МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

КАФЕДРА ЕЛЕКТРОНІКИ І КОМП'ЮТЕРНОЇ ТЕХНІКИ

ПОЯСНЮВАЛЬНА ЗАПИСКА

ДО КВАЛІФІКАЦІЙНОЇ РОБОТИ МАГІСТРА

НА ТЕМУ:

СИСТЕМА AV-ВІДНОВЛЕННЯ ДВІЙКОВОЇ ІНФОРМАЦІЇ

ЗАВІДУВАЧ КАФЕДРИ А.С. ОПАНАСЮК

КЕРІВНИК РОБОТИ І.А. КУЛИК

КОНСУЛЬТАНТ

ПО ЕКОНОМІЧНІЙ ЧАСТИНІ О.М. МАЦЕНКО

РОЗРОБИВ СТУДЕНТ ГР. ЕС.м-91 О.В. ПИСАНИЙ

СУМИ 2020

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Факультет **Електроніки та інформаційних технологій**

Кафедра **Електроніки і комп'ютерної техніки**

Спеціальність 171 Електроніка

Освітня програма Електронні системи та компоненти

ЗАТВЕРДЖУЮ

Зав. кафедрою Опанасюк А.С.

"___"_____ 2020 р..

З А В Д А Н Н Я

на кваліфікаційну роботу магістра студентові

_____

1 Тема проекту (роботи) Система AV-відновлення двійкової інформації _

_____

затверджена наказом по університету "___"_____202__ р. №_____

2 Термін здачі студентом закінченої проекту (роботи) _____

3 Вихідні дані до проекту (роботи) _____

_____

_____

_____

4 Зміст розрахунково-пояснювальної записки (перелік питань, що належить розробити) 1. Огляд літератури та поставлення задачі проектування. 2. Наукова-дослідна частина. 3. Вибір та обґрунтування алгоритму функціонування та структурної схеми системи. 4. Розробка функціональної схеми блоків системи. 5. Вибір елементної бази та розробка принципових електричних схем блоків.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) 1. Схема алгоритму функціонування. 2. Схема електрична структурна. 3. Схема електрична функціональна. 4. Схема електрична принципова

_____

6 Консультанти по проекту (роботі), із зазначенням розділів проекту, що стосуються їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Економічна частина	МАЦЕНКО О.М.		

7 Дата видачі завдання _____

Керівник _____ _____

Завдання прийняв до виконання _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	**Назва етапів** дипломного проекту (роботи)	**Термін виконання** етапів проекту (роботи)	Примітка
1	Огляд літератури та поставлення задачі проектування	04.11.20-11.11.20	
2	Вибір та обґрунтування алгоритму функціонування та структурної схеми системи	12.11.20-16.11.20	
3	Науково-дослідна частина	17.11.20-25.11.20	
4	Розробка функціональної схеми блоків системи	26.11.20-01.12.20	
5	Вибір елементної бази та розробка принципових електричних схем блоків	02.12.20-14.12.20	
6	Економічна частина	15.12.20-18.12.20	

Студент-дипломник _____

Керівник проекту (роботи) _____

"___"_____2019 р.

РЕФЕРАТ

Поясньовальна записка**:** стор.67, 13 рис., 5 табл., 6 додатки, 9 схеми

Об’єкт дослідження — Система AV-відновлення двійкової інформації

Мета роботи — розробка та дослідження роботи AV-відновлення. Розробка функціональної схеми блоків системи. Розробка принципіальної схеми блоків системи.

Результати — розроблено алгоритм та зібрано функціональну систему забезпечення. Розроблено функціональну схему блоків системи. Розроблено принципіальні схеми блоків системи. Проведено економічні розрахунки собівартості системи.

Ключовими словами даного документа стали:

Стиснення, мікропроцесор, мікропроцесорна система, відновлення, блок.

ЗМІСТ

Введення……...2
1. Огляд літератури та постановка завдання проектування……4
1.1 Класифікація та порівняльна оцінка методів сжаття двійкової інформації…….4
1.2 Критерії оцінки методів та алгоритмів стиснення двійкової інформації ……....12
1.3 Способи практичної реалізації методів стиснення двійкової інформації ……..……17
1.4
Висновки до огляду літератури……..……...…22
2.Науково-дослідна частина……..…...23
2.1 Синтез структури системи адресно-векторного стиснення ……23
2.1.1 Структура пристрою пам'яті на основі адресно-векторного коду для підсистем зберігання даних АСУ……23
2.2 Структури пристроїв адресно-векторного кодування для підсистем збору та передачі даних АСУ ……....29
2.3 Опис методу АВ стиснення та відновлення інформації……..33
2.4.Розробка структурної схеми декодування інформації……36
3. Вибір та обгрунтування алгоритму функціонування та структурної схеми...39
3.1.1 Загальні алгоритми адресно-векторного стиснення і відновлення двійковій інформації……..……39
4. Розробка функціональної схеми АВ відновлення двійкової інформації…….…45
4.1 Розробка функціональної схеми системи…….….45
4.1.1 Центральний процесорний блок…….…47
4.1.2 Блок пам’яті…….….47
4.1.2.1 Блок оперативної пам’яті…….….47
4.1.2.2 Блок постійної пам’яті…….….47
4.1.3 Блок перешкод…….….48
4.1.4 Буферний блок……48

4.2 Структурна схема блоків відповідальних за декодуючи та кодуючи частини системи……...49
5. Вибір елементної бази та розробка принципових електричних схем блоків..50
5.1.1 Вибір елементної бази……..……50
5.1. Розробка принципової схеми блоків ПЗП……50
5.2 Розробка принципової схеми пристрою фазування по циклу…….51
5.3 Розробка принципової схеми декодеючої системи……..52
5.4 Розробка принципової схеми системи керування……53
6. Техніко - економічна частина……...55
6.1 Розрахунок собівартості виготовлення системи……..55
Список літератури…….60
Додаток 1..……..62
Додаток 2……63

ВВЕДЕННЯ

Ще вчора здавалося, що диск розміром в один гігабайт - це так багато, що навіть неясно, чим його заповнити, і вже звичайно, кожен про себе думав: був би у мене гігабайт пам'яті, я б перестав «скупитися» і стискати свою інформацію якими -то архіваторами. Але, мабуть, світ так влаштований, що «святе місце порожнім не буває», і як тільки у нас з'являється зайвий гігабайт - тут же знаходиться ніж його заповнити. Та й самі програми, як відомо, стають все більш об'ємними. Так що, мабуть, з терабайтами і екзабайтами буде те ж саме.

Тому, як би не росли обсяги пам'яті диска, упаковувати інформацію, схоже, не перестануть. Навпаки, у міру того як «місця в комп'ютері» стає все більше, число нових архіваторів збільшується, при цьому їх розробники не просто змагаються в зручності інтерфейсів, а в першу чергу прагнуть упакувати інформацію все щільніше і щільніше.

Однак очевидно, що процес цей не нескінченний. Де лежить цю межу, які архіватори доступні сьогодні, за якими параметрами вони конкурують між собою, де знайти свіжий архіватор - ось далеко не повний перелік питань, які висвітлюються в даній статті. Крім розгляду теоретичних питань ми зробили підбірку архиваторов, які можна завантажити з нашого диска, щоб самим переконатися в ефективності тієї чи іншої програми і вибрати з них оптимальну - в залежності від специфіки розв'язуваних вами завдань.

Так як наше інформаційне середовище розвивається в швидкому темпі і інформації становиться все більше і більше, приходить час задуматись як цю інформацію зберегти. І непросто зберегти а в меншому розмірі і так щоб вона не була пошкоджена.

Текст, графіка, відео і т.д. Также, джерела інформації різного виду мають нестаціонарний характер і можуть змінювати свої параметри протягом передачі не тільки інформаційного пакета, але і блоку даних. Безліч методів і алгоритмів стиснення налаштовуються на фіксовані значення або, в кращому випадку, вузькі області зміни параметрів інформаційного джерела (наприклад, ймовірностей появи довічних слів). У разі ж зміни характеристик джерела або виходу їх значень за зумовлені граничні значення такі методи стиснення втрачають свою ефективність

Для цього розглянемо такий процес як стиснення інформації, а також розглянемо декілька із її видів і один із них детальніше.

Стиснення - це процес перекодування інформації, в результаті якого зменшується обсяг файлу.

1. ОГЛЯД ЛІТЕРАТУРИ ТА ПОСТАНОВКА ЗАВДАННЯ ПРОЕКТУВАННЯ

1.1 Класифікація та порівняльна оцінка методів сжаття двійкової інформації

Для початку розглянемо основні методи стиснення та відновлення будь-якої (текст, зображення, файл, тощо) інформації.

1. Стиснення без втрати даних або 'повністю оборотне стиснення' засноване на методі розділення вихідного файлу, на невеликі частини і відновлення цих частин у вихідний файл при розпакуванні архіву. При цьому не відбувається втрата якості даних.

До стиснення без втрати даних відносяться такі види[1.2.3] :

- Кодування довжин серій - проста схема, що дає гарне стиснення даних, які містять багато значень, що повторюються
- LZW - використовується в gif.
- Deflate - використовується в gzip,
- LZMA - використовується в 7-zip.
- Free Lossless Audio Codec - FLAC
- Meridian Lossless Packing - MLP
- Monkey's Audio - Monkey's Audio APE
- OptimFROG
- RealPlayer - RealAudio Lossless
- JPEG-LS
- JPEG 2000 - (в режимі стиснення без втрат)
- LOCO-I
- AV – адресно-векторне стиснення
- Та інші

2. Стиснення з втратою даних.

Стиснення зі втратами зазвичай застосовується для зменшення обсягу звукової, фото- й відеоінформації і, як показує практика, для такого роду інформації це набагато вигідніше, але чим більша втрата даних при стисненні, тим помітніші в стиснених даних стають артефакти.

До стиснення з втратою даних відносяться такі види :

- Зниження глибини кольору
- Метод головних компонент
- Фрактальное стиснення
- Стиснення на основі провісників
- H.263
- H.264
- H.265
- Ogg Theora
- Sorenson video codec
- VC-1
- Та інші

Всі представлені методи стиснення відрізняються способом їх реалізації

Розглянемо по одному з видів

Першим розглянемо метод стиснення з втратами

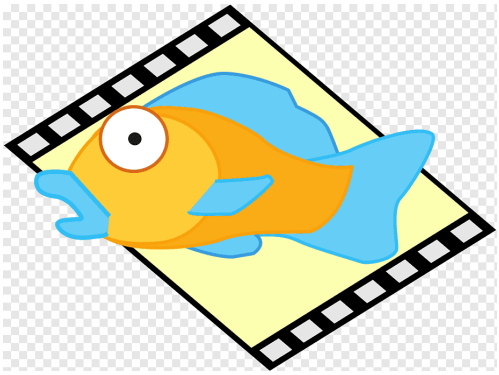
Ogg Theora

Theora[4.5] - вільний відеокодек, розроблений Фондом Xiph.Org як частина їхнього проекту Ogg. Метою цього проекту є інтеграція видеокодека On2 VP3, аудиокодека Vorbis і мультімедіаконтейнера Ogg в одне мультимедійне рішення, на зразок MPEG-4. Є аналогом кодеків MPEG-4 (таких, наприклад, як Xvid, DivX і H.264), RealVideo, Windows Media Video та інших.

Theora є форматом стиснення відео з втратами, заснованим на кодеку On2 VP3. Стислий в цьому форматі відео може бути збережено в будь-якому зручному медіаконтейнера. На 2007 рік для цієї мети найчастіше використовується контейнер Ogg в поєднанні зі звуком у форматі Ogg Vorbis.

Логотип компанії

Рисунок 1.1



Логотип формату стиснення

Рисунок 1.2

На відміну від платних для комерційного використання поширених аналогів (MPEG-4, MP3), комбінація з контейнера Ogg, відео в Theora і звуку в Ogg Vorbis є повністю відкритим, вільним в ліцензійному відношенні мультімедіаформатом

Theora видає кадри не у вигляді звичних для відеокарт RGB-даних, а використовує формат колірного змішування YUV, так як людське око краще розрізняє яскравість, ніж різниця кольорів.

Theora має три масиву для кадру: якщо в RGB дані трьох «по сусідству» байт відповідають за колір одного пікселя, то Theora має окремі три масиву для кожного кадру: чорно-білий, синій і червоний. При використанні формату YUV420 другий і третій кадри мають розмір в чотири рази менше, ніж перший. Наприклад, якщо перший кадр 1280x720, то другий і третій - 640x360..

Встановлений на таких відео-вмістких та відео-відворюючих платформах

-VLC Media Player

-Плагіни Xiph для реального програвача / продюсера;

-Компоненти Quicktime для Quicktime 6 та Quicktime 7 - модулі QuickTime та Macintosh OS X;

-Mplayer;

-SMPlayer;

-CorePlayer;

-Cortado;

-Mozilla Firefox;

-Opera;

-Google Chrome;

Другий роглянений кандидат

З виду методів стиснення без втрат.

А так як нас цікавлять методи без втрат, ми його розглянемо більш детально з різних сторін і різних векторів його розуміння. І так розглянемо один із швидкодійних методів стиснення Адресно-векторне стиснення [6], а так як це метод без втрат той відновлення.

І так як він займає гідне місце серед методів стискающого кодування, за рахунок високої швидкісті перетворення даних і простотою реалізації. А-В кодування полягає у переході від методу векторного до адресного кодування в залежності від числа *k* двійкових одиниць у *n*-розрядному слові.

Такий перехід здійснюється відповідно до системи нерівностей:

$$\begin{cases} 0 \le k < a - 1 \\ n - a + 1 < k \le n \end{cases} \qquad (1.1)$$

Де у нас $a = \frac{n}{\lceil log_2 n \rceil}$

У імовірнісному вигляді дана система нерівностей - умов стиснення - представляється в такий спосіб :

$$\begin{cases} 0 \le p < \frac{1}{\lceil log_2 n \rceil} - \frac{1}{n} + \frac{\gamma}{n} \\ 1 - \frac{1}{\lceil log_2 n \rceil} + \frac{1}{n} - \frac{\gamma}{n} < p \le 1 \end{cases} \qquad (1.2)$$

де p - ймовірність появи двійковій одиниці;

γ - допустиме задане відхилення випадкової величини k одиниць від математичного очікування.

Із отриманої системи випливає, що метод адресно- векторного стиснення також є "чутливим" до зміни параметрів джерела інформації, що генерує стискувані двійкові дані. Вихід значення p за зазначені області знижує коефіцієнт стиснення до одиниці.

Існуючі на сьогоднішній день алгоритми адресно-векторного стиснення є статичними з точки зору граничних значень умов стиснення, що обмежує їх застосування.

Таким чином, актуальною є задача розробки динамічних процедур і на їх основі динамічних алгоритмів адресно-векторного стиснення, які володіли б здатністю підлаштовуватися під змінне джерело інформації.

Як показують наші системи нерівностей одним з реальних способів для адаптації алгоритмів адресно-векторного стиснення є зміна довжини n стискаючих послідовностей.

Також хотілося б підмітити таку деталь як те що алгоритми стиснення без втрат не можуть гарантувати стиснення для усіх видів вхідних даних. Іншими словами, для будь-якого алгоритму стиснення без втрат, існує такий набір вхідних даних, які не зменшуються після обробки алгоритмом, а навпаки — збільшуються. Це було доведено раніше[⇨].

Будь-який алгоритм, що робить деякі файли меншими, повинен робити деякі файли більшими, але не обов'язково, що вони стануть *дуже* великими. Практично використовуються алгоритми, що забезпечують собі механізм «виходу», що зупиняє кодування файлів, які можуть стати більшими після дії стиснення. Теоретично, один лиш додатковий біт потрібен, щоб сказати декодеру, що кодування вимкнене для усіх вхідних даних; проте, більшість кодувальних алгоритмів використовують більше ніж один повний байт для цієї цілі. Наприклад, файли стисненні алгоритмом DEFLATE ніколи не збільшуються більше ніж на 5 байтів на 65 535 байтів вхідних даних.

Фактично, якщо ми розглядаємо усі рівноймовірні (тобто такі, чиє існування можливе з однаковою ймовірністю) файли довжини N, тоді для будь-якого стиснення без втрат, що зменшує розмір якогось файлу, очікуваний розмір стисненого файлу (в середньому серед усіх можливих файлів довжини N) повинен обов'язково бути більше ніж N. Таким чином, якщо ми нічого не знаємо про властивості даних, що збираємось стискати, нам не варто стискати їх взагалі. Алгоритми стиснення без втрат корисні тільки якщо ми швидше за все стискаємо певні види даних ніж інші; тоді алгоритм повинен бути розроблений для ефективного їх стискання.

Отже, головною думкою є не те, що можливо зробити гірше, а те, що не завжди можна отримати непоганий результат. Тоді під вибором алгоритму звичайно розуміється непрямий вибір підмножини з усіх файлів, що стануть корисно меншими. Це теоретична причина для того, що ми маємо мати різні алгоритми для різних видів даних: не існує такого алгоритму, що був би хорошим для будь-якого файлу.

«Трюк», що дозволяє алгоритмам стиснення без втрат (при використанні на даних для яких вони були спроектовані) послідовно стискати файли до меншого розміру, є те, що файли, для яких алгоритми спроектовані діяти,

мають деяку форму легко змодельованої надмірності, яку алгоритм повинен видаляти, таким чином зменшуючи їх розмір внаслідок цієї надмірності. Алгоритми в цілому цілком конкретно налаштовані на конкретний вид файлу: наприклад, програми для стиснення аудіо не працюють на текстах і навпаки.

Зокрема, файли, що складаються з випадкових даних, не можуть бути успішно стиснені ні одним із розумних алгоритмів: дійсно, результат такої дії використовується для визначення концепції випадковості в теорії алгоритмічної складності.

Доведено, що неможливо створити алгоритм, який міг би стискати без втрат будь-які дані. Втім, впродовж років компанії заявляють про досягнення «досконалого стиснення», при якому довільне число N випадкових біт можуть завжди бути стиснені до $N - 1$ біт. Ці заяви можуть бути надійно відкинені навіть без поглиблення у деталі реалізації схеми їх роботи. Ці алгоритми не можуть існувати через суперечність з основними законами математики, бо якщо такий алгоритм існує, він міг би використовуватись циклічно для стиснення даних до нульової довжини. З цієї причини, нібито досконалі алгоритми часто глузливо називають «магічними».

З іншого боку, було доведено що не існує жодного алгоритму визначення можливості стиснення файлу в сенсі колмогорівської складності. Хоча це можливо для будь-яких конкретних даних, навіть якщо вони здаються випадковими. Вони можуть бути істотно стиснені, навіть включаючи розмір декомпресора. Як приклад можна навести цифри числа , що виглядають випадковими, але можуть бути створені дуже маленькою програмою (для пі це пояснюється тим, що його можна уявляти у вигляді нескінченного ряду, що на комп'ютері обчислюється ітеративно). Проте, хоч не може бути визначено, чи конкретний файл нестисливий, проста теорема про нестисливі рядки показує, що більше ніж 99 % файлів будь-якої даної довжини не можуть бути стиснені більше ніж на один байт (включаючи розмір декомпресора).

А так же б хотілося підмітити один із найвідоміших способів стиснення відновлення без втрат – це кодування Хаффмена.

Кодування Хаффмена - один з найбільш відомих методів стиснення даних, який заснований на передумові, що в надлишкової інформації деякі символи використовуються частіше, ніж інші. Як уже згадувалося вище, в російській мові деякі букви зустрічаються з більшою ймовірністю, ніж інші,

однак в ASCII-кодах ми використовуємо для представлення символів однакову кількість бітів. Логічно припустити, що якщо ми будемо використовувати меншу кількість бітів для часто зустрічаються символів і більше для рідко зустрічаються, то ми зможемо скоротити надмірність повідомлення. Кодування Хаффмена якраз і грунтується на зв'язку довжини коду символу з ймовірністю його появи в тексті.

1.2 **Критерії оцінки методів та алгоритмів стиснення двійкової інформації**

Принципи стиснення даних

В основі будь-якого способу стиснення лежить модель джерела даних, або, точніше, модель надмірності. Іншими словами, для стиснення даних використовуються деякі апріорні відомості про те, якого роду дані стискаються. Не володіючи такими відомостями про джерело, неможливо зробити ніяких припущень про перетворення, яке дозволило б зменшити обсяг повідомлення. Модель надмірності може бути статичною, незмінною для всього, що стискається повідомлення, або будуватися або параметризрвані на етапі стиснення (і відновлення). Методи, що дозволяють на основі вхідних даних змінювати модель надмірності інформації, називаються адаптивними. Неадаптивними є зазвичай вузькоспеціалізовані алгоритми, що застосовуються для роботи з даними, що володіють добре певними і незмінними характеристиками. Переважна частина досить універсальних алгоритмів є в тій чи іншій мірі адаптивними.

Є такі види алгоритмов [7]

1. Поточні та словникові алгоритми. До цієї групи належать алгоритми сімейств RLE (run-length encoding), LZ * та ін. Особливістю всіх алгоритмів цієї групи є те, що при кодуванні використовується не інформація про частоти символів в повідомленні, а інформація про послідовності, що зустрічалися раніше.

2. Алгоритми статистичного (ентропійного) стиснення. Ця група алгоритмів стискає інформацію, використовуючи нерівномірність частот, з якими різні символи зустрічаються в повідомленні. До алгоритмів цієї групи відносяться алгоритми арифметичного і префіксного кодування (з використанням дерев Шеннона-Фанно, Хаффмана, січних).

В окрему групу можна виділити алгоритми перетворення інформації. Алгоритми цієї групи не виробляють безпосереднього стиснення інформації, але їх застосування значно спрощує подальше стиснення з використанням поточних, словникових та ентропійних алгоритмів.

Критерії стиснення двійкової інформації

А)Коефіцієнт стиснення[8]

Коефіцієнт стиснення - основна характеристика алгоритму стиснення . Вона визначається як відношення обсягу вихідних незжатих даних до обсягу стислих даних, тобто:

$$k \geq \frac{S_0}{S_0 + 1}$$

Таким чином, чим вище коефіцієнт стиснення, тим алгоритм ефективніше. Слід зазначити:

якщо k = 1, то алгоритм не виробляє стиснення, тобто вихідна повідомлення виявляється за обсягом рівним вхідному;

якщо k <1, то алгоритм породжує повідомлення більшого розміру, ніж нестиснене, тобто, здійснює «шкідливу» роботу.

Ситуація з k <1 цілком можлива при стисненні. Принципово неможливо отримати алгоритм стиснення без втрат, який за будь-яких даних утворював би на виході дані меншою або рівною довжини. Обгрунтування цього факту полягає в тому, що оскільки число різних повідомлень довжиною n біт становить рівно 2n, число різних повідомлень з довжиною меншою або рівною n (при наявності хоча б одного повідомлення меншої довжини) буде не більше 2n. Це означає, що неможливо однозначно зіставити всі вихідні повідомлення стисненим: або деякі вихідні повідомлення не матимуть стисненого уявлення, або декільком вихідним повідомленнями буде відповідати одне і те ж стислий, а значить їх не можна відрізнити. Однак навіть коли алгоритм стиснення збільшує розмір вихідних даних, легко домогтися того, щоб їх обсяг гарантовано не міг збільшитися більш, ніж на 1 біт.

Робиться це в такий спосіб: якщо обсяг стиснутих даних менше обсягу вихідних, повертаємо стислі дані, додавши до них «1», інакше повертаємо вихідні дані, додавши до них «0»).

Коефіцієнт стиснення може бути як постійним (деякі алгоритми стиснення звуку, зображення і т. П., Наприклад A-закон, μ-закон, ADPCM, усічений блочне кодування), так і змінним. У другому випадку він може бути визначений або для кожного конкретного повідомлення, або оцінений за деякими критеріями:

середній (зазвичай по деякому тестовому набору даних);

максимальний (випадок найкращого стиснення);

мінімальний (випадок найгіршого стиснення);

або будь-яким іншим. Коефіцієнт стиснення з втратами при цьому сильно залежить від допустимої похибки стиснення або якості, яке зазвичай виступає як параметр алгоритму. У загальному випадку постійний коефіцієнт стиснення здатні забезпечити тільки методи стиснення даних з втратами.

Б)Допустимість втрат

Основним критерієм відмінності між алгоритмами стиснення є описане вище наявність або відсутність втрат. У загальному випадку алгоритми стиснення без втрат універсальні в тому сенсі, що їх застосування безумовно можливо для даних будь-якого типу, в той час як можливість застосування стиснення з втратами повинна бути обгрунтована. Для деяких типів даних спотворення не припустимі в принципі. В тому числі символічні дані, зміна яких неминуче призводить до зміни їх семантики: програми і їх вихідні тексти, виконавчі масиви і т. п .;

життєво важливі дані, зміни в яких можуть призвести до критичних помилок: наприклад, одержувані з медичної вимірювальної апаратури або контрольних приладів літальних, космічних апаратів і т. п .;

багаторазово піддаються стисненню і відновленню проміжні дані при багатоетапної обробки графічних, звукових і відеоданих.

В)Системні вимоги алгоритмів

Різні алгоритми можуть вимагати різної кількості ресурсів обчислювальної системи, на яких вони реалізовані:

- оперативної пам'яті (під проміжні дані);
- постійної пам'яті (під код програми і константи);
- процесорного часу.

В цілому, ці вимоги залежать від складності і «інтелектуальності» алгоритму. Загальна тенденція така: чим ефективніше і універсальніше алгоритм, тим більші вимоги до обчислювальних ресурсів він пред'являє. Проте, в специфічних випадках прості і компактні алгоритми можуть працювати не гірше складних і універсальних. Системні вимоги визначають їх

споживчі якості: чим менш вимогливий алгоритм, тим на більш простий, а отже, компактною, надійної і дешевої системі він може бути реалізований.

Так як алгоритми стиснення і відновлення працюють в парі, має значення співвідношення системних вимог до них. Нерідко можна, ускладнивши один алгоритм, значно спростити інший. Таким чином, можливі три варіанти:

Алгоритм стиснення вимагає великих обчислювальних ресурсів, ніж алгоритм відновлення.

Це найбільш поширене співвідношення, характерне для випадків, коли одноразово стислі дані будуть використовуватися багаторазово. Як приклад можна привести цифрові аудіо- і відеопрогравачі.

Алгоритми стиснення і відновлення вимагають приблизно рівних обчислювальних ресурсів.

Найбільш прийнятний варіант для ліній зв'язку, коли стиснення і відновлення відбувається одноразово на двох її кінцях (наприклад, в цифрової телефонії).

Алгоритм стиснення істотно менш вимогливий, ніж алгоритм відновлення.

Така ситуація характерна для випадків, коли процедура стиснення реалізується простим, часто портативним, пристроєм, для якого обсяг доступних ресурсів досить критичний, наприклад, космічний апарат або велика розподілена мережа датчиків. Це можуть бути також дані, розпакування яких потрібно в дуже малому відсотку випадків, наприклад запис камер відеоспостереження.

Г)Алгоритми стиснення даних невідомого формату

Є два основні підходи до стиснення даних невідомого формату:

На кожному кроці алгоритму стиснення черговий стискається символ або міститься в вихідний буфер стискає кодера як є (зі спеціальним прапором, позначати, що він не був стиснутий), або група з декількох стискаються символів замінюється посиланням на збігається з нею групу з уже закодованих символів. Оскільки відновлення стислих таким чином даних виконується дуже швидко, такий підхід часто використовується для створення саморозпаковуються програм.

Для кожної сжимаемой послідовності символів одноразово або в кожен момент часу збирається статистика її народження в кодованих даних. На основі

цієї статистики обчислюється ймовірність значення чергового кодованого символу (або послідовності символів). Після цього застосовується той або інший різновид ентропійного кодування, наприклад, арифметичне кодування або кодування Хаффмана, для подання часто зустрічаються послідовностей короткими кодовими словами, а рідко зустрічаються - довшими.

1.3 Способи практичної реалізації методів стиснення двійкової інформації

Сучасні методи стиснення двійкової інформації.

Без перебільшення можна сказати, що відомі тисячі різних методів стиснення даних, однак багато хто з них помітно поступаються іншим по всіх параметрах і тому не представляють інтересу. Решта методи можна розбити на три великі класи.

Перед Вами методи і коротке визначення того як їх реалізовувати[9.10.11].

Алгоритми статистичного моделювання

Найкращі за якістю стиснення алгоритми статистичного моделювання джерел Маркова сімейств PPM (від англ. Prediction by Partial Matching), DMC (від англ. Dynamic Markov Compression), ACB (від англ. Associative Coding by Buyanovski) пророкують імовірність появи наступного символу на основі аналізу частоти появи різних послідовностей символів в раніше закодованої частини повідомлення.

Ці алгоритми мають дуже низькою швидкістю стиснення і вимагають великого обсягу оперативної пам'яті, швидкість декодування практично не відрізняється від швидкості кодування.

Незважаючи на дуже хороші характеристики в сенсі якості стиснення, використовувати алгоритми статистичного моделювання на практиці часто важко або неможливо з огляду на невисоку швидкість стиснення.

Крім того, багато запропонованих способів реалізації методів стиснення статистичним моделюванням для отримання оцінок ймовірностей появи символів використовують команди множення і / або ділення, а іноді і обчислення з плаваючою точкою. Так як такі реалізації пред'являють жорсткі вимоги до апаратного забезпечення, область їх застосування обмежена.

Алгоритми словникового стиснення

Алгоритми словникового стиснення замінюють подстроки кодируемой послідовності символів посиланнями в словник на ідентичні підрядка. З практичної точки зору найкращими представляються алгоритми сімейства LZ (вперше запропоновані Лемпелом і Зивом в 1977р.), Які замінюють початок не

переглянутої частини кодованого повідомлення посиланням на найдовше входження ідентичною підрядка в уже закодованої частини.

Зазвичай для прискорення пошуку співпадаючих подстрок і обмеження обсягу необхідної пам'яті область пошуку обмежується певною кількістю останніх символів закодованої частини: така модифікація LZ77 називається LZ77 зі змінним вікном (LZ77 with sliding window).

Алгоритми сімейства LZ в 1.3-1.7 рази поступаються методам статистичного моделювання за якістю стиснення, однак володіють дуже високою швидкістю кодування при порівняно невеликому обсязі необхідної пам'яті.

Величезна перевага алгоритмів сімейства LZ - надзвичайно висока швидкість декодування. Це дозволяє застосовувати їх в тих випадках, коли декодування здійснюється набагато частіше кодування або швидкість декодування дуже важлива (наприклад, при зберіганні даних на CD-ROM, в файлових системах із стисненням і т. Д.).

Велика частина сучасних промислових систем стиснення даних побудовано на основі різних варіантів алгоритму LZ77, протягом багатьох років заслужено вважалися найкращими за співвідношенням швидкості і якості стиснення.

Алгоритми стиснення сортуванням блоків

Алгоритми стиснення сортуванням блоків сімейства BWT / BS, розроблені в 1994р. Барроуз і Вілером, розбивають кодованих послідовність на блоки символів, представляють (оборотним чином) символи кожного блоку так, що з'являється багато повторень одного і того ж символу, а потім стискають перетворені дані будь-яким досить простим способом.

За якістю стиснення вони наближаються до методів статистичного моделювання (поступаючись їм в 1.2-1.3 рази), а за швидкістю - до алгоритмів сімейства LZ, при меншому в порівнянні з методами статистичного моделювання обсязі необхідної пам'яті; швидкість декодування також досить висока.

Методи ентропійного кодування

Як правило, перераховані вище методи стиснення застосовуються не самостійно, а в поєднанні з будь-яким методом ентропії кодування, що заміняє символи їх кодовими словами - рядками нулів і одиниць - так, що більш часто зустрічається символам відповідають коротші слова.

Такі методи кодування відомі з кінця 40-х рр. і добре вивчені. Їх можна розбити на два великі класи: префіксние (методи Хаффмана, Шеннона, Шеннона-Фано) і арифметичні.

Префіксні коди

Префіксні коди називаються так тому, що жодне кодове слово не є повним початком (т. Е. Префіксом) ніякого іншого слова, що гарантує однозначність декодування.

Відомо багато способів побудови префіксних кодів: коди Шеннона і Шеннона-Фано майже ідеальні, а код Хаффмана - оптимальний серед префіксних кодів.

Так як довжина кожного кодового слова виражається цілим числом бітів, то префіксние коди неефективні на алфавітах малої потужності (2-8 символів) або при наявності символів з дуже великою (понад 30-50%) ймовірністю появи і за якістю стиснення можуть поступатися арифметичним.

Застосування блокових кодів, що кодують не окремі символи, а блоки з k символів, дозволяє побудова кодів, як завгодно близьких за якістю кодування до арифметичним, однак через поліноміальної складності блочного кодування за розміром блоку і ряду інших причин блокового кодування за розміром блоку і ряду інших причин блокове кодування майже не застосовується на практиці .

Як правило, алгоритми словникового стиснення і стиснення сортуванням блоків використовують для кодування виходу основного алгоритму стиснення коди Хаффмана.

Арифметичні коди

Арифметичні коди не ставлять явної відповідності між символами і кодовими словами, вони засновані на інших принципах.

Якість арифметичного кодування краще, ніж у посимвольного префіксного кодування, і близько до теоретичного мінімуму та при малій

потужності алфавіту, і при дуже нерівномірному розподілі ймовірностей появи символів.

З іншого боку, кодування і декодування арифметичних кодів при досить великої потужності кодованого алфавіту помітно повільніше кодування і декодування префіксних кодів, а різниця в якості стиснення зазвичай незначна; з цих та ряду інших причин в більшості випадків префіксне кодування більш переважне для практичного використання.

Арифметичні коди зазвичай застосовуються в поєднанні з методами статистичного моделювання для кодування символів відповідно до передбаченими можливостями.

1.4.Постановка завадання проектування

По результатам огляду різноманітних методів стиснення і відновлення двійкових даних можна зробити наступні висновки:

- AV-метод стиснення і відновлення інформації є перспективним з точки зору простоти реалізації, швидкості стиснення і, що особливо важливо, швидкості відновлення;
- з огляду на можливість застосування AV-відновлення на термінальних пристроях з невеликою обчислювальною здатністю, які є досить поширеними в комп'ютерних мережах, різних інформаційних системах і багатоканальних системах зв'язку, буде проводитится розробка електронної системи AV-відновлення двійкових повідомлень;

Розроблювана система AV-відновлення повинна володіти наступними функціональними можливостями і задовольняти наступними технічним вимогам:

- універсальність системи з точки реалізації алгоритмів AV-відновлення, різних протоколів обміну і способів кодового захисту від помилок;
- практична реалізації фазування по циклу при прийманні AV-двійкових послідовностей;
- захист стиснених даних завадостійким кодом при передачі по каналу зв'язку;
- AV-відновлення двійкових послідовностей з 8, 16, 24 і 32 розрядів;
- наявність паралельного і послідовного каналу введення і виведення;
- наявність оперативної пам'яті для зберігання двійкових послідовностей, які відновлюються, обсягом не менше 2 Кбайта;
- наявність постійної пам'яті для зберігання алгоритмів AV-відновлення у програмному вигляді обсягом не менше 2 Кбайта;
- тактова частота синхронізації не менше 2 МГц.

ВИСНОВКИ ДО ОГЛЯДУ ЛІТЕРАТУРИ

Завдяки даному пошуку інформації було знайдено матеріал для підготовки до розробки структурної схеми даної системи та для подальшої розробки механізму роботи системи. Також було проаналізовано алгоритми інших систем стиснення та відновлення інформації. А так же знайдені книжні словники з елементною базою мікропроцесорів та мікроконтролерів.

2.НАУКОВА-ДОСЛІДНА ЧАСТИНА

2.1 Синтез структури системи адресно-векторного стиснення

2.1.1 Структура пристрою пам'яті на основі адресно-векторного коду для підсистем зберігання даних АСУ

Продуктивність АСУ в значній мірі визначаються характеристиками знаходиться в її складі підсистеми зберігання даних. Однією з найважливіших характеристик зазначеної підсистеми є ємність запам'ятовуючих пристроїв [12, 13].

У даній роботі розглянемо пам'ять з довільним доступом як найбільш дорогий і часто зустрічається в структурі обчислювальних систем, банків і архівів даних АСУ. У разі розробки постійного пам'яті адресно-векторне стиснення уможливлює іншу структуру пристрою зберігання, розглянуту в роботі [14].

Пропонована структура постійного пам'яті, яка приведена на рис.2.1, дозволяє, з одного боку, виключити пропорційну залежність між складністю інформаційного масиву (іншими словами, ймовірністю p появи логічних одиниць) і обсягом апаратних витрат [15], а з іншого боку, частково знизити надмірність збережених даних [12, 17] . Це досягається шляхом включення в структуру пристрою поряд з блоком векторної пам'яті блоку адресному пам'яті, введення додаткового-котельної адресації за ознакою методу кодування і використання ознаки логічного рівня адрес. В результаті цього загальна ємність пам'яті пристрою, необхідна для зберігання масиву даних, зменшується [14].

У пристрій постійної пам'яті заносяться виконавчі комбінації, для яких заздалегідь відомо число k логічних одиниць. Згідно нерівністю закодуєм запам’ятовуючий інформаційний масив адресно-векторного методом.

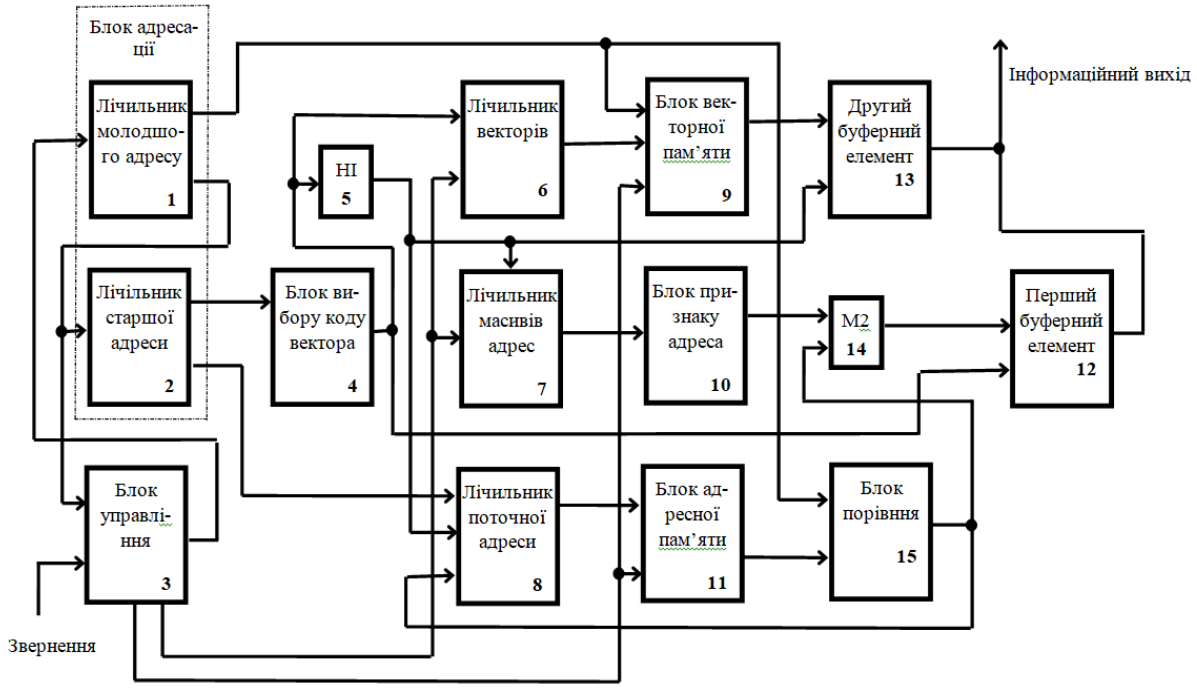


Рисунок 2.1 - Структурна схема запам'ятовуючого пристрою

У разі k, що не задовольняє , відповідний вектор залишається без змін і заноситься в блок векторної пам'яті. В іншому випадку - вектор перетворюється через виконавчі адреси логічних одиниць (якщо $k \leq n / 2$) або нулів (якщо $k > n / 2$) і заноситься в блок адресному пам'яті. Так як вектори і послідовності адрес розміщуються в різних блоках, то виникає необхідність їх додаткової адресації бітом-ознакою методу кодування (просто ознака методу). Крім цього, для розрізнення послідовностей з адресами одиниць від послідовностей з адресами нулів необхідно ввести ще один біт - біт ознаки логічного рівня адреси (просто, ознака рівня). Кожен з цих бітів формується окремими блоками, відповідно блоком ознаки методу і блоком ознаки рівня.

Процедура зчитування з використанням значень ознак відбувається наступним чином.

Залежно від значення ознаки методу адресується і підключається до виходу запо-Міна пристрою блок векторної або блок адресному пам'яті. Висновок довічних даних здійснюється повекторно. При виведенні вектора з блоку векторної пам'яті його вид залишається без змін. У процесі виведення вектора з блоку адресному пам'яті його адреси послідовно порівнюються в блоці порівняння з поточним значенням розгортає лічильника. В якості такого лічильника використовується лічильник молодшого адреси. При цьому

значення біта, що виводиться на вихід пристрою, визначається елементом суми по модулю два і залежить від результату порівняння та ознаки рівня.

Вихідна ємність I пам'яті, необхідна для зберігання масиву даних з m векторів довжини n, визначається як

$$I = mn. \quad (2.1)$$

Вважаючи, що відповідно до умови r векторів масиву кодуються адресним методом, а l - залишаються без зміни, де m = r + l, то результуюча ємність I_{av} пристрої з урахуванням службових біт ознак методу і рівня:

$$I_{av} = \lceil \log_2 n \rceil \sum_{i=1}^{r} k_i + l \cdot n + m + r \ . \quad (2.2)$$

Таким чином, використовуючи (2.1, 2.2), число вивільнених осередків пам'яті одно

$$R = I - I_{av} = m \cdot n - \left(\lceil \log_2 n \rceil \sum_{i=1}^{r} k_i + l \cdot n + m + r \right), \quad (2.3)$$

а коефіцієнт стиснення масиву -

$$K_{sh} = \frac{I}{I_{av}} = \frac{m \cdot n}{\lceil \log_2 n \rceil \sum_{i=1}^{r} k_i + l \cdot n + m + r} \ . \quad (2.4)$$

Очевидно з (2.3, 2.4), що зменшення числа R необхідних осередків і збільшення коефіцієнтом-та K_{sh} стиснення досягається при збільшенні числа r векторів, що задовольняють умові, і зменшенні загальної кількості $k_\Sigma = k_1 + \ldots + k_i + \ldots + k_r$ містяться в них одиниць.

Оперативна пам'ять на рис. 5.1 складається з лічильника 1 молодшого адреси і лічильника 2 старшого адреси, що утворюють блок адресації, блоку 3 управління, блоку 4 вибору коду вектора, елемента 5 НЕ, лічильника 7 масивів адрес, лічильника 8 поточного адреси, блоку 9 векторної пам'яті, блоку 10 ознаки адрес, блоку 11 адресному пам'яті, першого 12 та другого 13 буферних елементів, елемента 14 суми по модулю два, блоку 15 порівняння.

Пристрій працює наступним чином.

За сигналом звернення до пристрою блок 3 керування починає генерувати послідовність тактових імпульсів, яка запускає лічильник 1 молодшого адреси та 2 старшого адреси. Лічильник 1 молодшого адреси формує двійковий номер виведеного поточного розряду вектора, а лічильник 2 старшої адреси - двійковий номер самого вектора. Згідно номеру вектора блок 4 вибору коду вектора виставляє ознака методу кодування (логічний нуль або одиницю).

Залежно від рівня ознаки код вектора послідовно вибирається з блоку 9 векторної пам'яті або з блоку 11 адресному пам'яті.

При підключенні блоку 9 векторної пам'яті сигнал з блоку вибору коду вектора 4 розблокує по синхровході лічильник 6 векторів і замикає перший буферний елемент 12, а через елемент 5 "НЕ" відкриває другий буферний елемент 13, блокує лічильник 7 масивів адрес і лічильник 8 поточного адреси. Через час затримки по сигналу зміни адреси, що надходить з блоку 3 управління, лічильник 6 векторів змінює свій стан і виставляє адресу обраного вектора. Лічильник 6 векторів, кількість станів якого дорівнює числу векторів, закодованих векторних методом, і лічильник 1 молодшого адреси організовують адресацію блоку 9 векторної пам'яті. За сигналами дозволу відповідно до адресами лічильника 1 молодшого адреси і лічильника 6 векторів на виході блоку 9 векторної пам'яті з'являється послідовність нулів і одиниць, яка через відкритий другий буферний елемент 13 виводиться на інформаційний вихід пристрою.

При підключенні блоку 11 адресному пам'яті сигналом вже іншого рівня з блоку 4 вибору коду, навпаки, блокується лічильник 6 векторів і відкривається перший буферний елемент 12. І тим же сигналом, але вже через елемент 5 "НІ", розблоковуються по синхровходу лічильник 7 масивів адрес і лічильник 8 поточного адреси і закривається другий буферний елемент 13. При цьому за сигналом зміни адреси змінює свій стан вже лічильник 7 масивів адрес. Лічильник 7 масивів адрес формує двійковий номер вектора, закодованого адресним методом. Відповідно до номером вектора блок 10 ознаки адрес виробляє ознака закодованого рівня обраного вектора. Адресацію блоку 11 адресному пам'яті здійснює лічильник 8 поточного адреси, ємність якого дорівнює загальному числу адрес. За сигналом дозволу згідно стану лічильника 8 поточного адреси на виході блоку 11 адресному пам'яті з'являється номер, який порівнюється блоком 15 порівняння з номером поточного виведеного розряду з лічильника 1 молодшого адреси. Значення біта, що виводиться на вихід через відкритий перший буферний елемент 12, залежить від результату порівняння та значення ознаки рівня. Ця залежність відповідає функції, яка реалізується елементом 14 суми по модулю два. У разі збігу номерів блок 15 порівняння додає одиницю в лічильник 8 поточного адреси, який адресує наступну адресу вектора з блоку 11 адресному пам'яті. При розбіжності номерів лічильник 8 поточного адреси зберігає свій попередній стан.

Сигнал зміни адреси з блоку 3 управління надходить із затримкою, яка враховує час перехідних процесів, що відбуваються в блоці 4 вибору коду вектора, елементі 5 НЕ і лічильника 2 старшого адреси. Сигнал дозволу з блоку 3 управління синхронізує висновок послідовності і виключає перехідні процеси в лічильнику 7 масивів адрес, лічильнику 6 векторів і блоці 10 ознаки адрес. Зв'язки, що враховують затримку поширення сигналів в інших блоках, є другорядними і на структурній схемі (рис. 2.1) не наведено.

Після закінчення розгортки вектора лічильник 1 молодшого адреси встановлюється в початковий стан. Після закінчення виведення всього довічного масиву разом з лічильником 1 молодшого адреси в початковий стан встановлюється і лічильник 2 старшого адреси, лічильник 6 векторів, лічильник 7 масивів адрес і лічильник 8 поточного адреси.

Розглянута структура постійної пам'яті може знайти широке застосування в системах промислового телебачення, технічного зору, факсимільного зв'язку, наприклад, для зберігання образів досліджуваних об'єктів, моделей і шаблонів, а також масивів постійних величин [17, 19]. Крім того, вона придатна і для використання в системах зовнішньої пам'яті - накопичувачах на магнітних стрічках і дисках, що відтворюють пристроях оптичних дисків і т. Д.

На закінчення, розглянемо приклад кодування масиву інформації, що складається з $m = 10$ векторів розрядності $n = 16$ і заносимого в пристрій пам'яті (табл. 2.1).

Таблиця 2.1 - Приклад кодування масиву двійковій інформації для зберігання в пристрої

Вихідний масив інформації	Зберігаємий в системі масив інформації	Ознака метода	Ознака рівня
1. 0000100010000000	10110110	1	1
2. 1101101010000111	1101101010000111	0	-
3. 0000000000000001	0000	1	1
4. 1111101011101 01	1111101011101 01	0	-
5. 1101101111111111	11011010	1	0
6. 0000000000000000	-	1	1
7. 0010100010000000	0010100010000000	0	-
8. 0000000001000 10	01010001	1	1
9. 1111111111111111	-	1	0
10. 0111111111111111	1111	1	0

Як випливає з табл. 2.1 число перетворених векторів дорівнює $r = 7$. Це вектори 1, 3, 8, перетворені через адреси одиниць, вектори 5, 10, перетворені через адреси нулів, і вектори 6 і 9, замість яких в пам'ять заносяться тільки службові ознаки. Решта виконавчі комбінації - 2, 4, 7 - залишаються без змін і їх число $l = 3$. Необхідну кількість осередків пам'яті для блоків адресною і векторної пам'яті відповідно рівні:

$$I_a = (k_1 + k_3 + (n - k_5) + k_6 + k_8 + (n - k_9) + (n - k_{10}))\lceil \log_2 n \rceil = \\ = (2 + 1 + 2 + 0 + 2 + 0 + 1) * 4 = 32, \qquad (2.5)$$

$$I_v = l * n = 3 * 16 = 48.$$

Число ознак, що визначають метод кодування, дорівнює числу m всіх векторів, а число ознак рівня дорівнює r. Звідси, блок 4 вибору коду вектора повинен мати *10* станів(або число осередків пам'яті), а блок 10 ознаки адрес - *7*. Ємності лічильників 1 і 2 молодшого і старшого адреси відповідно визначаються як $\lceil \log_2 n \rceil = \lceil \log_2 16 \rceil = 4$ і $\lceil \log_2 n \rceil = \lceil \log_2 10 \rceil = 4$. Ємності лічильників 6 і 7 векторів і масивів адрес визначаються числом $l = 3$ довічних векторів, які залишилися без зміни: $\lceil \log_2 l \rceil = \lceil \log_2 3 \rceil = 2$, і числом $r = 7$ перетворених векторів: $\lceil \log_2 r \rceil = \lceil \log_2 7 \rceil = 3$ відповідно. Лічильник 8 поточної адреси повинен бути налаштований на підрахунок всіх адрес і число його станів -

$$\lceil \log_2 (k_1 + k_3 + (n - k_5) + k_6 + k_8 + (n - k_9) + (n - k_{10})) \rceil = \qquad (2.6)$$
$$= \lceil \log_2 (2 + 1 + 2 + 0 + 2 + 0 + 1) \rceil = 3.$$

В результаті використання запропонованої структури пристрою, що запам'ятовує для зберігання вищенаведеного масиву двійковій інформації число R зекономлених осередків пам'яті становить:

$$R = 10 \cdot 16 - (\lceil \log_2 16 \rceil (2 + 1 + 2 + 0 + 2 + 0 + 1) + 3 \cdot 16 + + 10 + 7) = 160 - 97 = 63, \qquad (2.7)$$

а коефіцієнт K_{sh} стиснення масиву - $K_{sh} = 160 / 97 \approx 1{,}65$.

Ефективність такого побудови пристрою, що запам'ятовує відчутно зростає при великої розрядності n двійкових комбінацій і великій кількості векторів, що кодуються адресним методом, серед їх загального числа.

2.2. Структури пристроїв адресно-векторного кодування для підсистем збору та передачі даних АСУ

Адресно-векторний кодуючий (АВК) пристрій розташовується між джерелом інформації і каналом зв'язку. Так як в роботі синтезовані два принципово відмінних кодуючих алгоритма АВК-1 і АВК-2, то і влаштування їх реалізують будуть також мати різні структури.

На рис. 2.2 побудовано пристрій (пристрій кодування) ПК-1 адресно-векторного кодування, функціонує за алгоритмом АВК-1 і використовує постійну пам'ять, ємність якої визначається згідно з формулою .

У структуру ПК-1 входять наступні блоки: блок 1 вхідних регістрів, блок 2 обчислення числа логічних одиниць, блок 3 пам'яті, блок 4 управління, блок 5 порівняння, комутатор 6 і блок 7 вихідних регістрів. Блок 1 призначений для логічної розв'язки пристрої кодування з джерелом інформації. Блок 2 обчислює число двійкових одиниць по одному з алгоритмів 2 або 3 і формує службове слово Bin k. Блок 3 зберігає адресні послідовності - образи відповідних вихідних двійкових слів ai, числа k яких задовольняють нерівності. Блок 4 управляє і синхронізує роботу всіх блоків пристрою. Блок 5 призначений для порівняння значень верхнього і нижнього умов стиснення з числом Bin k і прийняття рішення про використаний метод кодування. Блок 6 комутує в залежності від результату порівняння виходи блоку 1 вхідних регістрів і блоку 3 пам'яті. Блок 7 приймає і зберігає результуючий адресно-векторний код $f_{av}\,(a_i)$.

Принцип роботи пристрою ПК-1 полягає в наступному.

На інформаційну шину кодує пристрої виставляється вихідне двійкове слово a_i, яке за сигналом "Звернення" заноситься в блок 1 вхідних регістрів. За цим сигналом блок 4 управління виробляє сигнал "Підрахунок" для запуску блоку 2, який визначає число Bin k одиниць в кодованому слові. Після цього в блоці 5 порівняння це число порівнюється з верхнім і нижнім умовами адресно-векторного стиснення. У разі виконання нерівностей блок 5 виробляє сигнал для комутатора 6, котрий перемикає вихід блоку 3 пам'яті на вхід блоку 7 вихідних регістрів. На виході блоку 3 в цей момент знаходиться адресна послідовність VU (ai) або VZ (ai), відповідна розташуванню одиниць або нулів вихідного двійкового слова. Таким чином, на інформаційному вході блоку 7 присутні число Bin k з блоку 2 і адресна комбінація з блоку 3, складові адресно-

векторне слово $f_{av}\,(a_i)$.. За керуючому сигналу "Запис" з блоку 4 вони паралельно записуються в блок 7 вихідних регістрів. Далі послідовністю сигналів "Зрушення" отримане адресно-векторне слово розряд за розрядом переміщається в канал зв'язку. У разі невиконання умов комутатор 6 по сигналу блоку 5 порівняння підключає вихід блоку 1 до входу блоку 7. Тоді на інформаційних входах блоку 7 регістрів присутні Bin k і вихідне слово a_i. Їх запис і зрушення в канал зв'язку відбувається аналогічним чином. По закінченню зсуву $f_{av}\,(a_i)$. в канал зв'язку блок 4 формує сигнал "Готовність", що означає готовність до кодування і передачі наступного слова.

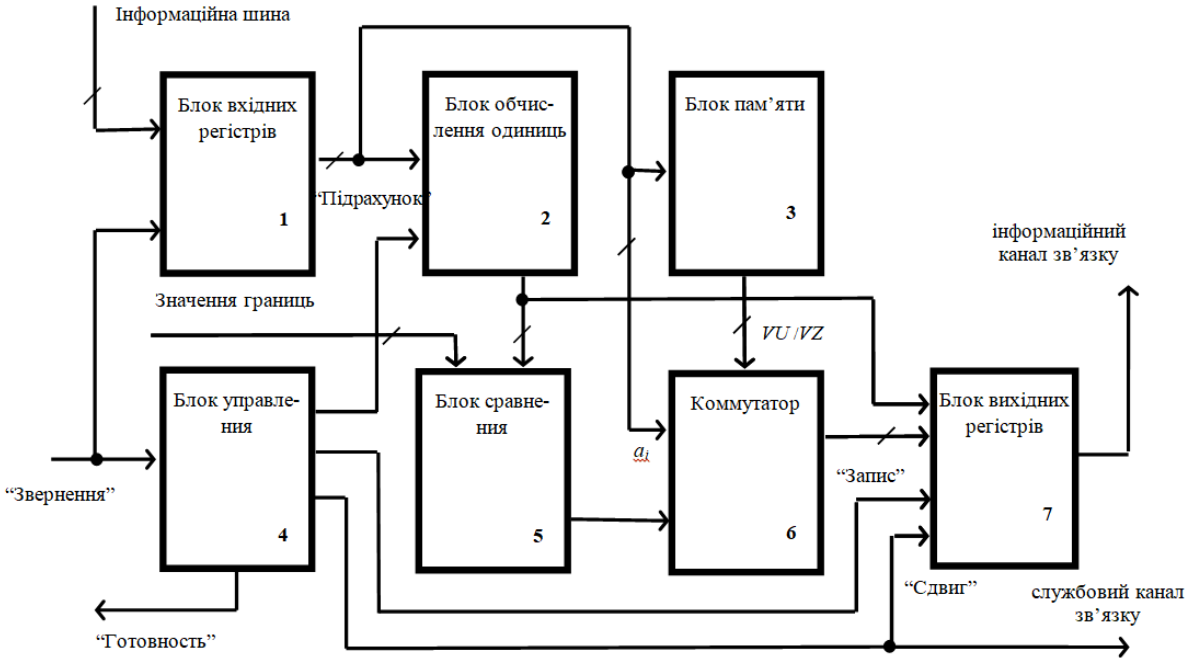


Рисунок. 2.2 - Структурна схема адресно-векторного кодування пристрою ПК-1

У разі кодування двійкових слів великої довжини доцільно проводити формування адресних послідовностей алгоритмічно. З цією метою для роботи кодує пристрої застосовується алгоритм АВК-2.

На рис. 2.3 побудовано пристрій ПК-2 адресно-векторного кодування, функціонує за алгоритмом АВК-2 і використовує елемент 2 "І", блок 3 лічильників, блок 5 дешифраторів, реєстрову пам'ять 8, блок 9 формувачів адреси для формування адресних послідовностей.

У структуру ПК-2 входять наступні блоки: блок 1 вхідних регістрів, блок 2 вибору, блок 3 лічильників, блок 4 підрахунку одиниць, блок 5 дешифраторів, блок 6 управління та синхронізації, блок 7 порівняння, реєстрова пам'ять 8, блок 9 формувачів адреси, комутатор 10, блок 11 вихідних регістрів. Блок 1 призначений для логічної розв'язки джерела інформації та кодує пристрої,

паралельної його видачі для підрахунку числа одиниць і послідовного порозрядного зсуву для формування адрес одиниць (або нулів). Блок 2 вибору в залежності від числа одиниць пропускає на блок 3 лічильників виконавчі одиниці або нулі. Блок 3 лічильників призначений для відстеження числа проглянутих одиниць (нулів). Блок 4 підрахунку одиниць апаратно реалізує алгоритми 2 і 3 і обчислює значення k для кодованого слова. Блок 5 дешифраторів адресує окремий регістр реєстрової пам'яті 8 для запису в нього сформованого адреса. Блок 6 управляє роботою всіх блоків пристрою і синхронізує їх взаємодія. Блок 7 призначений для порівняння числа Bin k з умовами стиснення. Реєстрова пам'ять 8 для запису і зберігання адресному послідовності, що складається з адрес логічних одиниць або нулів. Блок 9 формує адресу двійковій одиниці або нуля. Комутатор 10 організовує перемикання виходів блоку 1 і реєстрової пам'яті 8 на входи блоку 11 вихідних регістрів. Блок 11 вихідних регістрів записує, зберігає і зрушує адресно-векторне слово в канал зв'язку.

Принцип роботи пристрою ПК-2 полягає в наступному.

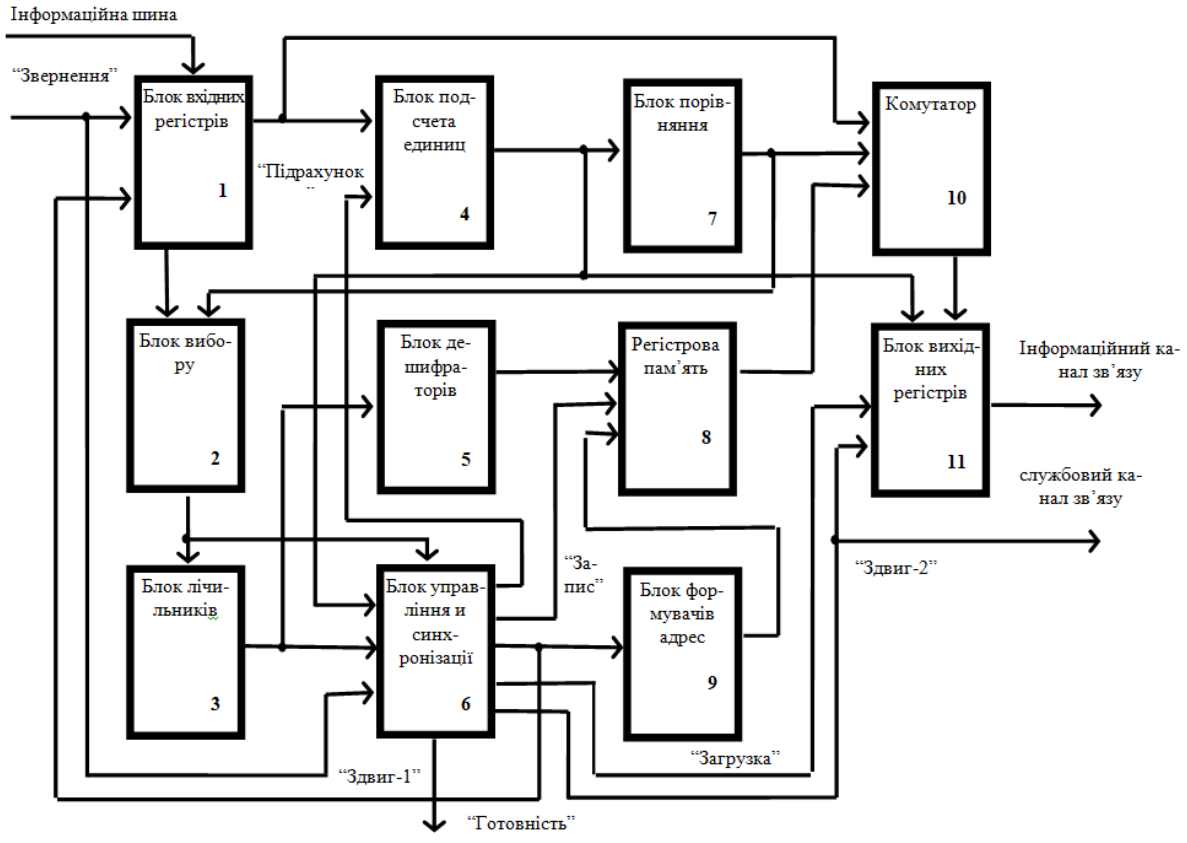


Рисунок 2.3 - Структурна схема адресно-векторного кодуючого пристрою ПК-2

На інформаційну шину кодує пристрої виставляється вихідне двійкове слово a_i, яке за сигналом "Звернення" заноситься в блок 1 вхідних регістрів. За

цим сигналом блок 6 управління і синхронізації виробляє сигнал "Підрахунок" для запуску блоку 4, який визначає число Bin k одиниць в кодованому слові. Після цього в блоці 7 це число порівнюється з верхнім і нижнім умовам адресно-векторного стиснення. У разі виконання або невиконання нерівностей блок 7 виробляє сигнал для комутатора 10, який підключає відповідно або вихід реєстрової пам'яті 8 або паралельний вихід блоку 1 на вхід блоку 11 вихідних регістрів. Крім цього, з цього ж сигналу блок 2 вибирає тип підраховують логічного біта, що подається на блок 3 лічильників. Домовимося, що $0 < \mathrm{Bin\,k} < \alpha - 1$ і підраховувати будуть виконавчі одиниці. По закінченню сигналу "Підрахунок" блок 6 починає виробляти сигнали "Здвиг-1", які одночасно надходять на блоки 1 і 9. У блоці 1 вхідних регістрів цей сигнал зрушує кодуються слово a_i на розряд через блок 2 на вхід блоку 3. Якщо розряд був одиничний, то вміст блоку 3 лічильників збільшується на 1. За значенням лічильників блоку 3 за допомогою блоку 5 дешифраторів адресується один з регістрів реєстрової пам'яті 8. у цей час в блоці 9 формується адреса розряду, в якому ця одиниця знаходиться. І сигналом "Запис" з блоку 6 дану адресу заноситься в пам'ять 8. За цим сигналом кожний раз вміст блоку 3 порівнюється вже з підрахованими Bin k. Якщо воно дорівнює даним значенням, то зрушення слова припиняється, і блок 6 формує сигнал "Завантаження", який записує адресну послідовність і число Bin k через комутатор 10 в блок 11 вихідних регістрів. Далі, сигналами "Здвиг-2" отримане адресно-векторне слово порозрядно надходить в інформаційний канал зв'язку. Якщо вміст блоку 3 менше Bin k, то відбувається знову зсув слова a_i в блоці 1 і в разі надходження одиниці процес повторюється. Якщо надходить двійковий нуль сигнал "Запис" не виробляється, процедури підрахунку і порівняння не відбуваються. У разі якщо $n\text{-}\alpha+1 < \mathrm{Bin}\ k < n$ робота кодує пристрої відбувається аналогічним чином, але вже накопичуються логічні нулі і в реєстрову пам'ять 8 записуються адреси нулів. У тому випадку, якщо $\alpha - 1 \le \mathrm{Bin\,k} \le \mathrm{n} - \alpha + 1$, то сигнал "Завантаження" виробляється відразу, без формування сигналів "Здвиг-1". У блок 11 при цьому заносяться вихідне слово a_i і двійкове число Bin k, які імпульсами "Здвиг-2" передаються далі по інформаційному каналу зв'язку. За службовому каналу зв'язку передаються синхроімпульсів. Після формування адресно-векторного слова і остаточної передачі його в канал зв'язку блоком 6 управління і синхронізації виробляється сигнал "Готовність" для прийому наступного довічного слова з зовнішнього пристрою.

Таким чином, як показано на рис. 2.2-2.3, блоки пристроїв ПК-1 і ПК-2 можна побудувати на нескладних функціональних елементах: регістрах, лічильниках, компараторах, мультиплексорах і т.д. Крім того, самі структури кодують пристроїв мають доста-точної простотою і наочністю. З огляду на все це в цілому, можна прийти до висновку про ефективність практичної реалізації цих пристроїв з точки зору витрат на проектування і налагодження їх роботи.

2.3 Опис методу АВ стиснення та відновлення інформації

Темою даної роботи є проектування системи що використовує алгоритм адресно-векторного кодування двійкових даних. Основу АВ - кодування складають виконавчий векторний метод та виконавчий адресний метод кодування.

Векторне кодування заключається в тім, що всім двійковим розрядам послідовності довжини n зіставляють будь-яку подію: при його наявності в двійковий розряд записується, наприклад, двійкова одиниця, а при його відсутності - двійковий нуль (або навпаки). Така послідовність називається вектором (керуючим, інформаційним, командним і т.д.). Недоліком такого кодування є те, що в безлічі випадків такий вектор може бути істотно не заповненим двійковими одиницями (або нулями), що сигналізує про його інформаційної надмірності. У цьому випадку ефективним з точки зору зниження надмірності є адресний метод кодування, який задає виконавчі адреси двійкових одиниць (або нулів) вектора. Довжина адресної послідовності визначається як [12].

$$L_a = \begin{cases} k \cdot \lceil \log_2 n \rceil, \ k \le n/2 \\ (n-k) \cdot \lceil \log_2 n \rceil, \ k > n/2 \end{cases} \qquad (2.8)$$

де k - число двійкових одиниць у вихідному двійковому векторі;

$\lceil \ \rceil$ - округлення в більшу сторону;

n - довжина кодової комбінації.

При малих кількостях адресованих двійкових 1 або 0 вдається помітно знизити довжину L_a інформаційного повідомлення в порівнянні з довжиною n вихідного вектора. Але адресний метод також має недолік - суттєве збільшення інформаційної надмірності, яке проявляється при числах двійкових одиниць або нулів близьких до половини довжини векторної

послідовності. Очевидним є побудова методу стиснення, заснованого на комбінуванні двох вищеназваних способах кодування – векторного і адресного.

Метод кодування $f_{av}: A \to B$, де $A = \{a_i: |a_i| = n, i = 1,..., |A|\}$ - вихідне, а $B = \{f_{av}(a_i); i = 1,..., |A|\}$ - результуюче безлічі двійкових слів, який полягає в переході від векторного методу кодування до адресного в залежності від числа k двійкових одиниць у вихідному слові згідно з умовою:

$$\begin{cases} 0 \le k < \alpha - 1 \\ n - \alpha + 1 < k \le n \end{cases} \qquad (2.9)$$

де, $\alpha = \dfrac{n}{\lceil \log_2 n \rceil}$

називається методом адресно-векторного стиснення (АВ-стиснення), а одержуваний код - адресно-векторного кодом (АВ-код), елементи якого утворюються згідно з правилом:

$$f_{av}(a_i) = \begin{cases} (\mathrm{Bin}\, k,\ VU(a_i)),\ 0 \le k < \alpha - 1 \\ (\mathrm{Bin}\, k,\ VZ(a_i)),\ n - \alpha + 1 < k \le n \\ (\mathrm{Bin}\, k,\ a_i),\ \alpha - 1 \le k \le n - \alpha + 1 \end{cases}, (2.10)$$

де $VU(a_i)$ і $VZ(a_i)$ - вектори адрес двійкових одиниць та нулів слова a_i [20,21].

Двійковий запис Bin k відіграє роль службового слова, що несе інформацію про кількість k двійкових одиниць вихідного слова, яке дозволяє визначити:

- метод кодування двійковій комбінації: векторний або адресний;
- її початок і очікувану довжину.

Для декодування адресно-векторних кодових комбінацій необхідно проводити аналіз відповідних їм довічних записів Bin k, по результату якого визначається метод кодування, що адресується двійковий рівень в разі адресного кодування і, далі, здійснюється зворотне перетворення до вихідних слів.

Застосування мікропроцесорів в системах стиснення даних (ССД) дозволяє підвищити стабільність характеристик та гнучкість систем передачі даних, уніфікувати обладнання для передачі і прийому сигналів з

різними видами модуляції, значно знизити вагу і габарити пристроїв. При цьому з'являються нові можливості вдосконалення ССД шляхом зміни програмного забезпечення.

У ССД мікропроцесори застосовують для: реалізації функцій управління дисциплін обслуговування абонентів; кодування і декодування інформації; виявлення та виправлення помилок; цифрової обробки сигналів; діагностики та контролю станів пристроїв і ін.

Основні напрямки розвитку мікропроцесорної елементної бази створення МПК БІС з фіксованою розрядністю і системою команд для широкого застосування, секціонованих разрядно-модульних МПК БІС, однокристальних мікро-ЕОМ і мікроконтролерів, спеціалізованих МПК БІС для цифрової обробки сигналів [22].

Тому доцільно вибрати алгоритм функціонування реалізація якого відповідала б вимогам, що пред'являються компактності, ергономіки, енергоспоживання. Таким вимогам повинені задовольняти алгоритми функціонування розроблюваного устрою. Для реалізації такого алгоритму буде вимагатися мікропроцесорна система, оскільки завдання дипломного проектування полягає в програмній, а не в апаратній реалізації, з використанням мінімуму апаратних затрат.

Необхідно розробити мікропроцесорний пристрій адресно-векторного відновлення даних. Пристрій необхідно розробити на мікропроцесорній системі. Відповідно до цього завданням розробимо алгоритм функціонування пристрою.

Надходить на вхід приймача закодована інформація повинна бути знову піддана кодування, з метою отримати дані в початковому вигляді. Будемо вважати, що отримана двійкова послідовність завдяки роботі пристроїв захисту від помилок була передана по каналу зв'язку без спотворень. Це дозволить отримати вихідну комбінацію ідентичну тій, яка надійшла на вхід передавача.

В алгоритмі функціонування пристрою відновлення АВ - відновлення даних матимуть місце такі моменти: на першому етапі аналізуються перші два біта заголовка, це необхідно для визначення розміру подальшого блоку, що містить інформацію про кількість одиниць у вихідній кодової комбінації; на другому етапі після зчитування другій частині заголовка, приймач на підставі таблиці критеріїв адресно-векторного стиснення приймає рішення

про те, чи була стиснута вихідна комбінація або вона була передана в початковому вигляді.

У разі, коли двійкова послідовність була піддана кодуванню, з вхідного потоку витягаються службові біти заголовка і комбінація в незмінному вигляді надходить на вихід приймача.

У вихідному регістрі формується нульовий або одиничний вектор на підставі даних другої частини заголовка, довжина якого визначається на підставі інформації, що міститься в двох перших бітах заголовка.

З поля даних послідовно зчитуються адреси елементів. Вони перетворюються з двійкового коду в унітарний код. А далі за допомогою логічної операції АБО накладається на сформований в вихідному регістрі вектор. Після обробки всіх адрес з поля даних, відновлена інформація надходить на вихід приймача.

Обробка поточної кодової комбінації вважається завершеною і очікується надходження наступної.

2.4.Розробка структурної схеми декодування інформації

Під розробкою структурної схеми зазвичай розуміють визначення функціонального складу вхідних в пристрій сполучень блоків (модулів), розрахунок і обгрунтування технічних вимог до вказаних блоків і встановлення необхідних електричних зв'язків між ними. У разі розробки прошивки для мікроконтролерів, структурна схема відображає основні блоки програми і підпрограм, а також встановлюються між ними логічні взаємозв'язки.

Пристрій має забезпечувати прийом кодових комбінацій, їх аналіз, кодування за розробленим раніше алгоритму і видачі обробленого пакета на вихід.

Необхідно розробити структурну схему. Ця схема повинна включати в себе: центральний процесор, блок постійної пам'яті, блок оперативної пам'яті, інтерфейсні блоки, для введення даних і для зв'язку ЦП з зовнішніми пристроями, блок переривань.

Принцип роботи має полягає в наступному. На центральний процесор через інтерфейсний блок виставляється вихідне двійкове слово, і при наявності сигналу готовності зчитується. Після зчитування записується в ОЗУ і на виході з'являється сигнал підтвердження отримання. Після цього

проводиться підрахунок кількості двійкових одиниць, після якого відповідно до алгоритму вибирається одне з наступних дій. Можлива ситуація, коли отримана комбінація буде оброблена і видана через комутатор в блок вихідних регістрів, де до неї прикріпиться заголовок зі службовою інформацією. В іншому ж випадку, після отримання наступної комбінації і відбувається об'єднання двох 8 бітних слів, які в подальшому розглядаються як єдине ціле. Об'єднані комбінації розміщуються в ОЗУ.

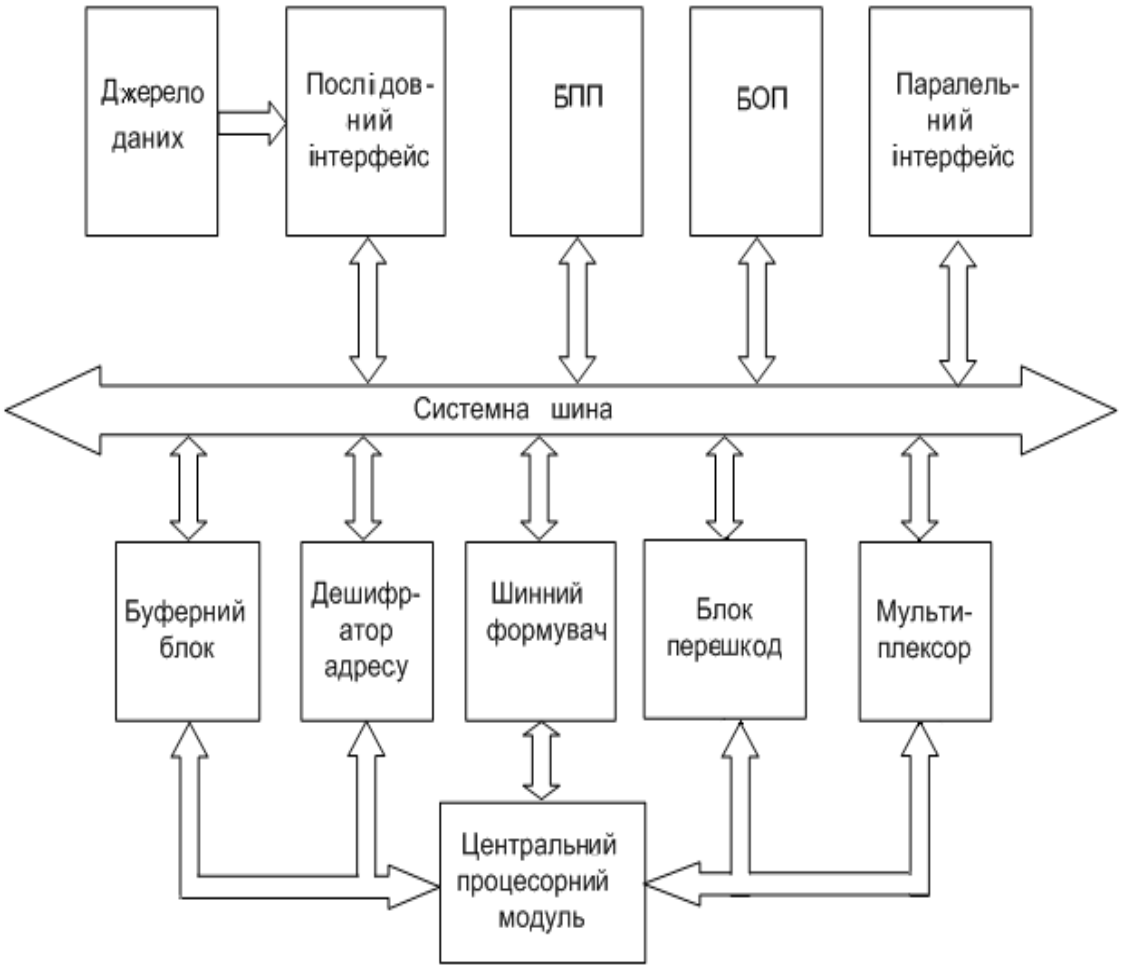


Рисунок 2.4 - Структурна схема системи

Мікропроцесорна система, будується за модульним принципом з шинної організацією зв'язків між блоками.

Проектована система складається з мікропроцесора, він управляє роботою пристрою, формуючи керуючі сигнали, проводить всі обчислення.

До його складу також входить дешифратор адреси, керуючий адресацією потоку даних до запам'ятовуючим пристроям.

Блок постійної пам'яті (БПП) призначений для постійного зберігання інформації (програм) необхідної для роботи мікропроцесорного пристрою, інформація з нього тільки зчитується.

Блок оперативної пам'яті - (БОП) - забезпечує тимчасове зберігання інформації.

Шини адреси (ША), даних (ШД), управління (ШУ) суміщені в одну системну шину - являють собою паралельні провідники (у фізичному сенсі), призначені для передачі багаторозрядних цифрових кодів або в двох, або в одному напрямку. Гідність шинних зв'язків - істотне зменшення числа сполучних проводів.

До складу буферного блоку входять два буферних регістра, вони потрібні для поділу сигналів шини даних і шини адреси.

Шинний формувач використовується для підвищення здатності навантаження МП, відповідно підвищуючи ефективність роботи пристрою.

Мультиплексор два в один перетворює сигнали процесора в сигнали читання / запису пам'яті і зовнішніх пристроїв.

Так як наш пристрій здійснює декодування цифровий комбінації яка надходить з каналу зв'язку в послідовному коді, то для її введення використовується послідовний інтерфейс.

Після декодування цифрова комбінація набуде вигляду паралельного коду для виведення якого потрібен паралельний інтерфейс.

Блок переривань служить для реалізації переривання від паралельного інтерфейсу, яке повідомляє процесору про те, що комбінація оброблена і видана в канал зв'язку. Після переривання процесор очищає пам'ять від проміжних даних і дозволяє надходження нової комбінації.

Побудова пристрою на жорсткій логіці для виконання такого завдання невиправдано збільшить апаратні витрати, і як наслідок габарити, масу приладу і енергоспоживання, що є істотнимчинником при розробці пристрою. Тому застосування мікропроцесора дозволить підвищити стабільність характеристик і гнучкість проектованого системи.

3. ВИБІР ТА ОБГРУНТУВАННЯ АЛГОРИТМУ ФУНКЦІОНУВАННЯ ТА СТРУКТУРНОЇ СХЕМИ

3.1. Загальні алгоритми адресно-векторного стиснення і відновлення двійковій інформації.

Алгоритм являє собою сувору систему правил, що визначає послідовність дій над деякими об'єктами [20, 21]. Відповідно до визначення алгоритму, він повинен мати наступні властивості: 1) кінцівка; 2) виразністю і ефективністю використовуваних операцій; 3) мати введення і виведення [20].

З усіх можливих найшвидшими алгоритмами кодування і декодування двійкових n-розрядних слів є алгоритми, що використовують повні пошукові таблиці. Вони отримують результат, по суті, за час однієї операції, але платою за таку швидкість є марнотратне використання пам'яті (2^n осередків), що для великих знань n робить такі алгоритми практично непридатними до вживання [40]. Таким чином, при існуючих обмеженнях на довжину програми або апаратні витрати необхідно знайти, дослідити і реалізувати алгоритми адресно-векторного кодування і декодування для включення в програмно-алгоритмічне забезпечення АСУ, які були б досить швидкодіючими і не вимагали великих обсягів пам'яті.

У разі адресно-векторного стиснення алгоритм визначається математичної функцією f_{av} (2.28), що ставить в однозначна відповідність двійковій n-розрядної послідовності a_i адресно-векторне кодове слово $f_{av}(a_i)$ зі змінною довжиною [33].

Загальний алгоритм АВК, що виконує адресно-векторне кодування $f_{av}: A \to B$; де $A = \{a_i : |a_i| = n, i = 1,\dots,|A|\}$ - вихідне джерело інформації, $B = \{f_{av}(a_i), i = 1,\dots,|A|\}$ - результуюча множина адресно-векторних кодових слів, виглядає наступним чином:

1. Початкова установка і прийом вихідного двійкового n-розрядного вектора a_i у вхідний регістр.

2. Обчислення числа k одиниць у векторі a_i.

3. Якщо число k логічних одиниць не задовольняє систему нерівностей (2.27), то двійковий вектор залишається без змін і перейти до кроку 7. В іншому випадку перейти до наступного кроку.

4. Якщо число k одиниць більше або дорівнює 0 і менше нижнього умови стиснення (2.27), то перейти до кроку 6. В іншому випадку перейти до наступного кроку.

5. Формування вектора $VZ(a_i)$ адрес логічних нулів і перейти до кроку 7.

6. Формування вектора $VU(a_i)$ адрес логічних одиниць і перейти до кроку 7.

7. Формування службового слова Bin k.

8. Передача адресно-векторного кодового слова $f_{av}(a_i)$ в вихідний регістр і останов алгоритму.

Блок-схема загального алгоритму АВК приведена на рис. 3.1.

Відповідно до визначення [20, 22, 23], даний алгоритм кінцевий, так як час кодування або число виконуваних операцій обмежена зверху завдовжки n вихідного двійкового слова. Алгоритм АВК має введення у вигляді слова $(a_i) \in A$ і висновок у вигляді адресно-векторного слова $f_{av}\ (a_i) \in B$. Алгоритм стиснення характеризується визначеністю і ефективністю операцій, оскільки, по-перше, має кінцевим набором досить простих математичних дій, а, по-друге, всі операції проводяться над простими числами.

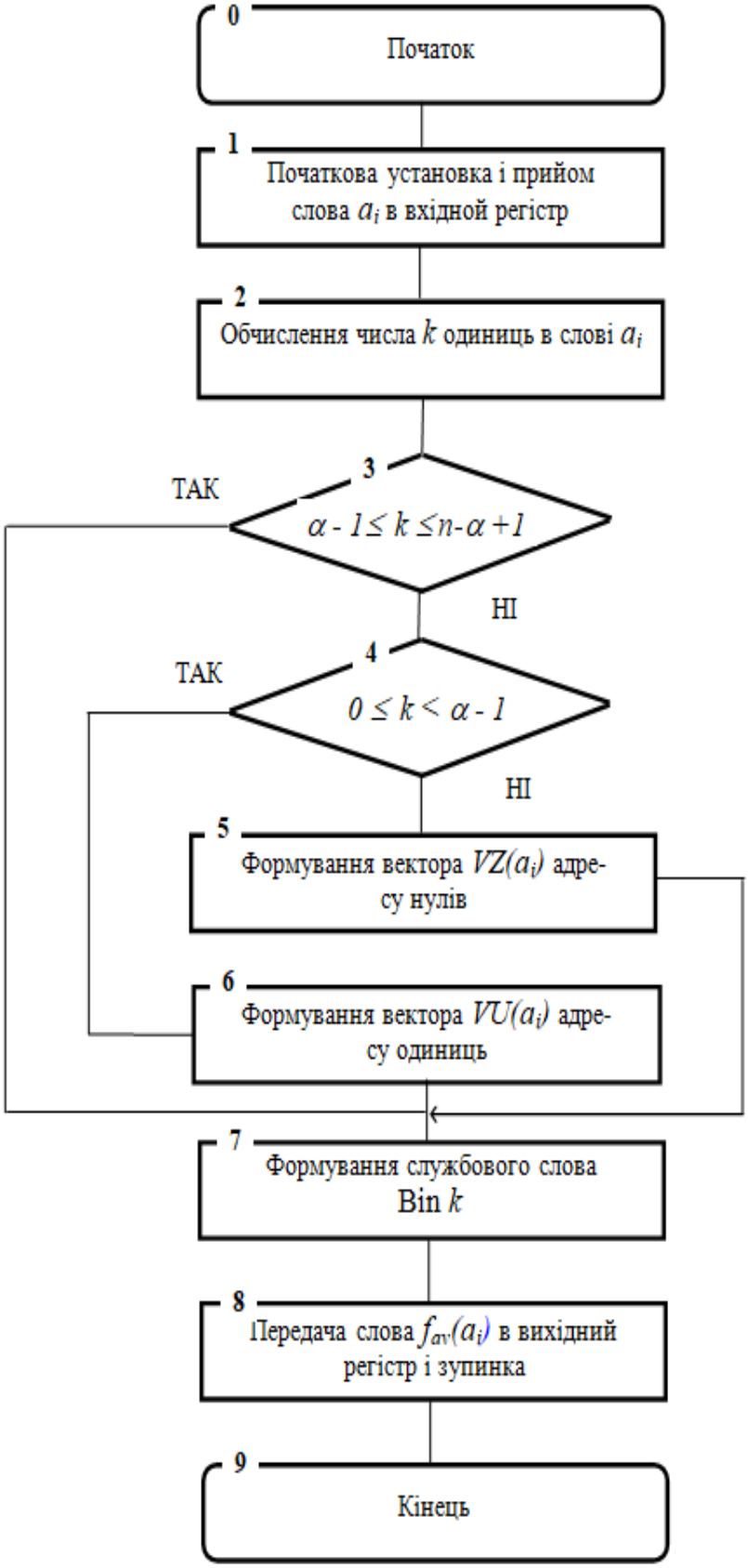


Рис. 3.1 - Блок-схема загального алгоритму АВК.

У разі адресно-векторного декодування відповідний алгоритм визначається математичної функцією g = $[f_{av}\ (a_i)]$, яка є зворотною до функції $f_{av}\ (a_i)$.

Загальний алгоритм АВД, що здійснює адресно-векторне декодування

$g_{av}: B \to A$, де $B = \{f_{av}\ (a_i); i = 1, \ldots, |A|\}$ - вихідна множина адресно-векторних кодових слів,

$A = \{a_i: |a_i| = n, i = 1, \ldots, |A|\}$ - результуюча множина двійкових векторів:

1. Початкова установка в нуль вхідного і вихідного регістрів і прийом вихідного адресно-векторного слова $f_{av}\ (a_i)$ у вхідний регістр.

2. Аналіз службового слова Bin k.

3. Якщо число k логічних одиниць дорівнює 0, то нульовий вміст вихідного регістра є результуючий двійковий вектор a_i і перейти до кроку 10.

4. Якщо число k логічних одиниць дорівнює n, то перейти до кроку 9.

5. Якщо k не задовольняє системі нерівностей, то перейти до наступного кроку. В іншому випадку проводиться перехід до кроку 7.

6. Передати в вихідний регістр без службового слова Bin k вміст вхідного регістру і перейти до кроку 10.

7. Відповідно до адресами вектора $VU\ (a_i)$ або вектора $VZ\ (a_i)$ в вихідному регістрі заповнюються логічними одиницями розряди з відповідними адресами номерами, і нулями всі інші розряди.

8. Якщо число k логічних одиниць менше нижнього умови стиснення, то перейти до кроку 9.

9. Інвертувати вміст вихідного регістра.

10. Зупинка алгоритму.

Блок-схема загального алгоритму АВД приведена на рис. 3.2.

Декодуючий алгоритм кінцевий, так як час його роботи або число тактів виконання обмежена зверху числом адрес векторів $VU\ (a_i)$ і $VZ\ (a_i)$, яких не може бути більше, ніж a-1. Алгоритм АВД має введення у вигляді довічного адресно-векторного слова $f_{av}\ (a_i)$ ∈B і висновок у вигляді перетвореного до первісного вигляду довічного слова a_i. Розглянутий алгоритм також володіє визначеністю і ефективністю використовуваних операцій, оскільки, по-перше, має кінцевий набором досить простих математичних дій, а, по-друге, всі операції проводяться над простими числами.

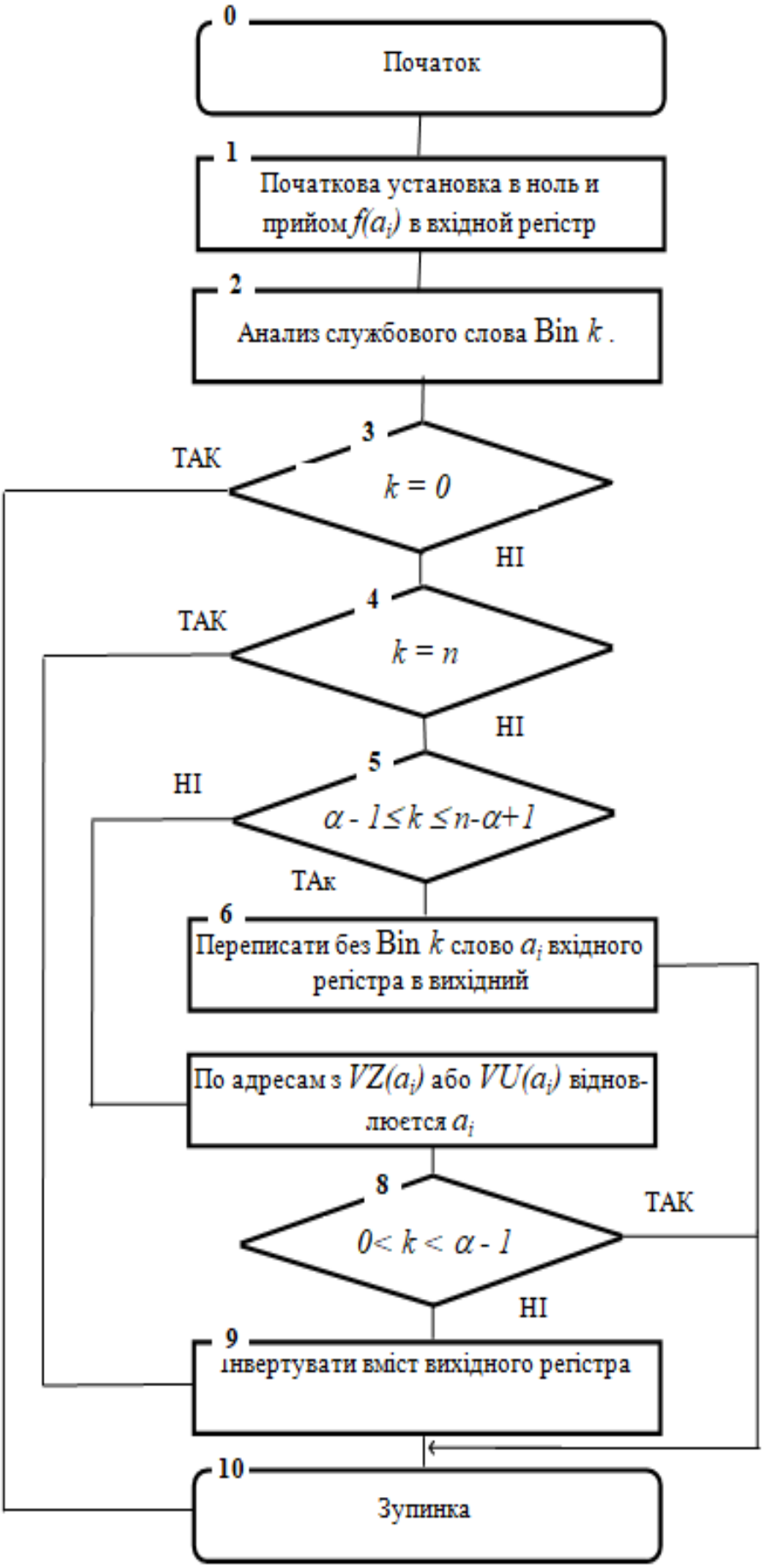


Рис. 3.2 - Блок-схема загального алгоритму АВД.

Згідно з методом структурного програмування [22, 23] подальша розробка загальних алгоритмів АВК і АВД вважає, що кожен крок має власну систему правил. Подальша алгоритмічна структуризація залежить від вимог, що пред'являються до складносткім характеристикам: числу кроків виконання, довжині програми і обсягом витрачається пам'яті. З точки зору складності та введених обмежень найбільший інтерес представляють підрахунок числа логічних одиниць і формування адрес - кроки 2 і 6 кодуючого алгоритму і відновлення двійкового вектора за адресами - крок 7 декодуючого алгоритму.

4. РОЗРОБКА ФУНКЦІОНАЛЬНОЇ СХЕМИ АВ ВІДНОВЛЕННЯ ДВІЙКОВОЇ ІНФОРМАЦІЇ

4.1 Розробка функціональної схеми системи

За структурною схемою, наведеної на рисунку 3.2 побудуємо функціональну схему системи адресно-векторного відновлення даних. Для цього опишемо призначення кожного з блоків структурної схеми. Отже, як вже описувалося вище, проектована система буде складається з вище представленої мікропроцесорної системи.

Центральний процесорний модуль є центральним блоком контролера. Він забезпечує управління і синхронізацію роботи всього пристрою, забезпечує прийом, видачу, зберігання і обробку даних. ЦП відповідає за реалізацію алгоритму відновлення даних і коректну роботу системи. У складі центрального процесорного модуля, також, присутній зовнішній резонатор для стабілізації частоти процесора.

Центральний процесор виконує всі логічні і арифметичні операції згідно алгоритму АВ – відновлення даних, зберігаючи проміжні обчислення в блоці БОП.

Для зберігання констант, проміжних даних, коду керуючої програми, результатів обчислень необхідна пам'ять. До складу блоку пам'яті в проектованому пристрої входить в блок постійної пам'яті (БПП) і блок оперативної пам'яті (БОП). У БПП буде зберігатися код програми, в БОП – вхідні змінні, проміжні дані, результат обчислень. БОП бувають статичного і динамічного типів. Так як обсяг оперативної пам'яті в проектованому контролері малий, то немає сенсу застосовувати в ньому динамічне БОП. Застосування БОП статичного типу дозволяє вирішити задачу збереження даних в пам'яті – на відміну від динамічного БОП статичну не вимагає циклів регенерації пам'яті. Це дозволяє істотно спростити апаратну частину пристрою.

Для поділу сигналів шини адреси і шини даних застосуються буферний блок. Він складається з двох буферних регістрів. Поява в першому такті машинного циклу на шині першого регістра старшого байта адреси, а на шині другого – молодшого, стробірується сигналом процесора, який використовується для дозволу запису в регістри. При передачі по шині даних цей сигнал відсутній. Таким чином, в регістрах буде записана адреса, а дані

будуть передаватися через шинний формувач. До того ж, регістри і шинний формувач виконують функцію збільшення навантажувальної здатності ЦПМ.

Мультиплексор потрібен для комутації сигналів процесора до зовнішніх пристроїв читання-запису (для БОП, БПП, і подібних схем).

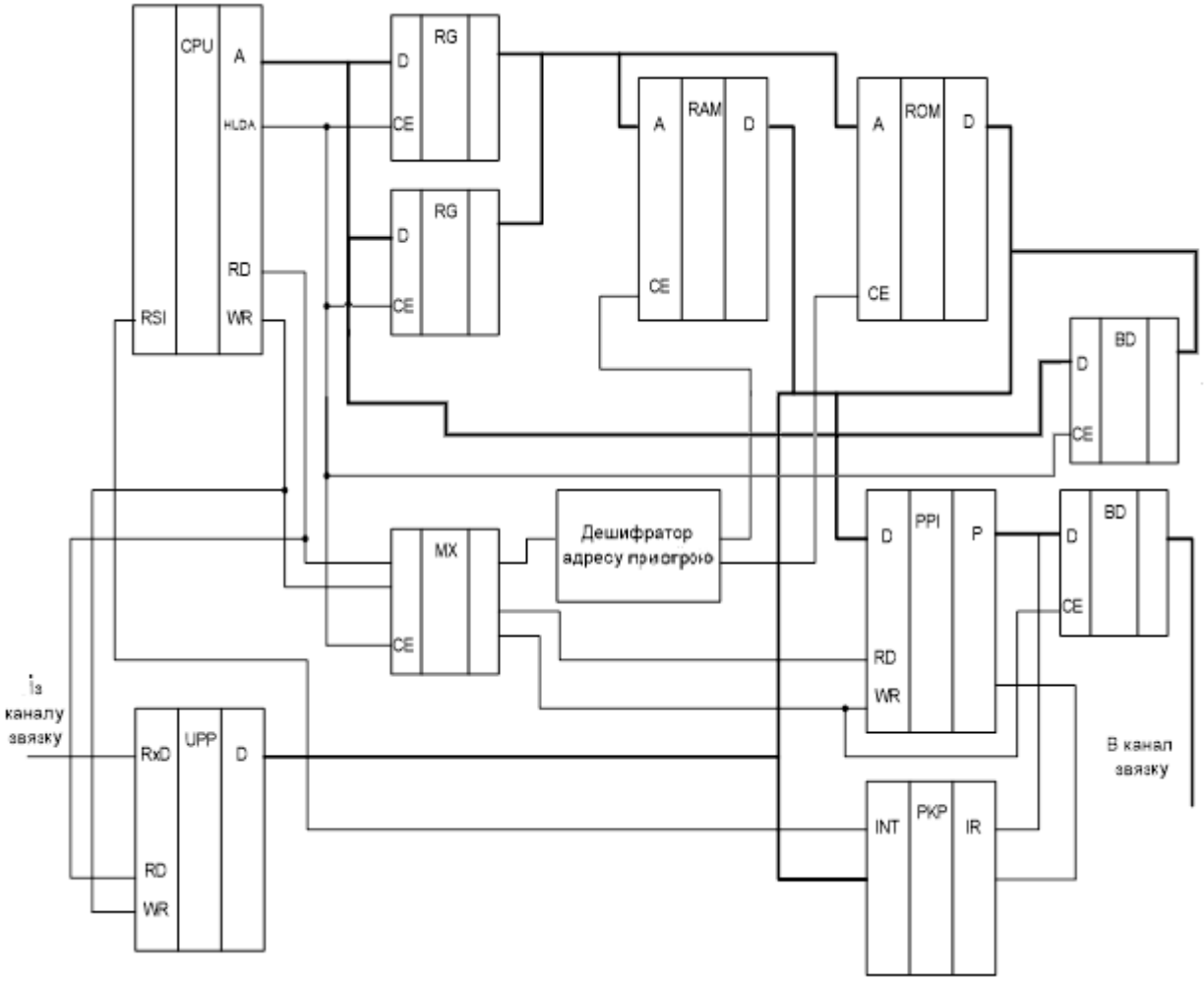


Рисунок 4.1 – Схема електрична функціональна системи

ЦП звертається не тільки до пам’яті, але і до зовнішніх пристроїв (пам’ять,46оперативний блок), для цього в системі присутній дешифратор адреси. Він складається з логічних елементів і по сигналу процесора здійснюють вибірку пристрою.

При заповненні всіх портів виведення паралельного інтерфейсу даними і їх подальшою передачею в канал зв’язку, виробляється сигнал, який надходить на блок перешкод. У свою чергу блок перешкод обробляє цей сигнал і виробляє переривання яке надходить на вхід процесора, реалізуючи, таким чином, скидання процесора в автоматичному режимі. Після обробки переривання процесор видає команду скидання БОП, буферних регістрів, шинних формувачів, і починається новий цикл обробки кодових комбінацій.

Послідовний інтерфейс служить для організації обміну між зовнішніми пристроями і центральним процесором в послідовному форматі. Цей блок необхідний так як дані на проектоване пристрій надходять в закодованому, послідовному вигляді. Схема електрична функціональна проектованого пристрою показана на малюнку 4.1.

І так пройдемося 47одрібніше по кожному блоку даної функціональної схеми.

Ми маємо такі основні блоки при роботі системи : Центральний процесорний блок, Блок постійної пам’яті, Блок оперативної пам’яті, Блок дешифратора, Блок перешкод, Буферний блок .

4.1.1. Центральний процесорний блок

Відповідає за роботу системи і є центральним блоком контролера. Він забезпечує управління і синхронізацію роботи всього пристрою, забезпечує прийом, видачу, зберігання і обробку даних, відповідає за реалізацію алгоритму відновлення даних і коректну роботу пристрою, виконує всі логічні і арифметичні операції згідно алгоритму АВ – відновлення даних.

4.1.2. Блок пам’яті

Даний блок застосовується для зберігання всієї інформації системи пристрою і поділяється на два блоки: блок оперативної пам’яті та блок постійної пам’яті.

4.1.2.1. Блок оперативної пам’яті

Даний блок застосовується для зберігання проміжні дані, результат обчислень.

4.1.2.2. Блок постійної пам’яті

Даний блок застосовується для зберігання коду програми та зберігання подальшої статичної інформації яка поступає в систему під час виконання операцій.

4.1.3. Блок перешкод

Відповідає за обробку сигналів і виробляє переривання яке надходить на вхід процесора, реалізуючи, таким чином, скидання процесора в автоматичному режимі. Після обробки переривання процесор видає команду скидання БОП, буферних регістрів, шинних формувачів, і починається новий цикл обробки кодових комбінацій.

4.1.4. Буферний блок

Складається даний блок з двох буферних регістрів. Слідкує за появою в першому такті машинного циклу на шині першого регістра старшого байта адреси, а також на шині другого – молодшого, стробіруется сигналом процесора, який використовується для дозволу запису в регістри. При передачі по шині даних цей сигнал відсутній, якщо сигнал присутній то він працює неправильно. Таким чином, в регістрах буде записана адреса, а дані будуть передаватися через шинний формувач. До того ж, регістри і шинний формувач разом виконують функцію збільшення навантажувальної здатності ЦПМ.

4.2. Структурна схема блоків відповідальних за декодуючи та кодуючи частини системи

Структурна схема (рис. 4.2) містить два блоки, пристрій фазування по циклу (ПФЦ) і декодер. ПФЦ складається з пристрою реєстрації ознаки фази, пристрої визначення маркерної комбінації, блоку захисту маркерної комбінації і пристрої управління ПФЦ. Декодер складається з регістра номера блоку, подільника, регістра інформаційних і перевірочних елементів, лічильника зрушень, ключів і пристрої управління декодером.

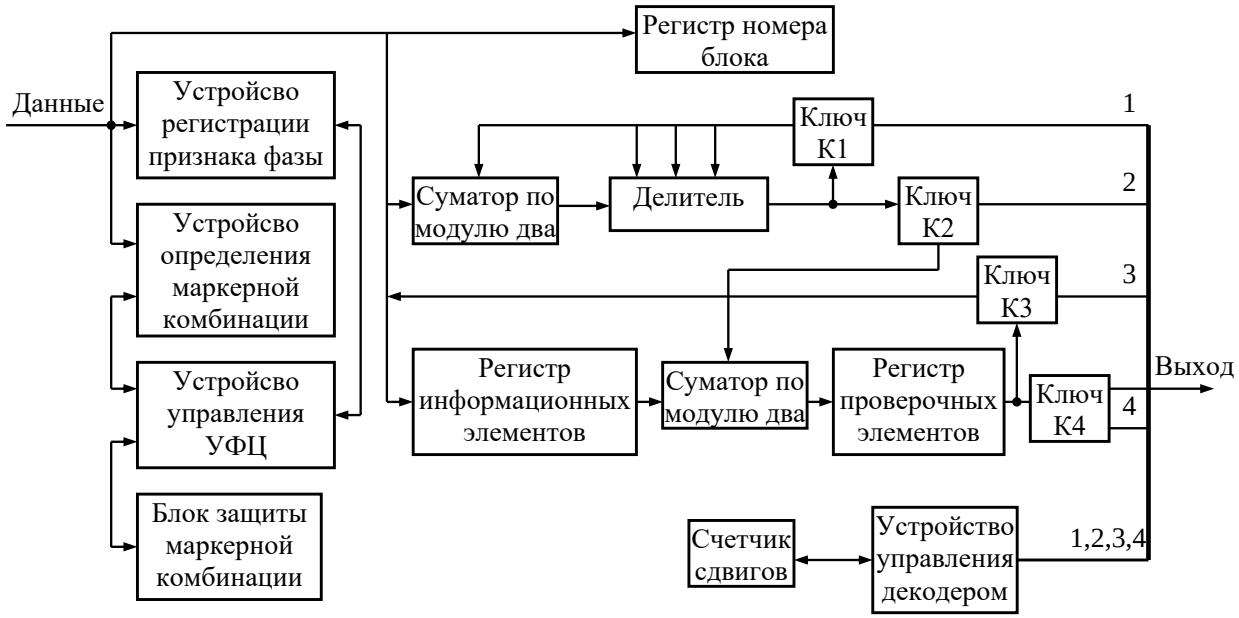


Рисунок 4.2 - Структурна схема ПЗП (пристрій захисту від похибок).

При появі на вході приймальної частини ПЗП двійковій послідовності включається пристрій реєстрації ознаки фази. Якщо на вхід надходить ознака фази, то включається режим фазування. Пристрій визначення маркерної комбінації і блок захисту маркерної комбінації встановлюють правильність маркера відповідно до алгоритму.

Після встановлення правильності маркера в декодер надходить повідомлення складається з номера блоку, інформаційної та перевірочної частин повідомлення. Номер блоку зберігається в спеціально відведеному регістрі. Інформаційна і перевірочна частини повідомлення передаються в пристрій виявлення помилок і декодування.

5. ВИБІР ЕЛЕМЕНТНОЇ БАЗИ ТА РОЗРОБКА ПРИНЦИПОВИХ ЕЛЕКТРИЧНИХ СХЕМ БЛОКІВ

5.1. Розробка принципової схеми блоків ПЗП

Розглянемо одну із основних частин даної системи на базі елементів блоку даного блоку.

5.1.1 Вибір елементної бази

При виборі елементної бази необхідно врахувати, що нам для передачі даних по каналу зі швидкістю 600 біт / сек. необхідна не менш швидкодіюча логіка. Реалізовувати принципову схему можна на "жорсткої" або "гнучкою" (програмованої) логіці. При заданих умовах і спираючись на складений алгоритм роботи УЗО і його структурну схему, можна зробити висновок про відносно невеликі апаратурних витратах, які будуть потрібні для реалізації даного пристрою. З урахуванням цього, цілком раціонально побудувати принципову схему, використовуючи "жорстку" логіку. Для досягнення необхідної швидкодії, необхідно використовувати мікросхеми серії КР1533 і КМ555.

Наведемо величини основних параметрів мікросхем обраних серій.

Таблиця 5.1. Параметри мікросхем КР1533 і КМ555.

Параметр	КР1533	КМ555
$I_{вх}$ (0) mA, не більше	-0,4	-0,2
$I_{вх}$ (1) mA, не більше	0,02	0,02
$U_{вих}$ (0) mA, не більше	0,4	0,4
$U_{ви5х}$ (1) mA, не менше	2,5	2,5
$k_{роз}$	10	20
$T_{здр}$ (1,0) нс, не більше	10	4
$P_{пот}$, mВт, не більше	2	1
$U_{пот}$, В, не більше [1]	0,7	0,8
f , не більше [2]	15	100

1 - найменший з 2-х значень допустимого рівня перешкоди.

2 - f - частота перемикання

де $I_{вх}$ (0) - вхідний струм логічного нуля;

$I_{вх}$ (1) - вхідний струм логічної одиниці;

$U_{вих}$ (0) - максимальне значення вихідної напруги, що відповідає рівню логічного нуля, при якому забезпечується нормальна робота наступних ІС;

$U_{вих}$(1) - мінімальне значення вихідної напруги, що відповідає рівню логічної одиниці, при якому забезпечується нормальна робота наступних ІС;

$k_{роз}$ - коефіцієнт розгалуження по виходу визначає число входів елементів даної серії, яке може бути без порушення працездатності підключено до виходу попереднього логічного елемента;

$T_{здр}$ (1,0) - час затримки переходу ІС зі стану логічного нуля в стан логічної одиниці і навпаки;

$P_{пот}$ - потужність, споживана базовим логічним елементом від джерела живлення;

$U_{пот}$ - максимально допустиме значення статичної перешкоди.

f - максимальна частота перемикання.

При розробці також необхідно відразу врахувати що мікросхеми, як правило, містять не один елемент, а кілька однакових. Тому вже при проектуванні необхідно намагатися використовувати однотипні елементи, які можна згруповують для простоти монтажу і розведення друкованої плати.

5.2 Розробка принципової схеми пристрою фазування по циклу

Блок перевірки ознаки фази і блок перевірки маркерной комбінації конструктивно схожі, тому наведемо в даному пункті розробку тільки одного з них. Принципова схема блоку перевірки маркерної комбінації приведена на малюнку 4.1. Дана схема містить два восьмирозрядних зсувних регістра з послідовним входом і паралельними виходами КР555ИР8. Вони призначені для прийому десяти розрядного маркера. Для порівняння прийнятого маркера з правильним використовується три восьмирозрядних пристрою порівняння КР555СП1. Для фіксації правильного маркера використовується тригер К555ТМ2.

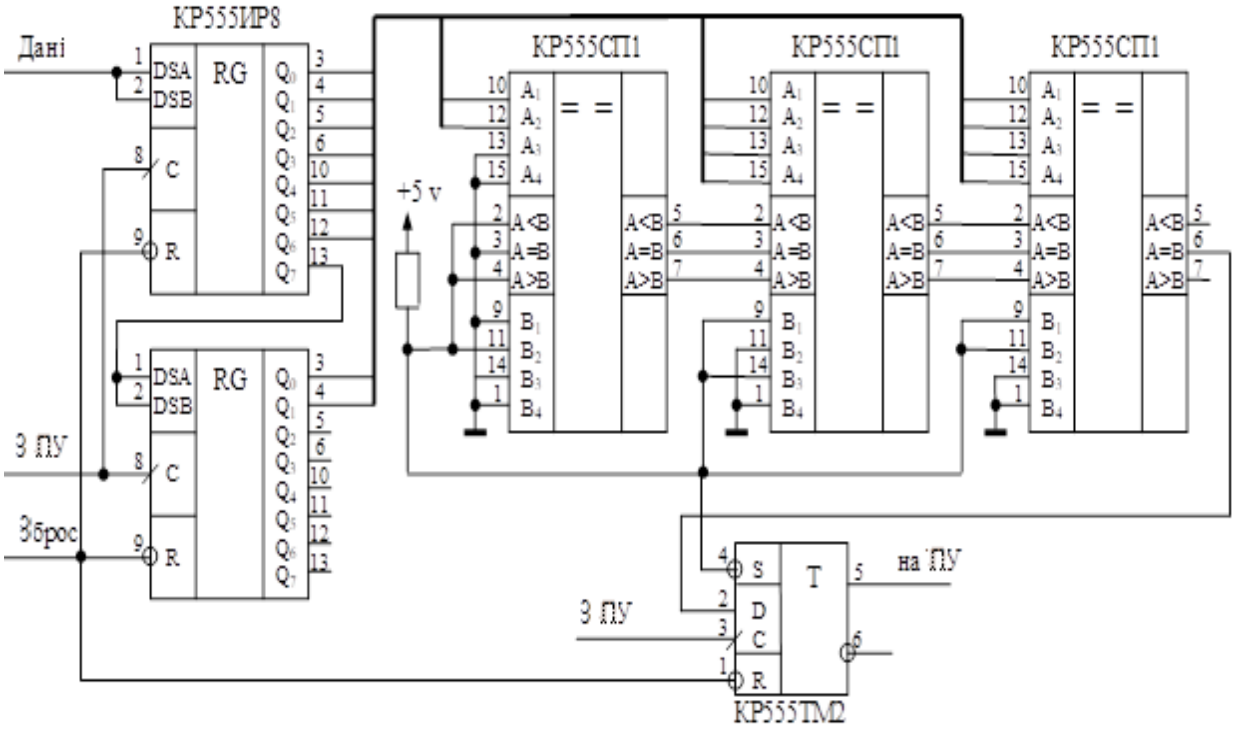


Рисунок 5.1 - Принципова схема блоку перевірки маркерної комбінації

Пристрій управління (ПУ) УФЦ практично ідентично пристрою управління декодера. Одне з них буде приведено в пункті 4.4.

5.3 Розробка принципової схеми декодеючої системи

Декодер представлено регістрами номера блоку, дільником і регістрами зсуву інформаційної та перевірочної частин повідомлення. Так як дільник представляє в цьому пристрої найбільшу складність, то наведемо його схему на малюнку 5.2. Дільник складається з п'яти суматорів К155ИМ1 і двох зсувних регістрів К155ИР13.

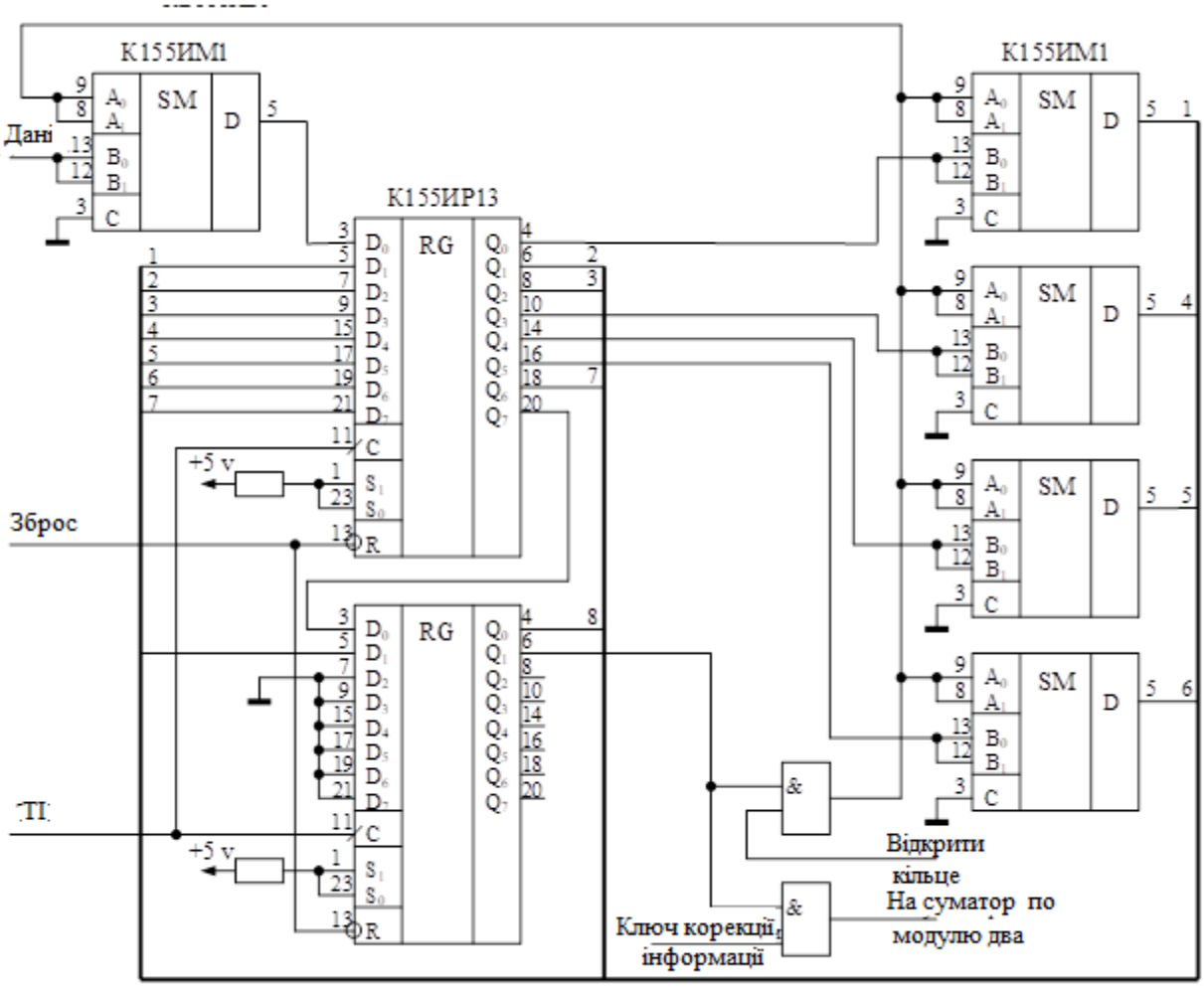


Рисунок 5.2 Принципова схема дільника

5.4 Розробка принципової схеми системи керування

Пристрій управління декодером і пристрій управління УФЦ практично ідентичні. Наведемо в цьому пункті принципову схему пристрою управління декодером (рис. 5.3). Цей пристрій складається з трьох довічних лічильників К555ИЕ7, ПЗУ, регістра К555ИР22, формувача імпульсів запису в регістр К555ИР22 на мікросхемі К555АГ3 і ключів на мікросхемі К555ЛИ1.

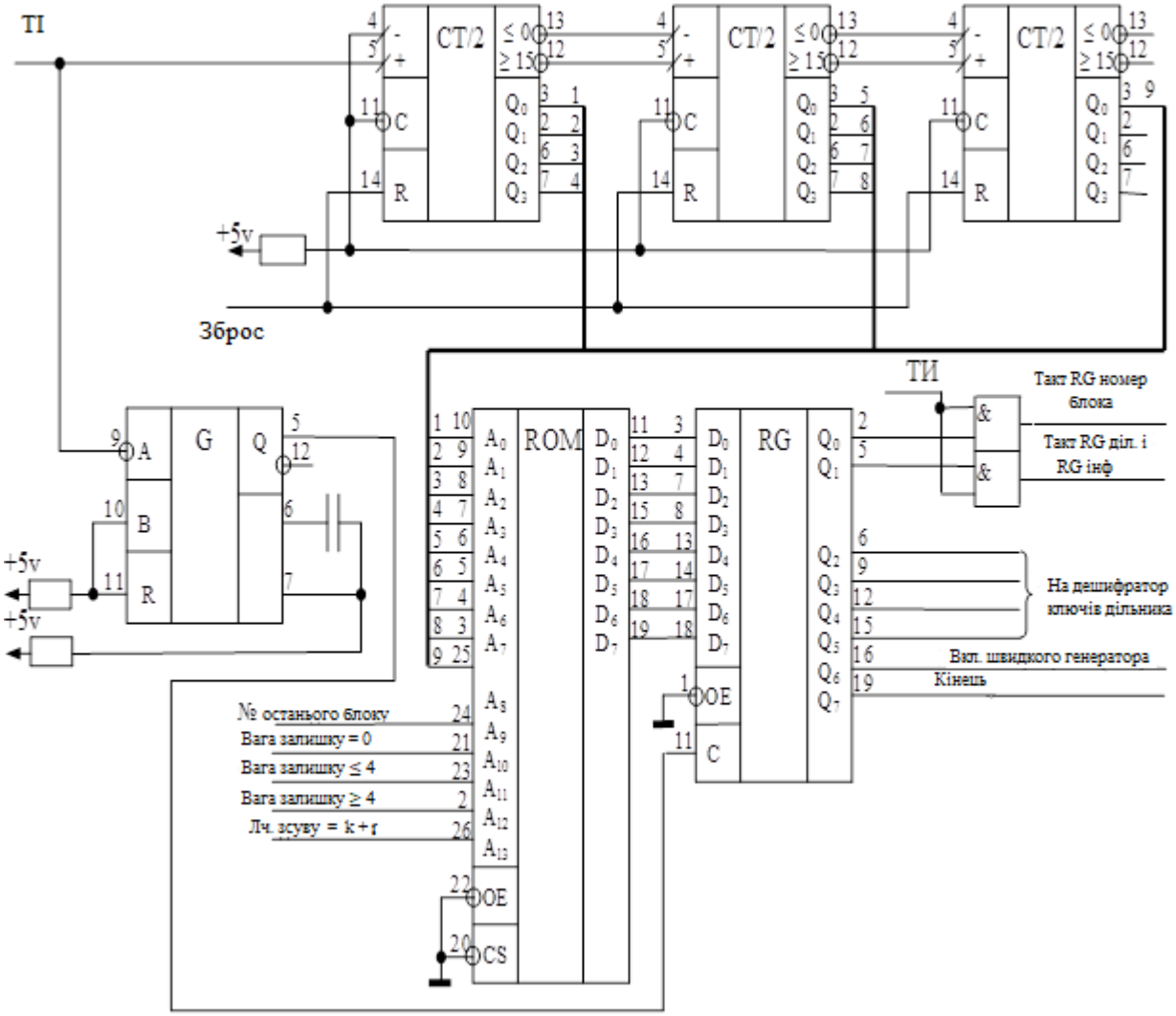


Рисунок 5.3 Принципова схема системи управління декодером

Дана схема являться фінальною для даного блоку і розкриває в собі всю систему даного методу.

6. ТЕХНІКО - ЕКОНОМІЧНА ЧАСТИНА

6.1 Розрахунок собівартості виготовлення системи

Собівартість системи – це виражені у фінансовій формі поточні витрати компанії на його виробництво і збут. Витрати на виготовлення системи формують виробничі витрати, а собівартість на виробництво і збут – повну собівартість. Розрахунок собівартості системи (установки) за статтями витрат та собівартістю називається калькуляцією.

Витрати, пов'язані з виготовленням та збутом реалізації системи (установки) групуються за такими статтями:

− матеріали та комплектуючі;

− основна заробітна плата;

− додаткова заробітна плата;

− відрахування на соціальні заходи;

− витрати на утримання і експлуатацію устаткування;

− загальновиробничі витрати;

− адміністративні витрати; − витрати на збут.

Витрати на матеріали та комплектуючі вироби визначаються виходячи з ціни за одиницю матеріалу/комплектуючого та їх необхідної кількості (табл. 6.1). Дані про ціни на матеріали та комплектуючі варто брати з відомостей (прайс-листів, каталогів, web-сайтів) виробників і постачальників матеріалів, сировини, комплектуючих, послуг в розрахунку на 1 одиницю випуску.

Таблиця 6.1 – Розрахунок коштів на комплектуючі

Найменування комплектуючих	Кількість, шт	Ціна за од., грн	Вартість, грн
Конденсатори			
К75-40б	2	9,5	19
К1-1	2	1,95	3,9
Мікросхеми			
К555АГ3	1	8,27	8,27

К555ЛИ1	1	5,40	5,40
КР1533	1	2,70	2,70
КМ555	1	7,66	7,66
С2-23-0,125	1	1,4	1,4
С2-23-0,125	1	3,2	3,2
Тригера			
К555ТМ2	1	6,50	6,50
Суматори			
К155ИМ1	5	6	30
Лічильники			
К555ИЕ7	3	4	12
Регістр			
КР555СП1	3	2,90	8,70
К555ИР22	1	6	6
КР555ИР8	2	15,16	30,32
К155ИР13	2	10	20
Акумулятори			
CR2032 – 3В	1	15	15
Сумарна вартість витрат			180,05

Таблиця 6.2 – Приклад розрахунку витрат на сировину та матеріали

Матеріал, сировина	Одиниця виміру	Норма витрат	Ціна за одиницю, грн	Вартість, грн.
Склотекстоліт	$м^2$	0,06	95	4,81
Каніфоль	кг	0,05	16	0,68
Флюс	кг	0,03	140	2,87
Припій	кг	0,07	260	15,8
Лак	кг	0,03	365	7,5
Сумарні витрати				31,66

З урахуванням транспортно-заготівельних витрат ($k_{т-з}$=5÷15%) вартість комплектуючих та матеріалів складі:

$$КМ = (180,05 + 31,66)\ (100+10) / 100 = 232,88 \text{ грн.} \qquad (6.1)$$

Витрати на основну заробітну плату (Зо):

$$Зо = \sum_{i=1}^{n} Тг_i \cdot Нч_i \cdot n \qquad (6.2)$$

де $Тг_i$ – годинна тарифна ставка окремого спеціаліста (інженера електронщика, лаборанта тощо), що задіяний у виробництві пристрою (установки), грн/год;

$Нч_i$ – витрачений час робітником на виробництво і наладку пристрою (установки);

n – кількість працівників, які задіяні у виробництві пристрою (установки).

Годинна тарифна ставка розраховується, виходячи з величини місячного окладу спеціаліста:

$$Тг_i = \frac{Тм_i}{Вф_i \cdot 8} = \frac{6000}{24 \cdot 8} = 31{,}25 \text{ грн.} \qquad (6.3)$$

$Тм_i$ – місячний оклад (ставка) спеціаліста, грн;

$Вф_i$ – фактично відпрацьований час за розрахунковий період (місяць), днів.

$$Зо = \sum_{i=1}^{n} Тг_i \cdot Нч_i \cdot n = 31{,}25 \cdot 40 \cdot 4 = 2500 \text{ грн.} \qquad (6.4)$$

Додаткова заробітна плата (10÷30% від Зо):

$$Зд = Зо \cdot \frac{Кд}{100} = 2500 \cdot \frac{25}{100} = 625 \text{ грн.} \qquad (6.5)$$

де Кд – відсоток додаткової заробітної плати.

Відрахування на соціальні заходи містять відрахування від суми основної і додаткової зарплати за встановленими ставками:

- на державне страхування від нещасних випадків;
- на обов'язкове державне соціальне страхування на випадок безробіття;
- у зв'язку з тимчасовою втратою працездатності і витратами, зумовленими народженням дитини і похованням

$$Всз = (Зо + Зд) \cdot \frac{36{,}3}{100} = (2500 + 625) \cdot \frac{36{,}3}{100} = 1134{,}38 \text{ грн.} \qquad (6.6)$$

Витрати на утримання та експлуатацію устаткування складають 120-150% від основної заробітної плати:

$$В_{УЕУ} = Зо*1,4 = 2500*1,4 = 3500 \text{ грн.} \quad (6.7)$$

Загально виробничі витрати визначаються із відомостей по аналізу повної собівартості виробу і в середньому можуть складати 130-250 % від основної заробітної плати.

$$В_{ЗВ} = 2500*1,8 = 4500 \text{ грн.} \quad (6.8)$$

Виробнича собівартість визначається як сума статей витрат:

$$С_В = КМ + Зо + З_Д + В_{СЗ} + В_{УЕУ} + В_{ЗВ.} \quad (6.9)$$

$$С_В = 180,05+2500+625+1134,38+3500+4500 = 12439,43 \text{ грн.} \quad (6.10)$$

Адміністративні витрати визначаються із відомостей по аналізу повної собівартості виробу і в середньому можуть складати 140-200% від основної заробітної плати.

$$В_А = Зо*1,5 = 2500* 1,5 = 3750 \text{ грн.} \quad (6.11)$$

Зовнішні виробничі витрати, які мають зв'язок зі збутом виробів, складають 5-10% від виробничої собівартості:

$$В_{ЗВ} = С_В *0,1 = 12439,43 *0,1 = 1243,94 \text{ грн.} \quad (6.12)$$

Повна собівартість:

$$ПС = С_В + В_А + В_{ЗВ} = 12439,43 + 3750 + 1243,94 = 17433,37 \text{ грн.} \quad (6.13)$$

Таблиця 6,3 – Калькуляція собівартості пристрою

Стаття калькуляції	Витрати, грн
Матеріали та комплектуючі	180,05
Витрати на основну заробітну плату	2500
Додаткова заробітна плата	625
Відрахування на соціальні заходи	1134,38
Витрати на утримання і експлуатацію устаткування	3500
Загальновиробничі витрати	4500
Виробнича собівартість	**12439,43**
Адміністративні витрати	3750
Витрати на збут	1243,94
Повна собівартість пристрою	**17433,37**

Прибуток визначається виходячи з нормативу (показника) рентабельності виробництва продукції, який встановлює підприємство

$$R = \frac{П}{С} \cdot 100\%, \tag{6.14}$$

де R – рентабельність пристрою в розмірі 30% від його собівартості.

Відповідно оптова ціна пристрою визначається

$$Ц_{опт} = С + \frac{R \cdot С}{100}, \tag{6.15}$$

$$Ц_{опт} = 17433,37 + \frac{17433,37 \cdot 0,3}{100} = 17485,67 грн. \tag{6.16}$$

Визначення відпускної ціни пристрою. Відпускна ціна включає податок на додану вартість:

$$Ц_{розд} = Ц_{опт} \cdot 1,2, \tag{6.17}$$

де 20% - ПДВ.

$$Ц_{розд} = 17485,67 \cdot 1,2 = 20982,8 грн. \tag{6.18}$$

СПИСОК ЛІТЕРАТУРИ

1. comp.compression FAQ list entry #9: Compression of random data (WEB, Gilbert and others)

2. ZIP file format specification by PKWARE, Inc., chapter V, section J

3. Nelson, Mark (2006-06-20). "The Million Random Digit Challenge Revisited".

4. Compression of random data.

5. Craig, Patrick (2001-03-26). The $5000 Compression Challenge (en).

6. Динамическое адресно-векторное сжатие двоичных последовательностей Кулик, Игорь Анатольевич Харченко, С.Н.

7. https://compress.ru/article.aspx?id=10581

8. http://mf.grsu.by/UchProc/livak/po/comprsite/theory_classification_01.html

9. http://mf.grsu.by/UchProc/livak/po/comprsite/theory_classification_02.html

10. http://www.compression.ru/arctest/descript/methods.htm

11. https://habr.com/ru/post/251295/

12. https://naurok.com.ua/velichini-h-tipi-ta-vlastivosti-137942.html

13. https://disted.edu.vn.ua/courses/learn/940

14.https://studopedia.com.ua/1_186299_postiyni-ta-zminni-velichini-vlastivosti-funktsiy.html

15. І. Алексюк. Константа // Філософський енциклопедичний словник / В. І. Шинкарук (гол. редкол.) та ін. — Київ : Інститут філософії імені Григорія Сковороди НАН України : Абрис, 2002. — 742 с. — 1000 екз. — ББК 87я2. — ISBN 966-531-128-X.

16. Поняття сталої величини // Вища математика в прикладах і задачах / Клепко В.Ю., Голець В.Л.. — 2-ге видання. — К. : Центр учбової літератури, 2009. — С. 174. — 594 с.

17. https://books.ifmo.ru/file/pdf/574.pdf

18. https://forum.sources.ru/index.php?showtopic=304454

19. Змінні і сталі величини // Термінологічний словник-довідник з будівництва та архітектури / Р. А. Шмиг, В. М. Боярчук, І. М. Добрянський, В. М. Барабаш ; за заг. ред. Р. А. Шмига. — Львів, 2010. — С. 101. — ISBN 978-966-7407-83-4.

20. Додаток 1. Генерування біноміальних чисел на основі властивості вкладеності **.** Кулик І.А., *доцент*; Писаний О.В.*, студент гр. ЕС.м-91* Сумський державний університет, м. Суми, Україна

21. 05.13.11 – "Математическое и программное обеспечение вычислительных машин, комплексов и компьютерных сетей"

22.Л. Г. Мельник, А. И. Каринцева. Экономика предприятия. «Университетская книга» 2002г.

23. Экономика предприятия. Конспект лекций: Учебное пособие. - Сумы: ИТД "Университетская книга", 2002. -400 с.

24. http://pcbfab.ru

25. http://www.petrocom.ru

26. http://www.gaw.ru/html.cgi/components/adc/adc_7.htm

27.http://radio-hobby.org/modules/analog/%D0%BA%D1%80580%D0%B2%D0%B255%D0%B0

28. https://www.chipdip.ru/product/k155ir13

29. https://eandc.ru/catalog/detail.php?ID=9964

30. Електроніка і мікросхемотехніка : Навч. посіб. для підгот. фахівців із спец. «Енергетика с.-г. вир-ва» в аграр. вищ. навч. закл. III — IV рівнів акредитації / В. Т. Дмитрів, В. М. Шиманський ; Львівський держ. аграр. ун-т. — Л. : Афіша, 2004. — 176 с. — Бібліогр.: с. 171—172.

31. Твердотільна електроніка : підручник / О. В. Борисов, Ю. І. Якименко ; за ред. Ю. І. Якименка ; М-во освіти і науки України, Нац. техн. ун-т України "Київ. політехн. ін-т". – Київ : НТУУ "КПІ", 2015. – 484 с. : іл. – Бібліогр.: с. 476-477 (27 назв). – ISBN 978-966-622-687-0

Додаток 1

Генерування біноміальних чисел на основі властивості вкладеності

Кулик І.А., *доцент*; Писаний О.В., *студент гр. ЕС.м-91*
Сумський державний університет, м. Суми, Україна

Вкладеність біноміальних систем числення надає можливість розробляти широкий спектр методів та алгоритмів генерування різноманітних комбінаторних конфігурацій, що, у свою чергу, дозволяє реалізовувати на практиці адаптивні підходи до передачі або стиснення інформації на основі біноміальних чисел. Адаптивна зміна параметрів біноміальних систем числення і перехід к біноміальним числам меншої або більшої розмірності на основі властивості вкладеності може, наприклад забезпечити оптимальне значення співвідношення "вірність/швидкість передачі" у системах зв'язку.

Перехід до вкладених біноміальних систем числення можна здійснювати через двійкові біноміальні числа, які належать підкласам $A'_{l'}$ або $B'_{q'}$ вихідної (n',k')-біноміальної системи числення. В результаті виключення $x_{r'}=1$ з (n',k')-двійкових біноміальних чисел $a'_i[r',l']\in A'_{l'}$ отримуємо множину $E'[r'-1,k']$ рівноважних комбінацій $e'_i[r'-1,k']$ з k' одиницями і довжини $r'-1$. Далі, до отриманих $e'_i[r'-1,k']$ застосуємо систему кодотворчих біноміальних обмежень $\psi: E'[r'-1,k']\to\alpha''_{l'}[n'',k'']$, де $\psi=\psi_A\vee\psi_B$, ψ_A – система обмежень, яка формує підкласи A_l і в цілому клас A, ψ_B – система обмежень, яка формує підкласи B_q і в цілому клас B. В підсумку переходимо до двійкових (n'',k'')-біноміальних чисел $a''_j[r'',l'']$ и $b''_j[r'',q'']$ вкладеної (n'',k'')-біноміальної системи виду $\alpha''_{l'}[n'',k'']$.

Подібна послідовність кроків виконується і у випадку застосування $B'_{q'}$ і їх (n',k')-біноміальних чисел $b'_i[r',q']\in B'_{q'}$, у підсумку отримуючи біноміальні числа $a''_j[r'',l'']$ і $b''_j[r'',q'']$ вкладеної (n'',k'')-біноміальної системи числення виду $\beta''_{q'}[n'',k'']$.

Додаток 2

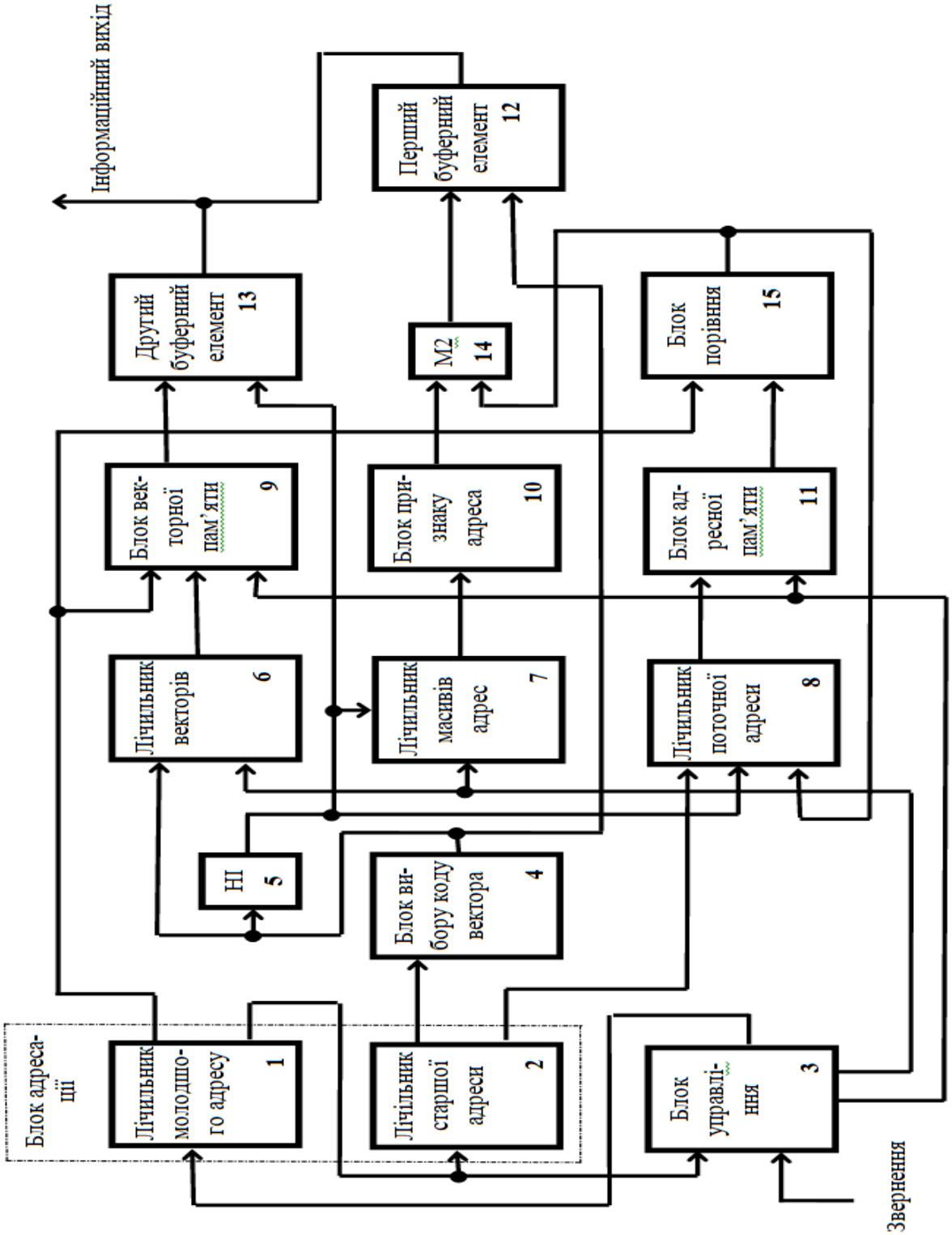


Рисунок 1 - Структурна схема запам'ятовуючого пристрою

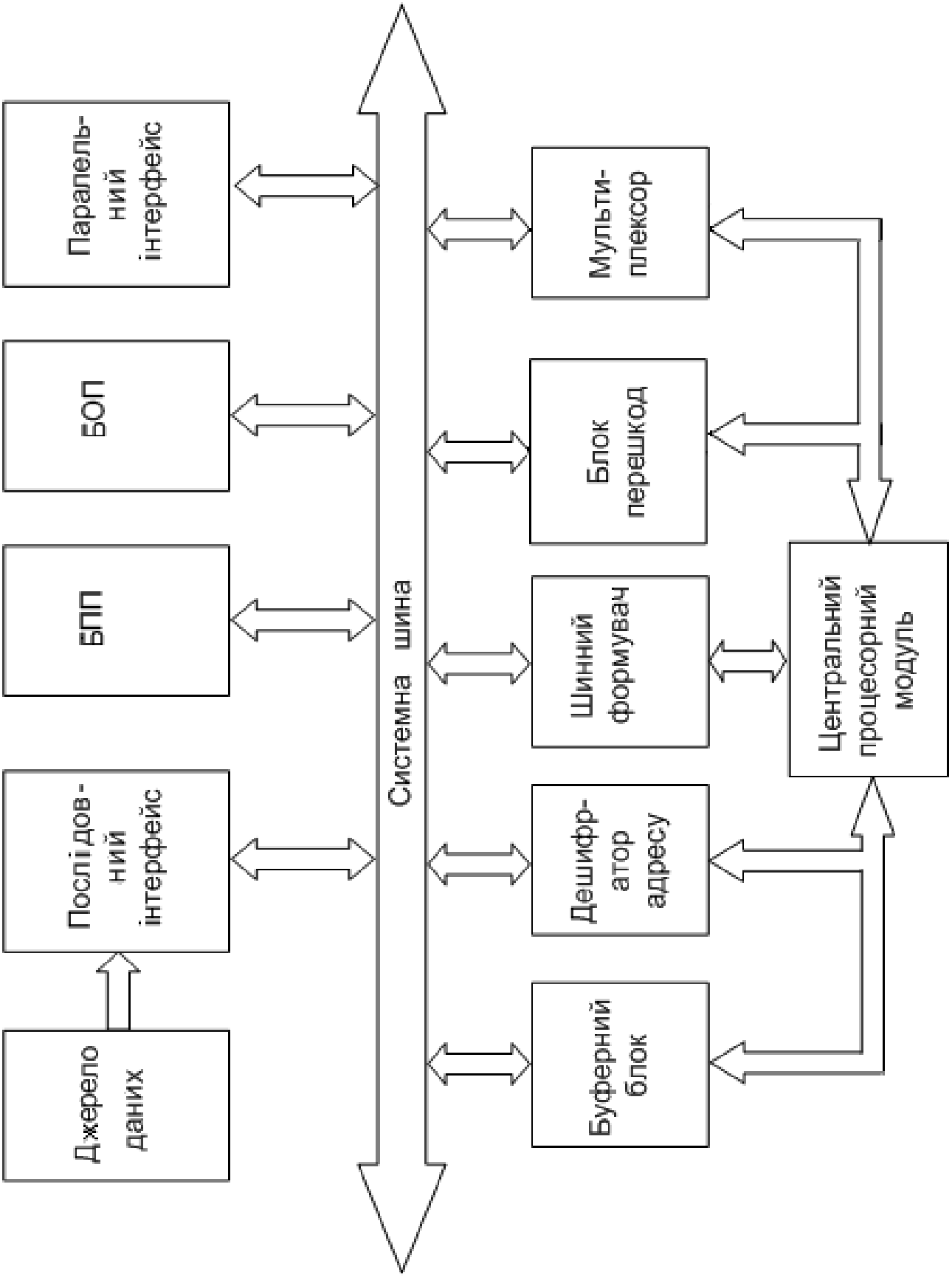


Рисунок 2 - Структурна схема системи

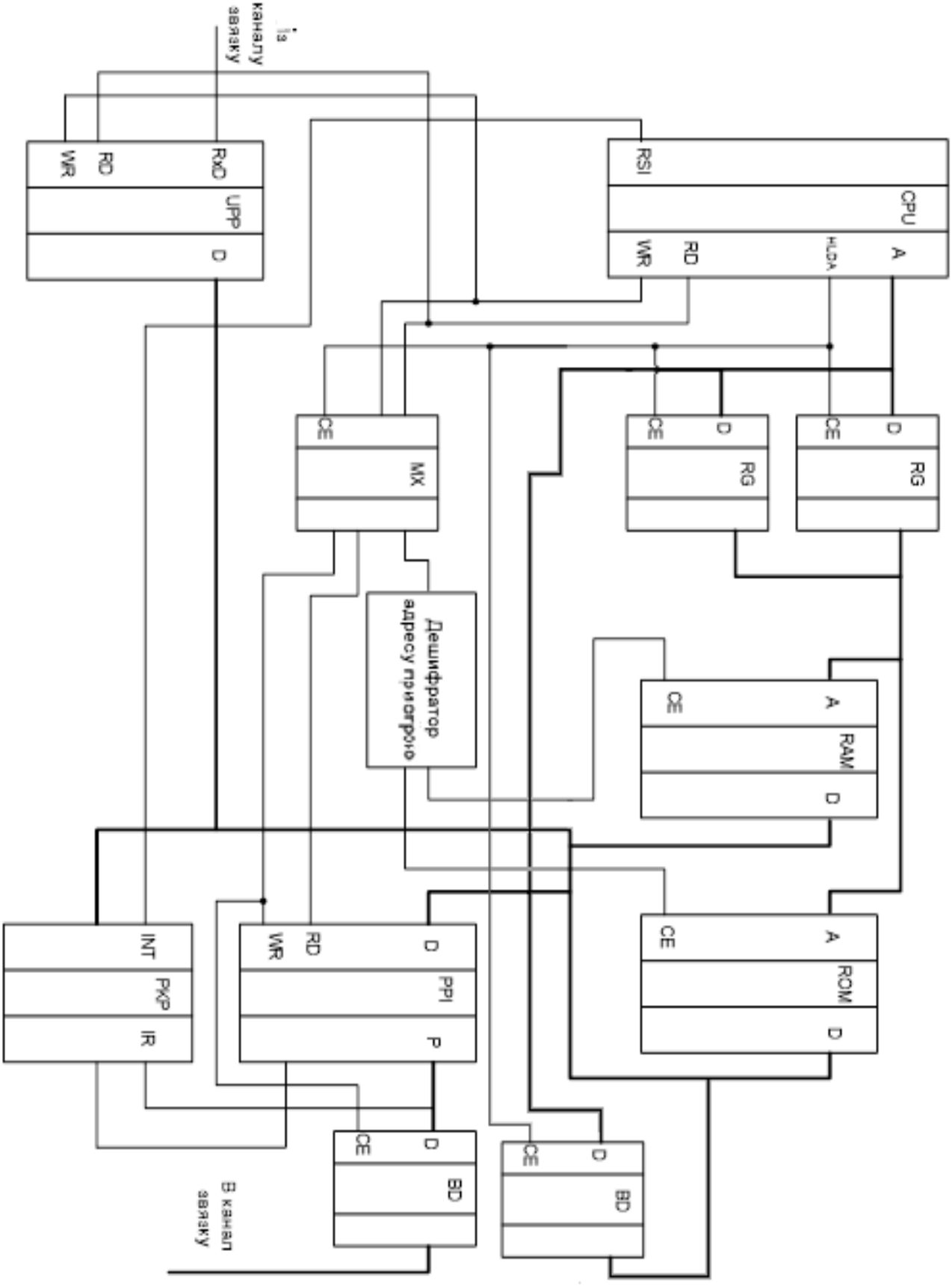


Рисунок 3 – Схема електрична функціональна системи

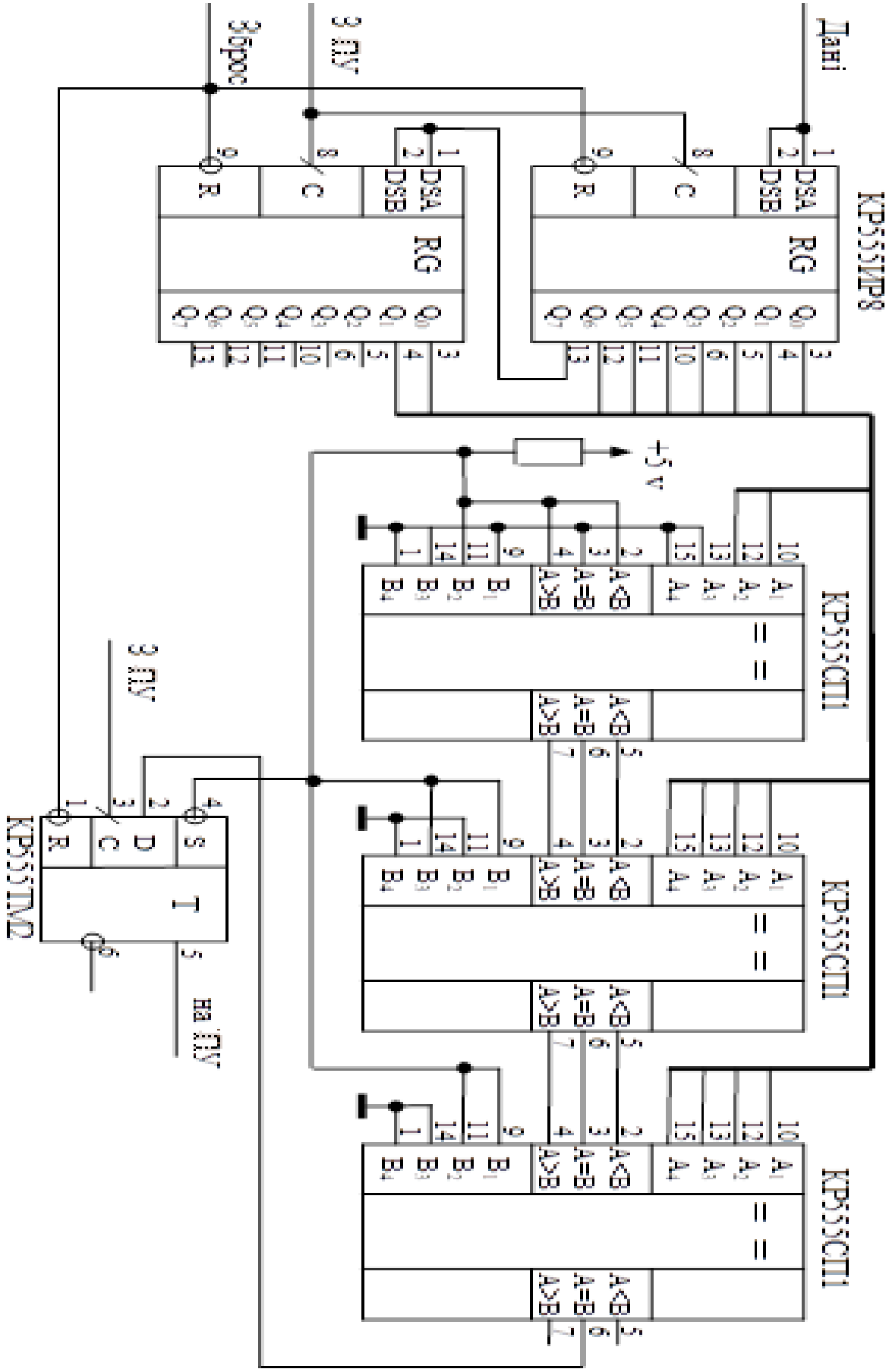


Рисунок 4 - Принципова схема блоку перевірки маркерної комбінації

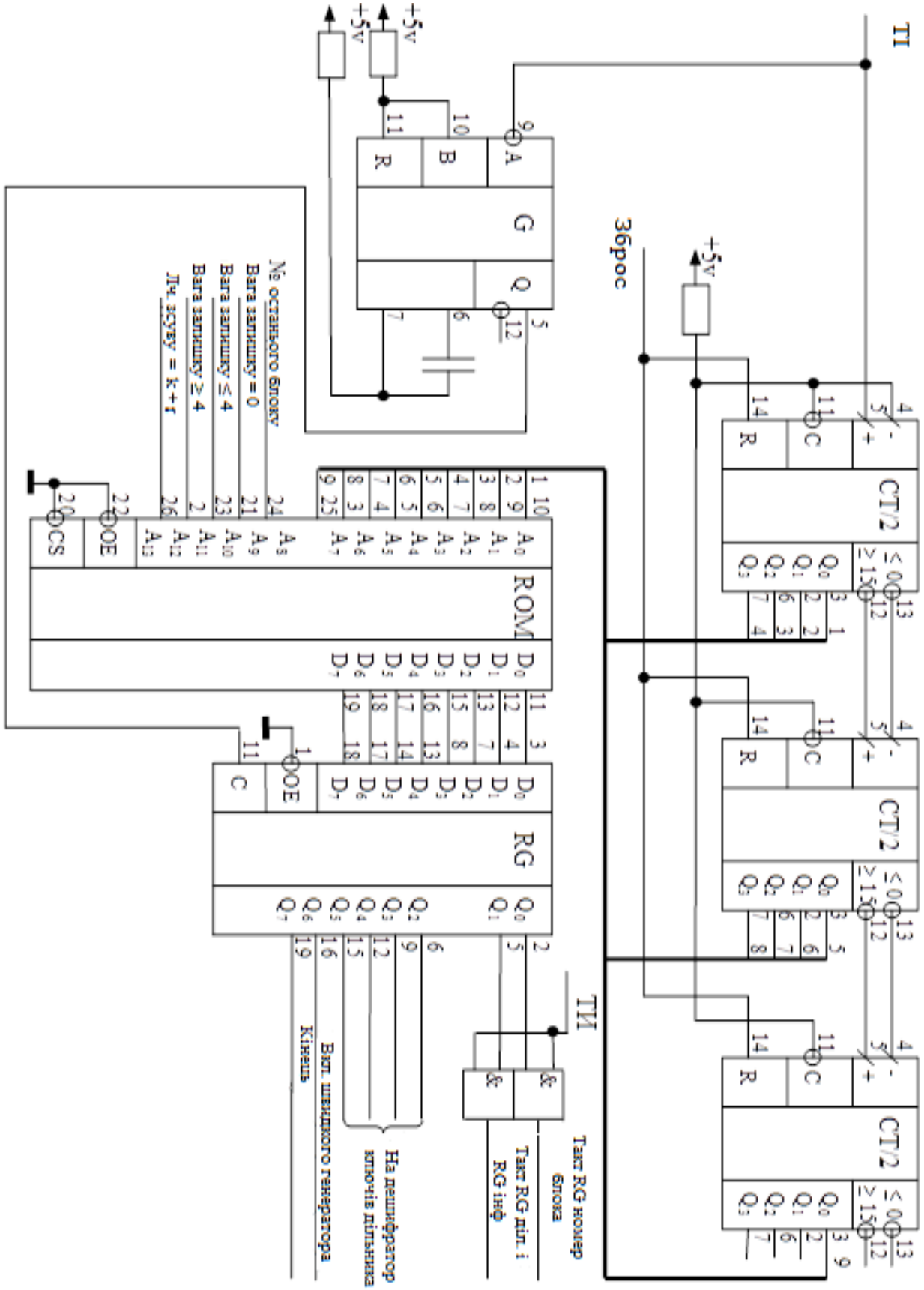


Рисунок 5 Принципова схема системи управління декодером