таблицей истинности. В таблице 3.1 приведено описание преобразователя четырехразрядного кода символов шестнадцатеричного алфавита в восьмиразрядный код ASCII.

Таблица 3.1

Символ	Hex-код	ASCII
0	0000	00110000
1	0001	00110001
2	0010	00110010
3	0011	00110011
4	0100	00110100
5	0101	00110101
6	0110	00110110
7	0111	00110111
8	1000	00111000
9	1001	00111001
A	1010	01000001
B	1011	01000010
C	1100	01000011
D	1101	01000100
E	1110	01000101
F	1111	01000110

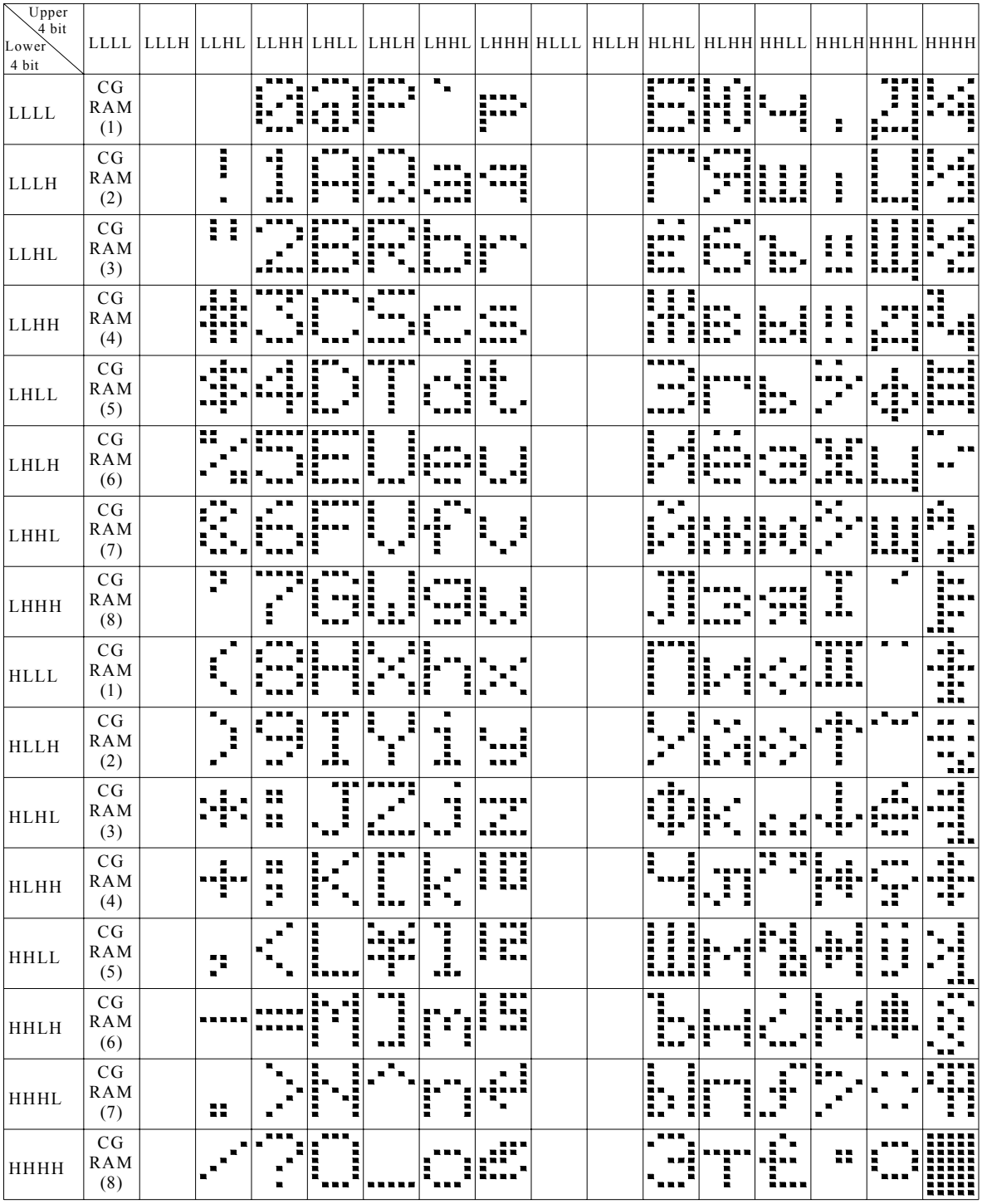


Рисунок 3.1

Задачу построения преобразователя кода на кристалле ПЛИС можно возложить на компилятор системы Quartus II, если описать его алгоритм функционирования на языке AHDL, используя средства описания таблиц истинности:

```
SUBDESIGN Hex_To_ASCII
(
Hexcod[3..0]:input;
```

```
ASCII[7..0]: output;
)
Begin
 Table
   Hexcod [] =>ASCII[];
    0  => b"00110000";
    1  => b"00110001";
    2  => b"00110010";
    3  => b"00110011";
    4  => b"00110100";
    5  => b"00110101";
    6  => b"00110110";
    7  => b"00110111";
    8  => b"00111000";
    9  => b"00111001";
   10 => b"01000001";
   11 => b"01000010";
   12 => b"01000011";
   13 => b"01000100";
   14 => b"01000101";
   15 => b"01000110";
 end Table;
End;  --.
```

Примем, что в качестве задатчика входного двоичного кода будут использоваться движковые переключатели SW[3..0] стенда SDK6.1. Отображаемый символ будет располагаться на месте младшего разряда первой строки. Для загрузки ASCII кода в ЖКИ типа PC1602 используем контроллер, реализованный в модуле Lcd_controller (файлы Lcd_controller.v и Lcd_controller.inc).

Исследование преобразователя на стенде SDK6.1 будем проводить в составе проекта, описанного в файле lr3.tdf. Данный проект описывает схему включения преобразователя кода, контроллера ЖКИ, а также функциональных узлов, обеспечивающих его правильное функционирование.

3.2 Задание к выполнению лабораторной работы

3.2.1 Сформировать AHDL-описание преобразователя кода и выполнить его моделирование в среде системы Quartus II.

3.2.2 Выполнить компиляцию проекта Lr3 и сформировать конфигурационный файл Lr3.rbf.

3.2.3 Выполнить конфигурирование ПЛИС.

3.2.4 Выполнить проверку работы преобразователя кода на стенде SDK6.1, по ее результатам сделать выводы.

3.2.5 Ответить на контрольные вопросы, оформить отчет.

3.3 Порядок выполнения работы

3.3.1 Запустите на выполнение Quartus II, выберите из меню *File* пункт *New Project Wizard…* - мастер создания новых проектов. В открывшемся окне нажмите кнопку *Next* и в окне текущей директории проекта заполните строки:

C:\altera\qdesigns\lr3

Hex_To_ASCII

Hex_To_ASCII

Нажмите кнопку *Finish* и подтвердите создание проекта. Текущий проект будет назван Hex_To_ASCII.

3.3.2 При помощи меню *File/New…* на закладке *Device Design File* выберите *AHDL File* и нажмите ОК. В открывшемся окне текстового редактора введите описание преобразователя кода. После окончания ввода сохраните текст в файле Hex_To_ASCII.tdf через меню *File/Save As…*

3.3.3 Для моделирования необходимо создать тест, включающий последовательность двоичных наборов входных сигналов из таблицы 3.1. Для этого откройте сигнальный редактор и создайте в нем файл с расширением .vwf. При помощи меню *Edit/End Time…* задайте время моделирования 160 нс, а в меню *Edit/Grid Size…* - шаг сетки моделирования для периода 10 нс. Далее в окне сигнального редактора двойным щелчком правой кнопки мыши на поле *Name* вызовите меню *Insert Node or Bus*, при помощи которого задайте имя входного вектора Hexcod в поле *Name* и число разрядов 4 в поле *Bus Width*. Для входных выводов схемы задайте значения сигналов на протяжении необходимого времени моделирования – 160 нс. В данном случае для формирования значений сигналов проще всего воспользоваться инструментом *Count Value*, который активируется после выбора сигнала в поле *Name.* Используя инструмент *Count Value,* задайте для Hexcod значения наборов 0000 … 1111 на интервале 160 нс (смена наборов через 10 нс задается значением параметра *Count Every*). В окне редактора входные сигналы представляются в форме временных диаграмм. Убедитесь, что сформированные сигналы полностью соответствуют замыслу модельного эксперимента (сформированы необходимые 16 двоичных наборов). В противном случае необходимо выполнить редактирование теста. После этого сохраните тест в файле Hex_To_ASCII.vwf (расширение присваивается автоматически) через меню *File/Save As…*

3.3.4 Для выполнения моделирования вызовите инструмент *Simulator Tool*. В окне инструмента в поле *Simulation Mode* выберите режим функционального моделирования *Functional* (без учета задержек). Для создания модели схемы нажмите кнопку *Generate Functional Simulation Netlist*. В поле *Simulation Input* выберите и задайте имя созданного файла тестовых сигналов, после чего нажмите кнопку *Start*. После завершения расчетов нажатием кнопки *Open* откройте окно сигнального редактора, в котором будут отражены временные диаграммы входных и выходного сигналов схемы. По результатам моделирования необходимо выполнить проверку правильности работы преобразователя кода путем сравнения с таблицей истинности.

3.3.5 Для подключения модуля Hex_To_ASCII.tdf к проекту Lr3 необходимо создать файл Hex_To_ASCII.inc с помощью меню *File*, пункт Create/*Update Create AHDL Include File for Current File*. После этого проект Hex_To_ASCII можно закрыть и приступить к созданию проекта верхнего уровня Lr3.

3.3.6 Выберите из меню *File* пункт *New Project Wizard…* - мастер создания новых проектов. В открывшемся окне нажмите кнопку *Next* и в окне для задания текущей директории проекта заполните три строки:

C:\altera\qdesigns\lr3

Lr3

Lr3

нажмите кнопку *Finish* и подтвердите создание проекта. Текущий проект будет назван Lr3 в директории по адресу C:\altera\qdesigns\Lr3.

3.3.7 Выполните анализ исходного описания и логический синтез проекта из меню *Processing/Start/Start Analysis & Synthesis*.

3.3.8 Через меню *Assignment/Device* активируйте окно назначения типа ПЛИС *Settings*. В окне *Family* выберите семейство Cyclone и в окне *Available devices* - тип ПЛИС EP1C3T144-8. Далее необходимо нажать кнопку *Device & Pin Options*, в открывшемся окне выбрать вкладку *Programming Files* и установить метку на пункте *Raw Binary File* (.rbf). Затем выберите вкладку *Unused Pins* и в группе переключателя *Reserve all unused pins* позицию *As inputs, tri-stated*. **Невыполнение или неправильное выполнение последнего требования может привести в выходу из строя ПЛИС и периферийных модулей стенда!** Закройте окна *Device & Pin Options и Settings*.

3.3.9 Через меню *Assignment/Pins* активируйте окно назначения используемых в проекте выводов ПЛИС. Необходимые для данной работы данные приведены в таблице 3.2.

Таблица 3.2

Вывод исследуемой схемы (сигнальная линия стенда)	Вывод ПЛИС
clk	Pin_93
SW0	Pin_112
SW1	Pin_120
SW2	Pin_113
SW3	Pin_121
E	Pin_131
RW	Pin_130
RS	Pin_132
Data_ind4	Pin_124
Data_ind5	Pin_125
Data_ind6	Pin_128
Data_ind7	Pin_129

3.3.10 Для запуска процесса компиляции выберите пункт *Start Compilation* из меню *Processing*. Если сообщения об ошибках (Error) отсутствуют, то файл конфигурации сформирован.

3.3.11 Подключите разъем J7 стенда SDK-6.1 к COM-порту ПК при помощи интерфейсного кабеля. По таблице 2.2 задайте страницу flash-памяти 3. ***Только затем*** подключите питание к стенду SDK-6.1.

3.3.12 Переключите стенд в режим загрузки конфигурационных файлов для EP1C3 по последовательному каналу. Для этого нажмите кнопку «RESET» при нажатой и удерживаемой кнопке «CONTROL». Запустите на Host-компьютере программатор sdk61fw со следующими параметрами:

sdk61fw write page3 Lr3.rbf com1

После того, как программатор завершит работу, путем нажатия кнопки «RESET» произведите рестарт системы. После конфигурирования ПЛИС должна выполнять функцию заданного цифрового устройства – в данном случае преобразователя кода.

3.3.13 Проверьте правильность работы преобразователя кода, последовательно устанавливая комбинации положений переключателей «SW0»-«SW3» и фиксируя символы, отображаемые на ЖКИ.

3.4 Содержание отчета

Отчет должен содержать:

- таблицу истинности преобразователя кода;
- AHDL-описание преобразователя кода;
- схему включения преобразователя кода при исследовании на стенде (сформировать по описанию Lr3.tdf);
- выводы по результатам исследования преобразователя кода.

3.5 Контрольные вопросы

3.5.1 В каких случаях используются преобразователи кода? Приведите примеры.

3.5.2 Каким образом можно сформировать таблицу истинности для преобразователя кода?

3.5.3 Каким образом можно описать функционирование цифровых узлов комбинационного типа на HDL? Поясните структуру AHDL-описания преобразователя кода.

3.5.4 Каким образом выполняется исследование работы преобразователя кода на стенде SDK6.1?

4 Исследование схем контроля

4.1 Описание работы с кратким изложением теории

Цель работы – экспериментально исследовать работу схемы контроля на учебном стенде SDK 6.1.

В процессе работы цифровых устройств иногда возникают ошибки. Ошибки, обусловленные неисправностями, могут быть вызваны либо выходом из строя элементов, либо отклонением от нормы параметров, например, изменением напряжения питания, или воздействием внешних помех. Вызванные этими нарушениями ошибки могут иметь постоянный или случайный характер. Постоянные ошибки (отказы) легче обнаружить и исправить. Случайные ошибки (сбои), обусловленные кратковременными изменениями параметров, наиболее опасны, и их труднее обнаружить. Эти ошибки (сбои и отказы) и являются объектом для работы системы контроля.

Система контроля - совокупность методов и средств, обеспечивающих определение правильности работы цифрового устройства в целом или отдельных его узлов, а также автоматическое исправление ошибки (если это возможно). Контроль работы цифровых устройств является одним из самых эффективных средств повышения их надёжности. Решение всех задач контроля становится возможным только при наличии определённой избыточности информации. Избыточность может быть создана либо аппаратными средствами, либо логическими или информационными средствами.

Систематический код содержит кроме информационных контрольные разряды, в которые записывается некоторая контрольная информация об исходном числе. Поэтому можно считать, что систематический код обладает избыточностью. При этом абсолютная избыточность будет выражаться количеством контрольных разрядов k, а относительная избыточность - отношением k/n, где $n = m + k$ - общее количество разрядов в кодовом слове, а m - количество информационных разрядов.

Понятие *корректирующей способности* кода обычно связывают с возможностью обнаружения и исправления ошибки. Количественно корректирующая способность кода определяется вероятностью обнаружения или исправления ошибки.

Корректирующая способность кода связана также с понятием *кодового расстояния.* Кодовое расстояние $d(A, B)$ для кодовых комбинаций A и B определяется как вес такой третьей комбинации, которая получается поразрядным сложением исходных комбинаций по модулю 2.

Вес кодовой комбинации V(A) - количество единиц, содержащихся в кодовой комбинации.

Любая позиционная система счисления отличается тем свойством, что минимальное кодовое расстояние равно 1 (рисунок 4.1,а). Систематический код

обладает способностью обнаруживать ошибки только тогда, когда минимальное кодовое расстояние для него больше или равно 2*t*, т.е.

$$d_{min} \geq 2t \ ,$$

где *t* - кратность обнаруживаемых ошибок.

Это означает, что между соседними разрешенными кодовыми словами должно существовать, по крайней мере одно кодовое слово (рисунок 4.1, б,в).

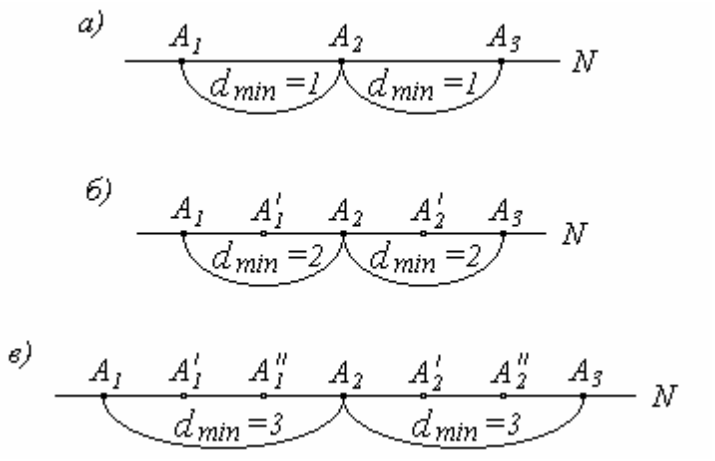


Рисунок 4.1 - Кодовые расстояния

В тех случаях, когда необходимо не только обнаружить ошибку, но и исправить её (т.е. указать место ошибки), минимальное кодовое расстояние:

$$d_{min} \geq 2t + 1 \ .$$

Систематический код, предложенный в 1949 г. американским учёным Ричардом Хэммингом, обладает способностью обнаруживать и исправлять одиночные ошибки. Рассмотрим принцип построения кода Хэмминга с минимальным кодовым расстоянием $d = 3$, т.е. способностью обнаруживать место появления любой однократной ошибки. Построение кода состоит в разбиении *m* разрядов слова на взаимно пересекающиеся подмножества, причём каждому подмножеству ставится в соответствие один контрольный разряд проверки на чётность. Количество контрольных разрядов *k* кода Хэмминга определяется из следующего соотношения:

$$2^k\text{-}1 \geq m+k > 2^{k-1}.$$

Контрольные разряды перемежаются с информационными и располагаются в позициях с номерами $N_k=2^i$, где $i = 0, 1, 2, ...$, т.е. 1, 2, 4, 8, 16 и т.д.

Формирование подмножеств производится на основе анализа номера разряда при записи его в двоичной системе счисления. Все разряды кодового слова, имеющие единицу в первом (младшем) разряде своего номера, включаются в первое подмножество, во втором - во второе и т.д. Затем подсчитывается ко-

личество единиц в разрядах, относящихся к каждому подмножеству, и в соответствующий контрольный разряд записывается единица, если это количество нечётно, и ноль - если чётно.

В таблице 4.1 показано разбиение разрядов на подмножества для 7-разрядного кодового слова, состоящего из 4 информационных и 3 контрольных разрядов. Пример кодирования 7-разрядного слова представлен на рисунке 4.2.

Таблица 4.1

Номера разрядов кодового слова		Подмножества		
десятичный	двоичный	3	2	1
1	001	*		*
2	010		*	
3	011	*	*	*
4	100	*		
5	101	*		*
6	110	*	*	
7	111	*	*	*

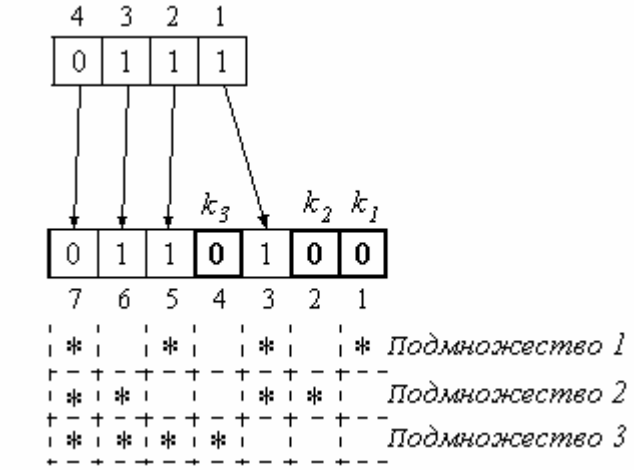


Рисунок 4.2 - Кодирование информационного слова по Хэммингу

При декодировании (процесс обнаружения ошибки) вычисляется *синдром ошибки* $S = s_{k-1} \ldots s_2 s_0$, по которому можно судить о наличии ошибки и её месте. Число разрядов в синдроме ошибки равно числу контрольных разрядов кода Хэмминга для данного слова. В рассматриваемом случае будем иметь трехразрядный синдром ошибки $S = s_2 s_1 s_0$. Для вычисления разрядов синдрома ошибки s_{i-1} нужно сложить по модулю 2 разряды, относящиеся к i-му подмножеству, включая i-й контрольный разряд кода Хэмминга. Если все разряды синдрома ошибки равны нулю $S = 0\ 0\ 0$, то это свидетельствует о том, что в закодированном слове ошибки отсутствуют. Если синдром не равен нулю, то в слове имеется ошибка, а значение синдрома указывает на ошибочный разряд.

Например, ошибка произошла в пятом справа разряде закодированного слова, причём первым считается крайний правый разряд. Это приведёт к нарушению чётности единиц в первом и третьем подмножествах. Следовательно, разряды синдрома ошибки будут равны: $s_3 = 1$, $s_2 = 0$, $s_1 = 1$. Синдром ошибки $S = 101$ указывает на пятый разряд. Для исправления ошибки достаточно инвертировать значение указанного разряда.

Однако если в закодированном слове будет присутствовать ошибка кратности 2 или более, синдром ошибки укажет на разряд, не содержащий ошибки.

Легко построить модифицированный код Хэмминга с минимальным кодовым расстоянием $d = 4$ для определения места одиночной и факта двойной ошибки. Для этого нужно ввести в разрядную сетку ещё один дополнительный разряд для проверки на чётность общего количества единиц в закодированном слове. Схема кодирования и декодирования для модифицированных кодов Хэмминга приведена на рисунке 4.3.

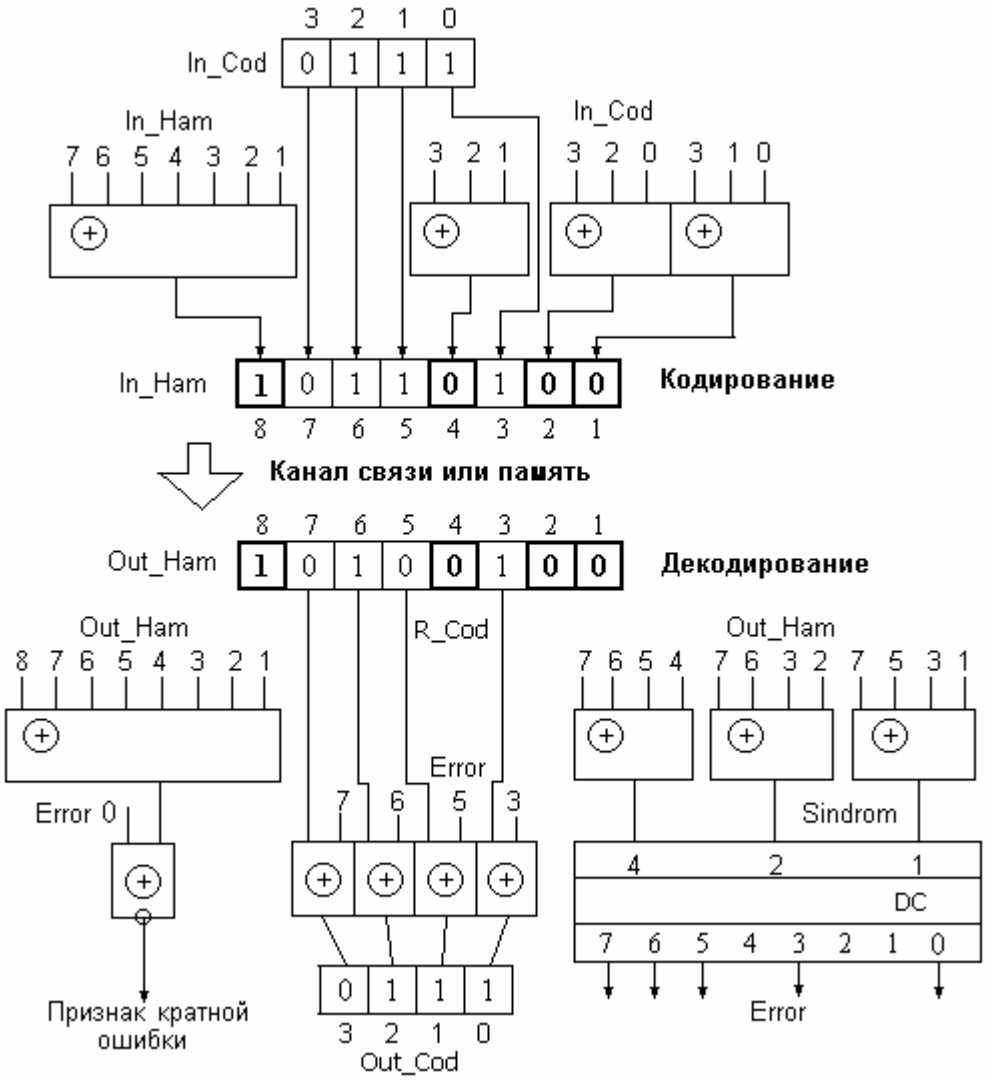


Рисунок 4.3 – Схема кодирования и декодирования для кодов Хэмминга

Если произойдёт одиночная ошибка, то её выявит проверка на чётность всего слова, а место ошибки определится синдромом ошибки. При искажении

содержимого двух разрядов общая проверка на чётность ошибку не зафиксирует, но синдром ошибки будет отличен от нуля. Это указывает на то, что ошибка есть, хотя определить её место становится уже невозможным. Ошибки кратности 3, 5, 7, ... будут восприниматься как одиночная ошибка. Однако ввиду того, что наиболее вероятными являются одиночные ошибки (с увеличением кратности ошибки на единицу вероятность её появления уменьшается примерно на порядок), данный недостаток кода можно считать несущественным.

В таблице 4.2 показаны варианты решений, принимаемых при декодировании модифицированного кода Хэмминга.

Таблица 4.2

Общая чётность	Синдром ошибки	Решение
= 0	= 0	Ошибки нет
= 0	≠ 0	Неисправимая ошибка кратности 2, 4, 6, 8, ...
≠ 0	= 0	Неисправимая пакетная ошибка кратности 2^i-1, где i = 2, 3, 4, ... , располагающаяся в 2^i-1 младших разрядах закодированного слова
≠ 0	≠ 0	Ошибка кратности 1, 3, 5, 7, ... Наиболее вероятна одиночная ошибка – исправлена

По методу Хэмминга могут быть построены коды различной длины. При этом, чем больше длина кода, тем меньше относительная избыточность. Коды Хэмминга используют в основном для контроля полупроводниковых запоминающих устройств и контроля передачи информации по каналам связи. При контроле запоминающих устройств их надёжность увеличивается в 60 - 85 раз.

По схеме (рисунок 4.2) реализован проект на языке AHDL. В качестве задатчика входного 4-хразрядного двоичного кода In_Cod[3..0] использованы движковые переключатели SW[7..4] стенда SDK6.1.

Контрольные разряды входного слова In_Ham формируются суммированием по модулю 2 соответствующих разрядов входного кода In_Cod[3..0]. Дополнительный контрольный разряд 8 формируется суммированием по модулю 2 разрядов слова In_Ham[7..1].

В качестве задатчика информационных разрядов принимаемого двоичного кода R_Cod[3..0] использованы движковые переключатели SW[3..0]. Появление ошибки имитируется несовпадением кодовых комбинаций, задаваемых переключателями SW[7..4] и SW[3..0]. Принятое слово Out_Ham[8..1] формируется из информационных разрядов R_Cod[3..0] и контрольных разрядов слова In_Ham.

Разряды синдрома ошибки Sindrom[2..0] и контрольный разряд четности Even формируются суммированием по модулю 2 разрядов слова Out_Ham[8..1]. Синдром поступает на двоичный дешифратор, который вырабатывает единичный сигнал на одной из линий Error[7..1], соответствующей номеру ошибочно-

го разряда. Исправление одиночной ошибки выполняется путем суммирования по модулю 2 информационных разрядов Out_Ham[8..1] и сигналов Error[7,6,5,3]. Исправленный двоичный код Out_Cod отображается на индикаторах LED[7..4].

На линии Error[0] вырабатывается единичный сигнал и отображается на индикаторе LED[8], если синдром ошибки равен нулю. На индикатор LED[3] выводится признак двойной ошибки, на индикаторы LED[2..0] – синдром ошибки. Для управления индикаторами использованы буферы с открытым коллектором (OPNDRN).

4.2 Задание к выполнению лабораторной работы

4.2.1 Выполнить компиляцию проекта Lr4 и сформировать конфигурационный файл Lr4.rbf.

4.2.2 Выполнить конфигурирование ПЛИС.

4.2.3 Выполнить проверку работы схемы контроля на стенде SDK6.1, по ее результатам сделать выводы.

4.2.4 Ответить на контрольные вопросы, оформить отчет.

4.3 Порядок выполнения работы

4.3.1 Используя мастер создания новых проектов, создайте проект Lr4 в директории по адресу C:\altera\qdesigns\Lr4.

4.3.2 Выполните анализ исходного описания и логический синтез проекта из меню *Processing/Start/Start Analysis & Synthesis*.

4.3.3 Через меню *Assignment/Device* задайте семейство Cyclone и тип EP1C3T144-8. ***Важно -*** установите тип файла конфигурации *Raw Binary File* (.rbf) и режим включения неиспользуемых выводов - *As inputs, tri-stated*.

4.3.4 Через меню *Assignment/Pins* выполните назначение используемых в проекте выводов ПЛИС. Необходимые данные приведены в таблице 4.3.

4.3.5 Выполните компиляцию проекта. Если сообщения об ошибках (Error) отсутствуют, то файл конфигурации сформирован.

4.3.6 Подключите разъем J7 стенда SDK-6.1 к COM-порту ПК при помощи интерфейсного кабеля. По таблице 2.2 задайте страницу flash-памяти 4. ***Только затем*** подключите питание к стенду SDK-6.1.

4.3.7 Переключите стенд в режим загрузки конфигурационных файлов для EP1C3 по последовательному каналу. Для этого нажмите кнопку «RESET» при нажатой и удерживаемой кнопке «CONTROL». Запустите на Host-компьютере программатор sdk61fw со следующими параметрами:

sdk61fw write page4 Lr4.rbf com1

После того, как программатор завершит работу, путем нажатия кнопки «RESET» произведите рестарт системы. После конфигурирования ПЛИС должна выполнять функцию заданного цифрового устройства.

Таблица 4.3

Вывод исследуемой схемы	Сигнальная линия стенда	Вывод ПЛИС
In_Cod[0]	SW4	Pin_114
In_Cod[1]	SW5	Pin_122
In_Cod[2]	SW6	Pin_119
In_Cod[3]	SW7	Pin_123
R_Cod[0]	SW0	Pin_112
R_Cod[1]	SW1	Pin_120
R_Cod[2]	SW2	Pin_113
R_Cod[3]	SW3	Pin_121
Out_Cod[0]	LED4	Pin_108
Out_Cod[1]	LED5	Pin_109
Out_Cod[2]	LED6	Pin_110
Out_Cod[3]	LED7	Pin_111
Sindrom[2..0]	LED0	Pin_104
Sindrom[2..0]	LED1	Pin_105
Sindrom[2..0]	LED2	Pin_106
double	LED3	Pin_107
Error[0]	LED8	Pin_103

4.3.8 Проверьте правильность работы схемы контроля, последовательно устанавливая комбинации положений групп переключателей «SW0»-«SW3» и «SW4»-«SW7» и фиксируя состояния светодиодов. Для полной проверки необходимо рассмотреть все 2^8 =256 наборов. Частичную проверку работы схемы контроля необходимо выполнить для случаев:

- ошибки отсутствуют (8 наборов);
- одиночные ошибки в различных разрядах (8 наборов);
- двойные ошибки (8 наборов).

4.4 Содержание отчета

Отчет должен содержать:

- схему кодирования и декодирования для кодов Хэмминга;
- результаты исследования схемы контроля на стенде, выводы.

4.5 Контрольные вопросы

4.5.1 Для чего необходим контроль передачи и хранения данных?

4.5.2 Как выполняется контроль с использованием кодов Хэмминга?

4.5.3 Охарактеризуйте корректирующую способность кода классического и модифицированного кода Хэмминга?

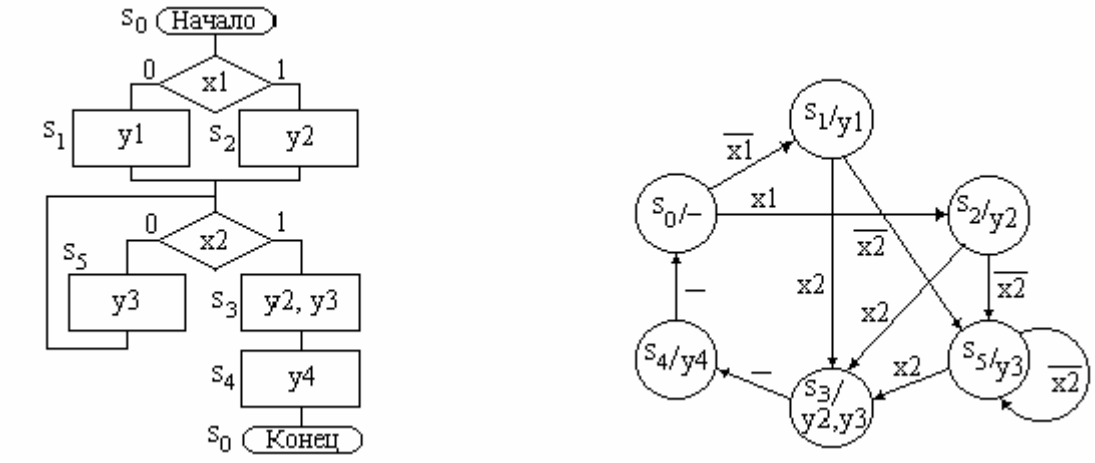
4.5.4 Объясните результаты исследования работы схемы контроля.

5 Исследование микропрограммного автомата

5.1 Описание работы с кратким изложением теории

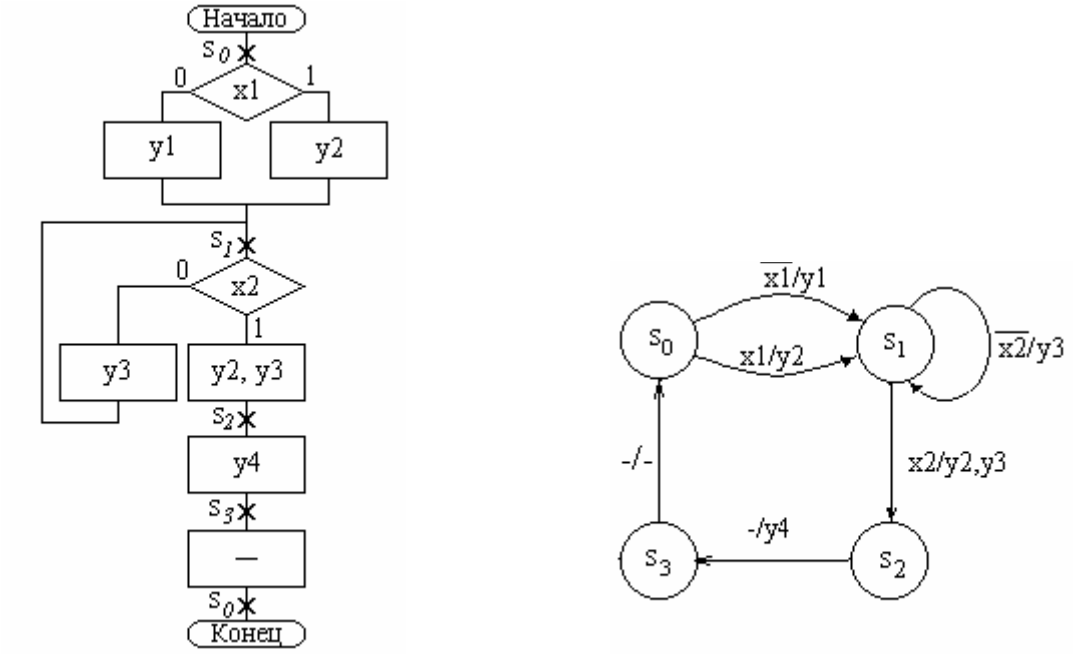
Цель работы - исследовать работу микропрограммного автомата в среде симулятора Quartus II и на лабораторных установках SDK 6.1 в потактовом режиме.

Рассмотрим пример исследования управляющего автомата, выполняющего микропрограмму, заданную граф-схемой алгоритма (ГСА) на рисунке 5.1. На ГСА выполнена отметка состояний для построения автомата Мура, рядом представлен граф автомата Мура. Отметка состояний на ГСА и граф для автомата Мили приведены на рисунке 5.2.



а) б)

Рисунок 5.1



а) б)

Рисунок 5.2

В языке AHDL конечный цифровой автомат определен как переменная перечислимого типа, которая имеет имя и набор параметров (фактически декларируется память автомата). Рекомендуется использовать сокращенную форму задания (без явного указания кодов внутренних состояний). В этом случае указываются только символические обозначения состояний в виде списка, а кодирование состояний выполняется компилятором системы Quartus II.

У конечного автомата в языке AHDL определены управляющие входы:

- инициализации <имя_автомата>.reset – единичное значение переводит автомат в исходное состояние (первое из списка);

- синхронизации <имя_автомата>.clk – смена состояний происходит по положительному фронту этого сигнала;

- разрешения <имя_автомата>.ena – изменение состояния (кроме сброса) может происходить при единичном значении этого сигнала (если вход не используется, переходы разрешены).

Составим описания автоматов на языке AHDL, для задания функций переходов и выходов используем оператор CASE. Описание автомата Мура:

```
SUBDESIGN Moore
(clk, rst, en, xr[2..1] : input;
 y[4..1] : output; )
VARIABLE
  ss : machine with states(s0,s1,s2,s3,s4,s5);
BEGIN
ss.(clk, reset, ena)  = (clk, rst, en);
CASE ss is
  WHEN s0 =>
  if xr[1] then ss = s2; else ss = s1; end if;
  WHEN s1 =>
  if xr[2] then ss = s3; else ss = s5; end if;
  y[1] = vcc;
  WHEN s2 =>
   if xr[2] then ss = s3; else ss = s5; end if;
   y[2] = vcc;
  WHEN s3 =>
   ss = s4;
   y[2] = vcc; y[3]=vcc;
  WHEN s4 =>
   ss = s0;
   y[4] = vcc;
  WHEN s5 =>
   if xr[2] then ss = s3; else ss = s5; end if;
   y[3] = vcc;
 END CASE;
END;
```

Особенностью автомата Мили является зависимость значений выходных сигналов не только от текущего состояния, но и от значений входных сигналов. Поэтому автомат Мили отличается способом задания значений функции выходов. Кроме этого, выходные сигналы автомата Мили с асинхронными выходами могут изменяться непосредственно вслед за изменениями входных сигналов, не дожидаясь прихода фронта синхросигнала Clk. Чтобы исключить такую возможность, достаточно на выходе автомата Мили включить элементы памяти (регистр), управляемые синхросигналом Clk и сигналом сброса Rst. Для этого достаточно в разделе переменных описать регистр Y[4..1] : DFFE;
а затем описать управляющие сигналы регистра:

y[].(clk,clrn,ena)=(clk,!rst,en); --.

Учитывая, что выходные сигналы автомата Мили формируются на один такт раньше, чем у эквивалентного автомата Мура, а также, что включение синхронного регистра на выходе вызовет задержку на один такт, можно утверждать, что построенные с учетом изложенных выше рекомендаций автоматы Мура и Мили будут работать идентично. Описание автомата Мили:

```
SUBDESIGN Mealy
( clk, rst, en, xr[2..1] : input;
 y[4..1] : output; )
VARIABLE
  ss : machine with states(s0,s1,s2,s3);
  y[4..1] : DFFE;
BEGIN
 ss.(clk, reset, ena)  = (clk, rst, en);
 y[].(clk,clrn,ena)=(clk,!rst,en);
 CASE ss is
  WHEN s0 =>
    ss = s1;
    if xr[1] then y[2] = vcc; else y[1] = vcc; end if;
  WHEN s1 =>
    if xr[2] then ss = s2; y[2] = vcc; else ss = s1; end if;
    y[3] = vcc;
  WHEN s2 =>
    ss = s3;
   y[4] = vcc;
  WHEN s3 =>
   ss = s0;
 END CASE;
END;
```

Сформированные описания автоматов можно использовать для синтеза и анализа работы в среде системы Quartus II. Для автомата Мура алгоритмом предусмотрено выполнение переходов:

s0 – s1; s0 – s2;
s1 – s3; s1 – s5;
s2 – s3; s2 – s5;
s3 – s4;
s4 – s0;
s5 – s3; s5 – s5.

Необходимо сформировать такую цепочку из этих переходов, чтобы в ней были все переходы, а общее число переходов было минимально, например:

s0 – s1 – s5 – s5 – s3 – s4 – s0 – s2 – s3 – s4 – s0 – s2 – s5.

Для данной последовательности сформируем тест, задавая последовательность соответствующих значений входных сигналов xr1, xr2, необходимых для выполнения каждого перехода.

На рисунках 5.3 и 5.4 соответственно приведены диаграммы сигналов рассмотренных автоматов Мура и Мили. Анализ результатов позволяет сделать выводы:

- эти автоматы эквивалентны;
- автоматы работают в соответствии с заданным алгоритмом.

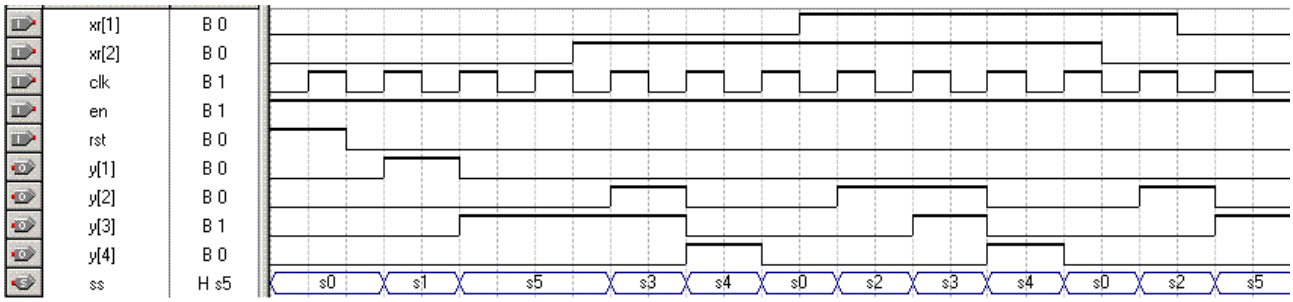


Рисунок 5.3

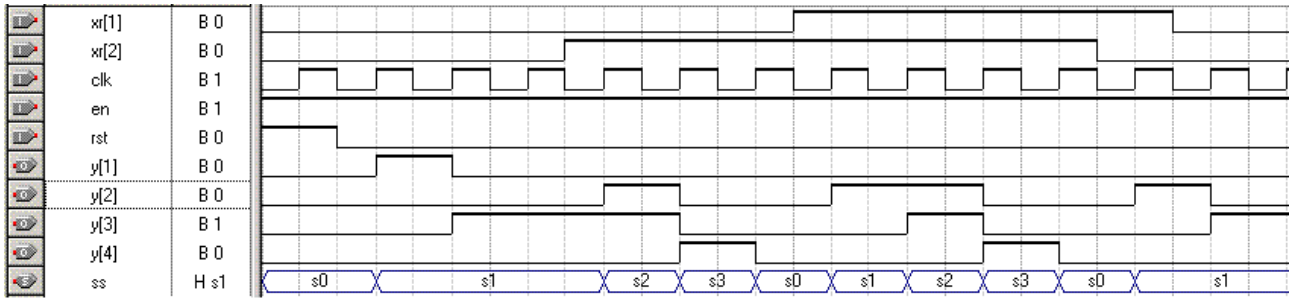


Рисунок 5.4

Исследование работы автомата на стенде SDK 6.1 удобно выполнять в потактовом режиме. Потактовый режим работы обеспечивается за счет использования формирователя одиночного импульса, выделяющего из импульсной по-

следовательности высокой частоты импульс при нажатии кнопки SB. Схема подключения кнопки показана на рисунке 5.5.

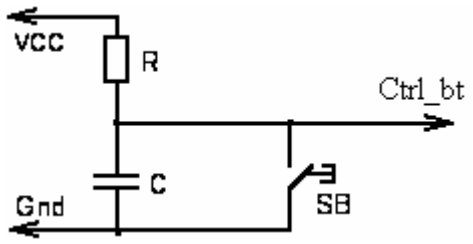


Рисунок 5.5 – Схема подключения кнопки

В нажатом состоянии на линии Ctrl_bt будет логический уровень нуля. В свободном состоянии – уровень единицы.

Следует обратить внимание, что при нажатиях на кнопку возможен “дребезг” сигнала, который может приводить к многократному переключению автомата вместо правильного однократного. Конденсатор С несколько улучшает ситуацию, однако не является абсолютным решением проблемы со стабильностью логического уровня на линии. Для устранения “дребезга” используем специальный модуль Debounce.tdf.

Полный текст описания устройства для исследования автомата в потактовом режиме приведен в файле Lr5.tdf.

5.2 Задание к выполнению лабораторной работы

5.2.1 Выполнить неполную компиляцию модуля автомата в среде Quartus II.

5.2.2 Создать файл с описанием тестовых сигналов (средствами Waveform Editor), выполнить моделирование модуля и убедиться в правильности его работы (в противном случае найти и устранить ошибки).

5.2.3 Создать прототип модуля (команда *Create AHDL Include File for Current File*).

5.2.4 Выполнить полную компиляцию проекта Lr5.tdf в среде системы Quartus II.

5.2.5 Выполнить программирование конфигурационного ПЗУ стенда.

5.2.6 Произвести проверку работы автомата в потактовом режиме на стенде SDK6.1. По результатам исследования в среде симулятора и на стенде сделать выводы о соответствии реализованного автомата заданному алгоритму функционирования.

5.2.7 Ответить на контрольные вопросы, оформить отчет.

5.3 Порядок выполнения работы

5.3.1 Используя мастер создания новых проектов, создайте проект Lr5 в директории по адресу C:\altera\qdesigns\Lr5.

5.3.2 Выполните анализ исходного описания и логический синтез проекта из меню *Processing/Start/Start Analysis & Synthesis*.

5.3.3 Через меню *Assignment/Device* задайте семейство Cyclone и тип EP1C3T144-8. ***Важно -*** установите тип файла конфигурации *Raw Binary File* (.rbf) и режим включения неиспользуемых выводов - *As inputs, tri-stated*.

5.3.4 Через меню *Assignment/Pins* выполните назначение используемых в проекте выводов ПЛИС. Необходимые для данной работы данные приведены в таблице 5.1.

Таблица 5.1

Вывод исследуемой схемы	Сигнальная линия стенда	Вывод ПЛИС
xr[1]	SW0	Pin_112
xr[2]	SW1	Pin_120
clk	clk	Pin_93
Ctrl_bt	Ctrl_bt	Pin_133
y[1]	LED0	Pin_104
y[2]	LED1	Pin_105
y[3]	LED2	Pin_106
y[4]	LED3	Pin_107

5.3.5 Выполните компиляцию проекта. Если сообщения об ошибках (Error) отсутствуют, то файл конфигурации сформирован.

5.3.6 Подключите разъем J7 стенда SDK-6.1 к COM-порту ПК при помощи интерфейсного кабеля. По таблице 2.2 задайте страницу flash-памяти 5. ***Только затем*** подключите питание к стенду SDK-6.1.

5.3.7 Переключите стенд в режим загрузки конфигурационных файлов для EP1C3 по последовательному каналу. Для этого нажмите кнопку «RESET» при нажатой и удерживаемой кнопке «CONTROL». Запустите на Host-компьютере программатор sdk61fw со следующими параметрами:

sdk61fw write page5 Lr5.rbf com1

После того, как программатор завершит работу, путем нажатия кнопки «RESET» произведите рестарт системы. После конфигурирования ПЛИС должна выполнять функцию заданного цифрового устройства.

5.3.8 Проверьте правильность работы автомата. Для этого с помощью переключателей «SW0»-«SW1» последовательно задавайте значения входных сигналов, нажатием кнопки «CONTROL» переводе автомат в следующее состояние и по состоянию светодиодов «D0»-«D3» определяйте значения выходных сигналов.

5.4 Содержание отчета

Отчет должен содержать:

- описание проекта на языке AHDL.
- результаты исследования автомата в среде симулятора (временные диаграммы) и на стенде (словарное преобразование), выводы.

5.5 Контрольные вопросы

5.5.1 Как в языке AHDL определен конечный цифровой автомат?

5.5.2 Как в языке AHDL можно описать функции переходов и выходов автомата?

5.5.3 Как отличается функционирование автомата Мура и эквивалентного ему автомата Мили с асинхронными выходами?

5.5.4 Как можно устранить эти отличия?

5.5.5 Как обеспечить функционально полный анализ работы конечного цифрового автомата?

6 Исследование Flash ROM

6.1 Описание работы с кратким изложением теории

Цель работы - исследовать работу с Flash ROM в среде САПР Quartus II и на лабораторных установках SDK 6.1.

В состав лабораторного стенда SDK 6.1 включено устройство Flash ROM типа Am29LV800BB. Схема включения Flash ROM в стенде SDK 6.1 приведена на рисунке 6.1. Выводы Flash ROM подключены к выводам ПЛИС:

- шина адреса A[18..0] = a[19..1];
- шина данных DQ[15..0] = d[15..0];
- выборка кристалла #CE = fl_ce_b;
- разрешение выхода #OE = oe_b;
- разрешение записи #WE = we_b;
- инициализации #RESET = int_res_b.

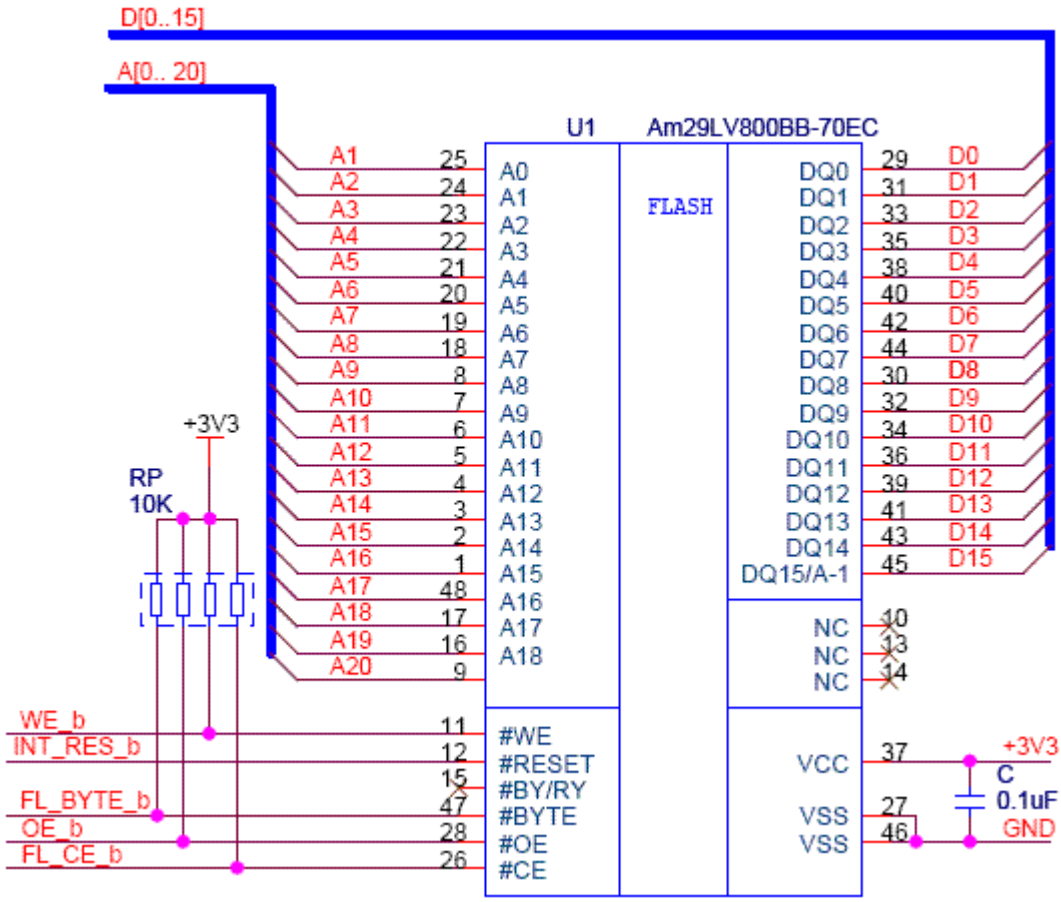


Рисунок 6.1 – Структурная схема Flash ROM

Адресное пространство Flash ROM разбито на 19 секторов. Распределение адресов Flash ROM по секторам приведено в таблице 6.1. Основные режимы работы Flash ROM приведены в таблице 6.2.

Таблица 6.1

Sector	A18	A17	A16	A15	A14	A13	A12	Sector Size (Kbytes/ Kwords)	Address Range (x16) Address Range
SA0	0	0	0	0	0	0	X	16/8	00000h-01FFFh
SA1	0	0	0	0	0	1	0	8/4	02000h-02FFFh
SA2	0	0	0	0	0	1	1	8/4	03000h-03FFFh
SA3	0	0	0	0	1	X	X	32/16	04000h-07FFFh
SA4	0	0	0	1	X	X	X	64/32	08000h-0FFFFh
SA5	0	0	1	0	X	X	X	64/32	10000h-17FFFh
SA6	0	0	1	1	X	X	X	64/32	18000h-1FFFFh
SA7	0	1	0	0	X	X	X	64/32	20000h-27FFFh
SA8	0	1	0	1	X	X	X	64/32	28000h-2FFFFh
SA9	0	1	1	0	X	X	X	64/32	30000h-37FFFh
SA10	0	1	1	1	X	X	X	64/32	38000h-3FFFFh
SA11	1	0	0	0	X	X	X	64/32	40000h-47FFFh
SA12	1	0	0	1	X	X	X	64/32	48000h-4FFFFh
SA13	1	0	1	0	X	X	X	64/32	50000h-57FFFh
SA14	1	0	1	1	X	X	X	64/32	58000h-5FFFFh
SA15	1	1	0	0	X	X	X	64/32	60000h-67FFFh
SA16	1	1	0	1	X	X	X	64/32	68000h-6FFFFh
SA17	1	1	1	0	X	X	X	64/32	70000h-77FFFh
SA18	1	1	1	1	X	X	X	64/32	78000h-7FFFFh

Таблица 6.2

Последовательность команд	Кол-во циклов	Циклы											
		1		2		3		4		5		6	
		Addr	Data	Addr	Data	Addr	Data	Addr	Data	Addr	Data	Addr	Data
Чтение	1	RA	RD										
Программирование	4	555	AA	2AA	55	555	A0	PA	PD				
Стирание сектора	6	555	AA	2AA	55	555	80	555	AA	2AA	55	SA	30

RA – адрес ячейки при чтении;
RD – считываемые данные;
PA – адрес ячейки при программировании (записи);
PD – записываемые данные;
SA – адрес стираемого сектора.

Из таблицы 6.2 следует, что только операция чтения выполняется непосредственно при обращении к ячейке памяти по адресу RA. Чтение выполняется в течение одного цикла. Остальные операции выполняются в течение 4 – 6 циклов, при этом во внутреннее устройство управления (State control) должна

быть записана соответствующая последовательность команд. Поэтому для работы с Flash ROM необходимо использовать внешнее устройство управления (контроллер).

В приложении А приведено AHDL-описание устройства для работы с Flash ROM в пошаговом режиме. Устройство содержит следующие функциональные узлы:

- делители частоты на 5 (для тактирования автомата последовательности команд), на 2^{14} (для тактирования схемы опроса кнопки);
- автомат цикла чтения-записи;
- автомат последовательности команд;
- счетчик;
- формирователь адреса;
- регистр считанных данных;
- селектор данных;
- селектор адреса;
- модуль индикации.

Автомат последовательности команд (модуль mem_contr.tdf) анализирует состояние переключателей SW[7..6] и после срабатывания схемы запуска от кнопки button_b формирует сигналы для управления автоматом цикла чтения-записи (модуль mem_cycle.tdf), селекторами адреса и данных.

Положения переключателей SW[7..6] соответствуют режимам работы:

00 – чтение ячейки;

01 – запись в ячейку;

10 – стирание сектора.

Автомат цикла чтения-записи формирует значения сигналов, управляющих непосредственно Flash ROM (fl_ce_b, oe_b, we_b), регистром считанных данных (ld), трехстабильным буфером данных (buf_oe).

В работе исследуется выполнение основных операций со страницами SA11 – SA18 Flash ROM стенда SDK 6.1 в диапазоне адресов 0x40000h - 0x7FFFFh (при a[19] = vcc, a[15..9]=0).

Значения старших разрядов адреса Flash ROM a[18..16] задаются положением движковых переключателей SW[2..0]. Для формирования младших разрядов адреса a[8..1] используется восьмиразрядный счетчик cntr[7..0], срабатывающий при нажатии на кнопку button_b. Выходные сигналы этого счетчика также используются как данные для записи в ячейки Flash ROM. Состояние счетчика и содержимое регистра считанных данных выводятся на ЖКИ.

Описание проекта на языке AHDL сохранено в файле Lr6.tdf.

6.2 Задание к выполнению лабораторной работы

6.2.1 Выполнить полную компиляцию проекта Lr6.tdf .

6.2.2 Произвести проверку работы проекта в среде симулятора для циклов чтения (SW[7..6]=00), записи (SW[7..6]=01) и стирания (SW[7..6]=10). Зарисовать диаграммы сигналов clk, mwr_b, mrd_b, csf_b, d[7..0], a[17..0].

6.2.3 Выполнить программирование конфигурационного ПЗУ.

6.2.4 Произвести проверку работы проекта на стенде SDK6.1:

- выполнить стирание заданного сектора Flash ROM;
- выполнить программирование 32 ячеек в заданном секторе Flash ROM;
- считать записанную информацию.

Занести в отчет результаты исследования.

6.2.5 Ответить на контрольные вопросы, оформить отчет.

6.3 Порядок выполнения работы

6.3.1 Используя мастер создания новых проектов, создайте проект Lr6 в директории по адресу C:\altera\qdesigns\Lr6.

6.3.2 Выполните анализ исходного описания и логический синтез проекта из меню *Processing/Start/Start Analysis & Synthesis*.

6.3.3 Через меню *Assignment/Device* задайте семейство Cyclone и тип EP1C3T144-8. ***Важно -*** установите тип файла конфигурации *Raw Binary File* (.rbf) и режим включения неиспользуемых выводов - *As inputs, tri-stated*.

6.3.4 Через меню *Assignment/Pins* выполните назначение используемых в проекте выводов ПЛИС. Необходимые для данной работы данные приведены в таблице 6.3.

6.3.5 Выполните компиляцию проекта. Если сообщения об ошибках (Error) отсутствуют, то файл конфигурации сформирован.

6.3.6 Подключите разъем J7 стенда SDK-6.1 к COM-порту ПК при помощи интерфейсного кабеля. По таблице 2.2 задайте страницу flash-памяти 5. ***Только затем*** подключите питание к стенду SDK-6.1.

6.3.7 Переключите стенд в режим загрузки конфигурационных файлов для EP1C3 по последовательному каналу. Для этого нажмите кнопку «RESET» при нажатой и удерживаемой кнопке «CONTROL». Запустите на Host-компьютере программатор sdk61fw со следующими параметрами:

sdk61fw write page6 Lr6.rbf com1

После того, как программатор завершит работу, путем нажатия кнопки «RESET» произведите рестарт системы. После конфигурирования ПЛИС должна выполнять функцию заданного цифрового устройства.

6.3.8 Проверьте правильность работы устройства в соответствии с пунктом 6.2.4.

6.4 Содержание отчета

Отчет должен содержать

- структурную схему Flash ROM;
- структурную схему контроллера для работы с Flash ROM (сформировать по описанию проекта на языке AHDL);
- результаты моделирования, выводы;
- результаты исследования проекта на стенде, выводы.

Таблица 6.3

Сигнальная линия стенда	Вывод ПЛИС	Сигнальная линия стенда	Вывод ПЛИС
SW0	Pin_112	a[0]	PIN_70
SW1	Pin_120	a[1]	PIN_71
SW2	Pin_113	a[2]	PIN_72
SW3	Pin_121	a[3]	PIN_73
SW4	Pin_114	a[4]	PIN_74
SW5	Pin_122	a[5]	PIN_75
SW6	Pin_119	a[6]	PIN_76
SW7	Pin_123	a[7]	PIN_77
clk	Pin_93	a[8]	PIN_78
Ctrl_bt	Pin_133	a[9]	PIN_79
d[0]	PIN_48	a[10]	PIN_82
d[1]	PIN_49	a[11]	PIN_83
d[2]	PIN_50	a[12]	PIN_84
d[3]	PIN_51	a[13]	PIN_85
d[4]	PIN_52	a[14]	PIN_91
d[5]	PIN_53	a[15]	PIN_94
d[6]	PIN_56	a[16]	PIN_96
d[7]	PIN_57	a[17]	PIN_97
d[8]	PIN_58	a[18]	PIN_98
d[9]	PIN_59	a[19]	PIN_99
d[10]	PIN_60	a[20]	PIN_100
d[11]	PIN_61	E	Pin_131
d[12]	PIN_62	RW	Pin_130
d[13]	PIN_67	RS	Pin_132
d[14]	PIN_68	Data_ind4	Pin_124
d[15]	PIN_69	Data_ind5	Pin_125
fl_ce_b	PIN_38	Data_ind6	Pin_128
oe_b	PIN_41	Data_ind7	Pin_129
we_b	PIN_40		

6.5 Контрольные вопросы

6.5.1 Поясните основные особенности построения и работы Flash ROM.
6.5.2 Как выполняется чтение, программирование и стирание Flash ROM?
6.5.3 Объясните результаты исследования работы проекта.

7 Встроенный логический анализатор

7.1 Описание работы с кратким изложением теории

Цель работы - получить практические навыки регистрации сигналов реализованных на кристалле ПЛИС цифровых устройств с помощью встроенного логического анализатора.

Логический анализатор – контрольно-измерительный прибор, предназначенный для сбора данных о поведении дискретных систем, для обработки этих данных и представления их человеку на различных уровнях абстракции.

Объем памяти логического анализатора ограничен, поэтому регистрация сигналов выполняется на интервале времени, привязанном к событию, с которым связывают условие *запуска*. Запуск логического анализатора при появлении на входах логического анализатора определенной, заранее выбранной комбинации значений сигналов называется запуском по *кодовому* слову. Собранная логическим анализатором информация, предшествующая запускающему событию называется *предысторией***.** Информация, собранная логическим анализатором после запускающего события, называется *послеисторией.*

Основные технические характеристики встроенного логического анализатора:

- число каналов – 8;
- число бит информации по каждому каналу – 512;
- режим регистрации – синхронный;
- запуск безусловный или по кодовому слову.

Запуск по кодовому слову может выполняться:

- с предысторией из 16 слов;
- с послеисторией из 16 слов;
- с равными предысторией и послеисторией (по 256 слов).

Регистрация синхронная по положительному фронту системного синхросигнала ПЛИС, либо по положительному/отрицательному фронту любого из сигналов проекта.

Структурная схема включения встроенного логического анализатора (ЛА) приведена на рисунке 7.1. Управление настройками логического анализатора выполняется с помощью программы LA, установленной на инструментальном компьютере. Эта же программа принимает и сохраняет в файле собранную логическим анализатором информацию. Обмен информацией между инструментальным компьютером и встроенным в ПЛИС логическим анализатором осуществляется с помощью интерфейса RS232. Поэтому в состав отлаживаемого проекта кроме тестируемого устройства и логического анализатора включается модуль асинхронного приемопередатчика UART.

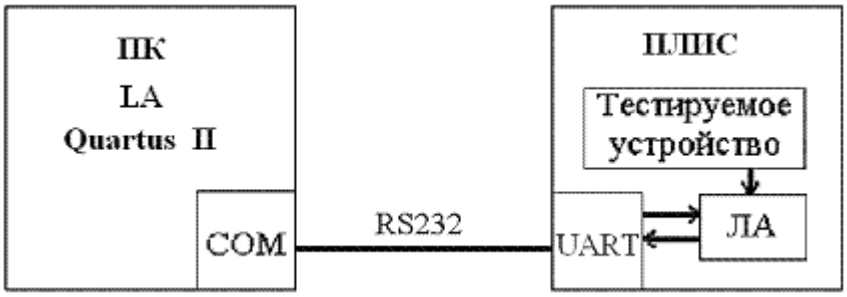


Рисунок 7.1

Встроенный логический анализатор реализован в форме мегафункции, которую можно включать в любые проекты для их отладки. Мегафункция реализована на языке AHDL. В качестве входных параметров мегафункции логического анализатора задаются имена восьми информационных входов DI[7..0], выделенного синхросигнала cke, системного синхросигнала clk и входа приемника UART rs232_txd:

FUNCTION LA (DI[7..0], cke, clk, rs232_txd);--.

Функция передает в отлаживаемый проект выходной сигнал передатчика UART:

RETURNS (rs232_rxd);--.

Чтобы включить в проект мегафункцию логического анализатора, достаточно:

- использовать директиву ее включения при компиляции проекта;
- использовать оператор вызова мегафункции;
- включить в описание внешнего интерфейса проекта сигналы модуля UART (rs232_txd : input; rs232_rxd : output;).

Окно программы LA приведено на рисунке 4.2. После выбора используемого COM-порта компьютера необходимо:

- задать скорость обмена (115200 бод);
- открыть порт;
- задать кодовое слово запуска, тип синхронизации и вид запуска;
- включить опрос порта;
- передать сигнал «Готовность».

Принимаемые от логического анализатора данные отображаются в окне «Чтение». После приема пакета данных необходимо закрыть порт и завершить выполнение программы.

Синхронизация возможна:

- по положительному фронту системного синхросигнала (clk);
- по положительному фронту выделенного сигнала (_/cke);
- по отрицательному фронту выделенного сигнала (_cke).

Запуск по кодовому слову возможен:

- с предысторией из 496 слов и послеисторией из 16 слов (_ _0);
- с равными предысторией и послеисторией по 256 слов (_0_);
- с предысторией из 16 слов и послеисторией из 496 слов (0_ _).

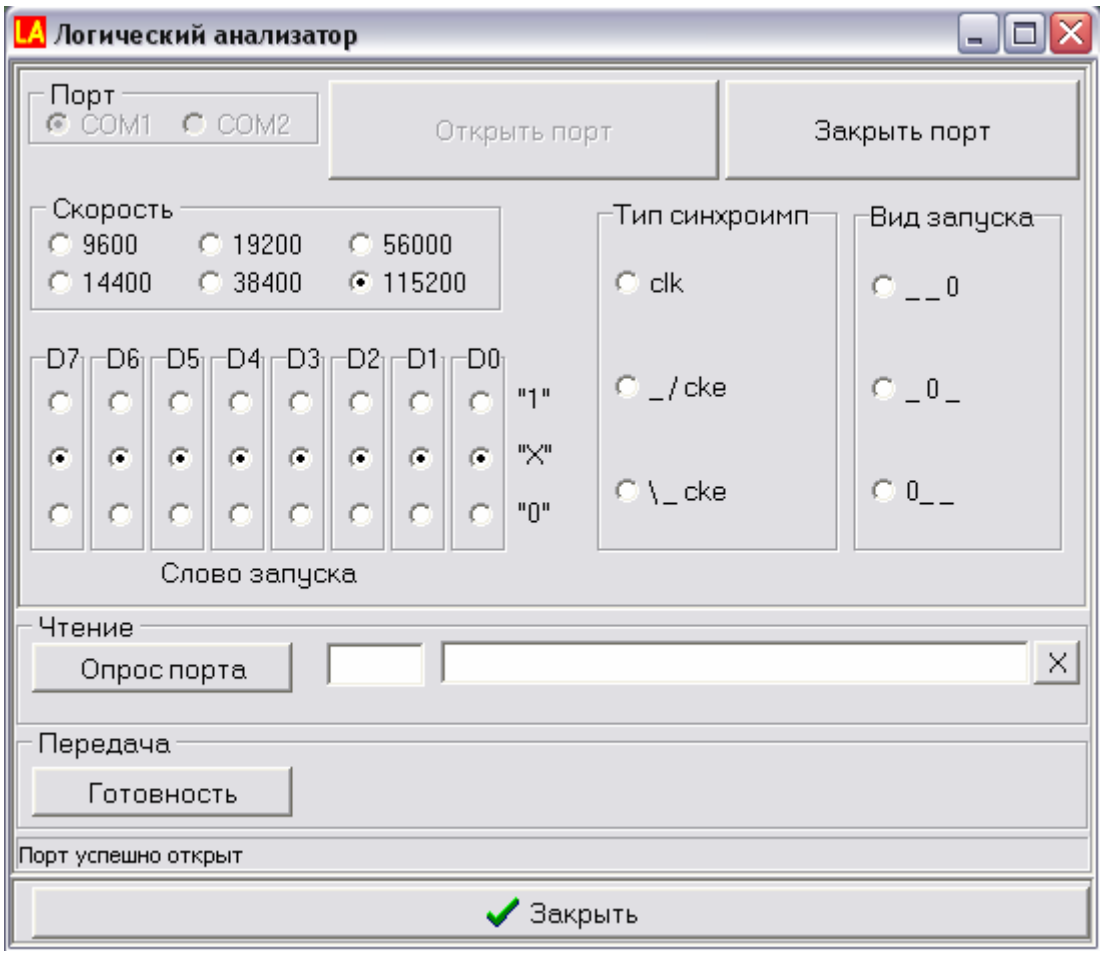


Рисунок 7.2 – Окно программы LA

Результатом работы данной программы является текстовый файл LA.tbl. Этот файл можно открыть в окне редактора временных диаграмм Waveform Editor программы Quartus II, то есть собранные данные об исследуемом объекте представляются в форме временных диаграмм.

В лабораторной работе рассматривается пример использования встроенного логического анализатора для регистрации сигналов схемы, представленной на рисунке 7.3.

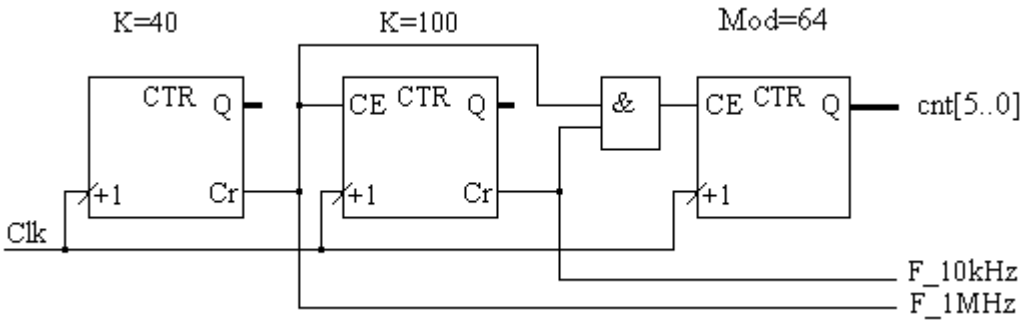


Рисунок 7.3

Схема состоит из двух делителей частоты на 40 и 100 соответственно. На вход первого делителя частоты подаются синхроимпульсы clk частотой 40 МГц. На выходах делителей частоты формируются импульсные последовательности частотой 1 МГц и 10 кГц соответственно. Схема также содержит двоичный синхронный счетчик импульсов 10 кГц по модулю 64. AHDL-описание проекта приведено в файле Lr7.tdf.

7.2 Задание к выполнению лабораторной работы

Выполнить регистрацию выходных сигналов делителей частоты и счетчика в соответствии с вариантом задания:

1) в течение 12.4 мкс до достижения состояния счетчика 8;
2) в течение 496 мкс до достижения состояния счетчика 33;
3) в течение 12.4 мкс после достижения состояния счетчика 15;
4) в течение 496 мкс после достижения состояния счетчика 0;
5) в течение 6.4 мкс до и 6.4 мкс после достижения состояния счетчика 10;
6) в течение 248 мкс до и 248 мкс после достижения состояния счетчика 63.

7.3 Описание последовательности выполнения

7.3.1 Создайте проект Lr7, задайте тип ПЛИС, выполните назначение выводов ПЛИС (необходимые данные приведены в таблице 7.1).

7.3.2 Выполните полную компиляцию проекта Lr7.tdf .

7.3.3 Выполните программирование страницы 7 конфигурационного ПЗУ стенда SDK6.1. Загрузите конфигурацию в ПЛИС.

7.3.4 Запустите на выполнение программу LA, выполните настройку логического анализатора в соответствии с вариантом задания, включите опрос порта, передайте сигнал «Готовность» и дождитесь запуска логического анализатора.

7.3.5 После окончания процесса передачи данных от логического анализатора в инструментальный компьютер откройте в окне редактора временных диаграмм *Waveform Editor* системы Quartus II файл la.tbl. Выполните анализ временных диаграмм.

7.3.6 Занести в отчет результаты исследования.

7.3.7 Ответьте на контрольные вопросы, оформите отчет о выполненной работе.

Таблица 7.1

Вывод исследуемой схемы	Сигнальная линия стенда	Вывод ПЛИС
clk	clk	Pin_93
rs232_txd	rs232_txd	Pin_139
rs232_rxd	rs232_rxd	Pin_134

7.4 Оформление работы

Отчет по лабораторной работе должен содержать:
- схему включения логического анализатора;
- параметры режимов работы логического анализатора;
- временные диаграммы.

7.5 Контрольные вопросы

7.5.1 С какой целью используется логический анализатор при отладке цифровых устройств?

7.5.2 Каким образом выполняется подготовка к работе встроенного логического анализатора?

7.5.3 Каким образом выполняется регистрация сигналов?

8 Сигнатурный анализ

8.1 Описание работы с кратким изложением теории

Цель работы - получить практические навыки диагностики неисправностей цифровых устройств с использованием сигнатурного анализатора.

Сигнатурный анализатор – прибор, предназначенный для диагностики и локализации неисправностей в цифровых устройствах, в том числе микропроцессорных системах, с точностью до интегральной микросхемы. Работа выполняется с использованием сигнатурного анализатора АС-817.

Работа анализатора сигнатурного основана на преобразовании (сжатии) входных последовательных двоичных потоков с помощью заторможенного генератора последовательности максимальной длины, который представляет собой регистр сдвига с обратными связями по модулю два. Начало и конец обработки (окна измерения) определяются внешними управляющими сигналами ПУСК и СТОП. Входная двоичная последовательность поступает по входу ДАННЫЕ на регистр сдвига с обратными связями синхронно с управляющим сигналом ТАКТ, также снимаемым с проверяемой схемы. Результирующий двоичный код, получаемый в регистре после прихода сигнала СТОП, дает сигнатуру, которая отображается в шестнадцатеричном коде.

Структурная схема анализатора АС-817 приведена на рисунке 8.1, она включает:

- пробник одноканальный - для подачи на прибор сигнала данных с контролируемой точки проверяемой схемы;
- пробник трехканальный - для подачи на прибор управляющих сигналов ПУСК, СТОП, ТАКТ;
- селектор активных фронтов - обеспечивает возможность выбора активного фронта независимо у каждого из управлявших сигналов;
- формирователь окна измерения - формирует временной интервал, в течение которого осуществляется обработка входных данных;
- формирователь - синхронизирует сигналы, поступающие с пробника одноканального, сигналами ТАКТ, формирует сигнал индикации уровня сигнала ДАННЫЕ;
- регистр сдвига длиной 16 бит с сумматором по модулю 2 - обеспечивает сжатие входной последовательности в сигнатуру;
- буферный регистр - для запоминания результата обработки и подачи его на дешифратор;
- память - для запоминания результатов обработки в предыдущем окне измерения;
- компаратор - для сравнения следующих друг за другом результатов обработки с целью обнаружения нестабильных сигнатур;
- дешифратор - для преобразования четырехразрядного двоичного кода в семисегментный;

- индикатор - отображает результаты обработки на передней панели анализатора;

- формирователь однократного режима - обеспечивает возможность измерения сигнатур в одиночном окне;

- синхронизатор индикации - обеспечивает синхронную работу индикатора и дешифратора;

- генератор теста самопроверки - обеспечивает возможность самопроверки прибора;

- генератор импульсов начальной установки – обеспечивает исходное состояние схем прибора при его включении;

- генератор задающий - вырабатывает набор опорных частот для генератора теста самопроверки;

- формирователь сигнала проверки дешифратора - обеспечивает проверку схем дешифратора и индикации при включении прибора и теста.

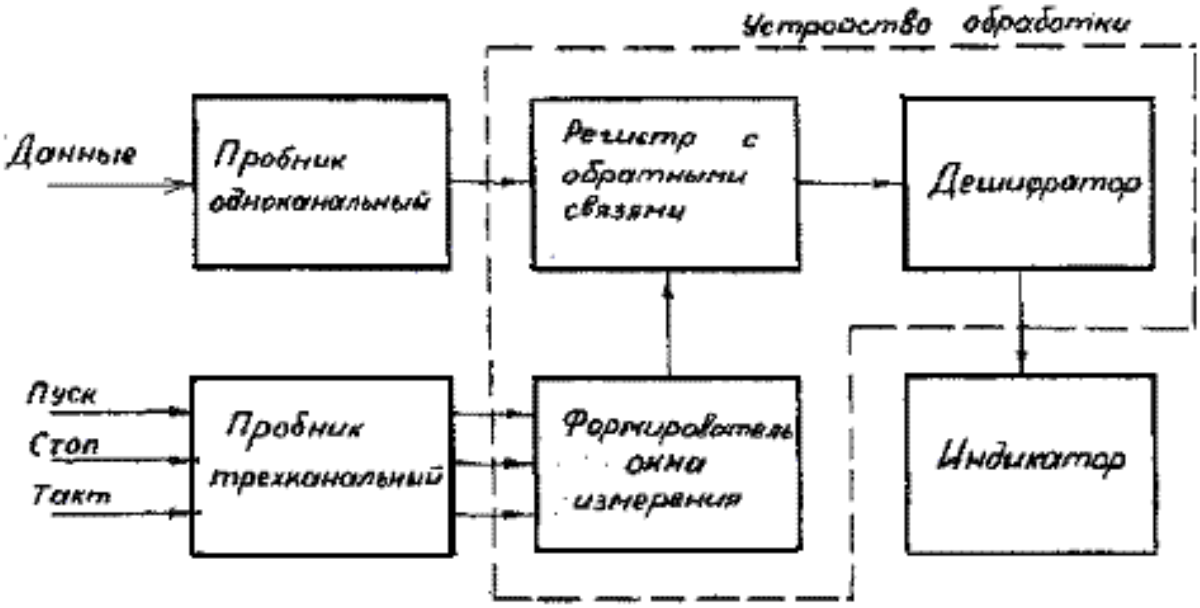


Рисунок 8.1

Органы управления и соединительные разъемы расположены на передней панели прибора (рисунок 8.2):

- СЕТЬ ВКЛ. (1) - тумблер включения анализатора сигнатурного;

- ПУСК, СТОП, ТАКТ (2,3,4) - переключатели выбора активных фронтов управляющих сигналов ПУСК, СТОП, ТАКТ;

- ОДНОКР. (5) - переключатель перевода анализатора сигнатурного в однократный режим работы;

- СБРОС. (6) - кнопка установки прибора в исходное состояние;

- ТЕСТ ВКЛ. (7) - переключатель перевода прибора в режим самопроверки;

- СТРОБ. (8) - индикатор состояния окна измерения;

- НЕСТАБ. (9) - индикатор появления нестабильных сигнатур;

- ПУСК, СТОП, ТАКТ, ДАННЫЕ (10 – 13) - гнезда для подключения соот-

ветствующих входов пробников одноканального и трехканального в режиме самопроверки (выходы теста самопроверки);

- разъем для подключения одноканального пробника (14);
- разъем для подключения трехканального пробника (15);
- индикатор сигнатуры (16);
- пробник одноканальный (17);
- пробник трехканальный (18).

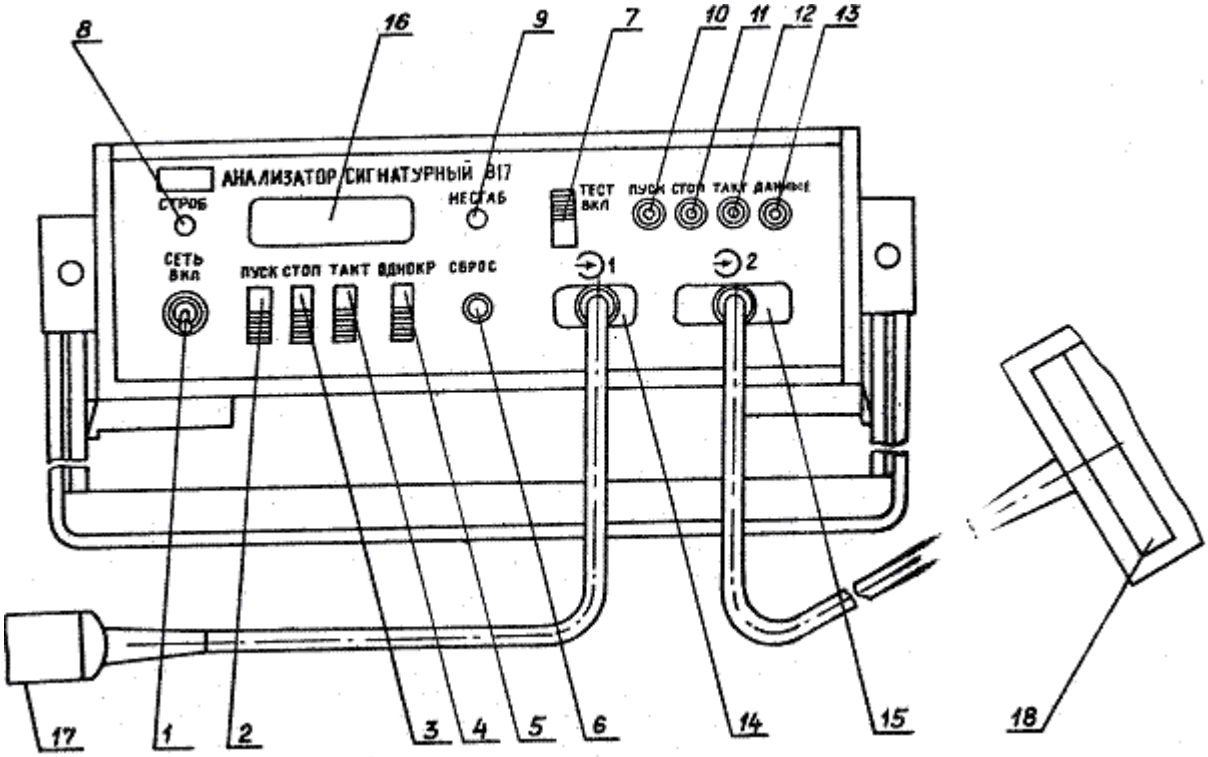


Рисунок 8.2

Анализатор сигнатурный работает следующим образом. Входной сигнал ДАННЫЕ считывается с контролируемой точки проверяемой схемы и через пробник одноканальный и формирователь поступает на вход сумматора по модулю 2. На другие входы этого сумматора поданы сигналы с выводов 7, 9, 12, 16 регистра сдвига. Сигнал с выхода сумматора поступает на вход регистра сдвига.

Входные управляющие сигналы ПУСК и СТОП определяют начало и конец окна измерения, в течение которого сигнал поступает в регистр сдвига синхронно с внешним тактовым сигналом ТАКТ (рисунок 8.3). Селектор активного фронта позволяет выбрать с помощью соответствующих переключателей полярность перехода, воздействующего на схему формирователя окна измерения независимо для каждого из входных управляющих сигналов.

По окончании окна измерения содержимое регистра сдвига записывается в буферный регистр, а затем регистр сдвига сбрасывается в ноль, так что к началу следующего окна измерения регистр сдвига находится в исходном нулевом состоянии.

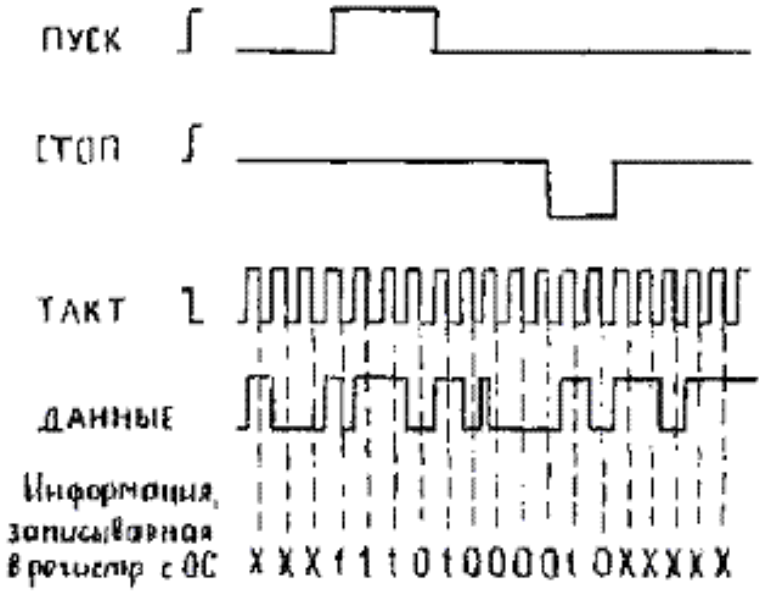


Рисунок 8.3

Установка резисторов сдвига в нулевое состояние при включении сети производится автоматически с помощью генератора установки нуля. Содержание буферного регистра отображается на индикаторе в виде четырехзначного шестнадцатеричного кода.

Последовательно с буферным регистром включена память с организацией 16 бит, запись в которую производится тем же управляющим сигналом, что и в буферный регистр. При этом информация на выходах буферного регистра и памяти - результаты обработки, соответствующие двум последовательным окнам измерения. Эта информация сравнивается шестнадцатиразрядным компаратором и при несовпадении сигнал о нестабильной сигнатуре отображается на индикаторе.

С целью повышения достоверности результатов измерения в схему прибора введен генератор теста самопроверки, который обеспечивает возможность проверки правильности функционирования прибора.

При включении теста самопроверки, а также при включении питания прибора, с формирователя сигнала проверки дешифратора подается на вход дешифратора последовательность 16-ти четырехразрядных двоичных чисел, которые обеспечивают проверку правильности работы дешифратора и индикатора. Кроме того, при включении теста, в приборе вырабатываются сигналы ПУСК, СТОП, ТАКТ, ДАННЫЕ с тактовой частотой 10 МГц, которые позволяют проверить правильность функционирования прибора на максимальной рабочей частоте (при подаче их на соответствующие входы прибора)..

Перед началом работы необходимо:

- подключить к анализатору пробники одноканальный и трехканальный.

- подключить к пробникам кабели соединительные в соответствии с цветовой кодировкой (красный – СТОП, зеленый – ПУСК, желтый - ТАКТ, синий - ДАННЫЕ);

- к гнездам "⊥" пробников подсоединить зажимы черного цвета;

- подсоединить шнур питания к напряжению сети (выключатель сети должен находиться в выключенном состоянии).

Для подготовки к проведению измерений необходимо:

- установить переключатели ОДНОКР. и ТЕСТ - в нижнее положение;

- остальные органы управления могут находиться в произвольном положении;

- переключатель СЕТЬ поставить в положение ВКЛ.;

- убедиться, что на индикаторе последовательно высветились сигнатуры

0000 1111 2222 3333 4444 5555 6666 7777 8888

9999 AAAA CCCC FFFF HHHH PPPP UUUU 0000;

- выполнить самопроверку прибора.

Дня выполнения самопроверки необходимо подключить входы трехканального пробника к соответствующим выходам генератора тестового сигнала на передней панели прибора, а вход пробника одноканального - к выходу ДАННЫЕ.

Переключатель ТЕСТ следует установить в верхнее положение и убедиться в том, что на индикаторе высветились последовательно шестнадцать сигнатур от 0000 до 0000, а затем в зависимости от положений переключателей активных фронтов ПУСК, СТОП, ТАКТ должна происходить циклическая индикация одного из наборов сигнатур, приведенных в таблице 9.1.

При совпадении отображаемых сигнатур с сигнатурами, приведенными в таблице 8.1 можно сделать вывод о работоспособности прибора.

После выполнения самопроверки необходимо заменить кабели соединительные в пробнике трехканальном на зажимы, а в пробнике одноканальном на щуп с иглой. Переключатель ТЕСТ - установить в нижнее положение.

Схема подключения прибора к проверяемому устройству в общем случае приведена на рисунке 8.4. Проверка конкретного устройства и поиск неисправности в нем с помощью анализатора сигнатурного осуществляется путем регистрации сигнатур в контрольных точках, для которых определены «эталонные» сигнатуры. Оператор подключает анализатор сигнатурный к проверяемому устройству и производит измерение сигнатур в определенной последовательности точек схемы. В случае несовпадения измеренных сигнатур с «эталонными» оператор производит поиск источника неисправности, следуя алгоритму поиска. Положения органов управления анализатора сигнатурного при измерении «эталонных» и контролируемых сигнатур должны совпадать. Чтобы определять сигнатуры в однократном режиме следует включить переключатель ОДНОКР. При этом для сброса сигнатуры и подготовки прибора к обработке необходимо нажать кнопку СБРОС.

Точки подключения входа ТАКТ к проверяемому устройству следует выбирать таким образом, чтобы информация на входе пробника ДАННЫЕ не изменялась в течение интервала времени, соответствующего высокому уровню

сигнала ТАКТ (при положении переключателя ТАКТ- ↓) или низкому уровню сигнала ТАКТ (при положении переключателя ТАКТ -↑).

Таблица 8.1

Положение переключателей			Сигнатура
ПУСК	СТОП	ТАКТ	
↑	↑	↑	2HH8 3F8H 4596 0000
↑	↑	↓	2HH8 3F8H 0000 4596
↑	↓	↓	4C2C 96PF 0000 CC34
↑	↓	↑	4C2C 96PF CC34 0000
↓	↓	↑	0000 2HH8 3F8H 4596
↓	↓	↓	2HH8 3F8H 0000 4596
↓	↑	↑	4C2C 96PF CC34 0000
↓	↑	↓	4C2C 96PF 0000 CC34

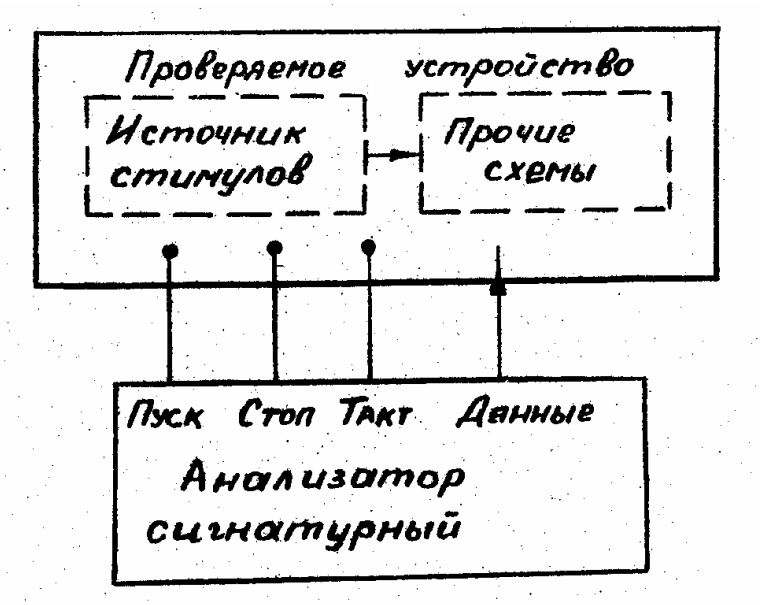


Рисунок 8.4

8.2 Задание к выполнению лабораторной работы

8.2.1 С помощью перемычек собрать на лабораторном стенде УМ12б заданную комбинационную схему. Подать на входы комбинационной схемы тестовые сигналы.

8.2.2 Подготовить сигнатурный анализатор к работе, подключить к лабораторному стенду. Включить питание стенда.

8.2.3 Измерить «эталонные» сигнатуры для исправного варианта комбинационной схемы.

8.2.4 После внесения преподавателем в схему стенда «неисправности» измерить сигнатуры для неисправного варианта комбинационной схемы.

8.2.5 Сравнить полученные результаты с “эталонными”. Определить неисправный логический элемент схемы. Обосновать принятое решение.

8.2.6 Занести в отчет результаты исследования.

8.2.7 Выключить питание стенда. Разобрать схему.

8.2.8 Ответить на контрольные вопросы, оформить отчет о выполненной работе.

8.3 Описание последовательности выполнения

8.3.1 В качестве тестовых сигналов следует использовать выходные сигналы четырех младших разрядов пятиразрядного двоичного счетчика: D, C, B, A (имеются прямые и инверсные выходы).

8.3.2 Выходной сигнал старшего разряда E (имеются прямой и инверсный выходы) пятиразрядного двоичного счетчика рекомендуется использовать для формирования временного окна измерения сигнатуры (сигналы ПУСК и СТОП).

8.3.3 В качестве сигнала ТАКТ сигнатурного анализатора использовать счетные импульсы пятиразрядного двоичного счетчика с выхода ТАКТ стенда УМ12б.

8.3.4 При выборе активного фронта сигнала ТАКТ в сигнатурном анализаторе принять во внимание, что состояние пятиразрядного двоичного счетчика изменяется по положительному фронту счетных импульсов.

8.4 Оформление работы

Отчет по лабораторной работе должен содержать:

- схему диагностируемого узла;
- схему подключения сигнатурного анализатора к проверяемому комбинационному узлу;
- измеренные сигнатуры для исправного и неисправного вариантов схемы;
- выводы.

8.5 Контрольные вопросы

8.5.1 С какой целью используют сигнатурный анализатор при диагностике цифровых устройств?

8.5.2 Каким образом выполняется подготовка к работе прибора АС -817?

8.5.3 Каким образом выполняется измерение сигнатур?

8.5.4 Как формируется сигнатура двоичной последовательности?

8.5.5 Как по результатам сигнатурного анализа можно определить неисправный элемент?

Список использованных источников

1 **Стешенко, В.Б.** ПЛИС фирмы ALTERA: элементная база, система проектирования и языки описания аппаратуры/В.Б. Стешенко.- М.: Додека, 2007.– 567 с.

2 **Антонов, А.П.** Язык описания цифровых устройств AlteraHDL: Практический курс / А.П. Антонов. – М.: ИП «Радиософт», 2001. – 224 с.

3 **Угрюмов, Е.П.** Цифровая схемотехника / Е.П. Угрюмов. – СПб: БХВ-Петербург, 2004. – 800 с.

4 Комплекс учебный лабораторный SDK6.1 [Электронный ресурс]: руководство пользователя. – СПб.: «ЛМТ», 2005. – 14 с. – Режим доступа : WWW.URL : http: // ifmo.embedded.ru/. – 27.06.2008.

5 Комплекс учебный лабораторный SDK6.1 [Электронный ресурс]: схема электрическая принципиальная. – Режим доступа : WWW.URL : http:// ifmo.embedded.ru/. – 27.06.2008.

6 Am29LV800B [Электронный ресурс]: Data Sheet. – AMD, 2005. – Режим доступа : WWW.URL : http: // www.amd.com/. – 27.06.2008.