0919

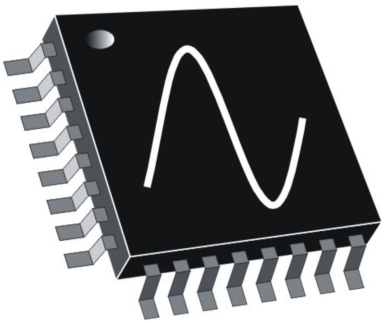
МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

РЯЗАНСКИЙ ГОСУДАРСТВЕННЫЙ РАДИОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ им. В. Ф. УТКИНА

РАЗРАБОТКА НА ПЛИС МОДУЛЯ ПОМЕХОУСТОЙЧИВОГО КОДИРОВАНИЯ ДЛЯ СИСТЕМЫ ЦИФРОВОЙ СВЯЗИ

Методические указания к курсовому проекту

П Л И С



Рязань 2020

УДК 621.372 (021)

Разработка на ПЛИС модуля помехоустойчивого кодирования для системы цифровой связи: методические указания к курсовому проекту / Рязан. гос. радиотехн. ун-т им. В. Ф. Уткина; сост.: А. Ю. Линович. — Рязань, 2020. — 28 с.

Содержат основные сведения для выполнения курсового проекта по дисциплине «Современные технологии ПЛИС». Приводятся краткие теоретические сведения по разработке систем помехоустойчивого кодирования. Описываются порядок выполнения и оформления курсового проекта, особенности подготовки описания для ПЛИС.

Предназначены студентам дневного отделения, обучающимся по направлению 11.04.02 «Инфокоммуникационные технологии и системы связи».

Табл. 7. Ил. 21. Библиогр.: 10 назв.

Программируемая логическая интегральная схема, кодер, декодер, циклический код, обнаружение и исправление ошибок

Печатается по решению редакционно-издательского совета Рязанского государственного радиотехнического университета им. В. Ф. Уткина.

Рецензент: кафедра ТОР Рязанского государственного радиотехнического университета им. В. Ф. Уткина (зав. кафедрой профессор В. В. Витязев)

Разработка на ПЛИС модуля помехоустойчивого кодирования
для системы цифровой связи

Составитель Л и н о в и ч Александр Юрьевич

Редактор М. Е. Цветкова
Корректор С. В. Макушина
Подписано в печать 05.02.20. Формат бумаги 60x84 1/16.
Бумага газетная. Печать трафаретная. Усл. печ. л. 1,75.
Тираж 50 экз. Заказ
Рязанский государственный радиотехнический университет
им. В. Ф. Уткина.
390005, Рязань, ул. Гагарина, 59/1.
Редакционно-издательский центр РГРТУ.

ВВЕДЕНИЕ

Курсовой проект направлен на формирование у студентов первого курса магистратуры навыков проектно-конструкторской деятельности. Перед студентами ставится задача разработки на программируемой логической интегральной схеме (ПЛИС) устройств, выполняющих кодирование и декодирование цифровых сигналов в системах подвижной радиосвязи. Дисциплина «Современные технологии ПЛИС», в рамках которой выполняется курсовой проект, тесно связана с дисциплиной «Современные методы и технологии ЦОС в системах связи». Компетенции, полученные в результате освоения дисциплины необходимы обучающемуся при изучении следующих дисциплин: «Технологии программно конфигурируемого радио», «Технологии мобильной связи нового поколения», «Подготовка к процедуре защиты и защита выпускной квалификационной работы». Программа курса ориентирована на возможность расширения и углубления знаний, умений и навыков студентов магистратуры для успешной профессиональной деятельности.

Выполнение курсового проекта начинается с постановки задачи. Важным критерием при разработке современных цифровых устройств является их эффективность, которая достигается в частности использованием современной элементной базы. За постановкой задачи следует этап компьютерного моделирования, позволяющий оценить возможности устройства в целом и отдельных его элементов, выявить слабые места, правильно настроить систему. Реализация устройства на ПЛИС предполагает знание архитектурных особенностей выбранного ПЛИС и отладочных средств, а также владение основами проектирования цифровых электронных устройств. Завершающим этапом является грамотное оформление полученных результатов в соответствии с требованиями государственных стандартов.

О качестве выполнения курсового проекта можно судить по решению задач оптимизации, направленных на повышение эффективности проектируемого устройства, по грамотности оформления полученных результатов, по своевременности выполнения пунктов задания и самостоятельном изучении дополнительной литературы.

Дополнительный справочный материал может быть заимствован в приведённом ниже библиографическом списке. Все сложные вопросы следует разбирать с преподавателями на консультациях по курсовому проекту, которые проводят не реже чем раз в две недели.

ЗАДАНИЕ

Разработать с использованием системы автоматизированного проектирования «Quartus II» модели, включающей в себя модуль кодирования и модуль декодирования, выполняющих преобразование двоичного кода в циклический код и обратное преобразование с обнаружением ошибок. Таблица вариантов заданий приводится в табл. 7 (раздел 3).

В отчёт по курсовому проекту включить: схему устройства, графическое описание системы кодирования-декодирования, описание кодера и декодера на языке «Verilog», временные диаграммы, подтверждающие правильность работы устройства.

1. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Образование циклического кода состоит из двух операций: умножения комбинации обычного двоичного кода $G(X)$ на одночлен X^m и последующего деления этого произведения на выбранный образующий многочлен $P(X)$. Полученные в остатке от деления контрольные символы приписываются к кодируемой комбинации. Таким образом, кодирующее устройство должно совмещать функции умножения и деления. Рассмотрим на конкретном примере методику построения кодирующего устройства. Предположим, что требуется составить схему кодирующего устройства для образующего многочлена $P(X) = X^4 + X^3 + 1$. Схема делителя на этот многочлен приведена на рис. 1, , а умножителя — на рис. 2.

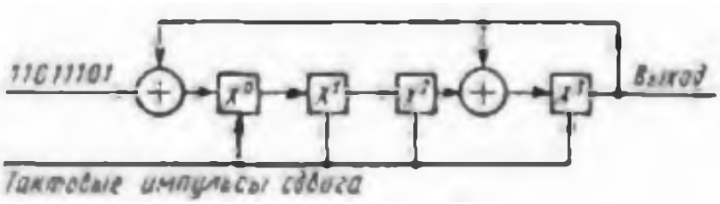


Рис. 1. Схема деления многочлена на многочлен $X^4 + X^3 + 1$

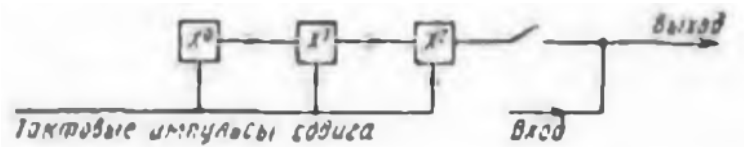


Рис. 2. Схема для умножения многочлена на многочлен $X^4 + X^3 + 1$

Разница в этих схемах состоит лишь в расположении одного сумматора и связей, поступающих с ячеек регистра на сумматоры. На рис. 3 дана схема, объединяющая обе эти схемы (цепь считывания не показана).

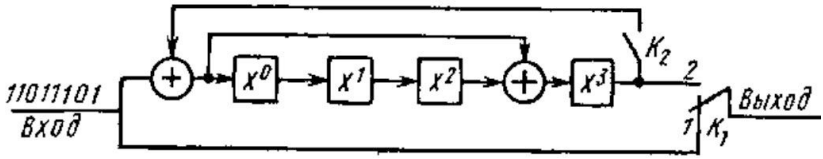


Рис. 3. Образование циклического кода с помощью образующего многочлена $X^4 + X^3 + 1$

По расположению ячеек сумматоров схема кодирующего устройства полностью повторяет схему делителя. Однако сумматор, стоящий на входе, как бы объединяет сумматоры перед ячейкой X^0 в схеме делителя (рис. 1) и после ячейки X^3 в схеме умножителя (рис. 2). Кроме того, обратная связь с выхода на оба сумматора показывает, что в схеме осуществляется деление, а прямая связь с сумматора на входе на сумматор перед ячейкой X^3 свидетельствует о том, что в схеме происходит умножение на одночлен X^3 (в общем случае на X^m).

Схема работает таким образом. В начале работы ключ K_1 находится в положении 1, ключ K_2 замкнут и все подлежащие кодированию информационные символы k, начиная со старшего разряда поступают одновременно, как и при умножении, на выход и через сумматор на входе в схему кодирования. После того как пройдёт последний символ k, ключ K_1 переключается в положение 2, ключ K_2 размыкается, регистр делает m шагов, равных числу ячеек, и весь остаток $R(X) = G(X)X^m/P(X)$ поступает на выход. Этот остаток представляет собой контрольные символы m, следующие за символами k.

Процесс кодирования комбинации $G(X) = X^7 + X^5 + X^4 + X^3 + X + 1$ (двоичный код 10111011) с помощью кодера на рис. 3, а показан в табл. 1. В такте *I* единица кодируемого записывается в ячейки X^0, X^3 и поступает на выход. Хотя в такте *II* на вход поступает нуль, единица с ячейки X^3 через сумматор снова записывается в те же ячейки, а в ячейку X^1 переходит единица с ячейки X^0. Дальнейший процесс кодирования ясен из табл. 1.

После такта *VIII* остаток $R(X)$ оказывается записанным в ячейках регистра. После переключения ключа K_1 в положение 2 и выключения ключа K_2 этот остаток в последующие четыре такта переписывается на выход вслед за информационными символами.

Таблица 1

Номер такта	Вход двоичного кода	Состояние ячеек регистра X^0	X^1	X^2	X^3	Выход циклического кода
		0	0	0	0	
I	1	1	0	0	1	1
II	0	1	1	0	1	0
III	1	0	1	1	0	1
IV	1	1	0	1	0	1
V	1	1	1	0	0	1
VI	0	0	1	1	0	0
VII	1	1	0	1	0	1
VIII	1	1	1	0	0	1
IX		0	1	1	0	0
X		0	0	1	1	0
XI		0	0	0	1	1
XII		0	0	0	0	1

На рис. 4 представлена схема кодирующего устройства для того же многочлена $P(X) = X^4 + X^3 + 1$, что и схема рис. 3, но с сумматорами, расположенными, как в схеме умножителя на рис. 2. Однако обратная связь с выхода на вход выполнена, как в делителе. Обе схемы выполняют одни и те же функции, что можно проверить по табл. 1.

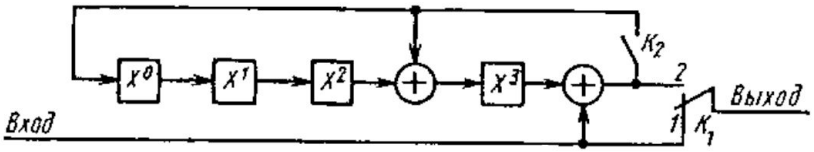


Рис. 4. Второй вариант образования циклического кода с помощью многочлена $X^4 + X^3 + 1$

На рис. 5 приведены примеры кодирующих устройств для других образующих многочленов. При кодировании комбинации 1100111 (рис. 5, б) образуется код 110011101101, при кодировании комбинации 1000101 (рис. 5, в) — код 100010101011. Схема кодера для многочлена $P(X) = X^6 + X^5 + X^2 + X + 1$ представлена на рис. 5, г. Кодирование комбинации 110110110 даёт циклический код 110110110010000.

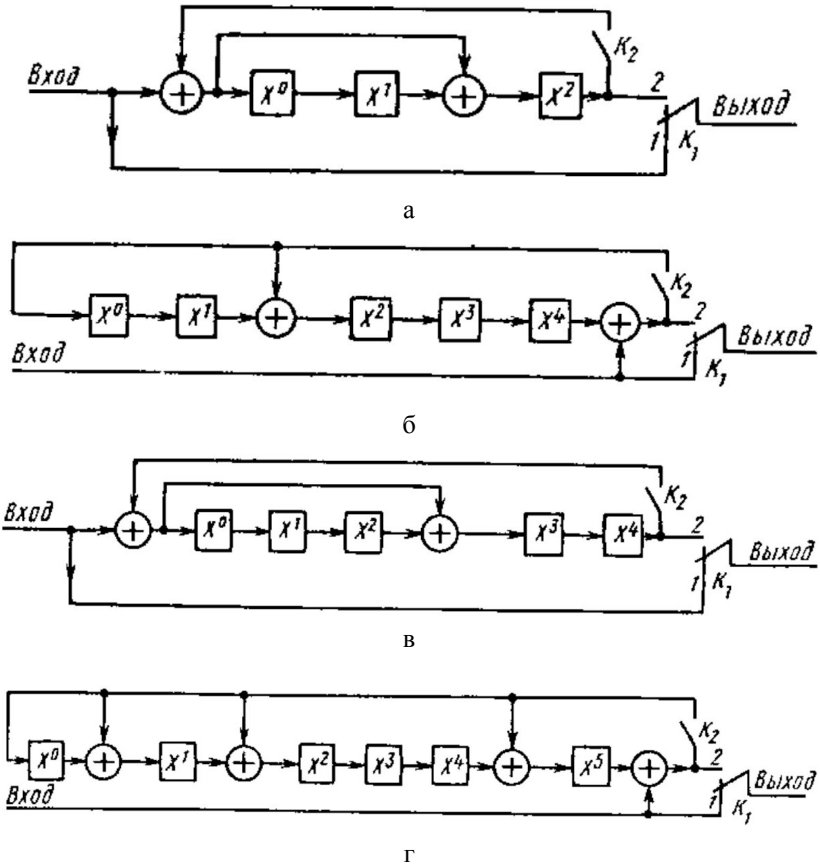


Рис. 5. Образование циклического кода с помощью образующих многочленов: а) $X^3 + X^2 + 1$, б) $X^5 + X^2 + 1$, в) $X^5 + X^3 + 1$, г) $X^6 + X^5 + X^2 + X + 1$

Структурная схема кодирующего устройства для образующего многочлена $P(X) = X^4 + X^3 + 1$ приведена на рис. 6. Здесь кодер и ключи K_1, K_2 аналогичны таким же на рис. 3. С помощью счётчика производятся подсчёт числа информационных и контрольных символов и переключение ключей K_1, K_2. Через ключ K_3 происходят включение счетчика и установка его в исходное состояние. Из функциональной схемы (рис. 6, б) следует, что ключ K_1 реализуется схемой $И_3$, а ключ

K_2 — схемами $И_1$ и $И_2$. На вход кодера комбинация двоичного кода подается начиная с единицы старшего разряда, которая одновременно поступает на регистр и сумматоры, проходит через схему $И_2$ на выход и переключает триггер $У$, вследствие чего имнульс с генератора $Г$ через схему $И_5$ начинает переключать счётчик, состоящий из триггеров $Т_8 – Т_6$. Так как схема $И_3$ в исходном состоянии кодера открыта, в регистре с сумматорами осуществляются кодирование и нахождение остатка. В нашем примере кодируемая комбинация состоит из восьми символов, поэтому, когда счётчик сосчитает до восьми, открывается схема $И_7$ и переключает триггер $Т_2$, что закрывает схемы $И_2$, $И_3$ и открывает схему $И_7$ в результате образованные в коде четыре контрольных символа начинают поступать на выход. После прохождения четырёх контрольных символов в такте *XII* открывается схема $И_6$, и переключает триггер $Т_1$ в исходное состояние; схема $И_5$ перестаёт пропускать импульсы на счётчик, а через схему $И_4$ триггеры $Т_8 – Т_6$ устанавливаются в исходное состояние.

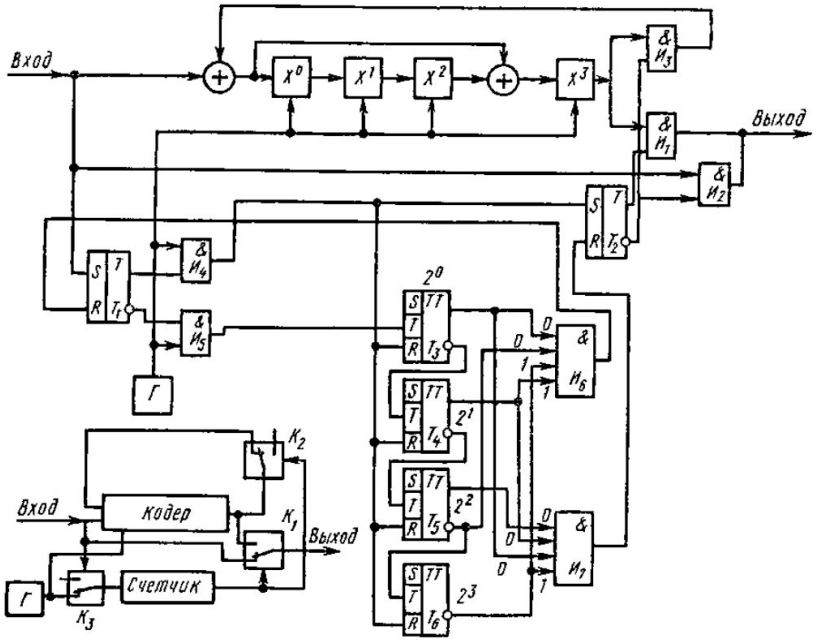


Рис. 6. Схемы образования циклического кода:
а) структурная, б) функциональная

Кодирующее устройство для кодов Боуза — Чоудхури и Файра составляется аналогично. Необходимо лишь знать образующий многочлен (методику его нахождения см. в [1]).

Декодирование циклического кода в этом случае заключается в делении принятой комбинации на заранее известный образующий многочлен. Если при делении остаток отсутствует, то это означает, что кодовая комбинация принята неискаженной. Наличие остатка свидетельствует о ее искажении.

Таким образом, декодирующее устройство должно состоять из делителя и схемы памяти (рис. 7). На вход декодера подается вся кодовая комбинация, состоящая из информационных и контрольных символов. Сначала поступают информационные символы. Они записываются в регистр памяти, который имеет число ячеек, равное числу информационных символов, и одновременно поступают в регистр деления. После прихода всех информационных символов ключ K_1 размыкается. В регистр деления продолжают поступать контрольные символы. Если в принятой кодовой комбинации отсутствуют ошибки, то в регистре деления записываются одни нули. Наличие в той или иной ячейке регистра деления единицы свидетельствует об ошибке. Если схема предназначена только для обнаружения ошибок, то информация в регистре памяти стирается. Если необходимо исправить ошибку, регистр деления продолжает переключаться и номер шага, на котором в первой ячейке регистра появится единица, а на остальных — нули, укажет, в каком месте комбинации появилась ошибка.

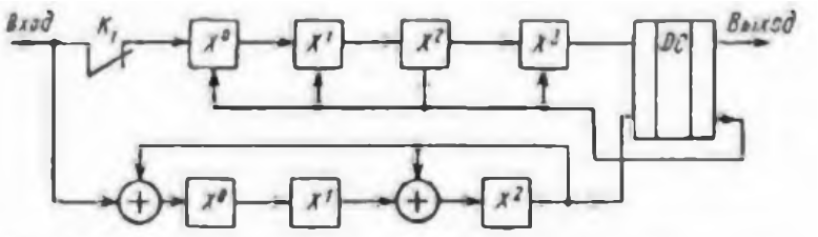


Рис. 7. Функциональная схема декодирования циклического кода

Далее рассмотрим примеры декодирования.

Пример 1. Пусть закодирована комбинация с помощью образующего многочлена $P(X) = X^3 + X^2 + 1$ (двоичный код 1101). Поступившая на декодирующее устройство комбинация имеет вид 1011100. Схема декодирования приведена на рис. 7. Регистр деления выполнен по правилам, изложенным в [1]. Процесс деления показан в табл. 2. Так как

нас интересует только остаток от деления, а не частное, последнее в таблице не приводится. Остаток в данном случае равен нулю, что свидетельствует об отсутствии ошибки.

Таблица 2

Номер такта	Вход кода	Состояние ячеек регистра		
		X^0	X^1	X^2
		0	0	0
I	1	1	0	0
II	0	0	1	0
III	1	1	0	1
IV	1	0	1	1
V	1	0	0	0
VI	0	0	0	0
VII	0	0	0	0

Пример 2. Пусть комбинация была искажена и на вход декодера она поступила в виде 1101100 (табл. 3). Остаток 100 указывает на наличие искажения, и дешифратор ошибок дает команду на стирание записанной комбинации. Число единиц в остатке не указывает на число ошибок. Действительно, в данном случае произошла двойная ошибка, но в остатке имелась одна единица. В то же время при одиночной ошибке (поступила комбинация 1001100 вместо 1011100, табл. 4) в остатке оказываются три единицы (111).

Таблица 3

Номер такта	Вход кода	Состояние ячеек регистра		
		X^0	X^1	X^2
		0	0	0
I	1	1	0	0
II	1	1	1	0
III	0	0	1	1
IV	1	0	0	0
V	1	1	0	0
VI	0	0	1	0
VII	0	0	0	1

Таблица 4

Номер такта	Вход кода	Состояние ячеек регистра		
		X^0	X^1	X^2
		0	0	0
I	1	1	0	0
II	0	0	1	0
III	0	0	0	1
IV	1	0	0	1
V	1	0	0	1
VI	0	1	0	1
VII	0	1	1	1
I		1	1	0
II		0	1	1
III		1	0	0

Если передается код с $d = 3$ и предполагается, что имеется одиночная ошибка, то с помощью того же делителя на рис. 7 можно определить её местоположение. Рассмотрим пример.

Пример 3. Пусть принята комбинация 1001100. После декодирования в ячейках регистра был обнаружен остаток 111. Это показано в такте *VII* табл. 4, который представляет собой последний такт деления Дальнейшая работа регистра происходит с теми же обратными связями до тех пор, пока в первой ячейке регистра не будет записана единица, а в остальных — нули. Как следует из табл. 4, это произошло в такте *III* после окончания деления, что свидетельствует о наличии ошибки в третьем символе, считая со старшего разряда. Поэтому была послана комбинация 1011100, а не 1001100. Дешифратор производит исправление ошибки, и код поступает на выход. Точно так же обнаруживаются ошибки, если они произошли в контрольных разрядах.

Рассмотрим схемную реализацию декодера циклического кода, способного выполнять обнаружение и исправление нескольких ошибок. Пусть имеется комбинация 100000011101000, искаженная двумя ошибками и принявшая вид 111000011101000. Декодер (рис. 8) состоит из делителя, выполненного для деления на многочлен $P(X) = X^8 + X^7 + X^6 + X^4 + 1$ и запоминающего устройства, представляющего собой регистр с сумматором символов k. Комбинация поступает одно-

временно на делитель и запоминающее устройство начиная со старшего разряда. Искаж`нные символы в комбинации отмечены точками. Вначале ключ K_1 замкнут, а ключ K_2 разомкнут. В табл. 5 показан процесс деления начиная с такта *VIII* так как в первые семь тактов происходит заполнение делителя и обратная связь ещё не проявляется.

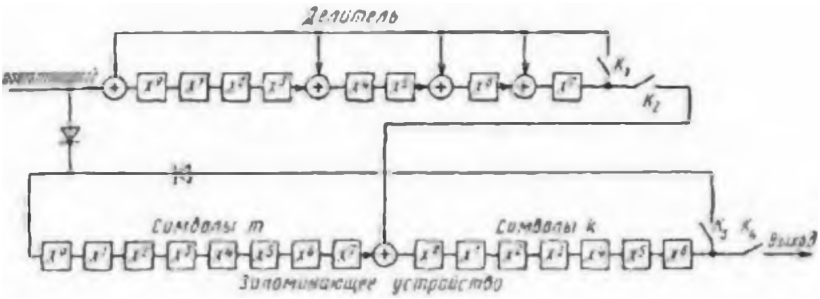


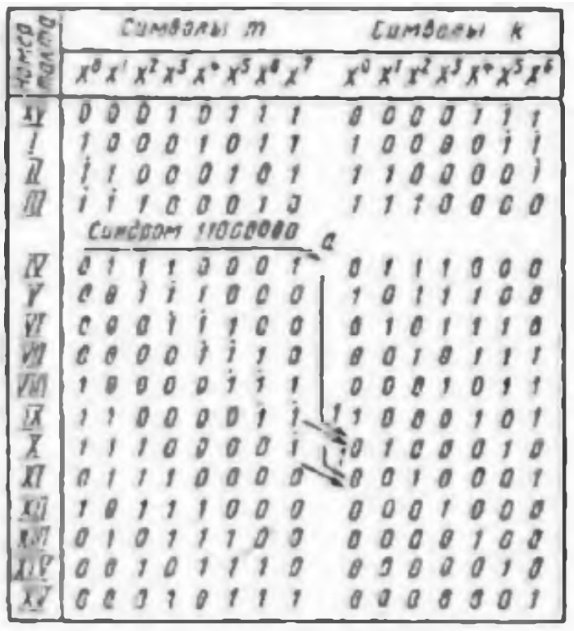
Рис. 8. Функциональная схема декодирования циклического кода с исправлением нескольких ошибок

Таблица 5

Номер такта	Делимое	Состояние ячеек делителя								Вес остатка
		X^0	X^1	X^2	X^3	X^4	X^5	X^6	X^7	
VIII	1	1	0	0	0	0	i	i	1	
IX	1	0	1	0	0	1	0	0	0	
X	1	1	0	1	0	0	1	0	0	
XI	0	0	1	0	1	0	0	1	0	
XII	1	1	0	1	0	1	0	0	1	
XIII	0	1	1	0	1	1	1	1	1	
XIV	0	1	1	1	0	0	1	0	0	
XV	0	0	1	1	1	0	0	1	0	4
I		0	0	1	1	1	0	0	1	4
II		1	0	0	1	0	1	1	1	5
III		1	1	0	0	0	0	0	0	2

В такте *XV* синдром (остаток от деления) оказывается записанным в ячейках регистра (01001110). Однако его вес $W = 4$ больше числа исправляемых ошибок поэтому делитель делает еще один шаг (такт *I*), в процессе которого снова осуществляется деление на многочлен $P(X)$ Синдром 10011100 опять имеет вес $W = 4$. Только после третьего такта $W = 2 = s$. В этот момент ключ K_1 размыкается, а ключ K_2 замыкается и синдром с делителя начинает поступать на сумматор запоминающего устройства, у которого ключ K_3 замкнут, а ключ K_2 разомкнут. Это устройство в такте *XV* первого этапа полностью заполнилось, а на втором этапе его работы начался циклический сдвиг записанной информации (табл. 6).

Таблица 6



Так в такте *I* единица из ячейки X^6 информационных символов переместилась в ячейку X^0 контрольных символов *m*. В такте *II* эта единица передвинулась в ячейку X^1, а её место в ячейке заняла следующая единица и т. д. Первые шесть пулей синдрома, поступающие на сумматор, не влияют на работу запоминающего устройства. Лишь в тактах *X* и *XI* две единицы синдрома, складываясь по модулю 2 с двумя ошибочными единицами символов *k* (обозначены точками), «уничтожают» их, т. е. исправляют ошибки. Регистр запоминающего устройства про-

должает переключаться до окончания второго цикла (этапа) его работы. После такта *XV* ключи K_2 и K_3 размыкаются, а ключи K_1 и K_4 замыкаются: начинается считывание исправленной комбинации и одновременная запись новой.

Таким образом, декодирование состоит их двух этапов. На первом этане осуществляются нахождение остатка и запись кодовой комбинации, на втором — её исправление и расстановка символов *k* и *m* на свои места.

2. РАЗРАБОТКА МОДЕЛИ ЦИФРОВОЙ СИСТЕМЫ

Элементарной ячейкой электронной памяти является триггер, способный сохранять 1 бит записанной в нем информации, однако часто требуется запоминать двоичные слова большей разрядности, для этого используются регистры.

Регистром называется устройство из нескольких триггеров, предназначенное для записи, хранения и выдачи информации. Каждый разряд двоичного числа записывается в отдельном триггере, поэтому число триггеров в регистре определяет разрядность записываемого числа.

Ввод и вывод информации могут выполняться параллельным или последовательным кодом с помощью параллельных или последовательных входов и выходов. Поэтому существуют различные типы регистров: параллельные, параллельно-последовательные, регистры сдвига.

Регистры обычно строятся на основе D-триггеров, которые могут быть с динамическим управлением по фронту или статическим управлением — по уровню.

Параллельный регистр

Параллельный регистр служит для запоминания двоичного слова. Количество триггеров, входящее в состав параллельного регистра определяет его разрядность.

Регистр с параллельной загрузкой строится на D-триггерах. К входам триггеров D подключается шина входных данных, а к выходам Q — шина выходных данных. При записи информации в параллельный регистр все двоичные разряды записываются одновременно, поэтому входы синхронизации всех триггеров соединяют параллельно. Запись данных в регистр происходит при поступлении импульса синхронизации C.

Схема четырёхразрядного параллельного регистра, построенного на D-триггерах с динамическим управлением, приведена на рис. 9.

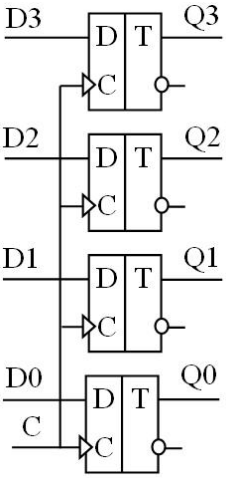


Рис. 9. Схема параллельного регистра

Запись данных в регистр с динамическим управлением происходит по переднему фронту импульса.

Параллельный регистр может быть построен на базе D-триггеров со статическим входом (в обозначении входов С будут отсутствовать треугольники), его называют регистр — защёлка (рис. 10, а). Этот регистр будет иметь прозрачный режим, что используется в некоторых схемах.

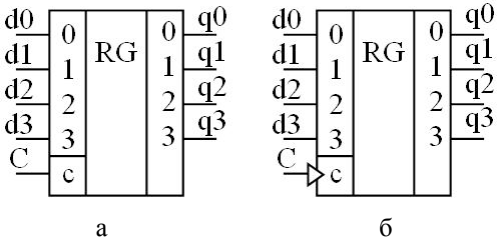


Рис. 10. Обозначения параллельных регистров:
а) статического, б) динамического

Существуют другие варианты схемы схем параллельных регистров. В интегральном исполнении выпускаются буферные регистры, имеющие мощные формирователи выходных сигналов с тремя состоя-

ниями выхода, или типа открытый коллектор. Они предназначены для работы на внешнюю нагрузку и для подключения устройств к общим шинам.

Описания рассмотренных регистров на языке «Verilog» несложно получить из описаний статического и динамического D-триггеров. В описаниях регистров, как и в описаниях триггеров использованы последовательные операторы с ключевым словом «always», выходные сигналы описаны как тип «reg». Отличие данных описаний в разрядности сигналов «d» и «q». Здесь они описаны как 4-разрядные векторы (параллельные коды).

```
// Пример 1. Статический     //1
// параллельный регистр      //2
module reg_latchs (c, d, q);  //3
input c;                      //4
input [3:0] d;                //5
output [3:0] q;               //6
reg  [3:0] q;                 //7
always  if (c)  q = d;        //8
endmodule                     //9

// Пример 2. Динамический //1
// параллельный регистр      //2
module reg_latchd (c, d, q);  //3
input c;                      //4
input [3:0] d;                //5
output [3:0] q;               //6
reg  [3:0] q;                 //7
always @ (posedge c) q = d;   //8
endmodule                     //9
```

Поведение регистра-защелки описывает строка 8 в примере 1, которая имеет смысл: «Всегда, если c = 1 выполнять присваивание q := d». Подобная строка в примере 2, описывающим динамический регистр описывает срабатывание по фронту и имеет смысл: «Всегда по событию — положительный фронт сигнала «c» выполнять присваивание q := d».

Сдвигающие регистры

Сдвигающий регистр —- цифровая схема, состоящая из последовательно включенных триггеров, содержимое которых можно сдвигать

на один разряд влево или вправо подачей тактовых импульсов. Сдвигающий регистр обычно служит для преобразования последовательного кода в параллельный код и наоборот.

Сдвигающий регистр, схема которого приведена на рис. 11, имеет последовательный вход SI («Serial Input») и параллельный выход Q3 – Q0. Кроме того, выход Q0 можно использовать как последовательный. Соединение триггеров между собой, когда сигнал передается с выхода старшего разряда на вход младшего разряда, позволяет получить сдвиг вправо.

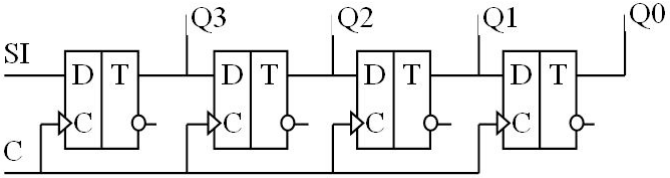


Рис. 11. Сдвигающий регистр

Работу схемы поясняют временные диаграммы (рис. 12).

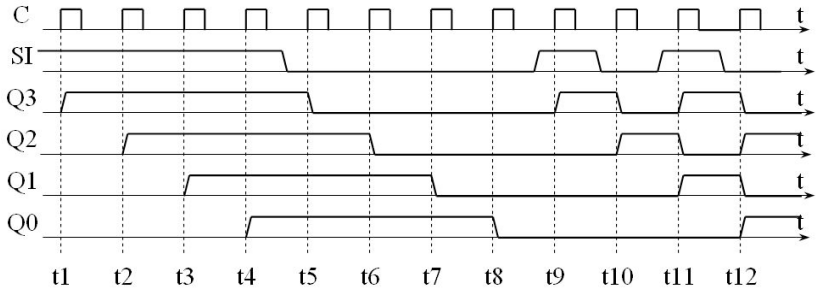


Рис. 12. Временные диаграммы, поясняющие работу сдвигающего регистра

На вход SI поступает последовательный код, значения которого будут определяться в моменты времени, соответствующие фронтам синхроимпульсов на входе C. По фронту первого синхроимпульса (момент t1) произойдёт запись входного сигнала регистра SI = 1 в триггер старшего разряда: Q3 := 1. Изменение сигнала Q3 произойдёт с задержкой во времени относительно фронта сигнала C. В момент фронта данного синхроимпульса сигнал Q3 остается равным нулю. Сигнал C подается параллельно на все триггеры. На входах всех триг-

геров, кроме первого к моменту поступления фронта первого импульса С присутствует сигнал, равный нулю, поэтому в момент t1 устанавливается сигнал Q3 := 1, а выходные сигналы остальных триггеров сохраняются нулевыми: Q2 := Q1 := Q0 := 0. После первого синхроимпульса в регистре с задержкой установится код 1000.

К моменту появления фронта второго импульса (момент t2) входные сигналы первого и второго триггеров равны единице. Первый триггер останется в состоянии 1 (Q3 := 1), следующий триггер установится в 1 (Q2 := 1) . Остальные триггеры остаются в нулевом состоянии (Q1 := Q0 := 0). После второго синхроимпульса в регистре будет установлен код 1100.

Каждый синхроимпульс записывает 4 разряда кодов: входной сигнал SI и коды, содержащиеся в разрядах Q3, Q2, Q1 в разряды Q3, Q2, Q1, Q0 соответственно.

После момента времени t4 на входе SI было зафиксировано 4 единичных значения, поэтому код регистра после действия указанного синхроимпульса будет равен 1111. В моменты времени t5, t6, t7, t8 подаются 4 нулевых значения сигнала SI, поэтому после действия последнего синхроимпульса в регистре будет установлен код 0000. На интервале t9 – t12 сигнал SI принимает последовательно значения 1010. Указанный последовательный код записывается в сдвигающий регистр младшими разрядами вперед. После синхроимпульса в момент t12 в регистре получен код 0101.

Анализ работы сдвигающего регистра позволяет определить последовательность действий для преобразования последовательного кода в параллельный.

До записи числа все триггеры регистра устанавливают в нулевое состояние. Затем на вход схемы SI подается серия импульсов, соответствующая записываемому числу, а на вход С подаются тактовые импульсы. Сначала на вход SI поступает импульс, соответствующий младшему из записываемых разрядов. В конце тактового импульса он даёт Q3 = 1 на выходе левого (старшего) триггера. В конце следующего тактового импульса указанный информационный импульс продвигается на выход следующего триггера. Одновременно продвигаются вправо и другие цифры записываемого числа. После прихода четырех тактовых импульсов все число оказывается записанным в четырех триггерах, причем старший разряд числа записи в левом триггере, а младший — в правом. Чтобы записанная информация сохранилась, дальнейший сдвиг прекращается. Это осуществляется прекращением подачи тактовых импульсов.

Рассмотренный сдвигающий регистр позволяет осуществить последовательный прием информации, а выдача информации может быть как параллельной, так и последовательной. При параллельной выдаче информация снимается одновременно с выходов всех триггеров. Последовательная выдача осуществляется с выхода Q0 в моменты действия тактовых.

```
// Пример 3. Сдвигающий    //1
// регистр                 //2
module shift_reg (c, si, q); //3
input c, si;               //4
output [3:0] q;            //5
reg  [3:0] q;              //6
always  @ (posedge c)
   q = {si,q[3],q[2],q[1]}; //7
endmodule                  //8
```

В описании сдвигающего регистра на языке «Verilog» представляет интерес строка 7, описывающая работу регистра. В соответствии с этой строкой всегда по фронту сигнала «c» должен выполняться оператор объединения векторов, который записывается как перечисление исходных векторов в фигурных скобках через запятую. В соответствии с данным оператором новое значение кода, содержащееся в регистре Q в виде последовательности двоичных разрядов q3 q2 q1 q0, должно быть получено из предыдущих значений сигналов si, q3, q2, q1. Предыдущее значение q0 теряется.

Универсальный сдвигающий регистр

Регистры сдвига изготавливаются обычно как универсальные, имеющие параллельные и последовательные входы и выходы, что позволяет преобразовывать не только последовательный код в параллельный, но и выполнять обратное преобразование параллельного кода в последовательный. Это свойство универсального последовательно - параллельного регистра используется, например, при реализации последовательного порта в микропроцессорной системе.

Универсальный сдвигающий регистр (рис. 13) имеет шину входного параллельного кода (D), шину выходного параллельного кода (Q), вход последовательного кода (SI — Serial Input), выход последовательного кода (SO — Serial Output). Входной сигнал каждого триггера формирует мультиплексор «2 в 1».

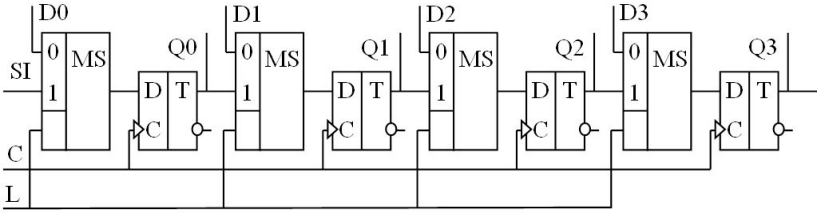


Рис. 13. Универсальный сдвигающий регистр

При разрешении параллельной загрузки (L = 1), входы триггеров подключаются к проводникам входной шины данных, и по положительному фронту синхроимпульса «C» произойдет запись входного кода в регистр.

При запрещении параллельной загрузки (L = 0) по положительному фронту синхроимпульса «C» произойдет циклический сдвиг влево: Q0 := SI; Q1 := Q0; Q2 := Q1; Q3 := Q2.

Универсальный сдвигающий регистр используют для преобразования параллельного кода в последовательный и наоборот.

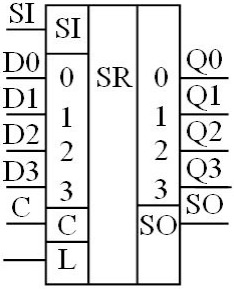


Рис. 14. Обозначение универсального сдвигающего регистра

Для преобразования параллельного кода в последовательный необходимо подать преобразуемый код на входы данных D, а сигнал параллельной загрузки — на вход L, а затем подать *n* тактовых импульсов на вход C, где *n* — количество разрядов. Последовательный код будет выдан с выхода SO.

Если на вход SI подавать «0», или «1», то регистр заполнят соответствующие константы. Можно соединить SI с SO, в результате за *n*

тактовых импульсов произойдет полный кольцевой сдвиг, и содержимое регистра сохранится.

Рассмотрим описание универсального сдвигающего регистра (пример 4). Начало описания выполнено по аналогии с предыдущими примерами. В строке 3 перечислены все сигналы в соответствии с обозначением регистра (рис. 14), в строке 4 описаны одноразрядные выходные сигналы, а в строке 5 — вектор. Аналогично в строках 6 – 8 описаны выходные сигналы, при этом вектор q указан как регистр, что необходимо при использовании последовательного оператора.

```
// Пример 4. Универсальный                    //1
// сдвигающий регистр                         //2
module univer_shift_reg (c, si, d, l, q, so);  //3
input c, si, l;                                //4
input [3:0] d;                                 //5
output so;                                     //6
output [3:0] q;                                //7
reg  [3:0] q;                                  //8
always  @ (posedge c) if (l) q=d ;
else q = {si,q[3],q[2],q[1]};                  //9
assign so=c & q[3];                            //10
endmodule                                      //11
```

Описание работы регистра содержит строка 9, разделенная на две части для удобства записи, содержащая последовательный условный оператор «if», записанный с ключевым словом «always», выполняющийся по фронту сигнала синхронизации «C».

В качестве условия оператора «if » записан сигнал «l». Если l = 1, то выполняется параллельная загрузка, описанная присваиванием q := d, в котором все переменные — 4-разрядные векторы. Если l = 0, то выполняется сдвиг, описанный после слова «else» аналогично примеру 3.

3. МОДЕЛИРОВАНИЕ РАЗРАБОТАННОЙ СИСТЕМЫ С ИСПОЛЬЗОВАНИЕМ САПР

1. Запустить программу «Quartus II».
2. Создать рабочую папку, в которой будет размещаться проект (например, «D:\919М\КР»).
3. Создать проект: File->New Project Wizard. В появившемся диалоговом окне указать путь к рабочей папке («D:\919М\КР») и строкой ниже — имя проекта (например, «Project_1»). Остальные на-

стройки можно оставить без изменения. В результате, в рабочей папке должен появиться файл проекта («Project_1.qpf») и другие вспомогательные файлы и папки.

4. Создать файл графического описания схемы: File->New->Design Files->Block Diagram/ Schematic File. Сохранить этот файл File -> Save As. При первом сохранении файла графического описания схемы можно дать ему любое имя, по умолчанию файл предлагается назвать именем «Block1.bdf», но имя главного файла проекта должно совпадать с именем проекта: если проект назван «Project_1.qpf», то файлу графического описания следует дать имя «Project_1.bdf». Имена создаваемых модулей должны совпадать с именами файлов, в которых эти модули описаны.

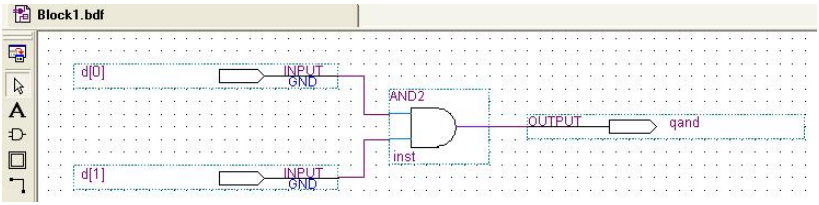


Рис. 15. Пример простейшего проекта

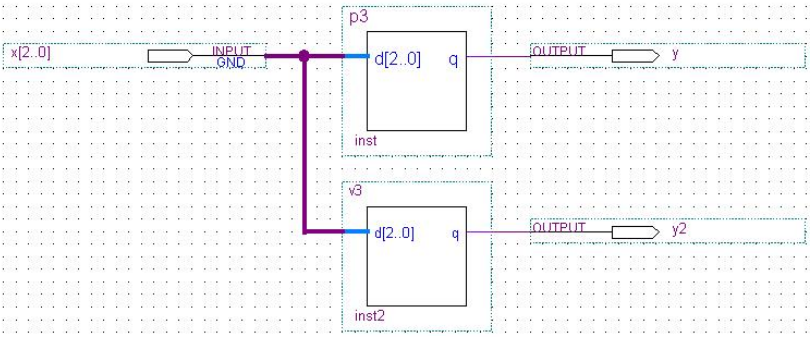


Рис. 16. Примеры описания схемы с использованием подсистем

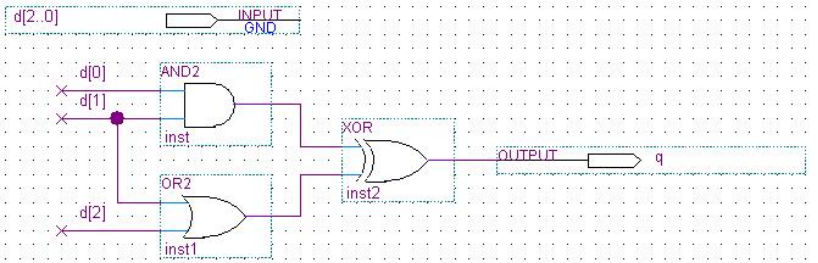


Рис. 17. Пример описания схемы с векторными сигналами

5. Вызвать навигатор проекта «Project Navigator», воспользовавшись меню MAX+PLUS II -> Hierarchy Display. Открыть закладку «Files»: на ней отображаются все файлы, добавленные в проект. Созданный ранее файл «Project_1.bdf» добавляется в проект автоматически. Для принудительного добавления новых файлов в проект или удаления файлов из проекта следует использовать меню Project -> Add/Remove Files in Project или контекстное меню, которое открывается при нажатии правой кнопки мыши над пиктограммой «Files» в окне навигатора проекта «Project Navigator».
6. Начертить схему проектируемого устройства. Варианты заданий приводятся в табл. 7. В каждом варианте требуется собрать схему кодера и схему декодера. Сначала достаточно создать схему кодера и убедиться, что поступающему на его вход двоичному сообщению соответствует на выходе правильная кодовая комбинация (информационная часть должна совпадать с поступившим сообщением, вся кодовая комбинация должна делиться без остатка на образующий многочлен).

Таблица 7

Номер варианта	Образующий многочлен
1	$x^5 + x^4 + x^3 + x^2 + 1$
2	$x^5 + x^4 + x^3 + x + 1$
3	$x^5 + x^4 + x^2 + x + 1$
4	$x^5 + x^3 + x^2 + x + 1$
5	$x^5 + x^3 + 1$
6	$x^5 + x^2 + 1$
7	$x^6 + x^5 + 1$
8	$x^6 + x + 1$
9	$x^4 + x^3 + 1$
10	$x^4 + x + 1$

7. На выходе кодера установить искажающее устройство, вносящее одиночные ошибки в сообщение. Для этого правильную кодовую комбинацию следует сложить (сумма по модулю 2) с искажающей последовательностью, которая содержит все нули и только в одном такте содержит единицу (в результате в образованной кодером правильной кодовой комбинации изменится один двоичный разряд).
8. После того как получена схема, содержащая кодер и искажающее устройство, требуется дополнить схему декодером, способным обнаруживать ошибки (рис. 18).

Рис. 18. Функциональная схема системы кодирования-декодирования

9. Выполнить декодирование правильной кодовой комбинации (без искажений) и декодирование кодовой комбинации, содержащей один искажённый разряд. Убедиться, что при декодировании правильной кодовой комбинации остаток от деления на образующий многочлен равен нулю, а в случае декодирования искажённой кодовой комбинации остаток нулю не равен.
10. Сохранить для отчёта полученные временные диаграммы (рис. 19).

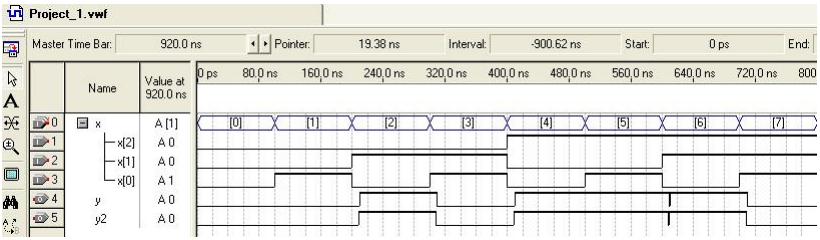


Рис. 19. Пример временных диаграмм

11. Описать кодер и декодер на языке «Verilog».

4. ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ПРОГРАММНОЙ И КОНСТРУКТОРСКОЙ ДОКУМЕНТАЦИИ

Разработка программного обеспечения и оформление отчёта должны выполняться в соответствии с требованиями государственных стандартов (ГОСТ). В настоящее время действуют следующие стандарты документирования:

- ГОСТ 19.201 — Единая система программной документации (ЕСПД);
- ГОСТ 2.015-2013 — Единая система конструкторской документации (ЕСКД).

Чертежи и схемы должны быть выполнены на листах стандартных форматов с соблюдением правил, установленных стандартами ЕСКД и ЕСПД. Форматы листов чертежей и других документов, выполненных в электронной и бумажной формах, предусмотренных стандартами на конструкторскую документацию всех отраслей промышленности и строительства, устанавливает ГОСТ 2.301-68.

Согласно ГОСТ 19.102-77 процесс разработки программного обеспечения включает 5 стадий: техническое задание, эскизный проект, технический проект, рабочий проект и внедрение. В данном курсовом проекте выполняются две из этих стадий: технический проект и рабочий проект. На стадии технического проекта уточняются структуры входных и выходных данных, выполняется разработка алгоритма решения задачи (алгоритм должен быть описан в форме блок-схемы), определяются формы представления входных и выходных данных, семантика и синтаксис языка программирования, разрабатывается структура программы, определяется конфигурация технических средств. Стадия выполнения рабочего проекта разбивается на 3 этапа: разработка программы (программирование и отладка), разработка программной документации, испытания программы.

Для описания программно-алгоритмического обеспечения принято использовать блок-схемы. Такие схемы представляют собой чертежи, на которых отдельные функции алгоритмов и программ, отображаются с учётом степени их детализации в виде условных графических обозначений, регламентируемых ЕСПД, а именно: ГОСТ 19.002-80, ГОСТ 19.003-80, ГОСТ 19.701-90. Пример блок-схемы показан на рис. 20. Цикл со счётчиком (оператор for) допускается обозначать символом, состоящим из двух частей, отображающих начало и конец цикла (рис. 21). Условия для инициализации, приращения, завершения помещаются внутри помещаются внутри символа в начале или в конце, в зависимости от расположения операции, проверяющей условие.

Правила выполнения программных документов независимо от их назначения и области применения устанавливает ГОСТ 19.106-78. Размеры шрифтов и стили оформления для текстов программ могут отличаться от текста обычных документов, что позволяет сделать чтение текстов программ более удобным.

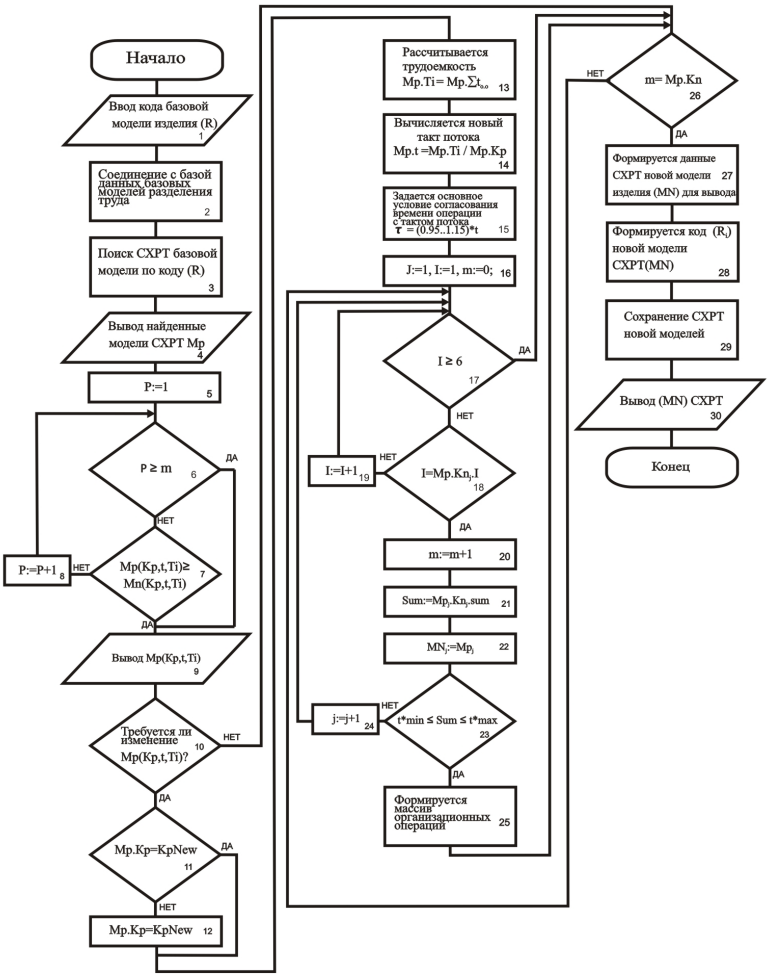


Рис. 20. Пример оформления блок-схемы.

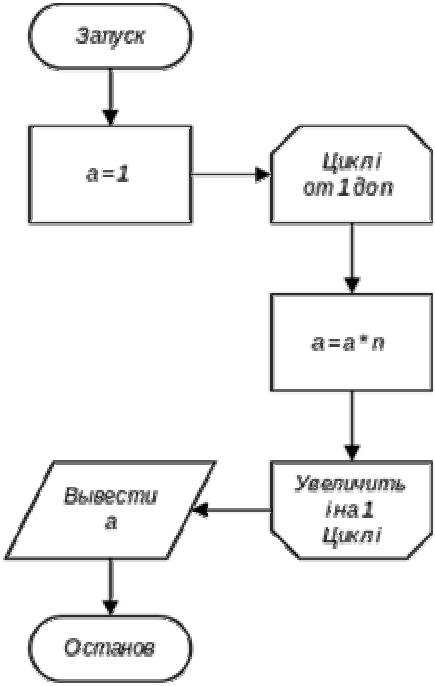


Рис. 21. Пример описания на блок-схеме цикла со счётчиком.

5. ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ ОТЧЁТА

Текст отчёта должен быть представлен в машинописном виде в соответствии с правилами, изложенными в [7, 8] на листах формата А4 (210×297 мм) и размещён на одной стороне листа при вертикальном его расположении, с полями: слева — 30 мм; справа — 10 мм; сверху и снизу — 20 мм. При наборе текста на компьютере необходимо использовать шрифт «Times New Roman» размером 14 пт. (сокращение пт., то есть пункт, — единица определения размера шрифта, в типографской системе мер 1 пункт равен 0,353 мм), выравнивание абзаца по ширине, автоматическую расстановку переносов в словах, полуторный межстрочный интервал, абзацный отступ 1,25 см. Заголовки таблиц, диаграмм и рисунков печатать через один интервал.

Пункты отчёта последовательно нумеруют арабскими цифрами (например, 1, 2 и т.д.), подпункты — двумя арабскими цифрами, разделёнными точкой: первая означает номер соответствующего пункта,

вторая — подпункта. После номеров пунктов и подпунктов точка не ставится. Например: 1.2 — это второй подпункт первого пункта и так далее. Номер пункта или подпункта указывают перед заголовком. Каждый пункт отчёта начинают писать с новой страницы. С новой страницы также пишут приложения, содержание. Заголовки пунктов и подпунктов оформляют без подчеркивания с прописной (заглавной) буквы. Например:

1. Подготовительный этап

1.1 Инструктаж по технике безопасности

Заглавными буквами печатаются аббревиатуры и слова «СОДЕРЖАНИЕ», «ПРИЛОЖЕНИЕ». Текст отчётов печатается строчными буквами. Заголовки пунктов при отсутствии подпунктов отделяются от текста расстоянием снизу 12 пт., подпункты отделяются от текста расстояниями сверху 18 пт., снизу 12 пт. Знаки, символы, обозначения, а также математические формулы могут быть набраны на компьютере или в отдельных случаях вписаны от руки тушью (чернилами, пастой) чёрного цвета.

Все страницы отчёта, включая приложения, нумеруются по порядку от титульного листа до последней страницы без пропусков и повторений. Первой страницей считается титульный лист. На нём цифра «1» не ставится. На следующей странице ставится цифра «2» и т.д. Нумерация страницы выполняется от центра на верхнем поле страницы без всяких дополнительных обозначений: точек, чёрточек, кавычек и так далее.

Библиографический список оформляется также в соответствии с требованиями ГОСТ [9].

6. КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Укажите назначение, устройство, сферы применения ПЛИС.
2. Каково назначение языков описания аппаратных средств?
3. Как выполняется анализ комбинационных схем?
4. Назовите известные вам операторы: арифметические, поразрядные, свёртки, отношения и сравнения, условного присваивания.
5. В чём заключается описание на языке «Verilog»: описание по логическим уравнениям, поведенческое описание.
6. В чём различие между параллельными и последовательными операторами? К каким операторам относится оператор условного перехода «if»?
7. Приведите пример использования оператора варианта «case».
8. Как выполняется анализ логических устройств комбинационного типа?

9. Как выполняется анализ комбинационной схемы, заданной логическими функциями?
10. Перечислите этапы синтеза комбинационных схем.
11. Приведите пример минимизация логических функций.
12. Опишите по шагам процесс минимизации логических функций методом карт Карно.
13. Приведите пример структурного описания по логическому уравнению. Приведите пример поведенческого описания.
14. Приведите пример описания устройств комбинационного типа на примере мультиплексора.
15. Приведите пример описания устройств комбинационного типа на примере шифратора.
16. Приведите пример описания устройств комбинационного типа на примере АЛУ комбинационного типа.
17. Что представляет из себя асинхронный RS-триггер с прямыми установочными входами? С инверсными установочными входами?
18. Что представляет из себя синхронный RS-триггер? Двухступенчатый RS-триггер?
19. Что представляет из себя статический D-триггер? D-триггер с динамическим управлением?
20. Как устроен и как работают JK-триггер и счётный триггер?
21. Как устроен и как работает параллельный регистр?
22. Как устроен и как работает сдвигающий регистр?
23. Как устроены и как работают синхронные и синхронные счётчики? Счётчик с параллельной загрузкой?
24. Как устроен и как работает формирователь заданной последовательности импульсов?
25. Как устроен и как работает генератор псевдослучайной последовательности?
26. Как устроены и как работают распределители импульсов?
27. Расскажите о синхронизации в цифровых устройствах. О принципе однофазной синхронизации.
28. Как выполняется синтез конечного автомата. Приведите порядок синтеза.
29. Поясните основные особенности цифровых автоматов с синхронными и асинхронными выходами.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Тутевич В. Н. Телемеханика: учеб. пособие для студентов вузов спец. «Автоматика и телемеханика». — 2-е изд , перераб. и доп.— М.: Высш. шк., 1985.— 423 с.

2. Кистрин А.В., Никифоров М.Б. Проектирование цифровых устройств: учебник для студентов учреждений среднего профессионального образования. — М.: Академия, 2016. — 282 с.
3. Стешенко В. Плис фирмы «ALTERA». Элементная база, система проектирования и языки описания аппаратуры. — М.: ДМК Пресс, 2016. — 576 с.
4. Максфилд К. Проектирование на ПЛИС. Архитектура, средства и методы. Курс молодого бойца. — М.: Додэка XXI, ДМК Пресс, 2015. — 408 с.
5. Грэхэм М., Джонсон Г. Конструирование высокоскоростных цифровых устройств. Начальный курс черной магии. — М.: Вильямс, 2015. — 624 с.
6. Бибило П. Основы языка VHDL. — М.: Либроком, 2016. — 328 с.
7. Пухальский Г., Новосельцева Т. Проектирование цифровых устройств: учеб. для вузов. — М.: Лань, 2012. — 896 с.
8. ГОСТ 2.105-95. Единая система конструкторской документации (ЕСКД). Общие требования к текстовым документам [Электронный ресурс]. — Введ. 1996-07-01. — Доступ: http://www.konstalin.ru/UserFiles/Files/ESKD/gost_2.105_95.pdf.
9. ГОСТ 7.32-2001. Отчёт о научно-исследовательской работе. Структура и правила оформления [Электронный ресурс]. – Введ. 2002-07-01. – Доступ: http://www.ifap.ru/library/gost/7322001.pdf.
10. ГОСТ 7.1-2003. Библиографическая запись. Библиографическое описание. Общие требования и правила составления [Электронный ресурс]. — Введ. 2004-07-01. — Доступ: http://diss.rsl.ru/datadocs/doc_291wu.pdf.

ОГЛАВЛЕНИЕ