Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Кафедра «Телекоммуникационные системы»
Специальность: 6М071900 «Радиотехника, электроника и телекоммуникации»

ДОПУЩЕН К ЗАЩИТЕ
Зав. кафедрой
к.т.н., профессор Байкенов А.С.
(ученая степень, звание, ФИО)

_____
(подпись)
«_____»_____2016 г.

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ
пояснительная записка

на тему: «Исследование кодирования и декодирования сверточных кодов»

Магистрант: Ибадуллина И.А. _____ группа МТСп-14-1
(Ф.И.О.) (подпись)

Руководитель: к.т.н., доцент _____ Чежимбаева К.С.
(ученая степень, звание) (подпись) (Ф.И.О.)

Рецензент: к.т.н., доцент _____ Орынбет М.М.
(ученая степень, звание) (подпись) (Ф.И.О.)

Консультант по ВТ: ст.преподаватель _____ Демидова Г.Д.
(ученая степень, звание) (подпись) (Ф.И.О.)

Нормоконтроль: ст.преподаватель _____ Кондратович А.П.
(ученая степень, звание) (подпись) (Ф.И.О.)

Алматы 2016

Некоммерческое акционерное общество
«АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ»

Факультет: «Радиотехники и связи»
Специальность: 6М071900 «Радиотехника, электроника и телекоммуникации»
Кафедра: «Телекоммуникационные системы»

ЗАДАНИЕ
на выполнение магистерской диссертации

Магистранту Ибадуллиной И.А.
(фамилия, имя, отчество)

Тема диссертации «Исследование кодирования и декодирования сверточных кодов»

утверждена Ученым советом университета № 141 от «28.10.2014»
Срок сдачи законченной диссертации «___»_____

Цель исследования: разработка схем, программных модулей кодирования и декодирования двоичных сверточных кодов, а также исследование помехоустойчивости сверточных кодов и разработка лабораторных работ по этой теме.

Перечень подлежащих разработке в магистерской диссертации вопросов или краткое содержание магистерской диссертации:

1. Кодирование двоичных сверточных кодов
2. Декодирование сверточных кодов
3. Характеристики помехоустойчивости сверточных кодов

Перечень графического материала (с точным указанием обязательных чертежей)
Рисунок 1 – Структурная схема кодера сверточного кода;
Рисунок 2 – Структурная схема кодера сверточного кода с параметрами m = 3, k = 1, n = 2;
Рисунок 3 – Систематический сверточный кодер;
Рисунок 4 – Структурная схема кодера сверточного кода с параметрами m = 3, k =1, n = 2;
Рисунок 5 – D-триггер;
Таблица 1 – Формирование кодовой последовательности;
Таблица 2 – Таблица переходов состояний кодера (*R=1/2, m = 3*).

Рекомендуемая основная литература:

1. Р. Морелос-Сарагоса. Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение. – М.: Техносфера, 2005. – 320 с.
2. Блейхут Р. Теория и практика кодов, контролирующих ошибки: Пер. с англ. – М.: Мир, 1986. – 576 с.
3. М. Вернер. Основы кодирования. Учебник для вузов/ М.: Техносфера, 2004. – 288 с.

ГРАФИК
подготовки магистерской диссертации

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю	Примечание
1. Информационный обзор сверточных кодеров	30.09.2014	
2. Изучение основных параметров сверточных кодеров	03.11.2014	
3. Описание среды Altera QUARTUSII	02.11.2015	
4. Декодирование сверточных кодов	30.11.2015	
5. Характеристики помехоустойчивости сверточных кодов	30.12.2015	

Дата выдачи задания_____

Заведующий кафедрой _____ (Байкенов А.С.)
(подпись) (Ф.И.О.)

Руководитель диссертации _____ (Чежимбаева К.С.)
(подпись) (Ф.И.О.)

Задание принял к исполнению
магистрант _____ (Ибадуллина И.А.)
(подпись) (Ф.И.О.)

Аннотация

Сверточные коды обладают высокими корректирующими свойствами. Они применяются в различных системах помехоустойчивой передачи информации, в системах сотовой и космической связи. При выполнении работы были использованы методы и алгоритмы теории помехоустойчивого кодирования-декодирования. Разработаны принципиальные электрические схемы кодеров в среде QUARTUSII. Сделанные виртуальные лабораторные работы являются хорошим «тренажером» при освоении теории и практики по сверточным кодам.

Аңдатпа

Түзетуші кодтар жоғары түзетушілік қасиеттерге ие. Олар әр түрлі кедергіге қарсы тұратын ақпараттарды жіберу жүйелерінде, ұялы және ғарыштық жүйелерде қолданылады. Жұмысты атқару барысында кедергіге қарсы тұратын кодтау-қайта кодтау теорияларының алгоритмдері мен тәсілдері қолданылды. QUARTUSII ортасында кодерлердің принциптік электрлік сұлбалары құрылды. Құрастырылған виртуалы лабораториялық жұмыстар түзетуші кодтардың дәрісі мен тәжірибесін меңгеру кезінде жақсы жаттығу құрылғысы табылады.

Annotation

Conventional codes have high correction reliability. They are implemented in a variety of noise-immune information transmission systems, in cellular systems and space communications. Presented dissertation is based on methods and algorithms of the theory of error-correcting coding. Circuit diagrams have been designed on QUARTUS II software. Virtual labs presented as part of the paper could be used as useful training material for theoretical and practical learning of conventional codes.

Содержание

Введение

Актуальность темы диссертации.

Тема работы - кодирование и декодирование сверточных кодов.

Настоящая работа посвящена разработке и исследованию структурных и принципиальных электрических схем, программ кодирования и декодирования двоичных сверточных кодов, а также методик оценки помехоустойчивости систем связи, использующих двоичные сверточные коды.

Сверточные коды обладают высокими корректирующими свойствами. Они применяются в различных системах помехоустойчивой передачи информации, в системах сотовой связи, в магнитных накопителях информации с высокой плотностью записи, в системах дальней космической связи. Реализация устройств кодирования и декодирования на интегральных микросхемах существенно расширяет круг технически реализуемых решений.

Таким образом, широкое использование сверточных кодов в системах цифровой связи является весьма актуальным. Для эффективного изучения сверточных кодов и их применения можно использовать лабораторные работы с соответствующим методическим обеспечением, предлагаемые в данной работе (см. Приложение А, Б).

Цель и задачи исследования.

Целью диссертационной работы является разработка схем, программных модулей кодирования и декодирования двоичных сверточных кодов, а также исследование помехоустойчивости сверточных кодов и разработка лабораторных работ по этой теме.

Для достижения поставленной цели необходимо решить описанные ниже задачи:

- исследовать влияния параметров на работу двоичных сверточных кодов;
- разработать методику построения электрических принципиальных схем кодеров и их моделирования в среде QUARTUS II;
- провести анализ способов кодирования сверточных кодов с помощью диаграмм состояний, древовидных диаграмм и решетчатых диаграмм;
- применить способ декодирования по Витерби;
- разработать программные модули кодирования и декодирования на основе математических моделей и алгоритмов;
- исследовать зависимость вероятности битовых ошибок (BER) от отношение сигнал/шум в цифровой системе передачи данных в среде MATLAB;
- создать виртуальные лабораторные работы по сверточным кодам.

Объект и предмет исследования.

Объектом исследования являются двоичные сверточные коды.

Методы проведенного исследования.

При выполнении работы использованы методы и алгоритмы теории помехоустойчивого кодирования-декодирования, методы объектно-ориентированного программирования.

Научная новизна и значимость полученных результатов.

Научная новизна диссертационной работы заключается в эффективном применении кодирования и моделирования в среде программирования логических интегральных схем QUARTUSII. Разработанные лабораторные работы позволяют на высоком уровне исследовать работу сверточных кодов.

Практическая значимость полученных результатов.

В результате проведенных исследований разработаны принципиальные электрические схемы кодеров в среде QUARTUSII, которые могут быть «зашиты» в кристалл программируемых логических интегральных схем (ПЛИС). Разработанные лабораторные работы являются хорошим «тренажером» при освоении теории и практики по сверточным кодам.

Личный вклад автора.

Автором диссертационной работы выполнена разработка принципиальных электрических схем кодеров в среде QUARTUSII, которые могут быть «зашиты» в кристалл программируемых логических интегральных схем (ПЛИС). Им лично выполнена доработка программных модулей кодирования и декодирования сверточных кодов в среде MATLAB,позволяющие исследовать помехоустойчивость сверточных кодов.

Основные положения, выносимые на защиту:

1. Исследованы влияния параметров на работу сверточных кодеров.

2. Изучены способы кодирования сверточных кодов с помощью диаграмм состояний, древовидных диаграмм и решетчатых диаграмм.

3. Разработаны принципиальные электрические схемы кодеров в среде QUARTUSII, которые могут быть «зашиты» в кристалл программируемых логических интегральных схем (ПЛИС).

4. Доработаны программные модули кодирования и декодирования по Витерби сверточных кодов в среде MATLAB,позволяющие исследовать помехоустойчивость сверточных кодов.

5. Разработаны для учебного процесса лабораторные работы по сверточным кодам.

Структура и объем диссертации.

Диссертационная работа состоит из введения, трех глав, заключения, списка литературы из 27 наименований и 3 приложений. Основная часть работы изложена на 57 страницах, содержит 35 рисунков и 12 таблиц.

1 Кодирование двоичных сверточных кодов

1.1 Определение сверточного кода

Двоичные сверточные коды – это коды, исправляющие ошибки, которые используют непрерывную или последовательную обработку информации короткими фрагментами (блоками) [1].

Наименование «сверточный код» происходит от того, что результат кодирования на выходе кодера образуется как свертка кодируемой информационной последовательности с импульсной характеристикой кодера [2].

Сверточные коды основаны на преобразовании входной последовательности двоичных символов в выходную последовательность двоичных символов, у которой на каждый символ входной последовательности формируется более одного символа выходной последовательности [1-3].

Сверточное кодирование удобнее всего изучать, анализируя работу кодирующего устройства [4].

В общем случае сверточный кодер состоит из m-разрядного регистра сдвига и n сумматоров по модулю два. Значение m называют памятью кода. Величина n – это число символов на выходе кодера, соответствующих k информационным символам, поступившим на вход кодера за один такт [5].

Разность $r=n-k$ называется числом проверочных символов. Значения n выходных кодовых символов равны линейным комбинациям соответствующих информационных символов. Отсюда выполняется свойство линейности сверточных кодов. На каждом такте работы на вход кодера подаётся k информационных символов и считывается n символов, предназначенных для передачи по каналу связи выходных символов. Способ подключения каждого сумматора к регистру отображается соответствующим порождающим полиномом.

Таким образом, сверточный кодер обладает памятью. Символы на его выходе зависят не только от информационных символов на входе, но и от предыдущих входных символов.

Стандартный сверточный кодер, продемонстрированный на рисунке 1.1, реализуется с m-разрядным регистром сдвига и n сумматорами по модулю два[6].

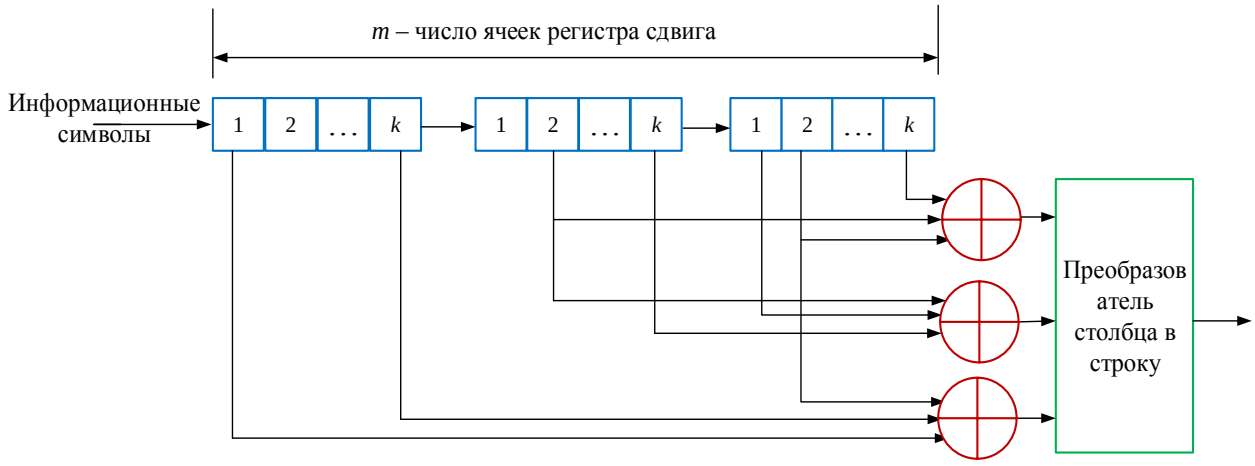


Рисунок 1.1 – Структурная схема кодера сверточного кода

В каждый момент времени на место первых k разрядов регистра перемещаются k новых бит; все биты в регистре смещаются на k разрядов вправо и на выходах сумматоров по модулю два образуется кодовые символы.

Входы сумматоров соединены с определенными разрядами регистра сдвигов. Преобразователь на выходе кодера устанавливает порядок отправки кодовых символов в канал. За время одного информационного символа на выходе образуется n кодовых символов. Затем эти символы кода применяются модулятором для образования сигналов, чтобы передать по каналу связи [5].

Для примера рассмотрим сверточный кодер с параметрами: $m=3,k=1,n=2$.

Структурная схема кодера показана на рисунке 1.2.

Изначально все ячейки регистра сдвига расположены в нулевом состоянии. В случае если первый входной бит равен единице «1,он без задержки появится на выходе первой (левой) ячейки регистра и, соответственно, на двух входах выходного преобразователя (мультиплексор). Преобразователь поочерёдно выдаёт содержимое входов, и выходная последовательность будет 11.

Допустим, что второй входной бит «0». Он записывается в первую ячейку регистра, проталкивает предыдущий бит («1») во вторую ячейку и на входах преобразователя (сверху вниз) появляются 01. Тогда вторая выходная последовательность 01.

Если третий входной бит 1, выходная последовательность 00 и так далее.

Таким образом, в ответ на каждый входной бит($k=1$) сверточный кодер кодирует двумя битами, по числу сумматоров по модулю два($n=2$).

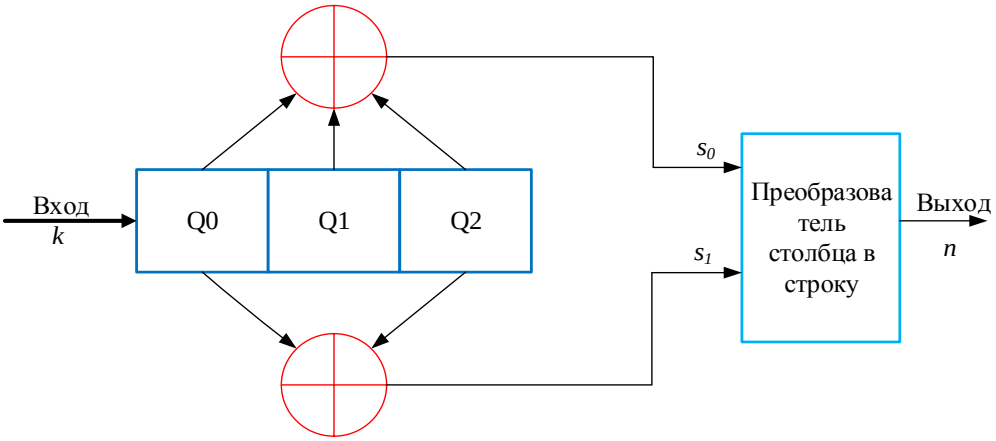


Рисунок 1.2 – Структурная схема кодера сверточного кода с параметрами *m=3, k=1, n=2*

где Q_0, Q_1, Q_2 – ячейки регистра сдвига;
s_0, s_1– выходы сумматоров.

Рассмотрим на примере, как формируется выходная кодовая последовательность для входного сигнала (110100) (таблица 1.1).

Таблица 1.1 –Формирование кодовой последовательности

Номера тактовых импульсов						
Q_0						
Q_1						
Q_2						
s_0						
s_1						
S	1	1	1	0	0	1

На выходе кодера получим кодовую последовательность:
s = (11 01 01 00 10 11).

Сверточный код удобно задавать посредством порождающих (производящих) многочленов. Порождающие многочлены целиком определяют структуру кодера сверточного кода.

1.2 Полиномиальное задание сверточного кодера

Сверточное кодирование – это итеративная обработка потока битов, создающая зависимость каждого бита от нескольких предыдущих.

Сверточный код задают посредством порождающих полиномов, которые определяют структуру двоичного кодера сверточного кода.

Кодовое слово на выходе такого кодера составляется в виде в двух последовательностей, которые в двоичной форме представляют коэффициенты соответствующих порождающих полиномов [7].

Для описания сверточного кода необходимо несколько порождающих полиномов, число которых определяются числом*n* выходных символов, передаваемых в канал связи за каждый такт [4].

Пусть $g_{i\,j}(x),\ i=1,...,k,\ j=1,...,n$ - множество порождающих полиномов. Они могут быть объединены в порождающую матрицу размера *k*n*.

$$G(x)=\left[g_{i\,j}(x)\right]. \tag{1.1}$$

Для примера рассмотрим сверточный кодер на рисунке 1.2. Порождающие полиномы имеют вид:

$$\begin{aligned} g_1(x)&=1+x+x^2;\\ g_2(x)&=1+x^2. \end{aligned} \tag{1.2}$$

Порождающая матрица для этого кодера имеет вид:

$$G(x)=\left[1+x+x^2\ \ 1+x^2\right]. \tag{1.3}$$

Рассмотрим, кодирование последовательности информационных символов (110100). Последовательность символов, поступающих на вход кодера, представим в виде многочлена:

$$a(x)=1+x+x^3, \tag{1.4}$$

где x^i– это оператор задержки сдвигающего регистра на *i* тактов;

a_i –0;1 информационные двоичные символы.

В силу линейности сверточного кода:

$$s(x)=G(x)\cdot a(x). \tag{1.5}$$

В результате на выходе кодера будет образована последовательность *s(x)*:

$$s(x)=\left[1+x+x^3\right]\cdot\left[1+x+x^2\ \ 1+x^2\right]=\left[1+x^3+x^5\ \ 1+x+x^2+x^5\right]\Rightarrow s=[100101\ \ 111001].$$

Отсюда видно, что на выходе кодера формируется кодовая последовательность s=(11 01 01 10 00 11), которая совпадает с результатом в таблице 1.1.

Вместе с этим для формирования сверточных кодов следует перечислить параметры, определяющие структуру кодов.

1.3 Основные параметры сверточных кодов

Число контактов мультиплексора и тактовая частота переключения в сверточных кодерах зависит от относительной скорости кода равной $R = k/n$и обуславливает избыточность, вводимую при кодировании.

В соответствии с этим частота переключения должна в nраз превышать входную тактовую частоту. Так, при скорости $R=1/2$у мультиплексора переключение должно производиться с частой в 2 раза большей тактовой частоты [4, 8].

Избыточность кода определяется по формуле:

$$\chi = 1 - R = 1 - \frac{k}{n}. \tag{1.6}$$

Таким образом, при $R=1/3$количество символов в выходной последовательности больше их количества во входной информационной последовательности в три раза.

Полная длина кодового ограничения по выходу кодера– это число последовательных кодовых символов на выходе кодера, зависящих от выбранного информационного символа, и определяется по формуле [4]:

$$l_2 = \frac{\mathrm{m}}{k} \cdot n. \tag{1.7}$$

При k=1 длина кодового ограничения по выходу равна $l_2 = m \cdot n.$

Значение l_2 по выходу можно определить максимальной степенью порождающего полинома по формуле [2]:

$$l_2 = n \cdot \max[\deg g_{j,i}(x) + 1], \tag{1.8}$$

где $degg_{,j,i}(x)$ – степень порождающего полинома.

Значение l_2 показывает на какое максимальное число выходных символов влияет данный информационный символ.

Рассмотрим для примера сверточный кодер на рисунке. 1.2. Скорость кода этого кодера $R=1/2$.

Найдем избыточность кода по формуле (1.6):

$$\chi = 1 - R = 1 - 0,5 = 0,5 \tag{1.9}$$

Полная длина кодового ограничения по выходу(1.7) равна:

$$l_2 = 3 \cdot 2 = 6. \qquad (1.10)$$

По второй формуле получаем тот же результат:

$$l_2 = n \cdot \max[\deg g_{j,i}(x) + 1] = 2 \cdot (2 + 1) = 6 \qquad (1.11)$$

Отсюда можно сказать, что поступающий один информационный символ влияет на шесть выходных символов.

1.4 Сверточные коды в блочном виде. Систематические блочные сверточные коды

Кодирование может производиться в непрерывном или блочном режимах. В последнем случае информационные последовательности разбиваются на блоки конечной длины[1, 9-10].

В блочном режиме за *a* двоичным символом (последним) в кодер должны быть введены *m-1* нулей для того, чтобы очистить регистр, которые иногда называют хвостом кода. Это необходимо для того, чтобы сделать код конечным.

Сверточный код можно рассматривать так же, как длинный блочный код и вводимое здесь ограничение длины с помощью *m-1* нулей позволяет очистить регистр кодера для следующего блока [9, 12].

Если в последовательности кодовых символов,формируемых кодером, можно отделить *r=n-k*проверочных символов от *k* информационных, то код называют систематическим.

Систематическим сверточным кодом называют код, для которого в выходной последовательности сформированных кодовых символов без изменения содержится породившая ее последовательность информационных символов. В остальных случаяхсверточный код является несистематическим.

При использованиисистематического кодера на *k* выходах будут информационные последовательности. На остальных(*n–k)* выходах будут последовательности проверочных символов, образованные как линейные комбинации информационных [5].

Для примера рассмотрим систематический сверточный кодер с параметрами *k=1, n=2*и*m=3* (рисунок 1.3).

Для того чтобы получить информационную последовательность вместе с выходной в кодерах с*k=1*один из порождающих многочленов, формирующих систематические коды,равен единице, то есть $g_1(x)=1$ или $g_2(x)=1$, [4].

На рисунке 1.3 видно, что это систематический сверточный кодер. Пусть на вход кодера поступает информационная последовательность

($a_0a_1a_2a_3$). На выходе мы можем отделить b_i проверочных символов от a_i информационных символов. Кодовое ограничение равно трем, тогда в кодер будут введены $m - 1=2$ нулей (рисунок 1.3). Тогда код будет систематическим блочным кодом.

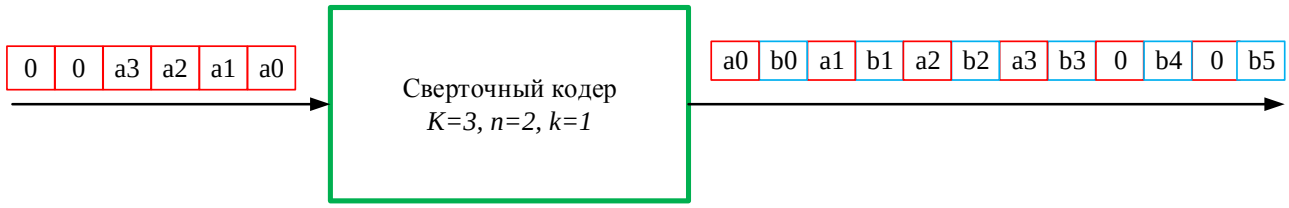


Рисунок 1.3 – Систематический сверточный кодер

Преимуществом систематических кодовявляется получение оценки информационных символов на приемной стороне без декодирования или другой обработки входных символов [5, 11].

1.5 Принципиальные электрические схемы кодеров в среде программируемых логических интегральных схем QUARTUSII

Для практической реализации сверточных кодеров необходимо разработать их до уровня принципиальных электрических схем. Для примера рассмотрим несколько сверточных кодеров.

1.5.1 Сверточный кодер с параметрами *m=4, k=1, n=2*

Рассмотрим сверточный кодер с параметрами:

$m=4$ – число ячеек регистра сдвига;

$k= 1$ – число информационных символов, поступающих за один такт на вход кодера;

$n=2$–число символов на выходе кодера, соответствующих k;

$R=1/2$–скорость кода;

$g_1=17$, $g_2=15$ – коэффициенты порождающих полиномов в восьмеричной форме.

Схема кодера показана на рисунке 1.4 [4].

Информационные символы на схеме поступают слева, и для каждого информационного символа на выходах двух сумматоров по модулю два образуются два выходных символа [13-14].

Связь между ячейками регистра сдвига и сумматорами удобно описывать порождающими полиномами: верхний и нижний сумматоры представляются соответственно полиномами:

$$g_1(x) = 1 + x + x^2 + x^3;$$
$$g_2(x) = 1 + x + x^3. \qquad (1.12)$$

Коэффициенты порождающих полиномов в двоичной форме:

$$g_1 = (1\,1\,1\,1);$$
$$g_2 = (1\,1\,0\,1). \qquad (1.13)$$

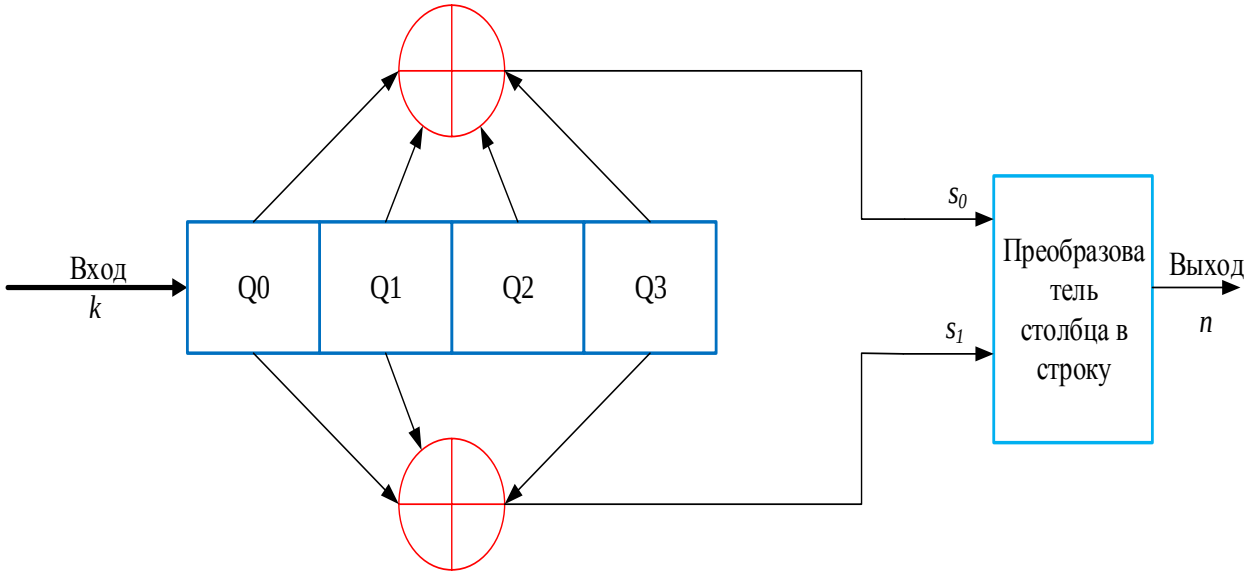


Рисунок 1.4 – Структурная схема кодера сверточного кода с параметрами *m=3, k=1, n=2*

здесь Q_0, Q_1, Q_2, Q_3 – ячейки регистра сдвига;

s_0, s_1– выходы сумматоров;

Рассмотрим на примере, как формируется выходная кодовая последовательность для входного сигнала (1101000) (таблица 1.2).

Таблица 1.2– Формирование кодовой последовательности

Номера тактовых импульсов							
Q_0							
Q_1							
Q_2							
Q_3							
s_0							
s_1							
s	1	0	1	0	0	0	1

На выходе кодера получим кодовую последовательность:

s *= (11 01 00 10 10 01 11).*

На рисунке 1.2 и 1.4 изображены структурные схемы кодера. Для практической реализации таких кодеров необходимо разработать их до уровня принципиальных электрических схем. Такую задачу можно решить применяя, например, среду программируемой логики (Altera QUARTUSII).

1.5.2 Описание среды Altera QUARTUSII

QUARTUSII представляет собой автоматизированную систему сквозного проектирования цифровых устройств на кристаллах ПЛИС фирмы Altera [15].

Он предоставляет пользователю широкие возможности по вводу описаний проекта, логическому синтезу, компиляции проекта, программированию ПЛИС, функциональному и временному моделированию, временному анализу и анализу потребляемой мощности проекта, реализации внутрисистемной отладки. В QUARTUSII используется удобный графический интерфейс и простая в применении справочная система, содержащая всю необходимую для выполнения проектирования информацию.

Также пакет позволяет использовать командную строку для выполнения каждого этапа проектирования.

QUARTUSII интегрирует в себе большое количество программных модулей, предназначенных для выполнения различных этапов проектирования. Задание параметров и выполнение типовых команд выполняется в отдельных модулях одинаково, что значительно облегчает работу пользователя. Редакторы исходных файлов проекта (графический, текстовый, редактор символов, содержимого модулей памяти, временных диаграмм, конечных автоматов) используют одинаковые подходы и приёмы, а также похожие оконные формы, применяемые при создании и редактировании исходных файлов с описанием модулей проектируемого устройства [15].

В состав стандартной библиотеки QUARTUSII входит большое количество базовых элементов, включая мега функции и макрофункции. Составной частью мега функций являются операционные устройства, созданные по стандарту библиотеки параметризируемых модулей (LPM – libraryofparameterizedmodules).

Принципиальная электрическая схема сверточного кодера в среде программируемых логических интегральных схем QUARTUS II строится с помощью D-триггера.

1.5.3 Принцип работы D-триггера

D-триггер (от английского delay) называют информационным триггером, а также триггером задержки. D-триггер бывает только синхронным. Он может управляться (переключаться) как уровнем тактирующего импульса, так и его фронтом. Для триггера типа D, состояние в интервале времени между сигналом на входной линии и следующим состоянием триггера формируется проще, чем для любого другого типа. Схемное обозначение D-триггера приведено на рисунке 1.5.По

синхроимпульсу D-триггер принимает то состояние, которое имеет входная линия. На рисунке 1.6 приведены временные диаграммы, поясняющие его работу. D-триггер имеет как минимум две входные линии: одна – для подачи синхроимпульсов; другая- информационных сигналов [16].

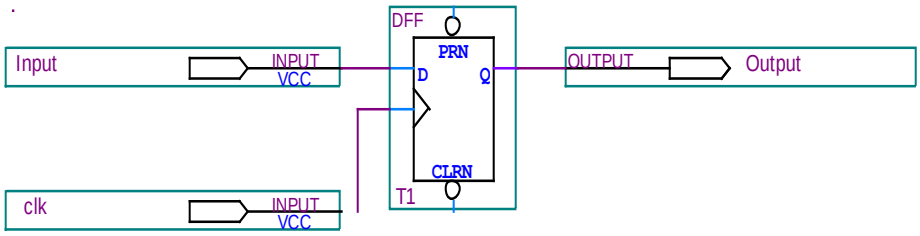


Рисунок 1.5 – D-триггер

D-триггер запоминает состояние входа и выдает его на выход. D-триггеры имеют как минимум два входа: информационный D и синхронизации C. После прихода активного фронта импульса синхронизации на вход CD-триггер открывается. Сохранение информации в D-триггерах происходит после спада импульса синхронизации, и эта информация остается неизменной до прихода следующего импульса синхронизации. Также из-за переходных процессов существует задержка между временем прихода определенного фронта синхроимпульса и временем запоминания информационного бита в память триггера [16].

Принципиальная схема D-триггера в среде программируемых логических интегральных схем QUARTUS II представлена на рисунке 1.5.

На рисунке 1.6 представлены временные диаграммы работы D-триггера.

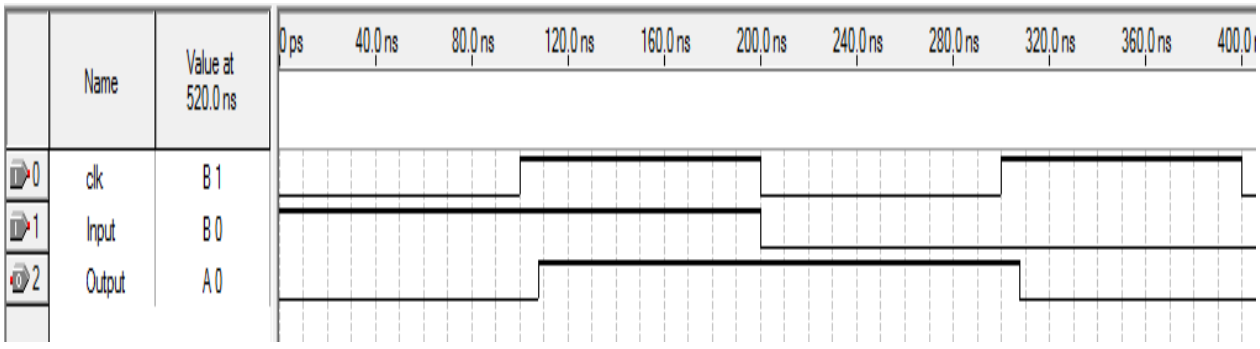


Рисунок 1.6 – Временные диаграммы работы D-триггера

Как видно из рисунка 1.6, при приходе на вход триггера единицы и переднего фронта синхроимпульса на выход D-триггера также передается единица до времени прихода следующего синхроимпульса, но с некоторой задержкой, о которой было сказано ранее.

1.5.4 Построение кодера в среде программируемых логических интегральных схем QUARTUSII

Построим сверточный кодер (рисунок 1.2) в среде программируемых логических интегральных схем QUARTUS II (рисунок 1.7) [15].

Сверточный кодер на рисунке 1.7 состоит из D-триггеров и элементов XOR (сумматоров по модулю два) и преобразователя (мультиплексора).

Здесь отсутствует первая ячейка памяти. Это объясняется тем, что регистр сдвига можно рассматривать либо как регистр, содержимое которого сдвигается на один разряд вправо при введении в него слева каждого нового двоичного символа (содержимое самого правого разряда при этом теряется) (рисунок 1.7), либо как цифровую линию задержки, в которой каждый элемент задержки хранит один двоичный символ до поступления нового входного двоичного символа (рисунок 1.2).

Эти две схемы (рисунок 1.2 и рисунок 1.7) эквивалентны.

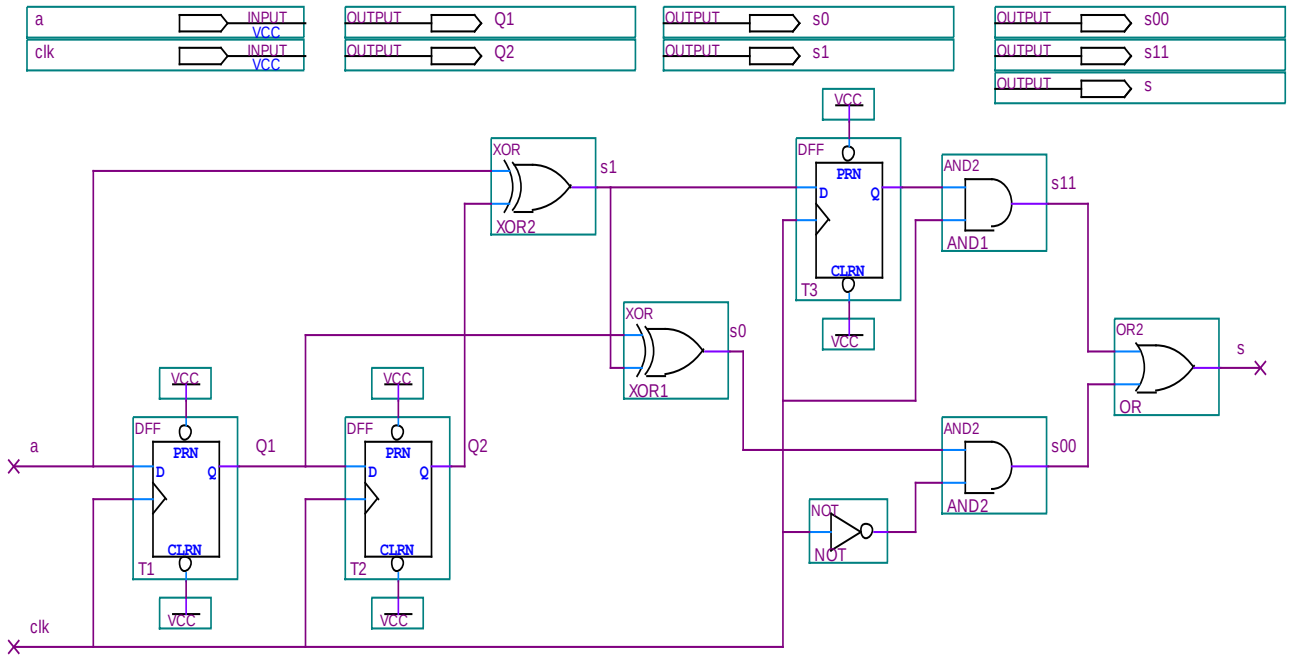


Рисунок 1.7 – Схема кодера в среде программируемых логических интегральных схем QUARTUSII

Мультиплексор состоит из: D-триггера и логических элементовAND, NOT, OR(рисунок 1.8).

На вход мультиплексора поступает параллельные символы (выходы двух сумматоров). Мультиплексор за первую половину такта работы кодера считывает данные сначала с логического элемента XOR2, а вторую половину такта с логического элемента XOR3.

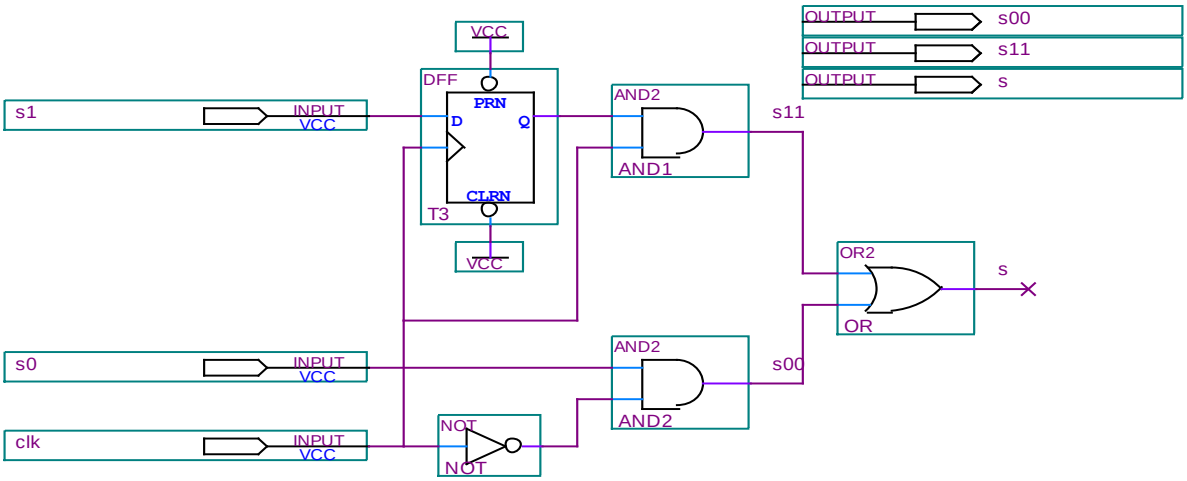


Рисунок 1.8 – Мультиплексор в среде QUARTUS II

Выход первого сумматора поступает на первый вход логического элемента AND2, а на второй вход поступает инвертированный тактовый импульс (с помощью логического элемента NOT). Если на вход поступает 1+1, то на выходе AND2 получим 1. В остальных случаях 0.

Выход второго сумматора поступает на информационный вход Dтриггера, а на второй вход поступает тактовый импульс. Этот символ появляется на выходе триггера в момент первого фронта.

Далее он поступает на первый вход логического элемента AND1, а на второй вход поступает тактовый импульс. Если на вход поступает 1+1, то на выходе AND1получим 1. В остальных случаях 0.

Далее выходы AND1 и AND2 поступают на вход логического элемента OR. Если на вход поступают 0+0 на выходе получим 0. В остальных случаях 1 (рисунок 1.9). Таким образом получим кодовую последовательность на выходе мультиплексора.

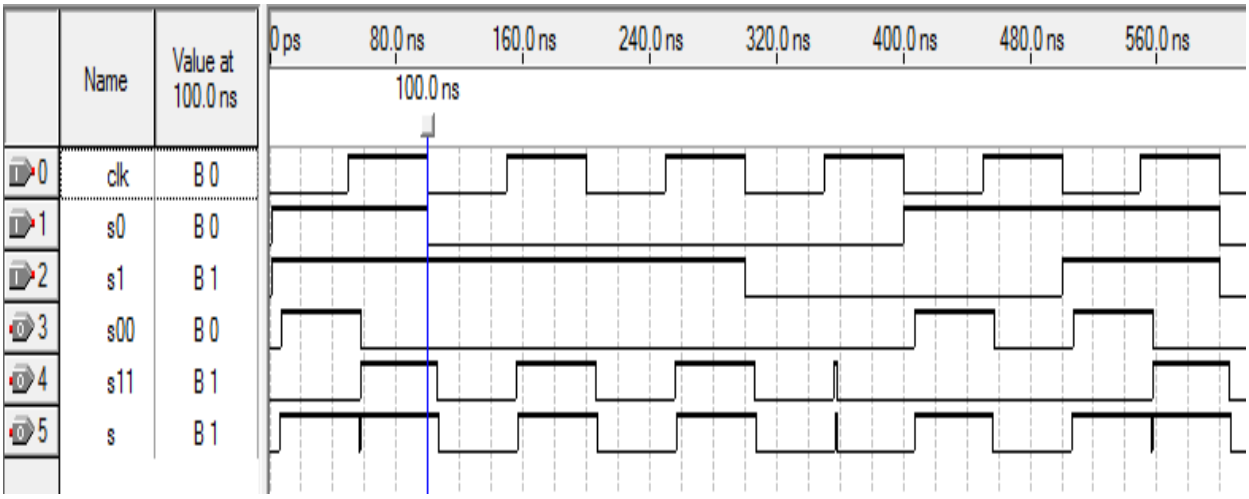


Рисунок 1.9 – Преобразование параллельного сигнала в последовательный сигнал

По временной диаграмме (рисунок 1.10) можно заметить, что при входной последовательности бит (110100) выходная последовательность будет (11 01 01 00 10 11).Получен тот же результат, который найден в таблице 1.1.

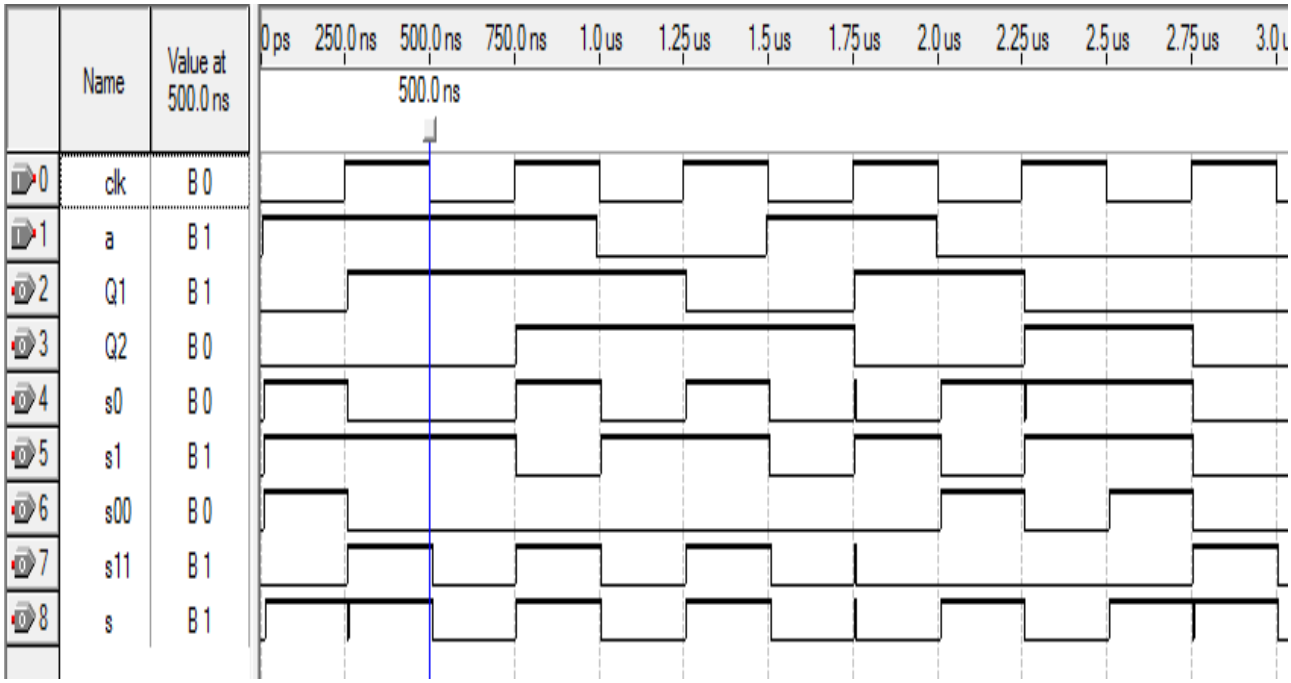


Рисунок 1.10 – Временные диаграммы кодера в среде QUARTUS II

Стоит отметить, что важной особенностью принципа формирования дибитов является то, что значение каждого формируемого дибита зависит как от входящего информационного бита, так и от двух предыдущих бит, значения которых хранятся в двух запоминающих ячейках.

Таким образом, если принять, что a_i – входящий бит, то значение элемента XOR1 будет определяться выражением:

$$a_i \oplus a_{i-1} \oplus a_{i-2}, \tag{1.14}$$

а значение элемента XOR2 выражением:

$$a_i \oplus a_{i-2}. \tag{1.15}$$

Соответственно, дибит формируется из пары битов, значение первого из которых равно (1.13), а второго – (1.14).

Иначе говоря, значение дибита зависит от трех состояний: значения входного бита, значения первой запоминающей ячейки и значения второй запоминающей ячейки.

Построим сверточный кодер на рисунке 1.4 в среде программируемых логических интегральных схем QUARTUS II (рисунок 1.11).

Сверточный кодер на рисунке 1.11 состоит из трех запоминающих ячеек (D триггеры) и элементов XOR (сумматор по модулю 2) и преобразователя (мультиплексор).

Принцип работы преобразователя рассмотрен выше.

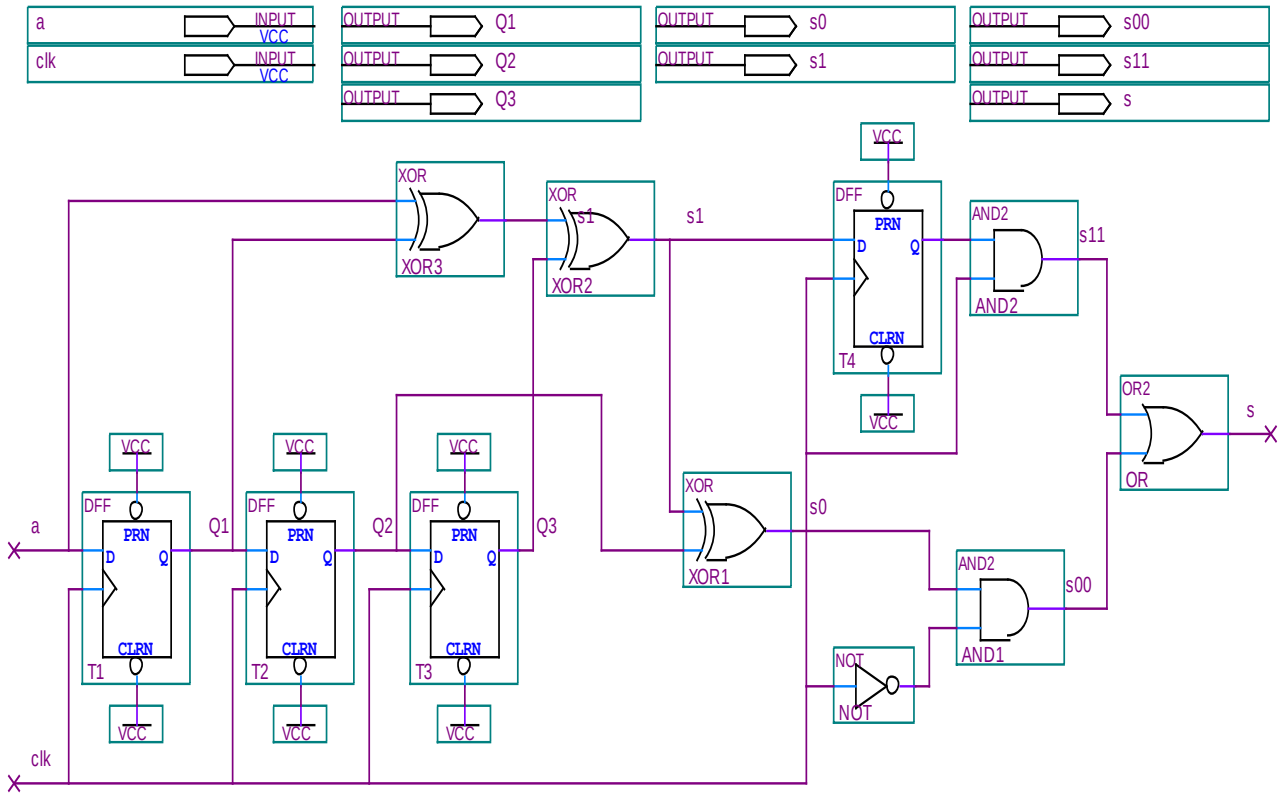


Рисунок 1.11 – Схема кодера в среде QUARTUS II

Как видно из временной диаграммы (рисунок 1.12) при входной последовательности бит (1101000) выходная последовательность будет (11 00 01 10 00 10 11). Получен тот же результат, который найден в таблице 1.2.

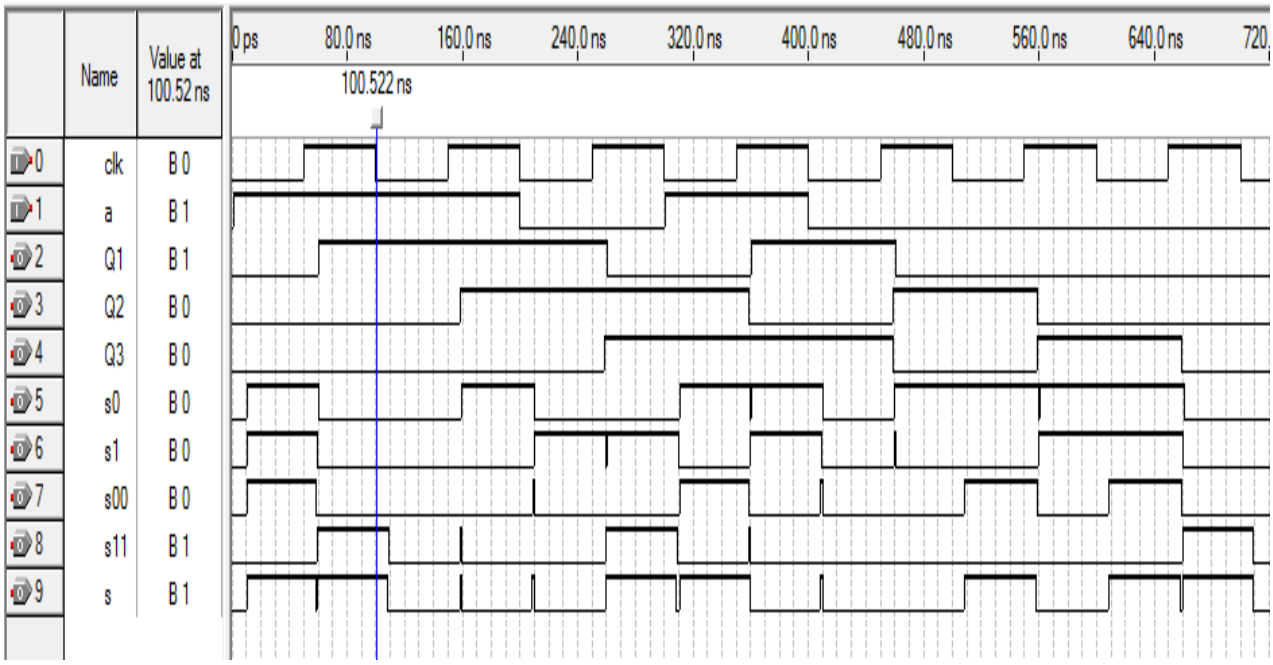


Рисунок 1.12 – Временные диаграммы кодера в QUARTUS II

1.5.5 Сверточный кодер(*m=4, k=1, n=3*)

Рассмотрим сверточный кодер со скоростью *R=1/3*. Структурная схема кодера показана на рисунке 1.13 [17].

В этом примере информационные символы на схеме поступают слева, и для каждого информационного символа на выходах трех сумматоров по модулю 2 образуются три выходных символа.

Порождающие полиномы сверточного кодера имеют вид:

$$\begin{aligned} g_1(x) &= 1 + x + x^2 + x^3; \\ g_2(x) &= 1 + x + x^3; \\ g_3(x) &= 1 + x^2 + x^3. \end{aligned} \tag{1.16}$$

Коэффициенты порождающих полиномов в двоичной форме:

$$\begin{aligned} g_1 &= (1\,1\,1\,1); \\ g_2 &= (1\,1\,0\,1); \\ g_3 &= (1\,0\,1\,1). \end{aligned} \tag{1.17}$$

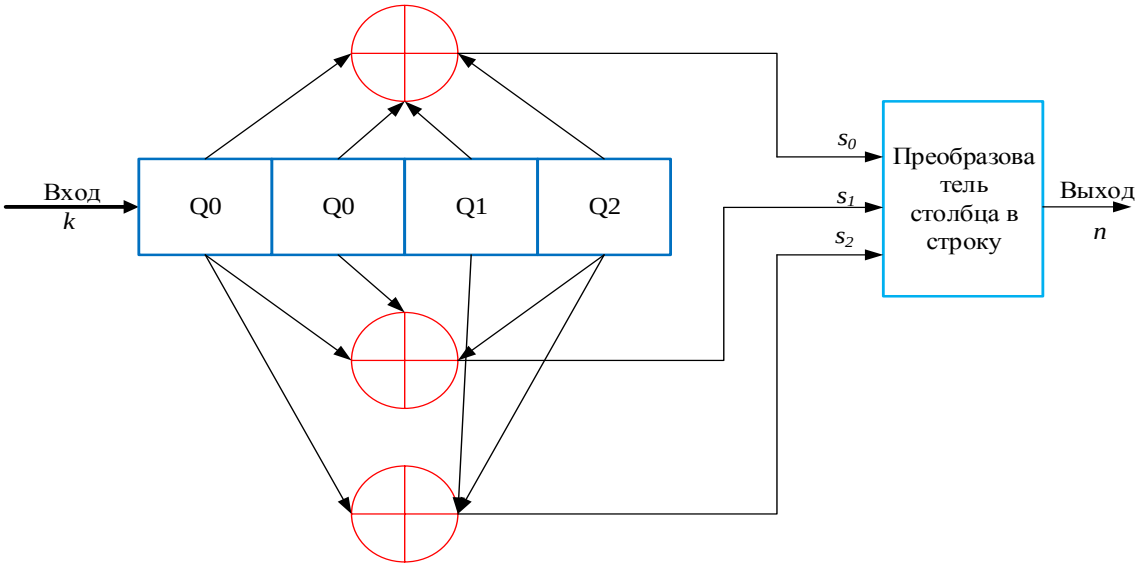


Рисунок 1.13 – Структурная схема кодера с параметрами
m= 4, k=1, n=3

Рассмотрим на примере, как формируется выходная кодовая последовательность для входного сигнала (1101000) (таблица 1.3).

Таблица 1.3– Формирование кодовой последовательности

Номера тактовых импульсов							
Q_0							
Q_1							
Q_2							
Q_3							
s_0							
s_1							
s_2							
s	11	01	11	01	01	01	11

На выходе кодера получим кодовую последовательность:
s *= (111 001 011 101 001 101 111).*

Построим сверточный кодер на рисунке 1.13 в среде программируемых логических интегральных схем QUARTUS II (рисунке 1.14).

Сверточный кодер на рисунке 1.14состоит: из трех запоминающих ячеек (D триггеров), элементов XOR (сумматоров по модулю 2) и преобразователя (мультиплексор[16].

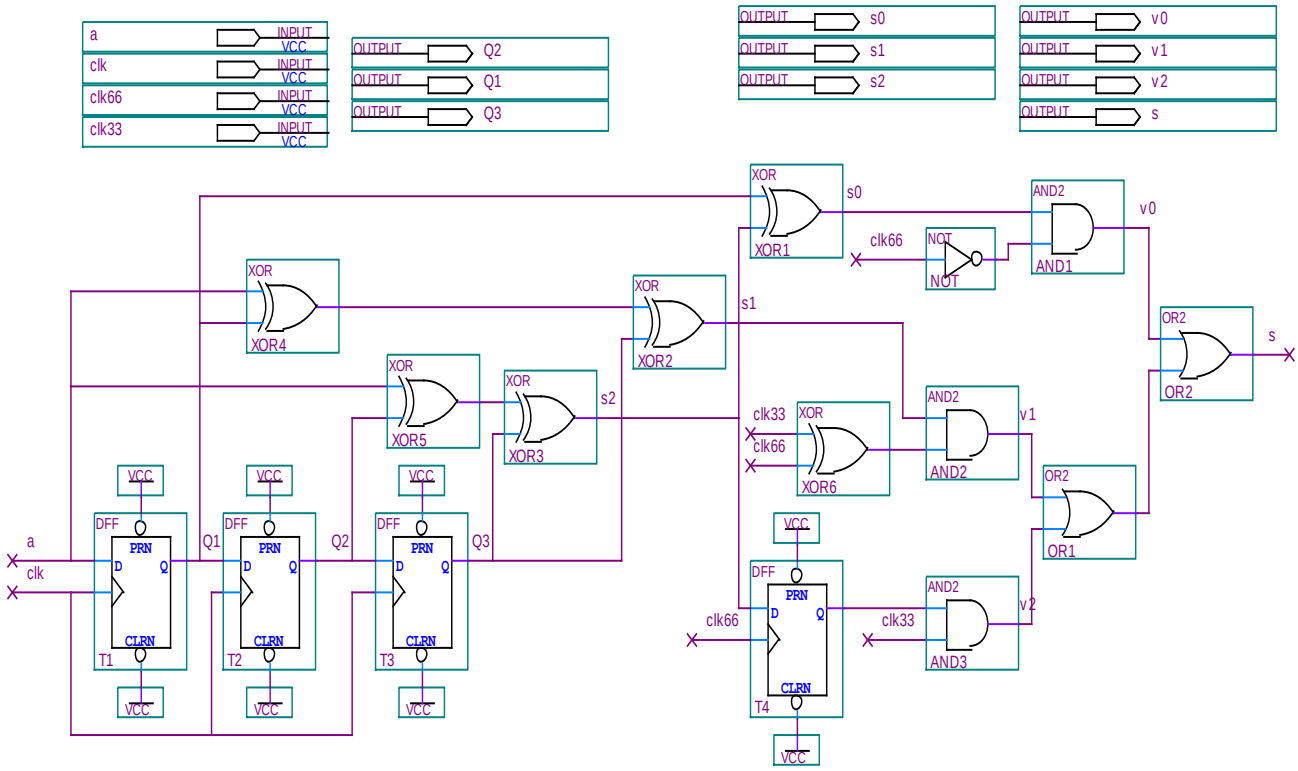


Рисунок 1.14 – Схема кодера в среде QUARTUS II

Мультиплексор состоит из: D-триггера и логических элементов NOT, AND, OR, XOR (рисунок 1.15).

Учитывая, что мультиплексор работает с частотой в три раза большей, чем скорость поступления бит на вход кодера, скорость выходного потока будет втрое выше скорости входного потока. С помощью clk33, clk66 разделим длительность выходного тактирующего элемента на три.

Мультиплексор за первую треть такта кодера получает данные сначала с логического элемента XOR1, следующую треть такта — с логического элемента XOR2, а последнюю треть — с логического элемента XOR3.

Таким образом, каждому входному биту соответствует три выходных бита, то есть три бита, первый бит которого формируется элементом XOR1, второй элементом XOR2, а третий элементом XOR3.

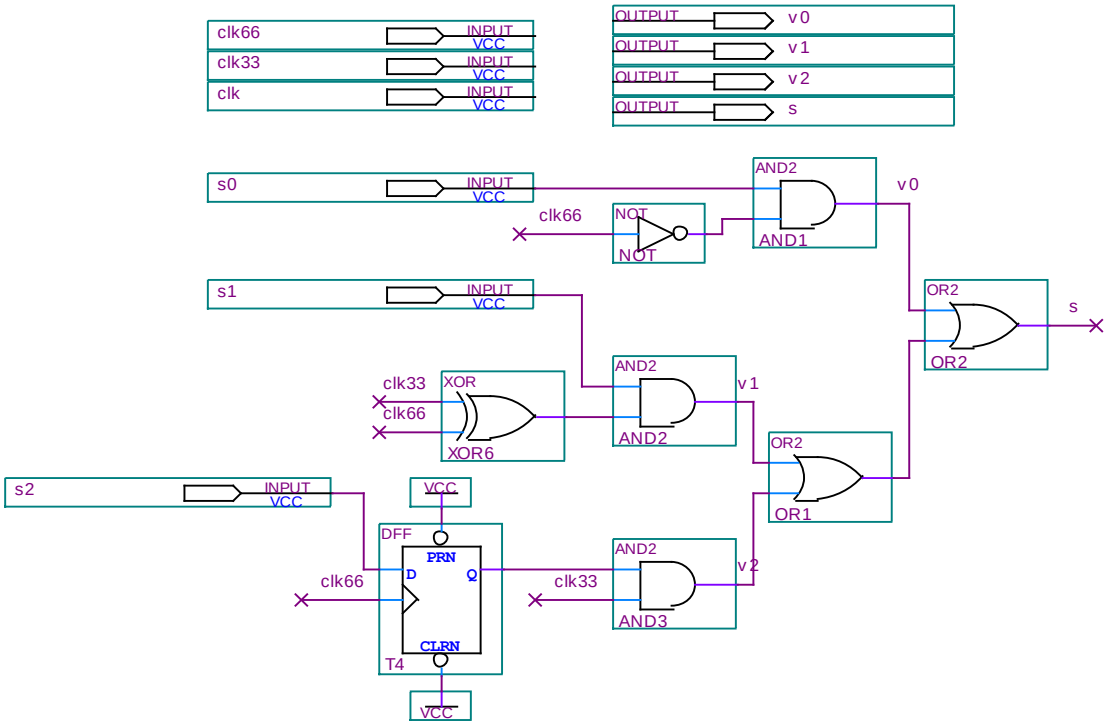


Рисунок 1.15 – Мультиплексор

Принцип работы мультиплексора представлен в виде временных диаграмм (рисунок 1.16).

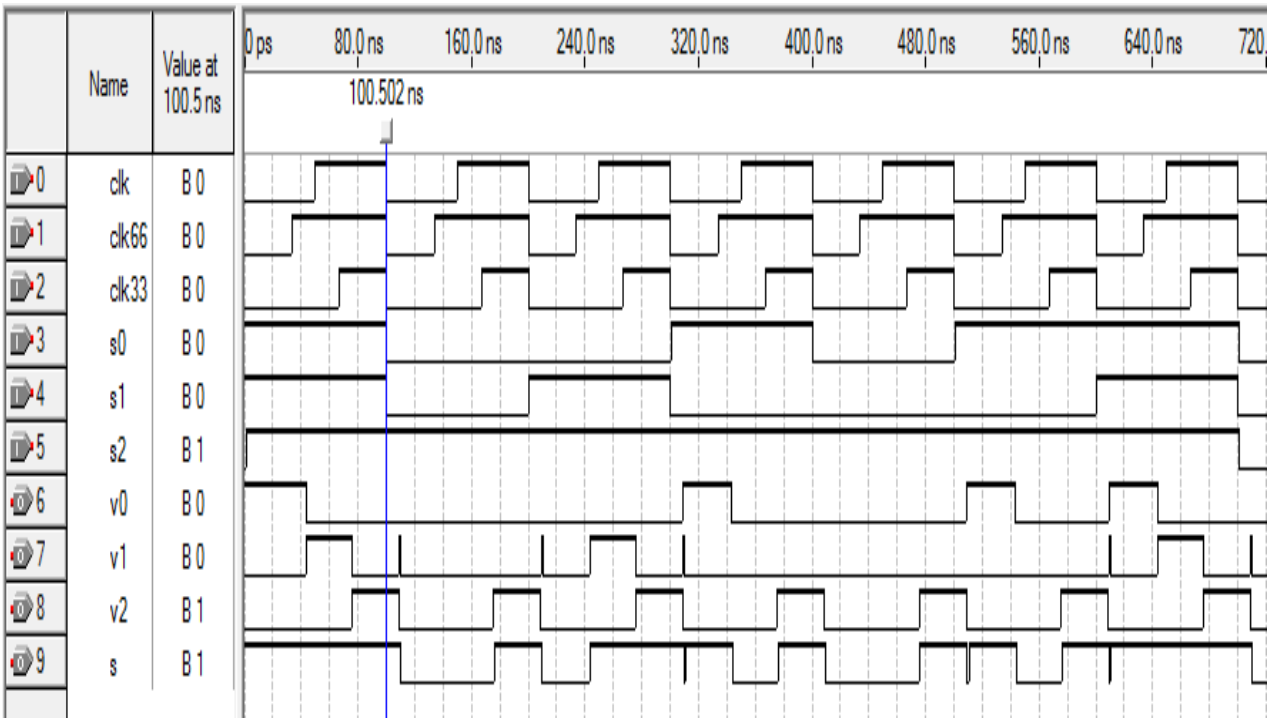


Рисунок 1.16– Преобразование параллельного сигнала в последовательный сигнал

По временной диаграмме (рисунок 1.17) состояния кодера видно, что при входной последовательности бит (1101000) выходная последовательность будет (111 001 011 101 001 101 111).

Получен тот же результат, который найден в таблице1.3.

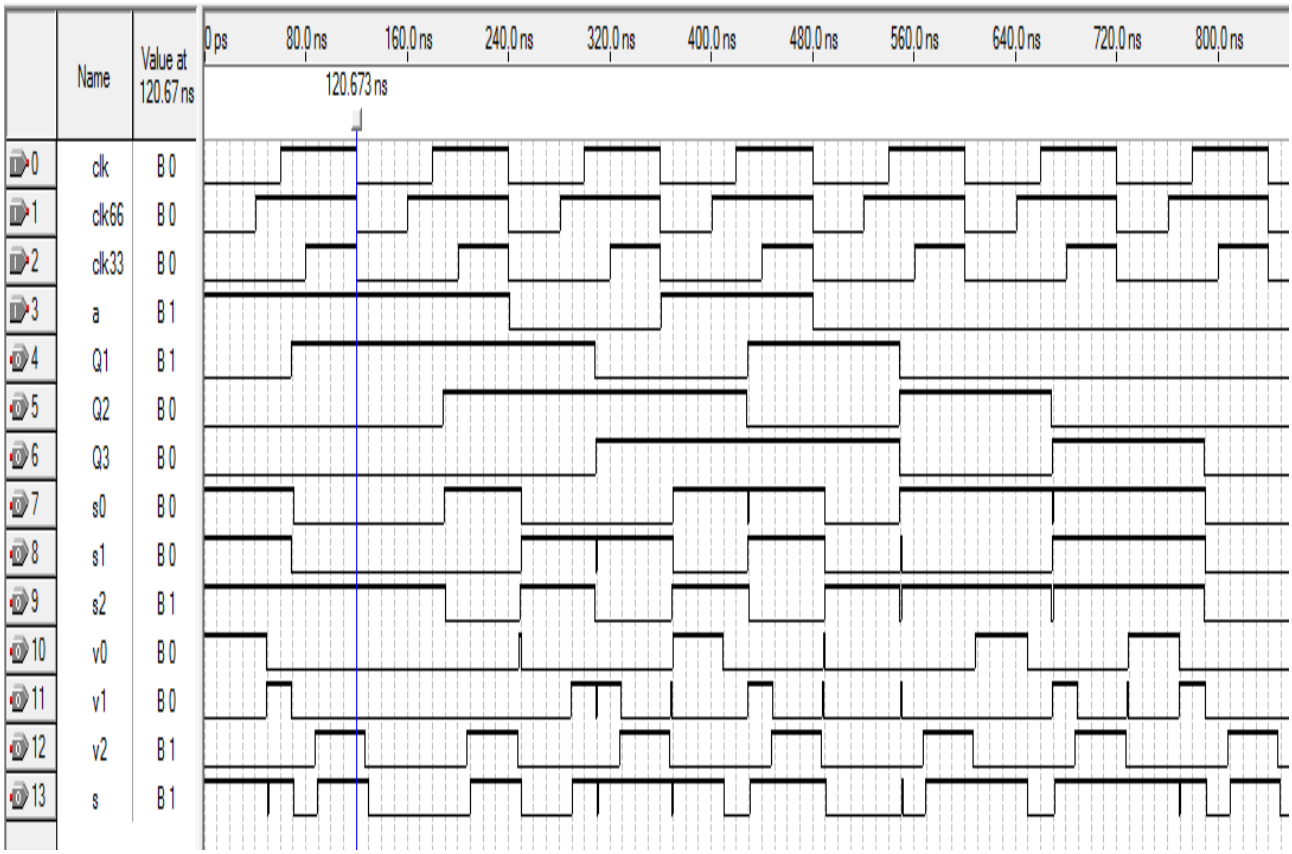


Рисунок 1.17 – Временные диаграммы кодера в среде QUARTUSII

1.5.6 Сверточный кодер (*m=2, k=2, n=3*)

При скоростях $R=k/n$, где $k>1$, ситуация становится более сложной. Рассмотрим для примера свёрточный кодер, показанный на рисунке 1.18, где скорость кодирования равна 2/3. В этом кодере каждый раз два бита поступают на вход регистров сдвига, затем сумматоры по модулю 2 определяют три символа, и на выходе генерируется три бита [12].

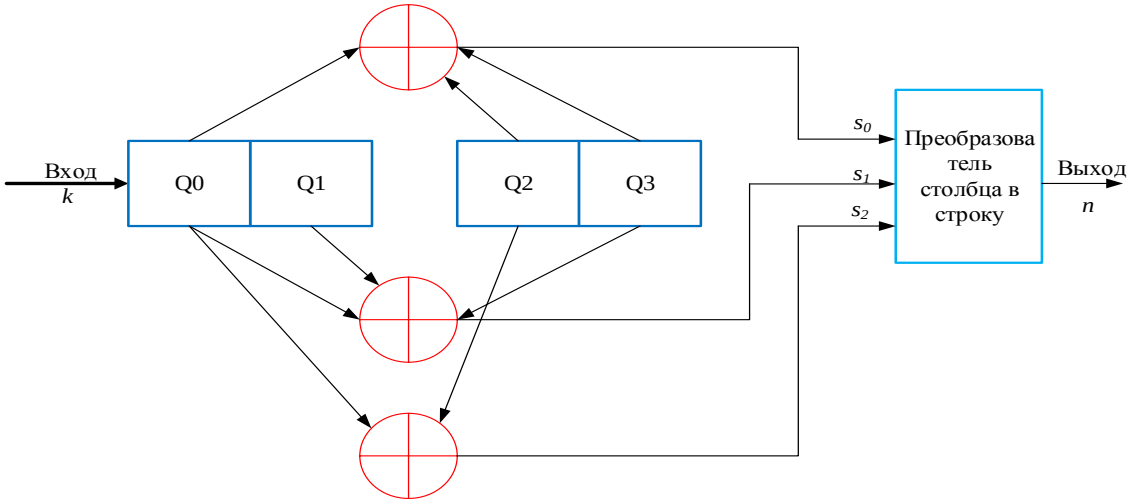
Коэффициенты порождающих полиномов представлены в восьмеричной форме.

Коэффициенты порождающих полиномов в двоичной форме:

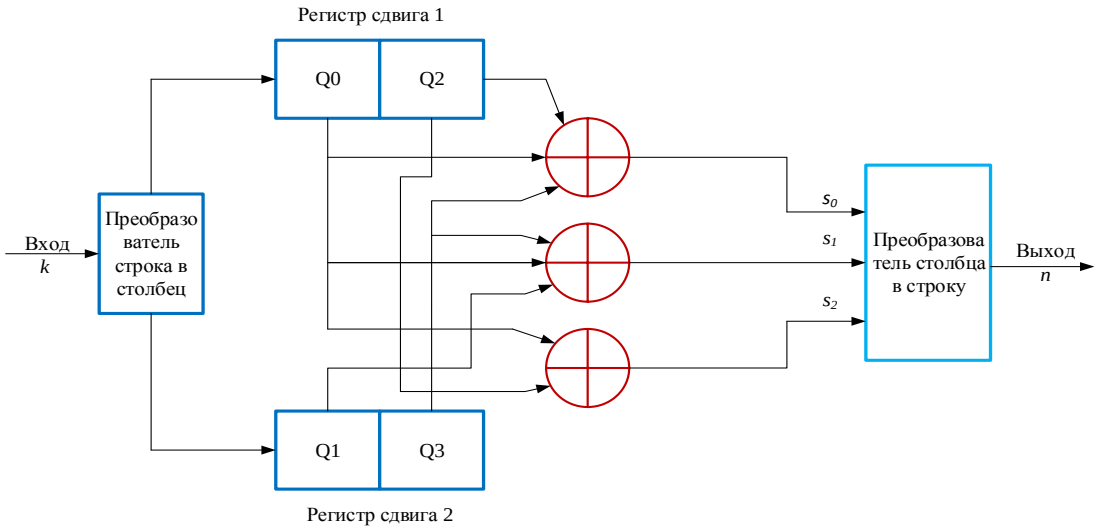
$$\begin{aligned} g_1 &= (1\ 0\ 1\ 1); \\ g_2 &= (1\ 1\ 0\ 1); \\ g_3 &= (1\ 0\ 1\ 0). \end{aligned} \qquad (1.18)$$

Порождающие полиномы имеют вид:

$$g_1(x) = 1 + x^2 + x^3;$$
$$g_2(x) = 1 + x + x^3; \qquad (1.19)$$
$$g_3(x) = 1 + x^2.$$



а)



б)

Рисунок 1.18 – Структурные схемы кодера с параметрами *m=2, k=2, n=3*

Эти две схемы эквивалентны. Регистры сдвига 1 и 2 (число регистров равно *k*) имеют по две ячейки памяти и три сумматора по модулю 2 (число сумматоров равно *n*), формирующих символы кода в соответствии с видом производящих полиномов. Преобразователь распределяет входные

информационные символы между регистрами, преобразователь столбца в строку формирует кодовую последовательность на выходе кодера из выходных символов сумматоров [9].

Первые все возможные два входных бита могут быть 11, 01, 10 или 00. Соответственно, выходные биты будут представлены в виде 000, 010, 111, 101. Когда следующая пара входных битов входит в кодер, первая пара передвигается в следующую ячейку. Соответствующие выходные биты зависят от пары битов, переместившихся во вторую ячейку и новой пары входных битов.

Рассмотрим на примере, как формируется выходная кодовая последовательность для входного сигнала (1101101100) (таблица 1.4).

Таблица 1.4– Формирование кодовой последовательности

Номера тактовых импульсов	0	1	2	3	4
Входные символы, *a*	1 1	0 1	1 0	1 1	0 0
Q_0					
Q_1					
Q_2					
Q_3					
s_0					
s_1					
s_2					
s	10 1	10 0	1 11	0 11	0 11

На выходе кодера получим кодовую последовательность:

s *= (101 100 111 011 011).*

Построим сверточный кодер на рисунке 1.18в среде программируемых логических интегральных схем QUARTUS II (рисунок 1.19). Сверточный кодер на рисунке 1.19 состоит из двух запоминающих ячеек (D-триггеров) и элементов XOR (сумматоров по модулю 2) и преобразователя (мультиплексора).

Принцип работы мультиплексора приведен выше [16].

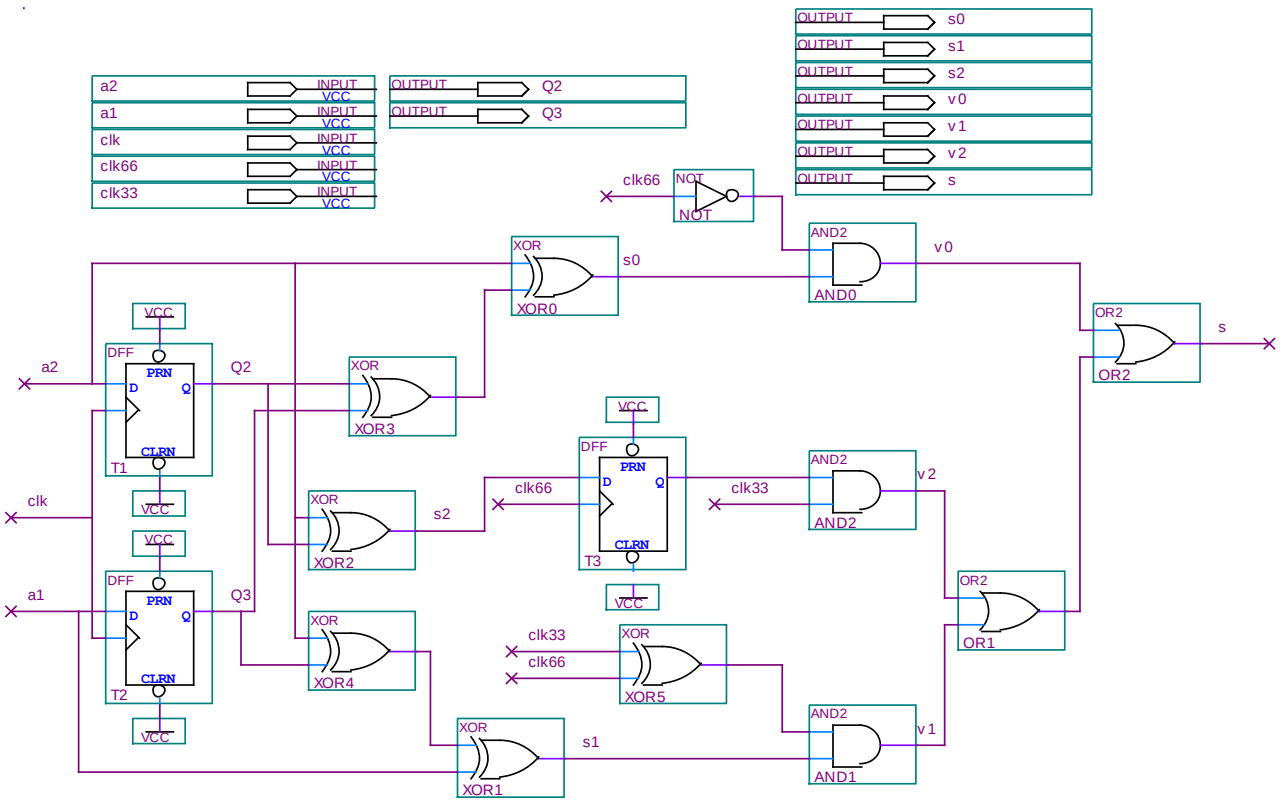


Рисунок 1.17 – Схема кодера в среде QUARTUS II

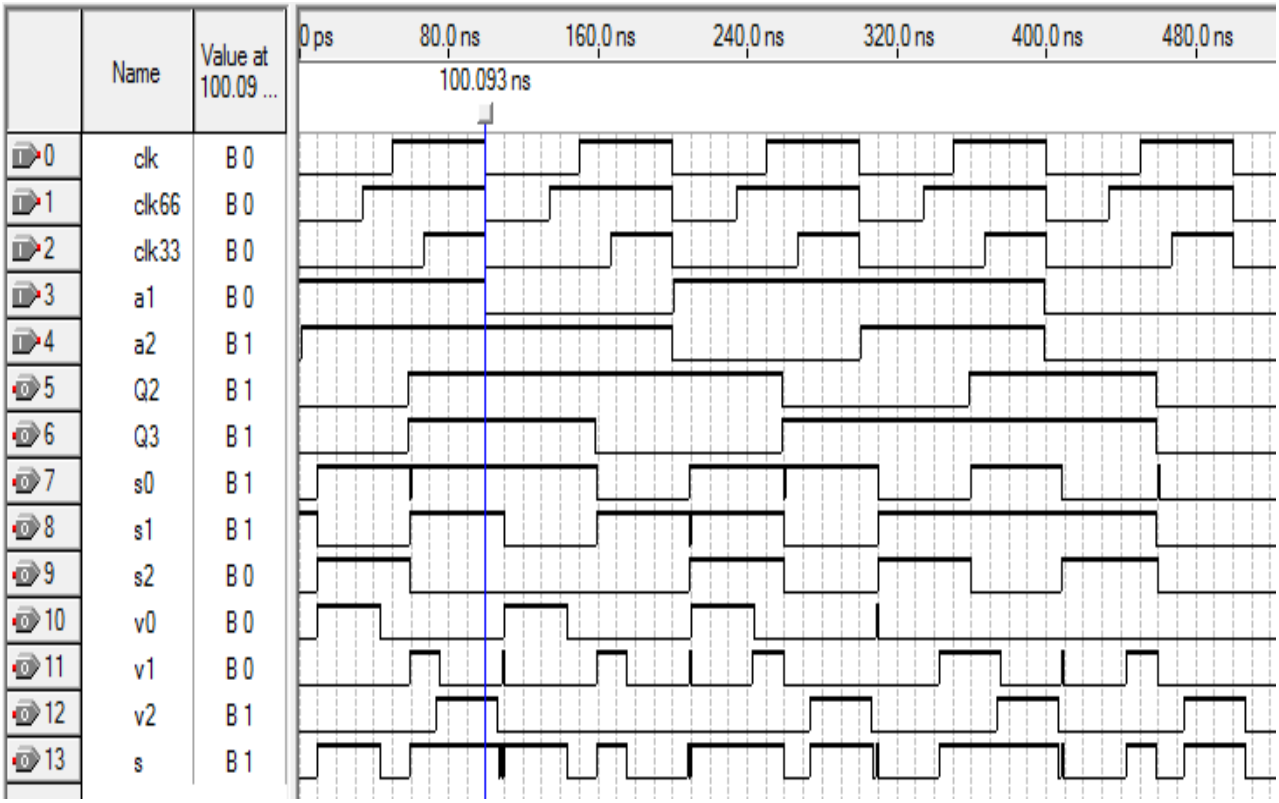


Рисунок 1.18 – Временные диаграммы кодерав среде QUARTUS II

По временной диаграмме (рисунок 1.18) можно заметить, что при входной последовательности бит (1101101100) выходная последовательность будет (101 100 111 011011).

Получен тот же результат, который найден в таблице 1.4.

Вывод.

Показаны способы кодирования двоичных сверточных кодов. Разработана методика построения структурных схем сверточных кодеров до уровня электрических принципиальных схем и их моделирования в среде QUARTUS II. Разработанные принципиальные электрические схемы кодеров в среде QUARTUSII могут быть «зашиты» в кристалл программируемых логических интегральных схем (ПЛИС).

2 Декодирование свёрточных кодов

Для сверточных кодов разработаны методы порогового, последовательного декодирования и декодирования согласно алгоритма Витерби по максимуму правдоподобия. На практике широко используется последний метод, что объясняется простотой реализации при небольших длинах кодового ограничения и получаемым выигрышем от кодирования [1-4].

Для декодирования сверточных кодов используются: диаграмма состояний, древовидная диаграмма, решетчатая диаграмма сверточных кодов.

2.1 Диаграмма состояний кодера

Работу кодера удобно рассматривать на основе не временных диаграмм, а так называемой диаграммы состояния [4-5].

Состояние кодера сверточных кодов – это содержимое *m-1*крайних правых разрядов регистра сдвига.

Диаграмма состояний представляет собой направленный граф и описывает все возможные переходы кодера из одного состояния в другое, а также содержит символы выходов кодера, которые сопровождают эти переходы. В диаграмме должно быть2^{m-1} состояний.

Пути (ветви) между состояниями соотносятся с кодовыми словами выхода кодера. Из каждого состояния (прямоугольника) исходят два пути, соответствующие битам 0 и 1 на входе кодера.

Для примера рассмотрим диаграмму состояний (рисунок 2.1) для кодера на рисунке 1.2. Кодер имеет четыре состояния, где различные состояния кодера отмечены также буквами *a, b, c, d*.

При изображений путей, черной линией обозначается путь, связанный с нулевым входным битом, а красной линией обозначают путь, связанный единичным входным битом. Состояние кодера обозначается двумя значениями первой и второй запоминающих ячеек. Например, в случае если первая ячейка хранит значение 1 (Q_1=1), а вторая – 0 (Q_2=0), состояние кодера

равно значению 10. Иными словами, возможно четыре различных состояний кодера: 00, 01, 10 и 11 [14, 17].

Предположим, что в некоторый момент времени состояние кодера равно 00. Необходимо определить какой дибит будет сформирован в следующий момент времени.

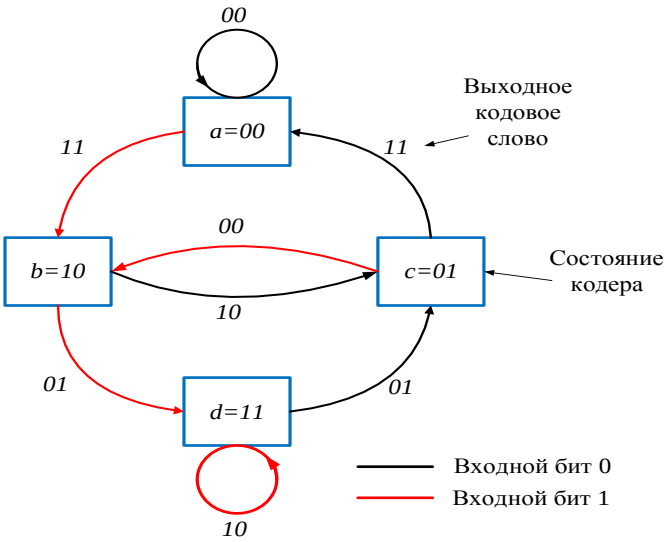


Рисунок 2.1 – Диаграмма состояний кодера (степень кодирования *1/2, m= 3*)

Ответ на данный вопрос зависит от значения бита, поступившего на вход кодера. В случае если на вход поступит 0, значение следующего состояния кодера также будет равным 00, если же поступит 1, состояние кодера после сдвига будет обозначено как 10. Значение формируемых при этом дибитов рассчитывается по формулам:

$$a_i \oplus a_{i-1} \oplus a_{i-2},\ a_i \oplus a_{i-2}. \qquad (2.1)$$

Если на вход кодера поступает 0, то будет сформирован дибит 00:

$$0 \oplus 0 \oplus 0 = 0,\ 0 \oplus 0 = 0.$$

Если же на вход поступает 1, то формируется дибит 11:

$$1 \oplus 0 \oplus 0 = 1,\ 1 \oplus 0 = 1.$$

Приведенные рассуждения удобно представить наглядно с помощью диаграммы состояний (рисунок 2.1).

Представим диаграмму состояний кодера в виде таблицы (таблица 2.1).

Таблица 2.1 – Таблица переходов состояний кодера (*R=1/2, m= 3*)

Состояние	Входной бит	Содержимое регистра сдвига	Выходная комбинация
a=00	1	100	11
	0	000	00
b=10	1	110	01
	0	010	10
c=01	1	101	00
	0	001	11
d=11	1	111	10
	0	011	01

Диаграмма состояний для кодера на рисунке 1.4 показана на рисунке 2.2. Кодер имеет $2^{m-1}=8$ различные состояния и отмечены буквами *a, b, c, d, e, f, g, h*.

Всего возможно восемь различных состояний кодера: 000, 001, 010, 011, 100, 101, 110 и 111.

Так как в кодер вводится по одному биту, то в каждое состояния входят и из каждого состояния исходят по два ветви.

Два бита, показанных на каждой ветви диаграммы состояний, представляют выходные биты. Красная линия означает, что входной бит 1, в то время как черная линия указывает, что входной бит 0.

Предположим, что в некоторый момент времени состояние кодера равно 000. Если на вход кодера поступит 0, то следующее состояние кодера также будет 000, если же напротив поступит 1, следующее состояние кодера после сдвига будет 100. Значение формируемых при этом дибитов рассчитывается по формулам:

$$a_i \oplus a_{i-1} \oplus a_{i-2} \oplus a_{i-3},\ a_i \oplus a_{i-1} \oplus a_{i-3}. \qquad (2.2)$$

Если на вход кодера поступает 0, то на выходе кодера будет сформирован дибит 00, ($0\oplus0\oplus0\oplus0=0,\ 0\oplus0\oplus0=0$), если же на вход поступает 1, то формируется дибит 11 ($1\oplus0\oplus0\oplus0=1,\ 1\oplus0\oplus0=1$).

Приведенные рассуждения удобно представить с помощью диаграммы состояний (рисунок 2.2).

Представим диаграмму состояний кодера в виде таблицы (таблица 2.2).

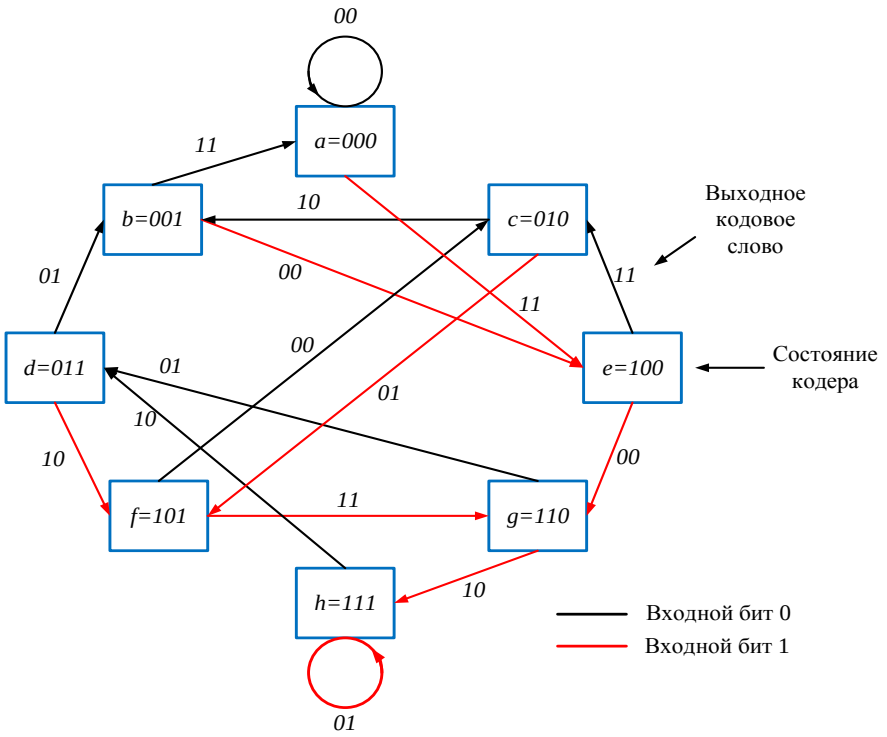


Рисунок 2.2 – Диаграмма состояний кодера (степень кодирования *1/2, m= 4*)

Таблица 2.2 – Таблица переходов состояний кодера (*R=1/2, m= 4*)

Состояние	Входной бит	Содержимое регистра сдвига	Выходная комбинация
a=000	1	1000	11
	0	0000	00
b=001	1	1001	00
	0	0001	11
c=010	1	1010	01
	0	0010	10
d=011	1	1011	10
	0	0011	01
e=100	1	1100	00
	0	0100	11
f=101	1	1101	11
	0	0101	00
g=110	1	1110	10
	0	0110	01
h=111	1	1111	01
	0	0111	10

Диаграмма состояний для кодера на рисунке 1.13 показана на рисунке 2.3. Кодер имеет $2^{m-1}=8$ различных состояний, отмеченных буквами *a, b, c, d, e, f, g, h.*

Всего возможно восемь различных состояний кодера: 000, 001, 010, 011, 100, 101, 110 и 111.

Эти состояния заносятся прямоугольники диаграммы состояний. Каждые 3 бита, показанных на ветвях диаграммы состояний, представляют биты на выходе.

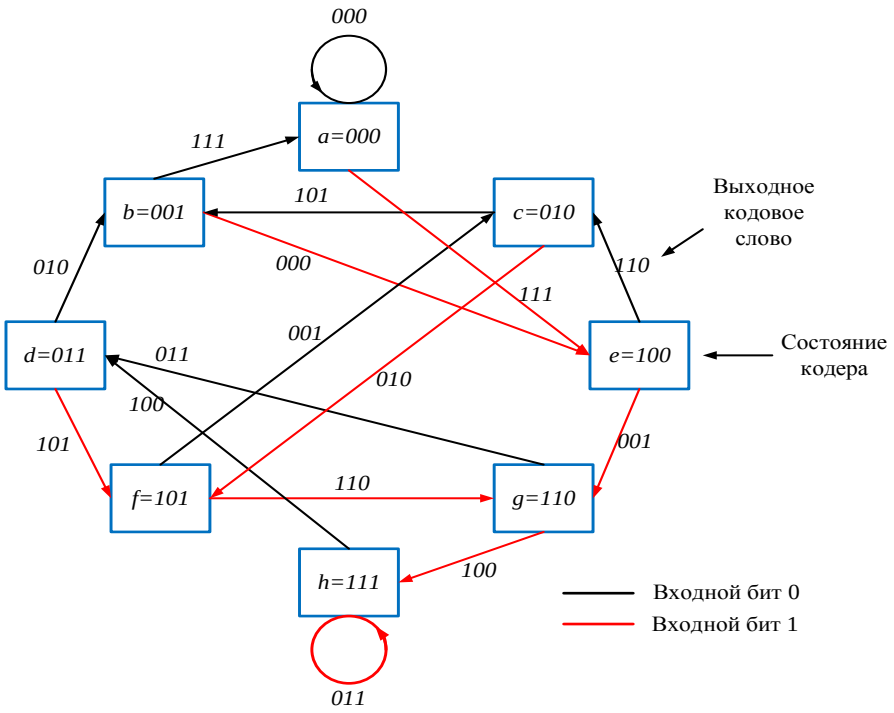


Рисунок 2.3 – Диаграмма состояний кодера (степень кодирования *1/3, m= 4*)

Красная линия означает, что входной бит 1, в то время как черная линия указывает, что входной бит 0.

Пусть в некоторый момент времени состояние кодера равно 000. Если на вход кодера поступит 0, то следующее состояние кодера также будет 000, если же поступит 1, то следующее состояние (то есть после сдвига) будет 100.

Значение формируемых при этом три битов рассчитывается по формулам:

$$a_i \oplus a_{i-1} \oplus a_{i-2} \oplus a_{i-3},\ a_i \oplus a_{i-1} \oplus a_{i-3},\ a_i \oplus a_{i-2} \oplus a_{i-3}. \qquad (2.3)$$

Если на вход кодера поступает 0, то на выходе кодера будет сформирован три бит 000, ($0\oplus0\oplus0\oplus0=0,\ 0\oplus0\oplus0=0,\ 0\oplus0\oplus0=0$).

Если же на вход поступает 1, то на выходе кодера формируется три бит

111 (1⊕0⊕0⊕0=1, 1⊕0⊕0=1, 1⊕0⊕0=1).

Приведенные рассуждения удобно представить с помощью диаграммы состояний (рисунок 2.3).

Представим диаграмма состояний кодера в виде таблицы (таблица 2.3).

Таблица 2.3 – Таблица переходов состояний кодера (*R=1/3, m=4*)

Состояние	Входной бит	Содержимое регистра сдвига	Выходная комбинация
a=000	1	1000	111
	0	0000	000
b=001	1	1001	000
	0	0001	111
c=010	1	1010	010
	0	0010	101
d=011	1	1011	101
	0	0011	010
e=100	1	1100	001
	0	0100	110
f=101	1	1101	110
	0	0101	001
g=110	1	1110	100
	0	0110	011
h=111	1	1111	011
	0	0111	100

Диаграмма состояний для кодера на рисунке 1.18 показана на рисунке 2.4. Кодер имеет четыре различных состояний, отмеченных буквами *a, b, c, d*.

Всего возможно четыре различных состояний кодера: 00, 01, 10 и 11.

В кодер два бита поступают на вход регистров сдвига каждый раз. Первые все возможные два входных бита могут быть 00, 01, 10 или 11.

Черная линия указывает на то, что входной бит 00, красная линия показывает, что входной бит 10, зеленая линия обозначает, что входной бит 01, в то время как синяя линия означает, что входной бит 11.

Пусть в некоторый момент времени состояние кодера равно 00.

Если на вход кодера поступит 00, то следующее состояние кодера также будет 00, если же поступит 1, то следующее состояние (то есть после сдвига) будет 10.

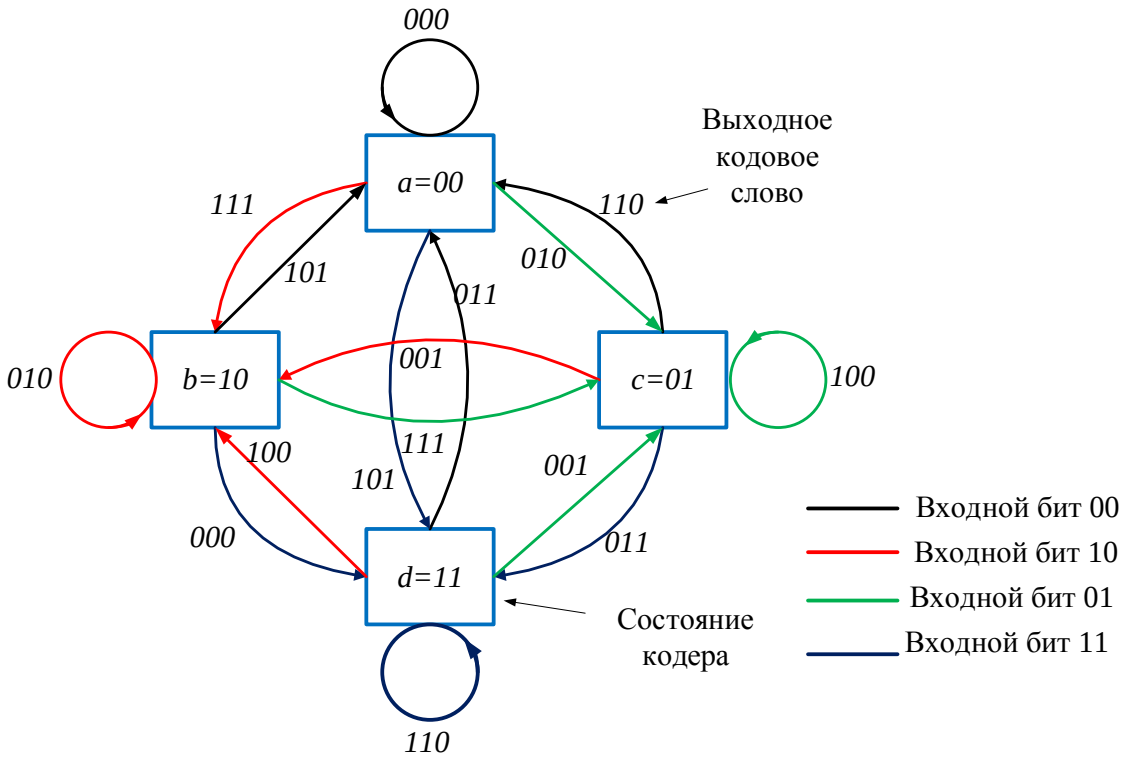


Рисунок 2.4 – Диаграмма состояний кодера (степень кодирования *2/3*, *m=2*)

Значение формируемых при этом дибитов рассчитывается по формулам:

$$a_i \oplus a_{i-2} \oplus a_{i-3},\ a_i \oplus a_{i-1} \oplus a_{i-3},\ a_i \oplus a_{i-2}. \qquad (2.4)$$

Если на вход кодера поступает 00, то на выходе кодера будет сформирован три бит 000, ($0\oplus0\oplus0=0,\ 0\oplus0\oplus0=0,\ 0\oplus0=0$), если же на вход поступает 10, то формируется три бит 111 ($1\oplus0\oplus0=1,\ 1\oplus0\oplus0=1,\ 1\oplus0=1$), если же на вход поступает 01, то формируется три бит 010 ($0\oplus0\oplus0=0,\ 0\oplus1\oplus0=1,\ 0\oplus0=0$), если же на вход поступает 11, то формируется три бит 101 ($1\oplus0\oplus0=1,\ 1\oplus1\oplus0=0,\ 1\oplus0=1$).

Приведенные рассуждения удобно представить с помощью диаграммы состояний (рисунок 2.4).

Представим диаграмма состояний кодера в виде таблицы (таблица 2.4).

Таблица 2.4 - Таблица переходов состояний кодера (*R=2/3, m=2*)

Состояние	Входной бит	Содержимое верхнего регистра сдвига	Содержимое нижнего регистра сдвига	Выходная комбинация
a=00	00	00	00	000
	01	00	10	010
	10	10	00	111
	11	10	10	101
b=10	00	01	00	101
	01	01	10	111
	10	11	00	010
	11	11	10	000
c=01	00	00	01	110
	01	00	11	100
	10	10	01	001
	11	10	11	011
d=11	00	01	01	011
	01	01	11	001
	10	11	01	100
	11	11	11	110

2.2 Древовидная диаграмма

На рисунке 2.5 показана древовидная диаграмма для сверточного кодера (рисунок 1.2).

Древовидная диаграмма (treediagram) прибавляет к диаграмме состояния временное изменение.

Вертикальные линии диаграммы (рисунок 2.5) названы ребрами дерева, горизонтальные линии – ветвями [3-5].

В зависимости от того, какое значение принимает входной сигнал, движение по ребрам может быть: при поступлении на вход нуля происходит движение по верхнему ребру, при поступлении единицы –по нижнему ребру.

Определенная ветвь соответствует определенному кодовому слову на выходе кодера. Каждый узел древовидной диаграммы имеет соответствующее состояние кодера, которое определяется содержимым двух правых регистров кодера. Состояние *a* соответствует

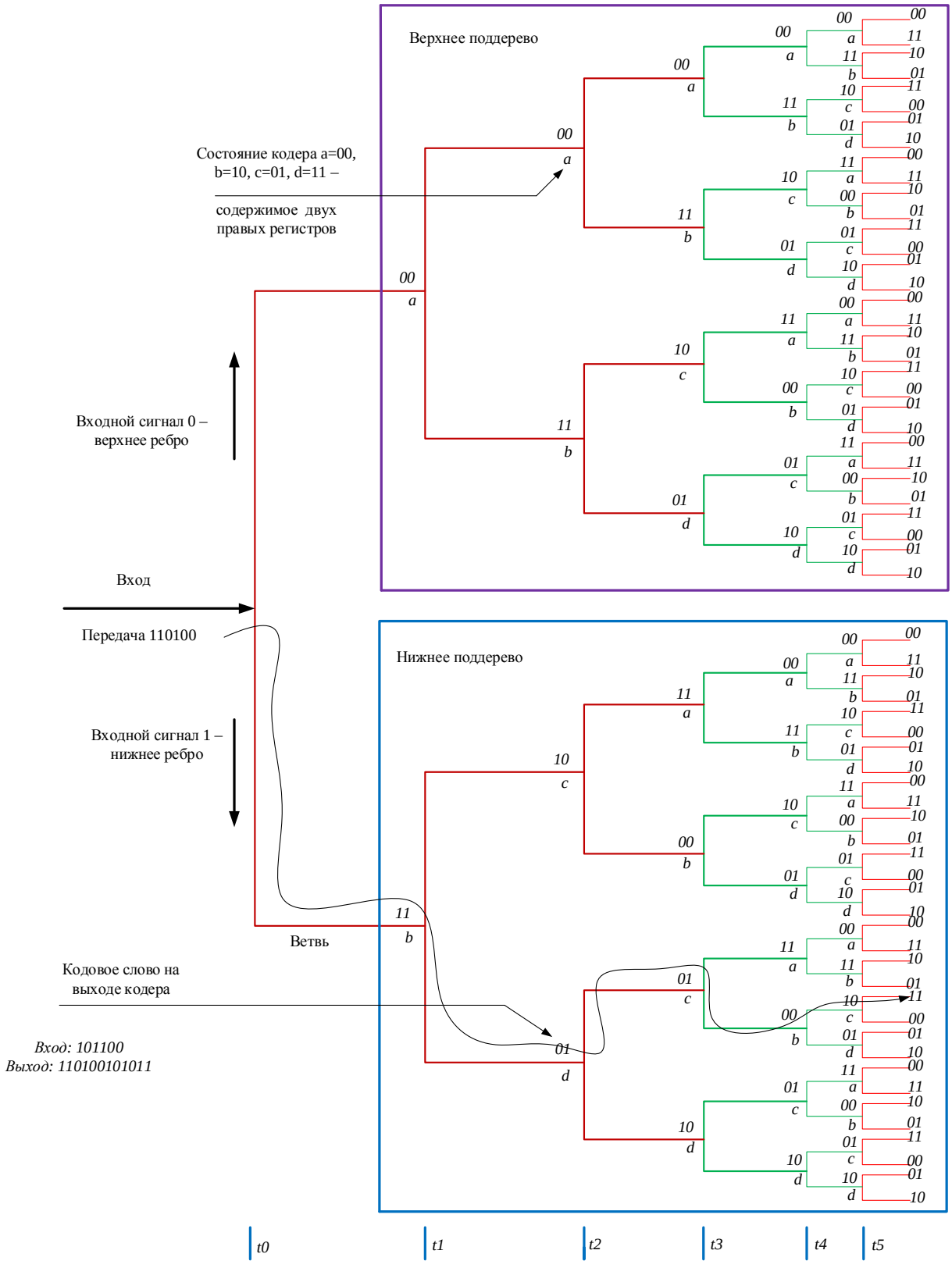


Рисунок 2.5 – Древовидная диаграмма кодера (степень кодирования *1/2*, *m= 3*)

Содержимому 00, b = 10, c = 01, d = 11. Состоянием крайнего левого регистра определяется переход по ребрам дерева.

Далее описан процесс формирования выходной кодовой последовательности символов для входного сигнала (110100).

Вначале состояние регистров кодера 00. При входном сигнале равном«1» движение идет по нижнему ребру дерева, на выход поступает 11. Затем на вход подается второй бит 1, идет движение по нижнему ребру, на выход поступает 01.

Последующий входной бит 0 влечет к переходу по верхнему ребру, что равно выходному сигналу 01. Далее на вход приходит 1, происходит переход по нижнему ребру, на выходе кодера образуется сигнал 00. Далее на вход поступает 0, происходит переход по верхнему ребру, на выходе кодера формируется сигнал 10.

Последний входной бит 0 приводит к переходу по верхнему ребру, на выход поступает 11. Таким образом, входной последовательности (110100) соответствует выходная последовательность (11 01 01 00 10 11).

С ростом длины входной последовательности число возможных путей на древовидной диаграмме резко возрастает, что накладывает ограничения на ее применение [18-20].

Таким образом, получается что начиная с 3-го уровня ветвей верхнее и нижнее поддерево полностью повторяют друг друга.

Период повторения древовидной диаграммы определяется количеством регистров кодера (размером регистра сдвига).

В данном случае все входные символы кодера используются для создания лишь 3 выходных кодовых слов, затем они удаляются из памяти.

2.3 Решетчатая диаграмма

Решетчатая диаграмма (решетка) является разверткой диаграммы состояний во времени.

На решетке состояния показаны узлами, а переходы – соединяющими их линиями. После каждого перехода из одного состояния в другое происходит смещение на один шаг вправо [1, 3-5].

Например, решетчатая диаграмма для кодера (рисунок 1.2) показана на рисунке 2.6, а для кодера 1.13 показана на рисунке 2.7.

На данном рисунке черные линии (ветви) соответствуют переходам, происходящим при получении информационного символа 0, а красные линии (ветви) – при информационном символе 1.

Как видно из решетчатой диаграммы, ее структура в конце«переходного процесса» в кодере становится повторяющейся [8-9].

На рисунке 2.6 структура решетчатой диаграммы повторяется после 3-го такта работы кодера, так как при поступлении в кодер чтевертого информационного символа первый символ пакидает регистр сдвига и далее не влияет на образование кодовых символов.

На рисунке 2.7 повторяемость структуры решетчатой диаграммы будет возможна после 4-го такта работы кодера, так как при поступлении в кодерследующего информационного символа 1-ый символ пакидает регистр

сдвига и затем не используется для получения кодовых символов.

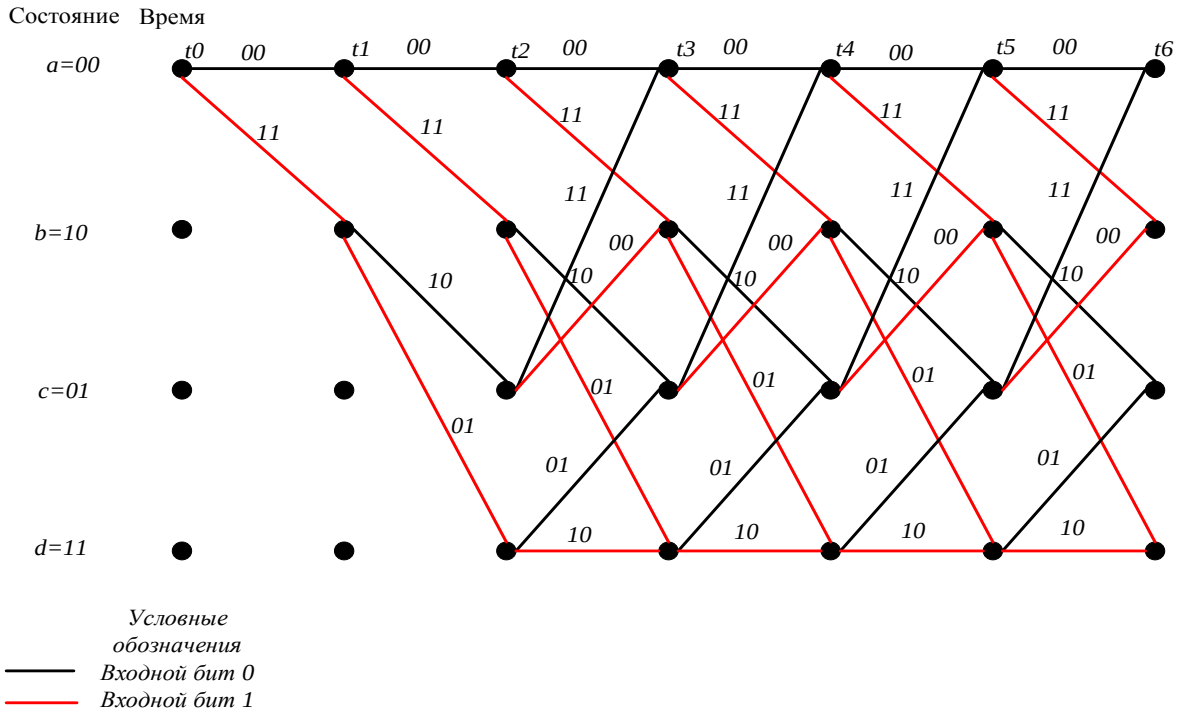


Рисунок 2.6 – Решетчатая диаграмма кодера (степень кодирования *1/2*, *m= 3*)

Удобство решетчатого представленияв том, что с ростом количества входных символов число вершин в решетке не увеличивается, а остается равным 2^{m-1}, где *m*– число ячеек регистра сдвига.

Решетчатая диаграмма дает возможность определить все разрешенные пути, по которым кодер может производитькодирование.

К примеру, при поступлении на вход кодера на рисунке 1.3 последовательности (110100) путь по решетке даст возможность получить выходную последовательность(11 01 01 00 10 11).

При поступлении на вход кодера на рисунке 1.13 последовательности (1101000) путь по решетке даст возможность получить выходную последовательность(111 001 011 101 001 101 111).

Каждая информационная последовательность символов имеет соответствующийуникальный путь (траекторию) на диаграмме.

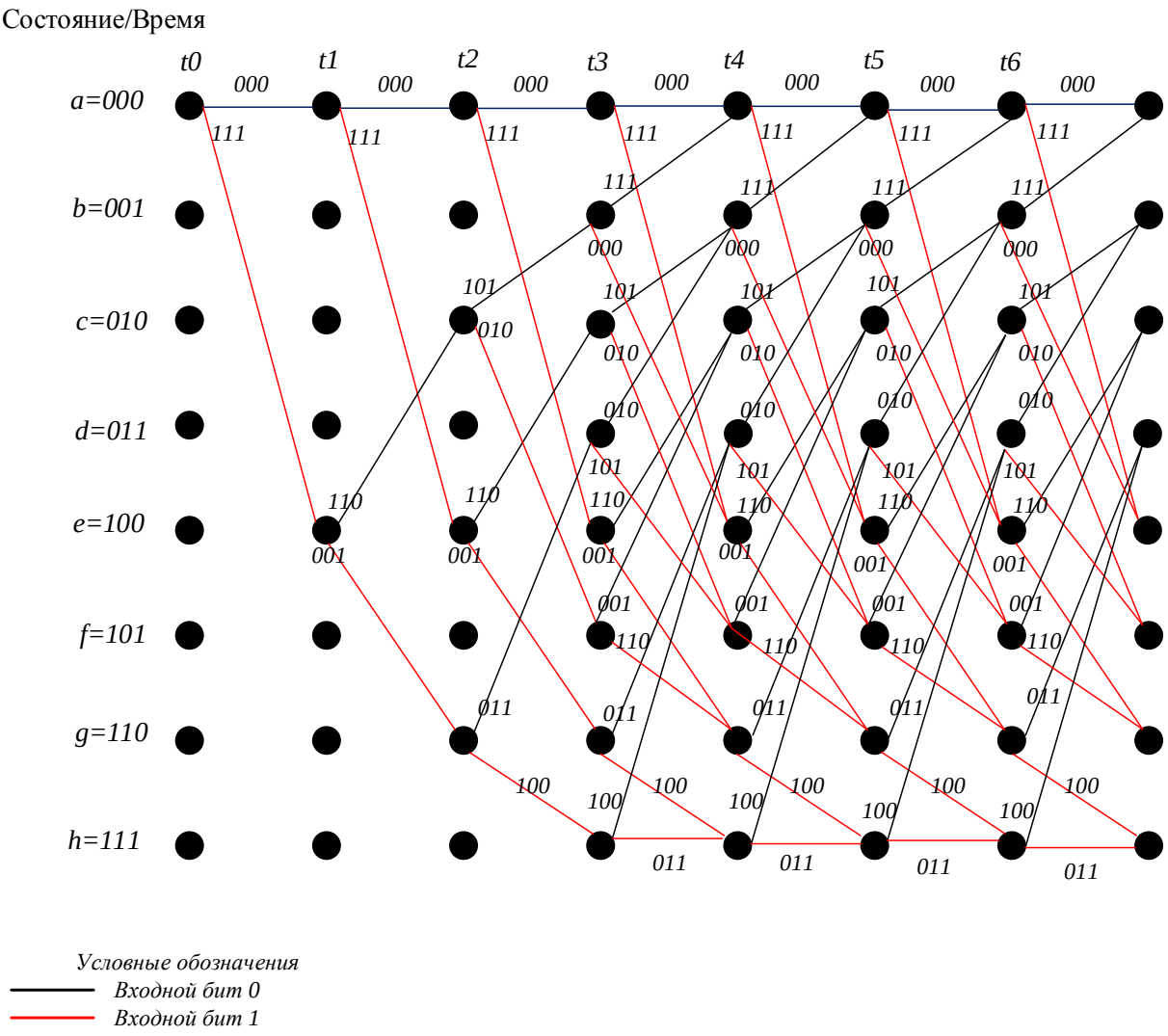


Рисунок 2.7 – Решетчатая диаграмма кодера (степень кодирования *1/3*, *m= 4*)

Кодовая последовательность на выходе формируется посредством считывания кодовых комбинаций над ветвями при отслеживании данной траектории.

2.4 Применение декодирования методом Витерби

Оптимальное декодирование сверточных кодов осуществляется с помощью алгоритма Витерби. Алгоритм декодирования Витерби по эффективности близок к алгоритму максимума правдоподобия. На каждом такте работы декодера Витерби вычисляются метрики рёбер как расстояния между принятым кодовым словом и каждым из путей, входящих в каждое состояние регистра, и метрики состояний путей как суммы сохраняемых метрик предыдущих состояний и метрик рёбер. После сравнения метрик состояний сохраняются метрики выживших путей с наименьшей метрикой.

Сверточные коды непрерывны и могут быть охарактеризованы минимальным расстоянием, определяемым длиною начальных сегментов

кодовых последовательностей. Число ячеек на приемной стороне в декодирующем устройстве определяется числом символов в принятой для обработки длине сегмента L.

Это число символов, хранящееся в памяти декодера с целью дальнейшей обработки принятой кодовой последовательности, называют глубиной декодирования.

В случае, если необходимо обнаружение и исправление максимального количества конфигураций ошибок, то зачастую увеличение глубины декодирования улучшает характеристики, однако в дальнейшем происходит насыщение [4].

Минимальная глубина декодирования должна быть равной длине кодового ограничения l_2 . Однако она в большинстве случаев в несколько раз больше l_2.

Наименьшее расстояние Хеминга для различных пар кодовых слов называется L-м минимальным свободным расстоянием сверточного кодаd_L. При условии, когда L равно l_2, L называется минимальным свободным расстоянием и как и в блочных кодах обозначается d_{min}.

Значение d_L сверточного кода определяется посредством использования диаграммы состояний, либос помощью решеточной диаграммы для соответствующего кодера.

Так как сверточный код относится к линейным, путь с весом равным нулю (нулевой путь) будет обязательно присутствовать среди различных путей на решеточной диаграмме кода. Нулевым путем называют путь, в котором последовательность кодовых символов состоит только из «0». Таким образом, минимальное свободное расстояние сверточного кода равно минимальному числу единиц, иначе говоря, минимальному весу путей, расходящимся и сливающимся с нулевым путем. Решеточная диаграмма кода может быть использована для определения d_L, если для каждой ее ветви записать вес соответствующих кодовых символов на выходе кодера, после чего подсчитать вес путей, расходящихся и сливающихся с нулевым путем.

Исправляющая способность свёрточных кодов оценивается минимальным расстоянием между кодовыми последовательностями, называющимся свободным расстоянием $d_{св}$. [10].

Далее рассмотрим принцип работы алгоритма Витерби для приема из канала i-й n-символьной группы последовательности$\boldsymbol{s}^*$. Рассматриваемые пути могут проходить через 2^{m-1} узлов (состояний) решетчатой диаграммы (где m – число ячеек регистра сдвига), и для каждого из них вычисляется расстояние от принятой последовательности, которое далее будет называться метрикой.

Затем на i-м шаге необходимо [8]:

1. Рассчитать расстояние Хэминга между входнойi-символьной группой и различными ветвями решетчатой диаграммы.

Поскольку из каждого из 2^{m-1} узлов выходит по две ветви, то необходимо вычислить $\boldsymbol{2^m}$таких расстояний.

2. Расстояния Хэмминга для всех ветвей добавляются к метрикам путей, из которых они выходят. В итоге получаются2^m возможных путей, ведущих в 2^{m-1} состояний.

3. Для каждого из 2^{m-1} состояний необходимо сравнить метрики 2-х входящих в него путей и путь с меньшей метрикой, который находится на минимальном расстоянии от пришедшей последовательности, становится выжившим. Путь с наибольшей метрикой не учитывается в дальнейших вычислениях.

4. Запомнить все 2^{m-1} выживших путей вместе с их метриками и перейти к выполнению $(i+1)$-го шага.

Алгоритм Витерби легче всего понять, рассматривая в качестве примера решетчатую диаграмму для сверточного кодера на рисунке 1.3.Решетчатая диаграмма кодера показана на рисунке 2.6. Декодирование по алгоритму Витерби искаженной последовательности с ошибкой первой кратности.

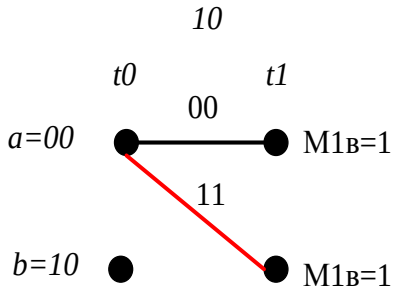
Пусть передается кодовое слово $\boldsymbol{s} = (11\ 01\ 01\ 00\ 10\ 11)$, а в канале произошла однократная ошибка, так, что принятая последовательность имеет вид $\boldsymbol{s}^* = (1\boldsymbol{0}\ 01\ 01\ 00\ 10\ 11)$.

Вычислим растояние Хэминга между принятой 1-ойсимвольной группой (10) и разными ветвями (t_0,t_1) решетчатой диаграммы (рисунок 2.8, а). На шаге 1 из состояния 00 выходят следующие пути: в состояние 00 с растоянием Хэминга до принятой из канала первой символьной последовательности равной 1, и в состояние 10 с метрикой 1 (рисунок 2.8, а).

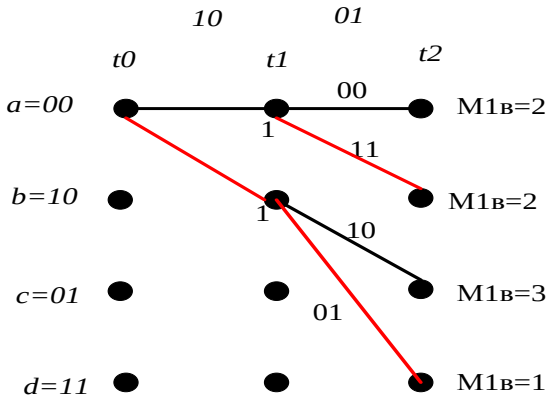
На шаге 2 формируются уже четыре пути, причем метрики состояний 00, 10, 01 и 11 становятся равными 2,2, 3 и 1 соответственно (рисунок 2.8, б).

Следующий 3-ий шаг работы является критичным для понимания всего алгоритма. Рассмотрим состояние 00, к которому можно прийти по двум различным путям: из состояния 00 и из состояния 01. Первый путь будет иметь метрику 3, так как метрика ветви (00-00) равна «1» и предыдущее значение метрики пути равно двум, а второй - четырем, так как метрика ветви (01-00) равна трем, и предыдущая метрика пути равна единице (рисунок 2.8, в). Сравнение метрик этих 2-х путей дает возможность выбрать наилучший, в данном случае это путь из состояния 00, так как он имеет меньшую метрику.

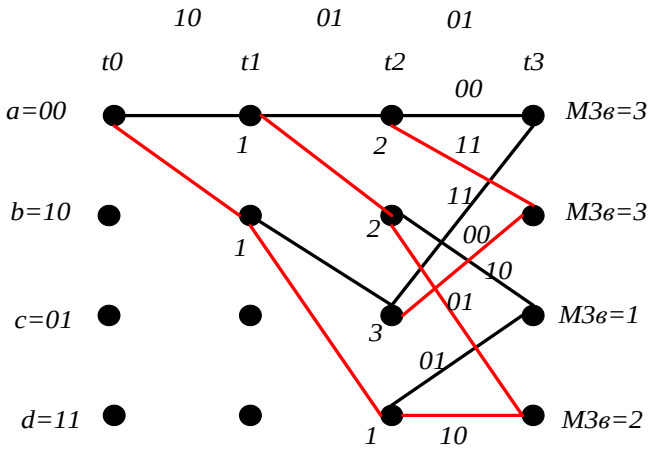
Таким же способом выбираются выжившие пути для остальных узлов решетки. Шаги 4 и 5 полностью аналогичны шагу 3 (рисунок 2.8, г, д). Особенностью четвертого шага является то, что метрики конкурирующих путей, входящих в одно состояние (состояние 00), равны. Простейшим способом разрешения данной неопределенности является случайный выбор выжившего пути.



а)

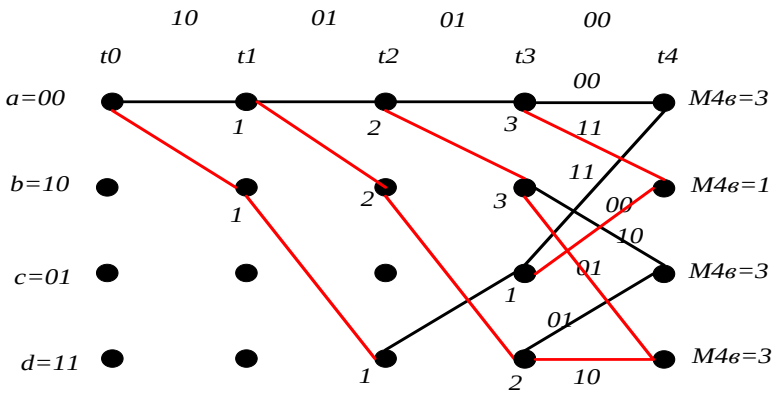


б)

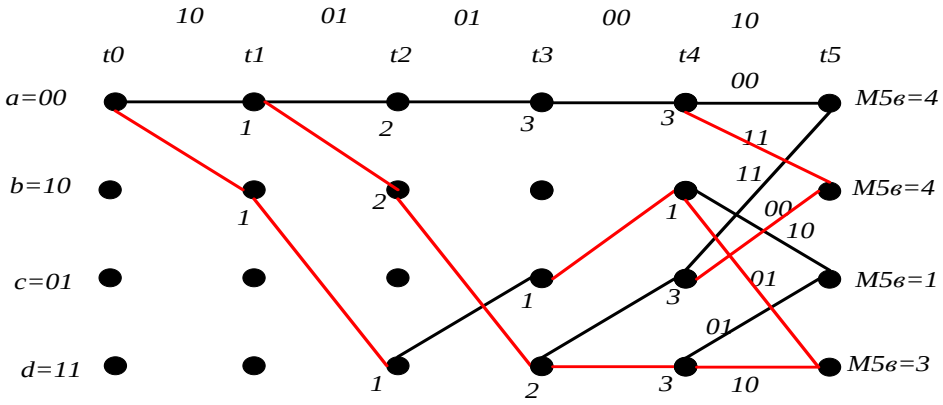


в)

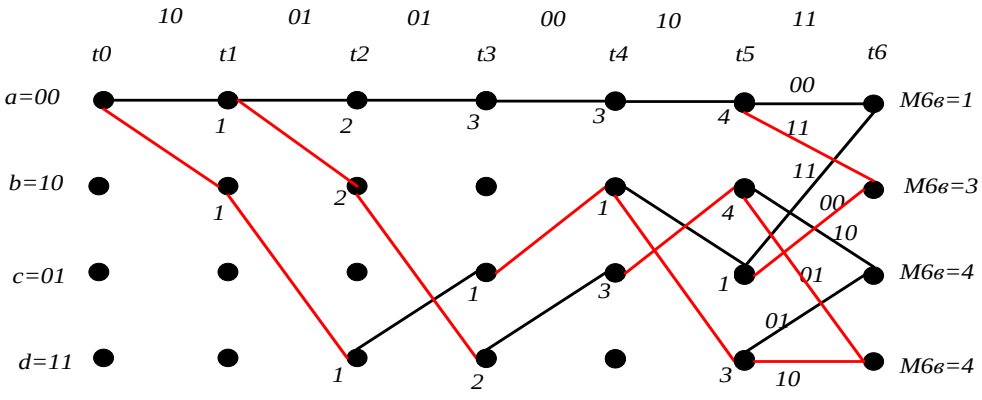
Рисунок 2.8 – Решетчатая диаграмма кодера(степень кодирования *1/2,m=3*)



г)

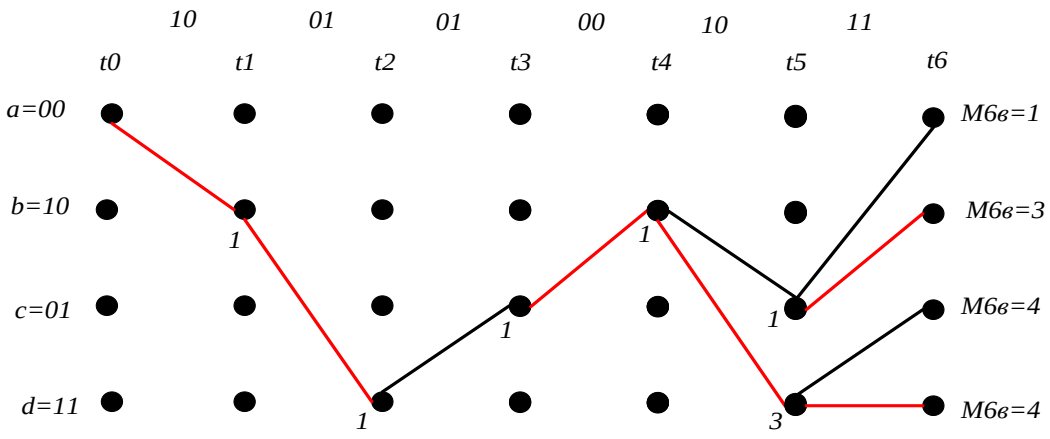


д)



е)

продолжение рисунка 2.8



ж)

продолжение рисунка 2.8

После выполнения шестого шага работы сложилась характерная ситуация, когда начальные части (первые пять ветвей) всех выживших путей слились и теперь можно принимать решение о первых пяти информационных битах (рисунок 2.8, е, ж).

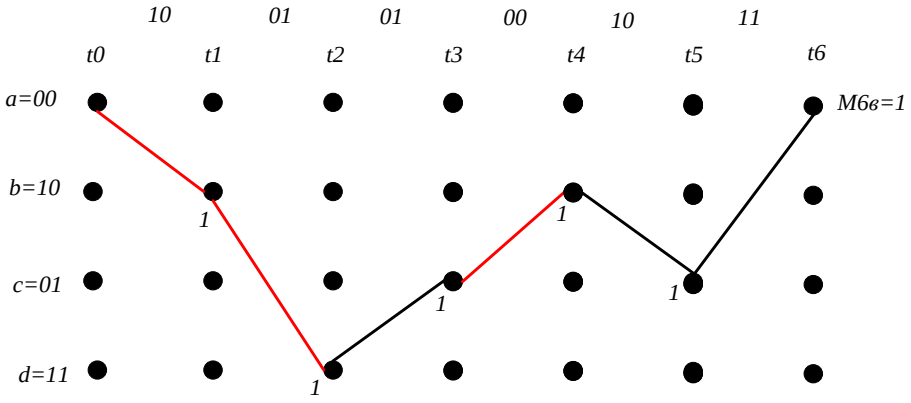


Рисунок 2.9 – Выживший путь

Начальная часть выживших путей определяет информационную последовательность (110100), т.е. одна канальная ошибка была исправлена (рисунок 2.9).

Вывод.

Показаны графическое представление и способы кодирования двоичных сверточных кодов. Разработана методика декодирования сверточных кодов по Витерби для лабораторных работ, которая показана в приложениях А и Б.

3 Характеристики помехоустойчивости сверточных кодов

3.1 Программирование кодирования и декодирования сверточных кодов

Для исследования свойств и проверки показателей качества работы двоичных сверточных кодов была создана модель цифрового канала связи в среде MATLAB. На рисунке 3.1 представлена обобщенная модель цифровой системы связи. Для моделирования модулятора, канала связи и сверточного кодера использовались стандартные объекты и функции MATLAB [23-24].В качестве критерия качества кода использовалась зависимость вероятности битовой ошибки от соотношения сигнал/шум [22].

Декодирование двоичных сверточных кодов производилось с помощью алгоритма Витерби.

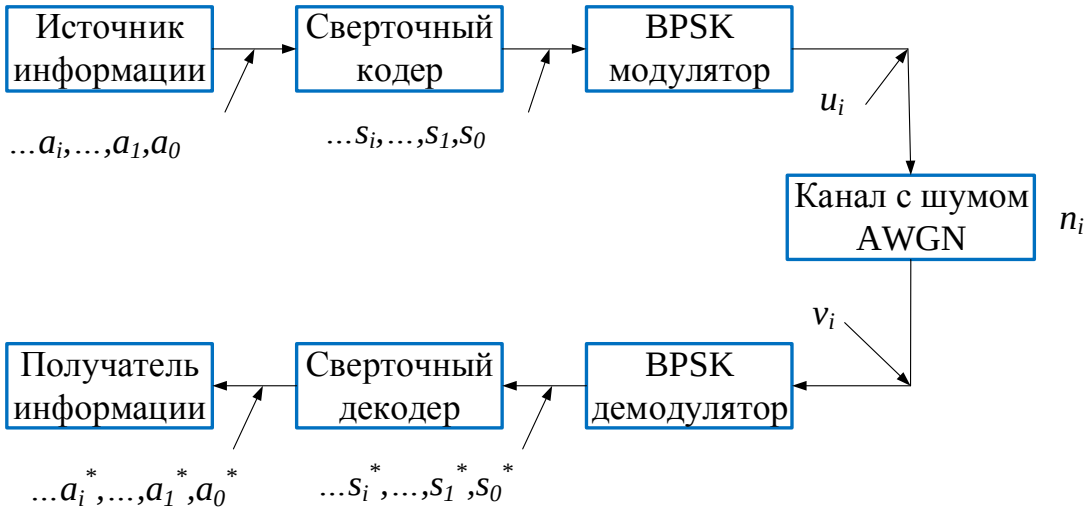


Рисунок 3.1 – Обобщенная схема цифровой системы передачи данных

В данной модели информационная последовательность a_i, задаваемая случайным образом, кодировалась с помощью сверточного кодера. Затем кодовое слово s_i модулировалось с помощью двоичной фазовой модуляции (BPSK), т.е. каждый отсчет выходного сигнала представлялся как 1 или -1. К сигналу u_i добавлялась случайная величина n_i (аддитивный белый гауссовский шум). Далее принятое сообщение демодулировалось и декодировалось, а на выходе системы получалась последовательность символов $\ldots a_i^*,\ldots,a_1^*,a_0^*$.Эта последовательность может отличаться от входной последовательности $\ldots a_i,\ldots,a_1,a_0$ из-за наличия ошибочных символов [21].

Рассмотрим процесс кодирования и декодирования по Витерби в этой системе, применяя программирование в среде MATLAB [23-24].

Пусть на передающей стороне информационная последовательность длинойN бит поступает на вход сверточного кодера со скоростью кодирования

1/2 и имеет число ячеек регистра сдвига, равное трем. На приемной стороне используется декодер Витерби.

Блок-схема программы сверточного кодера в среде MATLABпоказана на рисунке 3.2.

На вход кодера поступают *N*информационных символов. Вначале ячейки регистра сдвига располагаются в нулевом состоянии. Когда на вход кодера поступает первый информационный символ, он занимает место первой ячейки регистра сдвига.

Содержимое первой ячейки регистра сдвига занимает место второй ячейки.

Содержимое второй ячейки регистра сдвига занимает место третьей ячейки.

Сумматоры по модулю два вычисляют выходное кодовое слово и это кодовое слово появляется на выходе кодера.

На вход кодера поступает второй информационный символ.

Сдвиг в регистре мы осуществляем с помощью цикла. Индекс *i* меняется от двух до *N*.Условие: если *i*меньше или равно от *N*, тогда место первой ячейки регистра сдвига занимает *i*-ой информационный символ.

Далее сумматоры по модулю два вычисляют выходное кодовое слово и это кодовое слово появляется на выходе кодера.

Цикл повторяется до тех пор, пока на вход кодера поступает последний информационный символ. Если *i*>*N*, тогда кодирование будет завершено.

Блок-схема программы сверточного декодера по алгоритму Витерби в среде MATLAB показана на рисунке 3.3.

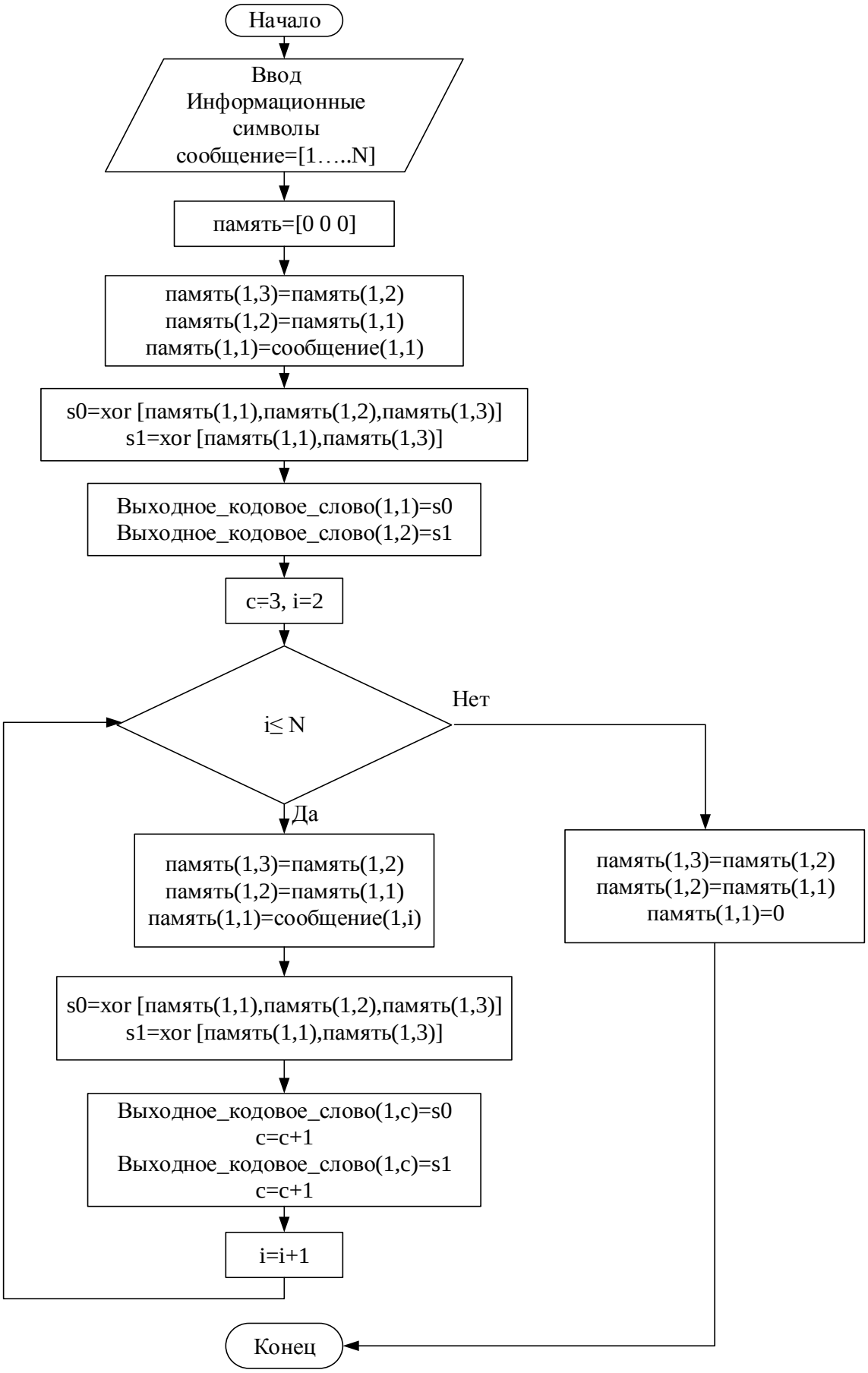


Рисунок 3.2 – Блок-схема программы сверточного кодера

На вход декодера поступает кодированная последовательность. Решетчатая диаграмма декодера имеет 4 состояния: *a=00, b=10, c=01, d=11* (рисунок 2.8).

Совокупность Узел=*[4, N+1]* определяет вертикальные и горизонтальные узлы решетчатой диаграммы. Метрика=*[8, N]* определяет число метрики. Начальное состояние 00 (Узел(1,1)).

Индекс *i* меняется от 1 до *N*. Найдем восемь метрик на *i*-м такте.

Далее выбираем минимальную метрику, исходящую из каждого узла. Этот процесс повторяется до тех пор, пока *i* не больше *N*. Когда *i* больше *N*, процесс буден завершен.

Далее найдем минимальную метрику на последнем такте работы декодера (*N+1* такт).

Далее с помощью файл-функции «prev_stage» будем восстанавливать декодированную последовательность.

Принцип работы файл-функции «prev_stage». Файл-функция prev_stage знает последнюю минимальную метрику и то, на каком узле она находится.

На этот узел есть два пути. Если узел – первое состояние, тогда в узел входят пути (узел (1) + метрика (1), узел (3) +метрика (5)). Если метрика пути (узел (1) + метрика (1)) меньше или равна метрики пути (узел (3) + метрика (5)), тогда декодированный бит 0 и состояние 1, в противном случае, декодированный бит 0 и состояние 3.

Если узел – второе состояние, тогда в узел входят пути (узел (1) + метрика (2), узел (3) +метрика (6)). Если метрика пути (узел (1) + метрика (2)) меньше или равна метрики пути (узел (3) + метрика (6)), тогда декодированный бит 1 и состояние 1, в противном случае, декодированный бит 1 и состояние 3.

Если узел – третье состояние, тогда в узел входят пути (узел (2) + метрика (3), узел (4) +метрика (7)). Если метрика пути (узел (2) + метрика (3)) меньше или равна метрики пути (узел (4) + метрика (7)), тогда декодированный бит 0 и состояние 2, в противном случае, декодированный бит 0 и состояние 4.

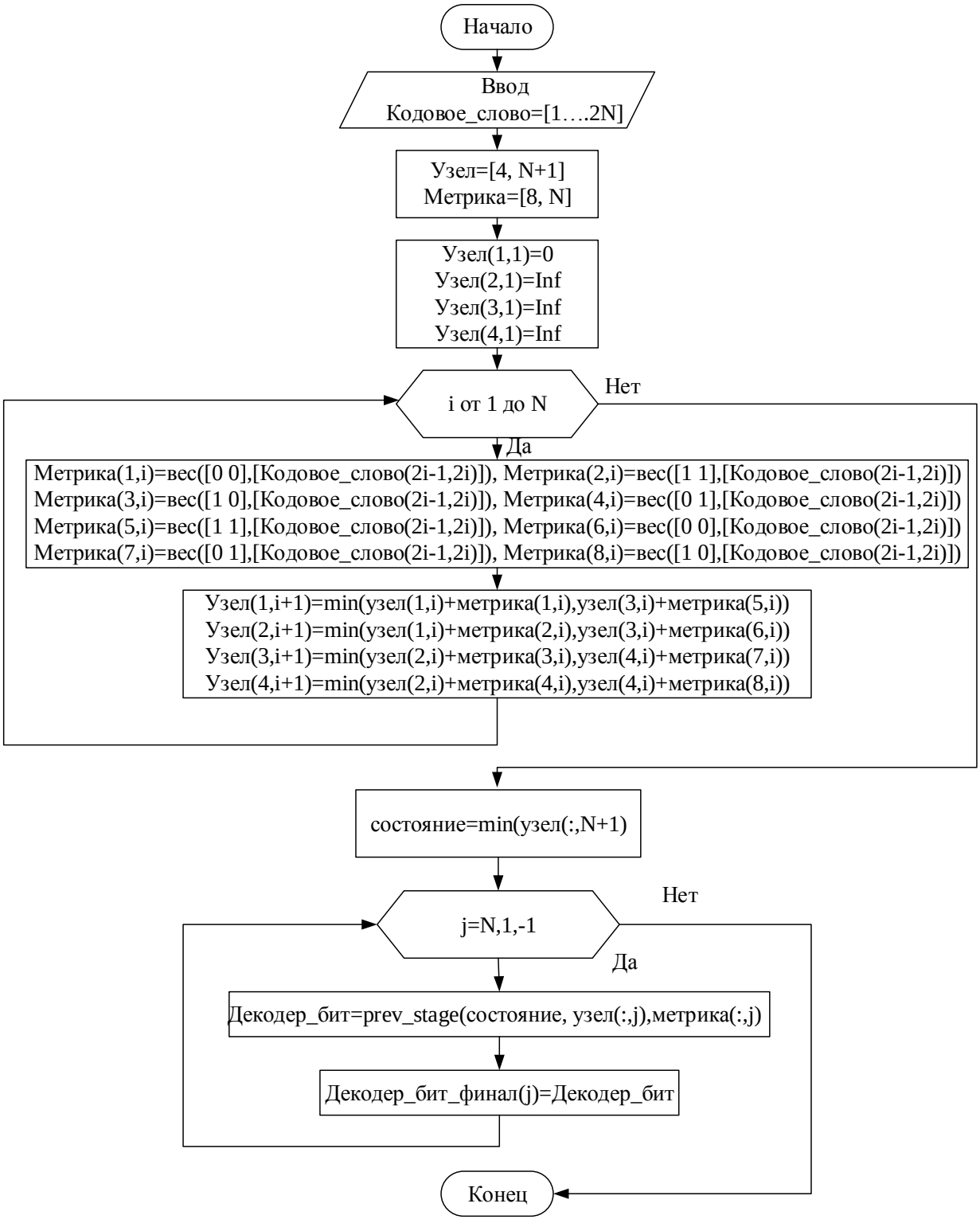


Рисунок 3.3 – Блок-схема программы сверточного декодера по алгоритму Витерби

Если узел - четвертое состояние, тогда в узел входят пути (узел (2) + метрика (4), узел (4) +метрика (8)). Если метрика пути (узел (2) + метрика (4)) меньше или равна метрики пути (узел (4) + метрика (8)), тогда декодированный бит 0 и состояние 2, в противном случае, декодированный

бит 0 и состояние 4. Блок-схема программы файл-функции «prev_stage» в среде MATLAB показана на рисунке 3.4.

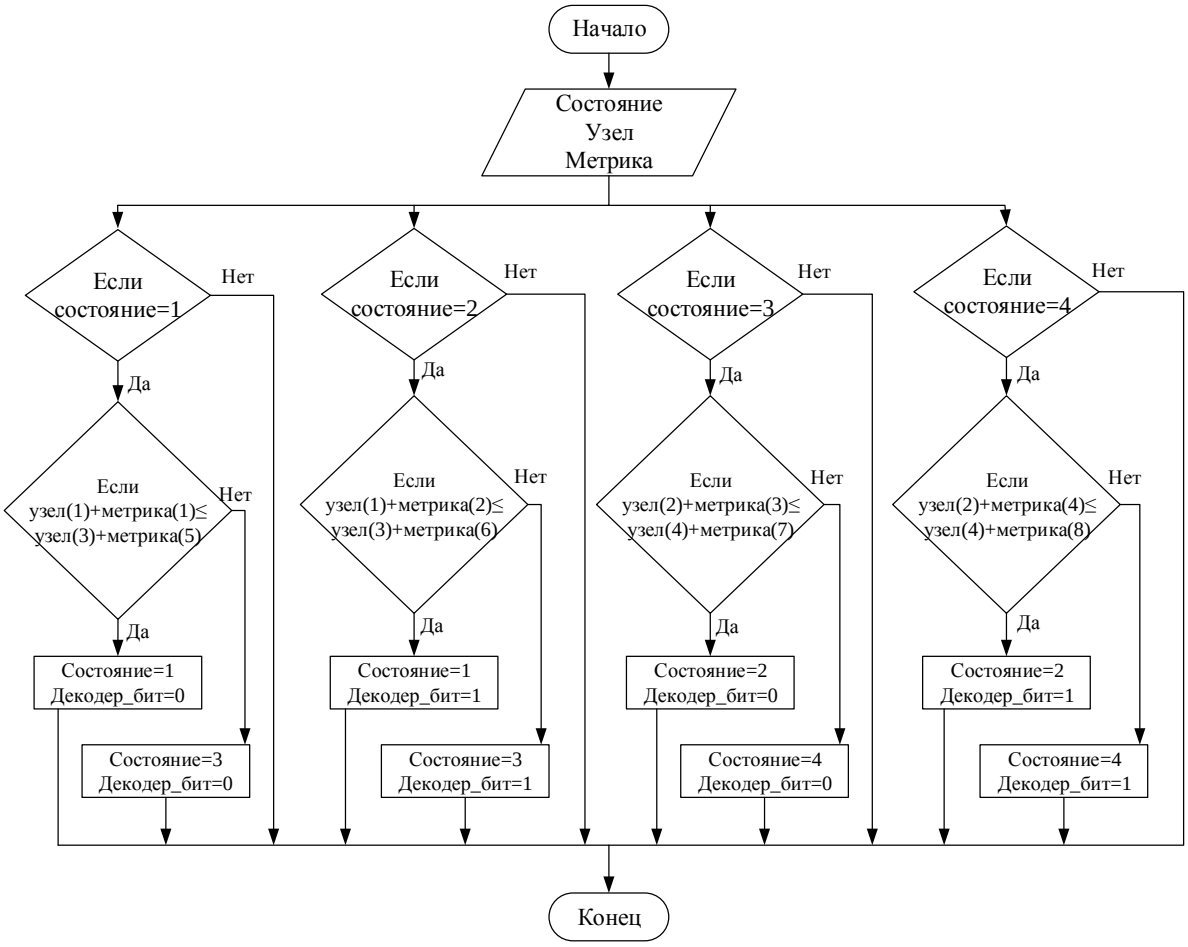


Рисунок 3.4 - Блок-схема программы файл-функции «prev_stage»

С помощью блок схемы модулируем кодера и декодера в среде MATLAB. Моделирование сверточного кодера показана в приложении В.

Далее используем этот модель для исследования помехоустойчивости двоичных сверточных кодов.

3.2 Исследование вероятности битовой ошибки

На основании сравнения декодированной информационной последовательности a_i^* и переданной информационной последовательности a_i составлялась статистика появления ошибок.

Вычисление точных выражений для вероятностей битовых ошибок – это весьма сложная задача даже для простейших кодов (по Витерби).

Было проведено сравнение кодов со скоростью *1/2, 1/3, 1/4*. При анализе сверточных кодов со скоростью равной*1/2* сравнивались коды с кодовым ограничением $m \leq 8$.Результаты моделирования этих кодов приведены на рисунке 3.5.

Статистические данные приведены в таблице 3.1.

Таблица 3.1 – Статистика сверточные коды со скоростью 1/2

С/Ш, дБ	Без код.	*m=3*	*m=4*	*m=5*	*m=6*	*m=7*	*m=8*
	Вероятность битовой ошибки						
1	0,131	0,131	0,1531	0,2012	0,2051	0,2549	0,2645
2	0,1043	0,0719	0,0828	0,108	0,1012	0,12	0,1186
3	0,079	0,0329	0,0337	0,0432	0,0314	0,034	0,0287
4	0,0565	0,0115	0,0102	0,0128	0,0065	0,0053	0,0034
5	0,0375	0,0032	0,0025	0,0026	0,000901	0,000509	0,000251
6	0,0228	0,000607	0,000406	0,000398	0,000061	0,00003	0,000001
7	0,0124	0,000082	0,000041	0,000056	0,000007	10^{-6}	10^{-7}
8	0,0059	0,00008	0,000001	0,000004	10^{-7}	10^{-7}	10^{-8}
9	0,0027	0,00001	10^{-7}	0,000001	10^{-8}	10^{-8}	10^{-9}

При анализе сверточных кодов со скоростью 1/3 сравнивались коды с кодовым ограничением $m \leq 8$.

Результаты моделирования этих кодов приведены на рисунке 3.6.

Статистические данные приведены в таблице 3.2.

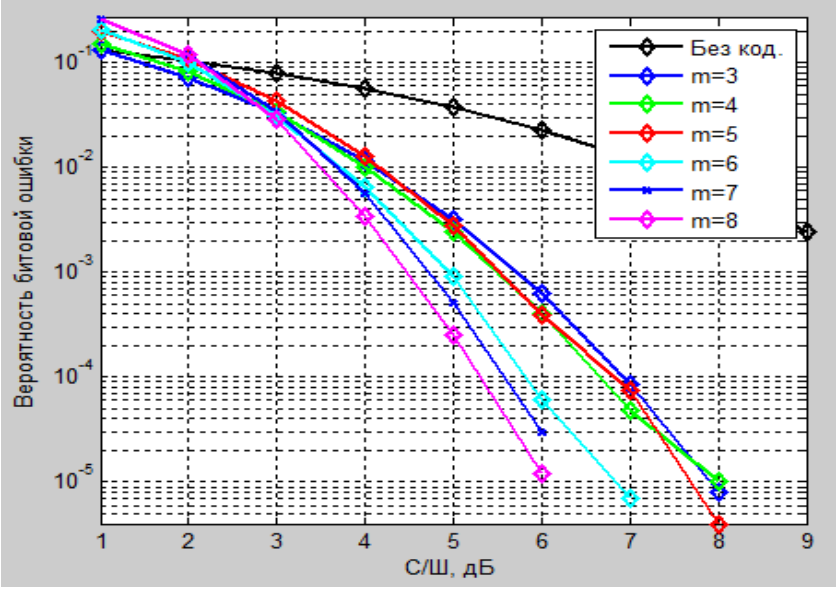


Рисунок 3.5 – Сравнение помехоустойчивости сверточных кодов с скоростью 1/2 и без кодирования

Таблица 3.2 – Статистика сверточные коды со скоростью 1/3

С/Ш, дБ	Без код.	*m*=3	*m*=4	*m*=5	*m*=6	*m*=7	*m*=8
	Вероятность битовой ошибки						
1	0,131	0,0614	0,0352	0,0339	0,034	0,0212	0,019
2	0,1043	0,0247	0,0115	0,0084	0,0079	0,0029	0,0021
3	0,079	0,0076	0,0028	0,0015	0,001	0,000244	0,000146
4	0,0565	0,0017	0,000443	0,000204	0,000091	0,000016	0,000014
5	0,0375	0,000342	0,000061	0,00001	0,000005	0,000003	10^{-6}
6	0,0228	0,000045	0,000008	10^{-6}	10^{-6}	10^{-7}	10^{-7}
7	0,0124	10^{-6}	10^{-6}	10^{-7}	10^{-7}	10^{-8}	10^{-8}
8	0,0059	10^{-7}	10^{-7}	10^{-8}	10^{-8}	10^{-9}	10^{-9}

При анализе сверточных кодов со скоростью 1/4 сравнивались коды с кодовым ограничением $m \leq 6$.

Результаты моделирования этих кодов приведены на рисунке 3.7. Статистические данные приведены в таблице 3.3.

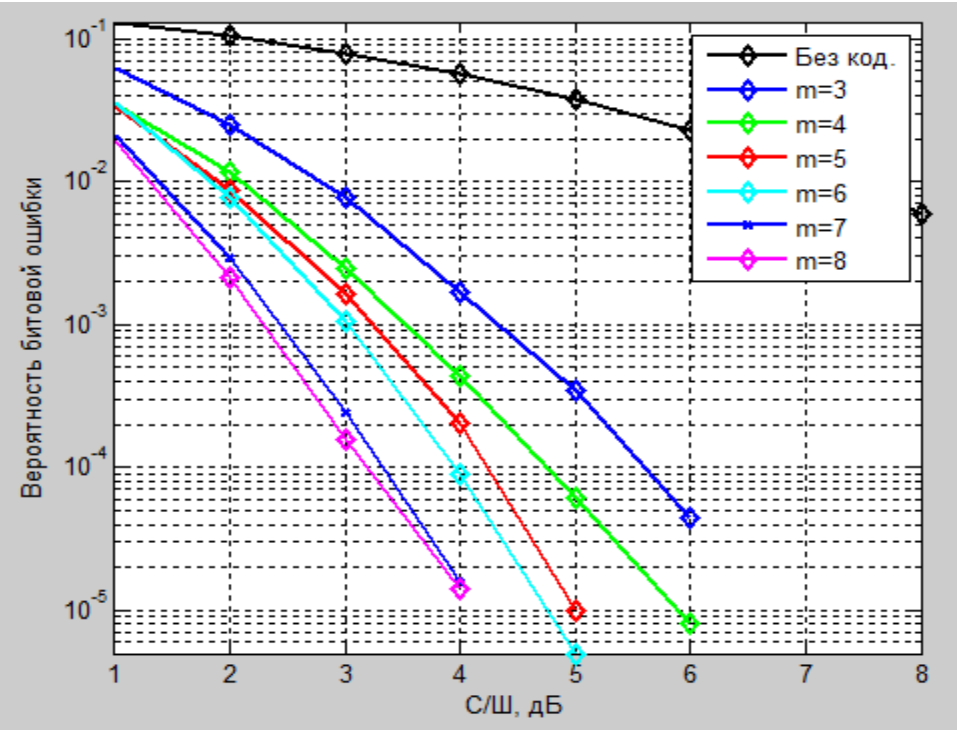


Рисунок 3.6 – Сравнение помехоустойчивости сверточных кодов с скоростью 1/3 и без кодирования

Таблица 3.3 – Статистика сверточные коды со скоростью 1/4

С/Ш, дБ	Без код.	*m=3*	*m=4*	*m=5*	*m=6*
	Вероятность битовой ошибки				
1	0,131	0,0229	0,0083	0,004	0,0025
2	0,1043	0,0064	0,0015	0,000551	0,000313
3	0,079	0,0014	0,000223	0,000071	0,000024
4	0,0565	0,000236	0,000017	0,000006	10^{-6}
5	0,0375	0,000005	10^{-6}	10^{-7}	10^{-7}
6	0,0228	10^{-6}	10^{-7}	10^{-8}	10^{-8}
7	0,0124	10^{-7}	10^{-8}	10^{-9}	10^{-9}

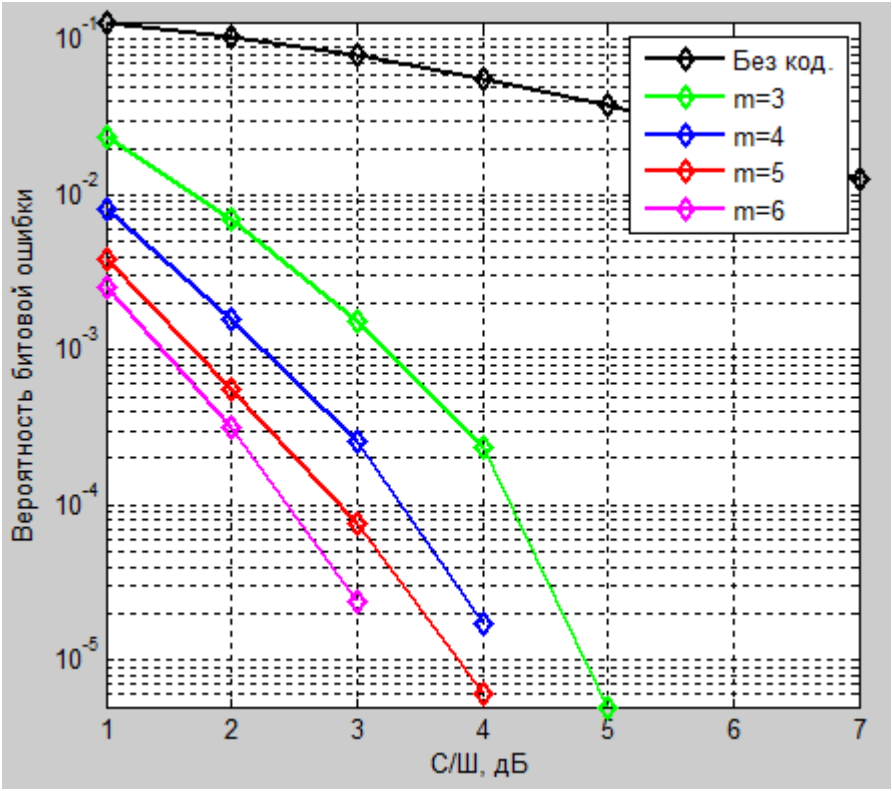


Рисунок 3.7 – Сравнение помехоустойчивости сверточных кодов скоростью 1/4 и без кодирования

Как следует из анализа кривых на рисунках 3.5 – 3.7 оценка вероятности битовой ошибки при применении сверточного кодирования с повышением число ячеек регистра сдвига *m*на единицу помехоустойчивость от кодирования увеличивается примерно от 0,2 до 0,5.

При одинаковых значениях *m* выигрыш от кодирования увеличивается примерно от 1 до 2.

Зададимся длиной сообщения *N*=10 бит и *N*=100 бит. Передачу сообщения будем осуществлять сверточным кодам с параметрами *R=1/2,m=3*.

Результаты моделирования приведены на рисунке 3.8.Статистические данные приведены в таблице 3.4.

Анализ графиков показывает, что применение сверточного кодирования с декодированием Витерби для передачи коротких сообщений (*N*≤10) нецелесообразно. Эффективность использования алгоритма Витерби при передаче сообщения большой длины (*N*=100) очевидна. Если в процессе декодирования сделана ошибка в выборе пути, то в течение нескольких тактов декодер вновь выходит на правильный путь. Увеличение длины сообщения приводит к увеличению шагов декодирования, что позволяет значительно повысить корректирующие способности сверточных кодов.

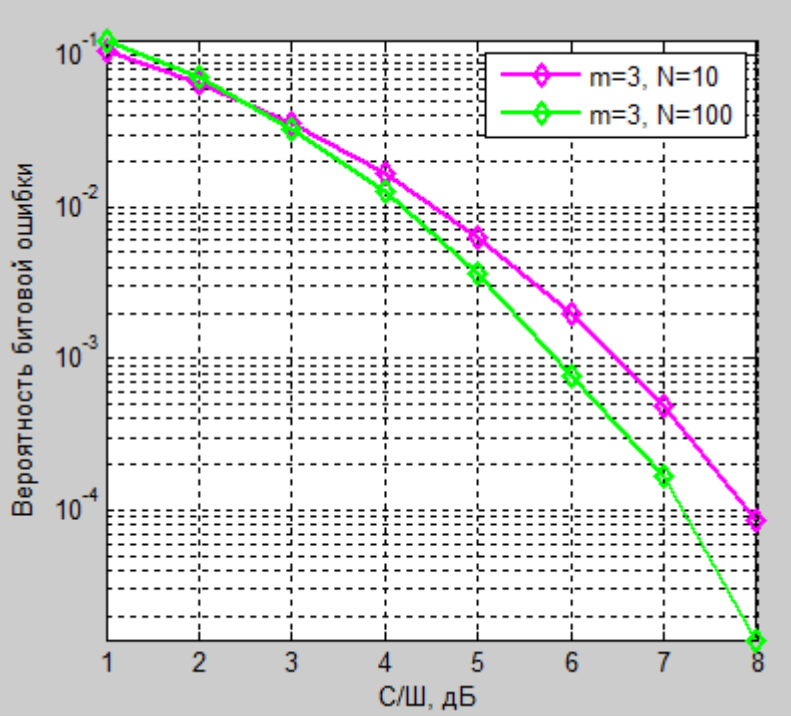


Рисунок 3.8 – Декодирование сверточного кода по Витерби при *N*=10 и *N*=100

В алгоритме Витерби те пути решетки, о которых заранее известно , что согласно принципу максимального правдоподобия они не могут быть оптимальными, не рассматриваются. Увеличение длины сообщения приводит к увеличению шагов декодирования. Предварительный отказ от маловероятных путей упрощает процесс декодирования. Сложность же алгоритма Витерби растет экспоненциально с ростом числа ячеек регистра сдвига.

Таблица 3.4 – Статистика декодирования по Витерби

С/Ш, дБ	1	2	3	4	5	6	7	8
N	Вероятность битовой ошибки							
10	0,1058	0,0657	0,0358	0,0165	0,0063	0,002	0,000485	0,000087
100	0,126	0,0712	0,0329	0,0125	0,0037	0,000764	0,000166	0,000014

Вывод.

Разработаны программные модули кодирования и декодирования по Витерби сверточных кодов.

Проведено исследование помехоустойчивости двоичных сверточных кодов в канале с АБГШ. Были исследованы зависимости вероятности ошибки от отношения сигнал-шум без сверточного кодирования и со сверточным кодированием. При моделировании со сверточным кодированием использовалась модель кодера со скоростью 1/2, 1/3, 1/4 и с малой длиной кодового ограничения ($m \leq 8$) [8].

При увеличении числа ячеек регистра кодера возрастает помехоустойчивость кодирования. Это можно объяснить возрастанием полной длины кодового ограничения по выходу, то есть увеличением числа зависимых выходных кодовых символов от информационных символов.

Помехоустойчивость уменьшается при увеличении скорости кода.

Применение сверточного кодирования и декодирования по Витерби эффективно при передаче сообщений большой длины.

Заключение

В данной диссертационной работе рассмотрены вопросы кодирования и декодирования двоичных сверточных кодов.

Основные результаты работы можно сформулировать следующим образом:

1 Изучены способы кодирования и исследованы влияния параметров на работу сверточных кодеров.

2 Изучены способы кодирования сверточных кодов с помощью диаграмм состояний, древовидных диаграмм и решетчатых диаграмм.

3 Разработаны принципиальные электрические схемы кодеров в среде QUARTUSII, которые могут быть «зашиты» в кристалл программируемых логических интегральных схем (ПЛИС).

4 Доработаны программные модули кодирования и декодирования по Витерби сверточных кодов в среде MATLAB,позволяющие исследовать помехоустойчивость сверточных кодов.

5 Исследованы зависимости вероятностей битовой ошибки от отношения сигнал/шум при различным числе ячеек регистра (m) и различном количестве кодовых символов (n). При этом применяется блочный режим кодирования.

6 Разработаны для учебного процесса две лабораторные работы по кодированию и декодированию сверточных кодов.

Таким образом, решены все поставленные задачи.

Полученные результаты позволяют сделать следующие выводы:

1. Разработанные принципиальные электрические схемы кодеров в среде QUARTUSII могут быть «зашиты» в кристалл программируемых логических интегральных схем (ПЛИС).

2. При увеличении числа ячеек регистра кодера возрастает помехоустойчивость кодирования. Это можно объяснить возрастанием полной длины кодового ограничения по выходу, то есть увеличением числа зависимых выходных кодовых символов от информационных символов.

3. Помехоустойчивость уменьшается при увеличении скорости кода.

4. Применение сверточного кодирования и декодирования по Витерби эффективно при передаче сообщений большой длины.

5. Разработанные виртуальные лабораторные работы являются хорошим «тренажером» при освоении теории и практики по сверточным кодам.

Список литературы

1 Р. Морелос-Сарагоса. Искусство помехоустойчивого кодирования. Методы, алгоритмы, применение. – М.: Техносфера, 2005. – 320 с.

2 Блейхут Р. Теория и практика кодов, контролирующих ошибки: Пер. с англ. – М.: Мир, 1986. – 576 с.

3 М. Вернер. Основы кодирования. Учебник для вузов/ М.: Техносфера, 2004. – 288 с.

4 Никитин Г.И.Сверточные коды: Учебное пособие. – :СПбГУАП. СПб., 2001. – 80 с.

5 Скляр Б. Цифровая связь. Теоретические основы и практическое применение. Изд. 2-е, испр.: Пер. с англ. – М.: Издательский дом «Вильямс», 2033. – 1104 с.

6 Акулиничев Ю.П. Теория электрической связи: Учебное пособие. – СПб.: Издательство «Лань», 2010. – 240 с.

7 Быков В. В., Меньшиков К. В. Помехоустойчивые коды цифрового телевидения. Технологии информационного общества, 2013, №9.

8 Золотарёв В. В., Овечкин Г. В. Помехоустойчивые кодирование. Методы и алгоритмы: Справочни/ Под. ред. Чл. – кор. РАН Ю. Б. Зубарева. – М.: Горячая линия – Телеком, 2004. – 126 с.

9 Витерби А. Д., Омура Дж. К. Принципы цифровой связи и кодирования: Пер. с англ./ Под ред. К. Ш. Зигангирова. – М.: Радио и связь, 1982. – 536 с.

10 Выскубова, Е. А., Хмырова Н. П. Применение пакета Simulink системы MATLAB при разработке программ сверточных кодеков. Техника радиосвязи, 2010, Выпуск 15.

11 Касами Т., Токура Н., Ивадари Е., Инагаки Я. Теория кодирования. Пер. с япон. А. В. Кузнецова / Под ред. Б. С. Цыбакова и С. И. Гельфанда М.: Мир, 1978. – 574 с.

12 Кларк Дж., Кейн Дж. Кодирование с исправлением ошибок в системах цифровой связи: Пер. с англ. – М.: Радио и связь, 1987. – 392 с.

13 У. Питерсон, Э Уэлдон. Коды, исправляющие ошибки. Пер. с англ. под ред. Р.Л. Добрушина и Самойленко. – М.: Изд. «Мир», 1976. – 595 с.

14Банкет В.Л. Сигнально-кодовые конструкции в телекоммуникационных системах. – Одесса: Феникс, 2009. – 180 с.

15 Комолов Д.А., Мяльк Р.А., Зобенко А.А., Филиппов А.С. Системы автоматизированного проектирования фирмы AlteraMAX+plusII и QuartusII. Краткое описание и самоучитель. – М.: ИП РадиоСофт, 2002. – 352 с.

16 Фрике К. Вводный курс цифровой электроники М.: Техносфера, 2003. – 432 с.

17 Банкет В. Л.Помехоустойчивое кодирование в телекоммуникационных системах: учебное пособие по изучению модуля 4 дисциплины ТЭС / В. Л. Банкет, П.В. Иващенко, Н. А. Ищенко. – Одесса: ОНАС им. А. С. Попова, 2011. – 104 с.

18 Королев А. И. Коды устройства помехоустойчивого кодирования информации. – Мн.: 2002. – 286 с.

19 Шульгин В. И. Основы теории передачи информации. Ч. 2. Помехоустойчивое кодирование. Учеб. пособие. – Харьков: Нац. аэрокосм. ун-т «Харьк. Авиац. ин-т», 2003. – 87 с.

20 Думачев В. Н. Теория информации и кодирования – Воронеж: Воронежский институт МВД России, 2012. -200 с.

21 Карпов Д. К., Куликов В. В., Манаенко С. С., Солчатов М. Э. Сравнительная характеристика блочных и сверточных кодов. Инфокоммуникационные технологии, Том 5, 2007, №3.

22 ПрокисД. Цифровая связь. Пер. с англ. / Под ред. Д.Д. Кловского. - М.: Радио и связь, 2000. - 800 с.

23 Дьяконов В.П. MATLAB 7.*/R2006/R2007: Самоучитель. – М.: ДМК Пресс, 2008. – 768 с.

24 ДьяконовВ.П. MATLAB 6.5 SPI/7 + Simulink 5/6. Основы применения. Серия «Библиотека профессионала». – М.: СОЛОН-Пресс, 2005. – 800 с.

Приложение А
Лабораторная работа № 1

Лабораторная работа №1. Кодирование свёрточных кодов

1.1 Цель лабораторной работы

Изучение и закрепление знания об основных параметрах сверточных кодов.

Изучение способов построения кодирующих устройств (кодеров) сверточных кодов в среде программируемых логических интегральных схем QUARTUS II.

1.2 Домашнее задание к лабораторной работе

1.2.1 Изучить кодирование сверточных кодов с использованием диаграммы состояний, древовидной диаграммы и решетчатой диаграммы.

1.2.2 Для простых порождающих полиномов сверточных кодов составить и начертить структурные схемы и исследовать их работу с использованием таблиц.

1.2.3 Изучить способы построения и свойства регистров сдвига, а также способы построения на них принципиальных электрических схем кодеров сверточных кодов в среде QUARTUS II.

1.3 Задание для выполнения лабораторной работы

1.3.1 Для заданных параметров сверточных кодов (таблица А.2) найти:

- порождающие многочлены в двоичном виде;
- скорость кода;
- коэффициент избыточности кода;
- полную длину кодового ограничения (память кода по выходу).

1.3.2 С использованием порождающих многочленов построить схему кодера, а с помощью таблицы закодировать входную последовательность.

1.3.3 Построить принципиальную электрическую схему сверточного кодера в среде QUARTUSII.

1.3.4 Получить временные диаграммы кодера в среде QUARTUSII и выписать символы сигналов, аналогичные символам в кодовой таблице.

1.4 Методическое указание для лабораторной работы

Методические указания содержат пример выполнения задания по лабораторной работе №1 при следующих значениях параметров: $m=3$, $k=1$, $n=2$, $g_1=5$, $g_2=7$, $a=15$.

Коэффициенты порождающих многочленов в двоичной форме:

$$g_1 = (1\ 0\ 1);$$
$$g_2 = (1\ 1\ 1).$$

Порождающие полиномы имеют вид:

$$g_1(x) = 1 + x + x^2;$$
$$g_2(x) = 1 + x^2.$$

Найдем скорость кода $R=k/n=1/2$.
Найдем избыточность кода:

$$\chi = 1 - R = 1 - 0,5 = 0,5$$

Теперь найдем для этого кодера полную длину l_2 кодового ограничения. Так как у нас $k=1$, $n=2$, $m=3$, то:

$$l_2 = \frac{n \cdot m}{k} = 3 \cdot 2 = 6.$$

По другой формуле, получим тот же результат:

$$l_2 = n \cdot \max[\deg g_{j,i}(x) + 1] = 2 \cdot (2 + 1) = 6,$$

где $g_{j,i}(x)$ – член порождающего полинома, *deg* – операция выделение степени.

Следовательно, шесть подряд следующих выходных символов кодера зависят от выбранного информационного символа.
Структурная схема кодера показана на рисунке А.1.

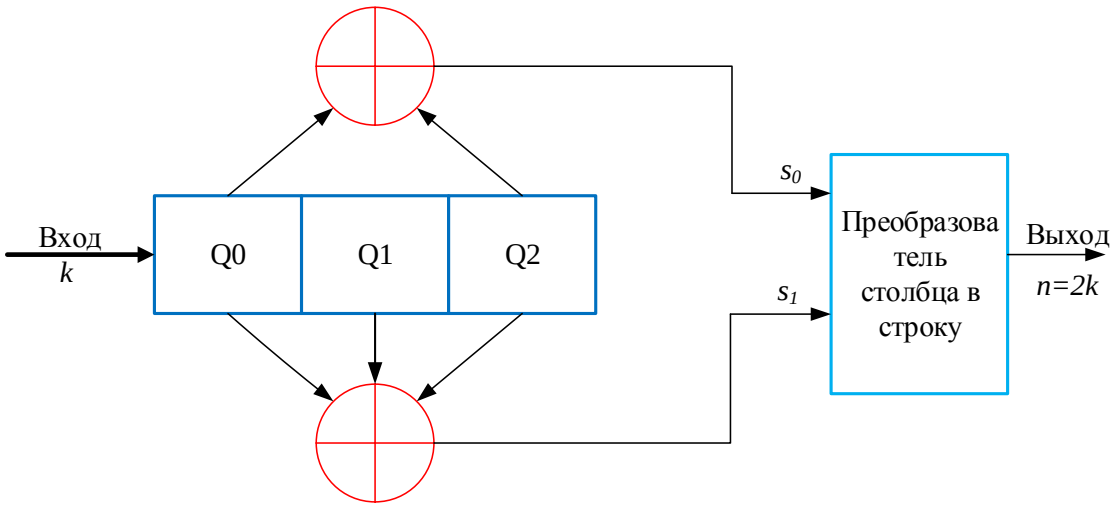


Рисунок А.1 – Структурная схема кодера сверточного кода с параметрами $m=3$, $k=1$, $n=2$

Здесь Q_0, Q_1, Q_2 – ячейки регистра сдвига и одновременно его выходные значения;

s_0, s_1– выходы сумматоров и одновременно их выходные значения.

В самом начале все ячейки регистра ставятся в нулевое состояние. Если предположить, что первый входной бит «1»,он без задержки отправляется на выход первой слева ячейки и, соответственно, на двух входах выходного мультиплексора.

Преобразователь поочерёдно выдаёт содержимое входов, и выходная последовательность из двух бит 11.

Второй входной бит возьмем равным «1». Он будет записан в 1-ую левую ячейку регистра, перемещая предыдущий бит («1») во вторую ячейку - и на входах мультиплексора (сверху вниз) появятся 10. Таким образом вторая выходная последовательность 10.

Если взять третий бит равным нулю, выходная последовательность будет 10 – и так далее.

Таким образом, свёрточный кодер кодирует двумя битамив ответ на каждый входной бит($k=1$) согласно числа сумматоров по модулю два($n=2$).

Рассмотрим на примере, как формируется входная кодовая последовательность для входного сигнала 110100 (таблица А.1).

Таблица А.1 – Формирование кодовой последовательности

Номера тактовых импульсов	0	1	2	3	4	5
Содержимое первого ячейки,Q_0	1	1	0	1	0	0
Содержимое второго ячейки,Q_1	0	1	1	0	1	0
Содержимое третьего ячейки,Q_2	0	0	1	1	0	1
Выходы первого сумматора,s_0	1	1	1	0	0	1
Выходы второго сумматора,s_1	1	0	0	0	1	1
Выход кодера,$\boldsymbol{s}$	11	10	10	00	01	11

На выходе кодера получим кодовая последовательность:

$\boldsymbol{s}$ = *(11 10 10 00 01 11).*

Построим сверточный кодер (рисунок А.1) в среде QUARTUS II (рисунок А.2).

Сверточный кодер на рисунке А.2 состоит из двух запоминающих ячеек (D-триггеров), элементов XOR (сумматоров по модулю 2) и преобразователя столбца в строку (мультиплексора).

Здесь отсутствует первая ячейка памяти Q_0. Это объясняется тем, что регистр сдвига можно рассматривать либо как регистр, содержимое которого сдвигается на один разряд вправо при введении в него слева каждого нового двоичного символа (содержимое самого правого разряда при этом теряется) (рисунок А.1), либо как цифровую линию задержки, в которой каждый элемент задержки хранит один двоичный символ до поступления нового

входного двоичного символа (рисунок А.2). Эти две схемы (рисунок А.1 и рисунок А.2) эквивалентны.

Мультиплексор состоит из: D-триггера и логические элементы (AND, NOT, OR)(рисунок А.3). На вход мультиплексора поступает параллельные символы (выходы два сумматора).

Мультиплексор во время первой половины такта кодера вначале считывает данные с логического элемента XOR 1, а во второй половине такта с логического элемента XOR 2.

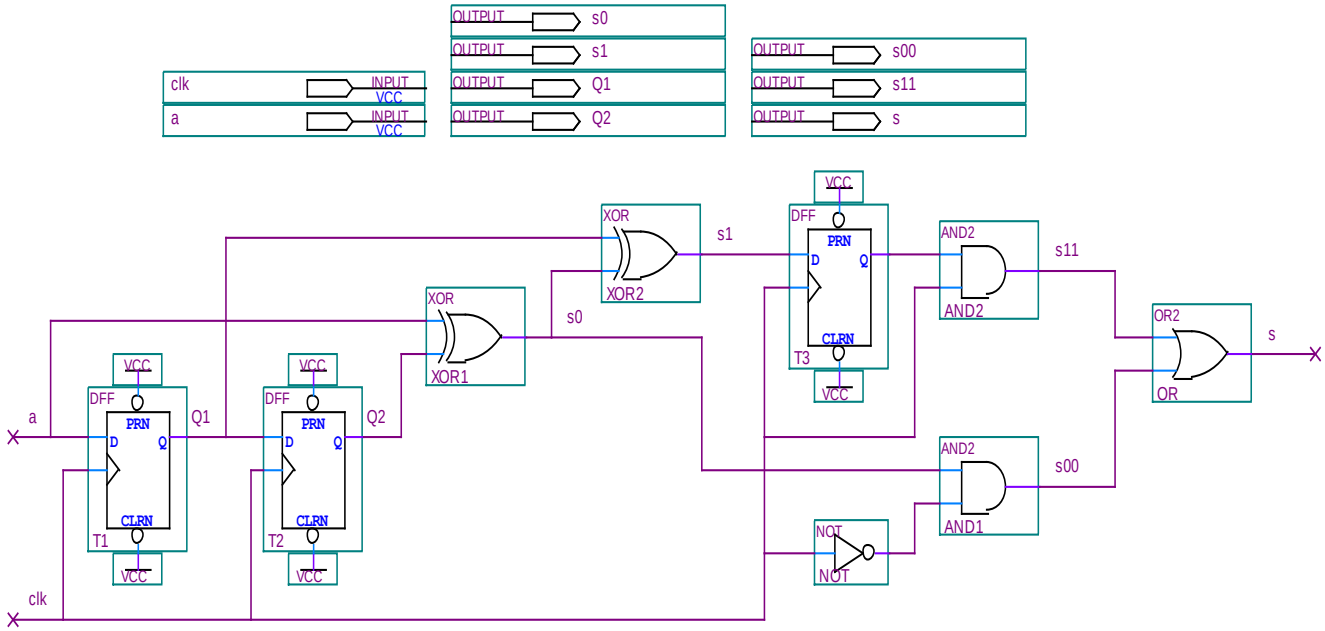


Рисунок А.2– Схема кодера в среде программируемых логических интегральных схем QUARTUS II

Выход первого сумматора поступает на первый вход логического элемента AND1, а на второй вход поступает инвертированный тактовый импульс (с помощью логического элемента NOT). Если на вход поступает 1+1, то на выходе AND1получим 1. В остальном случае 0.

Выход второго сумматора поступает на информационный вход Dтриггера, а на второй вход поступает тактовый импульс. Этот символ появляется на выходе триггера в момент первого фронта.

Далее он поступает на первый вход логического элемента AND2, а на второй вход поступает тактовый импульс. Если на вход поступает 1+1, то на выходе AND2получим 1. В остальном случае 0. Далее выходы AND1 и AND2 поступают на вход логического элемента OR. Если на вход поступают 0+0 на выходе получим 0. В остальном случае 1 (рисунок А.4). Так и мы получим кодовую последовательность на выходе мультиплексора.

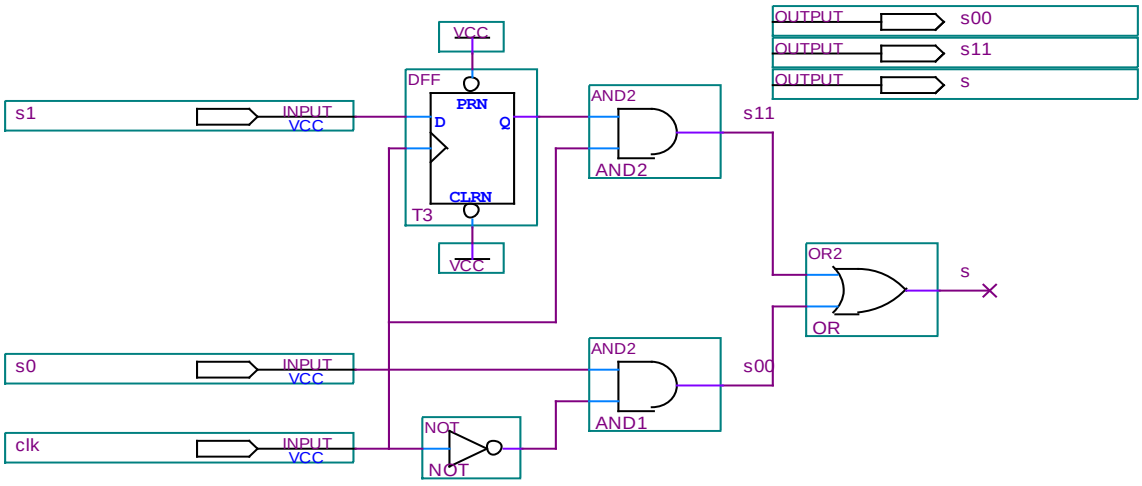


Рисунок А.3 – Мультиплексор в среде программируемых логических интегральных схем QUARTUS II

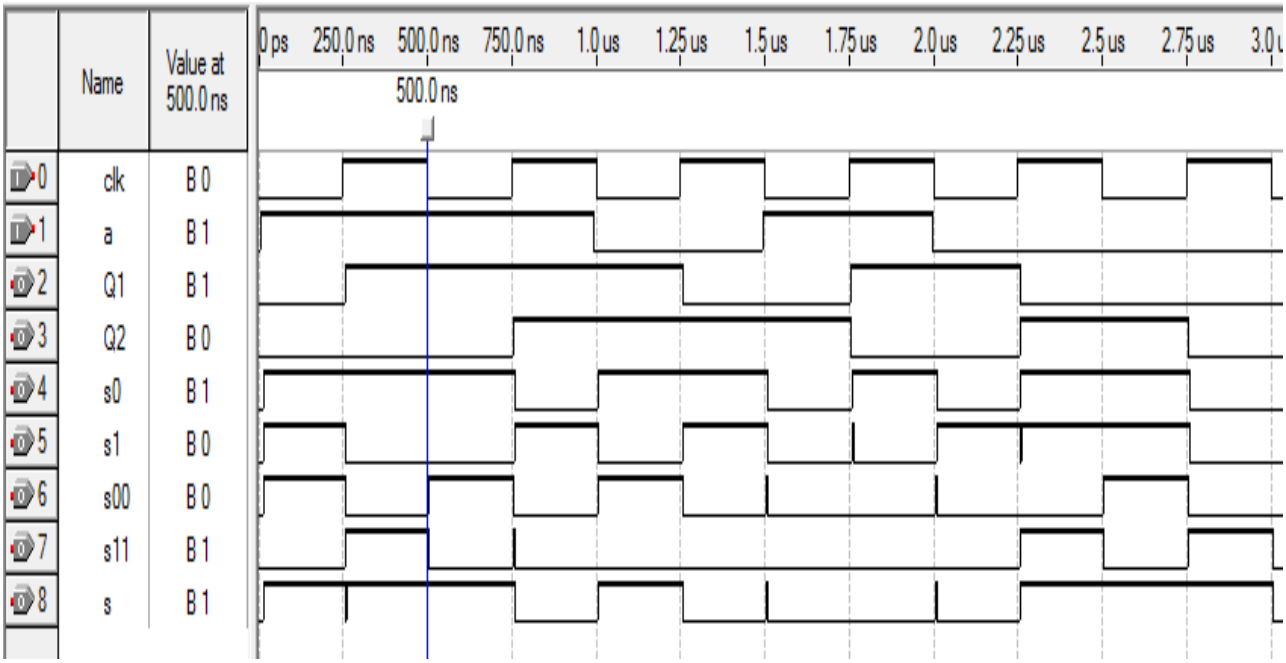


Рисунок А.4 – Временная диаграмма в среде программируемых логических интегральных схем QUARTUS II

По временной диаграмме (рисунок А.4) состояния кодера заметно, что при входной последовательности бит 110100, выходная последовательность будет – 11 01 01 00 10 11.

1.5 Требования к оформлению отчета по лабораторной работе №1

1. Отчет оформляется каждым студентом группы в отдельности.
2. Отчёт должен содержать:
- титульный лист;
- задание согласно варианту;

- краткую формулировку цели работы;
- структурную и принципиальную электрическую схему;
- результаты выполнения задания;
- результаты экспериментов в виде таблиц, диаграмм и графиков;
- краткие выводы по результатам исследований.

3. Защита работы проводится каждым студентом персонально.

1.6 Контрольные вопросы

1. Определение сверточного кода.
2. Определение скорости кода.
3. Назовите основные параметры сверточных кодов.
4. Какие существует способы сверточного кодирование?
5. В чем преимущества сверточного кодирование с использованием решетчатой диаграммы?
6. Пояснить, почему для сверточных кодов минимальное расстояние между принятым кодовым словом и каждым из путей, входящих в каждое состояние может быть вычислено как минимальный вес кодовых слов.
7. Приведите примеры кодеров систематического и несистематического сверточных кодов.
8. В чем отличие сверточных кодеров для скоростей 1/2, 1/3 и 2/3.
9. Дан сверточный код с параметрами: $m=4$ и со скоростью 1/2. Сколько различных кодовых слов последовательности, зависящих от выбранного информационного символа?

Таблица 2–Варианты задании параметров сверточных кодов

№	m	k	n	g_1	g_2	g_3	A
2	3	1	2	7	5	-	11
3	4	1	2	17	15	-	12
4	5	1	2	37	23	-	13
5	5	1	2	75	53	-	14
6	7	1	2	171	133	-	15
7	3	1	3	7	7	5	16
8	4	1	3	17	15	13	17
9	5	1	3	37	33	25	11
10	6	1	3	75	53	47	12
11	7	1	3	171	165	133	13
12	3	2	3	7	5	7	14
13	4	2	3	15	13	15	15
14	5	2	3	31	33	31	16
15	6	2	3	73	41	73	17

Приложение Б
Лабораторная работа № 2

Лабораторная работа №2.Декодирование свёрточных кодов

2.1 Цель лабораторной работы

Изучение способ декодирования сверточных кодов по Витерби.

Изучение и применение программных модулей кодирование-декодирование сверточных кодов по Витерби.

2.2 Задание для выполнения лабораторной работы

2.2.1 Построить решетчатую диаграмму сверточного кода с заданными параметрами.

2.2.2Кодировать и декодировать кодовую последовательность по алгоритму Витерби с помощью решетчатой диаграммы.

2.2.3Применяя среду MATLABи программу «viterbi», кодировать и декодировать кодовую последовательность.

2.3 Методическое указание для лабораторной работы

Методические указания содержат пример выполнения задания по лабораторной работе №2 при следующих значениях параметров: $m=4$, $k=1$, $n=3$, $g_1=17$, $g_2=15$,$g_3=13$,$a=15$.

Для построения решетчатой диаграммы надо знать все возможные состояния кодера. Число этих состояний равно $2^{m-1}=2^3=8$.Решетчатая диаграмма кодера показана на рисунке Б.1.

Пусть передается кодовое слово $\boldsymbol{s}$ = *(111 001 011 101 001 101 111)*.

Предположим, что в канале произошла однократная кодовая ошибка, так что принятая последовательность имеет вид $\boldsymbol{s}^*$ = *(111* ***1****01 011 101 001 101 111)*(ошибочный символ изображен жирной цифрой).

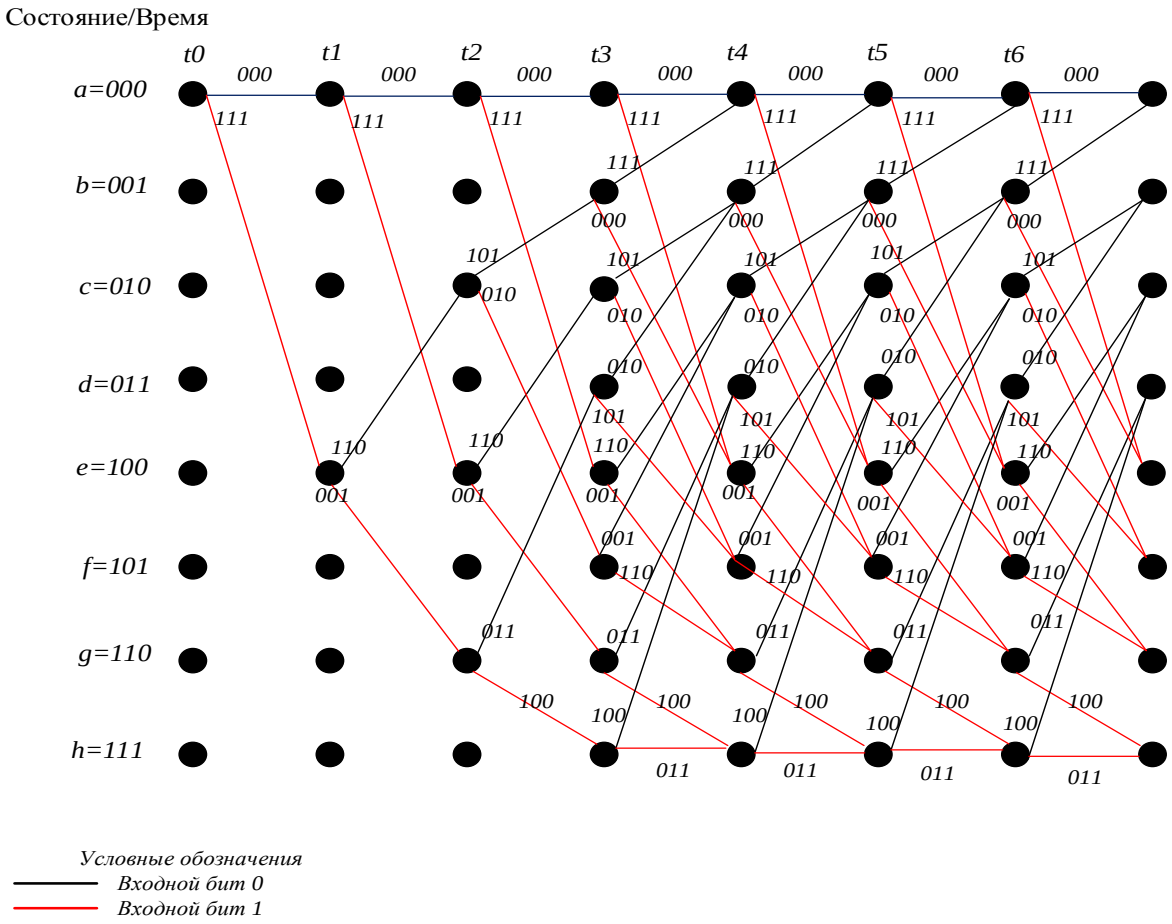
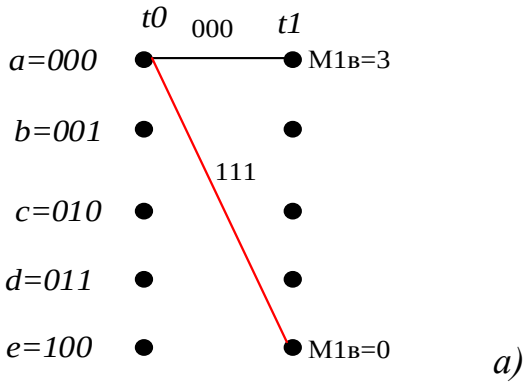
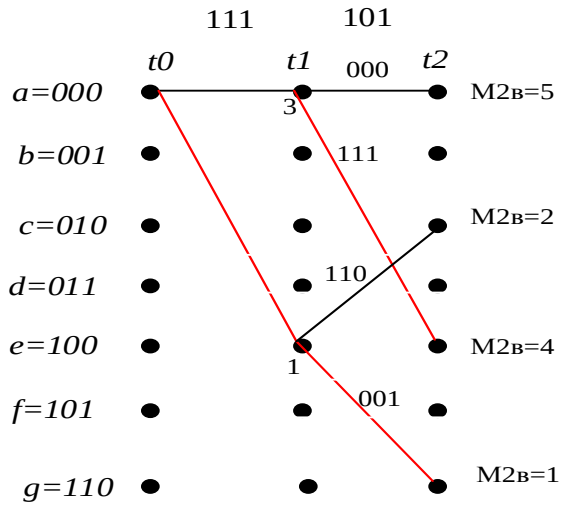


Рисунок Б.1 – Решетчатая диаграмма сверточного кода

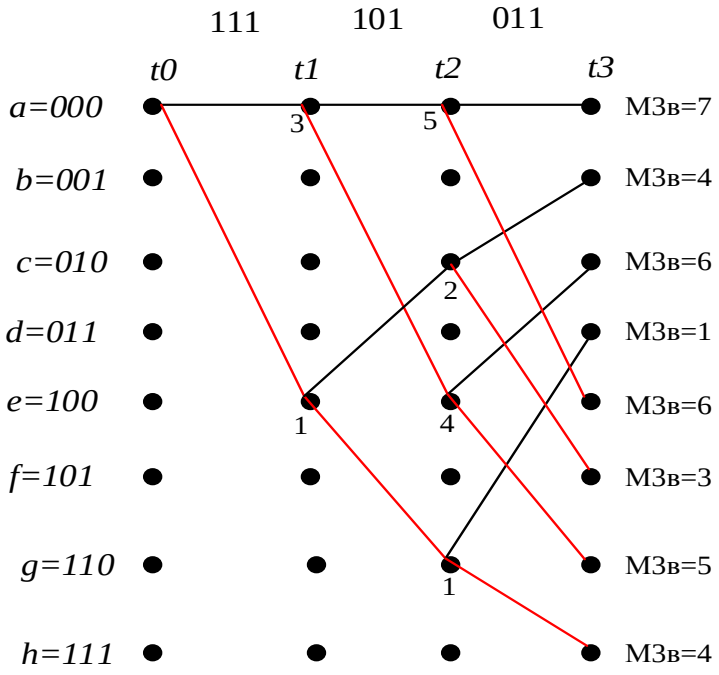
Для декодирования вычислим расстояние Хэмминга между принятой первой символьной группой (111) в $\boldsymbol{s}^*$ и всевозможными ветвями (t_0,t_1) решетчатой диаграммы (рисунок Б.2, а).



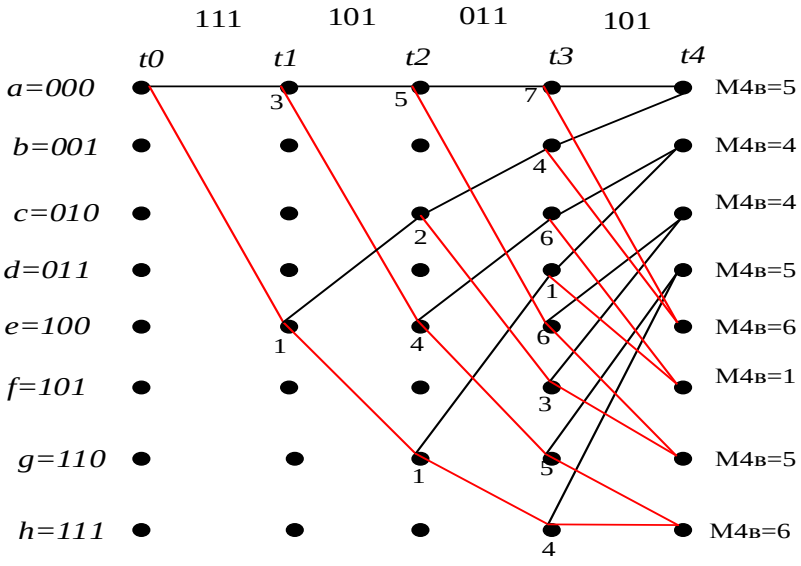
а)


111
101
t0
t1
000
t2
a=000
M2в=5
3
b=001
111
c=010
M2в=2
d=011
110
e=100
M2в=4
1
f=101
001
g=110
M2в=1

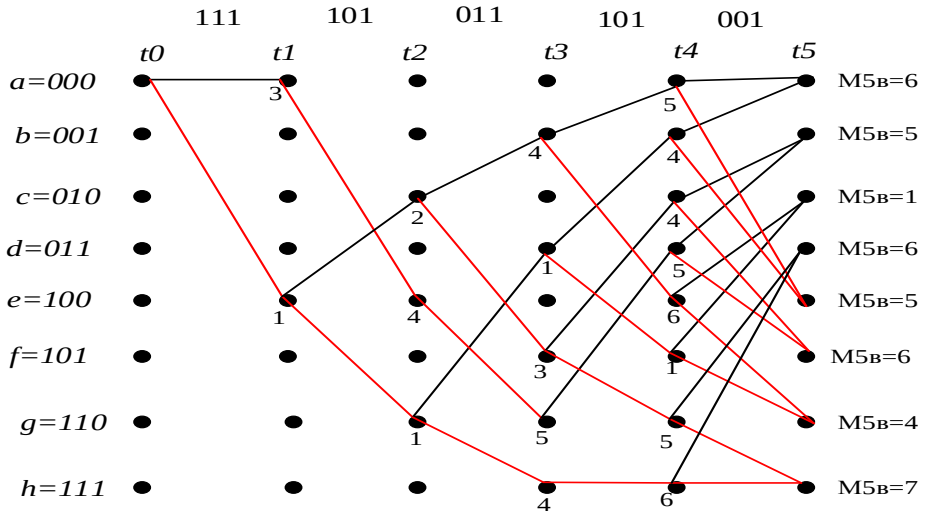
б)


111
101
011
t0
t1
t2
t3
a=000
M3в=7
3
5
b=001
M3в=4
c=010
M3в=6
2
d=011
M3в=1
e=100
M3в=6
1
4
f=101
M3в=3
g=110
M3в=5
1
h=111
M3в=4

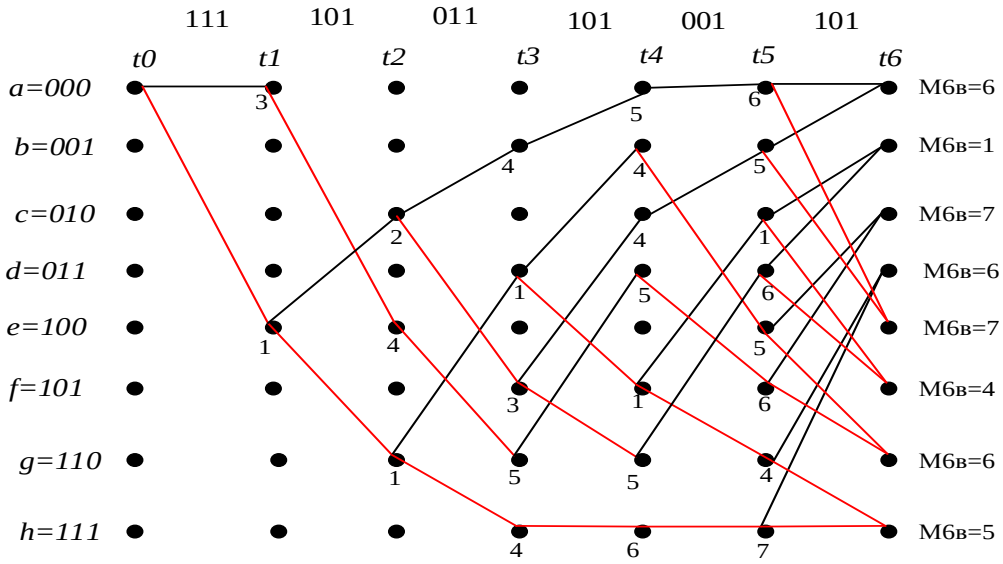
в)


111
101
011
101
t0
t1
t2
t3
t4
a=000
M4в=5
3
5
7
b=001
M4в=4
4
c=010
M4в=4
2
6
d=011
M4в=5
1
e=100
M4в=6
1
4
6
f=101
M4в=1
3
g=110
M4в=5
1
5
h=111
M4в=6
4

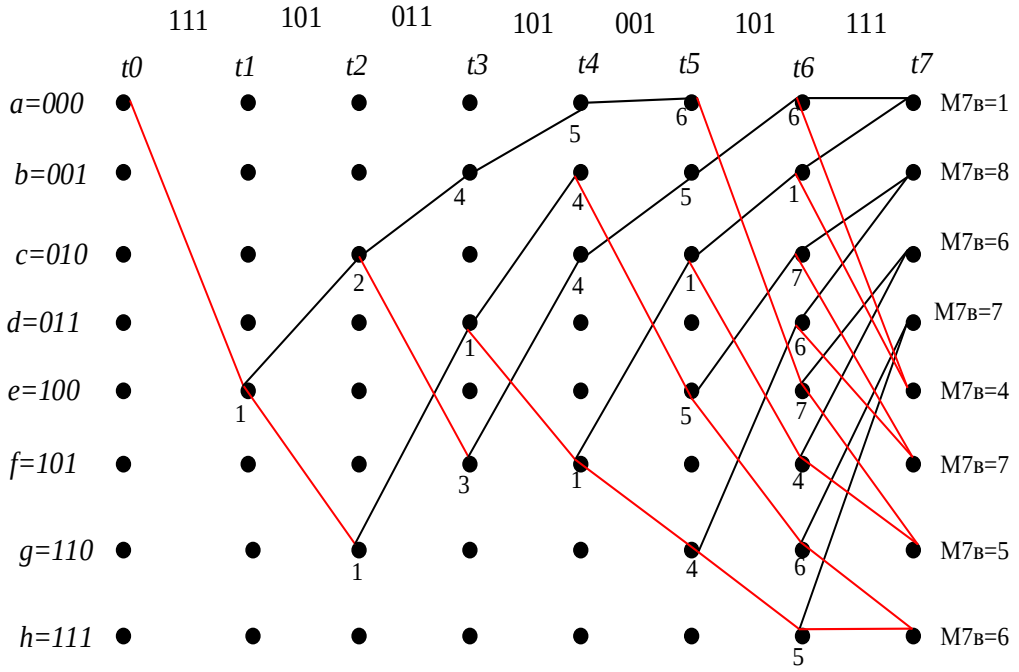
г)


111
101
011
101
001
t0
t1
t2
t3
t4
t5
a=000
b=001
c=010
d=011
e=100
f=101
g=110
h=111
М5в=6
М5в=5
М5в=1
М5в=6
М5в=5
М5в=6
М5в=4
М5в=7

д)


111
101
011
101
001
101
t0
t1
t2
t3
t4
t5
t6
a=000
b=001
c=010
d=011
e=100
f=101
g=110
h=111
М6в=6
М6в=1
М6в=7
М6в=6
М6в=7
М6в=4
М6в=6
М6в=5

е)


111
101
011
101
001
101
111
t0
t1
t2
t3
t4
t5
t6
t7
a=000
b=001
c=010
d=011
e=100
f=101
g=110
h=111
М7в=1
М7в=8
М7в=6
М7в=7
М7в=4
М7в=7
М7в=5
М7в=6

ж)

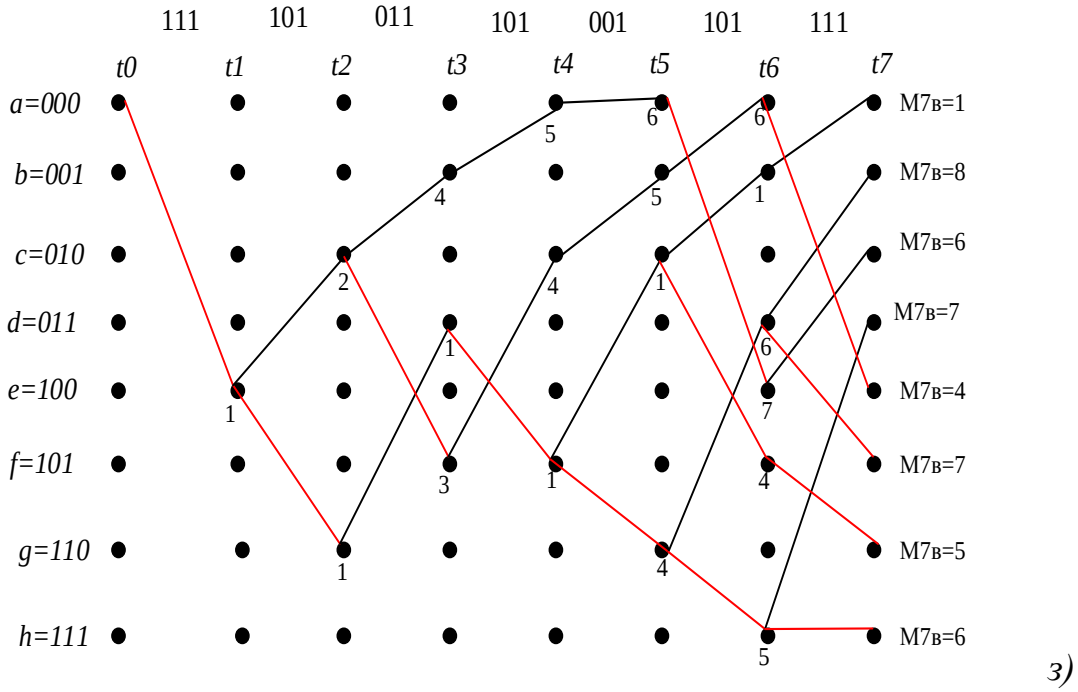


Рисунок Б.2 – Решетчатая диаграмма декодера (степень кодирования *1/3,m=4*)

На первом шаге из состояния 000 выходят два пути: в состояние 000 с расстоянием Хэмминга до принятой из канала первой символьной последовательности, равным 3 и в состояние 100 с метрикой 0. На втором шаге формируются уже четыре пути, причем метрики состояний 000, 010, 100 и 110 становятся равными 5, 2, 4 и 1 соответственно (рис. А6, б). На третьем шаге формируются уже восемь пути, причем метрики состояний 000, 001, 010, 011, 100, 101, 110 и 111 становятся равными 7, 4, 6, 1, 6, 3, 4 и 5 соответственно (рисунок Б.2,в).

Четвертый шаг работы является ключевым для понимания всего алгоритма. Рассмотрим состояние 000. В данное состояние можно прийти по двум различным путям: из состояния 000 и из состояния 001.

Первый из них будет иметь метрику 9, поскольку метрика ветви (000->000) равна двум и предыдущее значение метрики пути равно 7, а второй - 5, так как метрика ветви (001 —>000) равна 1 и предыдущая метрика пути равна 4.

Сравнение метрик этих двух путей позволяет выбрать выживший путь (путь из состояния 001, так как он имеет меньшую метрику) (рис. А6, г). Таким же образом выбираются выжившие пути для остальных узлов решетки.

Шаги 5 и 6 полностью аналогичны четвертому шагу (рисунок Б.2, д, е).

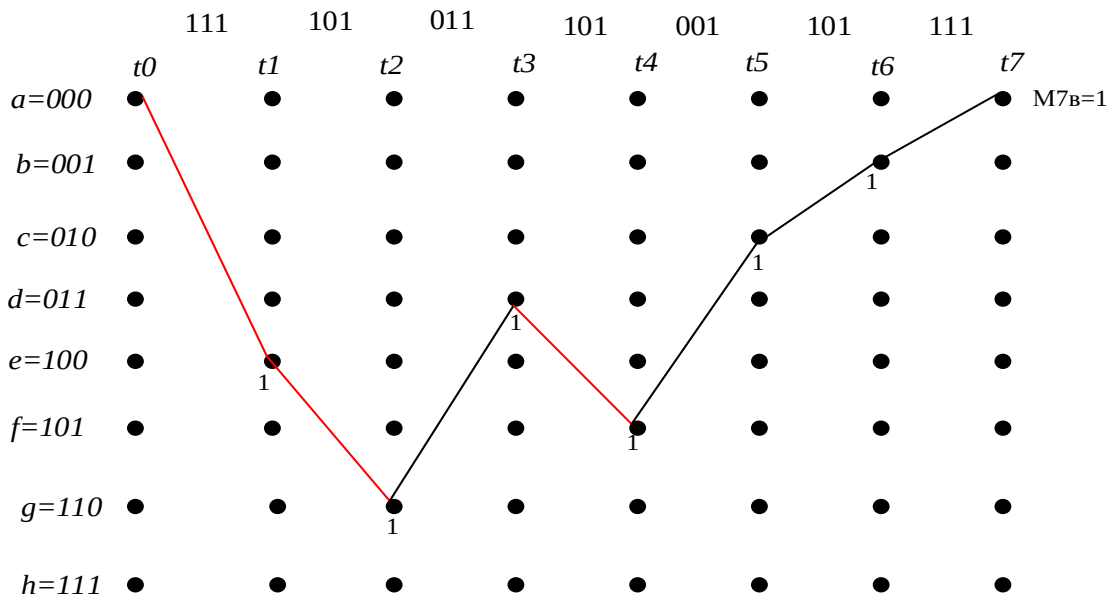


Рисунок Б.3 – Декодированная последовательность

После выполнения 7 шага работы сложилась характерная ситуация, когда начальные части (первые 6 ветвей) всех выживших путей слились и теперь можно принимать решение о первых 6 информационных битах (рисунок Б.2, ж).

Начальная часть выживших путей определяет информационную последовательность 1101000, т.е. один канальная ошибка была исправлена (рисунок Б.3).

2.4 Требования к оформлению отчета

1. Отчет оформляется каждым студентом группы в отдельности.
2. Отчёт должен содержать:
- титульный лист;
- задание согласно варианту;
- краткую формулировку цели работы;
- результаты выполнения задания;
- результаты экспериментов в виде таблиц, диаграмм и графиков;
- листинги программы;
- краткие выводы по результатам исследований.
3. Защита работы проводится каждым студентом персонально.

2.5 Контрольные вопросы

1. Как влияет длина полную кодового ограничения на работу декодера?
2. Какие алгоритмы декодирования сверточных кодов вы знаете?
3. Какова достоинства алгоритма Витерби? Какова недостатки алгоритма Витерби?
4. От каких условий зависит исправляющая способность сверточных кодов.

Приложение В

Моделирование сверточного кодера со скоростью 1/2 в среде MATLAB:

```
% Сверточное кодирование; k=1 бит -> n=2 биты; память кода m=3, Скорость
кода R=1/2
% Файл-функция
function [encoded_sequence]=convlenc(message)
enco_mem=[0 0 0]; % Память кода или число ячеек регистра сдвига m=3,
начальное состояние
encoded_sequence=zeros(1,(length(message))*2); % Выходноекодовоеслово
% на вход поступает первый информационный символ
enco_mem(1,3)=enco_mem(1,2); % содержимое третьей ячейки регистра сдвига
enco_mem(1,2)=enco_mem(1,1); % содержимое второй ячейки регистра сдвига
enco_mem(1,1)=message(1,1);  % содержимое первой ячейки регистра сдвига
% Вычисление выходы сумматоров
temp=xor(enco_mem(1),enco_mem(2));
o1=xor(temp,enco_mem(3));        % выход первого сумматора, коэффициенты
порождающего полинома 111
o2=xor(enco_mem(1),enco_mem(3)); % выход второго сумматора, коэффициенты
порождающего полинома 101
% выходное кодовое слово соответствующих первому информационному символу
encoded_sequence(1,1)=o1;
encoded_sequence(1,2)=o2;
% на вход поступает второй информационный символ
msg_len=length(message);
c=3;
for i=2:msg_len
enco_mem(1,3)=enco_mem(1,2); % содержимоетретьейячейкирегистрасдвига
enco_mem(1,2)=enco_mem(1,1); % содержимое второй ячейки регистра сдвига
if(i<=msg_len)
    enco_mem(1,1)=message(1,i); % если i <= длины информационных
символов, тогда содержимое первой ячейки регистра сдвига равно i
else
    enco_mem(1,1)=0; % противном случае содержимое первой ячейки
регистра сдвига равно 0
end
       temp=xor(enco_mem(1),enco_mem(2));
o1=xor(temp,enco_mem(3));          % выход первого сумматора,
коэффициенты порождающего полинома 111
       o2=xor(enco_mem(1),enco_mem(3));   % выход второго сумматора,
коэффициенты порождающего полинома 101
% выходное кодовое слово соответствующих i-му информационному символу
encoded_sequence(1,c)=o1;
       c=c+1;
       encoded_sequence(1,c)=o2;
       c=c+1;
end
```

Моделирование сверточного декодера по алгоритму Витерби в среде MATLAB:

```
%% Деокдирование по алгоритму Витерби
% Канал: AWGN, Модуляция: BPSK
%% Исходные данные
SNRdB=1:1:6;                    % Отношение сигнал/шум, дБ
SNR=10.^(SNRdB/10);             % Отношение сигнал/шум, в разах
ber=zeros(1,length(SNR));       % Вероятность битовой ошибки
bl=1E6;                         % Длина информационных символов
N=1E3;                          % Длина блока
```

```
      info_bits=floor(2*rand(1,bl)); % Информационные символы (0 или 1) длиной
bl
      for i=N-1:N:bl % Последний и предпоследний биты блока длиной N равно 0.
Очистка регистра сдвига.
      info_bits(i)=0;info_bits(i+1)=0;
      end
      %% Сверточное кодирование. Используем файл-функции convlenc
      code=convlenc(info_bits);
      code_bit0=[];
      code_bit1=[];
      n=1;
      for m=1:2:length(code)
          code_bit0(n)=code(m);
          n=n+1;
      end
      r=1;
      for u=2:2:length(code)
          code_bit1(r)=code(u);
          r=r+1;
      end
      %% BPSK модулятор
      mod_code_bit0=2*code_bit0(1:bl)-1;
      mod_code_bit1=2*code_bit1(1:bl)-1;
      fprintf('Encoding completed...\n');
      %% Каналсшумом AWGN
      for k=1:length(SNR)
      % Добавлениешума
          y0=(sqrt(SNR(k))*mod_code_bit0)+randn(1,bl);
          y1=(sqrt(SNR(k))*mod_code_bit1)+randn(1,bl);
      % Демодулятор
          y00=qamdemod(y0,2);
          y11=qamdemod(y1,2);
      % Сверточное декодирование
      for p=1:N:bl
      % Решетчатая диаграмма декодера
              distance=zeros(4,N+1);      % Состояние кодера 4,число узлов N+1
              metric=zeros(8,N);          % Количество метрик
              distance(1,1)=0;            % Первое состояние
              distance(2,1)=Inf;          % Второе состояние
              distance(3,1)=Inf;          % Третье состояние
      distance(4,1)=Inf;          % Четвертоесостояние
      for i=1:N
                  metric(1,i)=norm([0,0]-[y00(i+p-1),y11(i+p-1)]);            %
Метрикаветвиот 00 к 00
                  metric(2,i)=norm([1,1]-[y00(i+p-1),y11(i+p-1)]);            %
Метрикаветвиот 00 к 10
                  metric(3,i)=norm([1,0]-[y00(i+p-1),y11(i+p-1)]);            %
Метрикаветвиот 10 к 01
                  metric(4,i)=norm([0,1]-[y00(i+p-1),y11(i+p-1)]);            %
Метрикаветвиот 10 к 11
                  metric(5,i)=norm([1,1]-[y00(i+p-1),y11(i+p-1)]);            %
Метрикаветвиот 01 к 00
                  metric(6,i)=norm([0,0]-[y00(i+p-1),y11(i+p-1)]);            %
Метрикаветвиот 01 к 10
                  metric(7,i)=norm([0,1]-[y00(i+p-1),y11(i+p-1)]);            %
Метрикаветвиот 11 к 01
                  metric(8,i)=norm([1,0]-[y00(i+p-1),y11(i+p-1)]);            %
Метрикаветвиот 11 к 11
      distance(1,i+1)=min(distance(1,i)+metric(1,i),distance(3,i)+metric(5,i))
;          %          Минимальнаяметрикав          00          вовремя          i
distance(2,i+1)=min(distance(1,i)+metric(2,i),distance(3,i)+metric(6,i));     %
Минимальнаяметрикав                10                вовремя                i
```

```
distance(3,i+1)=min(distance(2,i)+metric(3,i),distance(4,i)+metric(7,i));     %
Минимальнаяметрикав                  01                  вовремя                i
distance(4,i+1)=min(distance(2,i)+metric(4,i),distance(4,i)+metric(8,i));     %
Минимальнаяметрикав 11 вовремя i
      end
              [~,state]=min(distance(:,N+1));   %   Выборминимальнуюметрику   в
последнем узле
      for                                                                  j=N:-
1:1[state,decoded_bit]=prev_stage(state,distance(:,j),metric(:,j));
      decoded_bit_final(j+p-1)=decoded_bit;          %          Декодированная
последовательность
      end
      end
      % Вычисление вероятность битовой ошибки
      ber(k) = (sum ( abs( decoded_bit_final-info_bits ) ) )/bl;
          string   =   [num2str(round(k/length(SNR)   *   100))   ,   '%   decoding
completed ...'];
      disp(string);
      end
      % Зависимость вероятность битовой ошибки от отношение сигнал/шум
      semilogy(SNRdB,ber,'b-d','linewidth',2.0)
      axis tight;grid;
      legend('m=3, R=1/2');
      xlabel('С/Ш(dB)');ylabel('Вероятностьбитовойошибки');
      %Принцип работы файл-функции prev_stage
      function
[prev_state,decoded_bit]=prev_stage(curr_state,distance_prev,metric)
      if(curr_state==1)
      if(distance_prev(1)+metric(1) <= distance_prev(3)+metric(5))
                  prev_state=1;decoded_bit=0;
      else
                  prev_state=3;decoded_bit=0;
      end
      end
      if(curr_state==2)
      if(distance_prev(1)+metric(2) <= distance_prev(3)+metric(6))
                  prev_state=1;decoded_bit=1;
      else
                  prev_state=3;decoded_bit=1;
      end
      end
      if(curr_state==3)
      if(distance_prev(2)+metric(3) <= distance_prev(4)+metric(7))
                  prev_state=2;decoded_bit=0;
      else
                  prev_state=4;decoded_bit=0;
      end
      end
      if(curr_state==4)
      if(distance_prev(2)+metric(4) <= distance_prev(4)+metric(8))
                  prev_state=2;decoded_bit=1;
      else
                  prev_state=4;decoded_bit=1;
      end
      end
      end
```