4. Промоделюйте отриману схему у Proteus та виведіть відповідні графіки для вхідних та вихідних сигналів, що будуть в повній мірі описувати роботу розробленого вами автомату.

Контрольні питання:

1. Коли доцільно використовувати в якості керуючого автомата автомат Мура на базі регістра зсуву?
2. Чи можна вважати регістр зсуву автоматом Мура на базі регістра зсуву?
3. Чи можна в якості пам'яті автомата Мура на базі регістра зсуву використати Т-тригери?

Лабораторна робота № 9

Принципи захисту інформації від втрат. Коди Хемінга

***Мета роботи**: Ознайомлення з основами захисту інформації від втрат. Провести кодування, декодування та корекцію помилок за допомогою кодів Хемніга.*

Вступ.

Код Хемінга являє собою блочний код, який дозволяє виявити та виправити помилково переданий біт в межах переданого блоку.

Цей алгоритм дозволяє закодувати певне інформаційне повідомлення певним чином і після передачі (наприклад по мережі) визначити чи з'явилася якась помилка в цьому

повідомленні (наприклад через перешкоди) і, при можливості, відновити це повідомлення.

В цій роботі буде розглянуто найпростіший алгоритм Хемінга, який може виправити лише поодиноку помилку в блоці. Також варто зазначити, що існують більш вдосконалені модифікації даного алгоритму, які дозволяють виявляти (і якщо можливо виправляти) більшу кількість помилок.

Відразу варто зазначити, що лабораторна робота з Кодами Хемінга складається з двох частин. Перша частина полягає в кодуванні вихідного повідомлення, вставляючи в нього в визначених місцях контрольні біти (обчислені певним чином). Друга частина роботи полягає в отриманні вхідного повідомлення із блоку закодованого кодом Хемінга (з виправленням можливої поодинокої помилки в блоці). Перевірка на помилку відбувається шляхом перевірки контрольних бітів. Якщо всі обчислені контрольні біти збігаються з визначеними, то повідомлення отримано без помилок. В іншому випадку, виводиться повідомлення про помилку і при можливості помилка виправляється.

Розглянемо принцип роботи алгоритму на прикладі:

Нехай у нас є 16-ти бітне слово:

0100 0100 0011 1101

Задача передати його, від відправника до отримувача. Під час передачі пакету передбачена втрата до 1 біту інформації.

Для забезпечення цільності пакету додаємо контрольні біти на позиції, які відповідають степеням двійки. Їх кількість визначається за принципом: $m = n + k < 2^k - 1$, де n – кількість інформаційних бітів, k – контрольних, m – загальна кількість бітів в пакеті.

Отже, для 16-ти бітного числа буде додано 5 контрольних біт на позиціях 1, 2, 4, 8, 16:

Наш паке набуває вигляду:

**0* 100* 0100 001* 1110 1, де * - місця контрольних бітів.

Після перетворення, та внесення контрольних бітів наш пакт стане 21-бітним.

Обчислення контрольних бітів

Визначення значень контрольних бітів відбувається за наступними правилами:

1. Нумеруємо біти нашого пакету від 1 до 21 в числами в двійковій системі числення.
2. Розбиваємо наші біти на групи, за законом: одиниці в двійковому номері біта вказують, до яких груп належить даний біт. (на приклад п'ятий біт має номер 00101, молодша одиниця вказує, що він належить до молодшої (першої) групи, а 3 одиниця показує, що даний біт належить також і до третьої групи). Див. таблицю 9.1.
3. Якщо, в номері біта одна одиниця, то цей біт є контрольним для групи в якій даний біт також є одиницею (на приклад перший біт має номер 00001, молодша одиниця вказує, що він належить до молодшої (першої) групи, а в силу того що одиниця одна то він є контрольним бітом першої групи). Див. таблицю 9.1.
4. Визначаємо значення контрольних бітів. Значення визначається такип чином, щоб кількість одиниць в кожній групі була парною.

В таблиці 9.1. символом "X" позначені біти, що належать відповідним контрольних групам. Контрольні біти груп відмічені написом (кб) в першому стовпці таблиці.

Таблиця. 9.1. Визначення контрольних груп.

Номер біта в пакеті	Номер в двійковій системі	Біти 1 групи (1)	Біти 2 групи (2)	Біти 3 групи (4)	Біти 4 групи (8)	Біти 5 групи (16)
1 - (кб)	00001	X				

2 - (кб)	00010		X			
3	00011	X	X			
4 - (кб)	00100			X		
5	00101	X		X		
6	00110		X	X		
7	00111	X	X	X		
8 - (кб)	01000				X	
9	01001	X			X	
10	01010		X		X	
11	01011	X	X		X	
12	01100			X	X	
13	01101	X		X	X	
14	01110		X	X	X	
15	01111	X	X	X	X	
16 - (кб)	10000					X
17	10001	X				X
18	10010		X			X
19	10011	X	X			X
20	10100			X		X
21	10101	X		X		X

Далі підраховується кількість одиниць, серед відповідних інформаційних бітів. Якщо їх кількість парна, то контрольний біт встановлюється в 0, інакше – 1.

Таким чином отримаємо наступне число:

100**1** 100**0** 0100 001**0** 1110 1

Декодування та виправлення помилок

Порядок контрольних біт, та місце інформаційних бітів в пакеті дозволяють легко визначити та виправити поодиноку помилку. Наприклад, ми отримали закодоване повідомлення, яке прийшло з помилкою. Припустимо, що 11-й біт був переданий неправильно:

100**1** 100**0** 01**1**0 001**0** 1110 1

Для визначення помилки потрібно підрахувати значення контрольних біт, для отриманого повідомлення:

010**1** 100**1** 01**1**0 001**0** 1110 1

Порівнявши значення отриманих контрольних біт з обрахованими, видно, що біти 1, 2 та 8 не співпадають. Підрахувавши суму номерів некоректних біт:

$$1 + 2 + 8 = 11$$

отримуємо позицію помилкового біту та інвертуємо його

100**1** 100**0** 01**0**0 001**0** 1110 1

Під час виконання роботи слід розробити 2 схеми. Перша отримує вхідний пакет бітів, доповнює їх контрольними бітами та утворює пакет для передачі. На рис. 9.1. дана схема позначена як “Кодер”. Друга схема має отримати пакет даних, виправити помилку в пакеті (якщо вона існує) та видає інформаційні біти. На рис. 9.1. друга схема позначена як “Декодер”.

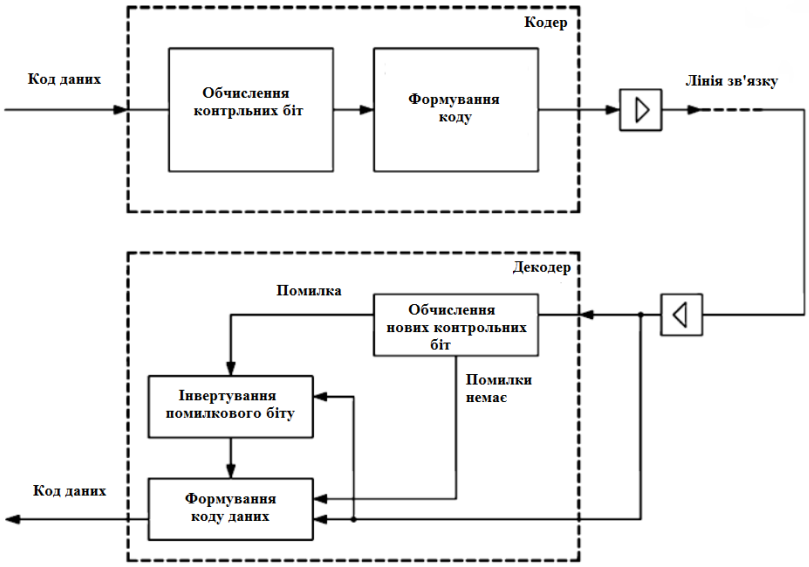

Рис. 9.1 Принципова схема роботи

Для виявлення подвійних помилок можна модифікувати код Хемінга. Дана модифікація дасть можливість як і раніше визначати та виправляти поодиноку помилку, та лише виявляти подвійну помилку.

В модифікованому коді Хемінга в пакет додається ще один додатковий біт (m+1). Даний біт визначає, чи була парна кількість одиниць в попередніх m бітах пакету. Декодування нового пакету набуває наступного вигляду:

1. Контрольні біти груп співпадають, останній контрольний біт не змінився – помилок в пакеті немає.
2. Контрольні біти груп не співпадають, останній контрольний біт змінився – поодинока помилка, що може бути виправлена. Біти контрольних груп, з помилками визначають адресу помилки.
3. Контрольні біти груп не співпадають, останній контрольний біт не змінився – синдром подвійної помилки (можливий варіант парної кількості помилок).

Дані помилки виправити неможливо, пакет використовувати не можна.

4. Контрольні біти груп співпадають, останній контрольний біт змінився – груповий синдром непарної кратності (можливий помилка лише в останньому контрольному біті). Пакет використовувати не можна.

Лабораторне завдання

1. Згідно Вашого варіанту, розробіть функціональні схеми “Кодера” та “Декодера”.

Довжина слова:

Таблиця. 9.2.

H7	H8	H9	H10	Кількість інформаційних бітів
0	0	0	0	14
0	0	0	1	13
0	0	1	0	17
0	0	1	1	11
0	1	0	0	19
0	1	0	1	16
0	1	1	0	12
0	1	1	1	22
1	0	0	0	18
1	0	0	1	23
1	0	1	0	15
1	0	1	1	25
1	1	0	0	10
1	1	0	1	27
1	1	1	0	20
1	1	1	1	29

Функціональну схему синтезувати на елементах:

Таблиця. 9.3.

H5	H4	Логічні елементи
0	0	І, АБО, НЕ, Виключне АБО
0	1	І-НЕ, Виключне АБО
1	0	АБО-НЕ, Виключне АБО
1	1	І, АБО-НЕ, Виключне АБО

2. Введіть у Proteus функціональні схеми.
3. Проведіть моделювання отриманих Вами схем. Занесіть помилку у вхідний пакет “Декодера”. Переконайтесь в коректності роботи Вашої схеми.
4. Занесіть додатковий біт парності для пакету інформації, що передається (для виявлення подвійної помилки).
5. Модернізуйте Ваші схеми для створення та обробки даних пакетів.
6. Проведіть моделювання отриманих Вами схем. Занесіть поодинокі та подвійні помилки в вхідний пакет “Декодера”. Проаналізуйте отримані Вами результати роботи схем.

Контрольні питання:

1. Які Ви знаєте коди, що дозволяють виявити та виправити помилки?
2. В чому полягають принципи алгоритмів виправлення помилок.
3. Яку мінімальну кількість контролюючих бітів треба внести в вхідне слово, щоб виявити та виправити поодиноку помилку.