(19) UA (11) 75 863 (13) C2

(51)МПК

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ

ГОСУДАРСТВЕННЫЙ ДЕПАРТАМЕНТ ИНТЕЛЛЕКТУАЛЬНОЙ СОБСТВЕННОСТИ

(12) ОПИСАНИЕ ИЗОБРЕТЕНИЯ К ПАТЕНТУ УКРАИНЫ

(21), (22) Заявка: 2001010727, 04.08.1999

(24) Дата начала действия патента: 15.06.2006

(30) Приоритет: 04.08.1998 US 09/129,022

(46) Дата публикации: 15.06.2006H03M 13/00 20060101CFI20060112BHUA H04L 1/00 20060101CLI20060112BHUA

(86) Заявка PCT:
PCT/US99/17659, 19990804

(72) Изобретатель:
Гансквин Дэйвид, US

(73) Патентовладелец:
КВАЛКОММ ИНКОРПОРЕЙТИД, US

(54) ПОСЛЕДОВАТЕЛЬНЫЙ ДЕКОДЕР ВИТЕРБИ (ВАРИАНТЫ) И СПОСОБ ПОСЛЕДОВАТЕЛЬНОГО ДЕКОДИРОВАНИЯ ПО ВИТЕРБИ

(57) Реферат:

Настоящее изобретение относится к последовательному декодеру Витерби, содержащему сверхоперативное запоминающее устройство с цепочной организацией данных и предназначенному для использования в аппаратах мобильной телефонной связи. В соответствии с одним из вариантов осуществления изобретения, декодер содержит устройство для обнаружения и исправления ошибок, устройство сложения, суммирования и выборки данных и устройство для цепочной обработки данных, содержащее оперативное запоминающее устройство с цепочной организацией данных, сверхоперативное запоминающее устройство с цепочной организацией данных и устройство управления. Сверхоперативное запоминающее устройство с цепочной организацией данных предназначено для запоминания результатов предыдущих циклов обработки данных, благодаря чему исключается необходимость выполнения всех операций, предусмотренных в соответствии с алгоритмом цепочной обработки данных. Сверхоперативное запоминающее устройство обеспечивает возможность считывания всех данных в цикле обработки данных. Наличие сверхоперативного запоминающего устройства позволяет существенно уменьшить мощность, потребляемую аппаратными средствами, и время обработки данных при незначительном усложнении аппаратных средств. В соответствии с другим вариантом, сверхоперативное запоминающее устройство с цепочной организацией данных отсутствует. В этом случае устройство для цепочной обработки данных содержит (L + 1)-разрядное оперативное запоминающее устройство, реверсивный двоичный счетчик и регистр сдвига, предназначенные для эмуляции сверхоперативного запоминающего устройства с цепочной организацией данных. В соответствии с еще одним вариантом, вместо (L + 1)-разрядного оперативного запоминающего устройства и реверсивного счетчика используется L-разрядный регистр сдвига. В декодере, выполненном по любому варианту, устройство для цепочной обработки данных может иметь структуру, обеспечивающую только одно считывание данных или считывание всех данных в каждом цикле обработки данных. В соответствии с еще одним вариантом, устройство для цепочной обработки данных выполняет операции алгоритма цепочной обработки данных по схеме "от *a* до *b*", то есть таким образом, что доступ к сверхоперативному запоминающему устройству для запоминания данных происходит после *a* считываний данных до тех пор, пока не будут записаны данные для *b* считываний. В соответствии с другими вариантами, устройство для цепочной обработки данных выполняет операции алгоритма цепочной обработки данных после считывания данных в нескольких циклах, а не в одном цикле обработки данных.

Официальный бюлетень "Промышленная

UA 75863 C2

UA 75863 C2

собственность". Книга 1 "Изобретения, полезные модели, топографии интегральных микросхем", 2006, N 6, 15.06.2006. Государственный департамент интеллектуальной собственности Министерства образования и науки Украины.

UA 75863 C2

UA 75863 C2

(19) UA (11) 75 863 (13) C2

(51) Int. Cl.

MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE

STATE DEPARTMENT OF INTELLECTUAL PROPERTY

(12) DESCRIPTION OF PATENT OF UKRAINE FOR INVENTION

(21), (22) Application: 2001010727, 04.08.1999

(24) Effective date for property rights: 15.06.2006

(30) Priority: 04.08.1998 US 09/129,022

(46) Publication date: 15.06.2006H03M 13/00 20060101CFI20060112BHUA H04L 1/00 20060101CLI20060112BHUA

(86) PCT application: PCT/US99/17659, 19990804

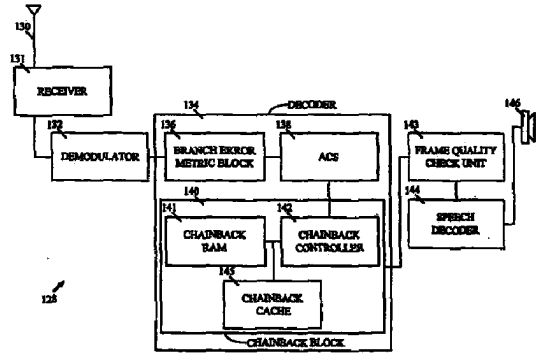
(72) Inventor: Hanskvin David, US

(73) Proprietor: QUALCOMM INCORPORATED, US

(54) SERIAL VITERBI DECODER (VARIANTS) AND A METHOD OF SERIAL VITERBI DECODING

(57) Abstract:

A serial Viterbi decoder having a chainback cache is provided for use in a mobile telephone. In one embodiment described herein, the decoder includes a branch error metric block, an add-compare-select unit, and a chainback block including a chainback RAM, a full chainback cache and chainback controller circuitry. The chainback cache caches decision bits from previous process cycles such that full chainback operations need not always be performed. The chainback cache is configured to cache on all reads. With the chainback cache, significant savings in power consumption and processing time may be achieved with only a relatively modest increase in me amount of circuitry required. In another embodiment, a full chainback cache is not provided. Rather, the chainback block instead includes an L+1 bit RAM, an updown counter and a shift register configured to emulate a chainback cache. In still another embodiment, an L bit shift register is employed instead of the combination of the L+1 bit RAM and updown counter. In the various embodiments, the chainback block may be configured to perform only one chainback read in each process cycle or may be configured to perform *m* chainback reads in each process cycle. In still other embodiments, the chainback block is configured to perform chainback operations based on *a through b* reads where the cache is accessed for each read after *a* reads have been done until *b* reads have been performed or *a* match is obtained. In still further embodiments, the chainback block is configured to perform chainback operations over multiple process cycles rather than only a single process cycle.



Official bulletin "Industrial property". Book 1 "Inventions, utility models, topographies of integrated circuits", 2006, N 6, 15.06.2006. State Department of Intellectual Property of the Ministry of Education and Science of Ukraine.

UA 75863 C2

UA 75863 C2

(19) UA (11) 75 863 (13) C2

(51)МПК

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ ДЕПАРТАМЕНТ
ІНТЕЛЕКТУАЛЬНОЇ ВЛАСНОСТІ

(12) ОПИС ВИНАХОДУ ДО ПАТЕНТУ УКРАЇНИ

(21), (22) Дані стосовно заявки:
2001010727, 04.08.1999

(24) Дата набуття чинності: 15.06.2006

(30) Дані стосовно пріоритету відповідно до Паризької конвенції : 04.08.1998 US 09/129,022

(46) Публікація відомостей про видачу патенту (деклараційного патенту): 15.06.2006H03M 13/00 20060101CFI20060112BHUA H04L 1/00 20060101CLI20060112BHUA

(86) Номер та дата подання міжнародної заявки відповідно до договору PCT:
PCT/US99/17659, 19990804

(72) Винахідник(и):
Гансквін Дейвід , US

(73) Власник(и):
КВАЛКОММ ІНКОРПОРЕЙТИД, US

(54) ПОСЛІДОВНИЙ ДЕКОДЕР ВІТЕРБІ (ВАРІАНТИ) І СПОСІБ ПОСЛІДОВНОГО ДЕКОДУВАННЯ ПО ВІТЕРБІ

(57) Реферат:

Декодер Вітербі з кешем зворотних кроків, призначений для використання у мобільному телефоні. У одному з описаних втілень декодер включає відгалужений блок показників помилок, вузол додання-порівняння-обрання, блок зворотних кроків з пам'яттю з довільним доступом, повний кеш зворотних кроків і контролер зворотних кроків. Кеш зворотних кроків приймає біти рішення з попередніх циклів обробки і тому немає потреби завжди виконувати повні процедури зворотних кроків. Кеш зворотних кроків приймає усі зчитування. Застосування кеш-пам'яті дозволило досягти значних знижень споживання і витрат часу на обробку з лише незначним ускладненням схем. Інше типове втілення не передбачає повного кешу зворотних кроків, але блок зворотних кроків включає пам'ять на L+1 біт з довільним доступом, реверсивний лічильник і зсувний регістр, що емулюють кеш зворотних кроків. У ще одному втіленні використовується зсувний регістр на L бітів замість комбінації пам'яті на L+1 біт з реверсивним лічильником. У різних втіленнях блок зворотних кроків може бути пристосований виконувати лише одне зчитування зворотних кроків у кожному циклі обробки або виконувати m зчитувань зворотних кроків у кожному циклі обробки. У інших втіленнях блок зворотних кроків пристосовують виконувати a і потім b зчитувань протягом кожної процедури зворотних кроків, причому після a зчитувань вміст кешу перевіряється для кожного чергового зчитування, доки протягом кожної процедури зворотних кроків не буде виконано b зчитувань або доки на буде досягнуто збігу. У ще одному втіленні блок зворотних кроків пристосовано виконувати процедури зворотних кроків для багатьох циклів обробки, а не єдиного такого циклу.

UA 75863 C2

UA 75863 C2

Опис винаходу

Винахід взагалі стосується послідовних декодерів Вітербі, зокрема послідовних декодерів Вітербі, призначених для використання у безпровідних системах зв'язку з паралельним доступом і кодовим ущільненням каналів (ПДКУ).

Фіг.1 містить блок-схему системи 10 ПДКУ з змінною швидкістю передачі, описану у внутрішньому стандарті TIA/EIA/S-95-A Асоціації телекомунікацій (TIA) (Стандарт сумісності мобільних і базових станцій для стільникових систем широкого спектру подвійного режиму). Ця передавальна система може бути реалізована, наприклад, у базовій станції стільникової системи передачі для передачі сигналів до мобільних телефонів у комірці, що оточує базову станцію.

Вхідною лінією 11 позначено сигнал мови або даних, який може бути цифровим або аналоговим. Ця вхідна лінія може бути аналоговим або цифровим каналом зв'язку комунальної комутаторної телефонної мережі (ККТМ) або іншого джерела мовного сигналу. Якщо вхідний мовний сигнал є аналоговим, сигнал квантується і цифрується у АЦП (не показаному). Джерело 12 даних змінної швидкості приймає цифровані порції мовного сигналу і кодує цей сигнал, щоб одержати пакети, тобто кадри кодованої мови однакової довжини. Джерело 12 даних може, наприклад, перетворювати цифровані порції вхідної мови у цифрові параметри, які репрезентують вхідний голосовий сигнал, використовуючи для цього процедуру лінійного кодування з прогнозуванням (ЛКП). У одному з втілень джерелом даних змінної швидкості є вокодер змінної швидкості, [описаний у патенті США 5 414 796]. Таке джерело створює пакети даних змінної швидкості з одною з чотирьох швидкостей у кадрі - 9600біт/с, 4800біт/с, 2400біт/с і 1200біт/с, які називають повною швидкістю, половинною швидкістю, швидкістю 1/2 і швидкістю 1/8. Пакети, кодовані з повною швидкістю містять 172 інформаційних біт, кодовані з половинною швидкістю - 80 інформаційних біт, кодовані з швидкістю 1/4-40 інформаційних біт і кодовані з швидкістю 1/8-16 інформаційних біт. Незалежно від розміру пакети мають тривалість 20мс. У інших системах можуть використовуватись інші швидкості даних і інші розміри пакетів. У цьому тексті терміни пакет і кадр є синонімами.

Пакети кодуються і передаються з різними швидкостями для компресування даних залежно почасти від складності або кількості інформації, репрезентованої кадром. Наприклад, якщо вхідний голосовий сигнал не має варіацій або варіації малі, можливо, тому, що промовлювач не розмовляє, інформаційні біти відповідного пакету компресуються і кодуються з швидкістю 1/9. Таке компресування призводить до втрати розрізнення у відповідній частині голосового сигналу, але, оскільки ця частина несе малу кількість або зовсім не несе інформації, це зниження розрізнення звичайне Е непомітним. Якщо ж вхідний голосовий сигнал, що відповідає пакету, несе багато інформації, можливо, внаслідок активної розмови, пакет кодується з повною швидкістю і інформаційні біти пакету не зазнають компресії.

Такі компресія і кодування використовуються для обмеження середньої кількості сигналів, що передаються одночасно, і завдяки цьому смуга частот передавальної системи використовується більш ефективно, що дозволяє одночасно обробляти більшу кількість телефонних сеансів зв'язку.

Пакети змінної швидкості, які створює джерело 12 даних, надходять до пакетувальника 13, який селективно додає біти КЦН (код циклічної надмірності) і хвостові біти, і від нього - до кодера 14, який кодує біти цих пакетів для виявлення і корекції помилок. У одному з втілень кодер 14 є згортаючим послідовним кодером Вітербі. Кодовані з згорткою символи надсилаються до модулятора 16, який генерує модульований сигнал. Такий модулятор [описано у патентах 5 103 459 і 4 901 307]. Модульований сигнал надходить до ЦАП 22 для перетворення у аналоговий сигнал і потім до передавача 24, який підсилює сигнал і підвищує його частоту для передачі антеною 26.

Фіг.2 ілюструє відповідні компоненти мобільного телефону 28 або іншої мобільної станції, що приймає переданий сигнал. Антена 30 приймає цей сигнал, після чого він підсилюється і його частота знижується, якщо необхідно, у приймачі 31. Далі сигнал де-модулюється демодулятором 32 у потік символів, які залишаються кодованими з згорткою. Після цього сигнал надходить до послідовного декодера 34 який декодує цей потік символів і також розділяє прийнятий сигнал на пакети і визначає відповідну кадрову швидкість для кожного з пакетів. Цю швидкість можна визначити, наприклад, через тривалість окремих біт кадру. Опис типового послідовного декодера Вітербі можна знайти [у заявці 08/126 477 на патент США від 24/09/1993], включеній сюди посиланням.

Для декодування потоку символів у декодері 34 передбачено використання відгалуженого блоку 36 метрики помилок, який приймає символи від модулятора, і блок 38 до-дання-порівняння-обрання (ДПО), який генерує біти рішення, грунтуючись на символах. Для підвищення ефективності декодер крокує назад від того, що він вважає метрикою найкращого режиму, використовуючи для цього блок 40 зворотних кроків, який обробляє біти рішення, прийняті від ДПО 38. У кожному циклі обробки блок зворотних кроків зберігає у пам'яті 41 зворотних кроків 2^{K-1} біт рішення, де К - довжина обмеження коду декодера. Режим, що відповідає найнижчій метриці найкращого режиму, передається від ДПО до блоку зворотних кроків як найкращий режим.

Після завершення L циклів обробки починається утворення ланцюжка зворотних кроків, яке здійснюється під керуванням контролера 42 зворотних кроків. Утворення ланцюжка зворотних кроків починається з зчитування з пам'яті зворотних кроків біт рішення для найкращого режиму попереднього (L-1) циклу обробки. Зчитаний біт рішення зсувається на місце наймолодшого біту найкращого стану. Далі блок зворотних кроків зчитує з пам'яті зворотних кроків біти рішення для найкращого режиму циклу L-2 обробки. Ця процедура виконується L разів з зчитуванням наприкінці біт рішення для найкращого режиму циклу 0 обробки. Останній біт рішення є бітом декодованої інформації. Кожний зчитаний біт модифікує адресу наступного зчитування. У наступному циклі

UA 75863 C2

UA 75863 C2

обробки (L+1) вся процедура повторюється, причому біти рішення щодо режиму зчитуються з циклів від L до 1. Це продовжується протягом стількох циклів обробки, скільки потрібно для одержання необхідної кількості інформаційних біт у даній системі.

Фіг.3 ілюструє приклади виконання зворотних кроків. Якщо процедура зворотних кроків виконується після 4 циклів обробки і після цих циклів найкращому режиму відповідає код 101, то зчитування, що виконуються для завершення процедури зворотних кроків, позначено входами, затемненими сірим. Буде зчитаний перший режим 101 циклу 3 обробки, потім режим 011 циклу 2, потім режим 111 циклу 1, потім режим 110 циклу 0, що дасть вихідний біт рішення 0. Якщо на початку циклу 5 обробки найкращому режиму відповідає код 010, то перші результати зчитування дадуть найкращий режим 101. Отже, наступні три зчитування будуть здійснені, як і раніше, а саме, як послідовність кроків, затемнених сірим. Цього разу, хоча вихідний біт рішення зчитаний з циклу 1, результуючий біт рішення буде 0. Якщо на початку циклу 6 обробки найкращому режиму відповідає код 001, то перші результати зчитування дадуть найкращий режим 010, а наступні три зчитування будуть здійснені, як і раніше, причому вихідний біт рішення зчитаний з циклу 2 і результуючий біт рішення будуть 1.

Декодер 34 (Фіг.2) формує декодований пакет разом з сигналом, що ідентифікує кадрову швидкість у цьому пакеті, які надсилаються до вузла 43 перевірки якості кадру, який намагається підтвердити відсутність помилок передачі або помилок визначення кадрової швидкості. У типовому втіленні вузол 43 перевіряє КЦН, частоту появи хибних символів і метрику Ямамото. Для перевірки частоти появи хибних символів вузол 43 перевірки рекодує символи декодованого пакету і порівнює рекодовані символи з символами, уведеними у вузол 43 для виявлення розбіжностей. Для перевірки метрики Ямамото вузол 43 перевірки надсилає прийняті кадри до матричного декодера і визначає прийнятність одержаної метрики. Прийнятні кадри спрямовуються до декодера 44 мови для перетворення назад у цифрові голосові сигнали. Ці сигнали перетворюються у аналогові у ЦАП (не показаному) для одержання вихідного сигналу, що подається до гучномовця 46 мобільного телефону, завдяки чому оператор може чути мову, введену у систему (лінія 11 Фіг.1).

Мобільний телефон (Фіг.2) може мати додаткові компоненти для введення аналогового мовного сигналу від оператора і для обробки і передачі сигналу з використанням ПДКУ. Ці додаткові компоненти можуть бути подібними до зображених на Фіг.1. Крім того передавальна система Фіг.1 може мати додаткові компоненти для прийому сигналів, переданих стільниковим телефоном, їх обробки і надсилання вихідного аналогового або цифрового мовного сигналу, наприклад, у лінію ККТМ. Додаткові компоненти Фіг.1 можуть бути подібними до зображених на Фіг.2.

Отже, важливим компонентом усієї системи є послідовний декодер Вітербі, призначений декодувати передані символи. Як уже відзначалось, у декодері 34 використовується процедура зворотних кроків. Для одержання суттєвого підвищення ефективності довжина ланцюжка зворотних кроків має у 3 - 5 разів перевищувати довжину обмеження кодера (K=9 для ПДКУ), причому ефективність зростає з зростанням довжини ланцюжка зворотних кроків. Однак, при цьому збільшуються розміри схеми і зростає споживання енергії. Збільшення розмірів схеми зумовлюється потребою у більшій пам'яті для зберігання біт рішення ланцюжка зворотних кроків. Наприклад, для декодера з довжиною обмеження K для кожного інформаційного біту треба зберігати 2^{K-1} біт рішення. При довжині ланцюжка зворотних кроків L необхідно зберігати $L*2^{K+1}$ біт. Споживання енергії зростає тому, що для формування одного біту даних необхідно виконати L зчитувань. Крім того, зростає затримка на виконання зворотних кроків. Подібно до системи з ПДКУ з використанням декодера Вітербі такі ж проблеми виникають і у інших системах, де використовуються такі і подібні їм декодери.

Отже, бажано винайти спосіб суттєвого зниження споживання енергії і часу на обробку у блоці зворотних кроків при невеликому збільшенні розмірів схеми. Це і є головним об'єктом винаходу.

Однією з задач винаходу є удосконалення послідовного декодера Вітербі, призначеного для декодування потоку кодованих з згорткою символів з використанням пам'яті зворотних кроків для зберігання біт рішення для кожного з сукупності циклів обробки. Удосконалення полягає у запровадженні кешу зворотних кроків, пов'язаного з пам'яттю зворотних кроків і призначеного для зберігання біт рішення, визначених у попередньому циклі обробки.

У одному з типових втілень послідовний декодер Вітербі включає відгалужений блок метрики помилок, ДПО, блок зворотних кроків з пам'яттю, повний кеш зворотних кроків і контролер зворотних кроків. Кеш зворотних кроків може приймати усі зчитування. Інше типове втілення не передбачає повного кешу зворотніх кроків, але блок зворотних кроків включає пам'ять на L+1 біт (тут і далі вважається, що це пам'ять з довільним доступом (RAM)), реверсивний лічильник і зсувний регістр, що емулюють кеш зворотних кроків. У ще одному втіленні використовується зсувний регістр на L біт замість комбінації пам'яті на L+1 біт з реверсивним лічильником. У різних втіленнях блок зворотних кроків пристосований виконувати лише одне зчитування зворотних кроків у кожному циклі обробки або виконувати m зчитувань зворотних кроків у кожному циклі обробки перед спробою використати кеш. У інших втіленнях блок зворотних кроків пристосовано виконувати a і потім b зчитувань протягом кожної процедури зворотних кроків, причому після a зчитувань кеш перевіряється для кожного чергового зчитування, доки у кожній процедурі зворотних кроків не буде виконано b зчитувань або доки на буде досягнуто збігу. У ще одному втіленні блок зворотних кроків пристосовують виконувати процедури зворотних кроків для багатьох циклів обробки, а не єдиного такого циклу. Винахід включає також різні комбінації елементів цих втілень.

У різних втіленнях застосування кеш-пам'яті для біт рішення з попередніх циклів обробки дозволило досягти значних знижень споживання і витрат часу на обробку з лише незначним ускладненням схем.

Особливості, об'єкти і переваги винаходу наведені у подальшому детальному описі з посиланнями на креслення, у яких:

UA 75863 C2

Фіг.1 - блок-схема компонентів передавальної системи змінної швидкості з ПДКУ,

Фіг.2 - блок-схема компонентів мобільного телефону або іншої мобільної станції, що приймає сигнал, переданий передавальною системою з ПДКУ (Фіг.1) і декодує сигнал декодером Вітербі, який містить блок зворотних кроків,

Фіг.3 - схема процедури зворотних кроків, яку виконує блок зворотних кроків мобільного телефону Фіг.2,

Фіг.4 - блок-схема, що на вищому рівні ілюструє відповідні компоненти мобільного телефону або іншої мобільної станції, у якій, згідно з винаходом, використано послідовний декодер Вітербі з блоком і кешем зворотних кроків,

Фіг.5А, 5В - детальна ілюстрація першого втілення блоку зворотних кроків мобільного телефону Фіг.4,

Фіг.6А, 6В - детальна ілюстрація другого втілення блоку зворотних кроків мобільного телефону Фіг.4,

Фіг.7 - частина третього втілення блоку зворотних кроків мобільного телефону Фіг.4.

Фіг.4 ілюструє належні компоненти мобільного телефону 128 або іншої мобільної станції, що приймає переданий сигнал ПДКУ. Вузли мобільного телефону 128 працюють подібно до телефону Фіг.2 і будуть описані лише побіжно.

Антена 130 приймає сигнал, після чого він підсилюється і його частота знижується, якщо необхідно, у приймачі 131. Далі сигнал демодулюється демодулятором 132 з перетворенням у потік символів, кодованих з згорткою. Після цього сигнал надходить до модифікованого послідовного декодера 134, який декодує цей потік символів, використовуючи відгалужений блок 136 метрики помилок, ДПО 138 і блок 140 зворотних кроків. Блок 140 має пам'ять 141 зворотних кроків, контролер 142 зворотних кроків і кеш 145 зворотних кроків, призначений лише для зчитувань. Режим, що відповідає найнижчій метриці найкращого режиму, передається від ДПО до блоку зворотних кроків як найкращий режим, і зберігається у пам'яті зворотних кроків, а також у кеші зворотних кроків, звідки вони можуть бути легко зчитані. Як буде описано нижче, декодер не обов'язково має фактично включати повну окрему кеш-пам'ять, як це показано на Фіг.4. Однак, у подальшому описі для кращого розуміння роботи системи зворотних кроків з кешем ми вважатимемо, що використовується повна кеш-пам'ять.

Процедура зворотних кроків з використанням кеш-пам'яті (Фіг.4) виконується шляхом зчитування з кешу 145 зворотних кроків біт рішення, що визначають метрику найкращого режиму, якщо поточна метрика найкращого режиму, обчислена після одного зчитування у новому циклі обробки, збігається з початковою метрикою найкращого режиму останнього циклу обробки. У випадку незбігу цих метрик виконуються звичайні зворотні кроки. Зокрема, на початку циклу обробки метриці найкращого режиму призначається змінна encstate. З пам'яті зворотних кроків зчитується перший біт рішення, місце якого визначається через encstate, і зсувається у наймолодший біт encstate. Далі нове значення encstate порівнюється з параметром, позначеним last_beststate, який містить метрику найкращого режиму попереднього циклу обробки. Якщо значення збігаються, зникає потреба у виконанні додаткових L-1 зчитувань для завершення процедури зворотних кроків. Завершальний біт може бути просто зчитаний з кешу (encstate і last_beststate не показані на Фіг.4, але є у інших Фіг., розглянутих далі). Якщо необхідно виконати L-1 додаткових зчитувань, блок зворотних кроків зчитує з пам'яті зворотних кроків біти рішення, які відповідають значенню encstate для біт рішень, записаних протягом циклу обробки L-2. Ця процедура виконується усі L разів з зчитуванням наприкінці біту рішення обчисленого encstate для циклу обробки 0. Цей остаточний біт рішення є декодованим інформаційним бітом. Кожний зчитаний біт змінює адресу для подальшого зчитування. У наступному циклі обробки (L+1) процедура повторюється з зчитуванням біт рішення для режиму для циклів обробки від L до 1. Це продовжується протягом стількох циклів обробки, скільки потрібно для одержання необхідної кількості інформаційних біт у даній системі.

Після завершення звичайної процедури зворотних кроків, уся послідовність біт рішення, генерована цією процедурою, зберігається у кеші 145 зворотних кроків і, отже, зникає необхідність виконувати повну процедуру зворотних кроків у наступному циклі обробки. Зокрема, після зворотних кроків першого циклу обробки у подальших циклах обробки перше зчитування призведе до прийняття змінною encstate значення, яке вона мала на початку попереднього циклу і тому значення encstate збігатиметься з значенням last_beststate. Це має місце не завжди, але у більшості випадків внаслідок властивості шляхової конвергенції, притаманної кодам з згорткою. Отже, найбільш імовірно, що у даному циклі обробки L-1 зчитувань будуть такими ж, як у попередньому циклі і остаточний біт рішення буде попереднім до останнього біту, зчитаного у попередньому циклі. Таким чином, використання кешу зворотних кроків дозволяє просто зчитати з кешу біти рішення L-1 зчитувань, включаючи остаточний біт рішення, без повторного обчислення, що підвищує ефективність роботи.

Декодер 134 формує декодований пакет разом з сигналом, що ідентифікує кадрову швидкість у цьому пакеті, і надсилає їх до вузла 143 перевірки якості кадру, який намагається підтвердити відсутність помилок передачі або помилок визначення кадрової швидкості, використовуючи для цього перевірку КЦН, частоти появи хибних символів і метрики Ямамото. Прийнятні кадри спрямовуються до декодера 144 мови для зворотного перетворення у цифрові голосові сигнали. Ці сигнали перетворюються у аналогові у ЦАП (не показаному) для одержання вихідного сигналу, що подається до гучномовця 146 мобільного телефону. Мобільний телефон (Фіг.4) може мати додаткові компоненти (не показані) для введення аналогового мовного сигналу від оператора і для обробки і передачі сигналу з використанням ПДКУ. Ці додаткові компоненти можуть бути подібними до зображених на Фіг.1.

Отже, Фіг.4 ілюструє на високому рівні мобільний телефон, оснащений послідовним декодером Вітербі, який має блок зворотних кроків з повним окремим кешем зворотних кроків, пристосованим для зберігання зчитувань. Логіка кеш-пам'яті за своєю природою забезпечує зберігання копії біт рішення різних зчитувань у її власній пам'яті. Загальної економії енергії можна досягти, якщо споживання енергії кеш-пам'ятью не перевищує зниження споживання, зумовленого відповідною відмовою від доступу до пам'яті зворотних кроків. Крім того,

UA 75863 C2

UA 75863 C2

залежно від втілення може бути знижений час на декодування порівняно з часом, що витрачається без використання кешу. Такий виграш часу зумовлюється тим, що у випадку збігу система виконує лише одне зчитування з кешу замість L-1 додаткових зчитувань з пам'яті зворотних кроків, якщо збігу не одержано. Зниження часу на декодування є особливо значним у системах, де пам'ять зворотних кроків працює повільно, а кеш швидко.

Порівняння роботи блоку зворотних кроків з кешем і таких же блоків без нього дає такі результати. У системі стандарту IS-95 для каналу швидкості 1 з відносною кількістю помилок 1% послідовний декодер Вітербі без кешу може виконати 289 процедур зворотних кроків з L=63. Відзначимо, що останні 72 біти пакету одержуються однією процедурою зворотних кроків. Повна кількість зчитувань з пам'яті зворотних кроків становить 289*63=18207. Для 100 кадрів даних encstate після одного зчитування збігається з значенням beststate попереднього циклу обробки у середньому приблизно 233 рази на кадр (з 289 процедур зворотних кроків). Отже, при використанні кешу у блоці зворотних кроків повна кількість зчитувань з пам'яті зворотних кроків становитиме лише 56*63+233=3761, тобто середня економія на кадр - 14446 зчитувань. У системі стандарту IS-95 для каналу швидкості 2 з відносною кількістю помилок 1% послідовний декодер Вітербі без кешу може виконати 437 процедур зворотних кроків з L=95. Відзначимо, що останні 104, біти пакету одержуються однією процедурою. Повна кількість зчитувань з пам'яті зворотних кроків становить 437*95=141515. Для 23 кадрів даних encstate після одного зчитування збігається з значенням beststate попереднього циклу обробки у середньому приблизно 383 рази на кадр (з 457 процедур зворотних кроків). Отже, при використанні кешу у блоці зворотних кроків повна кількість зчитувань з пам'яті зворотних кроків становитиме лише 99*95+338=9743, тобто середня економія на кадр - 31772 зчитувань. У інших системах результати можуть бути іншими.

Блок зворотних кроків (Фіг.4) може бути реалізований різними схемами, включаючи такі, що забезпечують подальше зниження споживання енергії або зменшення розмірів схеми. Дали описано деякі конкретні схеми.

Фіг.5А, 5В ілюструють більш досконале втілення блоку зворотних кроків, яке забезпечує додаткове зниження кількості зчитувань з пам'яті зворотних кроків шляхом використання невеликої пам'яті об'ємом L+1 біт або регістру для зберігання біт рішення, зчитаних у кожному циклі обробки. Блок зворотних кроків включає пам'ять 204 зворотних кроків на L+1 біт, реверсивний лічильник 206 і зсувний регістр 208, які мають зв'язки з іншими регістрами і логічними елементами, як це показано на Фіг. Після L циклів обробки перша процедура зворотних кроків завершується. Найкращий режим beststate для попереднього циклу зберігається у зсувному регістрі 208, вихід якого позначено раніше як encstate. L біт, зчитаних з пам'яті 202 зворотних кроків, зберігаються у L+1-бітовій пам'яті 204 (один біт додано для спрощення схеми). Окремий регістр 210 використовується для стеження за попередніми значеннями beststate, позначеними раніше як last_beststate. У наступному циклі нове значення beststate фіксується у зсувному регістрі 208. Перше зчитування циклу обробки дає біт рішення, який зсувається у положення наймолодшого біту регістру 208. Цей біт, як і раніше, також зберігається у пам'яті 204. Якщо encstate тепер збігається з last_beststate, то біт, що відповідає найменшому/найстарішому циклу, видаляється з пам'яті 204 і стає вихідним бітом. Цей біт є тим бітом, який був би одержаний в результаті повної процедури зворотних кроків з пам'яттю зворотних кроків; просто цей біт був одержаний коротшим і простішим шляхом. Якщо після одного зчитування encstate не збігається з last_beststate, виконується повна процедура зворотних кроків з одночасним заповненням усіх L+1 біт пам'яті 204. У будь-якому випадку наприкінці циклу обробки значення beststate заноситься у last_beststate і подальший цикл обробки виконується, як щойно описаний. Додання L+1-бітової пам'яті у блок зворотних кроків зменшує кількість процедур зчитування з пам'яті зворотних кроків і додатково знижує споживання енергії усією схемою декодера.

Втілення Фіг.5А, 5В може здаватись дещо складним, але порівняно з варіантом з блоком зворотних кроків з доступом до кешу у кожному циклі обробки блок зворотних кроків Фіг.5А, 5В вимагає лише додання реверсивного лічильника 206, L+1-бітової пам'яті 204 (або іншого запам'ятовуючого регістру) і різних однобітових регістрів і комбінаторної логіки. Це дає ту перевагу, що у циклах обробки з одержанням збігу необхідно виконати лише одне (замість L) зчитування з пам'яті зворотних кроків і одне зчитування з L+1-бітової пам'яті і запис, який є відносно незначним. У випадку незбігу необхідно виконати L зчитувань з пам'яті зворотних кроків і L записів у L+1-бітову пам'ять. Ці записи не вимагають багато енергії, оскільки розмір пам'яті незначний і частота незбігів звичайно є невеликою. При використання запам'ятовуючого регістру споживання енергії є ще меншим.

Блок зворотних кроків Фіг.5А, 5В одержує ряд сигналів керування, які генеруються іншими схемами (не показанами). Далі наведено ці сигнали.

reset - сигнал початкового встановлення логіки.

beststate - сигнал надходить від ДПО 138 (Фіг.4) і вказує на режим з найнижчою метрикою помилок для останнього циклу обробки. Цей сигнал змінюється до імпульсів start_chainback і після імпульсів done_chainback.

decision bit - генерується у ДПО. Це дані, що підлягають зберіганню у пам'яті зворотних кроків.

start_chainback - імпульси на початку кожного циклу обробки, які вказують на можливість виконання процедури зворотних кроків.

done_chainback - імпульси наприкінці кожного циклу обробки, які вказують на завершення процедури зворотних кроків.

enable_cache_read - уможливлює використання L+1-бітової пам'яті або регістру для одержання вихідного біту. Імпульси enable_cache_read синхронізують 1 часовий цикл з першим імпульсом cbread у кожному циклі обробки.

cbread - у кожному циклі обробки надсилає L імпульсів для виконання L зчитувань з пам'яті зворотних кроків. Перший надсилається після start_chainback і останній - перед done_chainback. Якщо має місце збіг,

UA 75863 C2
UA 75863 C2

виконується лише перше зчитування, а решта маскується схемою.

cbwrite - імпульс надсилається кожного разу, коли біт рішення готовий для збереження у пам'яті зворотних кроків.

chram_addr - нормальна адреса, що ініціює пам'ять зворотних кроків. Для економії енергії ці лінії маскуються і утримуються у статичному стані у випадку, коли останні L - 1 зчитувань з пам'яті зворотних кроків не виконуються.

do_compare - внутрішній сигнал, що вказує на необхідність використання результату порівняння, тобто блок зворотних кроків порівнює поточне значення encstate з значенням beststate останнього циклу обробки, збережене у last_beststate, і визначає наявність або, відсутність збігу.

match - внутрішній сигнал, що вказує на збіг значення encstate з значенням last_beststate останнього циклу обробки.

mismatch - внутрішній сигнал, що вказує незбіг значення encstate з значенням last_beststate після першого зчитування з пам'яті зворотних кроків.

cbread_muxed - внутрішній сигнал, подібний до cbread, але маскується у випадку збігу.

read_last_bit - внутрішній сигнал для пересилання вихідного біту з L+1-бітової пам'яті у регістр.

chram_dout - внутрішній сигнал , який є бітом, зчитаним з пам'яті зворотних кроків.

Фіг.6А, 6В ілюструють втілення, подібне до ілюстрованого Фіг.5А, 5В, де замість L+1-бітової пам'яті з реверсивним лічильником використано L-бітовий зсувний регістр. Зокрема, блок зворотних кроків Фіг.6А, 6В включає пам'ять 302 зворотних кроків, L-бітовий зсувний регістр 305 і зсувний регістр 308, пов'язані з різними регістрами і елементами логіки. Після першого зчитування у циклі обробки зчитаний біт зсувається у зсувний регістр 308, виходом якого є encstate. Якщо encstate тепер збігається з last_beststate, зчитаний біт зсувається на місце найстаршого біту L-бітового зсувного регістру, а наймолодший біт цього регістру стає вихідним бітом процедури зворотних кроків. Якщо encstate не збігається з last_beststate, зчитаний біт зсувається на місце наймолодшого біту L-бітового зсувного регістру, а інші L-1 зчитаних біт також зсуваються у наймолодший біт.

Слід відзначити, що L-бітовий зсувний регістр має бути здатним зсувати у обох напрямках, тобто L-бітовий зсувний регістр Фіг.6А, 6В відрізняється від стандартного тим, що включає додатковий вхід "ліворуч", що визначає напрямок зсуву. Крім того, при кожному зчитуванні біту з пам'яті зворотних кроків усі L бітів необхідно зсувати відразу, що збільшує споживання енергії порівняно з втіленням Фіг.5А. 5В. У іншому втіленні (не ілюстрованому) споживання енергії додатково знижується доданням декодуючої логіки для обрання кожного біту окремо з завантаженням окремо кожного біту зсувного регістру. Тоді при завантаженні L біт кожний з них завантажується окремо. Отже, регістр має виконувати зсув лише при наявності збігу (одноразово у циклі обробки), що знижує витрати енергії. Ще одне втілення передбачає схему перевірки збігу після перших двох або більше зчитувань у процесі виконання зворотних кроків, що збільшує імовірність збігу і додатково зменшує споживання енергії і тривалість декодування. Таку схему можна використати у втіленнях Фіг.5А, 5В або Фіг.6А, 6В і у інших втіленнях.

У описаних вище втіленнях схема блоку зворотних кроків працює у кожному циклі обробки і виконує одне зчитування перед прийняттям рішення про використання (або відмову від нього) кешу для завершення процедури зворотних кроків. Фіг.7 ілюструє втілення, у якому у кожному циклі обробки передбачено виконання m зчитувань перед прийняттям рішення про використання (або відмову від нього) кешу. Зокрема, на Фіг.1 зображено схему, призначену генерувати сигнал match на підставі m зчитувань. Схема Фіг.7 може бути використана у блоках зворотних кроків з кешем, ілюстрованих Фіг.5А, 5В або Фіг.6А, 6В, замість відповідної схеми формування сигналу match, використаної у них. Схема match Фіг.7 виконує у циклі обробки m зчитувань і порівнює поточне значення encstate з значенням є, одержаним після m-1 зчитувань, виконаних протягом попереднього циклу. У цьому втіленні імпульс сигналу enable_cache_read синхронізований з m-им зчитуванням. Крім того, замість збереження значення beststate на початку кожної процедури зворотних кроків зберігається значення encstate після m-1 зчитувань. Вибір m залежить від співвідношення між кількістю зчитувань (m) у кожному циклі і імовірністю збігу. Збільшення m знижує імовірність збігу після m зчитувань. Слід відзначити, що схема Фіг.7 приймає додатковий сигнал save_state для фіксації encstate після m-1 зчитувань. Сигнал do_compare також відрізняється від описаного вище тим, що виникає після m зчитувань, а не одного, і використовується для фіксації значення encstate, зафіксованого попереднім зчитуванням, що робить його доступним для порівняння у наступному циклі.

У іншому, більш узагальненому втіленні, замість перевірки значення encstate після 1 або m зчитувань у кожному циклі обробки encstate порівнюється після a і потім b зчитувань, тобто після a зчитувань зворотних кроків encstate порівнюється з значенням encstate, збереженим у попередньому циклі після a-1 зчитувань, а після наступного зчитування (a+1) encstate порівнюється з значенням encstate, збереженим у попередньому циклі після a зчитувань і т. д., доки у цьому циклі обробки не буде виконано b зчитувань.

У такому втіленні значення encstate для b-a+1 режимів зберігаються, бажано, у зсувному регістрі. Кожне наступне значення encstate є просто результатом лівого зсуву попереднього значення з новим наймолодшим бітом. Сигнал enable_cache_read повторюється для усіх зчитувань і припиняється після b зчитувань або після появи збігу. Вибір a і b визначається співвідношенням таких факторів, як складність і збереження енергії. Втілення Фіг.5А, 5Б і Фіг.6А, 6В відповідають значенням a=b=1, тобто одному зчитуванню, після якого схема швидко вирішує, був збіг або ні. Описане вище втілення, у якому виконуються m зчитувань, відповідає випадку a=m, b=m, коли порівняння здійснюється після m зчитувань.

Вибір значень a і b для кожної системи визначається типом системи, статистикою конвергенцій (тобто

UA 75863 C2

типовою кількістю зчитувань, необхідних для конвергенції до шляху, зчитаного у попередньому циклі обробки), складністю обладнання і енергетичними вимогами. Для спрощення обладнання b-a має бути малим. Для зниження споживання енергії a має бути малим, а b залежати від статистики системи. Взагалі, збільшення b підвищує імовірність знаходження збігу.

У інших втіленнях процедури зворотних кроків здійснюються у багатьох циклах обробки. У описаних вище втіленнях було передбачене одноразове виконання зворотних кроків у кожному циклі. Однак, ці втілення можна модифікувати і виконувати процедуру у кількох циклах обробки. Наприклад, у втіленні, де процедура зворотних кроків виконується кожні 4 цикли обробки і результатом є 4 декодовані біти, сигнальний імпульс enable_cache_read можна генерувати лише при четвертому зчитуванні зворотних кроків. Однак, генерування enable_cache_read для кожного 4-го зчитування не є обов'язковим. Навіть коли процедура зворотних кроків виконується на 4 циклах обробки, визначення, коли порівнювати encstate, може залежати від значень a і b. Сигнал enable_cache_read може бути повторений 4 рази, якщо мав місце збіг і це дало 4 біти декодованої інформації, зчитаних з кешу. Інше втілення передбачає виконувати процедуру, використовуючи 4-бітові відрізки (або іншого розміру), і тоді при виконання процедури зворотних кроків при виявленні збігу система здійснює перевірку після 4 зчитувань. У цьому випадку система зчитує останні 4 біти кешу і видає їх, у іншому разі система продовжує виконання зворотних кроків і зберігає останні 4 зчитування з пам'яті зворотних кроків.

Описані вище втілення стосуються канальних інформаційних систем з обробкою пакетованої інформації, тобто блок даних кодується з згорткою з доданням хвостових біт, для переустановлення стану кодера між пакетами. Отже, система чекає L+K циклів обробки, потім починає зворотні кроки і наприкінці виконує одну кінцеву процедуру зворотних кроків, формуючи L+K біт. Інші втілення винаходу передбачають використання у непакетованих інформаційних каналах зв'язку, наприклад, синхроканалах або пейджерних каналах згідно з IS-95. Для таких каналів дані кадруються, але між кадрами стан кодера не переустановлюється. Отже, декодер виконує процедуру зворотних кроків у кожному циклі обробки. Зрозуміло, що принципи винаходу можуть бути застосовані у майже будь-якому декодері Вітербі незалежно від типу каналів зв'язку системи.

Наведений вище опис типових втілень був ілюстрований схемами елементів пристроїв. Залежно від втілення, кожний апаратний елемент або його частина може бути реалізований схемно, програмно, програмно з використанням ПЗП або комбінаціями цих способів. Зрозуміло, що були ілюстровані або описані детально не всі необхідні для втілення елементи, а лише необхідні для повного розуміння винаходу. Наведений опис бажаних і типових втілень дозволяє будь-якому фахівцю використати винахід, виконавши, якщо необхідно, потрібні модифікації на підставі базових принципів винаходу. Отже, описані втілення не обмежують винаходу, концепції якого мають ширше поле застосування.

Формула винаходу

1. Послідовний декодер Вітербі, який має:

приймач, призначений приймати кодований із згорткою потік символів;

схему підсумовування-порівняння-вибору, призначену генерувати множину бітів рішення із кодованого із згорткою потоку символів протягом кожного з множини циклів обробки,

модуль, який включає:

пам'ять, призначену зберігати зазначену множину бітів рішення для кожного із зазначеної множини циклів обробки, причому схема підсумовування-порівняння-вибору генерує протягом кожного нового циклу обробки біт рішення, що представляє показник оптимального стану, починаючи з поточного початкового показника оптимального стану, з подальшим виконанням наступної операції зворотної послідовності для множини бітів рішення, збережених у пам'яті зворотної послідовності, для кожного з множини циклів обробки, і

кеш-пам'ять, що з'єднана з пам'яттю і використовується для зберігання послідовності бітів рішення, доступних протягом попереднього циклу зворотної послідовності, і для виводу біта рішення, який був би згенерований у іншому випадку,

причому кеш-пам'ять крім того використовується для зберігання показника оптимального стану з попереднього циклу обробки, для зберігання послідовності бітів рішення, доступних протягом попереднього циклу обробки, і прийому показника оптимального стану для поточного циклу обробки, причому модуль додатково включає:

контролер, призначений порівнювати зсунуту версію показника оптимального стану для поточного циклу обробки з показником оптимального стану попереднього циклу обробки і, якщо має місце збіг, виконувати вибірку самого раннього біта рішення з кеш-пам'яті.

2. Послідовний декодер Вітербі за п. 1, який **відрізняється** тим, що кеш-пам'ять виконує зчитування від a до b протягом кожного циклу обробки, причому після a зчитувань кеш-пам'ять виконує перевірку при кожному подальшому зчитуванні, доки не будуть виконані b зчитувань або не буде досягнутий збіг.

3. Послідовний декодер Вітербі за п. 2, який **відрізняється** тим, a = b = m, причому m - кількість зчитувань зворотної послідовності у кожному циклі обробки, до спроби використати кеш-пам'ять.

4. Послідовний декодер Вітербі за п. 2, який **відрізняється** тим, що a = b = 1.

5. Послідовний декодер Вітербі, який має:

приймач, призначений приймати кодований зі згорткою потік символів;

схему підсумовування-порівняння-вибору, призначену генерувати множину бітів рішення із кодованого зі згорткою потоку символів протягом кожного з множини циклів обробки,

UA 75863 C2

UA 75863 C2

модуль, який включає:

пам'ять, призначену зберігати зазначену множину бітів рішення для кожного із зазначеної множини циклів обробки, причому схема підсумовування-порівняння-вибору генерує протягом кожного нового циклу обробки біт рішення, що представляє показник оптимального стану, починаючи з поточного початкового показника оптимального стану, з подальшим виконанням наступної операції зворотної послідовності для множини бітів рішення, збережених у пам'яті зворотної послідовності, для кожного з множини циклів обробки, і

кеш-пам'ять, що з'єднана з пам'яттю і використовується для зберігання послідовності бітів рішення, доступних протягом попереднього циклу зворотної послідовності, і для виводу біта рішення, який був би згенерований у іншому випадку, причому кеш-пам'ять має:

регістр із зсувом вліво для прийому показника оптимального стану поточного циклу обробки, і

(L+1)-бітову оперативну пам'ять з довільним доступом для зберігання послідовності бітів рішення, доступних протягом попередньої операції зворотної послідовності, де L - довжина зворотної послідовності.

6. Послідовний декодер Вітербі за п. 5, який **відрізняється** тим, що регістр із зсувом вліво сконфігурований під регістр-засувку показника оптимального стану для попереднього циклу обробки.

7. Послідовний декодер Вітербі, який має:

приймач, призначений приймати кодований із згорткою потік символів;

схему підсумовування-порівняння-вибору, призначену генерувати множину бітів рішення із кодованого зі згорткою потоку символів протягом кожного з множини циклів обробки,

модуль, який включає:

пам'ять, призначену зберігати зазначену множину бітів рішення для кожного із зазначеної множини циклів обробки, причому схема підсумовування-порівняння-вибору генерує протягом кожного нового циклу обробки біт рішення, що представляє показник оптимального стану починаючи з поточного початкового показника оптимального стану з подальшим виконанням наступної операції зворотної послідовності для множини бітів рішення, збережених у пам'яті зворотної послідовності, для кожного з множини циклів обробки, і

кеш-пам'ять, що з'єднана з пам'яттю і використовується для зберігання послідовності бітів рішення, доступних протягом попереднього циклу зворотної послідовності, і для виводу біта рішення, який був би згенерований у іншому випадку, причому кеш-пам'ять має:

регістр-засувку для зберігання показника оптимального стану з попереднього циклу обробки,

регістр із зсувом вліво для прийому показника оптимального стану поточного циклу обробки і для зсуву бітів рішення,

компаратор для порівняння показника оптимального стану попереднього циклу обробки із зсунутою версією показника оптимального стану поточного циклу обробки і, якщо має місце збіг, виведення сигналу про збіг, і

регістр із зсувом на L бітів для зберігання послідовності бітів рішення, доступних протягом попередньої операції зворотної послідовності, де L - довжина зворотної послідовності,

вихідну схему, з'єднану з регістром із зсувом на L бітів і компаратором, для прийому сигналу про збіг від компаратора і керування регістром із зсувом на L бітів для виводу самого раннього біта рішення, збереженого у ній.

8. Послідовний декодер Вітербі, який має:

приймач, призначений приймати кодований із згорткою потік символів;

схему підсумовування-порівняння-вибору, призначену генерувати множину бітів рішення із кодованого із згорткою потоку символів протягом кожного з множини циклів обробки,

модуль, який включає:

пам'ять, призначену зберігати зазначену множину бітів рішення для кожного із зазначеної множини циклів обробки, причому схема підсумовування-порівняння-вибору генерує протягом кожного нового циклу обробки біт рішення, що представляє показник оптимального стану, починаючи з поточного початкового показника оптимального стану, з подальшим виконанням наступної операції зворотної послідовності для множини бітів рішення, збережених у засобі пам'яті зворотної послідовності, для кожного з множини циклів обробки, і

кеш-пам'ять, що з'єднана з пам'яттю і використовується для зберігання послідовності бітів рішення, доступних протягом попереднього циклу зворотної послідовності, і для виводу біта рішення, який у іншому випадку був би згенерований,

причому кеш-пам'ять має:

регістр із зсувом вліво для зсуву показника оптимального стану поточного циклу обробки,

множину послідовних регістрів для збереження раніше зсунутих версій показника оптимального стану,

компаратор для порівняння вихідних даних регістру із зсувом вліво з вихідними даними одного з останніх з множини послідовних регістрів і, якщо має місце збіг, виведення сигналу про збіг,

(L+1)-бітову оперативну пам'ять з довільним доступом для зберігання послідовності бітів рішення, доступних протягом попередньої операції зворотної послідовності, де L - довжина зворотної послідовності, і

вихідну схему, з'єднану з (L+1)-бітовою оперативною пам'яттю з довільним доступом і з компаратором для прийому від компаратора сигналу про збіг і для керування (L+1)-бітовою оперативною пам'яттю з довільним доступом для виведення самого раннього біта рішення, збереженого у ній.

9. Спосіб виконання послідовного декодування Вітербі, який включає операції:

прийому потоку кодованих зі згорткою символів,

генерування множини бітів рішення з потоку кодованих із згорткою потоку символів протягом кожного з множини циклів обробки,

збереження зазначеної множини бітів рішення у пам'яті для кожного з зазначеної множини циклів обробки,

UA 75863 C2

UA 75863 C2

визначення протягом кожного нового циклу обробки біта рішення, що представляє показник оптимального стану, починаючи з поточного початкового показника оптимального стану, з подальшим виконанням наступної операції зворотної послідовності для множини бітів рішення, збережених у пам'яті для кожного з множини циклів обробки, і

збереження послідовності бітів рішення, доступних протягом попередньої операції зворотної послідовності в кеш-пам'яті, і виведення біта рішення, який у іншому випадку був би згенерований наступною операцією зворотної послідовності, якщо зсунута версія показника оптимального стану для нового циклу обробки збігається з показником оптимального стану останнього циклу обробки.

10. Спосіб за п. 9, який **відрізняється** тим, що операція збереження послідовності бітів рішення, доступних протягом попередньої операції зворотної послідовності у кеш-пам'яті, і виведення біта рішення, що представляє показник оптимального стану, якщо поточний показник оптимального стану для нового циклу обробки вказує на початковий показник останнього циклу обробки, включає операції:

зберігання показника оптимального стану з попереднього циклу обробки,

зберігання послідовності бітів рішення, доступних протягом попередньої операції зворотної послідовності,

зсуву показника оптимального стану для поточного циклу обробки, і

порівняння показника оптимального стану попереднього циклу обробки із зсунутим показником оптимального стану поточного циклу обробки і, якщо має місце збіг, виведення самого раннього біта рішення, збереженого протягом попередньої операції зворотної послідовності.

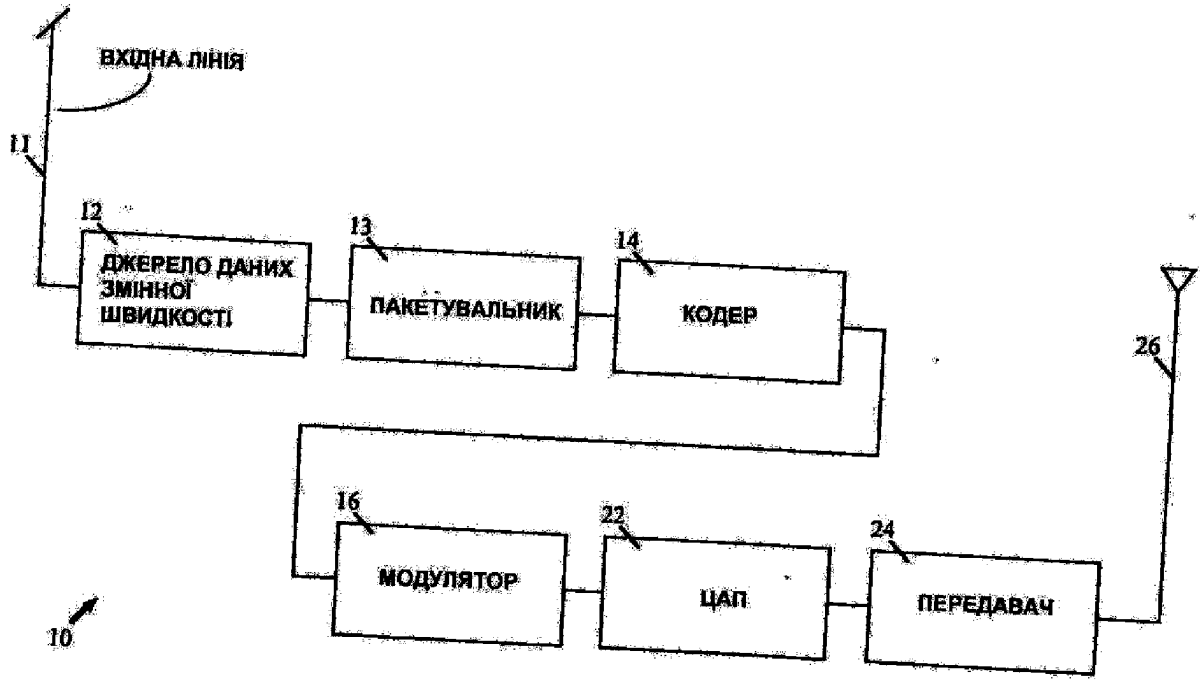
11. Спосіб за п. 10, який **відрізняється** тим, що операція збереження послідовності бітів рішення, доступних протягом попередньої операції зворотної послідовності в кеш-пам'яті, і виведення біта рішення, що представляє показник оптимального стану, якщо поточний показник оптимального стану для нового циклу обробки вказує на початковий показник контрольованого останнього циклу обробки, керує виконанням зчитувань від a до b протягом кожного циклу обробки, причому після a зчитувань засіб кеш-пам'яті виконує перевірку при кожному подальшому зчитуванні, доки не будуть виконані b зчитувань або не буде досягнутий збіг.

12. Спосіб за п. 11, який **відрізняється** тим, що a=b=m.

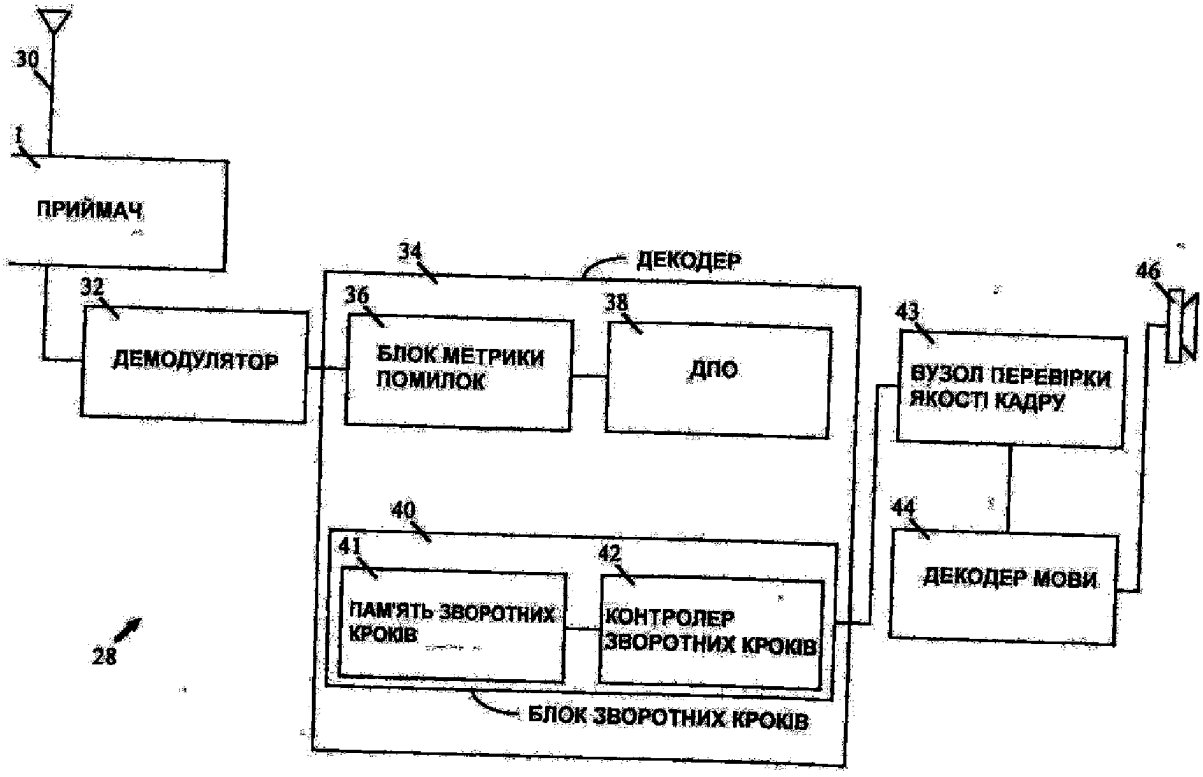
13. Спосіб за п. 11, який **відрізняється** тим, що a=b=1.

UA 75863 C2

UA 75863 C2



ФІГ.1



ФІГ.2

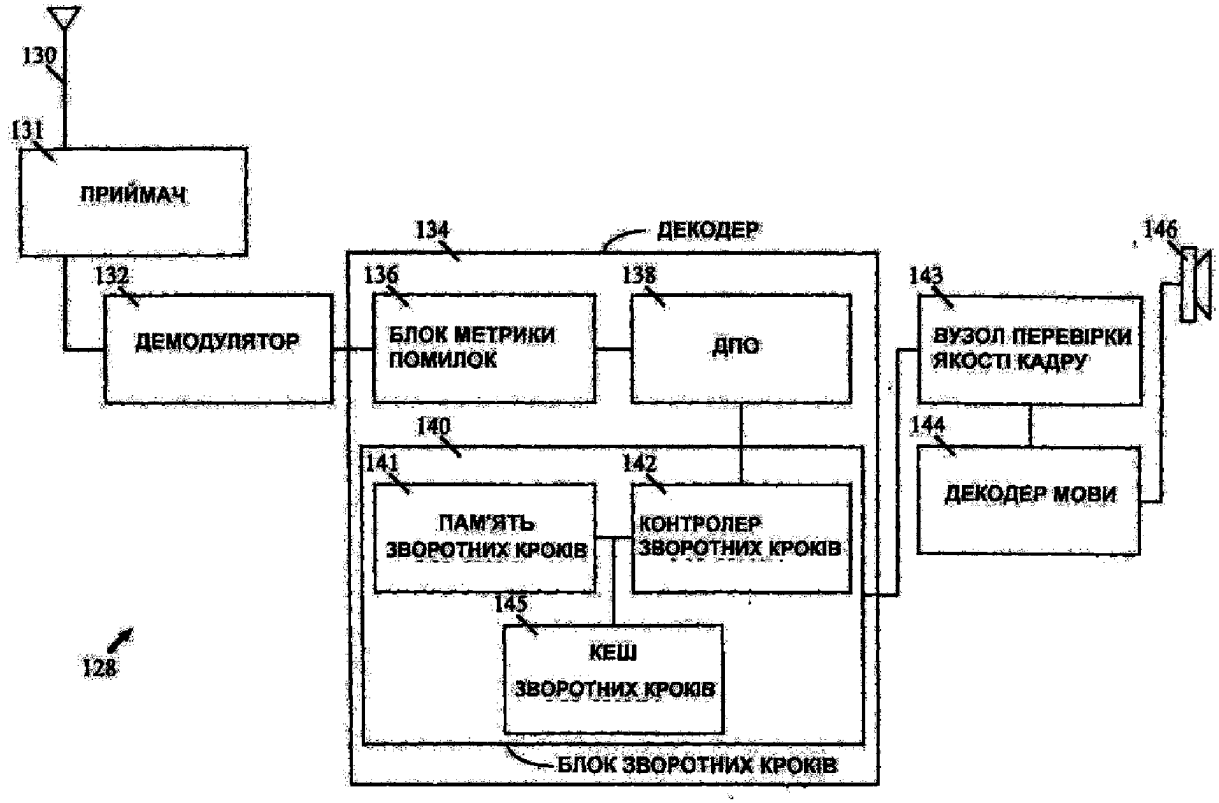
UA 75863 C2

UA 75863 C2

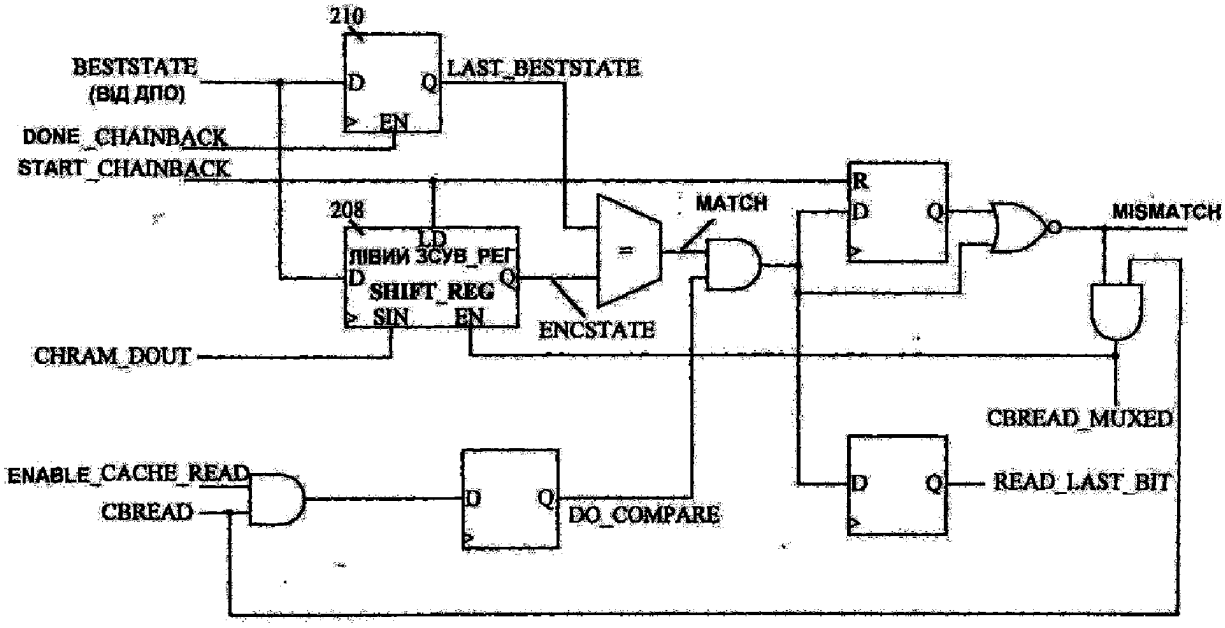
ФІГ.3

UA 75863 C2

UA 75863 C2

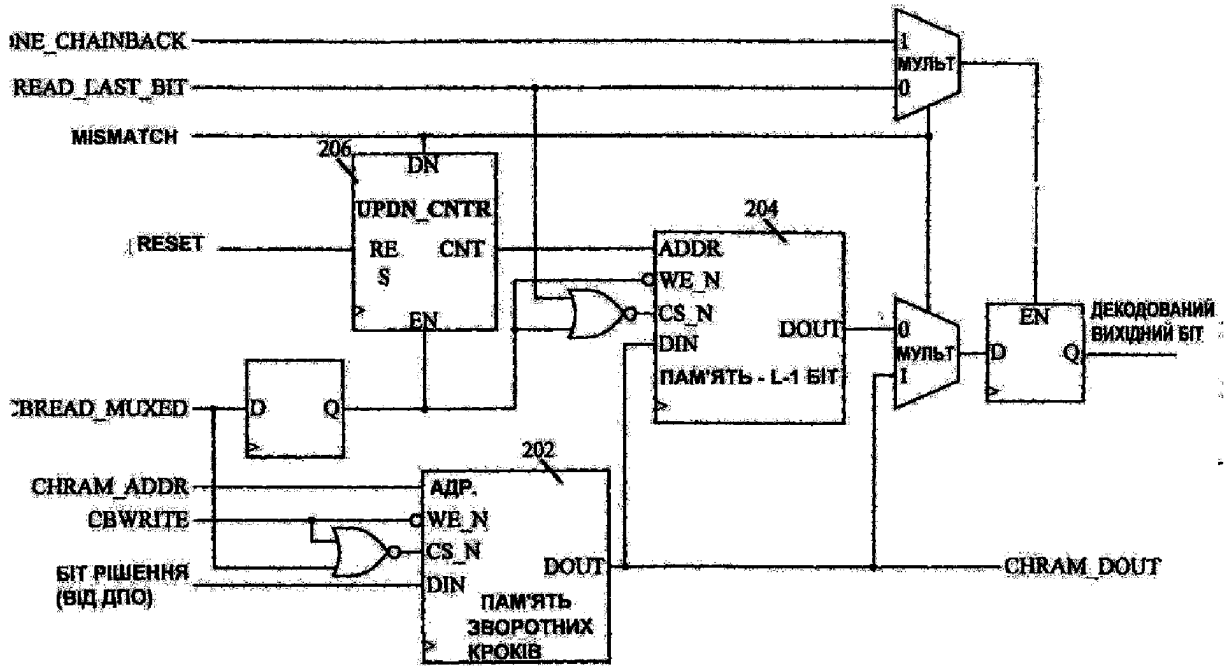


ФІГ.4

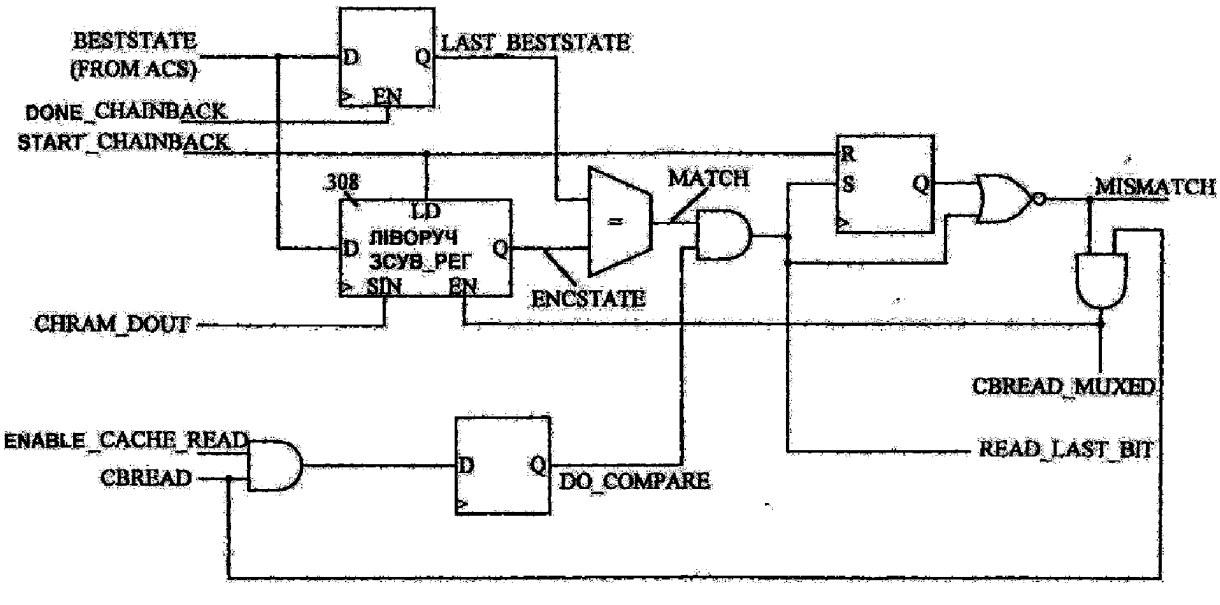


ФІГ.5а

UA 75863 C2

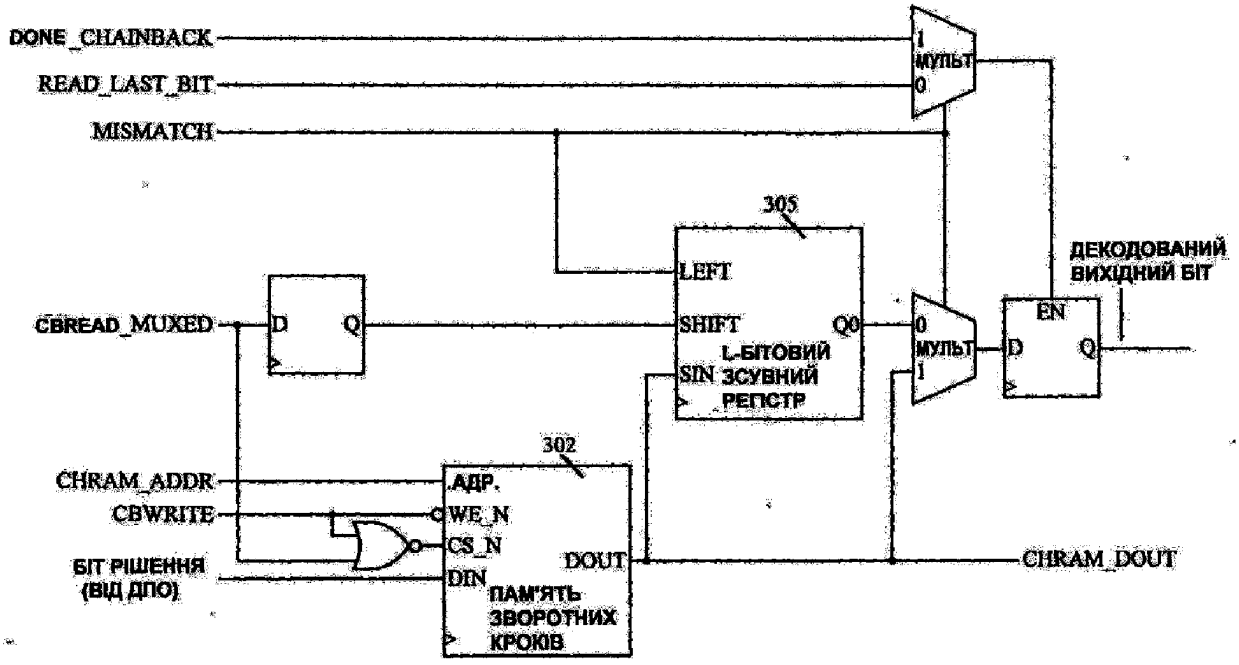


ФІГ.5б

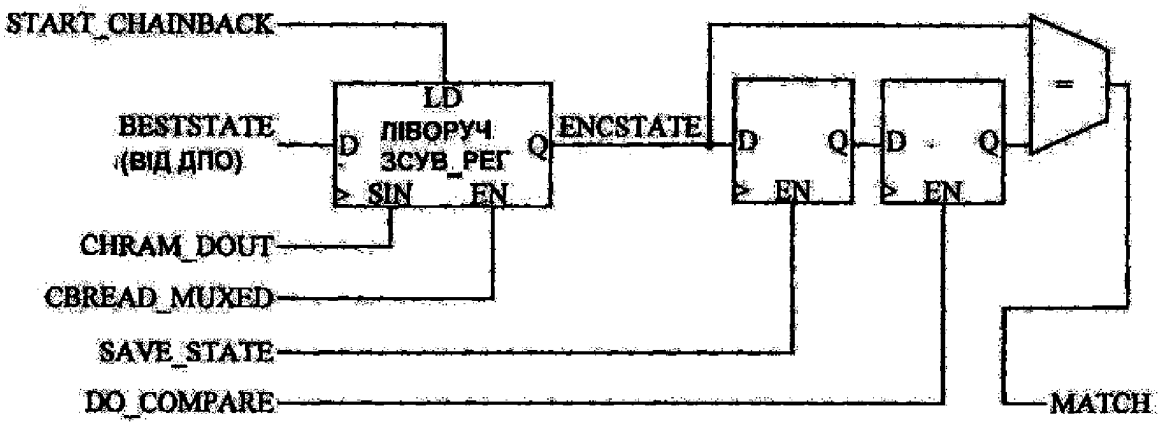


ФІГ.6а

UA 75863 C2



ФІГ.6б



ФІГ.7

Офіційний бюлетень "Промислоава власність". Книга 1 "Винаходи, корисні моделі, топографії інтегральних мікросхем", 2006, N 6, 15.06.2006. Державний департамент інтелектуальної власності Міністерства освіти і науки України.

UA 75863 C2

UA 75863 C2