

Майже у всіх формах електронних систем зв'язку і зберігання даних використовуються коди з виправленням помилок. Коди з виправленням помилок компенсують ненадійність передачі інформації, що вноситься, в цих системах, шляхом внесення надмірності в потік даних. Математичні основи виправлення помилок були закладені Шенноном. Шеннон розробив математичну ідею каналу, відповідно до якої спотворення сигналів в системах зв'язку моделюється як стохастичний процес. Найбільш важливим результатом Шеннона є теорема про канал з шумом, яка визначає для каналу «пропускну здатність» - якість, яка конкретно вказує максимальну швидкість, з якою можна надійно передавати інформацію по каналу. Надійна передача на швидкостях, що наближаються до пропускну здатності, вимагає використання кодів з виправленням помилок. Таким чином, коди з виправленням помилок призначені для досягнення достатньої надійності при одночасному як можна більшому наближенні до пропускну здатності. Складність втілення коду з виправленням помилок є додатковим фактором, який набирає чинності в практичних прикладеннях кодів з виправленням помилок. Останні досягнення в системах кодування з виправленням помилок, що є результатом винаходу турбо-кодів і подальшого повторного відкриття і розробки кодів з низькою щільністю контролю за парністю (LDPC), дозволяють запропонувати системи кодування з гнучкою складністю, що дозволяють досить близько підійти до пропускну здатності за Шенноном.

Коди з LDPC добре відображаються дводольними графами, які часто називають графами Таннера, такими, як граф 100, показаний на Фіг.1. У графах Таннера одна множина вершин - вершини 102 «змінних» - відповідає бітам кодового слова, а інша множина вершин - вершини 106 «обмежень», які іноді називають «контрольними» вершинами, - відповідають множині обмежень контролю за парністю, які і визначають код. Ребра 104 в графі зв'язують вершини змінних з вершинами, обмежень. Вершина змінної і вершина обмеження називаються сусідами, якщо вони з'єднані ребром в графі. Звичайно передбачають, що дві вершини з'єднані щонайменше одним ребром. Коди з LDPC можна еквівалентно представити, скориставшись матрицею 202 контролю за парністю. На Фіг.2 наведений приклад представлення матриці контролю за парністю, при цьому вектор «х», позначений позицією 206, є кодовим словом, якщо і тільки якщо $Hx=0$.

З кожною вершиною змінної зв'язаний один біт кодового слова. У деяких випадках, деякі з цих бітів можуть бути «виколоті». Виколоті біти можуть бути бажаними в деяких структурах кодів, і вони виключаються з кодового слова, що передається.

Послідовність бітів, що має однозначну відповідність з послідовністю вершин змінних, є кодовим словом коду, якщо і тільки якщо для кожної вершини обмеження біти, що сусідують з цим обмеженням (за допомогою свого зв'язку з вершинами змінних) в сумі дають нуль по модулю два, тобто є парна кількість таких бітів.

Декодера та алгоритми декодування, що використовуються для декодування кодових слів, які складаються з кодів з LDPC, працюють, обмінюючись повідомленнями в межах графа по ребрах і оновлюючи ці повідомлення шляхом проведення обчислень у вершинах на основі вхідних повідомлень. Такі алгоритми далі будуть узагальнено іменуватися алгоритмами передачі повідомлень. Кожна вершина змінної в графі спочатку забезпечена бітом м'якого рішення, який називається «значенням, що приймається» і вказує оцінку відповідного значення біта, що визначається шляхом спостережень, наприклад, за каналом зв'язку. В ідеальному випадку, оцінки для окремих бітів статистично незалежні. На практиці, цей ідеал може порушуватися, а часто - і порушується. Набір значень, що приймаються, складає «слово, що приймається». У контексті цієї заявки, словом, що приймається, можна визначити сигнал, що спостерігається, наприклад, приймачем в системі зв'язку.

Кількість ребер, з'єднаних з вершиною, тобто вершиною змінної або вершиною обмеження, називається «ступінь» вершини. «Однорідним» графом або кодом є той, для якого всі вершини змінних мають один і той самий ступінь, скажемо, j , і всі вершини обмежень мають один і той самий ступінь, скажемо, k . В цьому випадку можна сказати, що код є (j, k) -однорідним кодом. Це були коди, які першим розглянув Галлагер (Gallager, 1961). На відміну від «однорідного» коду, неоднорідний код має вершини обмежень i /або вершини змінних з різними ступенями. Наприклад, деякі вершини змінних можуть мати ступінь 4, інші - ступінь 3, а ще одні - ступінь 2.

Хоча представлення i /або втілення неоднорідних кодів може виявитися складнішим, показано, що неоднорідні коди з LDPC можуть забезпечити чудові робочі параметри виправлення та виявлення помилок в порівнянні з однорідними кодами з LDPC.

Потрібно зрозуміти, що слова, що приймаються, генеровані відповідно до кодування кодами з LDPC, можна обробляти, проводячи на цих словах операцію декодування кодів з LDPC, наприклад, операції виправлення та виявлення помилок, для генерування відновленої версії вихідного кодового слова. Відновлене кодове слово можна потім піддати декодуванню даних, щоб відновити вихідні дані, які були закодовані. Процес декодування даних може бути, наприклад, простим вибором конкретної підмножини бітів з відновленого кодового слова.

Як згадувалося вище, операції декодування кодів з LDPC звичайно передбачають застосування алгоритмів передачі повідомлень. Існує багато потенційно корисних алгоритмів передачі повідомлень, так що використання таких алгоритмів не обмежується декодуванням кодів з LDPC.

Щоб полегшити розуміння винаходу, що розглядається в нижченаведених розділах, наведемо тепер короткий математичний опис довірчого поширення.

Довірче поширення для (двійкових) кодів з LDPC можна виразити таким чином. Повідомлення, що передаються по ребрах графа, інтерпретуються як логарифмічні правдоподібності $\log(p_0/p_1)$, де p_x означає імовірність того, що біт буде приймати значення «х». Біти м'якого рішення, що надаються декодеру приймачем, також задаються в формі логарифмічної правдоподібності. Таким чином, значення, що приймаються, тобто елементи слова, що приймається, являють собою логарифмічні правдоподібності відповідних бітів, зумовлені спостереженням бітів, що надаються каналом зв'язку. У загальному випадку, повідомлення m представляє логарифмічну правдоподібність m , а значення «у», що приймається, представляє логарифмічну правдоподібність «у». Для проколотих бітів значення «у» логарифмічної правдоподібності, що приймається, задають рівним 0, вказуючи, що $p_0=p_1=1/2$.

Розглянемо правила передачі повідомлень згідно з довірчим поширенням.

Повідомлення позначаються символом m^{C2V} , якщо це повідомлення від контрольних вершин до вершин змінних, і символом m^{V2C} , якщо це повідомлення від вершин змінних до контрольних вершин. Розглянемо вершину змінної з d ребрами. Нехай для кожного ребра $j=1, \dots, d$ символ $m^{C2V}(i)$ означає вхідне повідомлення на ребрі i . При ініціалізації процесу декодування задамо $m^{C2V}=0$ для кожного ребра. У загальному випадку, вихідні повідомлення з вузлів змінних задаються таким чином:

$$m^{V2C}(j) = y + (\sum_{i=1}^d m^{C2V}(i)) - m^{C2V}(j).$$

Вихідне кодоване значення y кожного рішення з вершини (не повідомлення на ребрі), яке відповідає цій операції задається таким чином:

$$x_{out} = y + (\sum_{i=1}^d m^{C2V}(i)).$$

Вихідне тверде рішення, зв'язане з цим значенням, що виводиться, одержується із знаку для x_{out} .

У контрольних вершинах часто зручніше представляти повідомлення з використанням їх «знаків» і модулів. Тому нехай для повідомлення m вираз $m_p \in GF[2]$ означає «парність» повідомлення, тобто $m_p = 0$, якщо $m \geq 0$, і $m_p = 1$, якщо $m < 0$. Крім того, нехай вираз $m_p \in [0, \infty]$ означає модуль повідомлення m . Таким чином, маємо $m = -1^{m_p} m_p$. У контрольній вершині, оновлення для повідомлень m_p і m_r розділені. Для контрольної вершини ступеня d , маємо:

$$m_p^{C2V}(j) = (\sum_{i=1}^d m_p^{V2C}(i)) - m_p^{V2C}(j),$$

де все складання проводиться по $GF[2]$, і

$$m_r^{C2V}(j) = F^{-1}((\sum_{i=1}^d F(m_r^{V2C}(i))) - F(m_r^{V2C}(j))),$$

де складання є дійсним, і визначаємо функцію $F(x) = \ln \tanh(x/2)$. Зазначимо, що F являє собою і власне зворотне значення, тобто $F^{-1}(x) = F(x)$.

Алгоритмом, що часто згадується в літературі з кодів з LDPC, є так званий алгоритм мінімальної суми. Операцію оновлення в контрольних вершинах в цьому алгоритмі можна математично виразити таким чином:

$$m_r^{C2V}(j) = \min \{ (\bigcup_{i=1}^d m_r^{V2C}(i)) \setminus m_r^{V2C}(j) \}.$$

Таким чином, надійність повідомлення $C2V$ дорівнює мінімальній надійності повідомлень $V2C$, що входять з інших ребер. Щоб втілити цей алгоритм, достатньо зберегти в контрольній вершині найменшу і другу найменшу надійність (вони можуть бути одним і тим самим значенням, якщо це значення виникає щонайменше для двох вхідних повідомлень), і найменування ребра, що надає вхідне повідомлення найменшої надійності. Надійність вихідного повідомлення $C2V$ на ребрі, від якого виходить найменше надійне вхідне повідомлення, дорівнює другій найменшій надійності вхідних повідомлень, а надійність вихідного повідомлення $C2V$ на всіх інших ребрах дорівнює найменшій надійності.

У патенті США № 6633856 описана архітектура декодера кодів з LDPC. У цій архітектурі повідомлення від контрольних вершин до вершин змінних зберігаються в пам'яті. Якщо, наприклад, повідомлення містить 5 біт, а контрольна вершина має ступінь K , то ємність пам'яті, що використовується для цих повідомлень, становить $5K$ біт.

З точки зору втілення і витрат, в загальному випадку бажано втілювати декодер кодів з LDPC так, щоб він був відносно простим в проектуванні і вимагав відносно малої кількості апаратних засобів. Основним компонентом в багатьох конструкціях декодерів є пам'ять. Було б бажано, якби обсяг пам'яті, необхідний для втілення декодера, підтримувався б малим або мінімальним, щоб таким чином знизити витрати на апаратні засоби.

Хоча зменшення пам'яті є важливим моментом, при зниженні обсягу пам'яті часто потрібно уникати конструкції, яка могла б вносити неприйнятні затримки при обробці, що призводило б до неможливості відповідності одному або декільком реально існуючим часовим обмеженням декодування.

З огляду на розгляд, проведений вище, повинно стати очевидним, що для втілення декодерів кодів з LDPC і/або виробництва декодера кодів з LDPC, виконаного з можливістю проведення операцій декодування з відносно малим обсягом пам'яті, були б бажані ефективні з використання пам'яті способи та пристрої. Крім того, бажані і були б вигідні способи, що дозволяють уникнути внесення зайвих затримок в процес декодування кодів з LDPC, що передбачає втілення ефективного з використання пам'яті декодера.

Даний винахід стосується способів та пристроїв для проведення операцій декодування кодів з LDPC з ефективним використанням пам'яті. Різні ознаки даного винаходу стосуються способів обробки контрольних вершин і пристроїв для здійснення цих способів, які втілюються з ефективним використанням пам'яті, наприклад, з використанням методів ущільнення і/або розущільнення інформації одного або більше повідомлень. Додаткові ознаки даного винаходу стосуються запобігання і/або зменшення затримок в ефективних з використання пам'яті декодерах кодів з LDPC, тобто в декодері того типу, який описується в даній заявці, за рахунок використання кодів з LDPC, що мають структуру коду, яка запобігає внесенню значних затримок в процес декодування.

Автор даної заявки зрозумів, що обчислення контрольних вершин в декодерах кодів з LDPC має ту властивість, що надійність вихідних повідомлень може приймати тільки два значення, а одне з цих значень видається тільки по одному ребру, і що цю властивість можна використати для ефективного зберігання інформації повідомлень з контрольних вершин і для ефективного втілення блоку обробки контрольних вершин. Скориставшись цією властивістю, можна не зберігати повне повідомлення під час обробки контрольних вершин, що дозволяє зменшити місткість запам'ятовуючого пристрою до об'єму, який необхідний

для обробки і побудови повідомлень, що виводяться, коли обробка контрольних вершин, яка відповідає деякій вершині, завершена. Користуючись ущільненням повідомлень, вимоги до пам'яті для блоку обробки контрольних вершин можна істотно знизити в порівнянні з втіленнями, коли ущільнення повідомлень відповідно до винаходу не використовується.

Різні ознаки даного винаходу стосуються блоків обробки контрольних вершин. У цих блоках обробки інформація, яка відповідає повідомленням зв'язаним з кожною контрольною вершиною, зберігається в ущільненому форматі. Для цього використовується пам'ять станів контрольних вершин, що включає в себе, наприклад, один елемент даних для кожної контрольної вершини. Інформація про стан для контрольної вершини включає в себе інформацію, що створюється з повідомлень, які вводяться, для цієї конкретної контрольної вершини, а не всього набору повідомлень для цієї контрольної вершини. Таким чином, інформація про стан являє собою ущільнену інформацію набору повідомлень. Ущільнена інформація, яка відповідає кожній контрольній вершині, оновлюється при прийомі кожного повідомлення, що вводиться, наприклад, повідомлення від вершини змінної до контрольної вершини. Прийом повідомлень, що вводяться, можливий в будь-якому порядку, наприклад, повідомлення, що вводяться для кожної окремої контрольної вершини не обов'язково одержувати та обробляти послідовно. Таким чином, відповідно до винаходу, можна обробити повідомлення, що вводиться, яке відповідає одній контрольній вершині, потім - повідомлення, що вводиться, яке відповідає іншій контрольній вершині, і не потрібно спочатку одержати всі повідомлення, що вводяться, для іншої контрольної вершини. Це забезпечує прийом та обробку повідомлень від вершин змінних до контрольних вершин в порядку ребер вершин змінних, протилежному тому порядку, в якому ребра з'являються на стороні контрольних вершин графа коду з LDPC. Таким чином, якщо передбачити появу повідомлень в елементі обробки вершин змінних в порядку вершин змінних, то відповідає необхідність переупорядкування генерованих повідомлень перед обробкою в блоці обробки контрольних вершин згідно з винаходом.

Після обробки повного набору повідомлень від вершин змінних до контрольних вершин, який відповідає окремій контрольній вершині, здійснюється доступ до інформації повідомлень, яка відповідає повному набору повідомлень, зв'язаному з контрольною вершиною, і обробка цієї інформації, наприклад, її піддають процесам розуцільнення, які також називаються в даній заявці процесами витягування. Процес витягування обумовлює генерування повного набору повідомлень від контрольних вершин до вершин змінних, що формуються окремою контрольною вершиною, наприклад, так, як диктується конкретною структурою коду, що втілюється, і положенням окремої контрольної вершини в цій структурі коду.

У деяких варіантах здійснення, стан, що зберігається для кожної контрольної вершини включає в себе два значення, наприклад, перше і друге значення. Ці значення можуть бути значеннями модулів повідомлень. Стан також включає в себе інформацію про положення ребер, зв'язану з одним із згаданих значень. Ця інформація представляє стан, що використовується для генерування модульної частини вихідного повідомлення, яке формується конкретною контрольною вершиною. У доповнення до інформації про модуль повідомлення, яка являє собою інформацію про надійність, для кожної контрольної вершини зберігається значення накопиченого знакового біта. Цей накопичений знаковий біт генерується за допомогою операції «виключаюче АБО» над знаковим бітом кожного повідомлення, що вводиться, яке поступає в контрольну вершину, яка обробляється, з останнім значенням генерованого накопиченого знакового біта, щоб генерувати знаковий біт для кожного повідомлення, що виводиться.

В одному конкретному варіанті здійснення, додаткову інформацію про знаковий біт зберігають для кожного повідомлення, що вводиться, яке одержують відповідно до контрольної вершини. У такому варіанті здійснення знаковий біт, що вводиться, зберігається для кожного повідомлення, що приймається по ребру контрольної вершини. Тому в такому варіанті здійснення - крім накопиченого знакового біта для кожної контрольної вершини - зберігають знаковий біт, що вводиться, для кожного ребра контрольної вершини, при цьому кожне ребро відповідає відмінному повідомленню, що вводиться.

Хоча повний набір повідомлень, що вводяться, який відповідає окремій контрольній вершині, треба обробити перед генеруванням повідомлень, що виводяться, з цієї конкретної окремої контрольної вершини, за рахунок використання структури коду, в якій контрольні вершини не одержують повідомлення, що вводяться, з вершин змінних, які будуть оброблятися набагато пізніше протягом процесу декодування, повідомлення, що виводяться щонайменше для деяких контрольних вершин можна генерувати без необхідності обробки повідомлень з повної множини вершин змінних, представлених в структурі графа, що втілюється. Користуючись кодом, який враховує вигоди здійснення обробки контрольних вершин, присутніх в одній частині графа, таким чином, що вони не залежать від повідомлень з вершин змінних, які будуть оброблятися набагато пізніше, можна зменшити і/або мінімізувати часові затримки, зв'язані з генеруванням повідомлень контрольних вершин, що виводяться.

Повідомлення контрольних вершин, що виводяться, для деякої контрольної вершини можна генерувати відразу ж після того, як оброблений повний набір повідомлень, що вводяться, наприклад, по одному повідомленню для кожного ребра контрольної вершини, що вводиться. Щоб генерувати повідомлення контрольної вершини, що виводиться, яке служить як повідомлення, що вводиться, для вершини змінної, блок обробки контрольних вершин згідно з даним винаходом зчитує стан контрольної вершини, зв'язаний з контрольною вершиною, про повідомлення, що виводиться, якої йде мова. Виходячи з інформації, що зберігається, про стан і модулі, наприклад, про перше і друге значення модулів та ідентифікатор ребра, блок обробки контрольних вершин обробляє стан з тим, щоб витягнути, наприклад - генерувати, завершене повідомлення, що виводиться, для одного ребра, наприклад, повідомлення, що йде від контрольної вершини до вершини змінної. Цей процес буде повторюватися для кожного ребра контрольної вершини доти, доки не буде згенерований повний набір повідомлень, що виводяться. Обробка інформації про стан контрольної вершини, по суті, являє собою операцію розуцільнення.

У деяких варіантах здійснення використовують алгоритм мінімальної суми для зберігання інформації про модуль (надійність) повідомлення в ущільненому стані. У таких варіантах здійснення для кожної контрольної

вершини зберігають мінімальне значення модуля повідомлення, друге мінімальне значення модуля повідомлення та інформацію, яка вказує ребро, якому відповідає мінімальне значення модуля повідомлення. Ця інформація є додатковою до інформації накопиченого знакового біта. Ця інформація оновлюється кожен раз при прийомі повідомлення, що вводиться, яке відповідає конкретній контрольній вершині, якій відповідає інформація, доти, доки не буде оброблене кожне з повідомлень, що вводяться, для конкретної контрольної вершини.

Щоб генерувати модульну частину вихідного повідомлення, яка відповідає ребру, це ребро порівнюють з ідентифікатором ребра, що зберігається, який вказує ребро, по якому одержане мінімальне значення повідомлення. Якщо ребро, для якого генерується вихідне повідомлення, не відповідає ідентифікатору ребра, що зберігається, мінімальне значення повідомлення використовують як модуль вихідного повідомлення. Якщо ідентифікатор ребра, що зберігається, відповідає ребру, для якого генерується вихідне повідомлення, то використовують друге мінімальне значення як модуль вихідного повідомлення. Таким чином, модулі вихідних повідомлень, що генеруються контрольною вершиною, будуть мати або мінімальне значення, або друге мінімальне значення, при цьому друге мінімальне значення видається по єдиному ребру, яке обумовило подачу мінімального значення модуля, що приймається контрольною вершиною.

З обчислювальної точки зору, це зводиться до проведення операції мультиплексування (вибору) між двома можливими надійностями, А та В, де А та В - два можливих значення модулів вихідних повідомлень, наприклад, мінімальне або друге мінімальне значення модулів повідомлень, що приймаються, які одержані конкретною контрольною вершиною. Друге значення В модуля вихідного повідомлення вибирають для модульної частини вихідного повідомлення, якщо ребро (з'єднання вершини змінної з контрольною вершиною) відповідає індексу ребра, що зберігається, який вказує те ребро, по якому було одержане, значення А модуля (мінімальне значення). В іншому випадку, видається мінімальне значення А амплітуди.

Знаковий біт для вихідного повідомлення можна генерувати багатьма шляхами. В одному можливому варіанті здійснення, значення накопиченого знакового біта, яке відповідає деякій контрольній вершині, об'єднується, наприклад, за допомогою операції «виключаюче АБО», із значенням знакового біта, що приймається і зберігається, яке відповідає ребру, для якого генерується вихідне повідомлення, щоб набутися значення знакового біта вихідного повідомлення для такого ребра. Цей процес повторюється подібно до процесу генерування вихідного повідомлення для кожного ребра, для якого генерується вихідне повідомлення.

З точки зору втілення, спосіб обробки контрольних вершин згідно з даним винаходом має перевагу зменшення обсягу пам'яті, необхідного для втілення обробки контрольних вершин, за допомогою використання ущільнення повідомлень. З точки зору втілення, це може виявитися істотним, зокрема, тоді, коли контрольні вершини можуть мати великі кількості ребер, як у випадку з багатьма робастними кодами з LDPC, які використовуються в цей час і, ймовірно, будуть використовуватися в майбутньому.

Розглянемо, наприклад, випадок 5-бітових повідомлень, які включають в себе один знаковий біт і 4 біти модуля. При вищеписаних допущеннях, інформацію, що виводиться, з кожної контрольної вершини можна стиснути приблизно до $K + 8 \log_2 K$ біт, де K - кількість вхідних і вихідних повідомлень. У прикладі з 5-бітовим повідомленням, K біт пам'яті будуть використовуватися для зберігання знакових бітів, які відповідають кожному з K повідомлень, що вводяться, 8 біт будуть використовуватися для зберігання кожного з двох можливих значень модулів, які можна передбачити в повідомленнях, що виводяться, наприклад, мінімальне значення модуля повідомлення, що вводиться, і наступне найменше значення модуля повідомлення, що вводиться, а $\log_2 K$ біт використовуються для вказівки ребра (повідомлення), прийом по якому повинен відбуватися з другим значенням надійності, тоді як прийом по інших ребрах буде відбуватися з мінімальним значенням модуля повідомлення, що вводиться. Якщо число K велике, як буде у випадку кодів з LDPC для великих швидкостей передачі, то використання цього методу зберігання інформації про повідомлення може призвести до істотної економії в порівнянні з втіленнями, що передбачають зберігання повних множин бітів, які відповідають кожному повідомленню, що приймається, як частина процесу генерування повідомлень, що виводяться.

Хоча для генерування вихідних повідомлень на основі стану, що зберігається, який включає в себе, наприклад, мінімальний модуль повідомлення, другий мінімальний модуль повідомлення, положення ребра, яке відповідає мінімальному модулю повідомлення, і знакові біти, що зберігаються, які відповідають кожному з ребер, необхідна деяка додаткова обробка в порівнянні з варіантами здійснення, передбачаючих зберігання повних повідомлень, але - в свою чергу - можлива істотна економія пам'яті.

Блок обробки контрольних вершин згідно з даним винаходом має додаткову перевагу, яка полягає в тому, що повідомлення від вершин змінних до контрольних вершин, що служать як контрольні вершини, що вводяться для блоку обробки, можна приймати та обробляти поза зв'язком з яким-небудь конкретним порядком обробки повідомлень. Повідомлення, що виводяться, генеруються після завершення обробки кожного з повідомлень, що вводяться, для контрольної вершини.

Численні додаткові ознаки та переваги винаходу стануть зрозумілими з нижченаведеного докладного опису.

Короткий опис креслень

На Фіг.1 представлений граф, що ілюструє можливий код з LDPC і включає в себе десять вершин змінних і п'ять контрольних вершин.

На Фіг.2 наведене альтернативне представлення коду з LDPC згідно з Фіг.1, яке показує код за допомогою використання матричного представлення як альтернативи представленню у вигляді графа, показаному на Фіг.1.

На Фіг.3 зображений блок обробки вершин обмежень, втілений відповідно до винаходу.

На Фіг.4 зображений декодер кодів з LDPC, втілений відповідно до даного винаходу.

На Фіг.5 зображений ще один декодер кодів з LDPC, втілений з використанням N паралельних елементів обробки вершин обмежень і вершин змінних відповідно до даного винаходу.

На Фіг.6 зображена можлива структура коду з LDPC, яку можна використати для керування декодування, наприклад, в декодері згідно з Фіг.4.

На Фіг.7 зображена ще одна можлива структура коду з LDPC, яку можна використати для керування декодування, наприклад, в можливому декодері згідно з Фіг.4, відповідно до даного винаходу.

На Фіг.8 зображені результати здійснення операції декодування на множині значень, що вводяться, з використанням декодера згідно з Фіг.4 і структури коду з LDPC, зображеної на Фіг.7.

Нижченаведені (3) споріднені заявки згадуються в цьому описі для довідок і повинні розглядатися як частина даної заявки: заявка № 09/975331 на патент США, подана 10 жовтня 2001 р. під назвою «Methods and apparatus for decoding ldpc codes» («Способи та пристрої для декодування кодів з LDPC»); заявка № 10/117264 на патент США, подана 4 квітня 2002 р. під назвою «Node processors for use in parity check decoders» («Процесори вершин, призначені для використання в декодерах з контролем за парністю»); і заявка № 10/618325 на патент США, подана 11 липня 2003 р. під назвою «Methods and apparatus for decoding LDPC codes» («Способи та пристрої для кодування кодів з LDPC»).

Різні варіанти здійснення даного винаходу стосуються способів та пристроїв, які забезпечують просте, наприклад, з малою складністю апаратних засобів, втілення архітектури декодера кодів з LDPC. Способи та пристрої згідно з винаходом мають перевагу додержання закономірності, відповідно до якої - при деяких алгоритмах - надійності, що посиляються від контрольної вершини, приймають одне з двох можливих значень, і одне з цих значень посиляється вздовж тільки одного ребра повідомлення. Це випадок, наприклад, алгоритму мінімальної суми. Ті самі умови можуть існувати для інших алгоритмів, які можна використати для зберігання інформації повідомлень, виражених кодами з LDPC, в стислій формі, наприклад, алгоритмів, які - нарівні з алгоритмом мінімальної суми - мають ту властивість, що надійності, які посиляються від деякої контрольної вершини, можуть приймати тільки два можливих значення, а одне з цих значень посиляється якраз по тому ребру повідомлення, яке зв'язане з контрольною вершиною.

Відповідно до винаходу, з кожною контрольною вершиною в структурі коду з LDPC, що використовується для керування декодування, зв'язаний стан S . Цей стан буде включати в себе інформацію про надійність (модуль повідомлення) для вхідних повідомлень в поточній ітерації декодування, причому ця інформація буде використовуватися для генерування вихідних повідомлень. Нехай символ S_k означає стан, який приблизно включає в себе повідомлення $m_1 \dots, m_k$. Тоді, задавши функцію G оновлення стану, стан для заданої контрольної вершини, виникаючий внаслідок обробки повідомлення, що приймається, від вершини змінної до контрольної вершини, яке відповідає контрольній вершині, можна виразити таким чином:

$$S_{k+1} = G(m_{k+1}, S_k).$$

Відповідно до різних варіантів здійснення винаходу, операція оновлення стану здійснюється над станом, який представляє модульну частину набору повідомлень, яка відповідає контрольній вершині, в ущільненій формі. Таким чином, операція оновлення стану передбачає ущільнення вхідних повідомлень.

У випадку втілення, в якому використовується алгоритм мінімальної суми для зберігання інформації повідомлення в ущільненому форматі, стан повідомлення, що зберігається, може бути представлений, наприклад, в формі (m_A, m_B, A, s) . Якщо m_A - мінімальна надійність вхідного повідомлення (його мінімальний модуль), що розглядалася досі контрольною вершиною, якій відповідає згаданий стан, а m_B - друга найменша надійність, що розглядалася досі, то A означає ребро, по якому проходило вхідне повідомлення надійності m_A , і тим самим вказує, яке ребро повідомлення обумовило подачу мінімального значення m_A , а s означає операцію «виключаюче АБО» над знаками вхідних повідомлень, які відповідають контрольній вершині, якій відповідає інформація про стан.

Здійснюючи оновлення інформації про стан, яка відповідає контрольним вершинам, з використанням функції G , а також забезпечуючи втілення декодера, в якому виявляються можливими зберігання і вибірка стану, який відповідає окремим контрольним вершинам, для оновлення вхідними повідомленнями залежно від того, в яку контрольну вершину направлено повідомлення, можна зробити порядок повідомлень, що поступають в процесор контрольних вершин, по суті, довільним. Таким чином, відповідно до винаходу, повідомлення від вершин змінних до контрольних вершин можуть перебувати в порядку вершин змінних в процесор контрольних вершин, при цьому повідомлення обробляються в тому порядку, в якому вони приймаються. Це дозволяє створити декодери кодів з LDPC, такі, як декодери 400 та 500, зображені на Фіг.4 та 5, що втілюються відповідно до винаходу, в яких обидві сторони (вершин змінних і контрольних вершин), які втілюються у вигляді процесорів вершин змінних і контрольних вершин, відповідно оновлюються в порядку вершин змінних. Більше того, і елементи обробки вершин змінних, і елементи обробки контрольних вершин можуть працювати і працюють паралельно, наприклад - одночасно, в деяких варіантах здійснення.

Такі втілення декодера згідно з даним винаходом забезпечують істотну економію на вимогах до пам'яті в порівнянні з втіленнями декодера, в яких обробка вершин змінних здійснюється в порядку повідомлень вершин змінних, обробка контрольних вершин здійснюється в порядку повідомлень контрольних вершин, а пам'ять використовується для зберігання і забезпечення переупорядкування повідомлень, що проходить між процесорами контрольних вершин і вершин змінних.

Описавши деякі з основних принципів і переваг конструкції декодера кодів з LDPC згідно з даним винаходом, перейдемо тепер до опису можливих блоків і декодерів кодів з LDPC, які втілюють один або декілька ознак даного винаходу.

На Фіг.3 зображений блок 300 обробки вершин обмежень, також відомий під назвою «блок обробки контрольних вершин» і втілений відповідно до даного винаходу. Блок 300 приймає повідомлення (V2C) від вершин змінних до контрольних вершин через вхід 302, а інформацію керування - через вхід 324 керуючого сигналу, і генерує повідомлення (C2V) від контрольних вершин до вершин змінних, які виводяться через вихід 322. Блок 300 обробки контрольних вершин зберігає інформацію про повідомлення, яку можна використати для генерування повідомлень від контрольних вершин до вершин змінних, в ущільненому форматі.

Блок 300 обробки контрольних вершин включає в себе блок 310 пам'яті станів контрольних вершин, блок

309 керування, елемент 308 обробки контрольних вершин, пам'ять 312 знаків повідомлень, буферну пам'ять 314 станів контрольних вершин і блок 316 витягування контрольних вершин, які з'єднані одна з одною так, як показано на Фіг.3. У варіанті здійснення, що ілюструється, значення знаку для кожного повідомлення V2C, що приймається, відділено від значення модуля. Значення модуля повідомлення подається в процесор 308 контрольних вершин через вхід 304, а значення знаку, яке відповідає кожному повідомленню, що приймається, яке подається в елемент 308 обробки контрольних вершин, а також зберігаються в пам'яті 312, яка зберігає знаковий біт, який треба використати згодом при генеруванні вихідних повідомлень C2V на основі стану 321, 323, що зберігається.

Для кожної контрольної вершини в структурі коду, яка використовується для керування декодування, пам'ять 310 станів контрольних вершин включає в себе єдиний елемент 321, 323 пам'яті станів, який використовується для зберігання стану для конкретної контрольної вершини, якій відповідає цей елемент пам'яті. Кожна пам'ять 321, 323 станів контрольних вершин зберігає інформацію про стан, яка відповідає одній контрольній вершині, відповідно до структури графа коду з LDPC, що використовується для керування декодування.

У прикладі, показаному на Фіг.3, стан зберігається в стислому представленні, яке відповідає вищеописаному алгоритму мінімальної суми. Елемент даних скидається на початку обробки контрольних вершин відповідно до елемента даних для кожної ітерації обробки, яка відповідає повідомленням ребер, включеним в одне повне представлення графа коду з LDPC, що використовується для керування декодування. Обробка контрольних вершин від однієї ітерації до іншої ітерації графа може відбуватися перед завершенням першої ітерації, наприклад, відразу ж після завершення повної множини вершин змінних, які відповідають одній контрольній вершині. У такому випадку, скидання інформації 321, 323 про стан всіх множин контрольних вершин не буде відбуватися одночасно. Замість цього, стан контрольної вершини, який відповідає деякій контрольній вершині, буде повністю оновлений, а потім відбудеться його скидання після подачі в буферну пам'ять станів контрольних вершин для використання при генеруванні повідомлень C2V.

Кожний елемент 321, 323 даних стану відповідає одній контрольній вершині в структурі графа, що використовується для керування декодування. Кожний елемент 321, 323 даних стану включає в себе S, одне значення біта, яке є результатом операції «виключаюче АБО» на знакових бітах, які відповідають кожному повідомленню V2C, що приймається, направленою в контрольну вершину, якій згаданий елемент даних відповідає, мінімальне значення m_A , яке вказує мінімальний модуль повідомлення, що приймається, який відповідає конкретній контрольній вершині, друге мінімальне значення m_B модуля повідомлення, яке відповідає конкретній контрольній вершині, і індекс I_A , який вказує ребро повідомлення, по якому було прийняте найменше мінімальне значення m_A .

Вхід 324 керуючого сигналу приймає керуючий сигнал, який використовується для керування роботою блоку обробки контрольних вершин залежно від кількості ребер графа, а значить - і повідомлень, які будуть передані між елементами обробки вершин змінних і контрольних вершин. Сигнал 324 керування включає в себе інформацію, яка вказує, яке ребро, а значить - і яке повідомлення V2C, приймається на вході 302 в конкретний момент часу. Цей самий сигнал можна використати для запуску генерування повідомлень C2V у випадках, коли є фіксований або відомий взаємозв'язок між тактуванням повідомлень V2C, що вводяться, і тактуванням бажаних повідомлень C2V, що виводяться, що звичайно і буває. Керуючий сигнал 324 подається в блок 309 керування, пам'ять 312 знаків повідомлень і зчитуючий блок 316 витягування (обробки) контрольних вершин. Блок 309 керування використовує керуючий сигнал, що приймається, щоб визначити, якій контрольній вершині відповідає повідомлення V2C, що приймається. На основі інформації керування, що приймається, блок керування визначає, який набір інформації 321, 323 про стан контрольних вершин повинен зчитуватися з пам'яті 310 станів контрольних вершин з метою оновлення. Таким чином, блок керування визначає на основі інформації про ребро ту контрольну вершину, якій відповідає повідомлення, що приймається. Відразу ж після того як процесор контрольних вершин оновлює одержану внаслідок вибірки інформацію про стан з використанням інформації про повідомлення, що приймається, як буде описано нижче, оновлений стан записується зворотно в елемент 321, 323 даних стану контрольної вершини, з якого була прийнята інформація щодо стану з метою його оновлення. Цей процес буде продовжуватися доти, доки блок 309 керування не видасть в елемент 308 обробки контрольних вершин сигнал, який вказує, що стан контрольної вершини, зв'язаний з конкретною контрольною вершиною, повністю оновлений з використанням останнього повідомлення V2C, направленою в цю контрольну вершину протягом конкретної ітерації обробки. При повністю оновленому стані контрольної вершини, процесор контрольних вершин зробить скидання значень в згаданому стані контрольної вершини, записуючи значення за умовчанням в стан контрольної вершини і видаючи повністю оновлений стан контрольної вершини в буферну пам'ять 314 станів контрольних вершин, що використовується для витягування повідомлень. Буферна пам'ять 314 станів контрольних вершин «дізнається» про те, який елемент даних треба зберігати в оновленій інформації про контрольну вершину, на основі керуючого сигналу, одержаного з входу 324.

Пам'ять 310 станів контрольних вершин включає в себе вхід 327 повідомлень про оновлений стан контрольних вершин, призначений для прийому інформації про стан, яку треба зберегти, і вхід 327 керування для прийому керуючого сигналу, який вказує, до якого набору інформації 321, 323 про стан контрольних вершин треба здійснити доступ, а також чи треба зберігати або зчитувати оновлену інформацію про стан з вказаного набору інформації про стан контрольних вершин. Інформація 321 або 323 про стан контрольних вершин, що зчитується з пам'яті стану контрольних вершин, виводиться через вихід 329 і подається на вхід стану елемента 308 обробки контрольних вершин, де вона використовується на операції оновлення стану.

Операція оновлення стану, здійснювана процесором 308 контрольних вершин на наборі інформації про стан контрольних вершин, вибірка якої здійснювалася з пам'яті залежно від інформації, включеної в повідомлення V2C, що приймається, є такою: модуль m_i повідомлення C2V, що приймається, порівнюється з мінімальним модулем під в одержаній внаслідок вибірки інформації про стан, яка відповідає контрольній вершині, до якої направлено повідомлення, що приймається.

Якщо m_r менше m_A , то m_B задають рівним m_A , щоб створити оновлене значення m_A , а m_A задають рівним m_r , внаслідок чого m_r стає оновленим мінімумом, а попередній мінімум стає поточним другим мінімумом для контрольної вершини. Крім того, індекс I вказує ребро повідомлення, якому відповідає мінімум m_A , щоб указати те ребро повідомлення, по якому було прийняте повідомлення, яке підлягає обробці, наприклад, повідомлення, що забезпечує оновлене мінімальне значення модуля.

Якщо m_r менше m_B , але більше m_A , то m_B задають рівним m_r , не змінюючи m_A або I .

Якщо m_r дорівнює або більше m_B , то зчитані з пам'яті значення m_A , m_B і I не змінюються, а оновлений стан буде включати в себе значення цих елементів, які зчитані з пам'яті.

Незалежно від відносної величини значення повідомлення, що приймається, і мінімумів, що зберігаються, накопичений знаковий біт S для контрольної вершини буде оновлюватися з кожним повідомленням, яке одержується внаслідок вибірки, за рахунок здійснення операції «виключаюче АБО» над знаковим бітом в повідомленні, що приймається, причому цей знаковий біт зчитується із збереженого елемента даних стану контрольної вершини. Результат операції «виключаюче АБО» стає знаковим бітом S оновленого елемента даних для контрольної вершини, який записується зворотно в пам'ять у випадку контрольної вершини, яка не була повністю оновлена, або подається в буферну пам'ять 314 станів контрольних вершин у випадку повністю оновленого елемента даних контрольної вершини. Як згадувалося вище, у випадку повністю оновленого елемента даних контрольної вершини, в елемент 321, 323 даних контрольної вершини, з якого був зчитаний стан контрольної вершини, будуть записані значення за умовчанням, що призведе до скидання інформації для іншої ітерації обробки графа. Значення за умовчанням для мінімуму m_A і другого мінімуму m_B звичайно будуть максимальними значеннями, які можуть бути привласнені цим елементам.

Задаючи вищеописаний метод обробки контрольних вершин та оновлення станів, потрібно усвідомлювати, що елемент 308 обробки контрольних вершин може обробляти повідомлення від вершин змінних до контрольних вершин в довільному порядку, наприклад, в порядку ребер вершин змінних, а не контрольних вершин. Це дозволяє елементу 308 обробки контрольних вершин обробляти повідомлення V2C в тому порядку, в якому вони природно генеруються внаслідок роботи процесорів вершин змінних, з урахуванням того, що процесори вершин змінних звичайно працюють в порядку ребер вершин змінних.

Оскільки повністю оновлена інформація про стан для контрольної вершини присутня у вигляді представлення інформації, необхідної для генерування повідомлень C2V, що виводяться, в ущільненому форматі, для побудови реальних повідомлень C2V, що видаються через вихід 322, використовується додаткова обробка. Зчитуючий блок 316 обробки, наприклад, витягування контрольних вершин генерує повні повідомлення C2V, виходячи із стану, що зберігається в пам'яті 314 станів контрольних вершин і значень знаків, що зберігаються, які зберігаються в пам'яті 312, причому одне значення 313, 315 знаку зберігається для кожного повідомлення V2C, що приймається.

Таким чином, в порівнянні з втіленнями, де повідомлення не зберігаються в ущільненому форматі, для «зчитування» повідомлення C2V потрібна додаткова обробка, щоб здійснити розущільнення, яке не було потрібне би, якби не використовувався ущільнений формат зберігання. Значення 313, 315, що зберігаються, знаків вхідних повідомлень зчитуються безпосередньо з пам'яті знаків. Кожне значення знаку, що зчитується з пам'яті 312, зазнає операції «виключаюче АБО» з повністю оновленою сумою S , відповідною контрольній вершині, для одержання знаку вихідного повідомлення, що відповідає ребру повідомлення, зв'язаному з вихідним повідомленням C2V. Таким чином, знак вихідного повідомлення для ребра повідомлення генерується, виходячи із знаку повідомлення, прийнятого по ребру повідомлення, а значення S знаку, що накопичується, яке відповідає операції «виключаюче АБО» для повного набору повідомлень, які відповідають контрольній вершині, з якою з'єднане згадане ребро повідомлення в графі коду з LDPC. Надійність, наприклад, модульна частина повідомлення, витягується з мінімального значення, другого мінімального значення і значення індексу повідомлення, включених в стан, що зберігається, для контрольної вершини. Якщо індекс ребра, що зчитується, відповідає A , то видається надійність m_A , в іншому випадку видається надійність m_B . Існує багато можливостей представлення індексу A , який служить як ідентифікатор ребра повідомлення. Хоча проста кількість ребер є одним можливим шляхом представлення індексу A , можна також використати і інші схеми нумерації i /або способи індексації ребер.

Модуль повідомлення, що генерується блоком 316 витягування повідомлень, видається на виході 318 модуля, а знак вихідного повідомлення видається на виході 320, які об'єднані для формування повного повідомлення, що видається через вихід 322.

Описавши блок 300 обробки контрольних вершин, що пропонується, згідно з даним винаходом в зв'язку з Фіг.3, перейдемо тепер до опису, що наводиться з посиланнями на Фіг.4, використання цього блоку в можливому декодері кодів з LDPC, що втілюється відповідно до винаходу.

На Фіг.4 показаний можливий декодер 400 кодів з LDPC, втілений з використанням блоку 300 обробки контрольних вершин, показаного на Фіг.3.

Декодер 400 здійснює і обробку вершин змінних, і обробку контрольних вершин в порядку ребер вершин змінних. Декодер 400 включає в себе - крім блоку 300 обробки контрольних вершин - блок 402 керування декодером, елемент 404 обробки вершин змінних, буфер 406 введення м'яких рішень, буфер 412 виведення м'яких і жорстких рішень, блок 408 контролю обмежень і логічну схему 410 керування ітераціями.

Значення, що вводяться, які треба декодувати, подаються в буфер 406 введення м'яких рішень перед завантаженням в елемент 404 обробки вершин змінних. Блок 402 керування декодером реагує на генерування керуючих сигналів декодера відповідно до інформації про керування декодуванням, що зберігається, яка відповідає структурі графа коду з LDPC, що використовується на операціях керування декодуванням. Блок керування декодером генерує керуючі сигнали, що використовуються для керування роботою блоку обробки контрольних вершин, розглянутою вище в зв'язку з Фіг.3, а також роботою елемента обробки вершин змінних. Під керуванням блоку керування декодером, елемент обробки вершин змінних послідовно завантажується частинами даних, що вводяться, які підлягають обробці, наприклад, значенням повідомлення, що приймається, яке включає в себе інформацію про модуль та знаковий біт. Під час початкової ітерації

повідомлення обмежень, які повинні бути оброблені елементом 404 обробки вершин змінних, не існують, а повідомлення V2C генеруються шляхом обробки даних, що вводяться. Повідомлення V2C генеруються і видаються, виходячи з даних, що вводяться, в порядку ребер повідомлень вершин змінних. Повідомлення V2C, що генеруються, які включають в себе значення модуля і знаку, подаються на вхід 302 повідомлень V2C блоку 300 обробки контрольних вершин. Ці повідомлення також подаються в блок 408 контролю обмежень, а буфер виведення м'яких і жорстких рішень з поперемінним перемиканням приймає знаковий біт, зв'язаний з кожним повідомленням V2C, що генерується. Блок 408 контролю обмежень здійснює контроль, визначаючи, чи задовольняють значення знаків повідомлень, що приймаються, які відповідають одній ітерації графа декодера, заздалегідь визначеному обмеженню декодування, наприклад, здійснює контроль за парністю. Якщо це відбувається в кінці ітерації передачі повідомлення, протягом якої був генерований один повний набір повідомлень V2C та C2V, блок 408 контролю обмежень визначає, чи задовільний контроль за парністю. Блок 408 генерує сигнал завершення декодування, якщо виявляє, що поточна ітерація декодування передачі повідомлення призвела до успішного декодування, або - якщо декодування не визначене як успішне - генерує сигнал завершення ітерації. Сигнал, що генерується блоком 408 контролю обмежень, подається в логічну схему 410 керування ітераціями. У кінці кожної ітерації, збережені значення знаків, відповідні ребрам графа, можуть бути видані з буфера 412 виведення як м'які рішення в припущенні, що контроль обмежень привів до незадовільного результату, або як жорсткі рішення - у випадку, якщо контроль обмежень призвів до задовільного результату.

Логічна схема 410 керування ітераціями просигналізує про безуспішне декодування, видаючи сигнал тайм-ауту, після проведення заздалегідь вибраної кількості ітерацій передачі повідомлення, які не призвели до задовільного сигналу контролю за парністю, або просигналізує про успішне декодування, якщо контроль за парністю дав задовільний результат, перед тайм-аутом. Таким чином, логічна схема 410 керування передбачає функцію тайм-ауту, яка призведе до припинення декодування і у випадку, якщо декодування не завершилося успішно в межах заздалегідь вибраної кількості ітерацій декодування.

Декодер 400 кодів з LDPC здійснює операції обробки і вершин змінних, і контрольних вершин, роблячи це в порядку ребер вершин змінних і тим самим зменшуючи потребу в зберіганні повідомлень між елементами 308, 404 обробки вершин змінних і контрольних вершин в порівнянні з іншими системами, в яких обробка контрольних вершин здійснюється в порядку ребер контрольних вершин, а обробка вершин змінних здійснюється в порядку ребер вершин змінних. Крім того, як буде розглянуто нижче, шляхом ретельного вибору структури графа коду з LDPC можна організувати перекриття при обробці кожної ітерації структури графа коду з LDPC без необхідності забезпечення повного набору дублюючих елементів 321, 323 даних пам'яті станів вершин обмежень.

Ітерація декодування з використанням декодера 400 може відбуватися таким чином. Здійснюється оновлення вершин змінних по одній за раз. Повідомлення зчитуються з пам'яті станів контрольних вершин, що виводяться, і пам'яті знаків, і формуються вихідні повідомлення контрольних вершин. Ці повідомлення підсумовуються у вершинах змінних, а потім віднімаються з суми, дотримуючись стандартної обробки вершин змінних. По мірі формування вихідних повідомлень, вони посилаються безпосередньо в процесор контрольних вершин, який також здійснює вибірку відповідного конкретного стану з пам'яті. Цей стан оновлюється і повертається в пам'ять часткових станів. Якщо повідомлення V2C є останнім для обмеження в поточній ітерації, то цей стан можна записати в пам'ять станів, що виводяться і скинути пам'ять часткових станів.

Блок 300 обробки контрольних вершин згідно з даним винаходом і звичайну систему декодера кодів з LDPC, показану на Фіг.4, можна легко адаптувати відповідно до даного винаходу для підтримки використання численних елементів обробки вершин обмежень і вершин змінних, які скомпоновані і працюють паралельно. На Фіг.5 показано, що декодер 500 працює з використанням N вершин обмежень, схем витягування станів та елементів обробки вершин змінних, працюючих паралельно. Показані на Фіг.5 блок 308' контролю обмежень, логічна схема 310' керування ітераціями і буфери 312' та 306' працюють так само або аналогічно тому, як працюють елементи зі схожими позиціями, описані в зв'язку з Фіг.4. Інші елементи в блоці 300' обробки контрольних вершин також працюють так само або аналогічно тому, як працюють елементи зі схожими позиціями, що не включають в себе символ «>», описані в зв'язку з Фіг.4. Однак у варіанті здійснення, показаному на Фіг.5, елемент 308' включає в себе N процесорів 308 обмежень, розташованих паралельно, а пам'ять 324 станів вершин обмежень також призначена для роботи із збільшеною в N разів кількістю наборів станів контрольних вершин, тоді як блок 316' витягування станів вершин обмежень включає в себе N схем витягування повідомлень. Зазначимо, що пам'ять 310', підтримуючи доступ до N наборів інформації про стан одночасно, не обов'язково повинна включати в себе всі додаткові елементи даних станів і порівнянні з варіантом здійснення згідно з Фіг.4, оскільки кількість елементів даних стану визначається кількістю контрольних вершин в графі, а не тим, скільки вершин втілено паралельно.

Варіант здійснення згідно з Фіг.5 включає в себе блок 502 керування декодером, призначений для підтримки N операцій широкого декодування на основі використання опису малих графів та інформації керування, яка вказує, як змінити малий граф, щоб генерувати більший граф, призначений для керування декодування. Зміни, що вносяться в малий граф, можуть бути втілені у вигляді переупорядкувань, наприклад - циклічних зсувів, повідомлень, що передаються між елементами обробки вершин обмежень і контрольних вершин, при цьому передача наборів з N повідомлень проходить паралельно. Зокрема, у варіанті здійснення згідно з Фіг.5, перемикачі 530, 526, 528 використовуються для керування переупорядкуванням повідомлень, коли ті передаються між різними елементами декодера 500. Переупорядкування залежить від інформації опису графа, що зберігається в керуючій логічній схемі 518, а сигнали циклічного зсуву, карти перестановок, що генеруються пам'яттю 522, залежать від лічильника 520 стовпців, що запускається керуючою логічною схемою 518. Керуючі сигнали циклічного зсуву, що використовуються для керування переупорядкуванням повідомлень, що подаються в блок 300' обробки контрольних вершин, генеруються шляхом затримки за допомогою лінії 524 затримки деяких сигналів циклічного зсуву, повідомлень, що використовуються для керування змінним переупорядкуванням, що подається на вхід елементів 504 обробки вершин змінних.

Описавши блок 300 обробки контрольних вершин і різні декодери 400 та 500 кодів з LDPC, що втілюються відповідно до даного винаходу, перейдемо тепер до розгляду різних ознак, що стосуються схеми графа, структури коду LDPC і втілень декодування з використанням конкретних структур графів в декодерах, що втілюють даний винахід.

При втіленні декодера так, як описано на Фіг.5, для високошвидкісних прикладень, малий граф, який описується інформацією опису графа, що зберігається, і іноді називається графом, що проектується, звичайно буде досить малим, тобто він буде мати малу кількість вершин змінних. Як показано на кресленні, зображена архітектура здійснює одне оновлення ребра, що проектується, за цикл синхронізації, що відповідає обробці N повідомлень під час кожного циклу синхронізації. Якщо декодування завершеної ітерації графа повинно закінчуватися перед тим, як можна буде почати наступну, в декодуючу систему будуть вноситися затримки. Конвеєрна затримка являє собою додаткові цикли синхронізації, необхідні для завершення операцій декодування на кожному з ребер після того, як кожне ребро ініціювало обробку. Якщо конвеєрна затримка велика в порівнянні з розміром графа, що проектується, то ця додаткова затримка зможе достатньо сповільнити декодер.

Разом з тим, завдяки розумній схемі графа можна пом'якшити проблему затримки незалежно від того, де - в декодері 400 або паралельному декодері 500 - вона виникає.

Припустимо, що множину вершин V змінних, що проектується, можна розділити на декілька множин, наприклад щонайменше на множини $SV1$ та $SV2$, де $SV1$ передус $SV2$, а множину контрольних вершин C , що проектується, можна розділити щонайменше на дві множини $SC1$ та $SC2$ таким чином, що контрольні вершини в $SC1$ з'єднані ребрами в графі тільки з вершинами змінних в $SV1$. Потім, якщо обробка здійснюється в порядку вершин змінних, то першими будуть оброблятися повідомлення, які відповідають вершині $V1$ змінної, потім - повідомлення, які відповідають вершині $V2$ змінної, потім - повідомлення, які відповідають вершині $V3$ змінної, і так далі. У такому випадку, обробка повідомлень, які відповідають першій множині $SC1$ контрольних вершин, буде завершена до моменту, коли буде оброблена остання вершина змінної в $SV1$, або до цього моменту, якщо остання контрольна вершина в $SC1$ не з'єднана з останнім ребром останньої вершини змінної в $SV2$. Стан повного завершення, що одержується, який відповідає контрольним вершинам у множині $SC1$, можна зберігати в пам'яті станів контрольних вершин для використання при генеруванні повідомлень $V2C$, коли починається обробка повідомлень $V2C$ з вершини $V1$ змінної для наступної ітерації графа. Таким чином, наступну ітерацію обробки графа можна починати, коли блок 300 обробки вершин обмежень ще обробляє повідомлення $V2C$, які відповідають незавершених ітерацій обробки графа.

Це виявляється можливим тому, що коли обробка повідомлень з першої множини $SV1$ вершин змінних починається на ітерації $n+1$, обмеження в $C1$ вже повністю оновлені на ітерації n . Обмеження графа наявністю розділення цього типу забезпечує перекриття послідовних ітерацій у часі, внаслідок чого виникає можливість ефективного зменшення і/або виключення витрат, зумовлених конвеєрною затримкою.

На Фіг.6 показані код з LDPC і відповідна структура 600 графа з такими властивостями. Показаний на Фіг.6 граф включає в себе 8 контрольних вершин 602, позначених символами C_1 - C_8 , і 16 вершин змінних, позначених символами V_1 - V_{16} . Зазначимо, що до моменту, коли повідомлення з вершин V_1 - V_{12} змінних, включених у множину $SV1$ вершин змінних, обробляються блоком 300 обробки вершин обмежень, стан, який відповідає першим 4-м вершинам C_1 - C_4 обмежень, буде повністю оновлений. Цей повністю оновлений стан буде переданий в буферну пам'ять 314 вершин обмежень з видаленням елементів даних вершин обмежень, які відповідають вершинам C_1 - C_4 обмежень, які треба використати на наступній ітерації обробки графа. Таким чином, оскільки обробка застосовно до вершин обмежень у множині $SC2$, тобто вершин C_5 , C_6 , C_7 та C_8 обмежень, продовжується протягом однієї ітерації, повідомлення $C2V$ для цієї ітерації, які відповідають вершинам у множині $SC1$, будуть генеруватися, і в той самий час виникне можливість почати обробку вершин обмежень для наступної ітерації, що перекривається з виникаючою обробкою, зв'язаною з поточною ітерацією.

Способи та пристрої, що передбачають застосування декодера кодів з LDPC, згідно з даним винаходом придатні, зокрема, для одержання кодів для високої швидкості передачі даних, коли кількість контрольних вершин часто істотно менше, ніж кількість ребер в графі. У такому випадку, виявиться тенденція до того, що пам'ять станів, яка використовується відповідно до даного винаходу, стане істотно меншою, ніж пам'ять ребер, яка була потрібна би в іншому випадку, якби з метою обробки вершин обмежень забезпечувалося зберігання повних повідомлень.

Фіг.7 ілюструє відносно простий граф, який буде використовуватися для пояснення обробки за допомогою декодера, наприклад, обробки, яка відбувається, коли декодер 400 згідно з Фіг.4 використовується для декодування можливого набору даних, що вводяться, які мають значення 0, -1, 5 та 4 повідомлень, які завантажуються у вершини змінних під час першої ітерації обробки графа.

Показаний на Фіг.7 граф 700 включає в себе в загальній складності три контрольні вершини 702 і чотири вершини 706 змінних, з'єднані одна з одною ребрами, кількість яких становить 9, в порядку вершин змінних від 0 до 8. Зазначимо, що якби ребра були пронумеровані з боку контрольних вершин, то ребра мали б іншу послідовність нумерації, тому що вони з'являються в контрольних вершинах в іншому порядку, ніж у вершинах змінних. Повідомлення передаються назад і уперед між вершинами 706 змінних і контрольними вершинами 702 по зображених ребрах. Контрольні вершини 702 включають в себе з першої по третю контрольні вершини 710, 712, 714, а вершини 706 змінних включають в себе з першої по четверту вершини 720, 722, 724, 726 змінних.

На Фіг.8 показані різні результуючі значення використання декодера 400 і структури коду, показаної на Фіг.7. Фіг.8 ілюструє результати двох повних ітерацій обробки за допомогою декодера, які йдуть за початковою ітерацією, що використовується для завантаження повідомлень (0, -1, 5 та 4), що приймаються, в декодер 400 і початкової обробки цих повідомлень. Знак «мінус» використовується перед другим значенням, що вводиться, щоб указати негативний знаковий біт, еквівалентний біту «1» і зв'язаний з повідомленням, тоді як відсутність знаку «мінус» вказує позитивний знаковий біт, еквівалентний біту «0». Таким чином, друге повідомлення -1 має негативний знаковий біт, тоді як перше, третє і четверте повідомлення мають позитивний знаковий біт. У

кінці першої ітерації обробки, елементи даних пам'яті станів, відповідні C1, C2 та C3, будуть включати в себе значення, показані вверху Фіг.8, в наборі значень 802 елементів даних пам'яті станів.

Ітерація починається витягуванням повідомлень V2C. На початковій ітерації повідомлення C2V, що виділяються, задаються рівними 0. Як частини поточної ітерації, ці повідомлення підсумовуються у вершинах змінних, і генеруються повідомлення V2C. Ці повідомлення приймаються процесором контрольних вершин, який оновлює стан контрольних вершин, зв'язаний з поточною ітерацією. Ітерація завершується, коли оновлені всі стани контрольних вершин.

На тій частині креслення, яка знаходиться під набором значень 802 елементів даних пам'яті станів, кожний рядок відповідає обробці, зв'язаній з відмінним ребром повідомлення, наприклад, витягуванню повідомлення C2V, генеруванню повідомлення V2C, яке повинне бути передане вздовж ребра, що вказується, і оновленню стану контрольної вершини, зв'язаної із заданим ребром. Кожна ітерація передбачає генерування та обробку за одним повідомленням C2V на ребро графа згідно з Фіг.7, а також генерування та обробку одного повідомлення V2C для кожного ребра.

Взагалі кажучи, стовпці 810, 812, 816 та 820 ілюструють витягування повідомлень C2V з використанням результатів, які виявилися на попередній ітерації декодування. Стовпець 822 показує суми вершин змінних, які є сумами всіх повідомлень, що входять у вершину змінної, і значення, що приймається, зв'язаного з вершиною змінною.

Стовпець 824 показує повідомлення V2C, яке будуть генеруватися по відповідному ребру в ітерації. Воно дорівнює сумі вершин змінних за вирахуванням вхідного повідомлення C2V. Ці повідомлення C2V будуть прийматися процесором вершин обмежень і використовуватися для оновлення інформації про стан, як показано в стовпцях 826, 828, 830. Стовпці 826, 828, 830 показують, як стан контрольної вершини, що обробляється, оновлюється під час поточної ітерації обробки у відповідь на прийом кожного повідомлення V2C, показаного в стовпці 824.

Інформація 810 показує стан контрольних вершин, генерований під час попередньої ітерації обробки графа, зчитаної з буферної пам'яті 314 станів вершин обмежень, для генерування повідомлень C2V. Цей повністю оновлений стан, що відповідає кожній контрольній вершині, який генерується під час попередньої ітерації обробки, використовується для генерування повідомлень C2V, які обробляються процесором 304 вершин змінних на поточній ітерації графа для генерування повідомлень V2C. Стовпець 812 ілюструє знаковий біт, який відповідає повідомленню V2C з попередньої ітерації, який йде по ребру і вибірка якого здійснюється з пам'яті 312 знакових бітів для побудови повідомлення C2V в поточній ітерації обробки, яке буде введене в процесор вершин змінних, що генерує повідомлення V2C для поточної ітерації обробки. Стовпець 814 ілюструє оновлений знаковий біт, що генерується станом вершини обмеження, що витягується за допомогою операції «виключаюче АБО» над знаковим бітом 812, що зберігається, з попереднього повідомлення V2C, що приймається, і накопиченого знакового битого S, що зберігається, який генерується на попередній ітерації і одержується із стану, який відповідає контрольній вершині (C1, C2 або C3), якій відповідає ребро повідомлення, зв'язане з повідомленням C2V. Стовпець 816 вказує результат контролю, що проводиться блоком витягування станів вершин обмежень для того, щоб визначити, чи потрібно було видавати перший мінімум m_A або другий мінімум p під час попередньої ітерації обробки. Згаданий контроль передбачає порівняння індексу ребра повідомлення C2V, що генерується з індексом I ребра, який відповідає першому мінімуму, що зберігається. Тут «Y» вказує результат «Так», який означає, що індекс ребра відповідає ребру, яке забезпечувало передачу мінімального модуля в попередній ітерації, а «N» вказує результат «Ні», який означає, що індекс ребра не відповідає ребру, яке забезпечувало передачу мінімального модуля в попередній ітерації. Стовпець 812 ілюструє результат вибору значення m_A або m_B , яке повинне бути використане як модуль вихідного повідомлення C2V. Стовпець 820 показує реальне повідомлення C2V, яке буде передане по вказаному ребру для використання процесором 304 вершин змінних під час поточної ітерації. Процесор 304 вершин змінних генерує суму повідомлень C2V, що приймаються під час ітерації обробки. Результуюча сума вершин змінних, показана в стовпці 822, потім використовується для генерування повідомлення C2V, що подається в блок 300 обробки вершин обмежень по ребру повідомлення, наприклад, для обробки вершини обмеження під час поточної ітерації.

За зображенням блоку обробки вершин обмежень можна сказати, що інформація, показана в стовпцях 810, 812, 814, 816, 818, 820 та 822, відображає операції, які відбуваються на основі результату, обчисленого на попередній операції обробки графа. Як сказано вище, за умови ретельного опрацювання схеми графа, в ньому може бути забезпечене деяке перекриття, коли ці операції проходять протягом однієї ітерації обробки графа і коли відбувається оновлення стану повідомлення для наступної ітерації обробки вершин обмежень.

Стовпець 824 ілюструє повідомлення V2C, яке буде прийняте та оброблене під час другої і третьої ітерацій, що проводяться блоком обробки вершин обмежень і відповідають обробці можливих даних, що вводяться, з використанням структури графа, показаної на Фіг.7. Кожна ітерація графа передбачає обробку дев'яти повідомлень V2C, які відповідають повідомленням, що передаються по ребрах E0-E8. Обробка повідомлень блоком 300 обробки вершин обмежень відбувається в порядку ребер вершин змінних. На початку цієї обробки, відбудеться повторна ініціалізація пам'яті обмежень, яка відповідає контрольним вершинам. У припущенні відсутності перекриття на ітераціях обробки, стан кожної з контрольних вершин C1, C2 та C3 буде повторно ініціалізований для прийому першого повідомлення V2C, яке відповідає ребру E0. Кожний з стовпців 826, 828, 830 показує стан, що зберігається, який відповідає одній з контрольних вершин C1, C2, C3, відповідно, після його оновлення повідомленням V2C, що приймається, при цьому кожний рядок відповідає обробці відмінного повідомлення V2C. Вміст стану, оновлений у відповідь на деяке конкретне повідомлення, наприклад, повідомлення, що передається по ребру, якому відповідає рядок, показане напівжирним шрифтом. Символи X використовуються в стовпцях 826, 828, 830, щоб показати значення, які задані рівними початковому значенню. Ці значення є «байдужими» значеннями, оскільки вони не будуть використовуватися в повідомленні C2V, що виводиться, яке генерується.

Зазначимо, що в кінці другої ітерації повністю оновлені стани C1, C2, C3 повідомлення будуть оновлені з

досягненням значень ($m_A=0$, $m_B=4$, $l=2$, $S=0$) для C1, ($m_A=1$, $m_B=3$, $l=1$, $S=1$) для C2, і ($m_A=1$, $m_B=3$, $l=3$, $S=1$) для C3. У кінці третьої ітерації повністю оновлені стани C1, C2, C3 повідомлення будуть оновлені з досягненням значень ($m_A=1$, $m_B=3$, $l=2$, $S=0$) для C1, ($m_A=0$, $m_B=3$, $l=1$, $S=0$) для C2, і ($m_A=3$, $m_B=3$, $l=3$, $S=0$) для C3.

Хоча стан не показаний як такий, що стирається після завершення обробки повідомлень для контрольних вершин, у варіантах здійснення, де між ітераціями обробки графа є перекриття, скидання стану, який відповідає повністю оновленій вершині обмеження, повинно бути збережене в буферній пам'яті 324 станів вершин обмеження, а скидання інформації про стан - в пам'яті станів вершин обмежень. У прикладі, показаному на Фіг.8, таке скидання не показано просто для того, щоб полегшити розуміння послідовності оновлення станів вершин обмеження, а також розуміння того, як генерується повний набір інформації про оновлений стан під час обробки, зв'язаної з повною ітерацією графа.

При завданні різних структур кодів, з якими можна використати декодер і способи згідно з даним винаходом, виявляється можливою значна гнучкість застосовно до величини перекриття, яке може виникати між ітераціями обробки. Щоб зменшити затримки при декодуванні в зв'язку з перекриттями застосовно до проведення обробки контрольних вершин відповідно до різних ітерацій декодування, структуру коду можна вибрати такою, що забезпечує перекриття в ітераціях обробки графа, що становить 10%, 20%, 30%, 40% або більше.

Тепер будуть описані різні втілення можливого способу декодування та їх можливі варіанти.

Один можливий спосіб проведення операцій декодування кодів з малою щільністю і контролем за парністю включає в себе етапи, на яких: зберігають в пам'яті інформацію про стан повідомлень, яка відповідає повідомленням, що приймаються множиною контрольних вершин; приймають повідомлення, що вводиться, для контрольної вершини, направлене в одну із згаданої множини контрольних вершин; на основі контрольної вершини, в яку направлене згадане повідомлення, що приймається, вибирають один із згаданої множини наборів інформації про стан для використання на операції оновлення стану; здійснюють вибірку із згаданої пам'яті вибраного набору інформації про стан повідомлень; і оновлюють вибраний набір інформації про стан повідомлень залежно від згаданого повідомлення, що приймається. Спосіб додатково включає в себе етап, на якому записують згаданий оновлений набір інформації про стан повідомлень в комірку пам'яті, з якої була зроблена вибірка згаданої інформації про стан повідомлень, і послідовно повторюють згадані етапи прийому, вибору, вибірки та оновлення для кожного з множини повідомлень, що приймаються в першій послідовності, яка відповідає порядку, в якому ребра, які відповідають згаданим повідомленням, що приймаються, з'єднані з вершинами змінних, що обробляються, в графі, який відображає код з LDPC. У деяких втіленнях, послідовність, яка відповідає порядку, в якому ребра, які відповідають згаданим повідомленням, що приймаються, з'єднані з вершинами змінних, що обробляються, в графі, який відображає код з LDPC, відрізняється від другої послідовності, в якій ребра, які відповідають згаданим повідомленням, що приймаються, з'єднані з вершинами змінних, що обробляються, в графі, який відображає код з LDPC. Можливий спосіб додатково включає в себе етап, на якому здійснюють операцію витягування повідомлення від контрольної вершини до вершини змінної щонайменше на одному наборі інформації про стан, який відповідає контрольній вершині, для якої прийнятий повний набір повідомлень від вершин змінних до контрольних вершин, причому операцію витягування здійснюють багато разів для генерування множини повідомлень від контрольних вершин до вершин змінних, при цьому повідомлення від контрольних вершин до вершин змінних, що направляються в окрему одну із згаданої множини вершин змінних, генерують послідовно для одержання послідовності повідомлень від контрольних вершин до вершин змінних, яка направляється в згадану окрему одну із згаданої множини вершин змінних.

У можливому способі, повністю оновлений стан генерують для кожної контрольної вершини перед збереженням повністю оновленого стану в буферній пам'яті для використання в процесі витягування повідомлень C2V. Повністю оновлений стан - це стан, який оновлений повним набором повідомлень V2C, який відповідає контрольній вершині, якій відповідає згаданий стан. Під час обробки, яка відповідає повному графу, будуть оновлюватися численні набори даних стану. Численні ітерації декодування, які відповідають графу, звичайно здійснюються з кожною ітерацією, яка відповідає обробці одного повного набору повідомлень, які використовуються для відображення коду з LDPC, що використовується для керування декодування. Стан контрольної вершини можна оновлювати відповідно до різних ітерацій обробки графа протягом одного і того самого періоду часу при допущенні, що стан контрольної вершини для подальшої ітерації вже був повністю оновлений протягом попередньої ітерації.

Таким чином, в деяких втіленнях спосіб включає в себе оновлення ще одного набору стану, який відповідає згаданій першій контрольній вершині, як частині другої ітерації обробки повідомлень від вершин змінних до контрольних вершин перед тим, як повністю завершити згадане оновлення стану згаданої щонайменше однієї контрольної вершини в графі, під час згаданої першої ітерації обробки декодування.

Як частину способу декодування, полегшуючу перекриття ітерацій обробки графа, в деяких варіантах здійснення спосіб додатково включає в себе буферизацію згаданого повністю оновленого стану, який відповідає згаданій першій контрольній вершині, і витягування повідомлень від контрольних вершин до вершин змінних із згаданого буферизованого повністю оновленого стану. Етап витягування повідомлень від контрольних вершин до вершин змінних із згаданого буферизованого повністю оновленого стану включає в себе в деяких втіленнях генерування множини вихідних повідомлень із згаданого повністю оновленого стану та інформації про знак, що зберігається, відповідного множині повідомлень від контрольних вершин до вершин змінних, що використовуються для генерування згаданого повністю оновленого стану.

Деякі можливі способи декодування включають в себе повне завершення оновлення станів для першої множини контрольних вершин перед завершенням оновлень станів, які відповідають другій множині контрольних вершин, при цьому оновлення стану для контрольної вершини є повністю завершеним, коли стан для контрольної вершини оновлений один раз для кожного з множини ребер повідомлень, які відповідають згаданій контрольній вершині. У деяких втіленнях кожна з першої і другої множин контрольних вершин

включає в себе щонайменше 20% загальної кількості контрольних вершин в графі коду з LDPC, який відображає структуру коду з LDPC, що втілюється, яка використовується для керування декодування.

Операцію оновлення стану можна здійснювати як частину оновлення стану, який відповідає першій множині контрольних вершин. У деяких втіленнях, оновлення вершин до множини контрольних вершин здійснюють протягом першого періоду часу з використанням першого набору повідомлень від вершин змінних до контрольних вершин, при цьому спосіб додатково включає в себе: оновлення інформації про стан, яка відповідає другій множині контрольних вершин, під час другого періоду часу, причому згадана друга множина контрольних вершин включає в себе тільки контрольні вершини, які не включені в згадану першу множину контрольних вершин, при цьому згаданий другий період часу йде за згаданим першим періодом часу. У таких втіленнях, інформацію про контрольні вершини можна витягувати в різні моменти часу. В одному втіленні, спосіб включає в себе витягування повідомлень від контрольних вершин до вершин змінних з оновленої інформації про стан, яка відповідає згаданій першій множині контрольних вершин, протягом згаданого другого періоду часу. У деяких втіленнях, які реалізують цю ознаку тактування, перший і другий періоди часу однакові за тривалістю. Перший і другий періоди часу можуть бути - і часто бувають - відділені один від одного третім періодом часу, протягом якого відбувається оновлення стану, який відповідає третьому набору контрольних вершин. Це часто відбувається, коли використовують великі графи, що включають в себе багато вершин. У деяких втіленнях кожна з першої і другої множин контрольних вершин включає в себе щонайменше 10% контрольних вершин в графі, який відповідає коду з LDPC, що використовується для керування декодування. В інших втіленнях, кожна з першої і другої множин контрольних вершин включає в себе щонайменше 20% контрольних вершин в графі, який відповідає коду з LDPC, що використовується для керування декодування. У деяких втіленнях перший період часу становить менше 40% того часу, який потрібний для обробки N повідомлень від вершин змінних до контрольних вершин, де N дорівнює сумарній кількості ребер повідомлень в графі, який відповідає коду з LDPC, що використовується для керування декодування, тоді як в інших варіантах здійснення перший період часу становить менше 20% того часу, який потрібний для обробки N повідомлень від вершин змінних до контрольних вершин, де N дорівнює сумарній кількості ребер повідомлень в графі, який відповідає коду з LDPC, що використовується для керування декодування.

Можливі численні варіанти вищеописаних способів та пристроїв, що залишаються в рамках обсягу претензій винаходу. Наприклад, хоча застосовно до повідомлень мова йшла про ущільнений стан, інформацію знаку для повідомлень C2V можна генерувати трохи не так, як описано вище, але це все одно буде в рамках обсягу претензій винаходу. Альтернативне втілення передбачає використання структури з поперемінним перемиканням, при наявності якої одна пам'ять використовується як пам'ять станів, а інша містить повні стани з попередньої ітерації, причому в кінці ітерації буферні пам'яті міняються ролями одна з одною. У такому варіанті здійснення, із знаковим бітом треба поводитися таким чином. Зберігають знаки повідомлень V2C. У доповнення до інформації про надійність, підтримують як частину стану результат операції «виключає АБО» для вхідних знакових бітів. При витягуванні повідомлень C2V, піддають знак, що зберігається, повідомлення V2C операції «виключає АБО» з накопиченим результатом операції «виключає АБО» в цьому стані, одержуючи знаковий біт повідомлення C2V. Зазначимо, що відразу ж після зчитування знаковий біт повідомлення V2C більше не потрібен, так що поверх нього можна згодом записати знаковий біт повідомлення V2C, який незабаром буде генерований процесором вершин змінних.

Вищеописані способи можуть бути здійснені в комп'ютерній системі, яка включає в себе пам'ять, центральний процесор і один або декілька пристроїв введення-виведення, зв'язані один з одним. У такому варіанті здійснення, пам'ять включає в себе підпрограму, втілену відповідно до винаходу. При її виконанні, ця програма примушує центральний процесор здійснювати вибірку, обробку, наприклад - декодування, і виведення відповідно до даного винаходу. В альтернативному варіанті, етапи згідно з даним винаходом можуть бути втілені з використанням спеціалізованих апаратних засобів, наприклад, схем, і/або комбінації апаратних засобів та програмних засобів.

Перелік посилальних позицій

100 граф Таннера

102 множина вершин - вершини «змінних»

104 ребра в графі

106 множина вершин - вершини «обмежень» («контрольні»)

202 матриця контролю за парністю

206 вектор «х»

300, 300' Блок обробки контрольних вершин (вершин обмежень)

302, 322 Повідомлення V2C

304 Надійність m вхідного повідомлення (в порядку двонаправлених каналів вершин змінних)

304' N елементів обробки вершин змінних

306 Знак s вхідного повідомлення

306' Буфер виведення м'яких і жорстких рішень з поперемінним перемиканням 308, 308' Елемент обробки контрольних вершин

309 Блок керування

310 Пам'ять станів контрольних вершин

310' Логічна схема керування ітераціями, що забезпечує динамічний тайм-аут

310' Пам'ять станів вершин обмежень

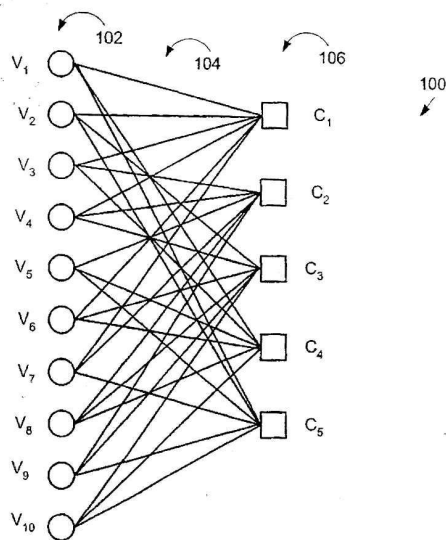
312 Пам'ять знаків повідомлень

312' Буфер виведення м'яких і жорстких рішень з поперемінним перемиканням

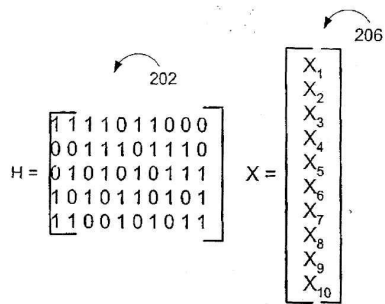
314 Буферна пам'ять станів контрольних вершин

316 Блок обробки (витягування) контрольних вершин, який видає m_d , якщо $A=j$, а в іншому випадку видає m_b , при цьому знак вихідного повідомлення являє собою результат операції що «виключає АБО» над S_j та S

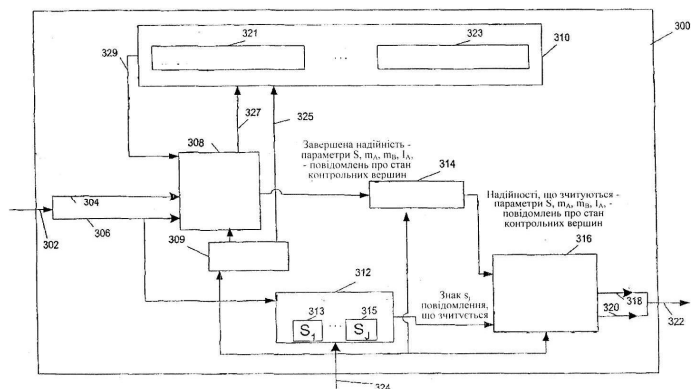
316' Блок витягування контрольних вершин
 318 Надійність m вихідного повідомлення
 321 Стан - параметри S, m_A, m_B, I_A , для контрольної вершини C_1
 323 Стан - параметри S, m_A, m_B, I_A , Для контрольної вершини C_K
 324 Вхід керуючого сигналу
 310', 324' Пам'ять станів вершин обмежень
 325 Керування
 327 Оновлені надійності - параметри S, m_A, m_B, I_A , - повідомлень про стан контрольних вершин
 329 Надійності-параметри S, m_A, m_B, I_A , - повідомлень про стан контрольних вершин
 320 Знак $S \mid S_j$ вихідного повідомлення
 400 Декодер кодів з LDPC
 402 Блок керування декодером
 404 Елемент обробки вершин змінних
 406 Буфер введення м'яких рішень з поперемінним перемиканням
 408 Блок керування обмежень
 410 Логічна схема керування ітераціями, що забезпечує динамічний тайм-аут
 412 Буфер виведення м'яких і жорстких рішень з поперемінним перемиканням
 500 Декодер кодів з LDPC (з N паралельними елементами обробки)
 502 Блок керування декодером
 518 Керуюча логічна схема
 520 Лічильник стовпців
 522 Пам'ять карти перестановок
 524 Лінія затримки
 802 Стан контрольних вершин, який відповідає контрольним вершинам, після початкової ітерації
 810 Стан контрольної вершини, що зчитується для витягування повідомлення C_{2V} , в кінці попередньої операції
 812 Знаковий біт з повідомлення V_{2C}
 814 Оновлений знаковий біт, що виводиться
 816 Чи відповідає індекс (I) ребру повідомлення, що обробляється?
 818 Модуль під або m_B , повідомлення вибраний для виведення
 820 Виведення C_{2V}
 822 Сума вершин змінних
 824 Повідомлення V_{2C} , що приймається
 826 Стан C_1
 828 Стан C_2
 830 Стан C_3



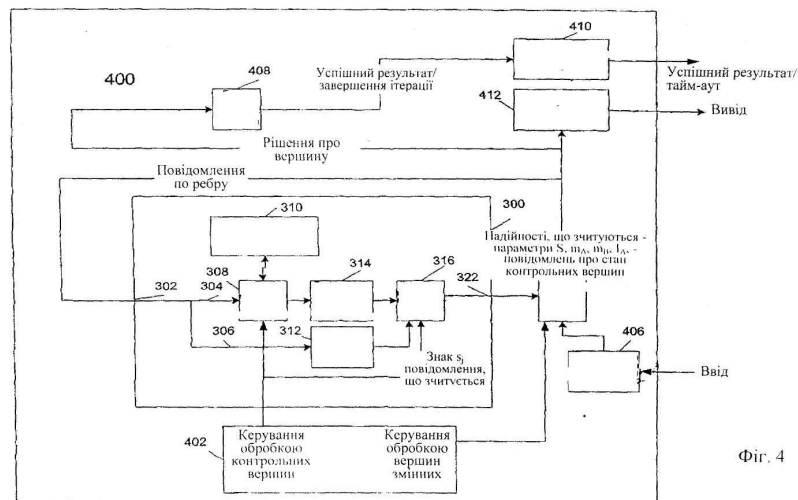
Фіг. 1



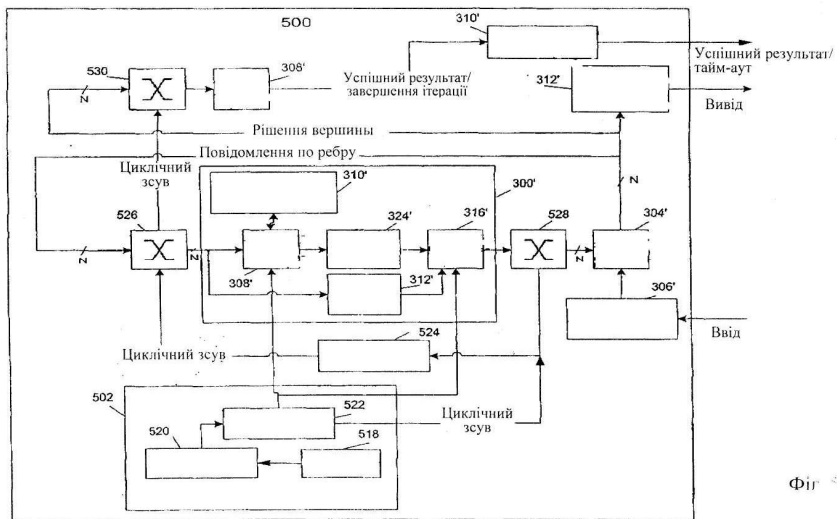
Фіг. 2



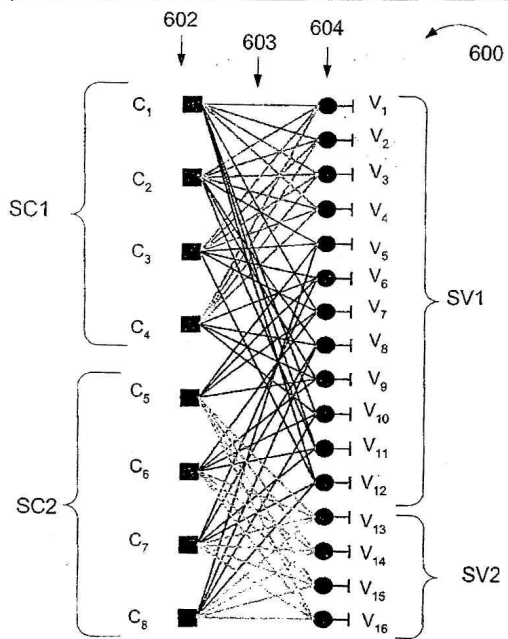
Фіг. 3



Фіг. 4



Фіг. 5



Фіг. 6

