МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп’ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Головецький Віктор Васильович

Апаратна реалізація алгоритмів кодування і декодування згорткових кодів / Hardware realization of coding and decoding algorithms of convolutional codes

напрям підготовки: 6.050102 - Комп’ютерна інженерія
фахове спрямування - Комп’ютерні системи та мережі
Бакалаврська робота

Виконав студент групи КСМз-41/2
В.В. Головецький

Науковий керівник:
Якименко І.З.

Тернопіль - 2018

ЗМІСТ

ВСТУП

Дипломна робота присвячена розробці та дослідженню структурних і принципових електричних схем, програм кодування і декодування двійкових згорткових кодів, а також методик оцінки завадостійкості систем зв'язку, що використовують двійкові згорткові коди.

Згорткові коди мають високі коректуючі властивості. Вони застосовуються в різних системах перешкодостійкої передачі інформації, в системах стільникового зв'язку, в магнітних накопичувачах інформації з високою щільністю запису, в системах далекого космічного зв'язку. Реалізація пристроїв кодування та декодування на інтегральних мікросхемах істотно розширює коло технічно реалізованих рішень.

Таким чином, широке використання згорткових кодів в системах цифрового зв'язку є досить актуальним. Для ефективного вивчення згорткових кодів і їх застосування можна використовувати лабораторні роботи відповідним методичним забезпеченням, пропоновані в даній роботі (див. додатки А, Б).

Мета і завдання дослідження.

Метою дипломної роботи є розробка схем, програмних модулів кодування і декодування двійкових згорткових кодів, а також дослідження завадостійкості згорткових кодів і розробка лабораторних робіт по цій темі.

Для досягнення поставленої мети необхідно вирішити описані нижче завдання:

- дослідити впливу параметрів на роботу довічних згорткових кодів;
- розробити методику побудови електричних принципових схем кодерів і їх моделювання в середовищі QUARTUS II;
- провести аналіз способів кодування згорткових кодів за допомогою діаграм станів, деревовидних діаграм і ґратчастих діаграм;

- застосувати спосіб декодування по Вітербо;

- розробити програмні модулі кодування і декодування на основі математичних моделей і алгоритмів;

- дослідити залежність ймовірності бітових помилок (BER) відношення сигнал / шум в цифровій системі передачі даних в середовищі MATLAB;

- створити віртуальну лабораторну роботу по згорткових кодами.

Особистий внесок автора.

Автором дипломної роботи виконано розробку принципових електричних схем кодерів в середовищі QUARTUSII, які можуть бути «зашиті» в кристал програмованих логічних інтегральних схем (ПЛІС). Ним особисто виконана доробка програмних модулів MATLAB, що дозволяють досліджувати стійкість згорткових кодів.

1 КОДУВАННЯ ДВІЙКОВИХ ЗГОРТКОВИХ КОДІВ

1.1 Визначення згорткових кодів

Двійкові згорткові коди - це коди, що виправляють помилки, які використовують безперервну або послідовну обробку інформації короткими фрагментами (блоками) [1].

Найменування «згорткове кодування» походить від того, що результат кодування на виході кодера утворюється як згортка інформаційної послідовності яка кодується з імпульсною характеристикою кодера [2].

Згорткові коди засновані на перетворенні вхідної послідовності двійкових символів в вихідну послідовність двійкових символів, у якій на кожен символ вхідної послідовності формується більш одного символу вихідний послідовності [1-3].

Згорткове кодування найзручніше вивчати, аналізуючи роботу кодових пристроїв [4].

В загальному випадку згортковий кодер складається з m-розрядного регістра зсуву incумматоров по модулю два. Значення m називають пам'яттю коду. Величина *n* - це число символів на виході кодера, відповідних kінформаціонним символам, що надійшли на вхід кодера за один такт [5].

Різниця *r* = *n*-*k* називається числом перевірочних символів. значення *n* вихідних кодових символів рівні лінійним комбінаціям відповідних інформаційних символів. Звідси виконується властивість лінійності згорткових кодів. На кожному такті роботи на вхід кодера подається *k* інформаційних символів і счітиваетсяnсімволов, призначених для передачі по каналу зв'язку вихідних символів. Спосіб підключення кожного суматора до регістру відображається відповідним породжує поліномом.

Таким чином, згорткових кодер має пам'ять. Символи на його виході залежать не тільки від інформаційних символів на вході, а й від попередніх вхідних символів.

Стандартний згорткових кодер, продемонстрований на рисунку 1.1, реалізується з *m* - розрядних регістром зсуву і *n* суматора за модулем два [6].

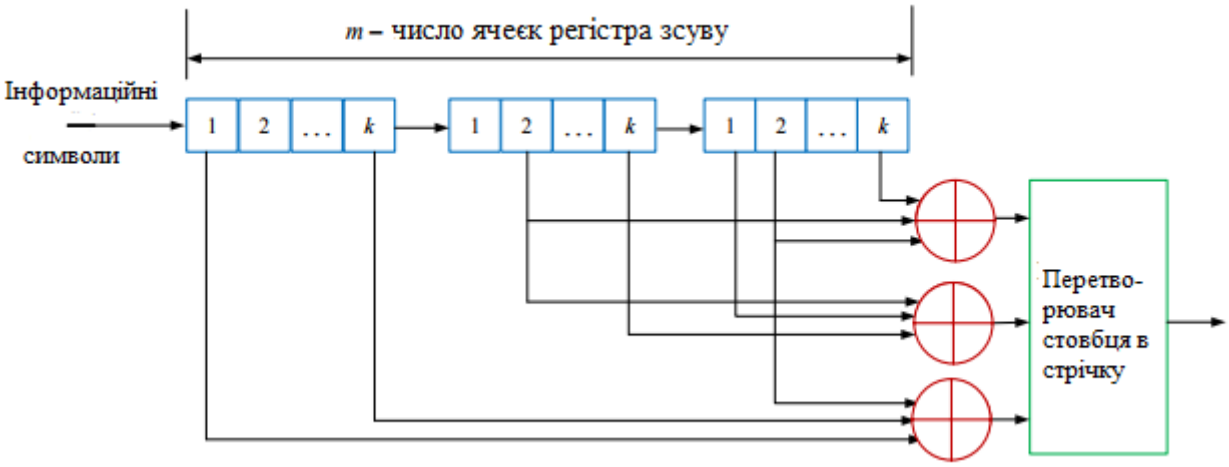


Рисунок 1.1 – Структурна схема кодера згорткового коду

В кожен момент часу на місце перших *k* розрядів регістра переміщаються *k* нових біт; всі біти в регістрі зміщуються на *k* розрядів вправо і на виходах суматорів за модулем два утворюється кодові символи.

Входи суматорів з'єднані з певними розрядами регістра зрушень. Перетворювач на виході кодера встановлює порядок відправки кодових символів в канал. За час одного інформаційного символу на виході утворюється n кодових символів. Потім ці символи коду застосовуються модулятором для освіти сигналів, щоб передати по каналу зв'язку [5].

Для прикладу розглянемо згортковий кодер з параметрами: *m*=3, *k*=1, *n*=2. Структурна схема кодера показана на рисунку 1.2.

Спочатку всі комірки регістра зсуву розташовані в нульовому стані. У разі якщо перший вхідний біт дорівнює одиниці «1, він без затримки з'явиться на виході першої (лівої) осередки регістру і, відповідно, на двох входах вихідного перетворювача (мультиплексор). Перетворювач черзі видає вміст входів, і вихідна послідовність буде 11.

Припустимо, що другий вхідний біт «0». Він записується в перший осередок регістру, проштовхує попередній біт («1») в другий осередок і на входах перетворювача (зверху вниз) з'являються 01. Тоді друга вихідна послідовність 01. Якщо третій вхідний біт 1, вихідна послідовність 00 і так далі.

Таким чином, у відповідь на кожен вхідний біт (k = 1) згорткових кодер кодує двома бітами, по числу суматори за модулем два (n=2).

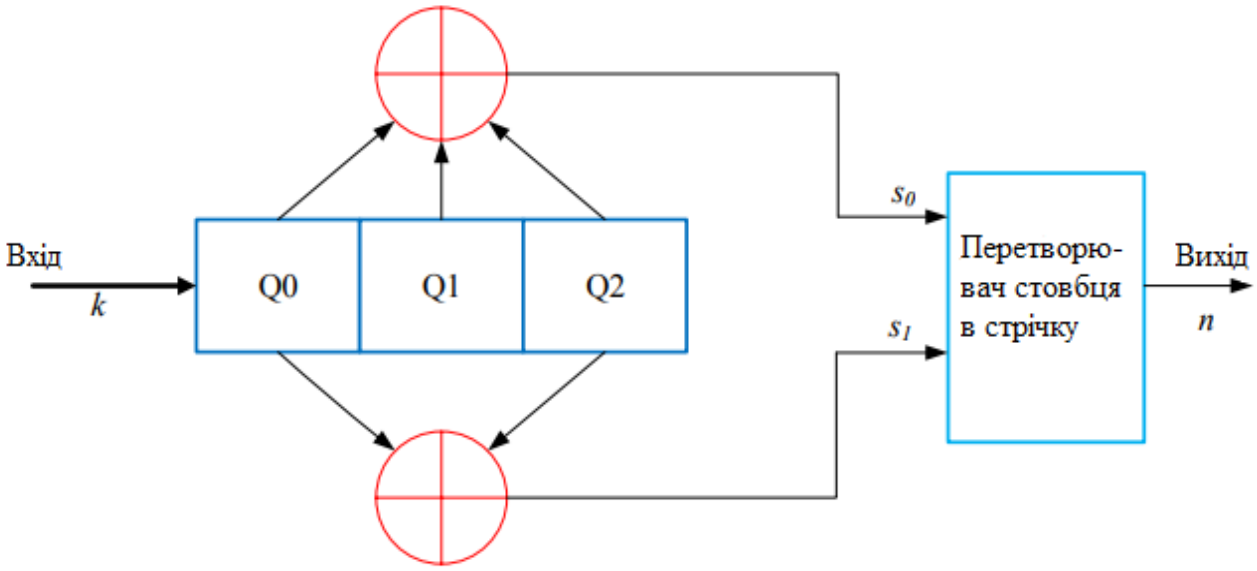


Рисунок 1.2 – Структурна схема кодера згорткового коду з параметрами m=3, k=1, n=2

де Q_0, Q_1, Q_2 – осередки регістру зсуву;

s_0, s_1– виходи суматорів.

Розглянемо на прикладі, як формується вихідна кодова послідовність для вхідного сигналу (110100) (таблиця 1.1).

На виході кодера отримуємо кодову послідовність: s = (11 01 01 00 10 11).

Згорткове кодування зручно задавати за допомогою породжують (виробляють) многочленів. Породжують многочлени цілком визначають структуру кодера згортальної коди.

Таблиця1.1 – Формування кодової послідовності

Номери тактових імпульсів						
Q_0						
Q_1						
Q2						
s_0						
s_1						
S	11	01	01	10	11	

1.2 Поліноміальне завдання згорткового кодера

Згорткове кодування - це ітеративна обробка потоку бітів, що створює залежність кожного біта від декількох попередніх.

Згорткове кодування задають за допомогою породжують полиномов, які визначають структуру двійкового кодера згортальної коди.

Кодове слово на виході такого кодера складається у вигляді в двох послідовностей, які в двійковій формі представляють коефіцієнти відповідних породжуючих поліномів [7].

Для опису згортальної коди необхідно кілька породжують полиномов, число яких визначаються чіслomn вихідних символів, переданих в канал зв'язку за кожен такт [4].

Нехай $g_{ij}(x), i = 1,..., k, \quad j = 1,..., n$ - безліч породжуючих поліномів. Вони можуть бути об'єднані в матрицю розміру $k*n$.

$$G(x) = [\, g_{ii}(x) \,]. \tag{1.1}$$

Для прикладу розглянемо згортковий кодер на рисунку 1.2. Породжуючі поліноми мають вигляд:

$$g_1(x)=1+x+x^2,\ g_2(x)=1+x^2. \quad (1.2)$$

Породжуюча матриця для цього кодера має вигляд:

$$G(x)=\left[1+x+x^2 \cdot 1+x^2\right]. \quad (1.3)$$

Розглянемо, кодування послідовності інформаційних символів (110100). Послідовність символів, що надходять на вхід кодера, представимо у вигляді многочлена:

$$a\,(x)=1\ +x+x^3 \quad (1.4)$$

де x^i– це оператор затримки зрушується регістру на і тактів;

a_i –0;1 інформаційні виконавчі символи.

В силу лінійності згортальної коди:

$$s\,(x)=G\,(x)\cdot a\,(x) \quad (1.5)$$

В результаті на виході кодера буде утворена послідовність *s(x)*:

$$S\,(x)=\left[1+x+x^3\right]\cdot\left[1+x+x^2*1+x^2\right]=\left[1+x^3+x^5*1+x+x^2+x^5\right]$$
$$\mathrm{S}=[100101111001] \quad (1.6)$$

Звідси видно, що на виході кодера формується кодова послідовність *s* = (11 01 01 10 00 11), яка збігається з результатом в таблиці 1.1.

Разом з цим для формування згорткових кодів слід перерахувати параметри, що визначають структуру кодів.

1.3 Основні параметри згорткових кодів

Число контактів мультиплексора і тактова частота перемикання в згорткових кодерах залежить від відносної швидкості коду рівній $R = k / n$ і обумовлює надмірність, що вводиться при кодуванні.

Відповідно до цього частота перемикання повинна в nраз перевищувати вхідну тактову частоту. Так, при швидкості $R = 1 / 2$ в мультиплексора перемикання повинно проводитися з частою в 2 рази більшою тактової частоти [4, 8].

Надмірність коду визначається за формулою:

$$x = 1 - R = 1 - \frac{k}{n} \tag{1.7}$$

Таким чином, при R = 1 / 3 кількість символів у вихідній послідовності більше їх кількості у вхідній інформаційній послідовності в три рази.

Повна довжина кодового обмеження щодо виходу кодера- це число послідовних кодових символів на виході кодера, що залежать від обраного інформаційного символу, і визначається за формулою [4]:

$$l_2 = \frac{m}{k} \cdot n \tag{1.8}$$

При $k = 1$ довжина кодового обмеження щодо виходу дорівнює $l_2 = m \cdot n$.

Значення l_2 по виходу можна визначити максимальним ступенем породжує полінома за формулою [2]:

$$l_2 = n \cdot \max\left[\deg g_{ij}(x) + 1\right] \tag{1.9}$$

де $deg\ g_{j,i}(x)$ – степінь полінома.

Значення l_2 показує на яку максимальну кількість вихідних символів впливає даний інформаційний символ.

Розглянемо для прикладу згорткових кодер на рисунку 1.2. Швидкість коду цього кодера R=1/2.

Знайдемо надмірність коду за формулою (1.8):

$$x = 1 - R = 1 - 0.5 = 0.5 \tag{1.10}$$

Повна довжина кодового обмеження щодо виходу (1.8) дорівнює:

$$l_2 = 3 \cdot 2 = 6 \tag{1.11}$$

За другою формулою отримуємо той же результат:

$$l_2 = n \cdot \max\left\lfloor \deg g_{ij}(x) + 1 \right\rfloor = 2 \cdot (2 + 1) = 6 \tag{1.12}$$

Звідси можна сказати, що вхідний один інформаційний символ впливає на шість вихідних символів.

1.4 Згорткові коди в блоковому вигляді

Кодування може проводитися в безперервному або блочному режимах. В останньому випадку інформаційні послідовності розбиваються на блоки скінченної довжини [1, 9-10].

У блочному режимі за a двійковим символом (останнім) в кодер повинні бути введені m-1 нулів для того, щоб очистити реєстр, які іноді називають хвостом коду. Це необхідно для того, щоб зробити код кінцевим.

Згорткове кодування можна розглядати так само, як довгий блоковий код і вводиться тут обмеження довжини за допомогою m -1 нулів дозволяє очистити реєстр кодера для наступного блоку [9, 12].

Якщо в послідовності кодових символів, що формуються кодером, можна відокремити $r = n\text{-}k$ перевірочних символів від k інформаційних, то код називають систематичним.

Систематичним згорткових кодом називають код, для якого в вихідний послідовності сформованих кодових символів без зміни міститься породила її послідовність інформаційних символів. В інших випадках згортковий код є несистематическим.

При використанні систематичного кодера на k виходах будуть інформаційні послідовності. На інших (n-k) виходах будуть послідовності перевірочних символів, утворені як лінійні комбінації інформаційних [5].

Для прикладу розглянемо систематичний згортковий кодер з параметрами k=1, n=2 і m=3 (рисунок 1.3).

Для того щоб отримати інформаційну послідовність разом з вихідною в кодерах ck = 1один з породжуючих многочленів, які формують систематичні коди, дорівнює одиниці, тобто $g_1(x)$=1 або $g_2(x)$=1, [4].

На рисунку 1.3 видно, що це систематичний згортковий кодер. Нехай на вхід кодера надходить інформаційна послідовність ($a_0a_1a_2a_3$). На виході ми можемо відокремити bi перевірочних символів від ai інформаційних символів. Кодова обмеження дорівнює трьом, тоді в кодер будуть введені m - 1 = 2 нулів (рисунок 1.3). Тоді код буде систематичним блоковим кодом.

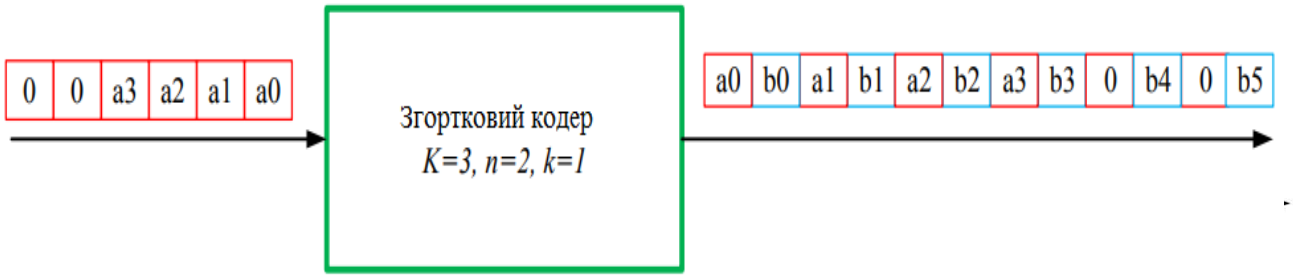


Рисунок 1.3 – Систематичний згортковий кодер

Перевагою систематичних кодерів є отримання оцінки інформаційних символів на приймальній стороні без декодування, або іншої обробки вхідних символів [5, 11].

1.5 Принципові електричні схеми кодерів в середовищі програмованих логічних інтегральних схем QUARTUSII

1.5.1 Згортковий кодер з параметрами m=4, k=1, n=2

Розглянемо згортковий кодер з параметрами:

– m=4 – число ячейок регістра зсуву;

– k= 1 – число інформаційних символів, що надходять за один такт на вхід кодера;

– n=2– число символів на виході кодера, відповідних k; R=1/2– швидкість коду;

– g_1=17, g_2=15 – коефіцієнти породжують полиномов в вісімковій формі.

Схема кодера показана на рисунку 1.4 [4].

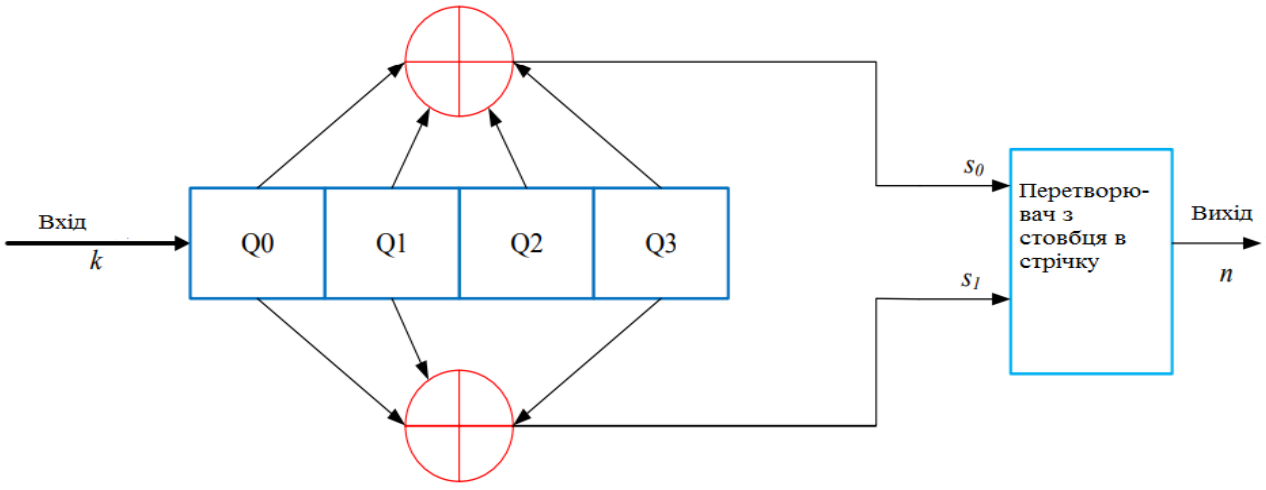


Рисунок 1.4 – Структура згорткового кодера з вхідними параметрами m=3, k=1, n=2

де Q_0, Q_1, Q_2, Q_3 – осередки регістру зсуву;

s_0, s_1– виходи суматорів.

Інформаційні символи на схемі надходять зліва, і для кожного інформаційного символу на виходах двох суматорів за модулем два утворюються два вихідних символи [13-14].

Зв'язок між осередками регістра зсуву і суматорами зручно описувати породжуючими поліномами: верхній і нижній суматори представляються відповідно поліномами:

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

Коефіцієнти породжують полиномов в двійковій формі:

$$g_2=(1101) \qquad (1.13)$$

Розглянемо на прикладі, як формується вихідна кодова послідовність для вхідного сигналу (1101000) (таблиця 1.2).

Таблиця 1.2– Формування кодової послідовності

Номера тактових імпульсів							
Q_0							
Q_1							
Q_2							
Q_3							
s_0							
s_1							
s	1	0	1	0	0	0	1

На виході кодера отримаємо кодову послідовність: *s* = (11 01 00 10 10 01 11). На рисунку 1.2 і 1.4 зображені структурні схеми кодера. Для практичної реалізації таких кодерів необхідно розробити їх до рівня принципових електричних схем. Таке завдання можна вирішити застосовуючи, наприклад, середу програмованої логіки (Altera QUARTUSII).

1.5.2 Опис середовища Altera QUARTUSII

QUARTUSII являє собою автоматизовану систему наскрізного проектування цифрових пристроїв на кристалах ПЛІС фірми Altera [15].

Він надає користувачеві широкі можливості по введенню описів проекту, логічного синтезу, компіляції проекту, програмування ПЛІС, функціональному та тимчасового моделювання, тимчасового аналізу та аналізу споживаної потужності проекту, реалізації внутрісистемної налагодження. У QUARTUSII використовується зручний графічний інтерфейс і проста в застосуванні довідкова система, що містить всю необхідну для виконання проектування інформацію.

Також пакет дозволяє використовувати командний рядок для виконання кожного етапу проектування.

QUARTUSII інтегрує в собі велику кількість програмних модулів, призначених для виконання різних етапів проектування. Завдання параметрів і виконання типових команд виконується в окремих модулях однаково, що значно полегшує роботу користувача. Редактори вихідних файлів проекту (графічний, текстовий, редактор символів, вмісту модулів пам'яті, тимчасових діаграм, кінцевих автоматів) використовують однакові підходи і прийоми, а також схожі віконні форми, що застосовуються при створенні і редагуванні вихідних файлів з описом модулів проектованого пристрою [15].

До складу стандартної бібліотеки QUARTUSII входить велика кількість базових елементів, включаючи мега функції і макрофункції. Складовою частиною мега функцій є операційні пристрої, створені за стандартом бібліотеки параметрізіруемих модулів (LPM – libraryofparameterizedmodules).

Принципова електрична схема згорткового кодера в середовищі програмованих логічних інтегральних схем QUARTUS II будується за допомогою D-тригера.

1.5.3 Принцип роботи D-тригера

D-тригер (від англійського delay) називають інформаційним тригером, а також тригером затримки. D-тригер буває тільки синхронним. Він може управлятися (перемикатися) як рівнем тактирующего імпульсу, так і його фронтом. Для тригера типу D, стан в інтервалі часу між сигналом на вхідний лінії і таким станом тригера формується простіше, ніж для будь-якого іншого типу. Схемне позначення D-тригера наведено на рисунку 1.5. За синхроімпульсів D-тригер приймає той стан, який має вхідна лінія. На рисунку 1.6 наведені часові діаграми, що пояснюють його роботу. D-тригер має як мінімум дві вхідні лінії: одна - для подачі синхроімпульсів; інша-інформаційних сигналів [16].

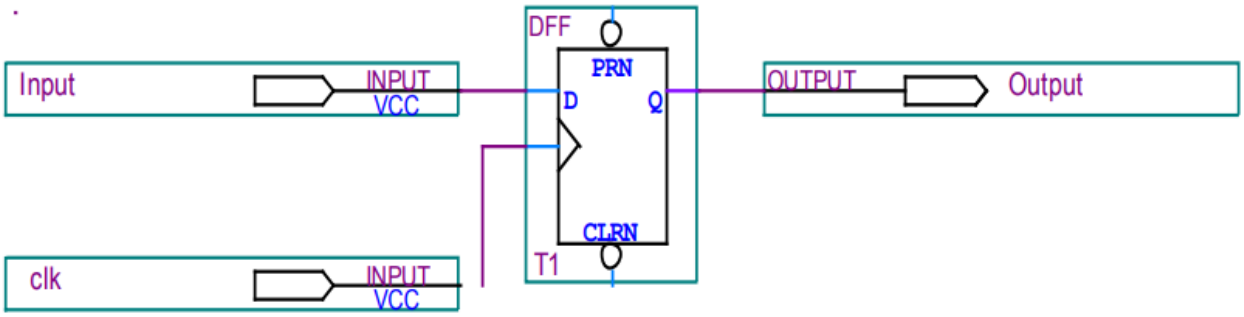


Рисунок 1.5 – D-тригер

D-тригер запам'ятовує стан входу і видає його на вихід. D-тригери мають як мінімум два входи: інформаційний D і синхронізації C. Після приходу активного фронту імпульсу синхронізації на вхід CD-тригер відкривається. Збереження інформації в D-тригерах відбувається після спаду імпульсу синхронізації, і ця інформація залишається незмінною до приходу наступного імпульсу синхронізації. Також через перехідних процесів існує затримка між часом

приходу певного фронту синхроимпульса і часом запам'ятовування інформаційного біта в пам'ять тригера [16].

Принципова схема D-тригера в середовищі програмованих логічних інтегральних схем QUARTUS II представлена на рисунку 1.5.

На рисунку 1.6 представлені часові діаграми роботи D-тригера.

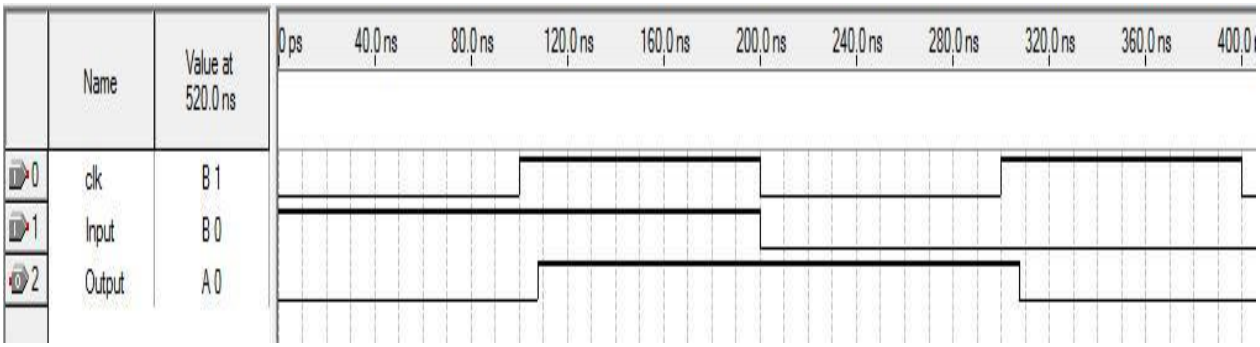


Рисунок 1.6 – Часові діаграми роботи D-тригера

Як видно з малюнка 1.6, при приході на вхід тригера одиниці і переднього фронту синхроимпульса на вихід D-тригера також передається одиниця до часу приходу наступного синхроімпульса, але з деякою затримкою, про яку було сказано раніше.

1.5.4 Побудова кодера в середовищі програмованих логічних інтегральних схем QUARTUSII

Побудуємо згортковий кодер (див. рисунок 1.2) в середовищі програмованих логічних інтегральних схем QUARTUS II (рисунок 1.7) [15].

Згортковий кодер на рисунку 1.7 складається з D-тригерів і елементів XOR (суматори за модулем два) і перетворювача (мультиплексора).

Тут відсутній перший осередок пам'яті. Це пояснюється тим, що регістр зсуву можна розглядати або як регістр, вміст якого зсувається на один розряд вправо при введенні в нього зліва кожного нового двійкового символу (вміст самого правого розряду при цьому втрачається, або як цифрову лінію затримки, в

якої кожен елемент затримки зберігає один двійковий символ до надходження нового вхідного двійкового символу (див. рисунок 1.2).

Ці дві схеми (рисунок 1.2 і рисунок 1.7) еквівалентні.

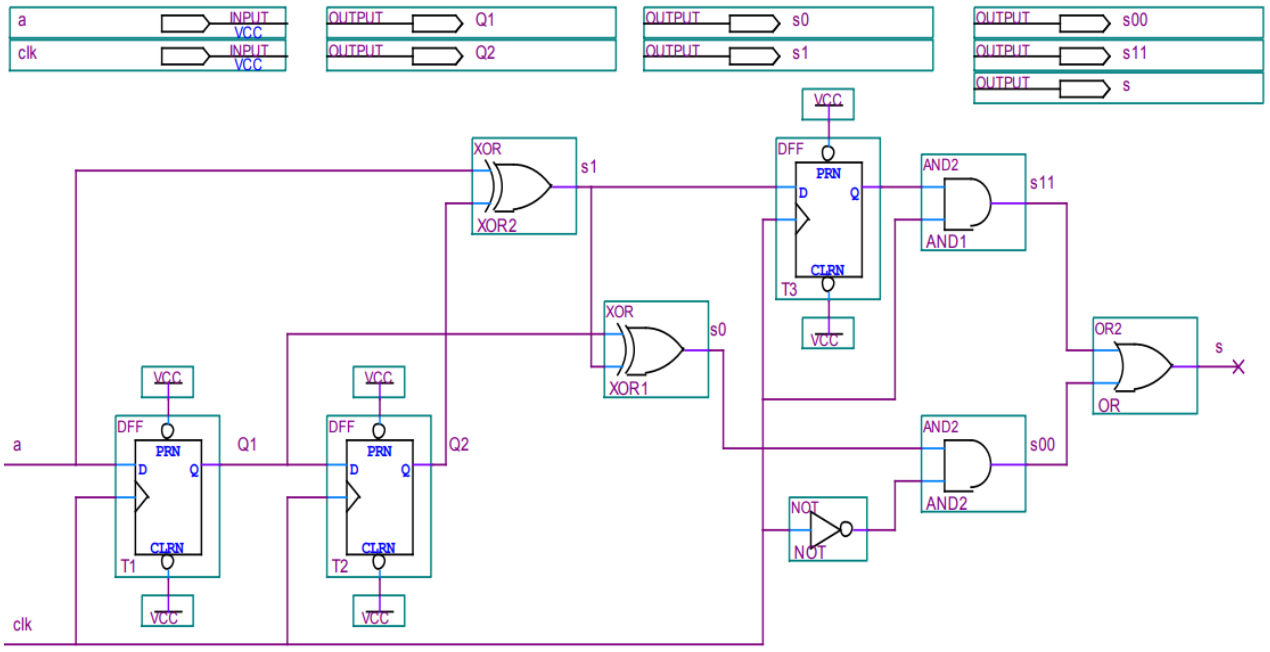


Рисунок 1.7 – Схема кодера в середовищі програмованих логічних інтегральних схем QUARTUSII

Мультиплексор складається з: D-тригера і логічних елементів AND, NOT, OR (рисунок 1.8).

На вхід мультиплексора надходять паралельні символи (виходи двох суматорів). Мультиплексор за першу половину такту роботи кодера зчитує дані спочатку з логічного елемента XOR2, а другу половину такту з логічного елемента XOR3.

Вихід першого суматора надходить на перший вхід логічного елемента AND2, а на другий вхід надходить інвертований тактовий імпульс (за допомогою логічного елемента NOT). Якщо на вхід надходить 1 + 1, то на виході AND2 отримаємо 1. В інших випадках 0

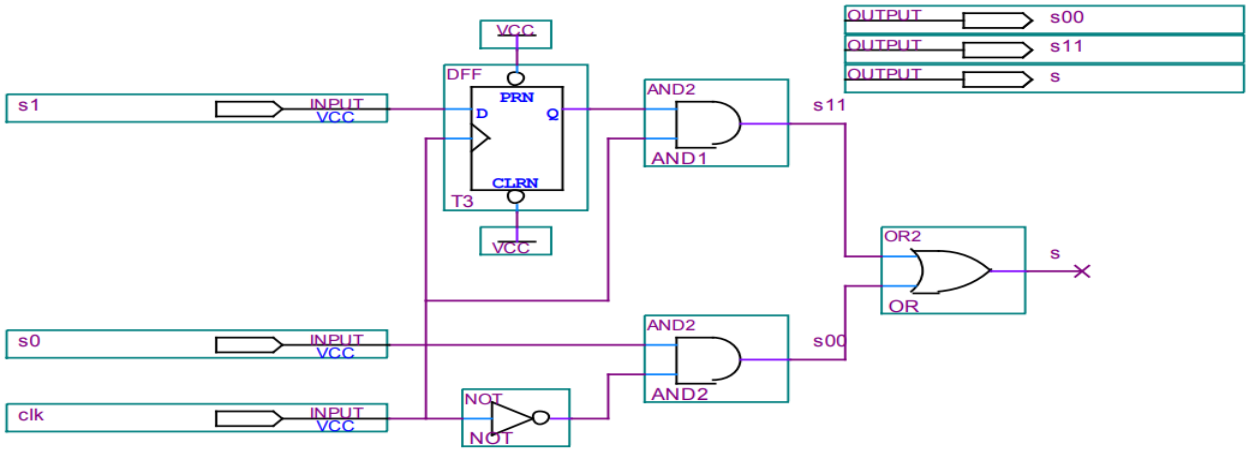


Рисунок 1.8 – Мультиплексор в середовищі QUARTUS II

Вихід другого суматора надходить на інформаційний вхід D-тригера, а на другий вхід надходить тактовий імпульс. Цей символ з'являється на виході тригера в момент першого фронту.

Далі він надходить на перший вхід логічного елемента AND1, а на другий вхід надходить тактовий імпульс. Якщо на вхід надходить 1 + 1, то на виході AND1отримаємо 1. В інших випадках 0.

Далі виходи AND1 і AND2 надходять на вхід логічного елемента OR. Якщо на вхід надходять 0 + 0 на виході отримаємо 0. У решті випадків 1 (рисунок 1.9). Таким чином отримаємо кодову послідовність на виході мультиплексора.

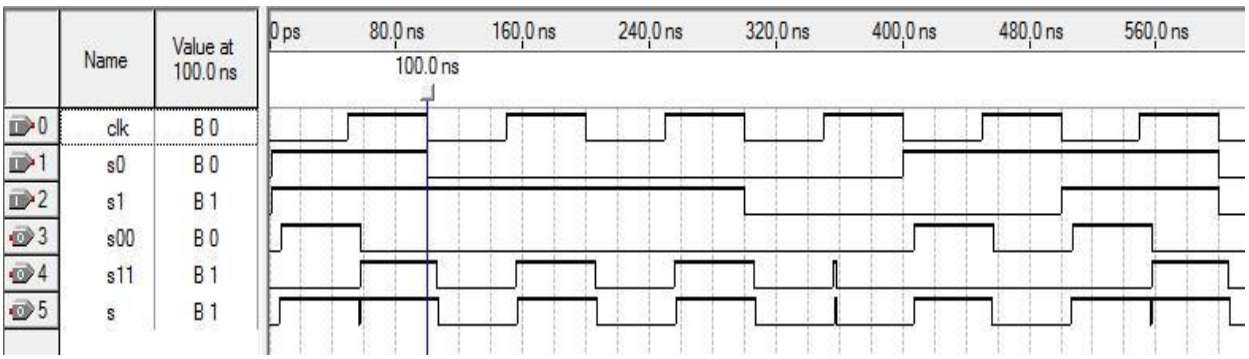


Рисунок 1.9 – Перетворення паралельного сигналу в послідовний сигнал

За часової діаграми (рисунок 1.10) можна помітити, що при вхідній послідовності біт (110100) вихідна послідовність буде (11 01 01 00 10 11). Одержимо той же результат, який знайдений в таблиці 1.1.

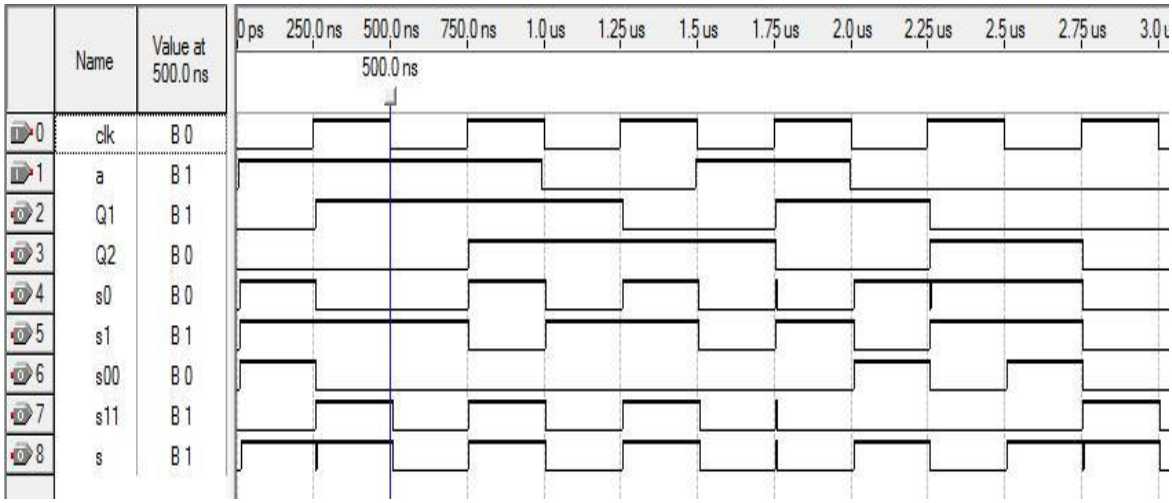


Рисунок 1.10 – Часові діаграми кодера в середовищі QUARTUS II

Варто зазначити, що важливою особливістю принципу формування дібітів є те, що значення кожного формованого дібіт залежить як від вхідного інформаційного біта, так і від двох попередніх бітів, значення яких зберігаються в двох запам'ятовуючих осередках.

Таким чином, якщо прийняти, що a_i - вхідний біт, то значення елемента XOR1 буде визначатися виразом:

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

а значення елемента XOR2 виразом:

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

Відповідно, дібіт формується з пари бітів, значення першого з яких одно (1.13), а другого – (1.14).

Інакше кажучи, значення дібіт залежить від трьох станів: значення вхідного біта, значення першої пам'ятного осередку і значення другого пам'ятного осередку.

Побудуємо згортковий кодер на рисунку 1.4 в середовищі програмованих логічних інтегральних схем QUARTUS II (рисунок 1.11).

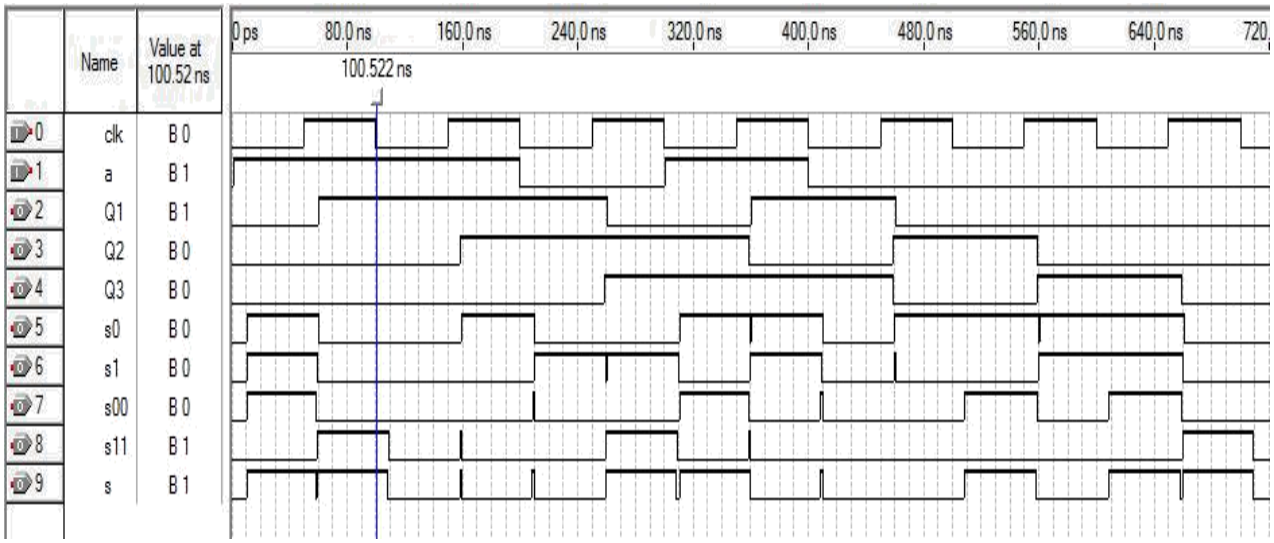


Рисунок1.11 – Схема кодера в середовищі QUARTUS II

Згортковий кодер на рисунку 1.11 складається з трьох запам'ятовуючих ячейок (D тригери) і елементів XOR (суматор за модулем 2) і перетворювача (мультиплексор).

Принцип роботи перетворювача розглянуто вище.

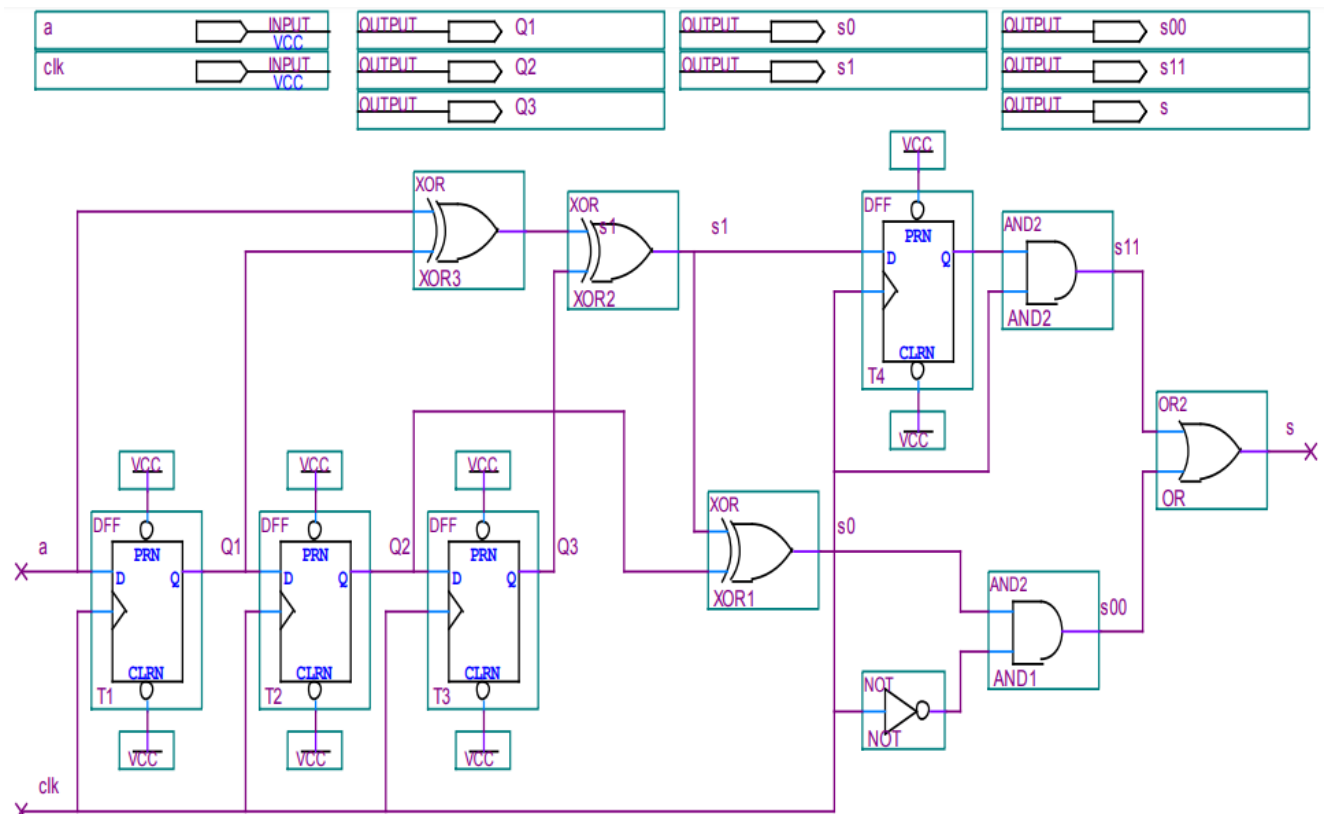


Рисунок 1.12 – Часові діаграми кодера в QUARTUS II

Як видно з часової діаграми (рисунок 1.12) при вхідній послідовності біт (1101000) вихідна послідовність буде (11 00 01 10 00 10 11). Отримано той же результат, який знайдений в таблиці 1.2.

1.5.5 Згортковий кодер(m=4, k=1, n=3)

Розглянемо згортковий кодер зі швидкістю R = 1/3. Структурна схема кодера показана на рисунку 1.13 [17].

У цьому прикладі інформаційні символи на схемі надходять зліва, і для кожного інформаційного символу на виходах трьох суматори за модулем 2 утворюються три вихідних символу.

Породжують поліноми згорткового кодера мають вигляд:

$$g_1(x)=1+x+x^2+x^3,$$
$$g_2(x)=1+x+x^3, \qquad (1.16)$$
$$g_3(x)=1+x^2+x^3.$$

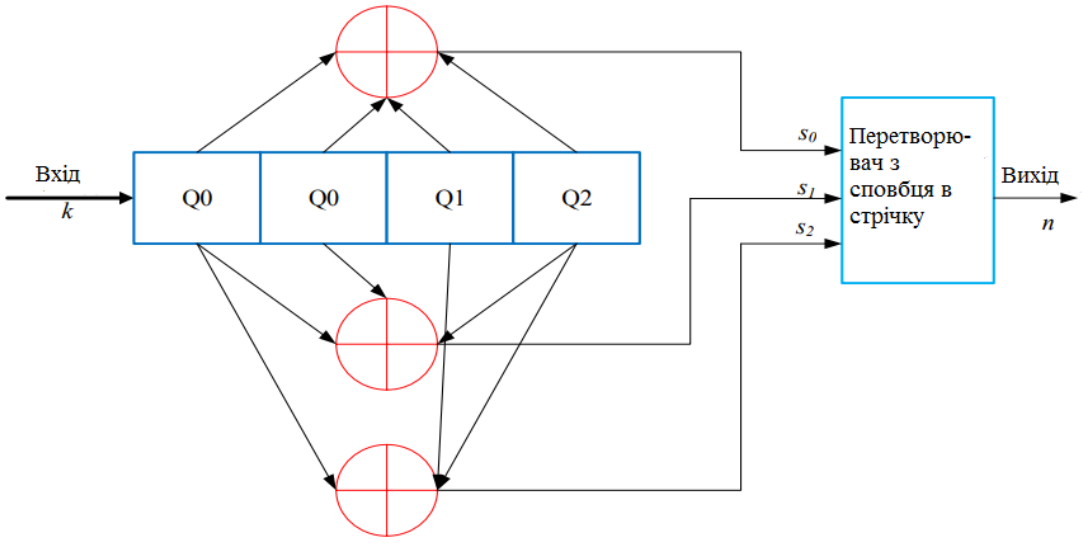


Рисунок 1.13–Структурна схема кодера з параметрами m= 4, k=1, n=3

де Q_0, Q_1, Q_2 – ячейки регістра зсуву;

s_0, s_1– виходи суматорів.

Розглянемо на прикладі, як формується вихідна кодова послідовність для вхідного сигналу (1101000) (таблиця 1.3).

Таблиця 1.3– Формування кодової послідовності

Q_0						
Q_1						
Q_2						
s_0						
s_1						
S	1	1	1	0	0	1

На виході кодера отримаємо кодову послідовність S = (111 001 011 101 001 101 111).

Побудуємо згортковий кодер на рисунку 1.13 в середовищі програмованих логічних інтегральних схем QUARTUS II (рисунок 1.14).

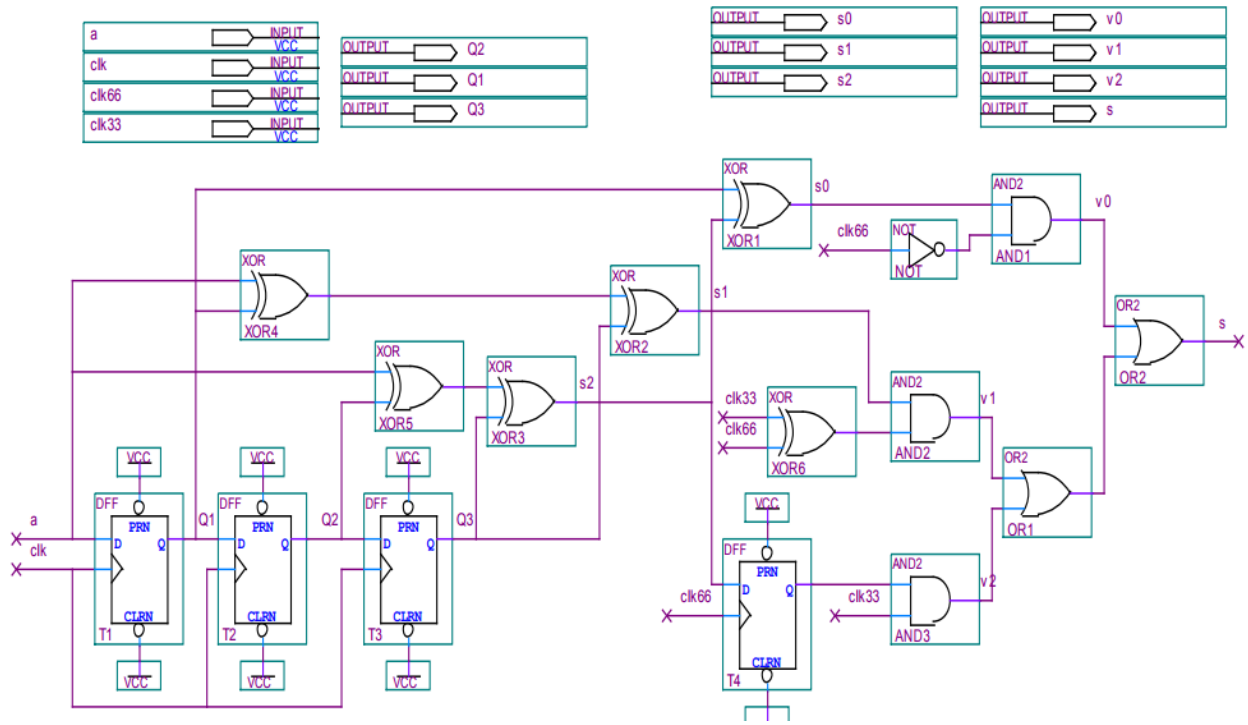


Рисунок 1.14 – Схема кодера в середовищі QUARTUS II

Згортковий кодер на рисунку 1.14 складається: з трьох запам'ятовуючих ячейок (D тригерів), елементів XOR (суматори за модулем 2) і перетворювача (мультиплексора) [16].

Мультиплексор складається з: D-тригера і логічних елементів NOT, AND, OR, XOR (рисунок 1.15).

З погляду на, те що мультиплексор працює з частотою в три рази більшою, ніж швидкість надходження бітів на вхід кодера, швидкість вихідного потоку буде втричі вищою за швидкість вхідного потоку. За допомогою clk33, clk66 розділимо тривалість вихідного тактуючого елемента на три.

Мультиплексор за першу третину такту кодера отримує дані спочатку з логічного елемента XOR1, наступну третину такту - з логічного елемента XOR2, а останню третину - з логічного елемента XOR3.

Таким чином, кожному вхідному біту відповідає три вихідних біта, тобто три біта, перший біт якого формується елементом XOR1, другий елементом XOR2, а третій елементом XOR3.

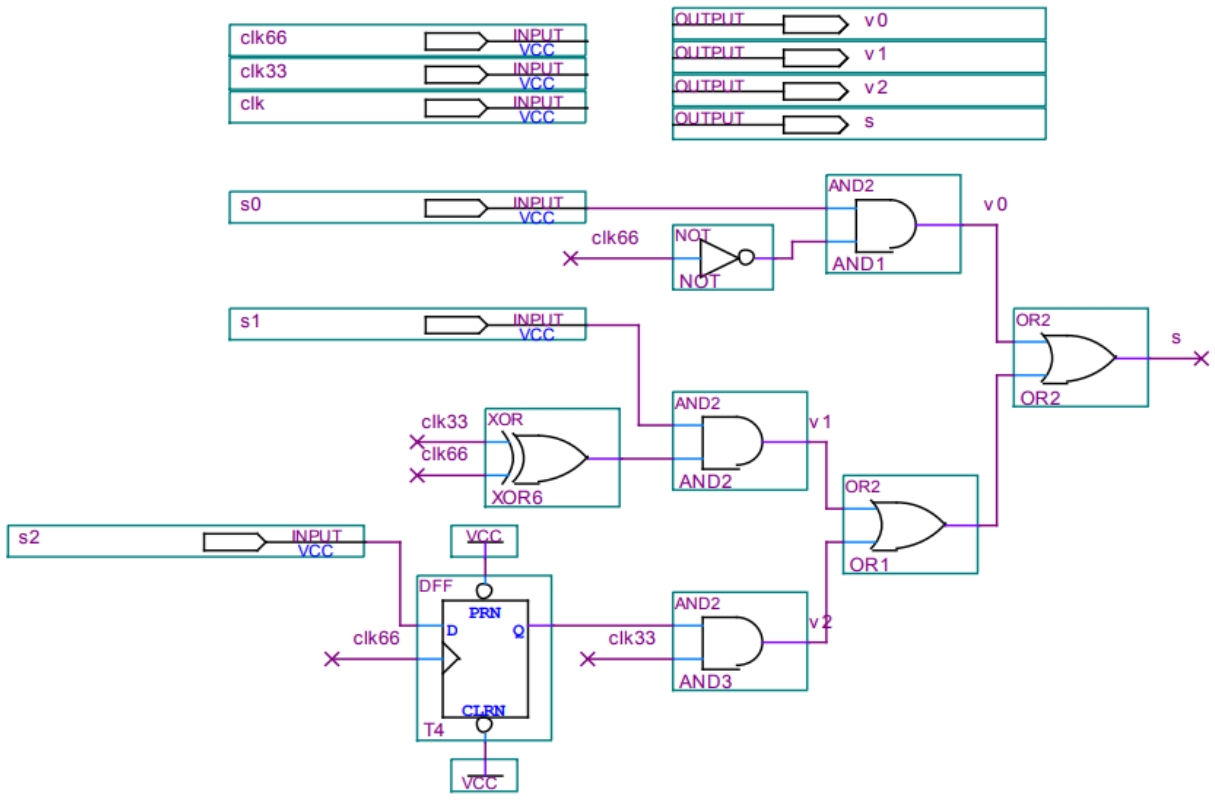


Рисунок 1.15 – Мультиплексор

Принцип роботи мультиплексора представлений у вигляді тимчасових діаграм (рисунок 1.16).

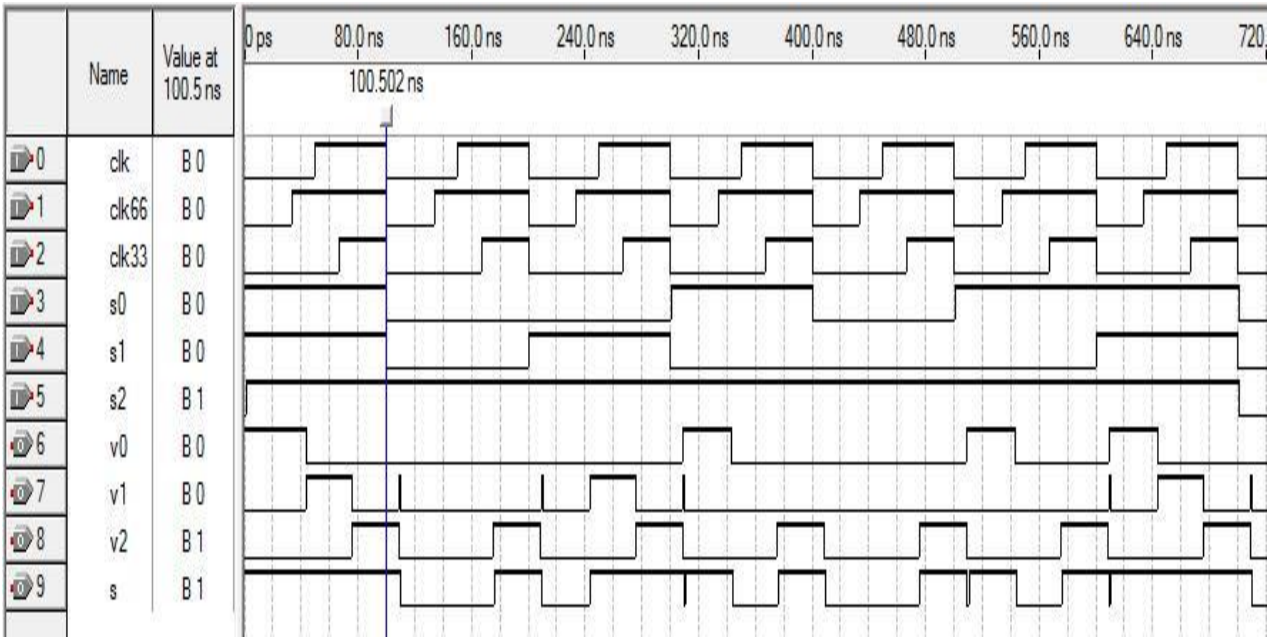


Рисунок 1.16– Перетворення паралельного сигналу в послідовний сигнал

З часової діаграми (рисунок 1.17) стану кодера видно, що при вхідній послідовності біт (1101000) вихідна послідовність буде (111 001 011 101 001 101 111).

Отримано той же результат, який знайдений в таблиці 1.3.

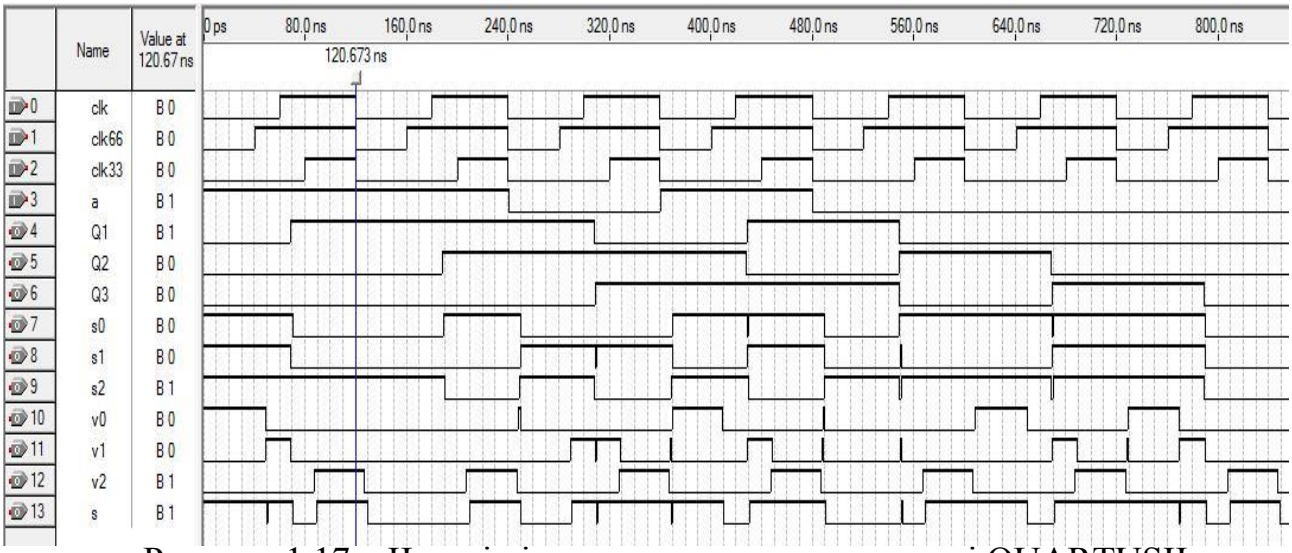


Рисунок 1.17 – Часові діаграми кодера в середовищі QUARTUSII

1.5.6 Згортковий кодер (m=2, k=2, n=3)

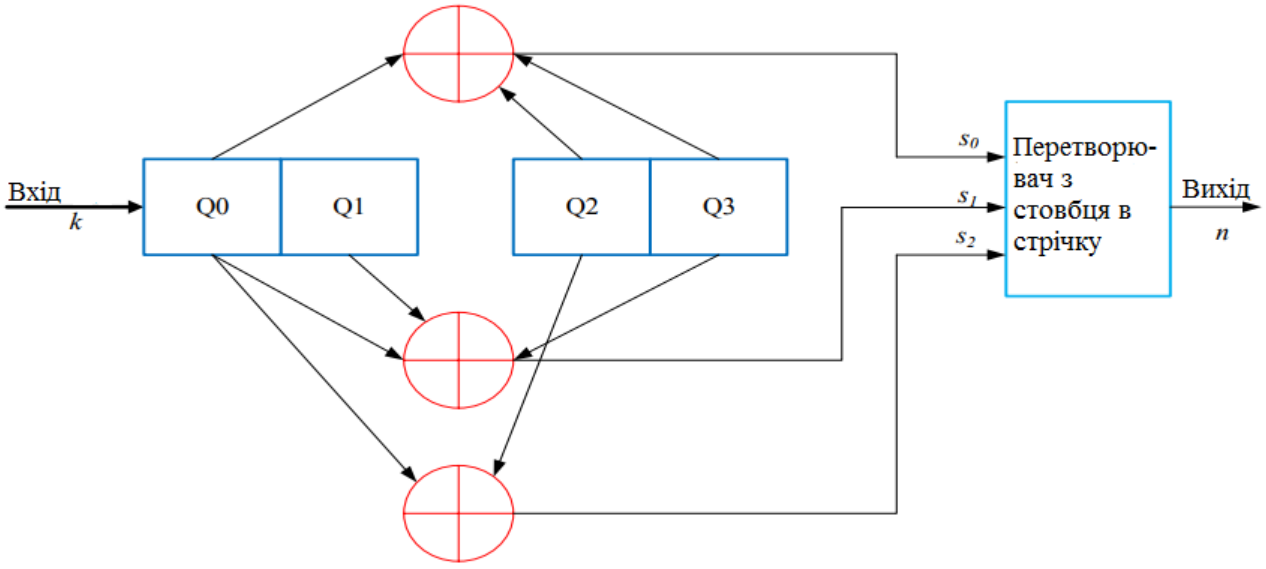
При швидкостях $R = k / n$, де $k > 1$, ситуація стає більш складною. Розглянемо для прикладу згортковий кодер, показаний на рисунку 1.18, де швидкість кодування дорівнює 2/3. У цьому кодері кожен раз два біта надходять на вхід регістрів зсуву, потім суматори за модулем 2 визначають три символи, і на виході генерується три біта [12].

Коефіцієнти породжуючих поліномів представлені в вісімковій формі. Коефіцієнти породжують полиноми в двійковій формі:

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

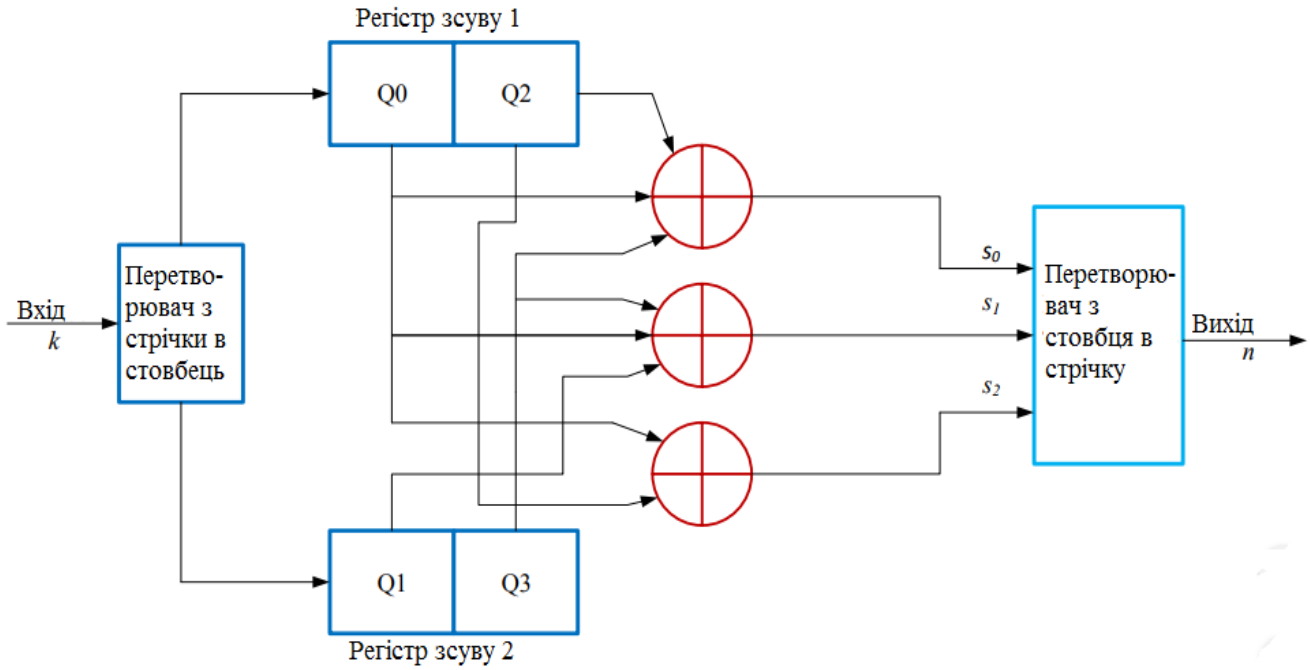
Породжуючі поліноми мають вигляд:

$$\begin{aligned} g_1(x) &= 1 + x^2 + x^3, \\ g_2(x) &= 1 + x + x^3, \\ g_3(x) &= 1 + x^2 \end{aligned} \tag{1.19}$$



а)

Продовження рисунка 1.18



б)

Рисунок 1.18 – Структурні схеми кодера з параметрами m=2, k=2, n=3

Ці дві схеми еквівалентні. Регістри зсуву 1 і 2 (число регістрів одно k) мають по два відділення пам'яті і три суматора по модулю 2 (число суматорів одно n), які формують символи коду відповідно до виду виробляють поліноми. Перетворювач розподіляє вхідні інформаційні символи між регістрами, перетворювач стовбчика в рядок формує кодову послідовність на виході кодера з вихідних символів суматорів [9].

Перші всі можливі два вхідних біта можуть бути 11, 01, 10 або 00. Відповідно, вихідні біти будуть представлені у вигляді 000, 010, 111,101. Коли наступна пара вхідних бітів входить в кодер, перша пара пересувається в наступний осередок. Відповідні вихідні біти залежать від пари бітів, що перемістилися в другий осередок і нової пари вхідних бітів.

Розглянемо на прикладі, як формується вихідна кодова послідовність для вхідного сигналу (1101101100) (таблиця 1.4).

Таблиця 1.4– Формування кодової послідовності

Номера вхідних імпульсів	0	1	2	3	4
Вхідні символи, *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 виході кодера отримаємо кодову послідовність: s = (101 100 111 011 011).

Побудуємо згортковий кодер на рисунку 1.18в середовищі програмованих логічних інтегральних схем QUARTUS II (рисунок 1.19). згортковий кодер на рисунку 1.19 складається з двох запам'ятовуючих ячейок (D-тригерів) і елементів XOR (суматори за модулем 2) і перетворювача (мультиплексора).

Принцип роботи мультиплексора наведено вище [16].

Розроблено методику побудови структурних схем згорткових кодерів до рівня електричних принципових схем і їх моделювання в середовищі QUARTUS II можна реалізовувати на ПЛІС.

Слід відмітити, що запропоновані в даному пункті принципові електричні схеми кодерів середовищі QUARTUSII можуть бути «зашиті» в кристал програмованих логічних інтегральних схем (ПЛІС) та ефективно використовуватися для вирішення задачі кодування загорткових кодів.

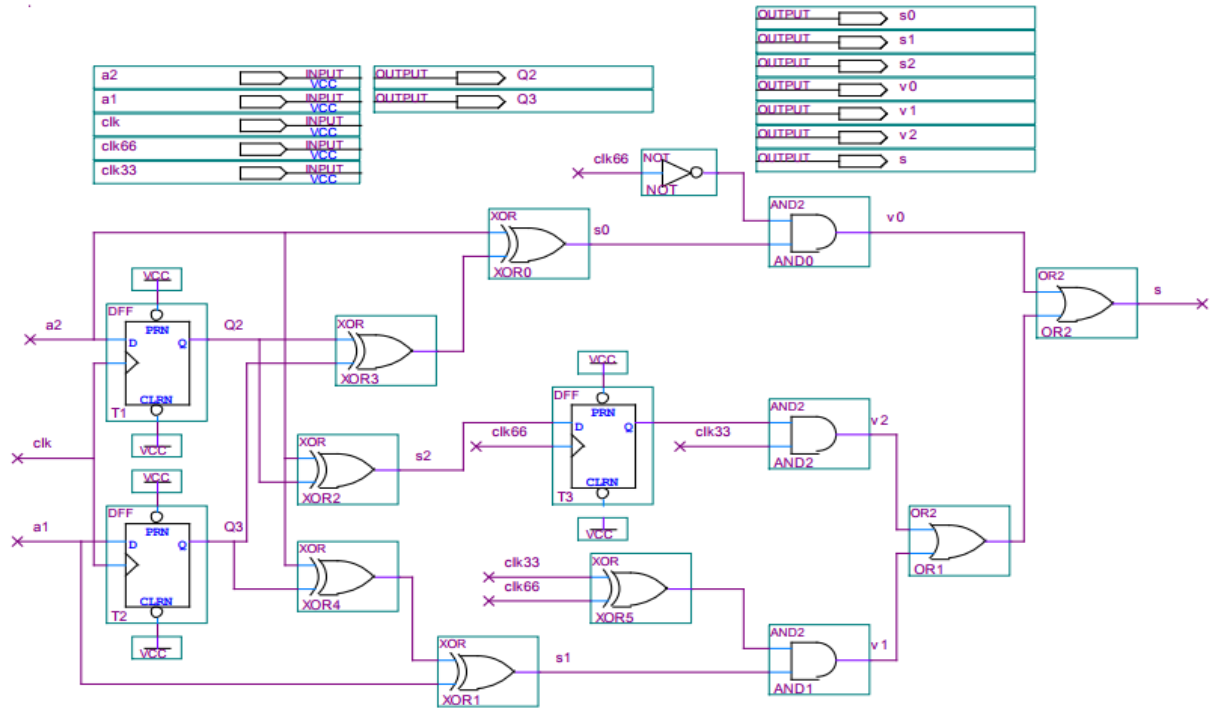


Рисунок 1.17 – Схема кодера в середовищі QUARTUS II

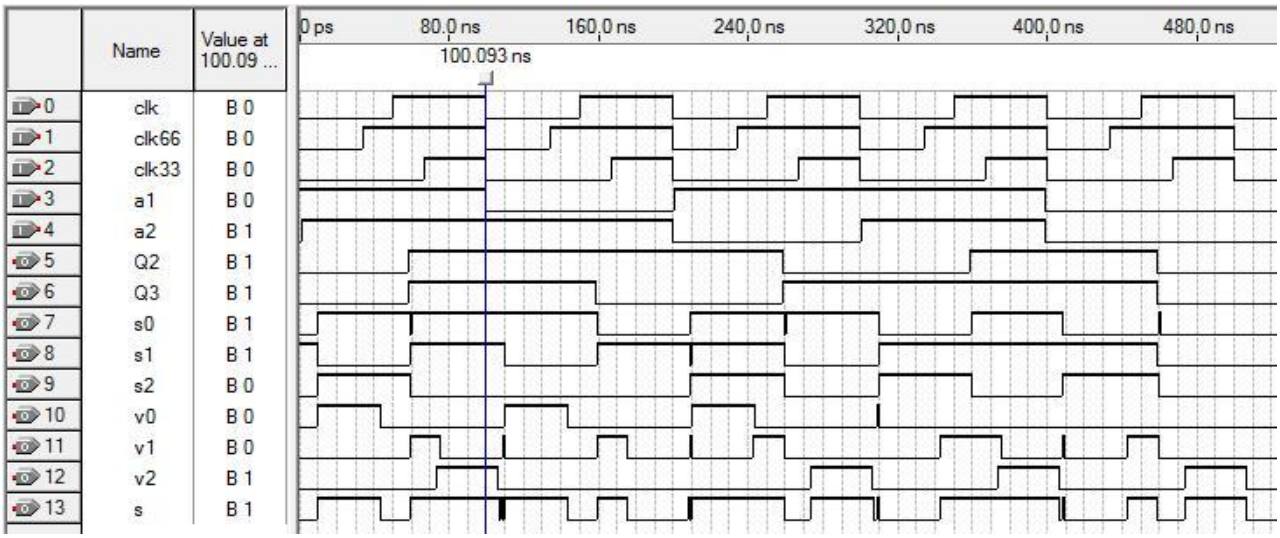


Рисунок 1.18 – Часові діаграми кодера в середовищі QUARTUS II

З часової діаграми (рисунок 1.18) можна помітити, що при вхідній послідовності біт (1101101100) вихідна послідовність буде (101 100 111 011011). Отримано той же результат, який знайдений в таблиці 1.4.

2 ДЕКОДУВАННЯ ЗГОРТКОВИХ КОДІВ

Для згорткових кодів розроблені методи порогового, послідовного декодування і декодування згідно алгоритму Вітербо по максимуму правдоподібності. На практиці широко використовується останній метод, що пояснюється простотою реалізації при невеликих довжинах кодового обмеження і одержуваних виграшем від кодування [1-4].

Для декодування згорткових кодів використовуються: діаграма станів, деревоподібна діаграма, решітчаста діаграма згорткових кодів.

2.1 Діаграма станів кодера

Роботу кодера зручно розглядати на основі не тимчасових діаграм, а так званої діаграми стану [4-5].

Стан кодера згорткових кодів - це вміст m-1крайніх правих розрядів регістра зсуву.

Діаграма станів є спрямований граф і описує всі можливі переходи кодера з одного стану в інший, а також містить символи виходів кодера, які супроводжують ці переходи. У діаграмі має бути 2^{m-1} станів.

Шляхи (гілки) між станами співвідносяться з кодовими словами виходу кодера. З кожного стану (прямокутника) виходять два шляхи, відповідні бітам 0 і 1 на вході кодера.

Для прикладу розглянемо діаграму станів (рисунок 2.1) для кодера на рисунку 1.2. Кодер має чотири стани, де різні стани кодера відзначені також буквами a, b, c, d.

При зображень шляхів, чорною лінією позначається шлях, пов'язаний з нульовим вхідним бітом, а червоною лінією позначають шлях, пов'язаний одиничним вхідним бітом. Стан кодера позначається двома значеннями першої і

другої запам'ятовуючих ячейок. Наприклад, в разі якщо перша осередок зберігає значення 1 (Q1 = 1), а друга - 0 (Q2 = 0), стан кодера дорівнює значенню 10. Іншими словами, можливо чотири різних станів кодера: 00, 01, 10 і 11 [14, 17].

Припустимо, що в деякий момент часу стан кодера рівно 0. Необхідно визначити який дебіт буде сформований в наступний момент часу.

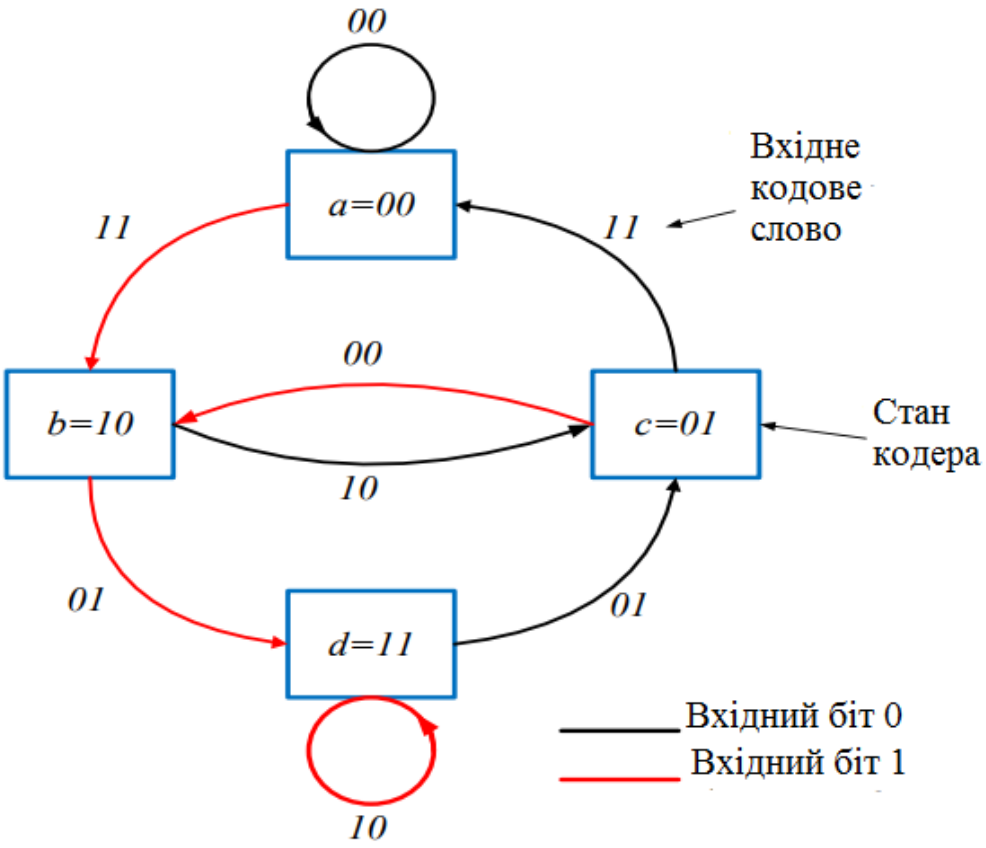


Рисунок 2.1 – Діаграма станів кодера (ступінь кодування 1/2, m= 3)

Відповідь на це питання залежить від значення біта, що надійшов на вхід кодера. У разі якщо на вхід надійде 0, значення наступного стану кодера також буде рівним 00, якщо ж надійде 1, стан кодера після зсуву буде позначено як 10. Значення формованих пріетомдібітоврассчітивается за формулами:

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

Якщо на вхід кодера надходить 0, то буде сформований дібіт 00:

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

Якщо ж на вхід надходить 1, то формується дібіт 11:

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

Наведені міркування зручно представити наочно за допомогою діаграми станів (рисунок 2.1).

Уявімо діаграму станів кодера у вигляді таблиці (таблица 2.1).

Таблицая 2.1 – Таблиця переходів станів кодера (R=1/2, m = 3)

Стан	Вхідний біт	Вміст регістра зсуву	Вихідна комбінація
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

Діаграма станів для кодера на рисунку 1.4 показана на рисунку 2.2. кодер має 2^{m-1}=8 різні стани і відзначені буквами a, b, c, d, e, f, g, h.

Всього можливо вісім різних станів кодера: 000, 001, 010, 011, 100, 101, 110 і 111.

Так як в кодер вводиться по одному біту, то в кожний стан входять і з кожного стану виходять по дві гілки.

Два біта, показаних на кожній гілці діаграми станів, представляють вихідні біти. Червона лінія означає, що вхідний біт 1, в той час як чорна лінія вказує, що вхідний біт 0.

Припустимо, що в деякий момент часу стан кодера дорівнює 0. Якщо на вхід кодера надійде 0, то наступний стан кодера також буде 000, якщо ж навпаки надійде 1, наступний стан кодера після зсуву буде 100. Значення формуються при цьому дебіт розраховується за формулами:

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

Якщо на вхід кодера надходить 0, то на виході сформований дебіт 00, ($0 \oplus 0 \oplus 0 \oplus 0 = 0,\ 0 \oplus 0 \oplus 0 = 0$), якщо ж на вхід надходить 1, то формується дебіт 11 ($1 \oplus 0 \oplus 0 \oplus 0 = 1, 1 \oplus 0 \oplus 0 = 1$).

Наведені міркування зручно представити за допомогою діаграми станів (рисунок 2.2).

Уявімо діаграму станів кодера у вигляді таблиці (таблиця 2.2).

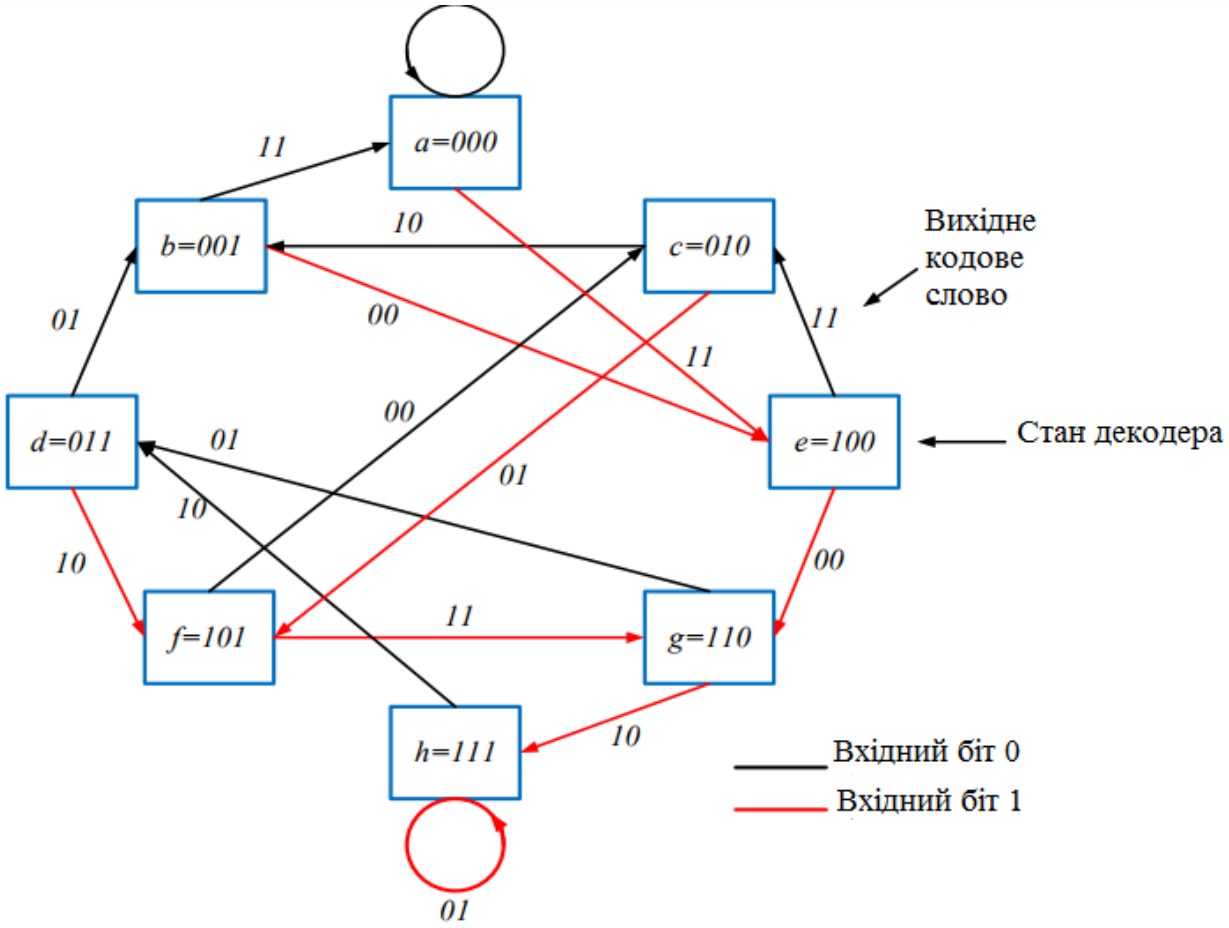


Рисунок 2.2 – Діаграма станів кодера (ступінь кодування 1/2, m = 4)

Таблиця 2.2 – Таблиця переходів станів кодера (R=1/2, m = 4)

Стан	Вхідний біт	Вміст регістра зсуву	Вихідна комбінація
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

Діаграма станів для кодера на рисунку 1.13 показана на рисунку 2.3. кодер має 2m-1 = 8 різніх станів, зазначеним літерами a, b, c, d, e, f, g, h.

Всього можливо вісім різніх станів кодера: 000, 001, 010, 011, 100, 101, 110 і 111.

Ці стани заносяться прямокутники діаграми станів. Кожні 3 біта, показаних на гілках діаграми станів, представляють біти на виході.

Червона лінія означає, що вхідний біт 1, в той час як чорна лінія вказує, що вхідний біт 0.

Нехай в деякий момент часу стан кодера 000. Якщо на вхід кодера надійде 0, то наступний стан кодера також буде 000, якщо ж надійде 1, то наступний стан (тобто після зсуву) буде 100.

Значення формуються при цьому трьох бітів розраховується за формулами:

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

Якщо на вхід кодера надходить 0, то на виході кодера буде сформований трійковий біт 000, ($0 \oplus 0 \oplus 0 \oplus 0 = 0$, $0 \oplus 0 \oplus 0 = 0$, $0 \oplus 0 \oplus 0 = 0$).

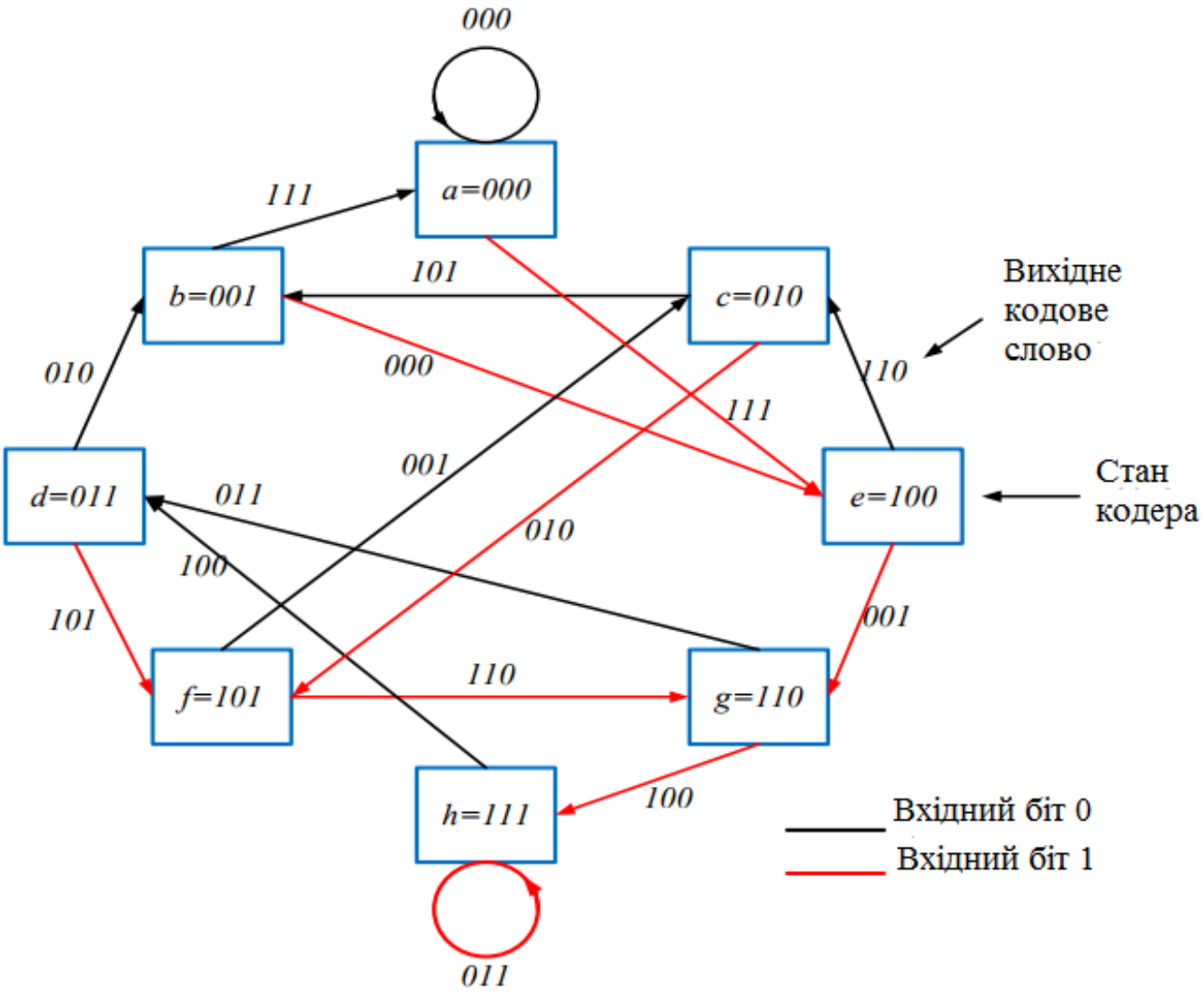


Рисунок 2.3 – Діаграма станів кодера (ступінь кодування 1/3, m = 4)

Якщо ж на вхід надходить 1, то на виході кодера формується трійковий біт 111 ($1 \oplus 0 \oplus 0 \oplus 0 = 1$, $1 \oplus 0 \oplus 0 = 1$, $1 \oplus 0 \oplus 0 = 1$).

Наведені міркування зручно представити за допомогою діаграми станів (рисунок 2.3).

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

Стан	Вхідний біт	Вміст регістра зсуву	Вихідна комбінація
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

Діаграма станів для кодера представлена на рисунку 2.3 згідно рисунку 2.4. Кодер має чотири різних станів, зазначених літерами a, b, c, d.

У кодер два біта надходять на вхід регістрів зсуву щораз. Перші всі можливі два вхідних біта можуть бути 00, 01, 10 или 11.

Чорна лінія вказує на те, що вхідний біт 00, червона лінія показує, що вхідний біт 10, зелена лінія позначає, що вхідний біт 01, в той час як синя лінія означає, що вхідний біт 11. Нехай в деякий момент часу стан кодера рівний 00.

Якщо на вхід кодера надійде 00, то наступний стан кодера також буде 00, якщо ж надійде 1, то наступний стан (тобто після зсуву) буде 10.

Отже, застосування даного підходу при кодуванні дозволяє визначати перхідні стани кодера, що суттєво полегшує процес та зменшує обчислювальну складність.

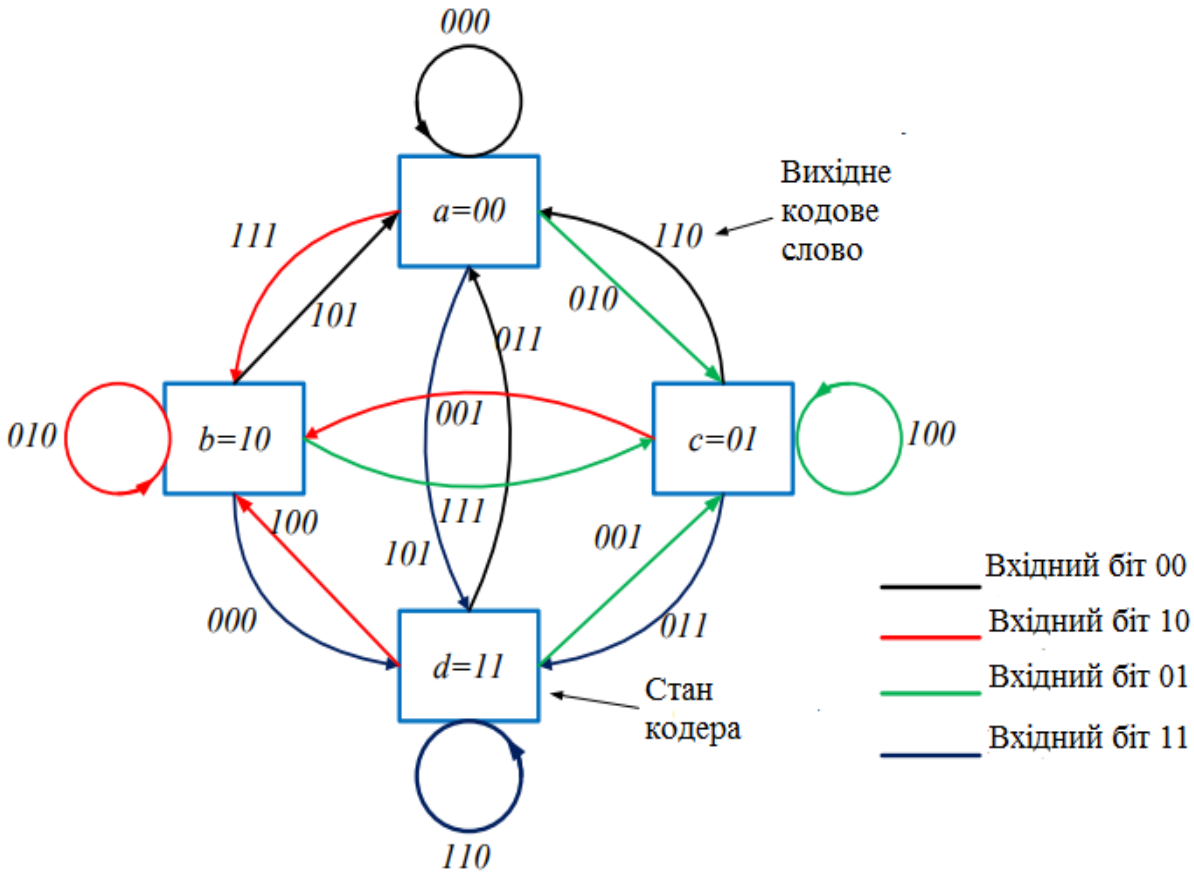


Рисунок 2.4 – Діаграма станів кодера (ступінь кодування 2/3, m=2)

Значення формуються при цьому дебіт розраховується за формулами:

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

Якщо на вхід кодера надходить 00, то на виході кодера буде сформований три біти 000, ($0 \oplus 0 \oplus 0 = 0,\ 0 \oplus 0 \oplus 0 = 0,\ 0 \oplus 0 = 0$), якщо ж на вхід надходить 10, то формується три біти 111 ($1 \oplus 0 \oplus 0 = 1,\ 1 \oplus 0 \oplus 0 = 1,\ 1 \oplus 0 = 1$), якщо ж на вхід надходить 01, то формується три біти 010 ($0 \oplus 0 \oplus 0 = 0,\ 0 \oplus 1 \oplus 0 = 1,\ 0 \oplus 0 = 0$), якщо ж на вхід надходить 11, то формується три біти 101 ($1 \oplus 0 \oplus 0 = 1,\ 1 \oplus 1 \oplus 0 = 0,\ 1 \oplus 0 = 1$).

Наведені міркування зручно представити за допомогою діаграми станів (рисунок 2.4).

Уявімо діаграма станів кодера у вигляді таблиці (таблиця 2.4).

Таблиця 2.4 - Таблиця переходів станів кодера (R=2/3, m=2)

Стан	Вхідний біт	Вміст верхнього регістра зсуву	Вміст нижнього регістра зсуву	Вихідна комбінація
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 Деревоподібна діаграма

В додатку А показана деревоподібна діаграма для згорткового кодера (рисунок 1.2). Деревоподібна діаграма (treediagram) додає до діаграми стану часову змінну.

Вертикальні лінії діаграми (додаток А) названі ребрами дерева, горизонтальні лінії - гілками [3-5].

Залежно від того, яке значення приймає вхідний сигнал, рух по ребрах може бути: при надходженні на вхід нуля відбувається рух по верхньому ребру, при надходженні одиниці по нижньому ребру. Певна гілка відповідає певному кодовому слову на виході кодера.

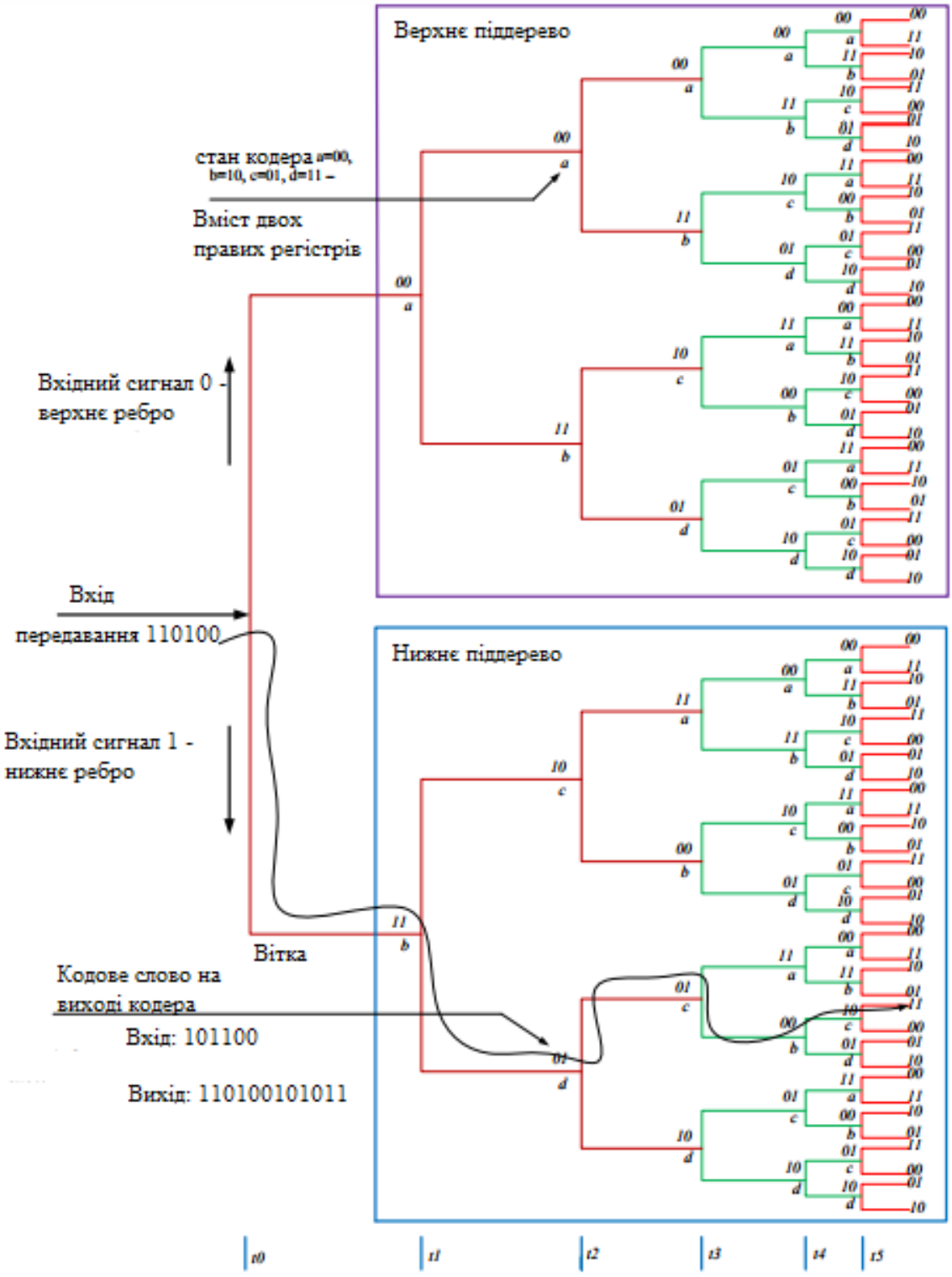


Рисунок 2.5 – Деревоподібна діаграма кодера (ступінь кодування 1/2, *m*=3)

Вміст 00, b = 10, c = 01, d = 11. Станом крайнього лівого регістра визначається перехід по ребрах дерева.

Далі описаний процес формування вихідної кодової послідовності символів для вхідного сигналу (110100).

Спочатку стан регістрів кодера 00. Коли надходить сигнал рівному «1» рух йде по нижньому ребру дерева, на вихід надходить 11. Потім на вхід подається другий біт 1, йде рух по нижньому ребру, на вихід надходить 01.

Наступний вхідний біт 0 тягне до переходу по верхньому ребру, що дорівнює вихідному сигналу 01. Далі на вхід приходить 1, відбувається перехід по нижньому ребру, на виході кодера утворюється сигнал 00. Далі на вхід надходить 0, відбувається перехід по верхньому ребру, на виході кодера формується сигнал 10.

Останній вхідний біт 0 призводить до переходу по верхньому ребру, на вихід надходить 11. Таким чином, вхідний послідовності (110100) відповідає вихідна послідовність (11 01 01 00 10 11).

З ростом довжини вхідної послідовності число можливих шляхів на деревовидної діаграмі різко зростає, що накладає обмеження на її застосування [18-20].

Таким чином, виходить що починаючи з 3-го рівня гілок верхнє і нижнє поіддерево повністю повторюють один одного.

Період повторення деревовидної діаграми визначається кількістю регістрів кодера (розміром регістру зсуву).

В даному випадку все вхідні символи кодера використовуються для створення лише 3 вихідних кодових слів, потім вони видаляються з пам'яті.

2.3 Гратчаста діаграма

Гратчаста діаграма (решітка) є розгорткою діаграми станів у часі. На решітці стани показані вузлами, а переходи - з'єднують їх лініями. Після кожного

переходу з одного стану в інший відбувається зміщення на один крок вправо [1, 3-5].

Наприклад, решітчаста діаграма для кодера (рисунок 1.2) показана на рисунку 2.6, а для кодера 1.13 показана на рисунку 2.7.

На даному рисунку чорні лінії (гілки) відповідають переходам, що відбувається при отриманні інформаційного символу 0, а червоні лінії (гілки) - при інформаційному символі 1.

Як видно з гратчастої діаграми, її структура в кінці «перехідного процесу» в кодері стає повторюваною [8-9].

На рисунку 2.6 структура гратчастої діаграми повторюється після 3-го такту роботи кодера, так як під час підходу до кодера четвертого інформаційного символу перший символ покидає регістр зсуву і далі не впливає на створення кодових символів.

На рисунку 2.7 повторюваність структури гратчастої діаграми буде можлива після 4-го такту роботи кодера, так як при вступі до кодера наступного інформаційного символу 1-ий символ покидає регістр зсуву і потім не використовується для отримання кодових символів.

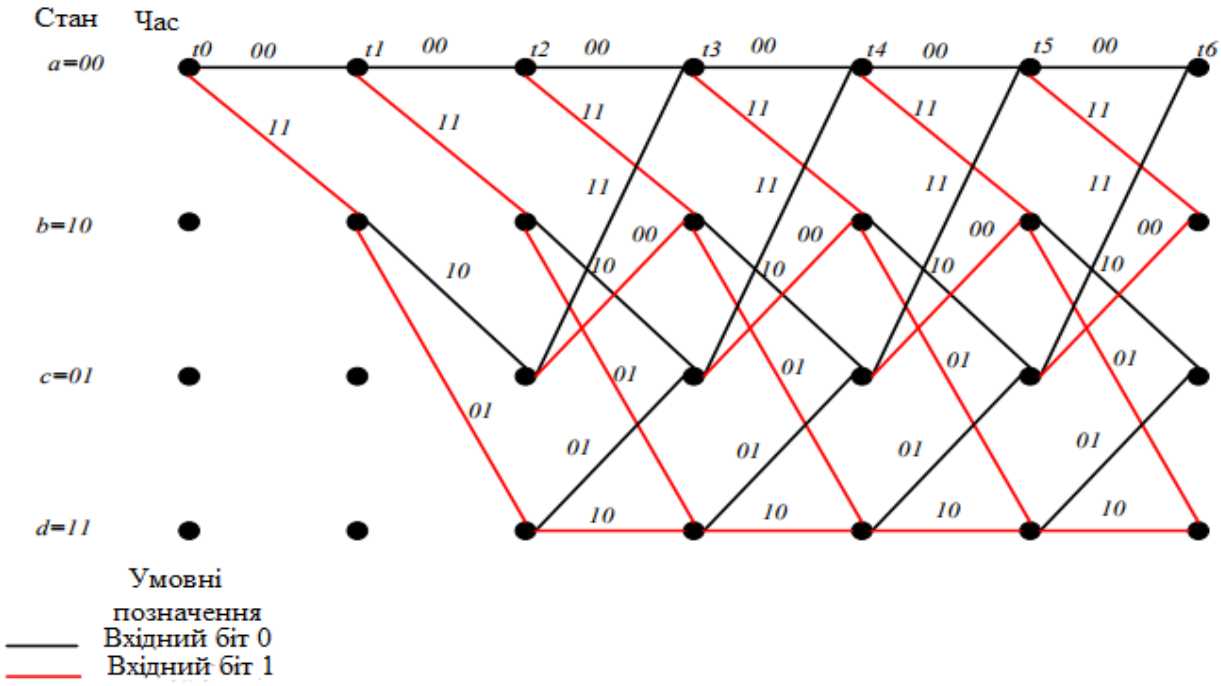


Рисунок 2.6 – Гратчаста діаграма кодера (ступінь кодування 1/2, *m*=3)

Зручність гратчастого представлення в тому, що зі зростанням кількості вхідних символів число вершин в решітці не збільшується, а залишається рівним 2^{m-1}, де m– число ячейок регістра зсуву.

Гратчаста діаграма дає можливість визначити всі дозволені шляхи, по яких кодер може робити кодування.

Наприклад, під час надходження на вхід кодера на рисунку 1.3 послідовності (110100) шлях по решітці дасть можливість отримати вихідну послідовність (11 01 01 00 10 11).

При надходженні на вхід кодера на рисунку 1.13 послідовності (1101000) шлях по решітці дасть можливість отримати вихідну послідовність (111 001 011 101 001 101 111).

Кожна інформаційна послідовність символів має відповідний унікальний шлях (траєкторію) на діаграмі.

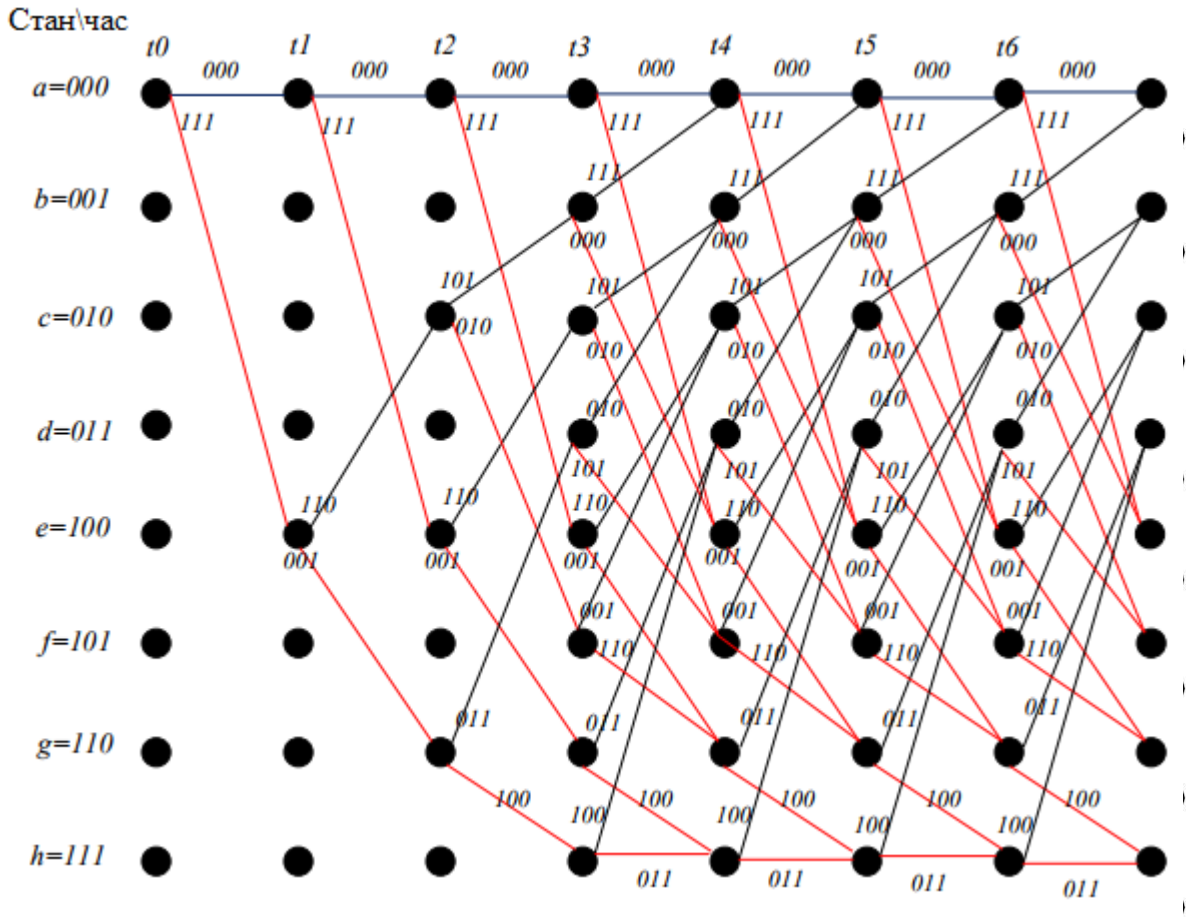


Рисунок 2.7 – Гратчаста діаграма кодера (ступінь кодування 1/3, m= 4)

Кодова послідовність на виході формується за допомогою зчитування кодових комбінацій над гілками камер при відстеженні даної траєкторії.

2.4 Застосування декодування методом Вітербо

Оптимальне декодування згорткових кодів здійснюється за допомогою алгоритму Вітербо. Алгоритм декодування Вітербо по ефективності близький до алгоритму максимуму правдоподібності. На кожному такті роботи декодера Вітербо обчислюються метрики ребер як відстані між прийнятим кодовим словом і кожним із шляхів, що входять в кожний стан регістра, і метрики станів шляхів як суми, що зберігаються метрик попередніх станів і метрик ребер. Після порівняння метрик станів зберігаються метрики виживши шляхів з найменшою метрикою.

Згорткові коди безупинні і можуть бути охарактеризовані мінімальною відстанню, що визначаються довжиною початкових сегментів кодових послідовностей. Число ячейок на приймальній стороні в декодері визначається числом символів в прийнятій для обробки довжині сегмента L.

Це число символів, що зберігається в пам'яті декодера з метою подальшої обробки прийнятої кодової послідовності, називають глибиною декодування.

У разі, якщо необхідно виявлення та виправлення максимальної кількості конфігурацій помилок, то найчастіше збільшення глибини декодування покращує характеристики, однак в подальшому відбувається насичення [4].

Мінімальна глибина декодування повинна бути рівною довжині кодового обмеження l_2 . Однак вона в більшості випадків в кілька разів більше l_2.

Найменша відстань Хемінга для різних пар кодових слів називається L-м мінімальною вільною відстанню згортального коду d_L. За умови, коли L однакове з l_2, тоді L називається мінімальною вільною відстанню і як в блокових кодах позначається d_{min}.

Значення *dL* згорткового коду визначається за допомогою використання діаграми станів, або з допомогою решітчатої діаграми для відповідного кодера.

Так як згорткове кодування відноситься до лінійних, шлях з вагою рівним нулю (нульовий шлях) повинен обов'язково бути присутнім серед різних шляхів на решітчатій діаграмі коду. Нульовим шляхом називають шлях, в якому послідовність кодових символів складається тільки з «0». Таким чином, мінімальна вільна відстань згорткового коду дорівнює мінімальному числу одиниць, інакше кажучи, мінімальній вазі шляхів, що розходяться і зливаються з нульовим шляхом. Гратчаста діаграма коду може бути використана для визначення *dL*, якщо для кожної її гілки записати вагу відповідних кодових символів на виході кодера, після чого підрахувати вагу шляхів, що розходяться і зливаються з нульовим шляхом.

Коректуюча здатність згорткових кодів оцінюється мінімальною відстанню між кодовими послідовностями, що називається вільним відстанню $d_{св}$. [10].

Далі розглянемо принцип роботи алгоритму Вітербо для прийому з каналу *i*-того *n*-символьного групи послідовності s^*. Розглянуті шляхи можуть проходити через 2^{m-1} вузлів (станів) гратчастої діаграми (де *m* - число ячейок регістра зсуву), і для кожного з них обчислюється відстань від прийнятої послідовності, якf далі буде називатися метрикою.

Потім на *i*-му кроці необхідно [8]:

1. Розрахувати відстань Хемінга між вхідною-символьною групою і різними гілками гратчастої діаграми.

Оскільки з кожного з 2^{m-1} вузлів виходить по дві гілки, то необхідно обчислити 2^m таких відстаней.

2. Відстані Хемінга для всіх гілок додаються до метрик шляхів, з яких вони виходять. У підсумку виходять 2^m можливих шляхів, що ведуть в 2^{m-1} станів.

3. Для кожного з 2^{m-1} станів необхідно порівняти метрики 2-х вхідних в нього шляхів і шлях з меншою метрикою, який знаходиться на мінімальній відстані від вхідної послідовності. Шлях з найбільшою метрикою не враховується в подальших обчисленнях.

4. Запам'ятати всі 2^{m-1} вижившi шляхи разом з їх метриками і перейти до виконання (i + 1) -го кроку.

Алгоритм Вітербо найлегше зрозуміти, розглядаючи в якості прикладу ґратчасту діаграму для згорткового кодера на рисунку 1.3. Решітчата діаграма кодера показана на рисунку 2.6. Декодування за алгоритмом Вітербо спотвореної послідовності з помилкою першої кратності.

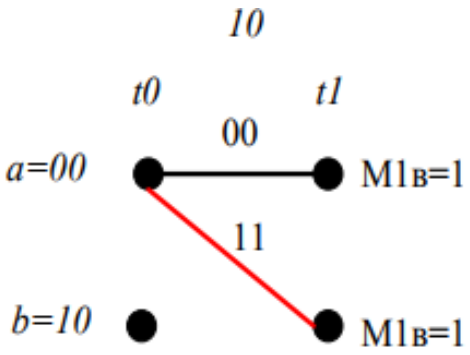
Нехай передається кодове слово s = (11 01 01 00 10 11), а в каналі відбулася одноразова помилка, так, что прийнята послідовність має вигляд s^* = (10 01 01 00 10 11).

Обчислимо відстань Хемінга між прийнятою 1-ою символьною групою (10) і різними гілками (t_0, t_1) гратчастої діаграми (рисунок 2.8, а). На кроці 1 зі стану 00 виходять наступні шляхи: в стан 00 з відстанню Хемінгом до прийнятої з каналу першої символьної послідовності дорівнює 1, і в стан 10 з метрикою 1 (рисунок 2.8, а).

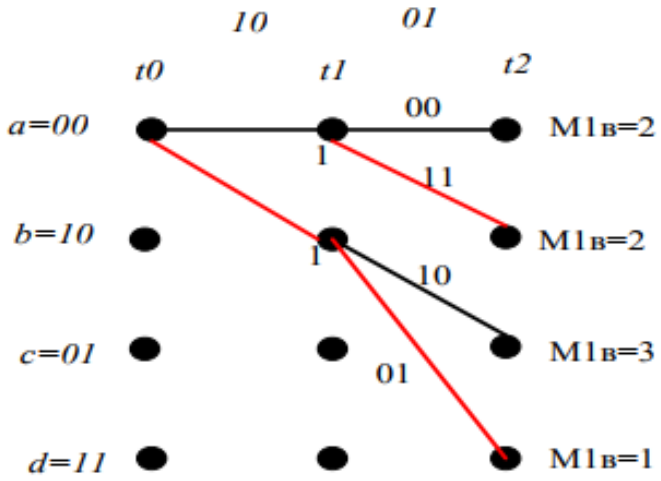
На кроці 2 формуються вже чотири шляхи, причому метрики станів 00, 10, 01 і 11 стають рівними 2,2, 3 і 1 відповідно (рисунок 2.8, б).

Наступний 3-ий етап роботи є критичним для розуміння всього алгоритму. Розглянемо стан 00, до якого можна прийти по двох різних шляхах: зі стану 00 і зі стану 01. Перший шлях буде мати метрику 3, так як метрика гілки (00-00) дорівнює «1» і попереднє значення метрики шляху дорівнює двом, а другий - чотирьом, так як метрика гілки (01-00) дорівнює трьом, і попередня метрика шляху дорівнює одиниці (рисунок 2.8, в). Порівняння метрик цих 2-х шляхів дає можливість вибрати найкращий, в даному випадку це шлях зі стану 00, так як він має меншу метрику.

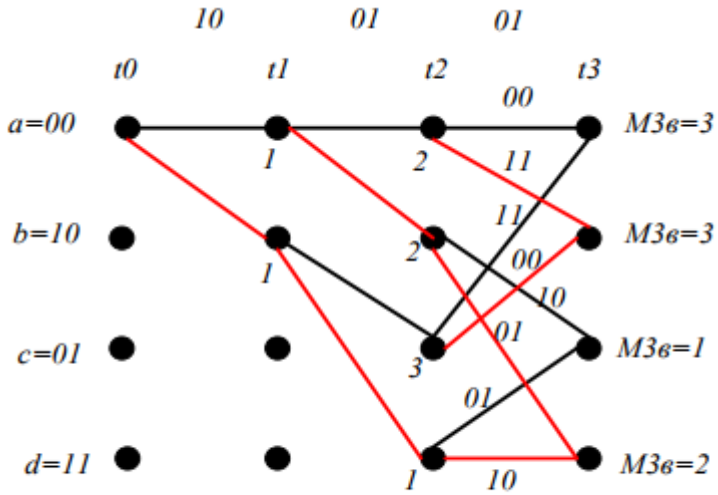
Таким же способом вибираються ті, що вижили шляхи для інших вузлів решітки. Кроки 4 і 5 повністю аналогічні кроку 3 (рисунок 2.8, г, д). Особливістю четвертого кроку є те, що метрики конкуруючих шляхів, що входять в один стан (стан 00), рівні. Найпростішим способом вирішення цієї невизначеності є випадковий вибору вижившого шляху.


10
t0
t1
00
a=00
M1в=1
11
b=10
M1в=1

а)

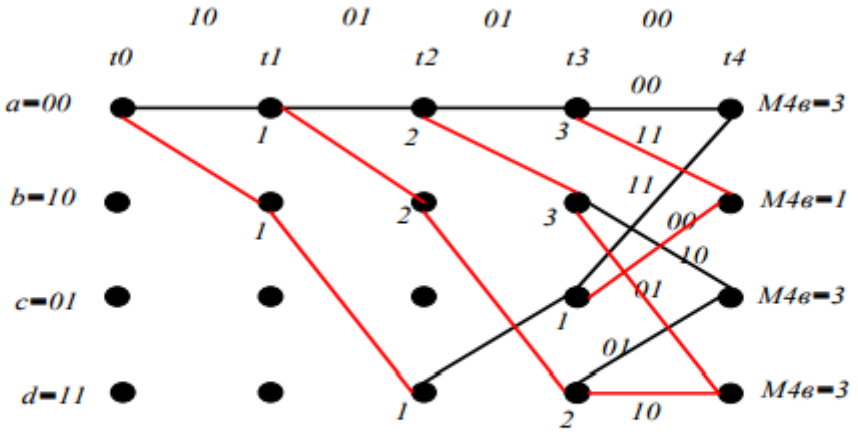

10
01
t0
t1
t2
00
a=00
M1в=2
1
11
b=10
M1в=2
1
10
c=01
M1в=3
01
d=11
M1в=1

б)

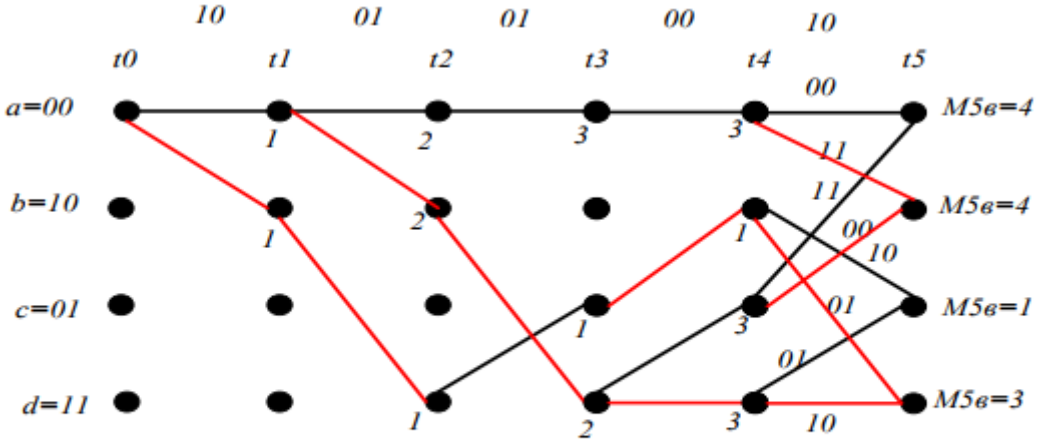

10
01
01
t0
t1
t2
t3
00
a=00
M3в=3
1
2
11
11
b=10
M3в=3
2
00
1
10
01
c=01
M3в=1
3
01
d=11
M3в=2
1
10

в)

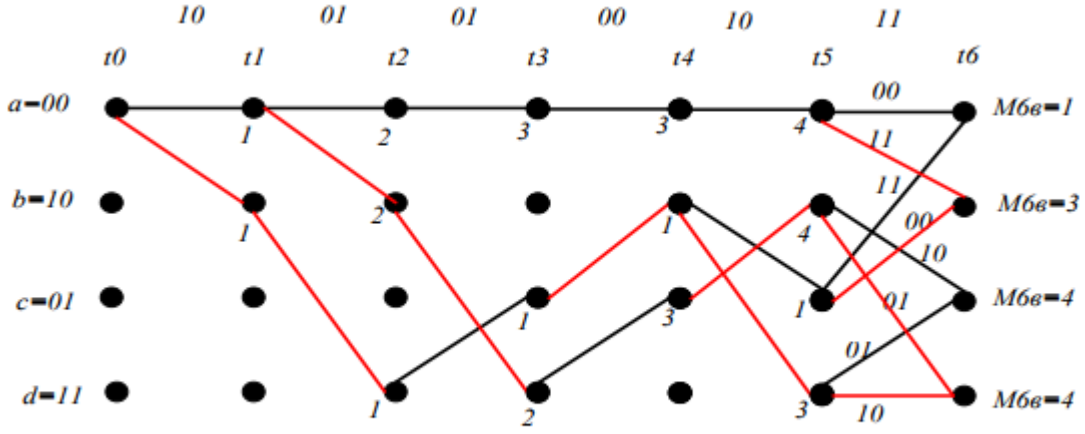
Продовження рисунка 2.8



г)

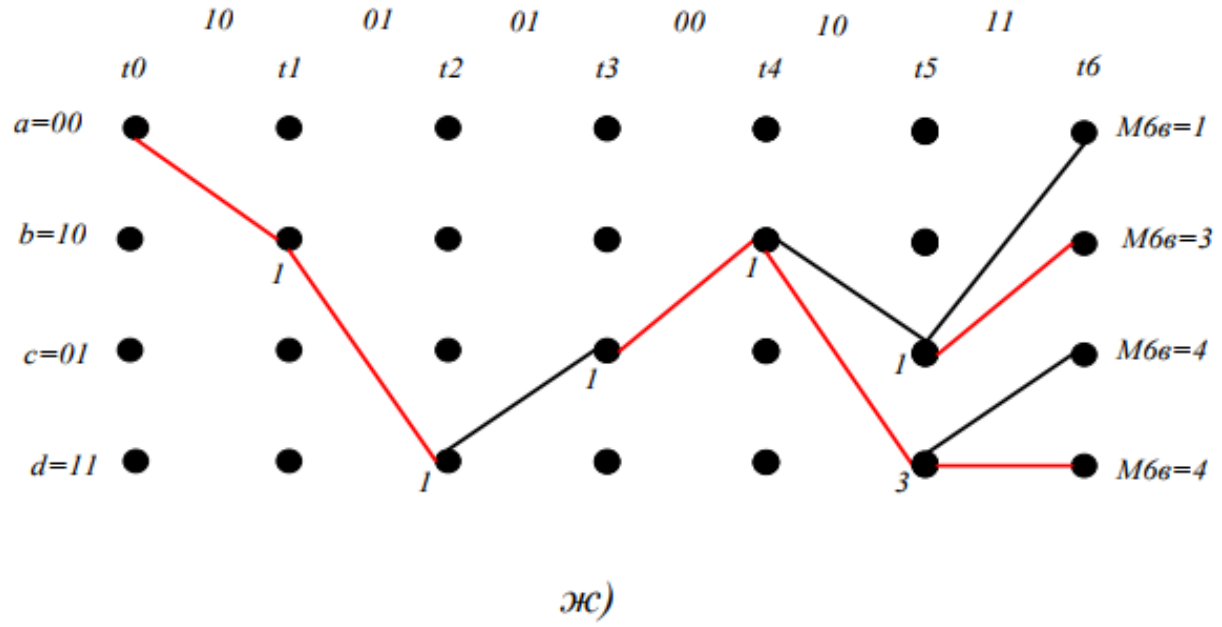


д)



е)

Продовження рисунка 2.8



ж)

Рисунок 2.8 – Гратчаста діаграма кодера (ступінь кодування 1/2, *m*=3)

Після виконання шостого кроку роботи склалася характерна ситуація, коли початкові частини (перші п'ять гілок) всіх тих, хто вижив шляхів злилися і тепер можна приймати рішення про перших п'яти інформаційних бітах (див. рисунок 2.8, е, ж).

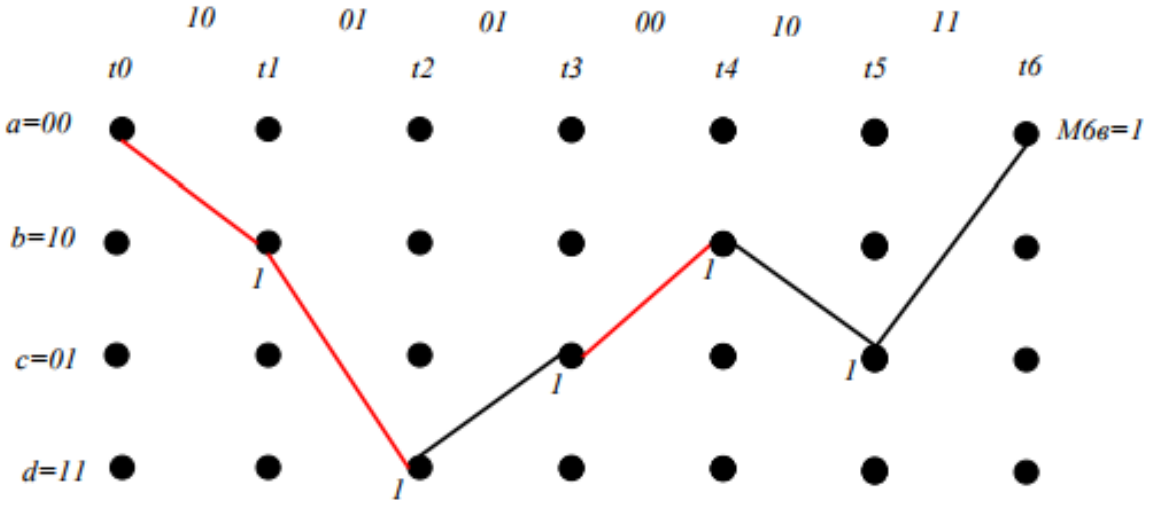


Рисунок 2.9 – Доступний шлях

Початкова частина тих, що вижили шляхів визначає інформаційну послідовність (110100), тобто одна канальна помилка була виправлена (рисунок 2.9).

3 РЕАЛІЗАЦІЯ КОДУВАННЯ І ДЕКОДУВАННЯ ЗГОРТКОВИХ КОДІВ

3.1 Програмування кодування і декодування згорткових кодів

Для дослідження властивостей і перевірки показників якості роботи довічних згорткових кодів була створена модель цифрового каналу зв'язку в середовищі MATLAB. На ДП.КСМ.11785/14.00.00.000.С.1 представлена узагальнена модель цифрової системи зв'язку. Для моделювання модулятора, каналу зв'язку та згорткового кодера використовувалися стандартні об'єкти і функції MATLAB [23-24]. Як критерію якості коду використовувалася залежність ймовірності бітової помилки від співвідношення сигнал / шум [22].

Декодування двійкових згорткових кодів проводилося за допомогою алгоритму Вітербо.

У даній моделі інформаційна послідовність a_i, що задається випадковим чином, кодувалася за допомогою згорткового кодера. Потім кодове слово s_i модулювати за допомогою двійковій фазової модуляції (BPSK), тобто кожен відлік вихідного сигналу представлявся як 1 або -1. До сигналу u_i додавалася випадкова величина n_i (адитивний білий гаусовський шум). Далі прийняте повідомлення демодулювати і декодувати, а на виході системи виходила послідовність символів ... a_i*, ..., a_1*, a_0* .Ця послідовність може відрізнятися від вхідної послідовності ... a_i, ..., a_1, a_0 через наявність помилкових символів [21].

Розглянемо процес кодування і декодування по Вітербо в цій системі, застосовуючи програмування в середовищі MATLAB [23-24].

Нехай на передавальній стороні інформаційна послідовність довжиною N біт надходить на вхід згорткового кодера зі швидкістю кодування 1/2 і має число ячейок регістра зсуву, рівне трьом. На приймальній стороні використовується декодер Вітербо.

Блок-схема програми згорткового кодера в середовищі MATLAB показана на рисунку 3.1.

На вхід кодера надходять N інформаційних символів. Спочатку осередки регістру зсуву розташовуються в нульовому стані. Коли на вхід кодера надходить перший інформаційний символ, він займає місце першого осередку регістра зсуву. Вміст першого осередку регістра зсуву займає місце другого осередку. Вміст другого осередку регістра зсуву займає місце третього осередку.

Суматори по модулю два обчислюють вихідне кодове слово і це кодове слово з'являється на виході кодера. На вхід кодера надходить другий інформаційний символ.

Зрушення в регістрі ми здійснюємо за допомогою циклу. Індекс i змінюється від 2 до N. Умова: якщо i меньше або дорівнює від N, тоді місце першого осередку регістра зсуву займає i-ий інформаційний символ.

Далі суматори за модулем два обчислюють вихідне кодове слово і це кодове слово з'являється на виході кодера.

Цикл повторюється до тих пір, поки на вхід кодера надходить останній інформаційний символ. Якщо $i > N$, тоді кодування буде завершено.

Блок-схема програми згорткового декодера за алгоритмом Вітербо в середовищі MATLAB показана на ДП.КСМ.11785/14.00.00.001.С.1.

На вхід декодера надходить кодуюча послідовність. Гратчаста діаграма декодера має 4 стани: a = 00, b = 10, c = 01, d = 11 (рисунок 2.8).

Сукупність Вузол =[4, N+1] визначає вертикальні і горизонтальні вузли гратчастої діаграми. Метрика =[8, N] визначає число метрики. Початковий стан 00 (Вузол (1,1)). Індекс i змінюється від 1 до N. Знайдемо вісім метрик на i-тому такті.

Далі вибираємо мінімальну метрику, що виходить із кожного вузла. Цей процес повторюється до тих пір, поки i не більше N. Коли i більше N, процес завершується.

Далі знайдемо мінімальну метрику на останньому такті роботи декодера (N+1 такт) і за допомогою файл-функції «prev_stage» будемо відновлювати декодування послідовності.

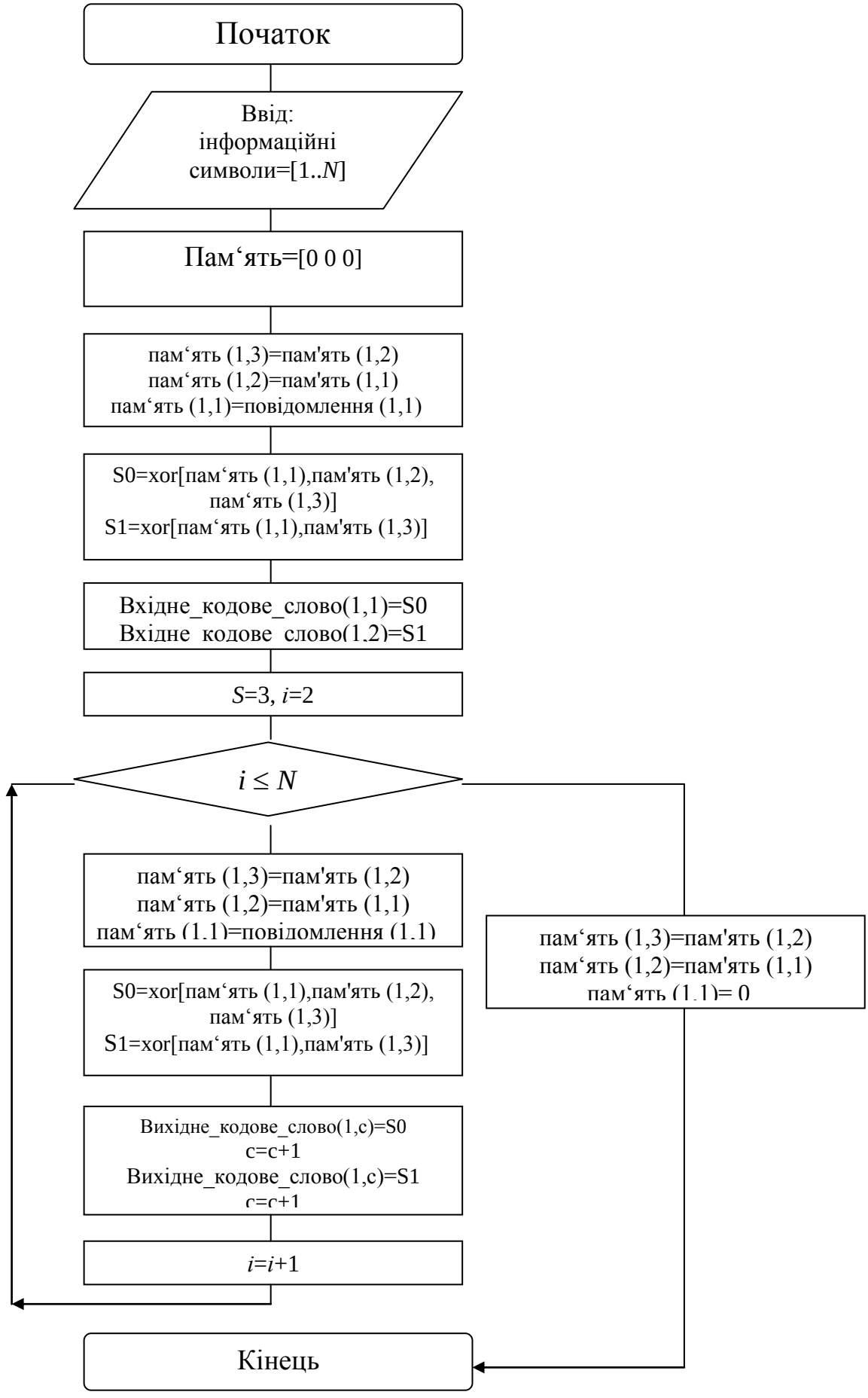


Рисунок 3.1 – Блок-схема програми згорткового кодера в середовищі MATLAB.

Принцип роботи файл-функції «prev_stage». Файл-функція prev_stage знає останню мінімальну метрику і то, на якому вузлі вона знаходиться.

На цей вузол є два шляхи. Якщо вузол - перший стан, тоді в вузол входять шляхи (вузол (1) + метрика (1), вузол (3) + метрика (5)). Якщо метрика шляху (вузол (1) + метрика (1)) менше або дорівнює метрики шляху (вузол (3) + метрика (5)), тоді перекодовується біт 0 і стан 1, в іншому випадку, перекодовується біт 0 і стан 3).

Якщо вузол - другий стан, тоді в вузол входять шляхи (вузол (1) + метрика (2), вузол (3) + метрика (6)). Якщо метрика шляху (вузол (1) + метрика (2)) менше або дорівнює метрики шляху (вузол (3) + метрика (6)), тоді перекодовується біт 1 і стан 1, в іншому випадку, перекодовується біт 1 і стан 3).

Якщо вузол - третій стан, тоді в вузол входять шляхи (вузол (2) + метрика (3), вузол (4) + метрика (7)). Якщо метрика шляху (вузол (2) + метрика (3)) менше або дорівнює метрики шляху (вузол (4) + метрика (7)), тоді перекодовується біт 0 і стан 2, в іншому випадку, перекодовується біт 0 і стан 4.

Якщо вузол - четвертий стан, тоді в вузол входять шляхи (вузол (2) + метрика (4), вузол (4) + метрика (8)). Якщо метрика шляху (вузол (2) + метрика (4)) менше або дорівнює метрики шляху (вузол (4) + метрика (8)), тоді перекодовується біт 0 і стан 2, в іншому випадку, перекодовується біт 0 і стан 4.

За допомогою блок схеми додульного кодера і декодера в середовищі MATLAB. Моделювання згорткового кодера показана в додатку В.

Далі використовуємо цей модель для дослідження завадостійкості двійкових згорткових кодів.

3.2 Дослідження ймовірності бітової помилки

На підставі порівняння декодованої інформаційної послідовності a_i* і переданої інформаційної послідовності a_i складалася статистика появи помилок.

Обчислення точних виразів для ймовірностей бітових помилок - це дуже складне завдання навіть для найпростіших кодів (по Вітербо).

Було проведено порівняння кодів зі швидкістю 1/2, 1/3, 1/4. При аналізі згорткових кодів зі швидкістю рівної 1/2 порівнювалися коди з кодовою обмеженням $m \leq 8$. Результати моделювання цих кодів наведені на рисунку 3.2. Статистичні дані наведені в таблиці 3.1.

Таблиця3.1 – Статистика сгорткових кодів зі швидкістю 1/2

С/Ш, Дб	Без код.	m=3	m =4	m =5	m =6	m =7	m =8
	Імовірність бітової помилки						
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 аналізі згорткових кодів зі швидкістю 1/3 порівнювалися коди з кодовим обмеженням $m \leq 8$. Результати моделювання цих кодів наведені на рисунку 3.3. Статистичні дані наведені в таблиці 3.2.

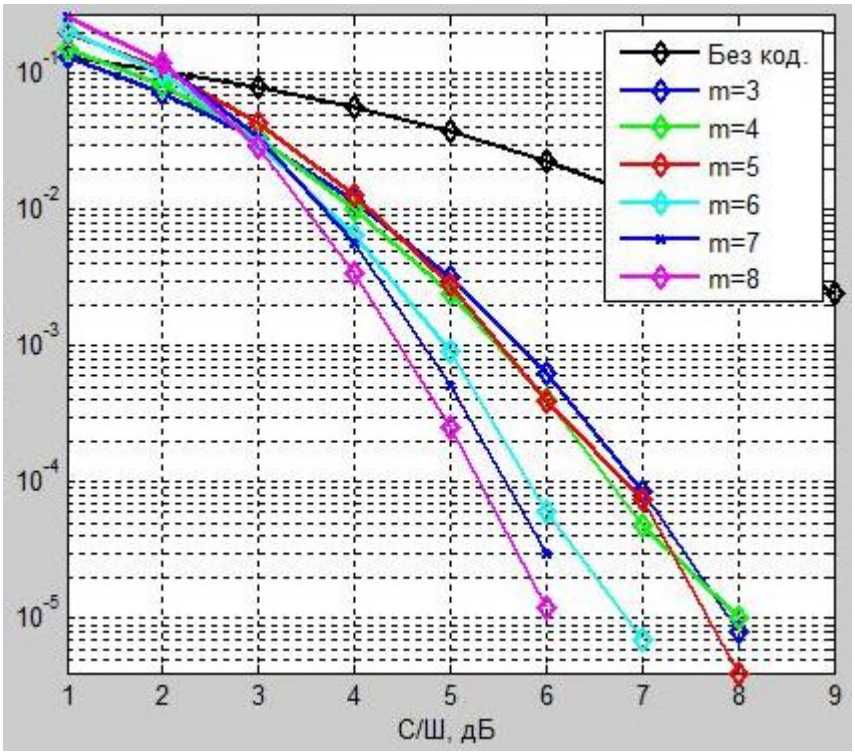


Рисунок 3.2 – Порівняння завадостійкості згорткових кодів з швидкістю 1/2 і без кодування

Таблиця 3.2 – Статистика згорткові коди зі швидкістю 1/3

С/Ш, Дб	Без код.	*m*=3	*m* =4	*m* =5	*m* =6	*m* =7	*m* =8
	Імовірність бітової помилки						
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}

3.3 Дослідження загорткових кодів різної швидкості з кодовою обмеженістю

При аналізі згорткових кодів зі швидкістю 1/4 порівнювалися коди з кодовою обмеженням $m \leq 6$. Результати моделювання цих кодів наведені на рисунку 3.4. Статистичні дані наведені в таблиці 3.3.

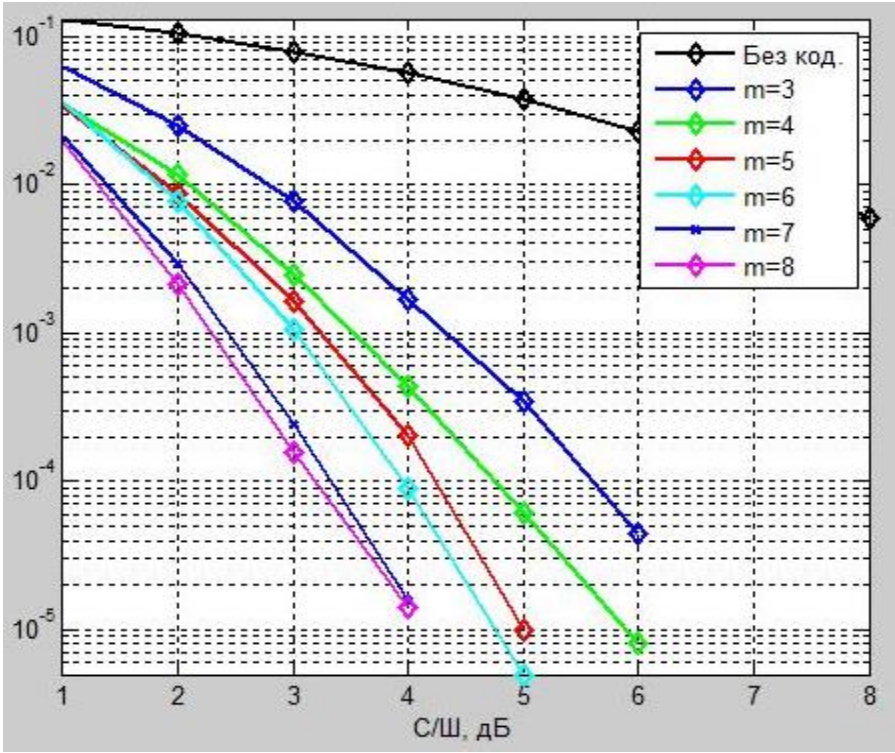


Рисунок 3.4 - Порівняння завадостійкості згорткових кодів з швидкістю 1/3 і без кодування

Таблиця 3.3 – Статистика згорткові коди зі швидкістю 1/4

С/Ш, Дб	Без код.	*m*=3	*m* =4	*m* =5	*m* =6
	Імовірність бітової помилки				
1	2	3	4	5	6
1	0,131	0,0229	0,0083	0,004	0,0025
2	0,1043	0,0064	0,0015	0,000551	0,000313

Продовження таблиці 3.3

1	2	3	4	5	6
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}

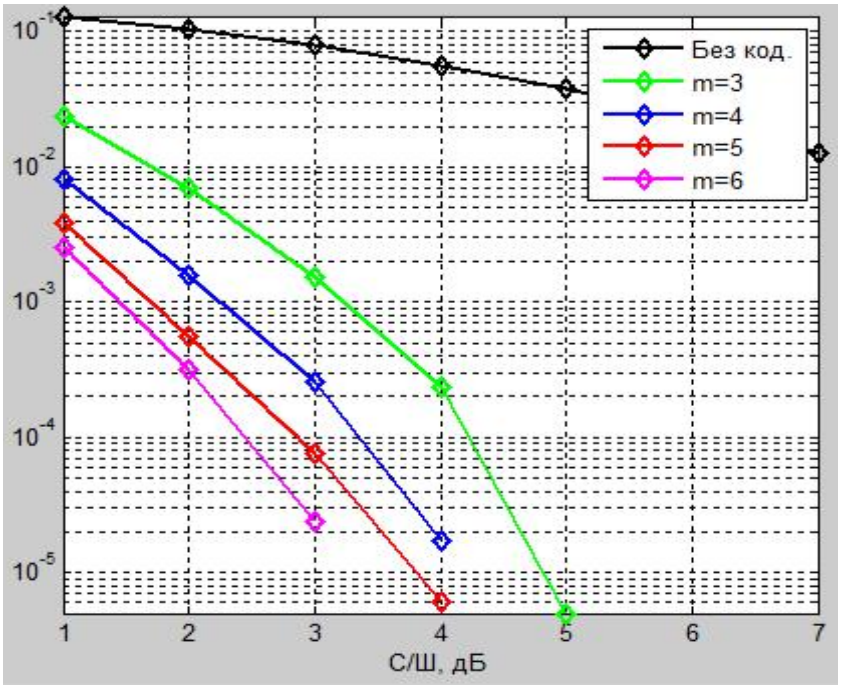


Рисунок 3.5 – Порівняння завадостійкості згорткових кодів швидкістю 1/4 і без кодування

Як випливає з аналізу кривих на рисунках 3.2 - 3.5 оцінка ймовірності бітової помилки при застосуванні згорткового кодування з підвищенням числа ячейок регістра зсуву *m* на одиницю стійкість від кодування збільшується приблизно від 0,2 до 0,5.

При однакових значеннях *m* виграш від кодування збільшується приблизно від 1 до 2.

Задамо довжину повідомлення $N = 10$ біт і $N = 100$ біт. Передачу повідомлення здійснюватимемо згортковими кодами з параметрами R=1/2, m=3.

Результати моделювання наведені на рисунку 3.6. Статистичні дані наведені в таблиці 3.4.

Аналіз графіків показує, що застосування згорткового кодування з декодуванням Вітербо для передачі коротких повідомлень ($N \leq 10$) недоцільне. Ефективність використання алгоритму Вітербо при передачі повідомлення великої довжини ($N = 100$) очевидна. Якщо в процесі декодування зроблена помилка у виборі шляху, то протягом кількох тактів декодер знову виходить на правильний шлях. Збільшення довжини повідомлення призводить до збільшення кроків декодування, що дозволяє значно підвищити коригувальні здібності згорткових кодів.

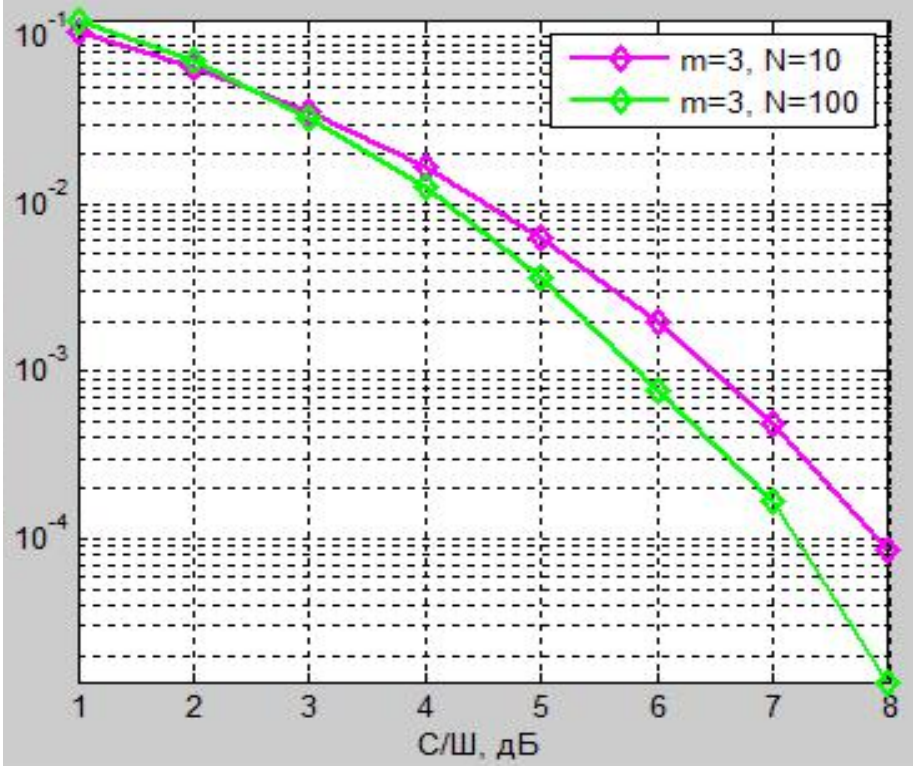


Рисунок 3.6 – Декодування згортальної коди по Вітербо при N=10 і N=100

В алгоритмі Вітербо ті шляхи решітки, про які заздалегідь відомо, що згідно з принципом максимальної правдоподібності вони не можуть бути оптимальними, не розглядаються.

Таблиця 3.4 – Статистика декодування по Вітербо

С/Ш, дБ	1	2	3	4	5	6	7	8
N	Імовірність бітової помилки							
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

Збільшення довжини повідомлення призводить до збільшення кроків декодування. Попередню відмову від малоймовірних шляхів спрощує процес декодування. Складність же алгоритму Вітербо зростає експоненціально з ростом числа ячейок регістра зсуву.

Отже, в даному розділі було отримано наступні результати:

– розроблено програмні модулі кодування і декодування по Вітербо згорткових кодів;

– проведено дослідження завадостійкості двійкових згорткових кодів в каналі з АБГШ. Були досліджені залежності ймовірності помилки від відносини сигнал-шум без згорткового кодування і зі згортковим кодуванням. При моделюванні з згортковим кодуванням використовувалася модель кодера зі швидкістю 1/2, 1/3, 1/4 і з малою довжиною кодового обмеження ($m \leq 8$);

– при збільшенні числа ячейок регістра кодера зростає стійкість кодування. Це можна пояснити зростанням повної довжини кодового обмеження щодо виходу, тобто збільшенням числа залежних вихідних кодових символів від інформаційних символів;

– перешкодостійкість зменшується при збільшенні швидкості коду. Застосування згорткового кодування і декодування по Вітербо ефективно при передачі повідомлень великої довжини.

4 ТЕХНІКО-ЕКОНОМІЧНИЙ РОЗДІЛ

Метою техніко – економічного розділу дипломної роботи є здійснення економічних розрахунків, спрямованих на визначення економічної ефективності апаратної реалізації алгоритмів кодування і декодування загорткових кодів та прийняття рішення про його подальший розвиток і впровадження або ж недоцільність проведення відповідної розробки. Для проведення даного дослідження необхідно провести ряд розразунків.

4.1 Розрахунок витрат на розробку програмного модуля

Витрати на розробку і впровадження програмного модуля для маршрутизації запитів у комп'ютерній мережі (K) включають:

$$K = K_1 + K_2,$$

де K_1 - витрати на розробку апаратного та програмного забезпечення грн.;

K_2 - витрати на відлагодження і дослідну експлуатацію програми рішення задачі на комп'ютері, грн.

Витрати на розробку апаратних та програмних засобів включають:

- витрати на оплату праці розробників ($B_{ОП}$);
- витрати на відрахування у спеціальні державні фонди ($B_ф$);
- витрати на матеріали та комплектуючі ($П_в$);
- накладні витрати (H);
- інші витрати ($I_в$)
- витрати на використання комп'ютерної техніки($B_{КТ}$).

Витрати на оплату праці включають заробітну плату (ЗП) всіх категорій працівників, безпосередньо зайнятих на всіх етапах проектування. Розмір ЗП обчислюється на основі трудоємності відповідних робіт у людино-днях та середньої ЗП відповідних категорій працівників.

У розробці проектного рішення задіяні наступні спеціалісти - розробники, а саме: керівник проекту; студент-дипломант; консультант техніко-економічного розділу (таблиця 4.1).

Таблиця 4.1 - Вихідні дані для розрахунку витрат на оплату праці

№п/п	Посада виконавців	Місячний оклад, грн.
1	Керівник ДП, викладач	6026
2	Консультант техніко-економічного розділу, доцент	6026
3	Студент	1100

Витрати на оплату праці розробників проекту визначаються за наступною формулою:

$$B_{ОП} = \sum_{i=1}^{N} \sum_{j=1}^{M} n_{ij} \cdot t_{ij} \cdot C_{ij} \ , \qquad (4.1)$$

де n_{ij} – чисельність розробників i-ої спеціальності j-го тарифного розряду, осіб;

t_{ij} – затрачений час на розробку проекту співробітником i-ої спеціальності j-го тарифного розряду, год;

C_{ij} – годинна ставка працівника i-ої спеціальності j-го тарифного розряду, грн.,

Середньогодинна ставка працівника може бути розрахована за такою формулою:

$$C_{ij} = \frac{C_{ij}^{0}(1+h)}{РЧ_i}, \qquad (4.2)$$

де C_{ij} – основна місячна заробітна плата розробника *i*-ої спеціальності *j*-го тарифного розряду, грн.;

h – коефіцієнт, що визначає розмір додаткової заробітної плати (при умові наявності доплат);

$РЧ_i$ - місячний фонд робочого часу працівника *i*-ої спеціальності *j*-го тарифного розряду, год. (приймаємо 168 год.).

Коефіцієнт *h*, який визначає розмір додаткової заробітної плати, для керівника та консультанта техніко-економічного розділу дорівнює 0,47.

Результати розрахунку записують до таблиці 4.2.

Таблиця 4.2 - Розрахунок витрат на оплату праці

№ п/п	Посада виконавців	Час розробки, год	Погодинна заробітна плата, грн/год.	Витрати на розробку, грн
1	Керівник ДП, доцент	16	88,6	1417,6
2	Консультант техніко-економічного розділу, доцент	2	88,6	177,2
3	Студент	144	6,55	943,2
Разом				2538

Відрахування на соціальні заходи. Величну відрахувань у спеціальні державні фонди визначають у відсотковому співвідношенні від суми основної та додаткової заробітних плат. Згідно діючого нормативного законодавства сума відрахувань у спеціальні державні фонди складає 20,5% від суми заробітної плати: $B_Ф = \frac{20,5}{100} \cdot 2538 = 520,29$ грн.

Загальна сума витрат на матеріальні ресурси (B_M) визначається за формулою:

$$B_M = \sum_{i=1}^{n} K_i \cdot Ц_i, \qquad (4.3)$$

де K_i - витрата *i*-го типу матеріалу, натуральні одиниці вимірювання;

$Ц_i$ - ціна за одиницю *i*-го типу матеріалу, грн.;

i - тип матеріального ресурсу;

n - кількість типів матеріальних ресурсів.

Таблиця 4.3 - Зведені розрахунки матеріальних витрат

Найменування матеріальних ресурсів	Од. виміру	Факт. витра-чено матері-алів	Ціна за одиницю, грн.	Сума, грн	Транс-портні витрати (10% від суми)	Загальна сума, грн
Arduino UNO	шт	1	412	412	41,2	452.2
Папір (формат А4)	уп	2	80	160	16	176
Ручка кулькова	шт	2	10	20	2	22
Датчик температ. та вологості	шт	1	100	100	10	110
Герконовий датчик	шт	2	35.1	70.2	7	77.2
Датчик руху	шт	1	127	127	12.7	139.7
Датчик відстані	шт	1	88.5	88.5	8,85	96,85
Р а з о м						1073,95

Витрати на використання комп'ютерної техніки ($В_{КТ}$) включають витрати на амортизацію комп'ютерної техніки, витрати на користування програмним забезпеченням, витрати на електроенергію, що споживається комп'ютером. За даними обчислювального центру ТНЕУ для комп'ютера типу IBM PC/ATX вартість години роботи становить 6 грн. Середній щоденний час роботи на комп'ютері – 2 години. Розрахунок витрат на використання комп'ютерної техніки приведений в таблиці 4.4.

Таблиця 4.4- Розрахунок витрат на використання комп'ютерної техніки

№ п/п	Назва етапів робіт, при виконанні яких використовується комп'ютер	Час використання комп'ютера, год.	Витрати на використання комп'ютера грн.
1	Проведення досліджень та оформлення їх результатів	60	360
2	Оформлення техніко-економічного розділу	8	48
3	Оформлення ДП	12	72
Разом		80	480

Накладні витрати проектних організацій включають три групи видатків: витрати на управління, загальногосподарські витрати, невиробничі витрати. Вони розраховуються за встановленими відсотками до витрат на оплату праці. Середньостатистичний відсоток накладних витрат приймемо 150% від заробітної плати: $H = 1{,}5 \cdot 1860{,}4 = 2790{,}6$ (грн).

Інші витрати є витратами, які не враховані в попередніх статтях. Вони становлять 10% від заробітної плати: $I_B = 1860{,}4 \cdot 0{,}1 = 186{,}04$ (грн).

Витрати на розробку програмного забезпечення складають:

$$K_1 = B_{ОП} + B_Ф + B_M + H + I_B + B_{КТ},$$

$$K_1 = 1860{,}4 + 381{,}38 + 1111{,}00 + 2790{,}6 + 186{,}04 + 480{,}00 = 6809{,}42 \text{ (грн)}.$$

Витрати на відлагодження і дослідну експлуатацію програмного продукту визначаємо за формулою:

$$K_2 = S_{м.г.} \cdot t_{від} \qquad (4.4)$$

де $S_{м.г.}$ - вартість однієї машино-години роботи ПК, грн./год;

$t_{від}$ - комп'ютерний час, витрачений на відлагодження і дослідну експлуатацію створеного програмного продукту, год.

Загальна кількість днів роботи на комп'ютері дорівнює 30 днів. Середній щоденний час роботи на комп'ютері – 2 години. Вартість години роботи комп'ютера дорівнює 6 грн., тому $K_2 = 6 \cdot 60 = 360$ грн.

4.2 Визначення експлуатаційних витрат

Для оцінки економічної ефективності розроблювальної системи моніторингу слід порівняти її з аналогом, тобто існуючим програмним забезпеченням ідентичного функціонального призначення.

Експлуатаційні одноразові витрати по програмному забезпеченню і аналогу включають вартість підготовки даних і вартість роботи комп'ютера (за час дії програми):

$$E_П = E_{1П} + E_{2П},$$

де $E_П$ - одноразові експлуатаційні витрати на ПЗ (аналог), грн.;

$E_{1П}$ - вартість підготовки даних для експлуатації ПЗ (аналогу), грн.;

$E_{2П}$ - вартість роботи комп'ютера для виконання проектного рішення (аналогу), грн.

Річні експлуатаційні витрати $B_{ЕП}$ визначаються за формулою:

$$B_{ЕП} = E_П * N_П,$$

де $N_П$ - періодичність експлуатації ПЗ (аналогу), раз/рік.

Вартість підготовки даних для роботи на комп'ютері визначається за формулою:

$$E_{1П} = \sum_{l=1}^{n} n_i t_i c_i \,,$$

де i - категорії працівників, які приймають участь у підготовці даних (i=1,2,...n);

n_i - кількість працівників i-ої категорії, осіб.;

t_i - трудомісткість роботи співробітників i-ої категорії по підготовці даних, год.;

c_i - середнього годинна ставка працівника i-ої категорії з врахуванням додаткової заробітної плати, що знаходиться із співвідношення:

$$c_i = \frac{c_i^0 (1+b)}{m},$$

де c_i^0 - основна місячна заробітна плата працівника i-ої категорії, грн.;

b - коефіцієнт, який враховує додаткову заробітну плату (приймемо 0,57);

m - кількість робочих годин у місяці, год.

Для роботи з даними як для проектного рішення так і аналогу потрібен один працівник, основна місячна заробітна плата якого складає: c = 3723 грн. Тоді:

$$c_1 = \frac{3723(1+0{,}57)}{22*8} = 33{,}21 \text{грн/год}$$

Трудомісткість підготовки даних для проектного рішення складає 1 год., для аналога 1,5 год.

Витрати на експлуатацію комп'ютера визначається за формулою:

$$E_{2П} = t * S_{МГ}$$

де t - витрати машинного часу для реалізації рішення (аналогу), год.;

$S_{МГ}$ - вартість однієї години роботи комп'ютера, грн./год.

Таблиця 4.5 - Розрахунок витрат на підготовку даних та реалізацію проектного рішення на комп'ютері

№	Час роботи співробітників, год.	Середньогодинна заробітна плата, грн./год.	Витрати , грн.
Проектне рішення			
1	1	33,21	33,21
Аналог			
2	1,5	33,21	66,42

Далі:

$$E_{2П} = 1* 6 = 6 \text{ грн.};\ E_{2А}=1,5*6 = 9 \text{ грн.}$$

$$E_{ІП}=33,21+6=39,21 \text{ грн. };\ E_{А}=66,42+9 = 75,42 \text{ грн.}$$

$$B_{ЕІП}= 39,21*252=9880,92 \text{ грн. };\ B_{ЕА}=75,42 *252=19005,84 \text{ грн.}$$

Накладні витрати пов'язані з обслуговуванням виробництва, утриманням апарату управління підприємства (фірми) та створення необхідних умов праці.

В залежності від організаційно-правової форми діяльності господарюючого суб'єкта, накладні витрати можуть становити 60–100 % від суми основної та додаткової заробітної плати працівників.

$$H_B = 0,7 * B_{ОП}, \quad (4.5)$$

де H_B – накладні витрати.

$$H_B=0,7*5845,11=4091,58 \text{ грн.}$$

Результати проведених розрахунків зведемо у таблицю 4.6.

Таблиця 4.6 - Кошторис витрат

№ п/п	Найменування витрат	Сума витрат, грн.
1	Витрати на оплату праці	1860,4
2	Відрахування у спеціальні державні фонди	381,38
3	Витрати на матеріали та комплектуючі	1074,00
4	Накладні витрати на розробку	2790,6
5	Інші витрати	186,04
6	Витрати на відлагодження і дослідну експлуатацію програмного продукту	360
7	Накладні витрати експлуатацію	4091,58
8	Річні експлуатаційні витрати	19005,84
Разом		29770,09

Договірна ціна ($Ц_Д$) для проектних рішень розраховується за формулою:

$$Ц_Д = B_{КС} \cdot \left(1 + \frac{p}{100}\right), \quad (4.6)$$

де $B_{КС}$ – кошторисна вартість, грн.;

p - середній рівень рентабельності, % (приймаємо 26% за погодженням з керівником): $Ц_Д = 29770,09 \cdot (1+0,26) = 37524,47$ грн.

4.3 Визначення економічної ефективності і терміну окупності капітальних вкладень

Економічна ефективність ($E_ф$) полягає у відношенні результату виробництва до затрачених ресурсів:

$$E_ф = \frac{П}{B_{КС}}, \quad (4.7)$$

де $П$ – прибуток, грн.;

$В_{КС}$ – кошторисна вартість, грн..

$$Е_ф = 7812,27 \text{ грн.} / 29786,09 \text{ грн.} = 0,25.$$

Поряд із економічною ефективністю розраховують термін окупності капітальних вкладень ($Тр$):

$$T_P = \frac{1}{E_P} . \quad (4.8)$$

Тобто: T_P =1/0,25 = 4р.

Прийнятним вважається термін окупності, близький до 7 років.

Розраховані економічні показники проекту занесемо до таблиці 4.7.

Таблиця 4.7 - Економічні показники розробки

№ п/п	Показник	Значення
1.	Собівартість, грн.	29786,09
2.	Плановий прибуток, грн.	7744,38в
3.	Ціна, грн.	37524,47
4.	Економічна ефективність	0,25
5.	Термін окупності, рік	4

Враховуючи основі економічні показники з таблиці 4.7, можна зробити висновок, що при економічній ефективності 0,25 та терміні окупності 4 роки проводити роботи по впровадженню даного програмного модуля є доцільним та економічно вигідним.

ВИСНОВКИ

У даній бакалаврській роботі розглянуті питання кодування і декодування двійкових згорткових кодів.

Основні результати роботи можна сформулювати наступним чином:

1 Вивчено способи кодування і досліджені впливу параметрів на роботу згорткових кодеров.

2 Вивчено способи кодування згорткових кодів за допомогою діаграм станів, деревовидних діаграм і ґратчастих діаграм.

3 Розроблено принципові електричні схеми кодерів в середовищі QUARTUSII, які можуть бути «зашиті» в кристал програмованих логічних інтегральних схем (ПЛІС).

4 Доопрацьовані програмні модулі кодування і декодування по Вітербо згорткових кодів в середовищі MATLAB, що дозволяють досліджувати стійкість згорткових кодів.

5 Досліджено залежності ймовірностей бітової помилки від відносини сигнал / шум при різних числі ячейок регістра (m) і різній кількості кодових символів (n). При цьому застосовується блоковий режим кодування.

6 Розроблено для навчального процесу дві лабораторні роботи з кодування і декодування згорткових кодів.

Таким чином, вирішені всі поставлені завдання. Отримані результати дозволяють зробити наступні висновки:

1. Розроблені принципові електричні схеми кодерів в середовищі QUARTUSII можуть бути «зашиті» в кристал програмованих логічних інтегральних схем (ПЛІС).

2. При збільшенні числа ячейок регістра кодера зростає стійкість кодування. Це можна пояснити зростанням повної довжини кодового обмеження щодо виходу, тобто збільшенням числа залежних вихідних кодових символів від інформаційних символів.

3. Перешкодостійкість зменшується при збільшенні швидкості коду.

4. Застосування згорткового кодування і декодування по Вітербо ефективно при передачі повідомлень великої довжини.

5. Розроблені віртуальні лабораторні роботи є хорошим «тренажером» при освоєнні теорії і практики по згорткових кодах.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

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

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

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

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

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

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

7 Быков В.В. Помехоустойчивые коды цифрового телевидения. Технологии информационного общества/ В.В. Быков, К.В. Меньшиков– СПб.: Издательство «Лань», 2013, №9. –195 с.

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

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

10 Выскубова Е.А. Применение пакета Simulink системы MATLAB при разработке программ сверточных кодеков./Е.А. Выскубова, Н.П. Хмырова // Техника радиосвязи. – 2010, Выпуск 15–254с.

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

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

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

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

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

16 Фрике К. Вводный курс цифровой электроники./ К.Фрике – 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./ А. И. Королев– Мн.: 2002. – 286 с.

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

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

21 Карпов Д.К. Сравнительная характеристика блочних и сверточных кодов./ Д.К. Карпов , В.В. Куликов, С.С. Манаенко, М.Э. Солчатов // Инфокоммуникационные технологии.- Том 5, 2007, №3. – 198 с.

22 Прокис Д. 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. / В.П. Дьяконов– М.: ДМК Пресс, 2008. – 768 с.

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

25. Методичні рекомендації до виконання дипломного проекту з освітньо-кваліфікаційного рівня “Бакалавр” напряму підготовки 6.050102 «Комп’ютерна інженерія» фахового спрямування «Комп’ютерні системи та мережі» / О.М. Березький, Л.О.Дубчак, Р.Б. Трембач, Г.М. Мельник, Ю.М. Батько, С.В. Івасьєв / Під ред. О.М. Березького. - Тернопіль: ТНЕУ, 2016.–65 с.

26. Методичні вказівки до написання техніко-економічного розділу для дипломних проектів на здобуття освітньо-кваліфікаційного рівня “Бакалавр” напряму підготовки 6.050102 «Комп’ютерна інженерія» / І.Р.Паздрій. - Тернопіль: ТНЕУ, 2015.– 36 с.