**1. АНАЛІТИЧНИЙ ОГЛЯД**

**1.1 Передача інформації по каналам зв'язку**

 Канал зв'язку - це сукупність засобів, призначених для передачі сигналів (повідомлень).

Для аналізу інформаційних процесів в каналі зв'язку можна використовувати його узагальнену схему, наведену на рис. 1.1.

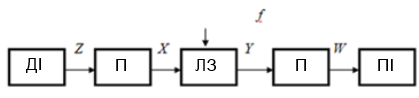


Рис. 1.1 Узагальнена схема каналу зв'язку

На рис. 1.1 прийняті наступні позначення: X, Y, Z, W - сигнали, повідомлення; f - перешкода; ЛЗ - лінія зв'язку; ДІ, ПІ - джерело і приймач інформації; П - перетворювачі (кодування, модуляція, декодування, демодуляція).

Існують різні типи каналів, які можна класифікувати за різними ознаками:

1. За типом ліній зв'язку: дротові; кабельні; оптико-волоконні; лінії електропередачі; радіоканали і т.д.
2. За характером сигналів: безперервні; дискретні; дискретно-безперервні (сигнали на вході системи дискретні, а на виході безперервні, і навпаки).
3. За завадозахищеністью: канали без перешкод; з перешкодами.

Канали зв'язку характеризуються:

1. Ємність каналу визначається як добуток часу використання каналу Tk, ширини спектра частот, що пропускаються каналом Fk і динамічного діапазону Dk., Який характеризує здатність каналу передавати різні рівні сигналів Vk = Tk Fk Dk.

Умова узгодження сигналу з каналом:

C ≤ Vk; Tc ≤ Tk; Fc ≤ Fk; Vc ≤ Vk; Dc ≤ Dk.

1. Швидкість передачі інформації - середня кількість інформації, що передається в одиницю часу.
2. Пропускна здатність каналу зв'язку - найбільша теоретично досяжна швидкість передачі інформації за умови, що похибка не перевершує заданої величини.
3. Надмірність - забезпечує достовірність переданої інформації (R = 0 ÷ 1).

Швидкість передачі даних в значній мірі залежить від передавального середовища в каналах зв'язку, в якості яких використовуються різні типи ліній зв'язку.

Провідні:

1. Провідні - кручена пара (що частково пригнічує електромагнітне випромінювання інших джерел). Швидкість передачі до 1 Мбіт/с. Використовується в телефонних мережах і для передачі даних.
2. Коаксіальний кабель. Швидкість передачі 10-100 Мбіт/с - використовується в локальних мережах, кабельному телебаченні і т.д.
3. Оптико-волоконна. Швидкість передачі 1 Гбіт/с.

В середовищах 1-3 затухання в дБ лінійно залежить від відстані, тобто потужність падає по експоненті. Тому через певну відстань необхідно ставити регенератори (підсилювачі).

Радіолінії:

1. Радіоканал. Швидкість передачі 100-400 Кбіт/с. Використовує радіочастоти до 1000 МГц. До 30 МГц за рахунок відбиття від іоносфери можливе поширення електромагнітних хвиль за межі прямої видимості. Але цей діапазон сильно зашумований (наприклад, аматорським радіозв'язком). Від 30 до 1000 МГц - іоносфера прозора і необхідна пряма видимість. Антени встановлюються на висоті (іноді встановлюються регенератори). Використовуються в радіо та телебаченні.
2. Мікрохвильові лінії. Швидкості передачі до 1 Гбіт/с. Використовують радіочастоти вище 1000 МГц. При цьому необхідна пряма видимість і гостро направлені параболічні антени. Відстань між регенераторами 10-200 км. Використовуються для телефонного зв'язку, телебачення і передачі даних.
3. Супутниковий зв'язок. Використовуються мікрохвильові частоти, а супутник служить регенератором (причому для багатьох станцій). Характеристики ті ж, що у мікрохвильових ліній.

**1.2 Завадостійкі коди**

Виявлення помилок в засобах зв'язку - дія, спрямована на контроль цілісності даних при записі/відтворенні інформації або при її передачі лініями зв'язку. Виправлення помилок (корекція помилок) - процедура відновлення інформації після читання її з пристрою зберігання або каналу зв'язку.

Для зменшення ймовірності помилок застосовують завадостійке кодування. Його сутність - введення при кодуванні додаткової надмірності, що збільшує можливість виявлення та виправлення помилок. Застосовувані при цьому коди називаються коригуючими.

Для виявлення помилок використовують коди виявлення помилок, для виправлення - коригувальні коди (коди, що виправляють помилки, коди з корекцією помилок, перешкодостійкі коди).

Перешкода - білий шум з гаусівським законом розподілу. У природі і техніці «чисто» білий шум (тобто білий шум, що має однакову спектральну потужність на всіх частотах) не зустрічається (зважаючи на те, що такий сигнал мав би нескінченну потужність), однак під категорію білих шумів потрапляють будь-які шуми, спектральна щільність яких однакова (або слабо відрізняється) у розглянутому діапазоні частот. Іноді помилково вважається, що гаусівський шум (тобто шум з гаусівським розподілом по амплітуді - нормальний розподіл) обов'язково є білим шумом. Однак ці поняття не еквівалентні. Гаусівський шум припускає розподіл значень сигналу у вигляді нормального розподілу, тоді як термін «білий» має відношення до кореляції сигналу в два різних моменти часу (ця кореляція не залежить від розподілу амплітуди шуму).

Нормальний розподіл, також звані гаусівським розподілом або розподілом Гауса - розподіл ймовірностей, яке відіграє найважливішу роль у багатьох галузях знань. Фізична величина підпорядковується нормальному розподілу, коли вона схильна до впливу величезного числа випадкових перешкод. Ясно, що така ситуація вкрай поширена, тому можна сказати, що з усіх розподілів в природі найчастіше зустрічається саме нормальний розподіл - звідси і походить одна з його назв. Нормальний розподіл залежить від двох параметрів - зсуву і масштабу, тобто є з математичної точки зору не одним розподілом, а цілим їх сімейством. Значення параметрів відповідають значенням середнього (математичного сподівання) і розкиду (стандартного відхилення).

**1.2.1 Способи боротьби з помилками**

В процесі зберігання даних і передачі інформації по мережам зв'язку неминуче виникають помилки. Контроль цілісності даних і виправлення помилок - важливі завдання на багатьох рівнях роботи з інформацією (зокрема, фізичному, канальному, транспортному рівнях моделі OSI).

В системах зв'язку можливі кілька стратегій боротьби з помилками:

* виявлення помилок у блоках даних і автоматичний запит повторної передачі пошкоджених блоків - цей підхід застосовується в основному на канальному і транспортному рівнях;
* виявлення помилок у блоках даних і відкидання пошкоджених блоків - такий підхід іноді застосовується в системах потокового мультимедіа, де важлива затримка передачі і немає часу на повторну передачу;
* виправлення помилок (forward error correction) застосовується на фізичному рівні.

**1.2.2 Коди виявлення та виправлення помилок**

Коригувальні коди - коди, для виявлення або виправлення помилок, що виникають при передачі інформації під впливом перешкод, а також при її зберіганні.

Для цього при записі (передачі) в корисні дані додають спеціальним чином структуровану надлишкову інформацію (контрольне число), а при читанні (прийомі) її використовують для того, щоб виявити або виправити помилки. Природно, що число помилок, яке можна виправити, обмежене і залежить від конкретного застосовуваного коду.

З кодами, що виправляють помилки, тісно пов'язані коди виявлення помилок. На відміну від перших, останні можуть тільки встановити факт наявності помилки в переданих даних, але не виправити її.

Насправді, використовувані коди виявлення помилок належать до тих же класів кодів, що і коди, що виправляють помилки. Фактично, будь-який код, що виправляє помилки, може бути також використаний для виявлення помилок (при цьому він буде здатний виявити більше число помилок, ніж був здатний виправити).

За способом роботи з даними коди, що виправляють помилки діляться на блокові, що ділять інформацію на фрагменти постійної довжини і обробляють кожен з них окремо, і згорткові, що працюють з даними як з безперервним потоком. Згорткові коди, як правило, породжуються дискретною лінійною інваріантною в часі системою. Тому, на відміну від більшості блокових кодів, згорткове кодування - дуже проста операція, чого не можна сказати про декодування. Кодування згортковим кодом проводиться за допомогою регістра зсуву (автомат Мілі), відводи від якого сумуються по модулю два. Таких сум може бути дві (найчастіше) або більше. Декодування згорткових кодів, як правило, проводиться за алгоритмом Вітербі, який намагається відновити передану послідовність згідно з критерієм максимальної правдоподібності. Згорткові коди ефективно працюють в каналі з білим шумом, але погано справляються з пакетами помилок. Більш того, якщо декодер помиляється, на його виході завжди виникає пакет помилок.

**1.3** **Класифікація** **кінцевих** **абстрактних** **автоматів**

 Абстрактний автомат (в теорії алгоритмів) - математична абстракція, модель дискретного пристрою, що має один вхід, один вихід і в кожен момент часу знаходиться в одному стані з безлічі можливих. На вхід цьому пристрою надходять символи однієї мови, на виході воно видає символи (в загальному випадку) іншої мови. За способом формування функцій виходів виділяють автомати Мілі і Мура.

**1.3.1 Автомат Мілі**

Автомат Мілі (англ. Mealy machine) - кінцевий автомат, вихідна послідовність якого (на відміну від автомата Мура) залежить від стану та вхідної послідовності.

Математична модель автомата складається з сукупність п'яти об'єктів:



де:  і - Кінцеві не порожні множини, а і - відображення виду:



 і

 зі зв'язком елементів множин *S, X* і *Y* в абстрактному часі T = {0, 1, 2, ...} рівняннями:





(Відображення і отримали назви, відповідно функції переходів і функції виходів автомата *A).*



Особливістю автомата Мілі є те, що функція виходів є двоаргументною і символ у вихідному каналі *y (t)* виявляється тільки при наявності символу у вхідному каналі *x (t).*

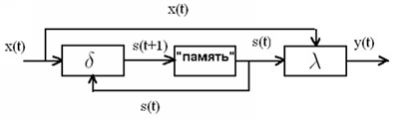


Рис. 1.2 Функціональна схема автомата Мілі

**1.3.2 Автомат Мура**

Залежність вихідного сигналу тільки від стану представлена ​​в автоматах типу Мура (англ. Moore machine). В автоматі Мура функція виходів визначає значення вихідного символу тільки по одному аргументу - станом автомата. Цю функцію називають також функцією міток, так як вона кожному стану автомата ставить мітку на виході.

Кінцевим детермінованим автоматом типу Мура називається сукупність п'яти об'єктів:



 де ,  і - Відповідають визначенню автомата типу Мілі, а є відображенням виду: .

із залежністю станів і вихідних сигналів у часі рівнянням:



 Особливістю автомата Мура є те, що символ *y(t)* в вихідному каналі існує весь час, поки автомат знаходиться в стані *s(t).*

Для будь-якого автомата Мура існує автомат Мілі, який реалізує тугіше саму функцію. І навпаки: для будь-якого автомата Мілі існує відповідний автомат Мура.

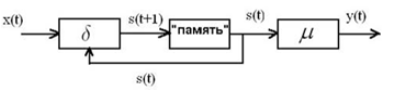


Рис. 1.3 Функціональна схема автомата Мура

**1.4 згорткове кодування**

 Типова функціональна схема системи цифрового зв'язку використовує згорткове кодування/декодування і модуляцію/демодуляцію, є показана на рис. 1.4. Вихідне повідомлення на вході позначається послідовністю  де , - двійковий знак (біт), a *i* ​​- індекс часу. Якщо бути точним, то елементи *m* слід було б доповнювати індексом члена класу (наприклад, для бінарного коду, 1 або 0) і індексом часу. Надалі для простоти будемо використовувати тільки індекс, що позначає час (або розташування елемента всередині послідовності). Будемо припускати, що всі , рівно ймовірно рівні одиниці або нулю і незалежні між собою. Будучи незалежною, послідовність бітів потребує деякої надмірності, тобто знання про біт , не дає ніякої інформації про біт , (При ). Кодер перетворює кожну послідовність *m* в унікальну послідовність кодових слів . Навіть попри те, що послідовність *m* однозначно визначає послідовність *U,* ключовою особливістю згортальних кодів є те, що даний *k* - кортеж всередині *m* неоднозначно визначає пов'язані з ним *n* - кортежі всередині *U,* оскільки кодування кожного з *k* - кортежів є функцією не тільки *k* - кортежів, а й попередніх К-1 *k* - кортежів. Послідовність *U* можна розділити на послідовність кодових слів: . Кожне кодове слово *U,* складається з двійкових кодових символів, часто званих канальними символами, канальними бітами, або бітами коду; на відміну від бітів вхідного повідомлення, кодові символи не є незалежними.

У типових системах зв'язку послідовність кодових слів *U* модулюється сигналом . У ході передачі сигнал спотворюється шумом, в результаті чого, як показано на рис. 1.4, виходить сигнал  і демодулювана послідовність  Задача декодера полягає в отриманні оцінки вихідної послідовності повідомлення за допомогою отриманої послідовності *Z* і апріорних знань про процедуру кодування.



Рис. 1.4 Кодування/декодування і модуляція/демодуляція в каналі зв'язку

Звичайний згортковий кодер, показаний на рис. 1.5, реалізується з *kK* - розрядним регістром зсуву і *n* суматорами по модулю 2, де *K* - довжина кодового обмеження. Довжина кодового обмеження - це кількість *k* - бітових зсувів, після яких один інформаційний біт може вплинути на вихідний сигнал кодера. У кожен момент часу на місце перших *k* розрядів регістра переміщуються *k* нових біт; всі біти в регістрі зміщуються на *k* розрядів вправо, і вихідні дані *n* суматорів послідовно дискретизуються, даючи, в результаті, біти коду. Потім ці символи коду використовуються модулятором для формування сигналів, які будуть передані по каналу. Оскільки для кожної вхідної групи з *k* біт повідомлення мається *n* біт коду, ступінь кодування дорівнює *k/n* біт повідомлення на біт коду, де *k < n.*



Рис. 1.5 Згортковий кодер з довжиною кодового обмеження K і ступенем кодування k/n

 Розглянемо тільки найбільш часто використовувані двійкові згорткові кодери, для яких *k* - 1, тобто ті кодуючі пристрої, в яких біти повідомлення зсуваються по одному біту за раз, хоча узагальнення на алфавіти більш високих порядків не викликає ніяких труднощів. Для кодера з *k* = 1, за *i* -й момент часу біт повідомлення , Буде переміщений на місце першого розряду регістра зсуву; всі попередні біти в регістрі будуть зміщені на один розряд вправо, а вихідний сигнал *n* суматорів буде послідовно оцифрований і переданий. Оскільки для кожного біта повідомлення мається  *n* біт коду, ступінь кодування дорівнює 1/*n.* Наявні в момент часу , *n* кодових символів складають *i*-е кодове слово гілки, де  (J - 1, 2, ..., *n)* - це *j*-й кодовий символ, що належить *i* -му кодовому слову гілки. Відзначимо, що для кодера зі ступенем кодування 1/*n, kK* розрядний регістр зсуву для простоти можна називати *K*-розрядним регістром, а довжину кодового обмеження *K,* яка виражається в одиницях розрядів *k*-кортежів, можна іменувати довжиною кодового обмеження в бітах.

**1.4.1 Представлення згорткового кодера**

Щоб мати можливість описувати згортковий код, необхідно визначити кодуючу функцію  так, щоб з даної вхідної послідовності *m* можна було швидко обчислити вихідну послідовність *U.* Для реалізації згорткового кодування використовується кілька методів; найбільш поширеними з них є графічний зв'язок, вектори, поліноми зв'язку, діаграма стану, деревоподібна і ґратчаста діаграми. Всі вони розглядаються нижче.

**1.4.2 Представлення зв'язку**

При розгляді згортальних кодерів в якості моделі використовуватимемо згорткових кодер, показаний на рис. 1.6. На цьому рис. зображений згортковий кодер з довжиною кодового обмеження *K* = 3. В ньому є *n* = 2 суматорів по модулю 2; отже, ступінь кодування коду *k/n* дорівнює 1/2.При кожному надходженні біт поміщається в крайній лівий розряд, а біти регістра зміщуються на одну позицію вправо. Потім комутатор на виході дискретизує виходи всіх суматорів по модулю 2 (тобто спочатку верхній суматор, потім нижній), в результаті чого формуються пари кодових символів, що утворюють кодове слово, пов'язане із щойно надійшовшим бітом. Це виконується для кожного вхідного біта. Вибір зв'язку між суматорами і розрядами регістра впливає на характеристики коду. Всяка зміна у виборі зв'язків призводить в результаті до різних кодів.

На відміну від блокових кодів, що мають фіксовану довжину слова *n,* в згортальних кодах немає певного розміру блоку. Однак за допомогою періодичного відкидання згортковим кодам часто примусово надають блочну структуру. Це вимагає деякої кількості нульових розрядів, приєднаних до кінця вхідної послідовності даних, які служать для очищення (або «промивання») регістра зсуву від біт даних. Оскільки додані нулі не несуть додаткової інформації, ефективна ступінь кодування буде нижче *k/n.* Щоб ступінь кодування залишалася близькою до *k/n,* період відкидання найчастіше роблять настільки великим, наскільки це можливо.



Рис. 1.6 Згортковий кодер (ступінь кодування 1/2, K = 3)

Один із способів реалізації кодера полягає у визначенні *n* векторів зв'язку, по одному на кожен з *n* суматорів по модулю 2. Кожен вектор має розмірність *K* і описує зв'язок регістра зсуву кодера з відповідним суматором по модулю 2. Одиниця на *i*-й позиції вектора вказує на те, що відповідний розряд в регістрі зсуву пов'язаний з суматором по модулю 2, а нуль в даній позиції вказує, що зв'язків між розрядом і суматором по модулю 2 не існує. Для кодера на рис. 1.6 можна записати вектор зв'язку  для верхніх зв'язків, a - для нижніх.





Припустимо тепер, що вектор повідомлення *m* = 1 0 1 закодований з використанням згортального коду і кодера, показаного на рис. 1.6. Введено три біта повідомлення, по одному в момент часу ,  і , як показано на рис. 1.7. Потім для очистки регістра в моменти часу *t4* і *t5* введені (К - 1) = 0 0, що в результаті призводить до зміщення кінцевої ділянки на всю довжину регістра. Послідовність на виході виглядає наступним чином: 1 1 1 0 0 0 1 0 1 1, де крайній лівий символ представляє першу передачу. Для декодування повідомлення потрібна повна послідовність на виході (що включає кодові символи). Для видалення із повідомлення з кодера вимагається на одиницю менше нулів, ніж мається розрядів в регістрі, або К - 1 очищених біт. В момент часу *t6* показаний нульовий вихід, це повинно дати читачеві можливість переконатися в тому, що в момент часу *t5* регістр встановлюється в початковий стан. Таким чином, в момент часу *t6* вже можна передавати нове повідомлення.

**1.4.3 Реакція кодера на імпульсне збурення**

Ми можемо описати кодер через його імпульсну характеристику, тобто у вигляді відгуку кодера на одиничний поступаючий біт. Розглянемо вміст регістра при проходженні через нього двійкової одиниці.



Рис. 1.7 Згорткове кодування послідовності повідомлення зі ступенем кодування 1/2 кодером з К = 3.

  Послідовність на виході при одиниці на вході називається відгуком кодера на імпульсне збурення, або його імпульсною характеристикою. Для вхідної послідовності m = 1 0 1 дані на виході можуть бути знайдені шляхом суперпозиції або лінійного складання зміщених у часі вхідних "імпульсів".

  Слід зазначити, що ці дані на виході такі ж, як і на рис. 1.7, що вказує на лінійність згортальних кодів - точно так само як і в блокових кодах. Назва згортковий кодер (convolutional encoder) виникла саме внаслідок цієї властивості генерації даних на виході за допомогою лінійного складання (або згортки) зміщених у часі імпульсів послідовності на вході з імпульсною характеристикою кодера. Такі пристрої часто описуються за допомогою матричного генератора нескінченного порядку.

Відзначимо, що в розглянутому вище прикладі вхідної послідовності з 3 біт і послідовності на виході з 10 біт ефективна ступінь кодування становить k/n = 3/10, що значно менше величини 1/2, яку можна було б очікувати, знаючи, що кожен біт даних на вході породжує пару канальних бітів на виході. Причина цього полягає в тому, що фінальні біти даних потрібно провести через кодер. Всі канальні біти на виході потрібні в процесі декодування. Якби повідомлення було довшим, скажімо 300 біт, послідовність кодових слів на виході містила б 640 біт і значення для ступеня кодування коду 300/640 було б значно ближче до 1/2.

**1.4.4 Поліноміальне представлення**

Іноді зв'язки кодера описуються за допомогою поліноміального генератора для опису реалізації зворотного зв'язку регістра зсуву циклічних кодів. Згортковий кодер можна представити у вигляді набору з n поліноміальних генераторів, по одному для кожного з n суматорів по модулю 2. Кожен поліном має порядок K - 1 або менше і описує зв'язок кодує регістра зсуву з відповідним суматором по модулю 2, майже так само як і вектор зв'язку. Коефіцієнти біля кожного доданка поліному порядку (К - 1) дорівнюють або 1, або 0, залежно від того, чи є зв'язок між регістром зсуву і суматором по модулю 2. Для кодера на рис 2.6 можна записати поліноміальний генератор для верхніх зв'язків і - Для нижніх.

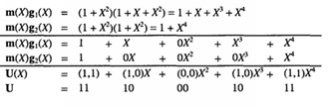




 Тут доданок самого нижнього порядку в поліномі відповідає вхідному розряду регістра. Вихідна послідовність знаходиться наступним чином:

 чергується з 

 Перш за все, висловимо вектор повідомлення m = 1 0 1 у вигляді полінома, тобто . Для очистки регістра ми знову будемо припускати використання нулів, наступних за бітами повідомлення. Тоді виходить поліном *U (X),* або що виходить послідовність *U* кодера для вхідного повідомлення *m* може бути знайдена в такий спосіб:



 В цьому прикладі ми почали обговорення з того, що згортковий кодер можна трактувати як набір регістрів зсуву циклічного коду. Ми представили кодер у вигляді поліноміальних генераторів, за допомогою яких описуються циклічні коди. Однак ми прийшли до тієї ж послідовності на виході, що і на рис. 1.7, і до тієї ж, що і в попередньому розділі, отриманої при описі реакції на імпульсне збурення.

**1.4.5 Представлення стану і діаграма станів**

Згортковий кодер належить класу пристроїв, відомих як кінцевий автомат. Для згортального коду зі ступенем кодування 1/n стан представлено вмістом K - 1 крайніх правих розрядів (рис. 1.7). Знання стану плюс знання наступних даних на вході є необхідною і достатньою умовою для визначення даних на виході. Отже, нехай стан кодера в момент часу , Визначається як  *i*-та гілка кодових слів *U,* повністю визначається станом *X,* і введеними в даний час бітами ; таким чином, стан *X,* описує передісторію кодера для визначення даних на його виході. Стани кодера вважаються Марківськими в тому сенсі, що ймовірність  знаходження в стані , обумовлена ​​всіма попередніми станами, залежить тільки від самого останнього стану , Тобто вона дорівнює .

Одним із способів подання простих кодуючих пристроїв є діаграма стану (state diagram); таке подання кодера, зображеного на рис. 1.6, показано на рис. 1.8. Стани, показані в рамках діаграми, являють собою можливий міст К - 1 крайніх правих розрядів регістра, а шляхи між станами - кодові слова гілок на виході, що є результатом переходів між такими станами. Стани регістра обрані наступними: а = 00, b = 10, с = 01 і d = 11; діаграма, показана на рис. 1.8, ілюструє всі можливі зміни станів для кодера, показаного на рис. 1.6. Існує всього два виходячих з кожного стану переходів, що відповідають двом можливим вхідним бітам. Далі для кожного шляху між станами записано кодове слово на виході, пов'язане з переходами між станами. При зображенні шляхів, суцільною лінією прийнято позначати шлях, пов'язаний з нульовим вхідним бітом, а пунктирною лінією - шлях, пов'язаний з одиничним вхідним бітом. Відзначимо, що за один перехід неможливо перейти з даного стану в будь-який довільний. Так як за одиницю часу переміщається тільки один біт, існує тільки два можливих переходи між станами, в які регістр може переходити за час проходження кожного біта.

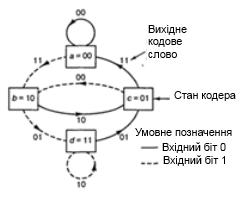


Рис. 1.8 Діаграма станів кодера (ступінь кодування 1/2, К = 3)

**1.4.6 Деревовидні діаграми**

Незважаючи на те, що діаграми станів повністю описують кодер, по суті, їх не можна використовувати для легкого відстеження переходів кодера в залежності від часу, оскільки діаграма не представляє динаміки змін. Деревоподібна діаграма (tree diagram) додає до діаграми стану часовий вимір.

Деревоподібна діаграма згорткового кодера, показаного на рис. 1.6, зображена на рис. 1.9. У кожен наступний момент проходження вхідного біта процедура кодування може бути описана за допомогою переміщення по діаграмі зліва направо, причому кожна гілка дерева описує кодове слово на виході. Правило розгалуження для знаходження послідовності кодових слів наступне: якщо вхідним бітом є нуль, то він зв'язується зі словом, яке знаходиться шляхом переміщення в наступну (у напрямку вгору) праву гілку; якщо вхідний біт - це одиниця, то кодове слово знаходиться шляхом переміщення в наступну (у напрямку вниз) праву гілку.

Передбачається, що спочатку кодер містив одні нулі. Діаграма показує, що якщо першим вхідним бітом був нуль, то кодовим словом гілки на виході буде 00, а якщо першим вхідним бітом була одиниця, то кодовим словом на виході буде 11. Аналогічно, якщо першим вхідним бітом була одиниця, а другим - нуль, на виході другим словом гілки буде 10. Якщо першим вхідним бітом була одиниця і другим вхідним бітом була одиниця, другим кодовим словом на виході буде 01. Слідуючи цій процедурі, бачимо, що вхідна послідовність 110 11 представляється жирною лінією, намальованою на деревовидній діаграмі (рис. 1.9). Цей шлях відповідає вихідний послідовності кодових слів 1 1 0 1 0 1 0 0 0 1.

Доданий вимір часу в деревовидній діаграмі (порівняно з діаграмою стану) допускає динамічний опис кодера як функції конкретної вхідної послідовності. Однак при спробі опису за допомогою деревовидної діаграми послідовності довільної довжини виникає проблема: число відгалужень росте як *2L,* де L - це кількість кодових слів гілок в послідовності.

**1.4.7 Ґратчаста діаграма**

Дослідження деревовидної діаграми на рис. 1.9 показує, що в цьому прикладі після третього розгалуження в момент часу t+4 структура повторюється (в загальному випадку деревоподібна структура повторюється після К відгалужень, де К - довжина кодового обмеження). Позначимо кожен вузол в дереві (рис. 1.9), ставлячи у відповідність чотири можливих стани в регістрі зсуву: а = 00, b = 10, с = 01 і d = 11.Перше розгалуження деревовидної структури в момент часу дає пару вузлів, помічених як а і b. При кожному наступному розгалуженні кількість вузлів подвоюється. Друге розгалуження в момент часу t2 дає в результаті чотири вузли, позначених як а, b, с і d. Після третього розгалуження всього мається вісім вузлів: два - а, два - b, два - з і два - d.

На малюнку 1.9 можна побачити, що всі гілки виходять з двох вузлів одного і того ж стану, утворюючи ідентичні гілки послідовностей кодових слів. В цей момент дерево ділиться на ідентичні верхню і нижню частини. Сенс цього стає яснішим після розгляду кодера, зображеного на рис. 1.6, коли четвертий вхідний біт входить в кодер зліва, перший вхідний біт праворуч викидається і більше не впливає на кодові слова на виході.

Отже, вхідні послідовності 1 0 0 х у ... і 0 0 0 х у ..., де крайній лівий біт є найбільш раннім, після (К = 3) -го розгалуження генерують однакові кодові слова гілок.

Це означає, що будь-які стани, що мають однакову мітку в один і той же момент  можна з'єднати, оскільки всі наступні шляхи будуть невиразні. Якщо ми зробимо це для деревовидної структури, показаної на рис. 1.9, отримаємо іншу діаграму яка називається ґратчастою.

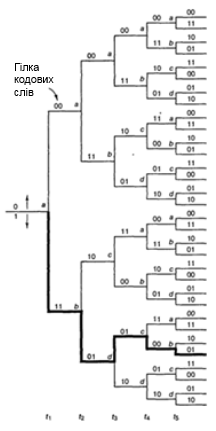


Рис. 1.9. Деревоподібне представлення кодера (ступінь кодування 1/2, К = 3)

 Ґратчаста діаграма, яка використовує повторювану структуру, дає більш зручний опис кодера, порівняно з деревовидною діаграмою. Ґратчаста діаграма для згорткового кодера, зображеного на рис. 1.6, показана на рис. 1.10.

При зображенні ґратчастої діаграми (рисунок 1.10) ми скористалися тими ж умовними позначеннями, що і для діаграми стану: суцільна лінія позначає вихідні дані, які генеруються вхідним нульовим бітом, а пунктирна - вихідні дані, які генеруються вхідним одиничним бітом.



Рис. 1.10 - Ґратчаста діаграма кодера (ступінь кодування 1/2, К = 3)

Вузли решітки представляють стани кодера; перший ряд вузлів відповідає стану a = 00, другий і наступні - станам b = 10, с = 01 і d - 11.У кожен момент часу для представлення 2К-1 можливих станів кодера решітка вимагає 2К-1 вузлів. У нашому прикладі після досягнення глибини решітки, яка дорівнює трьом (в момент часу *t4*), помічаємо, що решітка має фіксовану періодичну структуру. У загальному випадку фіксована структура реалізується після досягнення глибини *К.* Отже, з цього моменту в кожний стан можна увійти з будь-якого з двох попередніх станів. Також з кожного стану можна перейти в один з двох станів. З двох вихідних гілок одна відповідає нульовому вхідному біту, а інша - одиничному вхідному біту. На рис. 1.10 кодові слова на виході відповідають переходам між станами, показаними як мітки на гілках решітки.

Один стовпець тимчасового інтервалу сформованої ґратчастої структури кодування повністю визначає код. Кілька стовпців показані виключно для візуалізації послідовності кодових символів як функції часу. Стан згорткового кодера представлено вмістом крайніх правих K -1 розрядів в регістрі кодера. Деякі автори описують стан за допомогою крайніх лівих K -1 розрядів. Який опис правильний? Вони обидва вірні. Кожний перехід має початковий і кінцевий стан. Крайні праві K -1 розрядів описують початковий стан для поточних вхідних даних, які знаходяться в крайньому лівому розряді (ступінь кодування передбачається рівний 1/n). Крайні ліві К -1 розрядів є кінцевим станом для такого переходу. Послідовність кодових символів характеризується *N* гілками (що представляє *N* біт даних), які займають *N* інтервалів часу. Вона пов'язана з конкретним станом в кожен з *N* +1 інтервалів часу (від початку до кінця). Таким чином, ми запускаємо біти в моменти часу *t1, t2, ..., tN* і визначаємо метрику стану в моменти часу *t1, t2, ..., tN+1*. Тут використана така умова: поточний біт розташовується в крайньому лівому розряді, а крайні праві K-1 розрядів стартують зі стану з усіма нулями. Цей момент часу позначимо як початковий час, *t1*. Час завершення останнього переходу позначимо як час припинення роботи, *tN + 1*.

**1.5 Декодування по методу максимальної правдоподібності**

 Якщо всі вхідні послідовності повідомлень різновірогідні, мінімальна ймовірність помилки виходить при використанні декодера, який порівнює умовні ймовірності і вибирає максимальну. Умовні ймовірності також називають функціями правдоподібності , де *Z* - це прийнята послідовність, a - Одна з можливих переданих послідовностей. Декодер вибирає , якщо

 (1.1) по всіх .

Принцип максимального правдоподібності, який визначається рівнянням (1.1), є фундаментальним досягненням теорії прийняття рішень; це формалізація способу прийняття рішень, заснованого на "здоровому глузді", коли є статистичні дані про ймовірності. При розгляді двійковій демодуляції передбачається передача тільки двох рівноймовірно сигналів і . Отже, прийняття двійкового рішення на основі принципу максимальної правдоподібності, що стосується даного отриманого сигналу, означає, що в якості переданого сигналу вибирається , якщо   в іншому випадку вважається, що переданий сигнал був . Параметр *z* представляє собою величину *z* (T), значення прийнятого сигналу до детектування в кінці кожного періоду передачі символу t = T. Однак при використанні принципу максимальної правдоподібності в задачі згорткового декодування, в згортковому коді виявляється наявність пам'яті (отримана послідовність є суперпозицією поточних і попередніх двійкових розрядів). Таким чином, застосування принципу максимальної правдоподібності при декодуванні біт даних, закодованих згорткових кодом, здійснюється в контексті вибору найбільш імовірної послідовності, як показано в рівнянні (1.1). Зазвичай мається безліч можливих переданих послідовностей кодових слів. Що стосується двійкового коду, то послідовність з *L* кодових слів є членом набору з 2L можливих послідовностей. Але, в контексті максимальної правдоподібності можна сказати, що в якості переданої послідовності декодер вибирає , якщо правдоподібність  більше правдоподібності всіх інших можливо переданих послідовностей. Такий оптимальний декодер, здатний мінімізувати ймовірність помилки (коли всі передані послідовності різновірогідні), відомий як декодер, що працює за принципом максимальної правдоподібності (maximum likelihood detector). Функція правдоподібності за дається або обчислюється, виходячи із специфікації каналу. Припустимо, що ми маємо справу з адитивним білим гаусовим шумом з нульовим середнім, отже, каналом без пам'яті, тобто шум впливає на кожен символ коду незалежно від інших символів. При ступені кодування згортального коду, рівний 1/n, правдоподібність можна виразити таким чином:

 (1.2)

 де - *i*-та гілка прийнятої послідовності *Z,*

 - це гілка окремої послідовності кодових слів ,

 - це j-й кодовий символ ,

 - це j-й кодовий символ , а кожна гілка складається з n кодових символів.

Задача декодування полягає у виборі шляху крізь грати, показану на рис. 1.10 (кожен можливий шлях визначає послідовність кодових слів), таким чином, щоб вираз

 був максимальним. (1.3)

 Як правило, при обчисленнях зручніше користуватися логарифмом функції правдоподібності, оскільки це дозволяє вираз замінити підсумовуванням. Ми можемо скористатися таким перетворенням, оскільки логарифм є монотонно зростаючою функцією і, отже, не внесе змін у вибір остаточного кодового слова. Логарифмічну функцію правдоподібності можна визначити наступним чином:

 (1.4)

 Тепер завдання декодування полягає у виборі шляху вздовж дерева на рис. 1.9 або решітки на рис. 1.10 таким чином, щоб  було максимальним. При декодуванні згортальних кодів можна використовувати як деревоподібну, так і ґратчасту структуру. При деревовидному поданні коду ігнорується те, що шляхи знову об'єднуються. Для двійкового коду кількість можливих послідовностей, що складаються з L кодових слів, так само 2L. Тому декодування отриманих послідовностей, засноване на принципі максимального правдоподібності з використанням деревовидної діаграми, вимагає методу "грубої сили" або вичерпного зіставлення 2L накопичених логарифмічних метрик правдоподібності, що описують всі варіанти можливих послідовностей кодових слів.

Тому розглядати декодування на основі принципу максимальної правдоподібності за допомогою деревовидної структури практично неможливо. У попередньому розділі було показано, що при ґратчастому поданні коду декодер можна побудувати так, щоб можна було відмовлятися від шляхів, які не можуть бути кандидатами на роль максимально правдоподібної послідовності. Шлях декодування вибирається з якогось скороченого набору виживших шляхів. Такий декодер проте є оптимальним; в тому сенсі, що шлях декодування такий-же, як і шлях, отриманий за допомогою декодера критерії максимальної правдоподібності, діючого "грубою силою", однак попередня відмова від невдалих шляхів знижує складність декодування.

Існує декілька алгоритмів, які дають приблизні розв'язання задачі декодування на основі критерію максимального правдоподібності, включаючи послідовний і пороговий. Кожен з цих алгоритмів підходить для вузькоспеціальних завдань; проте всі вони близькі до оптимального. Алгоритм декодування Вітербі, навпаки, здійснює декодування на основі критерію максимального правдоподібності ширше, отже, є оптимальним. Це не означає, що алгоритм Вітербі в будь-якій реалізації є найкращим; при його використанні існують жорсткі умови, що накладаються на апаратне забезпечення. Алгоритм Вітербі буде описаний у наступних розділах.

**1.5.1 Алгоритм згорткового декодування Вітербі**

У 1967 році Вітербі розробив і проаналізував алгоритм, в якому, по суті, реалізується декодування, засноване на принципі максимальної правдоподібності; проте в ньому зменшується обчислювальне навантаження за рахунок використання особливостей структури конкретної решітки коду. Перевага декодування Вітербі, порівняно з декодуванням за методом "грубої сили", полягає в тому, що складність декодера Вітербі не є функцією кількості символів в послідовності кодових слів. Алгоритм включає в себе обчислення міри подібності (або відстані), між сигналом, отриманим в момент часу *ti*, і всіма шляхами решітки, що входять в кожний стан в момент часу *ti.* В алгоритмі Вітербі не розглядаються ті шляхи решітки, які, згідно з принципом максимальної правдоподібності, завідомо не можуть бути оптимальними. Якщо в один і той же стан входять два шляхи, вибирається той, який має кращу метрику; такий шлях називається вижившим. Відбір найкращих шляхів виконується для кожного стану. Таким чином, декодер поглиблюється в грати, приймаючи рішення шляхом виключення менш ймовірних шляхів. Попередню відмову від малоймовірних шляхів спрощує процес декодування. У 1969 році Омура (Omura) показав, що основу алгоритму Вітербі становить оцінка максимуму правдоподібності. Слід зазначити, що задачу відбору оптимальних шляхів можна виразити як вибір кодового слова з максимальною метрикою правдоподібності або мінімальною метрикою відстані.

**1.5.2 Приклад згорткового декодування Вітербі**

Для простоти припустимо, що ми маємо справу з каналом BSC; в такому випадку прийнятною мірою відстані буде відстань Хеммінга. Кодер для цього прикладу показаний на рис. 1.6, а решітчаста діаграма - на рис. 1.10. Для представлення декодера, як показано на рис. 1.11, можна скористатися подібною решіткою. Ми починаємо в момент часу *t1*, в стані 00 (внаслідок очистки кодера між повідомленнями декодер знаходиться в початковому стані). Оскільки в цьому прикладі можливі тільки два переходи, що вирішують інший стан, для початку не потрібно показувати всі гілки. Повна ґратчаста структура утворюється після моменту часу *t3*.Принцип роботи, що відбувається після процедури декодування можна зрозуміти, вивчивши решітку кодера на рис. 1.10 і решітку декодера, показану на рис. 1.11. Для решітки декодера кожну гілку за кожен часовий інтервал зручно помітити відстанню Хеммінга між отриманим кодовим символом і кодовим словом, відповідним до тої ж гілки з решітки кодера.

На рис. 1.11 показана послідовність повідомлень m, відповідна послідовності кодових слів U, і змінена шумом послідовність Z = 11 01 01 10 01... . Як показано на рис. 1.6, кодер характеризується кодовими словами, що знаходяться на гілках решітки кодера і попередньо відомими як кодеру, так і декодеру. Ці слова є кодовими символами, які можна було б очікувати на виході кодера в результаті кожного переходу між станами. Помітки на гілках решітки декодера накопичуються декодером в процесі. Іншими словами, коли отримано кодовий символ, кожна гілка решітки декодера позначається метрикою подібності (відстанню Хеммінга) між отриманим кодовою символом і кожним словом гілки за цей часовий інтервал. З отриманої послідовності *Z*, показаної на рис. 1.11, можна побачити, що кодові символи, отримані в (наступний) момент часу *t1* - це 11. Щоб помітити гілки декодера підходящою метрикою відстані Хеммінга в (минулий) момент часу *t1* розглянемо решітку кодера на рис. 1.10. Бачимо, що перехід між станами 00→00 породжує на виході гілки слово 00. Однак отримано 11. Отже, на решітці декодера помічаємо перехід між станами 00→00 відстанню Хеммінга між ними, а саме 2. Дивлячись знову на решітку кодера, бачимо, що перехід між станами 00→10 породжує на виході кодове слово 11, точно відповідне отриманому в момент *t*, кодовому символу. Отже, перехід на решітці декодера між станами 00→10 помічаємо відстанню Хеммінга 0. У підсумку, метрика входять в грати декодера гілок описує різницю (відстань) між тим, що було отримано, і тим, що "могло б бути" отримано, маючи кодові слова, пов'язані з тими гілками, з яких вони були передані. По суті, ці метрики описують величину, подібну кореляціям між отриманим кодовим словом і кожним з кандидатів на роль кодового слова. Таким же чином продовжуємо позначати гілки решітки декодера в міру одержання символів в кожен момент часу t1.В алгоритмі декодування ці метрики відстані Хеммінга використовуються для знаходження найбільш ймовірного (з мінімальною відстанню) шляху через грати. Сенс декодування Вітербі полягає в наступному. Якщо будь-які два шляхи зливаються в одному стані, то при пошуку оптимального шляху один з них завжди можна виключити. Наприклад, на рис. 1.12 показано два шляхи, що зливаються в момент часу *t5* в стані 00.



Рис. 1.11 - Ґратчаста діаграма декодера (ступінь кодування i/2, К = 3)

 Визначимо сумарну метрику шляху по Хеммінга для даного шляху в момент часу *t1*, як суму метрик відстаней Хеммінга гілок, по яких про ходить шлях до моменту *t1*.На рис. 1.12 верхній шлях має метрику 4, нижній - метрику 1. Верхній шлях не можна виділити як оптимальний, оскільки нижній шлях, що входить в той же стан, має меншу метрику. Це спостереження підтримується Марківською природою станів кодера. Справжній стан завершує історію кодера в тому сенсі, що попередні стани не можуть по впливати на майбутні стани або майбутні гілки на виході.



Рис. 1.12 - Метрики шляху для шляхів, що зливаються

У кожен момент часу *ti*, в решітці існує 2K-1 станів, де K - це довжина кодового обмеження, і в кожний стан може увійти два шляхи. Декодування Вітербі полягає в обчисленні метрики двох шляхів, що входять в кожний стан, та виключення одного з них. Такі обчислення проводяться для кожного з 2K-1 станів або вузлів в момент часу *ti*; потім декодер переходить до моменту часу *ti + 1* і процес повторюється. В даний момент часу метрика вижившого шляху для кожного стану позначається як метрика для цього стану в цей момент часу. Перші кілька кроків у прикладі декодування будуть наступними (рис. 1.13). Припустимо, що послідовність вхідних даних m, кодове слово U і отримана послідовність Z аналогічні показаним на рис. 1.11. Припустимо, що декодер знає вірний вхідний стан решітки. (Це припущення не є необхідним, але спрощує пояснення.) В момент часу *t1*, отримані кодові символи 11. Зі стану 00 можна перейти тільки в стан 00 або 10, як показано на рис. 1.13, а перехід між станами 00→10 має метрику гілки 0; перехід між станами 00→00 має метрику гілки 2. В момент часу *t2* з кожного стану також може виходити тільки дві гілки, як показано на рис.3.13, б.

Сумарна метрика цих гілок позначена як метрика станів Гa, Гb, Гc і Гd, відповідних кінцевим станам. В момент часу *t3* на рис. 1.13, в знову є дві гілки, що виходять з кожного стану. В результаті мається два шляхи, що входять в кожний стан, в момент часу *t4*. Один із шляхів, що входять в кожний стан, може бути виключений, а точніше - це шлях, що має велику сумарну метрику шляху. Якби метрики двох вхідних шляхів мали однакове значення, то шлях, який буде виключатися, вибирався б довільно.

Виживший шлях в кожному стані показаний на рис. 1.13, р У цій точці процесу декодування мається тільки один виживший шлях, який називається повною гілкою, між моментами часу *t1* і *t2*.Отже, декодер тепер може вирішити, що між моментами *t1* і *t2* відбувся перехід 00→10. Оскільки перехід викликається одиничним вхідним бітом, на виході декодера першим бітом буде одиниця.

Тут легко можна простежити процес декодування виживших гілок, оскільки гілки решітки показані пунктирними лініями для вхідних нулів і суцільною лінією для вхідних одиниць. Зауважимо, що перший біт НЕ декодується, поки обчислення метрики шляху не пройде далі вглиб решітки. Для звичайного декодера така затримка декодування може виявитися раз у п'ять більше довжини кодового обмеження в бітах.

На кожному наступному кроці процесу декодування завжди буде два шляхи для кожного стану; після порівняння метрик шляхів один з них буде виключений. Цей крок в процесі декодування показаний на рис. 1.13, д. В момент *t5* знову є по два вхідних шляхи для кожного стану, і один шлях з кожної пари підлягає виключенню. Ті шляхи, що вижили на момент *t5* показані на рис. 1.13. Зауважимо, що в нашому прикладі ми ще не можемо прийняти рішення щодо другого вхідного інформаційного біта, оскільки ще залишається два шляхи, що виходять в момент *t2* зі стану у вузлі 10. В момент часу *t6* на рис. 1.13, знову можемо бачити структуру шляхів, що зливаються, а на рис. 1.13, з) - виживших шляхи на момент *t6*.Тут же, на рис. 1.13, з), на виході декодера в якості другого декодованого біта показана одиниця як підсумок єдиного, що залишився шляху між точками *t2* і *t3*. Аналогічним чином декодер продовжує заглиблюватися в грати і приймати рішення, що стосуються інформаційних бітів, усуваючи всі шляхи, окрім одного.

Відсікання (шляхів, що сходяться) в решітці гарантує, що у нас ніколи не буде шляхів більше, ніж станів. У цьому прикладі можна перевірити, що після кожного відсікання (рис. 1.13) залишається тільки 4 шляхи. Якщо порівняти це з намаганням застосувати "грубу силу" (без залучення алгоритму Вітербі) при використанні для отримання послідовності принципу максимальної правдоподібності, то в цьому випадку число можливих шляхів (відповідає можливим варіантам послідовності) є ступеневою функцією довжини послідовності. Для двійкової послідовності кодових слів з ​​довжиною кодових слів L мається 2L можливих послідовностей.





Рис. 1.13 - Вибір вижили шляхів: а) ті, що вижили на момент t2; б) ті, що вижили на момент t3; в) порівняння метрик в момент t4; г) ті, що вижили на момент t5; д) порівняння метрик в момент t5; е) ті, що вижили на момент t5; ж) порівняння метрик в момент t6; з) вижили на момент t6

**1.5.3 Реалізація декодера**

В контексті ґратчастої діаграми, показаної на рис. 1.11, переходи за один проміжок часу можна згрупувати в 2v-1 непересічні комірки; кожна комірка буде зображувати чотири можливих переходи, причому  називається пам'яттю кодера (encoder memory). Якщо К = 3, то v = 2, і, отже, ми маємо 2v-1 = 2 комірки. Ці комірки показані на рис. 1.14, де букви а, b, с і d позначають стани в момент *ti* , а а', b', c' і d' - стани в момент часу *ti + 1*.Для кожного переходу зображена метрика гілки , індекси якої означають перехід зі стану х у стан у. Ці комірки і відповідні логічні елементи, які коригують метрики зі станів {Гx}, де х означає конкретну відстань стану, представляють основні складові елементи декодера.



Рис. 1.14 - Приклади комірок декодера

**1.5.4 Процедура складання, порівняння і вибору**

Повернемося наприклад до двох комірок з К = 3. На рис. 1.15 показаний логічний блок, відповідний комірці 1. Логічна схема здійснює спеціальну операцію, яка називається додавання, порівняння і вибір (add-compare-select - ACS). Метрика стану Гa обчислюється шляхом додавання метрики попереднього стану а, Гa' до метрики гілки  і метрики попереднього стану c, Гc до метрики гілки . Це дасть в результаті дві метрики шляхів як кандидатів для нової метрики стану Г a '.

Обидва кандидати порівнюються в логічному блоці, показаному на рис. 1.15. Найбільш правдоподібна з двох метрик шляхів (з найменшою відстанню) запам'ятовується як нова метрика стану Гa' для стану а'. Також зберігається нова історія шляхівдля стану a, де історія шляху інформації для даного стану, доповнена відомостями про вижив шляху.

На рис. 1.15 також показана логічна схема ACS для комірки 1, яка дає нову метрику стану Гb' нову історію стану . Операція ACS аналогічним чином здійснюється і для колій в інших осередках. Вихід декодера складають останні біти на шляхах з найменшими метриками станів.

**1.5.5 Вид процедури складання, порівняння і вибору на решітці**

Розглянемо той же приклад для опису декодування на основі алгоритму Вітербі. Нехай послідовність повідомлень має вигляд m = 1 1 0 1 1, послідовність кодових слів U = 11 01 01 00 01, а прийнята послідовність Z = 11 1 01 10 01.



Рис. 1.15 - Логічний блок, призначений для здійснення операцій додавання, порівняння і вибору

 Ґратчаста діаграма декодування, аналогічна показаній на рис. 1.11, зображена на рис. 1.16. Метрика гілки, яка описує кожну гілку, це відстань Хеммінга між прийнятим кодовим символом і відповідним кодовим словом з решітки кодера.

Ще на решітці (рис. 3.16) показані значення кожного стану х в кожен момент *t2-t6*, метрика стану яких позначена Гx. Операція ACS виконується після появи двох переходів, що входять в стан, тобто для моменту *t4* і пізніших.

Наприклад, в момент часу *t+4* значення метрики стану для стану а обчислюється підсумовуванням метрики стану Гa = 3 в момент *t3* і метрики гілки  = 1, що в підсумку дає значення 4. У той же час до метриці стану Гc = 2 в момент часу *t3* додається метрика гілки = 1, що дає значення 3. В ході процедури ACS відбувається відбір найбільш правдоподібною метрики (з мінімальним відстанню), тобто нової метрики стану; тому для стану a в момент *t4* новою метрикою стану буде Га' =3. Відібраний шлях зображений жирною лінією, а шлях, який був відкинутий, показаний світлою лінією.

На рис. 1.16 на решітці зліва направо показані всі метрики станів. Переконаємося, що в будь-який момент часу значення кожної метрики стану виходить підсумовуванням метрики стану, поєднаного з попереднім станом вздовж відібраного шляху (жирна лінія), і метрики гілки, що з'єднує ці стани.

Через деякий час на вихід декодера будуть подані виживші гілки, що прослідкували до найперших бітів. Щоб показати це, подивимося на рис. 1.16 в момент *t6*. Бачимо, що значення метрики стану, що відповідає мінімальній відстані, рівно 1. Відібраний шлях можна простежити зі стану d назад, до моменту *t1* і переконатися, що декодоване повідомлення збігається з вихідним. Відзначимо, що пунктирні і суцільні лінії відповідають двійковим одиниці та нулю.

**1.5.6 Пам'ять шляхів і синхронізація**

Вимоги до пам'яті декодера, працюючого згідно з алгоритмом Вітербі, ростуть із збільшенням довжини кодового обмеження як статична функція. Для коду зі ступенем кодування 1/n після кожного кроку декодування декодер тримає в пам'яті набір з 2 К-1 шляхів.



Рис. 1.16. Операція складання, порівняння і вибору при декодуванні за алгоритмом Вітербі

 З високим ступенем ймовірності можна стверджувати, що при значному перевищенні існуючої на даний момент глибини декодування ці шляху не будуть взаємно непересічними. Всі 2К-1 шляхи ведуть до певної гілки, яка врешті-решт розгалужується на різні стани. Тому, якщо декодер зберігає історію 2К-1 шляхів, найперші біти на всіх шляхах будуть однакові. Отже, простий декодер має фіксований обсяг історії шляхів та видає найперші біти довільного шляху щоразу, коли просувається на один рівень вглиб решітки. Необхідний обсяг шляхів, що зберігаються буде дорівнює наступному:



 Тут h - довжина історії шляху інформаційного біта на стан. При уточненні, яке проводиться для мінімізації h, замість самих перших бітів довільних шляхів на виході декодера використовуються самі перші біти найбільш ймовірних шляхів. Було показано, що значення h, рівного 4 або 5 довжинам кодового обмеження, достатньо, щоб характеристики декодера були близькі до оптимальних. Необхідний обсяг пам'яті і є основним обмеженням при розробці декодерів, що працюють згідно з алгоритмом Вітербі. В декодерах, що випускаються серійно довжина кодового обмеження дорівнює величині порядку К = 10. Спроба підвищити ефективність кодування за рахунок збільшення довжини кодового обмеження викликає експоненціальне зростання вимог до пам'яті (і складності).

Синхронізація кодових слів гілок - це процес визначення початку слова гілки в прийнятій послідовності. Таку синхронізацію можна здійснити, не додаючи нову інформацію до потоку переданих символів, оскільки можна бачити, що, поки прийняті дані не синхронізовані, у них непомірно висока частота появи помилок. Отже, синхронізацію можна здійснити просто: потрібно проводити супутнє спостереження за рівнем частоти появи помилок, тобто нас повинна цікавити частота, при якій збільшуються метрики станів, або частота, при якій зливаються виживші шляхи на решітці. Параметр, за яким стежать, порівнюється з пороговим значенням, після чого відповідним чином здійснюється синхронізація.