Лабораторная работа №2

«Логическое кодирование»

Введение

При цифровом кодировании дискретной информации применяют потенциальные и импульсные коды. В потенциальных кодах для представления логических единиц и нулей используется только значение потенциала сигнала, а его перепады во внимание не принимаются. Импульсные коды позволяют представить двоичные данные либо импульсами определённой полярности, либо частью импульса – перепадом потенциала определённого направления [1].

Для того чтобы приёмник точно знал, в какой момент времени считывать новую порцию информации с линии связи нужна синхронизация передатчика и приёмника. При передаче дискретной информации время всегда разбивается на такты одинаковой длительности и приёмник старается считать новый сигнал в середине каждого такта, то есть синхронизировать свои действия с передатчиком [2].

Проблема синхронизации в сетях решается сложнее, чем при обмене данными между близко расположенными устройствами, например между блоками внутри компьютера. На больших расстояниях неравномерность скорости распространения сигнала может привести к тому, что тактовый импульс придёт настолько позже или раньше соответствующего сигнала данных, что бит данных будет пропущен или считан повторно [2].

Распознавание и коррекцию искажённых данных сложно осуществить средствами физического уровня, поэтому чаще всего эту работу берут на себя вышележащие протоколы: канальный, сетевой, транспортный или прикладной. В то же время распознавание ошибок на физическом уровне экономит время, так как приёмник не ждёт полного помещения кадра в буфер, а отбраковывает его сразу при распознавании ошибочных битов внутри кадра [2].

Избыточные коды

При избыточном кодировании исходный двоичный код представляется в виде последовательностей нескольких битов – символами (порциями), каждая из которых заменяется новой последовательностью, содержащей большее количество бит, чем исходная [2,3].

К методам избыточного кодирования относятся: 4B/5B, 5B/6B, 8B/10B, 64B/66B [3].

Буква B в названии данных методов означает, что элементарный сигнал имеет два состояния (от английского binary – двоичный). Например, метод 4B/5B означает, что каждые 4 бита в исходном коде заменяются 5ю битами в результирующем коде. Имеются также коды и с тремя состояниями сигнала, например, в коде 8B/6T для кодирования 8 бит исходной информации используется код из 6 сигналов, каждый из которых имеет три

состояния. Избыточность кода 8B/6T выше, чем кода 4B/5B, так как на 256 исходных кодов приходится $3^6 = 729$ результирующих символов [2,3].

Логический код 4B/5B

Код 4B/5B, используется в технологии Fast Ethernet. Исходные символы длиной 4 бита заменяются символами длиной 5 бит. Поскольку результирующие символы содержат избыточные биты, общее количество битовых комбинаций в них больше, чем в исходных. В коде 4B/5B результирующие символы могут содержать 32 битовые комбинации (2^5=32), в то время как исходные символы – только 16 (2^4=16) (таблица 1). Таким образом, количество избыточных (запрещённых) кодов: 32-16=16. Появление запрещённых символов означает ошибку в передаваемых данных. Помимо устранения постоянной составляющей и придания коду свойства самосинхронизации, избыточные коды позволяют приёмнику распознавать искажённые биты. Если приёмник принимает запрещённый код, то это означает, что на линии произошло искажение сигнала [2,3].

После разбиения, получившийся код 4B/5B передаётся по линии путём преобразования с помощью какого-либо из методов потенциального кодирования, чувствительного только к длинным последовательностям нулей, например, NRZI. Символы кода 4B/5B длиной 5 бит гарантируют, что при любом их сочетании на линии не встретятся более трёх нулей подряд [2].

Таблица 1 – Соответствие исходных и результирующих кодов 4B/5B

Исходные символы	**Результирующий код**	**Исходные символы**	**Результирующий код**
0000	11110	1000	10010
0001	01001	1001	10011
0010	10100	1010	10110
0011	10101	1011	10111
0100	01010	1100	11010
0101	01011	1101	11011
0110	01110	1110	11100
0111	01111	1111	11101

В скоростных версиях 10G Ethernet и 100G Ethernet применяется избыточный код 64B/66B [2].

Можно отметить следующие достоинства избыточного кодирования:

- появляется свойство самосинхронизации, поскольку исчезают длинные последовательности нулей и единиц;
- сужается спектр сигнала в связи с отсутствием постоянной составляющей;
- появляется возможность обнаружения ошибок за счёт наличия запрещённых символов;
- простая реализация в виде таблицы перекодировки [3].

Недостатки избыточного кодирования:

- уменьшается полезная пропускная способность канала связи, так как часть пропускной способности тратится на передачу избыточных бит;
- возникают дополнительные временные затраты в узлах сети на реализацию логического кодирования [3].

Основным недостатком избыточного кодирования является появление «лишнего» бита, приходящегося на 4 информационных бита, т. е. избыточность кода 4B/5B составляет 25% (¼ = 0,25). Это означает, что реальная пропускная способность канала будет меньше номинальной на 20%. Для сохранения заданной пропускной способности необходимо увеличить тактовую частоту передатчика на 25%, что, в свою очередь, приведёт к увеличению спектра сигнала. Так, для передачи кодов 4B/5B со скоростью 100 Мбит/с требуется тактовая частота 125 МГц. При этом спектр сигнала на линии расширяется по сравнению со случаем, когда по линии передаётся не избыточный код. Тем не менее спектр избыточного потенциального кода оказывается уже спектра манчестерского кода, что оправдывает дополнительный этап логического кодирования, а также работу приёмника и передатчика на повышенной тактовой частоте [2,3].

В методе логического кодирования 8B/6T для кодирования 8 бит (B) исходного сообщения используется код из 6 троичных (T) символов с тремя состояниями сигнала. Количество избыточных (запрещённых) кодов: $3^6 – 2^8 = 729 – 256 = 473$. Таким образом, в 8B/6T доля запрещённых кодов больше, чем в 4B/5B (65% против 50%), что повышает эффективность обнаружения ошибок [3].

Потенциальный код NRZ обладает хорошим спектром с одним недостатком – у него имеется постоянная составляющая. Коды, полученные из потенциального путём логического кодирования, обладают более узким спектром, чем манчестерский, даже при повышенной тактовой частоте. Этим объясняется применение потенциальных избыточных и скремблированных кодов в современных технологиях, подобных FDDI, Fast Ethernet, Gigabit Ethernet, ISDN и т. п. вместо манчестерского и биполярного импульсного кодирования [3].

Избавиться от длинных последовательностей нулей в коде помогает такой приём, как скремблирование – «перемешивание» битов кода в соответствии с определенным алгоритмом, позволяющим приёмнику выполнить обратное преобразование [3].

Скремблирование

Скремблирование – преобразование исходных и полученных в предыдущих тактах битов по заданному алгоритму, позволяющему исключить или, по крайней мере уменьшить длинные последовательности нулей или единиц [3,4].

Например, скрэмблер может реализовывать следующее соотношение (1) [4].

$$B_i = A_i \oplus B_{i-3} \oplus B_{i-5} \quad (1)$$

Здесь B_i – двоичная цифра результирующего кода, полученная на *i-м* такте работы скрэмблера, A_i – двоичная цифра исходного кода, поступающая на *i-м* такте на вход скрэмблера, B_{i-3} и B_{i-5} — двоичные цифры результирующего кода, полученные на предыдущих тактах работы скрэмблера (соответственно на 3 и на 5 тактов ранее текущего такта) и объединённые операцией исключающего ИЛИ (сложение по модулю 2) [4].

Например, для исходной последовательности 110110000001 скрэмблер даст следующий результирующий код (первые три цифры результирующего кода будут совпадать с исходным кодом, так как ещё нет нужных предыдущих цифр):

$B_1 = A_1 = 1$

$B_2 = A_2 = 1$

$B_3 = A_3 = 0$

$B_4 = A_4 \oplus B_1 = 1\ 1 = 0$

$B_5 = A_5 \oplus B_2 = 1\ 1 = 0$

$B_6 = A_6 \oplus B_3 \oplus B_1 = 0\ 0\ 1 = 1$

$B_7 = A_7 \oplus B_4 \oplus B_2 = 0\ 0\ 1 = 1$

$B_8 = A_8 \oplus B_5 \oplus B_3 = 0\ 0\ 0 = 0$

$B_9 = A_9 \oplus B_6 \oplus B_4 = 0\ 1\ 0 = 1$

$B_{10} = A_{10} \oplus B_7 \oplus B_5 = 0\ 1\ 0 = 1$

$B_{11} = A_{11} \oplus B_8 \oplus B_6 = 0\ 0\ 1 = 1$

$B_{12} = A_{12} \oplus B_9 \oplus B_7 = 1\ 1\ 1 = 1$

Таким образом, на выходе скрэмблера появится код 110001101111, в котором нет последовательности из шести нулей, присутствовавшей в исходном коде [4].

После получения результирующей последовательности приёмник передаёт её дескрэмблеру, который восстанавливает исходную последовательность на основании обратного соотношения (2) [4].

$$C_i = B_i \oplus B_{i-3} \oplus B_{i-5} = A_i \qquad (2)$$

Различные алгоритмы скрэмблирования отличаются количеством слагаемых, дающих цифру результирующего кода, и сдвигом между слагаемыми. Так, в сетях ISDN при передаче данных от сети к абоненту используется преобразование со сдвигами на 5 и 23 позиции, а при передаче данных от абонента в сеть – со сдвигами на 18 и 23 позиции.

Основным достоинством скремблирования по сравнению с избыточным кодированием является сохранение полезной пропускной способности канала связи, поскольку отсутствуют избыточные биты [3].

Недостатками скремблирования следует считать:

- наличие дополнительных затрат (накладных расходов) в узлах сети на реализацию алгоритма скремблирования-дескремблирования;
- отсутствие 100% гарантии исключения длинных последовательности нулей и единиц, а также возможность появления других (новых) последовательности нулей и единиц в результирующем коде [3].

Лабораторная работа

Цель: изучение методов логического кодирования информации.

Задачи лабораторной работы:

- аналитический обзор методов логического кодирования информации;
- практическое изучение логического кодирования двоичной информации методом 4B/5B и скремблированием;
- закрепление полученных в ходе выполнения лабораторной работы знаний, умений и навыков.

Варианты индивидуальных заданий

Для закрепления материала по теме лабораторной работы, каждому обучающемуся необходимо применить к исходному сообщению метод 4B/5B и скремблирование. Затем, рассчитать контрольную сумму полученных данных методом контроля чётности (паритет).

В качестве исходного сообщения, подлежащего передаче, используется фамилия обучающегося выполняющего задание. Для перевода фамилии в численное представление используется таблица ASCII (с учётом регистра, таблицы 4 и 5).

Получившееся для передачи сообщение необходимо представить в шестнадцатеричном и двоичном кодах, например (таблица 2):

Таблица 2 – Пример создания исходного сообщения

Фамилия в виде текста:	Иванов
Фамилия в виде шестнадцатеричного кода:	88 A2 A0 AD AE A2
Фамилия в виде двоичного кода:	10001000 10100010 10100000 10101101 10101110 10100010

Далее, над полученным сообщением выполнятся логическое кодирование по методу 4B/5B (раздел «Логический код 4B/5B», таблица 1). Полученное сообщение необходимо записать в двоичном коде.

Следующим этапом, является скремблирование исходного сообщения полиномом, представленным в таблице 3 (при нехватке вариантов, начинать счёт сначала). В ответ необходимо записать полученное скремблированное сообщения в двоичном кодае.

Последним этапом выполнения лабораторной работы, является расчёт контрольной суммы методом контроля чётности (паритет) для каждого символа, полученного кодированием 4B/5B (для каждых 10 бит) и скремблированием (для каждых 8 бит).

Таблица 3 – Варианты заданий

Номер варианта	**Полиномы**
1	$B_i = A_i \oplus B_{i-3} \oplus B_{i-5}$
2	$B_i = A_i \oplus B_{i-5} \oplus B_{i-7}$
3	$B_i = A_i \oplus B_{i-2} \oplus B_{i-5}$
4	$B_i = A_i \oplus B_{i-3} \oplus B_{i-7}$
5	$B_i = A_i \oplus B_{i-5} \oplus B_{i-13}$
6	$B_i = A_i \oplus B_{i-9} \oplus B_{i-15}$
7	$B_i = A_i \oplus B_{i-3} \oplus B_{i-9}$
8	$B_i = A_i \oplus B_{i-3} \oplus B_{i-7}$
9	$B_i = A_i \oplus B_{i-7} \oplus B_{i-15}$
10	$B_i = A_i \oplus B_{i-9} \oplus B_{i-13}$

Состав отчёта по лабораторной работе

В отчёт по лабораторной работе должно обязательно входить следующее:

1. титульный лист (название министерства и ВУЗа, направление и направленности подготовки, номер курса, номер группы, название дисциплины, номер лабораторной работы, ФИО обучающегося и преподавателя, город и год);
2. цели и задачи лабораторной работы;
3. ход выполнения лабораторной работы;
4. выводы по лабораторной работе (отразить достижение целей и задач лабораторной работы, в выводах необходимо использовать ключевые слова: изучено, проанализировано, пришли к заключению и т.д.).

Использованная литература

1 Ситанов, С.В. Компьютерные сети : учебное пособие / С. В. Ситанов, С. С. Алаева; Минобрнауки России, Ивановский государственный химико-технологический университет, Кафедра информационных технологий. – Иваново : ИГХТУ, 2010. – 134 с. – URL: http://dit.isuct.ru/IVT/sitanov/Literatura/KompSeti.html.

2 Олифер, В. Компьютерные сети. Принципы, технологии, протоколы : Юбилейное издание / В. Олифер, Н. Олифер. – 6-е изд. – Санкт-Петербург : Питер, 2020. – 1008 с. – ISBN 978-5-4461-1426-9.

3 Алиев, Т.И. Компьютерные сети и телекоммуникации: задания и тесты : учебно-методическое пособие / Т. И. Алиев, В. В. Соснин, Д. Н. Шинкарук ; Минобрнауки России, федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО» : Санкт-Петербург : Университет ИТМО, 2018. – 112 с.

4 Дополнительные материалы к 6-ому юбилейному изданию учебника «Компьютерные сети» В. Олифер и Н. Олифер : сайт. – Лондон, 2020 – . – URL: http://www.olifer.co.uk/new_rus/dop-comp-seti-6.html (дата обращения 21.09.2020).

Таблица ASCII

Таблица 4 – Символы ASCII с кодами от 80h до FFh

Hex	Char	Cmd
00		NUL
01	☺	SOH
02	☻	STX
03	♥	ETX
04	♦	EOT
05	♣	ENQ
06	♠	ACK
07	•	BEL
08	◘	BS
09	○	TAB
0A	◙	LF
0B	♂	VT
0C	♀	FF
0D	♪	CR
0E	♫	SO
0F	☼	SI
10	►	DLE
11	◄	DC1
12	↕	DC2
13	‼	DC3
14	¶	DC4
15	§	NAK
16	▬	SYN
17	↨	ETB
18	↑	CAN
19	↓	EM
1A	→	SUB
1B	←	ESC
1C	∟	FS
1D	↔	GS
1E	▲	RS
1F	▼	US

Hex	Char	Cmd
20		(sp)
21	!	
22	"	
23	#	
24	$	
25	%	
26	&	
27	'	
28	(	
29	)	
2A	*	
2B	+	
2C	,	
2D	-	
2E	.	
2F	/	
30	0	
31	1	
32	2	
33	3	
34	4	
35	5	
36	6	
37	7	
38	8	
39	9	
3A	:	
3B	;	
3C	<	
3D	=	
3E	>	
3F	?	

Hex	Char	Cmd
40	@	
41	A	
42	B	
43	C	
44	D	
45	E	
46	F	
47	G	
48	H	
49	I	
4A	J	
4B	K	
4C	L	
4D	M	
4E	N	
4F	O	
50	P	
51	Q	
52	R	
53	S	
54	T	
55	U	
56	V	
57	W	
58	X	
59	Y	
5A	Z	
5B	[	
5C	\	
5D	]	
5E	^	
5F	_	

Hex	Char	Cmd
60	`	
61	a	
62	b	
63	c	
64	d	
65	e	
66	f	
67	g	
68	h	
69	i	
6A	j	
6B	k	
6C	l	
6D	m	
6E	n	
6F	o	
70	p	
71	q	
72	r	
73	s	
74	t	
75	u	
76	v	
77	w	
78	x	
79	y	
7A	z	
7B	{	
7C	|	
7D	}	
7E	~	
7F	⌂	DEL

Таблица 5 – Символы ASCII с кодами от 80h до FFh (кодировка IBM CP866)

Hex	Char
80	А
81	Б
82	В
83	Г
84	Д
85	Е
86	Ж
87	З
88	И
89	Й
8A	К
8B	Л
8C	М
8D	Н
8E	О
8F	П
90	Р
91	С
92	Т
93	У
94	Ф
95	Х
96	Ц
97	Ч
98	Ш
99	Щ
9A	Ъ
9B	Ы
9C	Ь
9D	Э
9E	Ю
9F	Я

Hex	Char
A0	а
A1	б
A2	в
A3	г
A4	д
A5	е
A6	ж
A7	з
A8	и
A9	й
AA	к
AB	л
AC	м
AD	н
AE	о
AF	п
B0	▒
B1	▒
B2	▓
B3	│
B4	┤
B5	╡
B6	╢
B7	╖
B8	╕
B9	╣
BA	║
BB	╗
BC	╝
BD	╜
BE	╛
BF	┐

Hex	Char
C0	└
C1	┴
C2	┬
C3	├
C4	─
C5	┼
C6	╞
C7	╟
C8	╚
C9	╔
CA	╩
CB	╦
CC	╠
CD	═
CE	╬
CF	╧
D0	╨
D1	╤
D2	╥
D3	╙
D4	╘
D5	╒
D6	╓
D7	╫
D8	╪
D9	┘
DA	┌
DB	▉
DC	▄
DD	▌
DE	▐
DF	▀

Hex	Char
E0	р
E1	с
E2	т
E3	у
E4	ф
E5	х
E6	ц
E7	ч
E8	ш
E9	щ
EA	ъ
EB	ы
EC	ь
ED	э
EE	ю
EF	я
F0	Ё
F1	ё
F2	Є
F3	є
F4	Ї
F5	ї
F6	Ў
F7	ў
F8	°
F9	∙
FA	·
FB	√
FC	№
FD	¤
FE	■
FF	