



УКРАЇНА

(19) UA (11) 90892 (13) C2
(51) МПК (2009)
G06F 9/38

МІНІСТЕРСТВО ОСВІТИ
І НАУКИ УКРАЇНИ

ДЕРЖАВНИЙ ДЕПАРТАМЕНТ
ІНТЕЛЕКТУАЛЬНОЇ
ВЛАСНОСТІ

ОПИС ДО ПАТЕНТУ НА ВИНАХІД

(54) СПОСІБ ТА СИСТЕМА БАГАТОПОТОКОВОГО ПРОЦЕСОРА ІЗ ЗМІННИМ ЧЕРГУВАННЯМ

1

(21) a200711354

(22) 14.03.2006

(24) 10.06.2010

(86) PCT/US2006/009782, 14.03.2006

(31) 11/080,239

(32) 14.03.2005

(33) US

(46) 10.06.2010, Бюл.№ 11, 2010 р.

(72) ДЖАМІЛЬ СУДЖАТ, US, ПЛОНДКЕ ЕРІХ, US,
КОДРЕСКУ ЛУЧІАН, US, АХМЕД МУХАММАД, US,
АНДЕРСОН УІЛЛЬЯМ К., US

(73) КВЕЛКОММ ІНКОРПОРЕЙТЕД, US

(56) WO 00/68780 A2; 16.11.2000

WO 99/21089 A; 29.04.1999

Context-switching techniques for decoupled multithreaded processors. Kreuzinger J; Ungerer T. EUROMICRO Conference, 1999. Proceedings. 25th Milan, Italy 8-10 Sept. 1999, 19990908; 19990908 - 19990910 Los Alamitos, CA, USA, IEEE Comput. Soc, US - ISBN 978-0-7695-0321-9 ; ISBN 0-7695-0321-7. Vol:1, Page(s):248 - 251. XP010352205

(57) 1. Спосіб обробки команд в багатопотоковому процесорі, причому багатопотоковий процесор призначений для обробки множини потоків, працюючих за допомогою множин процесорних конвеєрів, асоційованих з багатопотоковим процесором, причому спосіб полягає в тому, що:

динамічно визначають щонайменше одну запус-
каючу подію, щоб багатопотоковий процесор пе-
ремикався з першого потоку на другий потік, при-
чому згадана запускаяча подія визначається
динамічно, щоб оптимізувати продуктивність бага-
топотокового процесора;

обробляють перший набір команд з першого пото-
ку до виникнення згаданої запускаячої події; і
перемикають багатопотоковий процесор в обробці
з першого потоку на обробку з другого потоку після
виникнення згаданої запускаячої події;

обробляють другий набір команд з другого потоку
після виникнення згаданої запускаячої події;
перемикають багатопотоковий процесор в обробці
з другого потоку на обробку з наступного потоку
після виникнення згаданої запускаячої події;
продовжують етапи обробки і перемикання під час
роботи багатопотокового процесора.

2. Спосіб за п.1, в якому етап динамічного визна-
чення додатково містить етапи, на яких:

2

динамічно визначають щонайменше одну запус-
каючу подію, щоб багатопотоковий процесор пе-
ремикався з першого потоку на другий потік, при-
чому згадана запускаяча подія асоційована з
кількістю циклів процесора першого потоку, кіль-
кість циклів процесора визначається так, щоб оп-
тимізувати продуктивність багатопотокового про-
цесора; і

підраховують кількість циклів процесора для ви-
значення, чи дорівнює згадана підрахована кіль-
кість циклів процесора заздалегідь визначеній кі-
лькості циклів процесора, за допомогою цього
встановлюють наявність згаданої запускаячої по-
дії.

3. Спосіб за п.1, в якому етап динамічного визна-
чення додатково містить етапи, на яких:

динамічно визначають щонайменше одну запус-
каючу подію, щоб багатопотоковий процесор пе-
ремикався з першого потоку на другий потік, зга-
дана запускаяча подія асоціативно зв'язується із
динамічним чином програмованою подією, згадана
динамічним чином програмована подія визначена
для оптимізації продуктивності багатопотокового
процесора; і

контролюють події, виникаючі під час обробки ко-
жного з множини потоків, для визначення, чи вини-
кла згадана динамічним чином програмована по-
дія, внаслідок чого встановлюють наявність
згаданої запускаячої події.

4. Спосіб за п.1, який додатково містить етап, на
якому визначають згадану щонайменше одну за-
пускаючу подію такою, що є промахом кеша, який
має місце під час обробки множини потоків.

5. Спосіб за п.1, який додатково містить етап, на
якому визначають згадану щонайменше одну за-
пускаючу подію такою, що є відсутністю команд,
яка має місце під час обробки множини потоків.

6. Спосіб за п.1, який додатково містить етап, на
якому визначають згадану щонайменше одну за-
пускаючу подію такою, що є сигналом для вико-
нання обробки перемикання по сигналу для пере-
микання із згаданого першого потоку на згаданий
другий потік.

7. Спосіб за п.1, який додатково містить етап, на
якому визначають, що команда спробувала вико-
ристати відсутнє значення із завантаження як зга-
дану щонайменше одну запускаячу подію для
виконання обробки перемикання по використанню

(13) C2

(11) 90892

(19) UA

для перемикання із згаданого першого потоку на згаданий другий потік.

8. Спосіб за п.1, який додатково містить етапи, на яких: динамічно визначають другу запускаючу подію, щоб багатопотоковий процесор перемикався з першого потоку на другий потік, причому згадана друга запускаюча подія визначається динамічним чином, щоб оптимізувати продуктивність багатопотокового процесора;

динамічно керують тим, чи керує виникнення згаданої щонайменше однієї запускаючої події або виникнення згаданої другої запускаючої події перемиканням багатопотокового процесора в обробці з першого потоку на обробку з другого потоку.

9. Багатопотоковий процесор для обробки множини потоків, працюючих за допомогою множини процесорних конвеєрів, асоціативно зв'язаних з багатопотоковим процесором, який містить:

засіб для динамічного визначення щонайменше однієї запускаючої події, щоб багатопотоковий процесор перемикався з першого потоку на другий потік, причому згадана запускаюча подія визначається динамічним чином, щоб оптимізувати продуктивність багатопотокового процесора;

засіб для обробки першого набору команд з першого потоку до виникнення згаданої запускаючої події; і

засіб для перемикання багатопотокового процесора в обробці з першого потоку на обробку з другого потоку після виникнення згаданої запускаючої події;

засіб для обробки другого набору команд з другого потоку до виникнення згаданої запускаючої події;

засіб для перемикання багатопотокового процесора в обробці з другого потоку на обробку з наступного потоку після виникнення згаданої запускаючої події;

засіб для продовження етапів обробки і перемикання під час роботи багатопотокового процесора.

10. Багатопотоковий процесор за п.9, який додатково містить:

засіб для динамічного визначення щонайменше однієї запускаючої події, щоб багатопотоковий процесор перемикався з першого потоку на другий потік, причому згадана запускаюча подія асоціативно зв'язується з кількістю циклів процесора першого потоку, згадана кількість циклів процесора визначається, щоб оптимізувати продуктивність багатопотокового процесора; і

засіб для підрахунку кількості циклів процесора для визначення, чи дорівнює згадана підрахована кількість циклів процесора згаданий кількість циклів процесора, внаслідок цього встановлюючи наявність запускаючої події.

11. Багатопотоковий процесор за п.9, який додатково містить:

засіб для динамічного визначення щонайменше однієї запускаючої події, щоб багатопотоковий процесор перемикався з першого потоку на другий потік, згадана запускаюча подія асоціативно зв'язується із динамічним чином програмованою подією, згадана динамічним чином програмована подія визначена, щоб оптимізувати продуктивність багатопотокового процесора; і

засіб для контролю подій, виникаючих під час обробки кожного з множини потоків для визначення наявності згаданої динамічним чином програмованої події, внаслідок цього встановлюючи наявність згаданої запускаючої події.

12. Багатопотоковий процесор за п.9, який додатково містить засіб для визначення щонайменше однієї запускаючої події такою, що є промахом кеша, який виникає під час обробки множини потоків.

13. Багатопотоковий процесор за п.9, який додатково містить засіб для визначення щонайменше однієї запускаючої події такою, що є відсутністю потрібних команд, яка виникає під час обробки множини потоків.

14. Багатопотоковий процесор за п.9, який додатково містить засіб для визначення щонайменше однієї запускаючої події такою, що є сигналом для виконання обробки перемикання по сигналу для перемикання із згаданого першого потоку на згаданий другий потік.

15. Багатопотоковий процесор за п.9, який додатково містить засіб для визначення того, що команда спробувала використати відсутнє значення із завантаження як згадану щонайменше одну запускаючу подію для виконання обробки перемикання по використанню для перемикання із згаданого першого потоку на згаданий другий потік.

16. Багатопотоковий процесор за п.9, який додатково містить:

засіб для динамічного визначення другої запускаючої події, щоб багатопотоковий процесор перемикався з першого потоку на другий потік, причому згадана друга запускаюча подія визначається динамічним чином, щоб оптимізувати продуктивність багатопотокового процесора; і

засіб для динамічного керування тим, чи керує виникнення згаданої щонайменше однієї запускаючої події або виникнення згаданої другої запускаючої події перемиканням багатопотокового процесора в обробці з першого потоку на обробку з другого потоку.

17. Багатопотоковий процесор для обробки множини потоків, працюючих за допомогою множини процесорних конвеєрів, асоціативно зв'язаних з багатопотоковим процесором, який містить:

чергу команд для організації черги команд в множині потоків, асоціативно зв'язаних із згаданою множиною процесорних конвеєрів;

логіку видачі, яка асоціативно зв'язана із згаданою чергою команд для прийому згаданої множини потоків і містить логіку перемикання потоків для динамічного визначення щонайменше однієї запускаючої події, що змушує багатопотоковий процесор перемикатися з першого потоку на другий потік, причому згадана запускаюча подія визначається динамічним чином, щоб оптимізувати продуктивність багатопотокового процесора;

тракт даних виконання для обробки першого набору команд з першого потоку до виникнення згаданої запускаючої події;

згадана логіка перемикання потоків додатково призначена для перемикання багатопотокового процесора в обробці з першого потоку на обробку

з другого потоку після виникнення згаданої запускової події;

згаданий тракт даних виконання додатково призначений для обробки другого набору команд з другого потоку до виникнення згаданої запускової події;

згадана логіка перемикання потоків додатково призначена для перемикання багатопотокового процесора в обробці з другого потоку на обробку з наступного потоку після виникнення згаданої запускової події; і

згадана черга команд, згадана логіка видачі і згаданий тракт даних виконання додатково асоціативно зв'язані для продовження етапів обробки і перемикання під час роботи багатопотокового процесора.

18. Багатопотоковий процесор за п.17, в якому згадана логіка видачі додатково містить:

логіку оптимізації, асоціативно зв'язану із згаданою логікою перемикання потоків, для динамічного визначення щонайменше однієї запускової події, щоб багатопотоковий процесор перемикався з першого потоку на другий потік, зв'язування згаданої запускової події з кількістю циклів процесора першого потоку, при цьому згадана кількість циклів процесора визначається, щоб оптимізувати продуктивність багатопотокового процесора; і

логіку підрахунку циклів процесора для підрахунку згаданої кількості циклів процесора для визначення, чи дорівнює згадана підрахована кількість циклів процесора згаданій кількості циклів процесора, що внаслідок цього встановлює наявність згаданої запускової події.

19. Багатопотоковий процесор за п.17, в якому згадана логіка видачі додатково містить:

логіку оптимізації, асоціативно зв'язану із згаданою логікою перемикання потоків для динамічного визначення щонайменше однієї запускової події, щоб багатопотоковий процесор перемикався з першого потоку на другий потік, причому згадана запускова подія асоціативно зв'язана динамічно програмованою подією, згадана динамічно програмована подія визначена так, щоб оптимізувати продуктивність багатопотокового процесора; і

логіку контролю для відстеження подій, виникаючих під час обробки кожного з множини потоків, для визначення наявності згаданої динамічно програмованої події, що внаслідок цього встановлює наявність згаданої запускової події.

20. Багатопотоковий процесор за п.17, який додатково містить логіку контролю подій для визначення щонайменше однієї запускової події, що є промахом кеша, який виникає під час обробки множини потоків.

21. Багатопотоковий процесор за п.17, який додатково містить логіку контролю подій для визначення щонайменше однієї запускової події, що є відсутністю потрібних команд, яка виникає під час обробки множини потоків.

22. Багатопотоковий процесор за п.17, який додатково містить логіку контролю подій для визначення щонайменше однієї запускової події, що є сигналом для виконання обробки перемикання по сигналу для перемикання із згаданого першого потоку на згаданий другий потік.

23. Багатопотоковий процесор за п.17, який додатково містить логіку контролю подій для визначення, що команда спробувала використати відсутнє значення із завантаження як згадану щонайменше одну запускову подію для виконання обробки перемикання по використанню для перемикання із згаданого першого потоку на згаданий другий потік.

24. Багатопотоковий процесор за п.17, в якому згадана логіка перемикання потоків додатково містить:

логіку оптимізації для динамічного визначення другої запускової події, щоб багатопотоковий процесор перемикався з першого потоку на другий потік, причому згадана друга запускова подія визначається динамічним чином, щоб оптимізувати продуктивність багатопотокового процесора; і

логіку керування подіями перемикання для динамічного керування тим, чи керує виникнення згаданої щонайменше однієї запускової події або виникнення згаданої другої запускової події перемиканням багатопотокового процесора в обробці з першого потоку на обробку з другого потоку.

25. Машиночитаний носій, який містить записану на ньому програму, для обробки команд в багатопотоковому процесорі, причому багатопотоковий процесор призначений для обробки множини потоків, працюючих за допомогою множини процесорних конвеєрів, асоціативно зв'язаних з багатопотоковим процесором, спосіб містить етапи:

динамічного визначення щонайменше однієї запускової події, щоб багатопотоковий процесор перемикався з першого потоку на другий потік, при цьому згадана запускова подія визначається динамічним чином, щоб оптимізувати продуктивність багатопотокового процесора;

обробки першого набору команд з першого потоку до виникнення згаданої запускової події;

перемикання багатопотокового процесора в обробці з першого потоку на обробку з другого потоку після виникнення згаданої запускової події;

обробки другого набору команд з другого потоку до виникнення згаданої запускової події; і

перемикання багатопотокового процесора в обробці з другого потоку на обробку з наступного потоку після виникнення згаданої запускової події; і

продовження етапів обробки і перемикання під час роботи багатопотокового процесора.

26. Машиночитаний носій за п.25, який додатково містить:

машиночитану програму для динамічного визначення щонайменше однієї запускової події, щоб багатопотоковий процесор перемикався з першого потоку на другий потік, причому згадана запускова подія асоціативно зв'язується з кількістю циклів процесора першого потоку, згадана кількість циклів процесора визначається так, щоб оптимізувати продуктивність багатопотокового процесора; і

машиночитану програму для підрахунку згаданої кількості циклів процесора для визначення, чи дорівнює згадана підрахована кількість циклів процесора згаданій визначеній заздалегідь кількості циклів процесора, що внаслідок цього встановлює наявність згаданої запускової події.

27. Машиночитаний носій за п.25, який додатково містить машиночитану програму для динамічного визначення щонайменше однієї запускаючої події, щоб багатопотоковий процесор перемикався з першого потоку на другий потік, причому згадана запускаюча подія асоціативно зв'язується із динамічним чином програмованою подією, згадана динамічним чином програмована подія визначена так, щоб оптимізувати продуктивність багатопотокового процесора; і моніторингу подій, виникаючих під час обробки кожного з множини потоків для визначення наявності згаданої динамічним чином програмованої події, таким чином встановлюючи наявність згаданої запускаючої події.

28. Машиночитаний носій за п.25, який додатково містить машиночитану програму для динамічного визначення другої запускаючої події, щоб багатопотоковий процесор перемикався з першого потоку на другий потік, причому згадана друга запускаюча подія визначається динамічним чином, щоб оптимізувати продуктивність багатопотокового процесора; і динамічного керування тим, чи керує виникнення згаданої щонайменше однієї запускаючої події або виникнення згаданої другої запускаючої події перемиканням багатопотокового процесора в обробці з першого потоку на обробку з другого потоку.

Розкритий об'єкт винаходу стосується передачі даних. Більш точно, це розкриття стосується новітніх та вдосконалених способу та пристрою для обробки із змінним чергуванням у багатопотоковій процесорній системі.

Сучасні системи зв'язку повинні підтримувати різноманіття прикладних програм. Однією з таких систем зв'язку є система множинного доступу з кодовим розділенням каналів (CDMA), яка підтримує мовний та інформаційний зв'язок між користувачами через наземну лінію зв'язку. Використання технології CDMA в системі зв'язку множинного доступу розкрито в патенті США, №4901307, озаглавленому «SPREAD SPECTRUM MULTIPLE ACCESS COMMUNICATION SYSTEM USING SATELLITE OR TERRESTRIAL REPEATERS» («СИСТЕМА ЗВ'ЯЗКУ МНОЖИННОГО ДОСТУПУ З РОЗШИРЕНИМ СПЕКТРОМ, ЩО ВИКОРИСТОВУЄ СУПУТНИКОВІ АБО НАЗЕМНІ РЕТРАНСЛЯТОРИ»), і патенті США, №5103459, озаглавленому «SYSTEM AND METHOD FOR GENERATING WAVEFORMS IN A CDMA CELLULAR TELEHANDSET SYSTEM» («СИСТЕМА ТА СПОСІБ ДЛЯ ФОРМУВАННЯ КОЛИВАЛЬНИХ СИГНАЛІВ В СТИЛЬНИКОВІЙ ТЕЛЕФОННІЙ СИСТЕМІ CDMA»), права на які передані правонаступнику заявленого об'єкта винаходу.

Система CDMA звичайно побудована так, щоб відповідати одному або більшій кількості стандартів. Одним з таких стандартів першого покоління є «TIA/EIA/IS-95 Terminal-Base Station Compatibility Standard for Dual-Mode Wideband Spread Spectrum Cellular System» («Стандарт TIA/EIA/IS-95 сумісності терміналу та базової станції для двомодової широкосмугової стільникової системи з розширеним спектром»), що надалі указується посиланням як стандарт IS-95. Системи CDMA IS-95 здатні передавати мовну інформацію і пакетні дані. Стандарт більш нового покоління, який може передавати пакетні дані більш ефективно, запропонований консорціумом, названим «Проект партнерства 3^{го} покоління» (3GPP), і втілений в наборі документів, зокрема, номерах документів 3G TS 25.211, 3G TS 25.212, 3G TS 25.213 та 3G TS 25.214, які вже доступні громадськості.

Стандарт 3GPP, надалі вказується посиланням як стандарт W-CDMA (широкосмугового CDMA).

Цифрові сигнальні процесори (ЦСП, DSP) є такими, що часто використовуються в безпроводних телефонних трубках, що підпорядковуються вищенаведеним стандартам. Апаратна багатопотокова обробка стає потенційно корисною технологією в таких ЦСП. Декілька багатопотокових ЦСП були анонсовані промисловістю або вже впроваджені у виробництво в сферах високопродуктивних мікропроцесорів, процесорів аудіовізуальної інформації і мережних процесорів.

Вияв багатопотокової обробки в ЦСП може відбуватися на різних рівнях або при різних ступенях дроблення послідовностей операцій. Наприклад, форма багатопотокової обробки з дрібним розподілом, яку може виконувати ЦСП, використовує паралельно два або більше потоків керування у процесорному конвеєрі. Контексти двох або більше потоків керування часто зберігаються в окремих наборах регістрів всередині мікросхеми. Невживані командні інтервали, які є результатом затримок під час конвеєрного виконання однопотокових програм сучасним мікропроцесором, заповнюються командами інших потоків в багатопотоковому процесорі. Виконавчі блоки мультиплексуються між контекстами потоків, які завантажені в набори регістрів.

Для безпроводної телефонної трубки, що використовує багатопотокові ЦСП, є необхідність зберігати потужність або, більш точно, енергію (тобто, потужність протягом часу). Це відбувається тому, що мультимедійні безпроводні телефонні трубки є і будуть такими, що споживають зростаючі кількості енергії від акумулятора або джерела живлення. Наприклад, безпроводна телефонна трубка, що забезпечує прийом прямої телевізійної трансляції, вимагає, щоб безпроводна телефонна трубка здійснювала безперервне споживання енергії акумулятора, в протилежність періодичному, такому як відбувається при нормальному трафіку двостороннього телефонного виклику. Багатопотокові ЦСП для операцій безпроводної телефонної трубки вирішують цю проблему ефективним використан-

ням джерел живлення за допомогою обробки команд для як можна більших циклів процесора з використанням даної архітектури обробки. Однак, проблеми з існуючими підходами все ще існують.

Важлива проблема для вирішення у багатопотокових ЦСП стосується диспетчеризації потоків, тобто, способу, яким ЦСП визначає, яким чином перемикає обробку між потоками. На жаль, часто відбувається так, що суміші різних прикладних програм можуть бути оптимальними при різних інтервалах перемикання. Наприклад, для ЦСП з N потоками, може бути оптимальним здійснювати перемикання кожний цикл. Для іншого ЦСП з $N/2$ потоками, може бути оптимальним перемикання кожні два цикли. У деяких ситуаціях одна і та сама прикладна програма може бути оптимальною з одним інтервалом перемикання під час однієї частини прикладної програми, і іншим таким під час іншої частини. Тому є необхідність в способі та системі, які вирішують різноманіття проблем використання ресурсів, асоційованих з перемиканням потоків багатопотокової цифрової сигнальної обробки.

Спроби вирішити ці проблеми були безуспішними через традиційних архітектур ЦСП, що встановлюються або укорінені для спеціальної або негнучкої прикладної програми. Наприклад, проблематика прикладної програми з орієнтацією на користувача звичайно намагається більше одержати користі з деяких типів багатопотокових операцій, тоді як прикладні програми для наукових досліджень намагаються більше одержати користі з інших типів багатопотокових операцій. Як результат, різні процесори можуть і були спроектовані для різних прикладних програм, але ті ж самі процесори не оптимальні для обох прикладних програм. На жаль, безпроводні телефонні трубки продовжують вимагати і все більше і більше будуть вимагати, щоб їх ЦСП обробляв орієнтовані на користувача наукові і мультимедійні прикладні програми, а також багато інших типів прикладних програм, для яких єдиний підхід до багатопотокових операцій забезпечує працездатне рішення. Відповідно, існує потреба у багатопотоковому ЦСП безпроводної телефонної трубки, що допускає оптимальні операції при широкому різноманітті прикладних програм.

Розкриті способи обробки із змінним чергуванням за допомогою багатопотокової процесорної системи для поліпшення як роботи процесора, так і ефективного використання енергетичних ресурсів безпроводної телефонної трубки за допомогою забезпечення того, що багатопотоковий процесор обробляє команди протягом максимальної частини свого робочого часу.

Варіант здійснення розкриття пропонує спосіб для обробки команд в багатопотоковому процесорі. Багатопотоковий процесор обробляє множину потоків, працюючих за допомогою множини процесорних конвеєрів, зв'язаних з багатопотоковим процесором. Спосіб включає в себе етапи визначення наперед, щонайменше, однієї запускоючої (ініціюючої) події, щоб багатопотоковий процесор перемикався з першого потоку на другий потік. Запускаюча подія визначається змінним і динаміч-

ним чином, щоб оптимізувати продуктивність багатопотокового процесора. Спосіб та система обробляють перший набір команд з першого потоку до виникнення запускоючої події. Перемикання багатопотокового процесора з обробки першого потоку на обробку другого потоку відбувається по запускоючій події. Обробка другого набору команд з другого потоку продовжується до наступного виникнення запускоючої події. Спосіб та система продовжують етапи обробки і перемикання, доки багатопотоковий процесор не обробить всі набори команд, що вимагають обробки, які обробляються з множини потоків.

Запускаюча подія може бути динамічно визначеною кількістю циклів процесора, кількість яких може бути заздалегідь визначена, щоб оптимізувати продуктивність багатопотокового процесора. У такому випадку, варіант здійснення підраховує кількість циклів процесора, щоб визначати, чи дорівнює підрахована кількість циклів процесора заздалегідь визначеній кількості циклів процесора, внаслідок цього встановлюючи наявність запускоючої події. Як альтернатива, варіант здійснення може встановлювати запускоючу подію як змінну і динамічним чином визначену подію, таку, яка може виникати в блокованому багатопотоковому процесорі. По суті, запускаюча подія може бути відсутністю в кеш-пам'яті потрібних даних або команд. Більше того, розкритий варіант здійснення може комбінувати першу запускоючу подію заздалегідь визначеної кількості циклів процесора з другою запускоючою подією блокуючої події, обидві запускоючі події заздалегідь визначаються змінним і динамічним чином.

Ці та інші переваги розкритого об'єкта винаходу, а також додаткові нові ознаки, будуть очевидні з опису, наведеного в матеріалах даної заявки. Задум цього короткого викладу полягає не в тому, щоб зайняти місце вичерпного опису заявленого об'єкта винаходу, а швидше щоб надати короткий огляд деяких функціональних можливостей об'єкта винаходу. Інші системи, способи, ознаки та переваги, тут передбачені, будуть очевидними фахівцеві в даній галузі техніки при вивченні подальших фігур і докладного опису. Мається на увазі, що всі такі додаткові системи, способи, ознаки та переваги включені в цей опис, знаходяться в межах обсягу прикладеної формули винаходу.

Ознаки, природа та переваги розкритого об'єкта винаходу стануть більш очевидними з докладного опису, викладеного нижче, коли розглядається з посиланнями на креслення, на яких однакові символи посилять відповідно співпадають на всіх кресленнях, і при цьому:

Фіг.1 - спрощена структурна схема системи зв'язку, яка може реалізувати даний варіант здійснення;

Фіг.2 ілюструє архітектуру ЦСП для підтримки суті даного варіанту здійснення;

Фіг.3-6 показують діаграми видачі команд залежно від циклу процесора для відображення деяких аспектів деяких варіантів здійснення заявленого об'єкта винаходу; і

Фіг.7-9 - блок-схеми послідовностей операцій способів, які зображають різні послідовності опе-

рацій обробки, які можуть здійснювати різні варіанти здійснення способу та системи змінного багатопотокового процесора.

Фіг.1 - спрощена структурна схема системи 10 зв'язку, яка може реалізувати дані варіанти здійснення. У блоці 12 передавача дані відправляються, звичайно блоками, з джерела 14 даних в процесор 16 даних (TX) передачі, який форматує, кодує та обробляє дані для формування одного або більше аналогових сигналів. Аналогові сигнали потім видаються в передавач (TMTR) 18, який модулює, фільтрує, посилює і перетворює з підвищенням частоти сигналу основної смуги, щоб сформувати модульований сигнал. Модульований сигнал потім передається через антену 20 в один або більше блоків приймача.

У блоці 22 приймача переданий сигнал приймається антеною 24 і видається в приймач (RCVR) 26. У приймачі 26 прийнятий сигнал посилюється, фільтрується, перетворюється з пониженням частоти, демодулюється та відцифровується, щоб сформувати синфазні (I) та (Q) відліки (вибірки). Вибірки потім декодуються та обробляються процесором 28 даних (RX) прийому, щоб відновити передані дані. Декодування та обробка в блоці 22 приймача виконуються способом, комплементарним по відношенню до кодування та обробки, що виконуються у вузлі 12 передавача. Відновлені дані потім видаються в приймач 30 даних.

Обробка сигналу, описана вище, підтримує передачі мовних, відео, пакетних даних, передачу повідомлень та інші типи зв'язку в одному напрямку. Система двоспрямованого зв'язку підтримує двосторонню передачу даних. Однак, обробка сигналу для іншого напрямку не показана на Фіг.1 для простоти. Системою 10 зв'язку може бути система множинного доступу з кодовим розділенням каналів (CDMA), система зв'язку множинного доступу з часовим розділенням каналів (TDMA) (наприклад, система GSM), система зв'язку множинного доступу з частотним розділенням каналів (FDMA) або інша система множинного доступу, яка підтримує мовний та інформаційний зв'язок між користувачами через наземну лінію зв'язку. У конкретному варіанті здійснення система 10 зв'язку є системою CDMA, яка підпорядковується стандарту W-CDMA.

Фіг.2 ілюструє архітектуру ЦСП 40, яка може служити як процесор 16 даних передачі і процесор 28 даних прийому за Фіг.1. Потрібно зазначити, що ЦСП 40 представляє тільки один варіант здійснення з множини можливих варіантів здійснення цифрового сигнального процесора, які можуть ефективно використовувати ідеї і концепції, представлені тут. У ЦСП 40, тому, потоки з T0 по T5 (посилальні позиції 42-52), містять в собі набори команд з різних потоків. Схема 54 являє собою механізм доступу до команд і використовується для вибірки команд для потоків з T0 по T5. Команди для схеми 54 ставляться в чергу в черзі 56 команд. Команди в черзі 56 команд готові видаватися в процесорний конвеєр 66 (дивіться нижче). З черги 56 команд одиночний потік, наприклад, T0, може вибиратися логічною схемою 58 видачі. Регістровий файл 60 вибраного потоку зчитується, а зчитані дані відправляються в тракти 62 даних виконання

для слота0 по слот3. Слоти зі слот0 по слот3 в цьому прикладі передбачають комбінацію групування пакету, що застосовується в даному варіанті здійснення.

Вихідні дані з трактів 62 даних виконання поступають в схему 64 запису регістрового файлу, також сконфігуровану для вміщення потоків з T0 по T5, для повернення результатів операцій ЦСП 40. Таким чином, тракт даних зі схеми 54 і перед схемою 64 запису регістрового файлу, що розділяється на частини згідно з різними потоками, формує конвеєр 66 обробки.

Даний варіант здійснення може застосовувати гібрид системи мультипроцесору з функціонально різними процесорами (НЕР), яка використовує одиночний мікропроцесор з аж до шести потоків, з T0 по T5. Процесорний конвеєр 66 містить шість рівнів, співпадаючих з мінімальною кількістю циклів процесора, необхідних для вибірки елемента даних зі схеми 54 в регістри 60 та 64. ЦСП 40 одночасно виконує команди різних потоків з T0 по T5 в процесорному конвеєрі 66. Тобто, ЦСП 40 передбачає шість незалежних лічильників команд, механізм внутрішньої розмітки для розрізнення команд потоків з T0 по T5 в межах процесорного конвеєра 66, і механізм, який запускає перемикання потоків. Непродуктивні витрати на перемикання потоків варіюються від нуля до усього лише декількох циклів.

Даний варіант здійснення надає можливість перемикання потоків не тільки з настанням події заздалегідь визначеної кількості циклів синхронізації, але також з виникненням конкретної події, такої як зовнішня подія. Такою зовнішньою подією, наприклад, може бути відсутність в кешу потрібних даних або відсутність потрібних команд. Фактично, система може видавати переривання, таке переривання може використовуватися або оброблятися як зовнішня подія для ініціювання перемикання потоків. Тому, наприклад, при послідовності операцій, яка вимагає значних ресурсів процесора, даний варіант здійснення, наприклад, може надавати доступ до ресурсів процесора на один мільйон циклів синхронізації. Після одного мільйона циклів синхронізації процесор може перемикає потік керування на наступний потік керування. Якщо наступний потік керування вимагає тільки десять тисяч циклів синхронізації, то даний варіант здійснення змушує процесор виділяти потоку тільки необхідні десять тисяч циклів синхронізації.

Фіг.3-6 показують діаграми видачі команд залежно від циклу процесора для відображення деяких аспектів різних варіантів здійснення даного об'єкта винаходу. Зокрема, Фіг.3 представляє схему 70 видачі команд залежно від циклу процесора для операції IMT ЦСП 40.

Фіг.4 показує схему 72, що стосується операції VIIMT за даним варіантом здійснення.

Фіг.5 показує схему 74 для одного з варіантів здійснення операції VSOEMT з ЦСП 40.

Фіг.6 додатково представляє схему 76, щоб показати переваги комбінування обробки VSOEMT з обробкою VIIMT.

На всіх з Фіг.3-5, порожні слоти видачі, такі як порожній слот 78 (Фіг.3), можуть бути визначені як

вертикальні або горизонтальні втрати. Вертикальні втрати 80 з'являються, коли ЦСП 40 не видає в циклі ніяких команд, тобто, має місце очікування видачі команди. Горизонтальні втрати 82 з'являються, коли ЦСП 40 заповнює тільки не порожні піднабори слотів (інтервалів), доступні в даному циклі.

Як показує Фіг.3, IMT виконує перемикання потоку TS за допомогою перемикання потоку, що обробляється, в кожному циклі, незважаючи на те, чи виникає подія тривалої затримки. По суті, ресурси ЦСП 40 чергуються для сукупності готових потоків, з T0 по T5, при ступені дроблення в один цикл.

На Фіг.4, операція VIIMT відрізняється від перемикання IMT за допомогою перемикання з динамічно визначеним інтервалом; тут, три (3) цикли процесора. Зазначимо, що змінні цикли процесора, що встановлюються рівними трьома, все ще можуть мати результатом деякі вертикальні втрати 79.

Фіг.5 зображає цикли процесора залежно від виникнення видачі команд, при цьому запускаючи подія є динамічно визначеною, така як відсутність в кешу потрібних даних (промах кеша) або відсутність потрібних команд. Як можна бачити, цикли обробки між перемиканням потоків змінюються з чотирьох (4) циклів до тільки одного (1) циклу, наприклад, у випадку вертикальних втрат. Тобто, хоча схема може бути подібною традиційній схемі циклу процесора залежно від видачі команд VSOEMT, подія визначається динамічно в даному варіанті здійснення. Все ж, однак, в деяких випадках можуть виникати вертикальні втрати 84.

Як можна бачити на Фіг.6, комбінування VSOEMT та VIIMT істотно знижує як вертикальні втрати, так і горизонтальні втрати. Результат полягає в тому, що ЦСП 40 виконує команди для певною мірою більшої частини своїх робочих циклів.

Обробка VSOEMT за даним варіантом здійснення динамічно вибирає тип події, яка може мати результатом перемикання потоку. Звичайно така ситуація виникає, коли виконання команд досягає операції тривалої затримки або ситуації, де може виникати затримка. Такі події описані нижче, щоб проілюструвати гнучкість даного варіанту здійснення.

Наприклад, обробка VSOEMT може виконувати послідовність операцій перемикання по промаху кеша, яка перемикає потік, якщо завантаження або збереження відсутнє в кеш-пам'яті. У такій обробці тільки ті завантаження, які відсутні в кеш-пам'яті, і ті збереження, які не можуть бути буферизовані, мають тривалі затримки і викликають перемикання потоків. Ця обробка перемикання по сигналу перемикає потік при появі конкретного сигналу, наприклад, сигналізації переривання, внутрішнього переривання або надходження повідомлення. Обробка перемикання по використанню здійснює перемикання, коли команда намагається використати досі відсутнє значення із завантаження (яка, наприклад, була відсутня в кеш-пам'яті).

Інша подія, яка може визначатися динамічно, для якого може відбуватися перемикання, є умов-

ним перемикачем, який зв'язує команду явного перемикання з умовою. При такій обробці (послідовності операцій) потік перемикається, тільки коли задоволена умова, інакше перемикання потоку ігнорується. Команда умовного перемикання може використовуватися, наприклад, після групи команд завантаження/збереження. У такому випадку перемикання потоку ігнорується, якщо всі команди завантаження (в попередній групі) попадають в кеш-пам'ять. Інакше виконується перемикання потоку. Більше того, команда умовного переходу також може додаватися між групами завантажень та їх подальшим використанням, щоб реалізувати уповільнене перемикання потоку, замість реалізації моделі перемикання по використанню.

Фіг.7-9 представляють блок-схеми послідовностей операцій способів, зображаючи різні приклади способу та системи змінного багатопотокового процесора за даним варіантом здійснення. Із посиланням на Фіг.7 послідовність 90 операцій VIIMT може вважатися такою, що починається на етапі 92, коли ініціюються багатопотокові операції ЦСП. На етапі 94 послідовність 90 операцій VIIMT динамічно заздалегідь визначає кількість циклів, при якому ЦСП 40 перемикається з першого потоку на другий потік. Кількість циклів, визначена на етапі 94, може розглядатися як запускаюча подія, яка визначається змінним і динамічним чином, щоб оптимізувати продуктивність багатопотокового процесора. Такими факторами, що враховуються, може бути обсяг ресурсів ЦСП 40, необхідних для виконання набору команд, які містять потік. У той час як мають місце багатопотокові операції, послідовність операцій VIIMT перевіряє за запитом 96, чи була досягнута заздалегідь визначена кількість циклів. Якщо так, то послідовність операцій обробки переходить на етап 98, в момент якого ЦСП 40 перемикається з обробки першого потоку на обробку другого потоку. Слідом за цим послідовність операцій обробки переходить на етап 100 для обробки нового потоку. У послідовності 90 операцій VIIMT послідовність операцій продовжується в зворотному напрямку, до запиту 96, незмінно зв'язуючи кількість циклів процесора. Далі, якщо кількість циклів процесора ще не була задоволена, то послідовність 90 операцій VIIMT продовжується до запиту 102 для перевірки, чи завершені багатопотокові операції. Якщо так, послідовність операцій обробки переходить на етап 104 для завершення багатопотокових операцій. Інакше послідовність операцій обробки переходить на етап 100 для продовження здійснення обробки поточного потоку.

Фіг.8 показує послідовність 120 операцій обробки VSOEMT, яка починається, як і послідовність 90 операцій обробки VIMT, з етапу 92, на якому ЦСП 40 може розглядатися як такий, що ініціює багатопотокові операції. Послідовність операцій обробки потім переходить на етап 122, після чого послідовність 120 операцій обробки динамічно визначає запускаючу подію. Як тільки запускаюча подія була визначена, послідовність операцій обробки продовжується до запиту 124 для перевірки, чи відбулася запускаюча подія. Якщо запус-

каюча подія відбулася, то послідовність операцій обробки продовжується до етапів 98 та 100, відповідно, для перемикання потоку і продовження за допомогою ЦСП 40 обробки потоку. Інакше послідовність операцій обробки продовжується до запиту 102, а в іншому випадку, працює аналогічно послідовності 90 операцій обробки за Фіг.7.

Фіг.9 деталізує послідовність 130 операцій обробки, виходячи з комбінації корисних операцій послідовності 90 операцій обробки VIIMT з послідовністю 120 операцій обробки VSOEMT. Комбінування обох запускаючих подій на етапі 122 з кількістю циклів процесора на етапі 94 ще більше поліпшує багато потокові операції для ЦСП 40.

Розкритий об'єкт винаходу демонструє істотний ступінь гнучкості, коли різні потоки багатопотокового процесора вимагають відмінних обсягів ресурсів процесора. Так, у випадку, якщо набір команд в одному потоку вимагає більшої частки ресурсів процесора, даний варіант здійснення може виділяти ресурси процесора протягом значно більшої кількості часу, ніж кількість, виділена для інших потоків, що вимагають меншого обсягу ресурсів процесора.

Тому, даний варіант здійснення надає багатопотоковий процесор з чергуванням змінних інтервалів, який включає в себе лічильник інтервалів потоку. Лічильник інтервалів потоку містить динамічно визначену кількість циклів, протягом якої кожний потік працює до перемикання на наступний потік. Лічильник інтервалів потоку може оновлюватися або динамічно визначатися програмним забезпеченням, таким як системне програмне забезпечення. Послідовність операцій за таким варіантом здійснення використовує лічильник інтервалів потоку і динамічно визначену кількість циклів, щоб визначити, який потік працює наступним. Цей варіант здійснення вирішує проблему поліпшення продуктивності ЦСП за допомогою динамічної зміни лічильника інтервалів потоку для оптимізації ЦСП відносно даної прикладної програми або суміші прикладних програм. Лічильник інтервалів потоку може динамічно змінюватися під час різних стадій при роботі прикладної програми для досягнення оптимального інтервалу.

Варіант здійснення, що включає в себе спосіб та систему VISOEMT, підводячи підсумок вищесказаному, передбачає змінне на основі подій перемикання в поєднанні з роботою лічильника інтервалів потоку. Таким чином, з динамічно програмованим лічильником перемикання потоку, коли кількість циклів досягає динамічно визначеного значення тайм-ауту перемикання потоку або рахунку робочих циклів, процесор перемикається на наступний потік. Лічильник інтервалів потоку

також може відключатися програмним забезпеченням, в такому випадку процесор стає звичайним процесором SOEMT. Як результат, цей варіант здійснення надає багатопотоковому процесору можливість служити як SOEMT, так і IMT процесор, в бутність різних прикладних програм, яких може потребувати процесор.

Ознаки та функції обробки, описані в матеріалах даної заявки, можуть бути реалізовані різними способами. Наприклад, не тільки ЦСП 40 може виконувати вищеописані операції, але дані варіанти здійснення також можуть бути реалізовані в спеціалізованій інтегральній схемі (ASIC), мікроконтролері, мікропроцесорі або інших електронних схемах, призначених для виконання функцій, описаних в матеріалах даної заявки. Тому, попередній опис переважних варіантів здійснення наданий, щоб дати можливість будь-якому фахівцеві в даній галузі техніки створювати або використовувати заявлений об'єкт винаходу. Різні модифікації відносно цих варіантів здійснення будуть очевидні фахівцям в даній галузі техніки, а загальні принципи, визначені в матеріалах даної заявки, можуть застосовуватися до інших варіантів здійснення без використання винахідливої здатності. Таким чином, заявлений об'єкт винаходу не вважається обмеженим варіантами здійснення, показаними в матеріалах даної заявки, але повинен бути узгодженим, найбільш широким обсягом, що не суперечить принципам та новітнім ознакам, розкритим в матеріалах даної заявки.

Перелік посилальних позицій:

- 14 - Джерело даних
- 16 - Процесор даних передачі
- 18 - TMTR (передавач)
- 20, 24 - антена
- 26 - RCVR (приймач)
- 28 - Процесор даних прийому
- 30 - Приймач даних
- 42-52 - потоки
- 56 - Черга команд
- 58 - Логіка видачі
- 60 - Регістровий файл для читання 32, 32-битних регістри
- 64 - Регістровий файл для запису
- 92 - Ініціювати багатопотокові операції
- 94 - Динамічно визначити рахунок циклів
- 96 - Рахунок циклів задоволений?
- 98 - Перемикнути заданий потік
- 100 - Обробити заданий потік
- 102 - Багатопотокові операції завершуються?
- 104 - Закінчити багатопотокові операції
- 122 - Динамічно визначити запускаючу подію
- 124 - Запускаюча подія
- 82 - горизонтальні втрату.

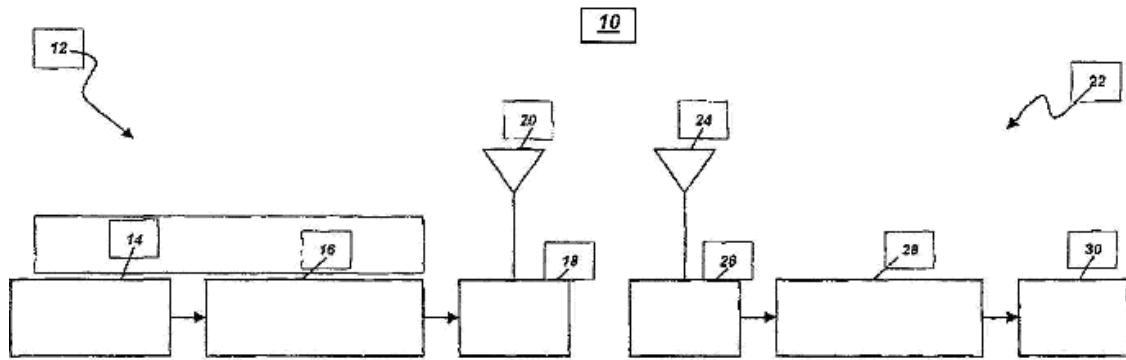


Fig. 1

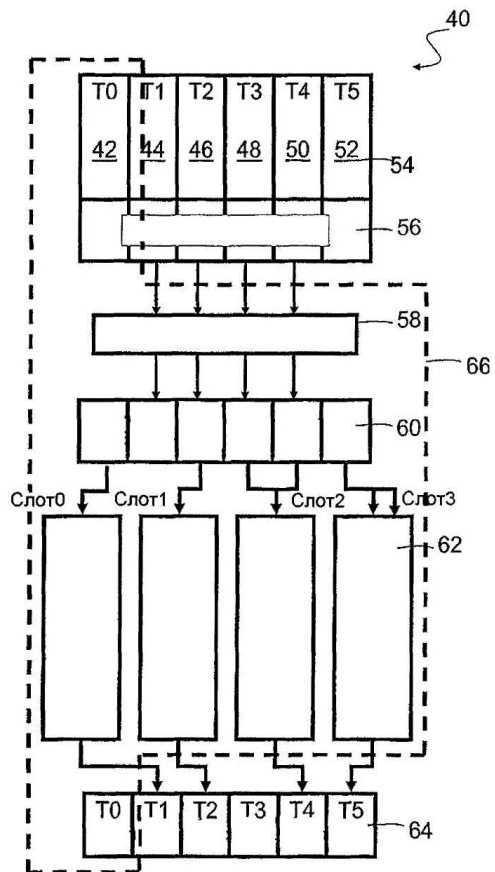


Fig. 2

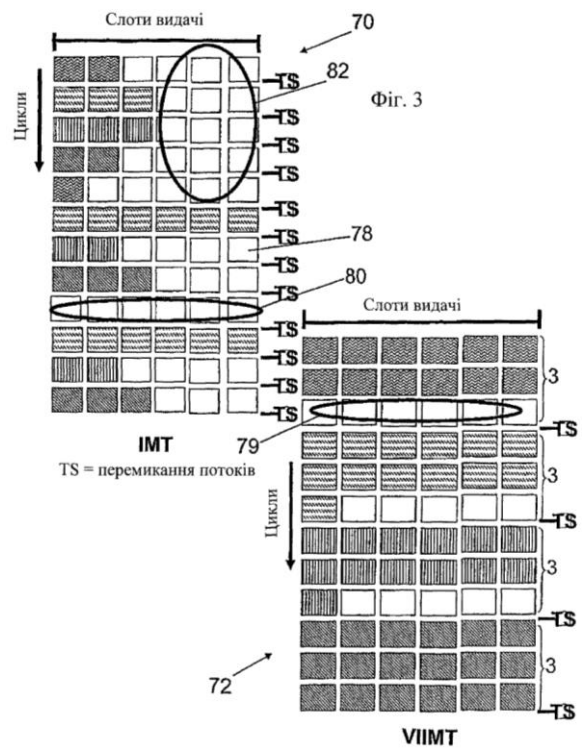
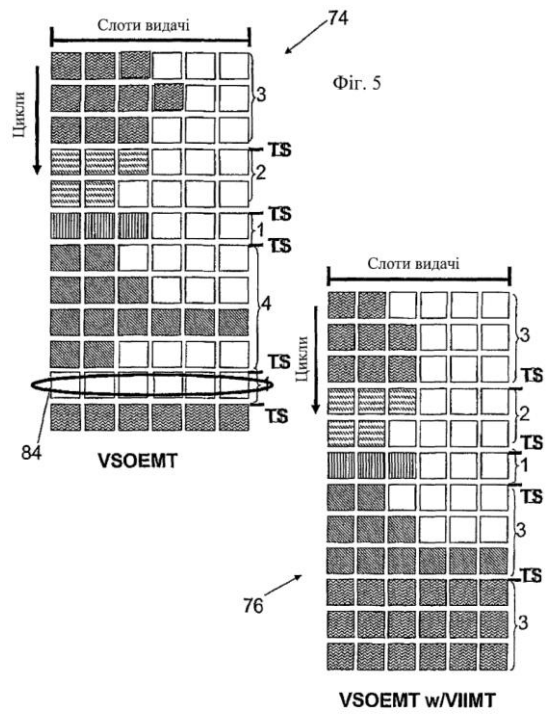
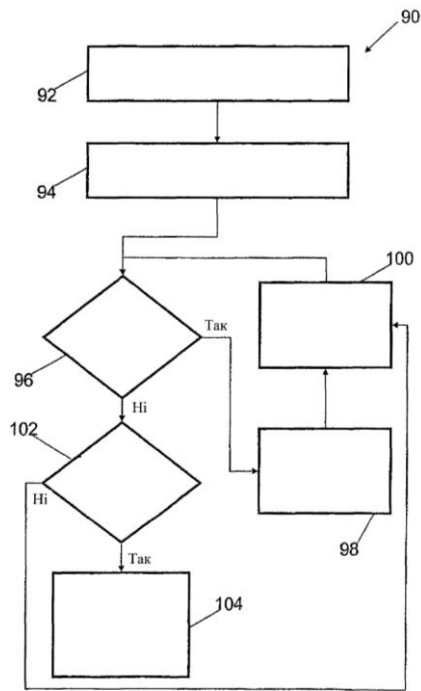


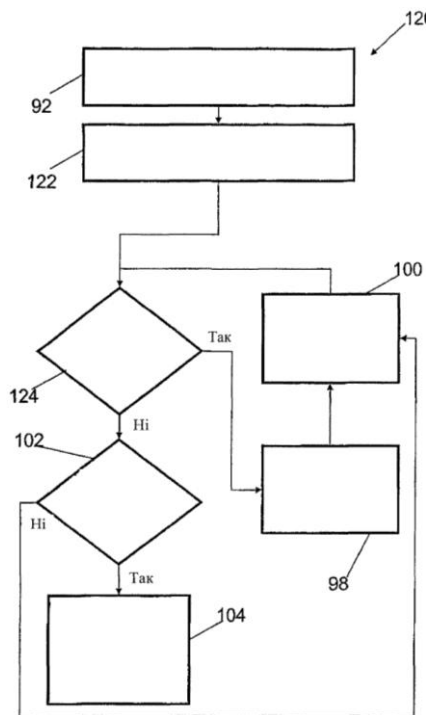
Fig. 4



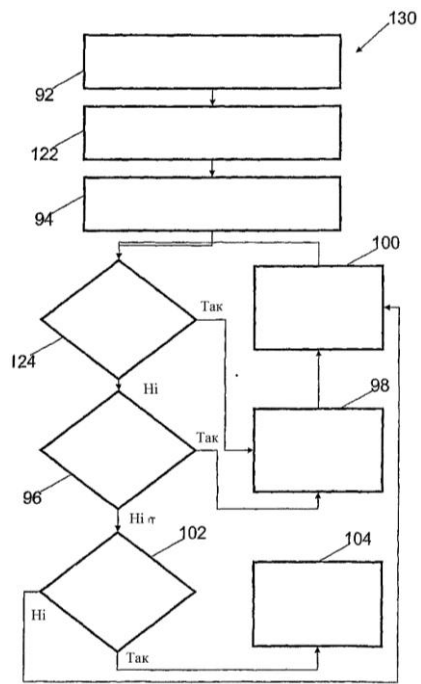
Фиг. 6



Фиг. 7



Фиг. 8



Фиг. 9