МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра електроніки і комп'ютерної техніки

«До захисту допущено»
Завідувач кафедри ЕКТ

_____ Анатолій ОПАНАСЮК
(підпис) (Ім’я та ПРІЗВИЩЕ)
____ _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня «магістр»

зі спеціальності 171 «Електроніка»

освітньо-професійної програми «Електронні системи та компоненти»

на тему:

ЕЛЕКТРОННА СИСТЕМА СТИСНЕННЯ ДАНИХ НА ОСНОВІ АРИФМЕТИЧНОГО КОДУВАННЯ

Здобувача групи ЕС.м-21 _____ Матьякубова Максима Сергійовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ _____
(підпис) (Ім’я та ПРІЗВИЩЕ)

Керівник, доцент, к.т.н., доцент Ігор КУЛИК _____
(підпис)

Консультант з техніко-економічної частини,
доцент, к.е.н., доцент Олександр МАЦЕНКО _____
(підпис)

Суми – 2023

СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Факультет електроніки та інформаційних технологій

Кафедра електроніки і комп'ютерної техніки

Напрям підготовки 171 «Електроніка»

Освітня програма Електронні системи та компоненти

ЗАТВЕРДЖУЮ

Зав. кафедрою Опанасюк А. С.

"___"_____ 2023 р.

З А В Д А Н Н Я

на кваліфікаційну роботу магістра

1. Тема роботи _____

затверджена наказом по університету "06" листопада 2023 р. № 1233-VI.

2. Термін здачі студентом завершеної роботи _____

3. Вихідні дані до роботи _____

4. Зміст розрахунково-пояснювальної записки (перелік питань, що належить розробити) 1) Огляд літератури та поставлення задачі роботи. 2) Науково-дослідна частина. 3) Розробка електронної системи з використанням отриманих результатів дослідження. 4) Техніко-економічна частина.

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

6. Консультанти з кваліфікаційної роботи

Розділи	Консультанти	Завдання видав	Завдання прийняв
Техніко-економічна частина	Маценко О. М.		

7. Дата видачі завдання ___

8. Керівник роботи _____

9. Завдання прийняв до виконання _____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів дипломного проекту	Термін виконання етапів роботи	Примітки
1	Огляд літератури та постановка завдання проектування	06.11.23 – 13.11.23	
2	Науково-дослідна частина	14.11.23 – 21.11.23	
3	Розробка алгоритму функціонування та структурної схеми електронної системи	22.11.23 – 29.11.23	
4	Розробка функціональної схеми електронної системи	30.11.23 – 04.12.23	
5	Розробка схеми електричної принципової електронної системи	05.12.23 – 12.12.23	
6	Техніко-економічна частина	13.12.23 – 14.12.23	
8	Оформлення пояснювальної записки	15.12.23 – 16.12.23	
9	Оформлення графічного матеріалу	17.12.23 – 18.12.23	
10	Представлення роботи керівнику і отримання відгуку	19.12.23	
11	Представлення роботи кафедрі для отримання рецензії	19.12.23	

Студент _____

Керівник роботи _____

«___» _____ 2023 р.

РЕФЕРАТ

Записка: 90 сторінок, 16 рисунків, 18 таблиць, 29 джерел.

Тема роботи: «Електронна система стиснення даних на основі арифметичного кодування».

Об'єктом розробки є електронна система стиснення на основі арифметичного кодування.

Мета роботи – розробка апаратної частини кодуючої електронної системи.

Пояснювальна записка складається з шости розділів, вступу, висновків та додатків з лістингом програми.

У першому розділі наданий огляд типів даних, моделей джерел інформації та методи оцінки їх характеристик.

У другому розділі проводиться аналіз властивостей поширених алгоритмів стискаючого кодування інформації та зберігання даних в електронних системах.

У третьому розділі наводиться алгоритм арифметичного стиснення необхідний для реалізації його в системі, а також схема алгоритму роботи проектованої електронної системи. Проводиться вибір структурної схеми та обґрунтування алгоритму її функціонування.

У четвертому розділі проводиться розробка функціональної схеми електронної системи і проведена характеристика всіх її блоків.

У п’ятому розділі провадиться розробка принципової схеми проектованої електронної системи. Розглянуто можливість виготовлення із застосуванням вітчизняної та імпортної елементної бази. Також описана програма, за якою електронна система працює.

У шостому розділі представлено схему розрахунку собівартості проектованої електронної системи при серійному її виробництві.

У висновках наводяться результати розробки електронної системи.

Ключові слова: стиснення даних, арифметичне кодування, алгоритми стиску, мікропроцесор, математичний співпроцесор.

ЗМІСТ

ВСТУП		5
1	ОГЛЯД ЛІТЕРАТУРИ ТА ПОСТАНОВКА ЗАВДАННЯ ПРОЕКТУВАННЯ	9
1.1	Загальні відомості	9
1.2	Типи даних і методи їх опису	11
1.3	Імовірнісні моделі джерел даних, що стискаються	12
1.4	Методи оцінки параметрів джерел інформації	15
1.5	Постановка задачі проектування	17
2	НАУКОВО-ДОСЛІДНА ЧАСТИНА. Алгоритми кодування інформації в системах передачі та зберігання даних	19
3	РОЗРОБКА АЛГОРИТМУ ФУНКЦІОНУВАННЯ СТРУКТУРНОЇ СХЕМИ ЕЛЕКТРОННОЇ СИСТЕМИ	29
3.1	Обґрунтування алгоритму функціонування	29
3.2	Алгоритм арифметичного кодування	32
3.3	Практична реалізація алгоритму кодування	35
3.4	Алгоритм роботи функціонування пристрою	36
3.5	Розробка структурної схеми електронної системи	38
4	РОЗРОБКА ФУНКЦІОНАЛЬНОЇ СХЕМИ ЕЛЕКТРОННОЇ СИСТЕМИ	44
5	РОЗРОБКА СХЕМИ ЕЛЕКТРИЧНОЇ ПРИНЦИПОВОЇ ЕЛЕКТРОННОЇ СИСТЕМИ	57
5.1	Вибір елементної бази	57
5.2	Розробка основних електронних вузлів	60
5.3	Розрахунок елементів та функціональних вузлів схеми	72
5.4	Розробка програмного забезпечення	75
6	ТЕХНІКО-ЕКОНОМІЧНА ЧАСТИНА	76

6.1 Розрахунок собівартості ……	76
6.2 Розрахунок повної собівартості електронної системи, що розробляється …….....	77
6.3 Розрахунок ціни електронної системи …..……...	82
ВИСНОВОК .....……	84
СПИСОК ЛІТЕРАТУРИ ……...	85
ДОДАТОК А. Лістинг програми ……..	87
ДОДАТОК Б. Перелік елементів …...……...	89

ВСТУП

Сучасне суспільство використовує цифровий вид подання інформації у багатьох сферах життєдіяльності. Своєрідним стрижнем, є створення ефективних автоматизованих систем управління (АСУ), навколо якого концентруються всі напрямки вдосконалення процесів управління технічними, соціальними та організаційно-економічними об'єктами.

На основі достовірної інформації про процеси, що протікають, необхідно ефективно керувати рядом таких об'єктів і підсистем: адміністративні, соціально-економічні та будь-якого іншого призначення, АСУ технологічними процесами, інформаційно-керуючі та інформаційно-вимірювальні системи.

За підсумками мереж електронно-обчислювальних машин (ЕОМ), будуються інформаційні підсистеми, є структурними елементами сучасних (комп'ютеризованих) систем управління організаційно-економічними об'єктами (процесами). Широко застосовуються мережі програмованих (промислових) контролерів у сучасних системах управління складними технологічними об'єктами (що включають локальні системи управління технологічними машинами), є різновидом управляючих ЕОМ. Інтенсивно розвивається напрямом є застосування Web-технологій управління в організаційно-економічних і технічних системах з використанням будь-яких відомих засобів зв'язку, в тому числі космічної [1]. Невід'ємною закономірністю функціонування систем управління будь-якого рівня та призначення є обмін інформацією між джерелами та споживачами. Істотно впливають на ефективність процесів керування, інформаційні підсистеми, що здійснюють передачу, обробку та зберігання інформації, наприклад, у сферах керування технологічними процесами на підприємствах, керування транспортними засобами, банківської діяльності та ін.

Розвиток перелічених інформаційних мереж та систем призвело до суттєвого зростання інформаційних потоків між територіально розташованими джерелами та одержувачами повідомлень. Тому в наш час існують величезні потреби у таких видах сервісів як передача відеозображень, факсимільний зв'язок, обмін базами даних,

відеоконференцзв'язок, телефонія, вимірювальна інформація в реальному та прискореному масштабах часу тощо.

Застосовуються різні методи та засоби, для підвищення ефективності використання інформаційно-обчислювальних та комунікаційних ресурсів зазначених систем. Тому серед них відіграють важливу роль методи скорочення надмірності, що забезпечують стиснення обсягів інформації, що запам'ятовується або передається [2]. Це дозволяє значно розвантажити канали зв'язку та системи обробки та зберігання даних за рахунок виключення надлишкових повідомлень, що еквівалентно підвищенню пропускних здібностей інформаційних систем або збільшення ємності пристроїв.

Серед систем збору даних широкого поширення набули системи на основі персонального комп'ютера (ПК). У силу їхньої універсальності, такі системи називають гнучкими системами збору даних, і найважливішим елементом таких систем є комп'ютер [3]. У системі на базі комп'ютера користувачі можуть збирати дані віддалено або безпосередньо через цей комп'ютер - від об'єднаних в мережу програмованих логічних контролерів або розподіленої системи управління.

Дуже часто в таких системах ефективніше використовувати систему збору даних на базі ПК, необхідно аналізувати процес у реальному часі та з метою оптимізації впливати на його змінні. Тому користувач може завантажити в систему управління технологічним процесом новий набір параметрів за допомогою засобів взаємодії процесів, таких як TCP/IP або DDE [4].

Часто необхідне зберігання та аналіз достатньо об'ємної кількості значень, отриманих із датчиків. Спеціалізовані системи зберігання та збору даних, звані логічними аналізаторами, призначені для збору, як правило, дискретних сигналів, і дозволяють проводити збір даних по десятках каналів одночасно з можливістю запам'ятовування до кількох десятків тисяч множин значень, отриманих в один момент часу.

Враховуючи частоту опитування датчиків, обсяг пам'яті необхідний зберігання зібраних даних може досягати десятків мегабіт в секунду. Зростання обсягів даних та розвиток сервісів інформаційних цифрових ресурсів також накладають подальші обмеження на роботу алгоритмів.

Передача та зберігання таких обсягів даних у реальному масштабі часу може створити проблему, яка вирішиться завдяки використанню систем стиснення даних.

Великий обсяг інформації вимагає великої протяжності та пропускної спроможності каналів передачі даних. На даний момент розвитку інформаційної інфраструктури, існуючі канали не справляються з необхідним трафіком. Отже, завдання стиснення даних є актуальним у багатьох додатках обробки та передачі [5].

У деяких випадках не потрібно точного відновлення інформації та допускається використовувати алгоритми, що реалізують стиск із втратами, яке, на відміну від стиснення без втрат, зазвичай простіше реалізується і забезпечує більш високий ступінь архівації. Такі методи зазвичай застосовуються для стиснення відео, звуку, певного типу зображень, де особливості психофізичного сприйняття даних людиною відіграють вирішальну роль.

Для управління технологічним процесом, автоматизованої системи аналізу даних та інших подібних завданнях навіть незначні втрати інформації при стисканні неприпустимі.

У зв'язку з вищевикладеним мають велику наукову та практичну значущість дослідження та розробки в галузі методів та засобів ефективного кодування.

Всі стандарти стиснення даних тією чи іншою мірою використовують алгоритми ентропійного кодування. Під ентропійним кодуванням розуміється кодування, у якому ентропія стислих даних збігається з ентропією вихідного джерела і за стислими даними можна повністю відновити вихідну інформацію. Такий підхід називають стисненням без втрат. Це стиснення, наприклад, застосовується у дистрибутивах програмних продуктів. Це найбільш вивчена сфера стиснення даних. Вона є дуже важливою, оскільки більшість методів компресії різних типів цифрової інформації часто використовують на певних стадіях алгоритми стиснення без втрат. Можна сміливо сказати, що компресія без втрат є екстремальним випадком стиску, у якому ентропія даних залишається незмінною.

Одним із найсучасніших і передових ентропійних алгоритмів стиснення є сімейство алгоритмів арифметичного стиснення. Вони дозволяють досягти

коефіцієнта стиснення, найближчого до максимального. Ціною за високі показники стиснення є трудомісткість алгоритмів, і, отже, їхня висока вимогливість до обчислювальних ресурсів.

Арифметичне кодування використовується для стиснення контекстних послідовностей даних в останніх стандартах стиснення зображень JPEG2000, H.264 (MPEG-4 p.10, AVC) і т. д. [5].

Актуальність теми, обумовлена вищезазначеною необхідністю підвищення ефективності передачі та зберігання даних в інформаційних системах.

Завданням роботи є розробка структурного методу роботи кодера арифметичного кодування та синтез алгоритмів його реалізуючих, які б задовольняли вимогам щодо точності та швидкодії перетворення інформації в АСУ.

Метою роботи є підвищення ефективності використання інформаційних систем передачі та зберігання даних в АСУ за збереження цілісності оброблюваної інформації.

Об'єктом вивчення у цій роботі є системи стиснення інформаційних завдань АСУ.

Предметом вивчення є метод та алгоритми структурного стиснення даних для інформаційних систем.

1 ОГЛЯД ЛІТЕРАТУРИ ТА ПОСТАНОВКА ЗАВДАННЯ ПРОЕКТУВАННЯ

1.1 Загальні відомості

Існує кілька різних підходів до проблеми стиснення інформації. Одні мають дуже складну теоретичну математичну базу, інші ґрунтуються на властивостях інформаційного потоку та алгоритмічно досить прості [2, 5]. Будь-який підхід і алгоритм, що реалізує стиснення або компресію даних, призначений для зниження обсягу вихідного потоку інформації в бітах за допомогою її оборотного або незворотного перетворення.

Усі алгоритми стиску оперують вхідним потоком інформації, мінімальною одиницею якої є біт. Метою процесу стиснення, як правило, є отримання компактнішого вихідного потоку інформаційних одиниць з деякого спочатку некомпактного вхідного потоку за допомогою деякого їх перетворення. Це є форма кодування [6].

Основними технічними характеристиками процесів стиснення та результатів їхньої роботи є [2, 5]:

– ступінь стиснення - відношення обсягів вихідного та результуючого потоків;

– швидкість стиснення - час, що витрачається на стиск деякого обсягу інформації вхідного потоку, до отримання еквівалентного вихідного потоку;

– якість стиснення – величина, що показує наскільки сильно упакований вихідний потік, за допомогою застосування до нього повторного стиснення цього ж або іншого алгоритму.

Ступінь стиснення показує у скільки разів обсяг вихідного потоку менше обсягу вхідного. Однак досить поширеним параметром є величина зворотна ступеня стиснення та звана коефіцієнтом стиснення:

$$K = \frac{L_{out}}{L_{in}}, \tag{1.1}$$

де K – коефіцієнт стиснення;

L_{out} – обсяг стиснутих даних;

L_{in} – обсяг вихідних даних

Очевидно, що ефективність стиснення методу тим вища, що менше значення коефіцієнта стиснення.

Іноді під коефіцієнтом стиснення мають на увазі значення відносної ентропії, що характеризує величину надмірності поточного подання інформації. Цей параметр більшою мірою характеризує теоретичну можливість стиснення даних, а не фактичні показники компактності їх уявлення.

І ступінь стиснення, і коефіцієнт стиснення іноді виражають у відсотках.

Процес кодування множини $A = \{a_i : | a_i | = n,\ i = 1...h\}$ вихідних даних представляє собою взаємно однозначне відображення $f : A \to V$, де $V = \{v_i : i = 1...h\}$ – результуюча множина двійкових слів $v_i = f(a_i)$.

Усі існуючі методи кодування можна розбити на три класи [2; 5; 7]:

1) фрагментам вихідної фіксованої довжини ставиться у відповідність кодові слова різної довжини;

2) фрагментам змінної довжини ставляться у відповідність кодові слова фіксованої довжини;

3) змінної довжини – змінної довжини. Суперпозиція кодів типу 2 та 1.

Для кодування вхідного потоку необхідна модель джерела інформації, що генерує цей потік [16].

Процес визначення типу і параметрів моделі може бути або явним, або прихованим. Імовірності елементів можуть використовуватись у методі як явним, так і неявним чином. Але завжди стиск досягається за рахунок усунення надмірності у поданні інформації [2; 5; 7].

Завданням процесу моделювання є оцінка ймовірностей кожного символу. З цих ймовірностей можна обчислити ентропія джерела. Важливо відзначити, що ентропія є властивістю моделі. Найкраща середня довжина коду досягається моделями, у яких оцінки ймовірності якомога точніші. Точність оцінок залежить від широти використання контекстуальних знань про джерело.

Модель по суті є набір ймовірностей розподілу по одному на кожен контекст, на підставі якого символ може бути закодований. Контексти називають класами умов, так як вони визначають оцінки ймовірності. Нетривіальна модель може містити тисячі умов класу.

1.2 Типи даних і методи їх опису

Оброблювані дані можуть бути розділені на два класи: дані певного типу або типізовані і дані, тип яких невідомий або важко віднести їх до будь-якого з існуючих.

Типізовані дані мають, як правило, деякі властивості, що визначають їх тип. До таких властивостей можна віднести характер взаємозв'язків між елементами даних. Залежно від того, що є елементами вихідної множини даних, мінімальною одиницею даних, якою оперує модель, може бути 1 або декілька біт (послідовність біт, байт, послідовність байт).

Так, наприклад, графічні дані, залежно від кількості кольорів у моделі, можуть оперувати мінімальною одиницею даних в 1 біт для бінарних зображень і аж до 8 байт для повнокольорових. Крім того, вони, як правило, мають виражену взаємозалежність між 14 елементами сусідніх рядків у зображенні. Залежно від типу графічних даних може бути взаємозалежність і між елементами одного рядка.

У текстових даних мінімальним елементом є 1 байт при поданні символів у кодуванні ANSI, ASCII, KOI-8 або 2 байти для UNICODE. Взаємозв'язок може виявлятися як на рівні букв, так і на рівні слів та речень.

Виконувані файли складаються з байтів і взаємозалежності можуть виявлятися між ними в залежності від типу програми і застосовуваного компілятора.

Результати різних експериментів або показання безлічі датчиків можуть не мати взаємозалежностей взагалі через незалежність процесів, які вони аналізують або описують. Наприклад, тривіальний випадок з гральними кістками, монетою або багатоканальна система збору дискретних даних.

Залежно від типу даних і відповідних цьому типу взаємозв'язків між елементами в повідомленнях, виділяють кілька моделей для математичного опису джерел інформації, що генерують подібні дані.

1.3 Імовірнісні моделі джерел даних, що стискаються

Розглянемо існуючі ймовірні моделі джерел стисливих даних. Незважаючи на різноманітність різних типів даних в основу їх аналізу покладено всього дві основні ймовірні моделі: модель Бернуллі і модель Маркова [5; 7], та їх різновиди.

Модель Бернуллі. Найпростіший є модель Бернуллі, коли між повідомленнями і символами вони немає взаємозв'язків, тобто. ймовірність появи наступного символу залежить від цього, які символи зустрілися проти нього:

$$X = \{x_i, i = 1... \mid X \mid\}, \tag{1.2}$$

$$P(X) = \{p(x_i), i = 1... \mid X \mid\}, \tag{1.3}$$

де X – алфавіт джерела інформації; $P(X)$ - множина ймовірностей символів $p(x_i)$.

Взаємозалежність між елементами множини $P(X)$ в даній моделі відсутня.

Саме такого джерела інформації застосовна формула Шеннона [12] визначення ентропії:

$$H(X) = -\sum_{i=1}^{|X|} p(x_i) \log_2 p(x_i), \tag{1.4}$$

де $H(X)$ – ентропія джерела інформації.

Якщо для будь-якого i вірно $p(x_i) \le p(x_{i+1})$ або $p(x_i) \ge p(x_{i+1})$, то таке джерело є ще і монотонним.

Прикладом джерел Бернуллі можуть бути безліч значень, отриманих в один момент часу, отримані з дискретних датчиків, рядки зображень, одержуваних в результаті деяких фізичних експериментів, значення випадкових величин. У ряді випадків, навіть за наявності деяких взаємозв'язків між повідомленнями, джерело інформації можна у вигляді безлічі бернуллієвських джерел, кожному з яких відповідає конкретний контекст.

Дана модель в багатьох випадках більш переважна через простоту [7].

Контекстно-обмежена модель. Більш складний шлях обчислення ймовірностей символів лежить через визначення їхньої залежності від попереднього символу:

$$P(X) = \{p(x_{ij}), i = 1...|X|, j = 1...|X|\}. \quad (1.5)$$

На відміну від моделі Бернуллі в даному випадку множина $P(X)$ є двовимірним масивом.

Наприклад, в англійському алфавіті ймовірність слідування за літерою літери становить понад 99%, а без урахування попереднього символу всього 2,4% [18]. З урахуванням контексту символ кодується 0,0014 біта та 5,38 біта в іншому випадку. Імовірність появи літери становить 31%, якщо поточним символом є , і 4,2%, якщо він невідомий, тому в першому випадку вона може бути закодована 1,69 біта, а в другому 4,6 біта. При використанні інформації про попередні символи, ентропія становить 3,6 біта/символ у порівнянні з 4,5 біта/символ у простих моделях.

Логічно припустити, що модель більшою мірою завжди досягає кращого стиску. Необхідно вміти оцінювати ймовірності щодо контексту будь-якої довжини, коли кількість ситуацій наростає експоненційно до ступеня моделі. Для обробки великих зразків тексту потрібний великий обсяг пам'яті. Розмірність множини $P(X)$ буде збільшуватися із зростанням розміру контексту. Для моделі R-го порядку вона буде рівна R+1.

Принцип контекстно-обмеженого моделювання може бути використаний для будь-якого алфавіту. 8-бітовий алфавіт ASCII зазвичай добре працює з максимальною довжиною контексту на кілька символів. Якщо звернення відбувається до біт, можна застосовувати двійковий алфавіт (наприклад, при стисканні зображень) [2; 5; 7].

Моделі 0-го порядку є найпростішою формою контекстно-обмеженого моделювання (модель Бернуллі).

Модель Маркова. Імовірнісні моделі з кінцевим числом станів ґрунтуються на кінцевих автоматах. Вони мають множину станів $S(i)$ і імовірностей переходу $P(i, j)$ моделі з стану i в стан j. При цьому кожен перехід позначається унікальним символом. Таким чином, через послідовність цих символів будь-який вихідний текст задає унікальний шлях моделі (якщо він існує). Часто такі моделі називають моделями Маркова, хоча іноді цей термін неточно використовується позначення контекстно-обмежених моделей.

Моделі з кінцевою кількістю станів здатні імітувати контекстно-обмежені моделі. Наприклад, модель 0-го ступеня простого англійського тексту має один стан із 27 переходами назад до цього стану: 26 для літер та 1 для пробілу. Модель 1-го ступеня має 27 станів, кожен із 27 переходами. Модель *n*-ої ступеня має 27^n станів з 27 переходами для кожного з них.

У таких моделях збір статистики появи кожного символу проводиться з урахуванням деякої кількості попередніх символів (у Марківському джерелі першого порядку ймовірність появи символу залежить тільки від одного останнього символу, другого - від двох і т. д.). Марківські моделі можуть давати більш точну картину джерела, проте кількість станів у них більша, відповідно більшим буде обсяг таблиць частот, що зберігаються.

Моделі з кінцевим числом станів здатні представляти складніші в порівнянні з контекстно-обмеженими моделями структури. Найпростіший приклад - модель станів для рядка, в якому символ "*a*" завжди зустрічається двічі поспіль. Контекстуальна модель цього уявити неспроможна, оскільки з оцінки ймовірності символу, наступного за послідовністю букв "*a*", мають бути розглянуті довільно великі контексти.

Нерідко генеровані джерелом слова є деякими комбінаторними комбінаціями, тобто. безліч дозволених комбінацій первинного алфавіту описується певним комбінаторним співвідношенням. Таке джерело можна описати моделлю з кінцевим числом станів. У цьому ймовірності всіх дозволених слів однакові.

Комбінаторна модель зручна для кодування. Існують методи комбінаторного кодування, які є оптимальними для таких джерел.

Крім здійснення кращого стиснення, моделі з кінцевим числом станів скоріш у принципі. Поточний стан може замінювати ймовірність розподілу для кодування, а наступний стан просто визначається дугою переходу (при описі автомата за допомогою граф-схеми). На практиці, стани можуть бути виконані у вигляді зв'язкового списку, що вимагає не набагато більше обчислень.

На жаль, задовільні методи створення хороших моделей з кінцевим числом станів виходячи з зразків рядків ще знайдено. Один підхід полягає у перегляді всіх моделей можливих для даного числа станів та визначення

найкращої з них. Ця модель зростає експоненційно до кількості станів і годиться тільки для невеликих масивів [8].

Єдиний із наведених у літературі працюючий метод досить швидко, щоб його можна було застосовувати на практиці, метод моделювання з кінцевим числом станів називається динамічним стисненням Маркова (DMC) [9]. DMC адаптивно працює, починаючи з простої початкової моделі 0-порядку і поступово розширює початкову модель у процесі стиснення (додає при необхідності нових станів). На жаль, виявляється, що вибір евристики та початкової моделі забезпечує створюваної моделі контекстно-обмежений характер [9], через що можливості моделі з кінцевим числом станів не використовуються на повну силу.

У порівнянні з іншими методами стиснення DMC зазвичай здійснює побітове введення, але принципової неможливості символьно-орієнтованої версії не існує. Проте, практично такі моделі найчастіше вимагають багато оперативної пам'яті. Моделі з побітовим введенням не мають проблем з пошуком наступного стану, оскільки в залежності від значення наступного біта існують лише два переходи з одного стану до іншого. Ще важливо, що модель з бітами на кожному кроці здійснює оцінку у формі двох ймовірностей $p(0)$ і $p(1)$.

1.4 Методи оцінки параметрів джерел інформації

Існує кілька методів оцінки параметрів джерел інформації. Методи стиснення можуть будувати модель джерела інформації адаптивно у міру обробки потоку даних або використовувати фіксовану модель, створену на основі апріорних уявлень про природу типизованих даних, що потребують стиснення.

У порядку функціональної відповідності декодер повинен мати доступ до тієї ж моделі, що і кодер. Для цього є три способи оцінки параметрів джерела інформації: статичний, напівадаптивний і адаптивний.

Статичний метод. Статичний метод використовує для всіх даних одну й ту саму модель. Вона задається при запуску кодера, можливо на підставі раніше отриманих та проаналізованих зразків очікуваних даних. Така сама

копія моделі зберігається разом із декодером. Недолік полягає в тому, що схема даватиме необмежено поганий стиск щоразу, коли кодований дані не вписується в обрану модель, тому статичний метод використовують тільки тоді, коли важливі насамперед швидкість та простота реалізації.

Статистичний кодер вимагає оцінки розподілу ймовірності для кожного кодованого символу. Найпростіше присвоїти кожному символу постійну ймовірність, незалежно від його положення в тексті, що створює просту контекстуально-вільну модель. Наприклад, в англійській мові ймовірність символів "*пробел*", "*e*", "*t*" и "*k*" звичайно складає 18%, 10%, 8%, 0,5% выдповыдно. Отже, у цій моделі дані літери можна закодувати оптимально 2,47, 3,32, 3,64 та 7,62 біт за допомогою арифметичного кодування. У такій моделі кожен символ буде представлений у середньому 4,5 бітами. Це значення ентропії моделі, заснованої на ймовірності розподілу літер в англійському тексті. Ця проста статична контекстуально-вільна модель часто використовується разом із кодуванням Хафмана [10].

Полуадаптивний метод. Напівадаптивний метод вирішує цю проблему, використовуючи для кожного тексту свою модель, яка будується ще до стиснення на підставі результатів попереднього перегляду типу даних і зібраної статистики. Перед тим, як завершено формування вихідного потоку, параметри моделі мають бути передані декодеру.

При цьому потрібно два проходи за даними - один для перегляду та збору статистичної інформації, другий - для кодування, що дещо обмежує сферу застосування таких алгоритмів, тому, таким чином, унеможливлюється однопрохідне кодування "на льоту", що застосовується в телекомунікаційних системах , де обсяг даних, під час не відомий, які повторна передача чи розбір може зайняти невиправдано багато часу). Незважаючи на додаткові витрати на передачу моделі, ця стратегія в загальному випадку окупається завдяки кращій відповідності моделі тексту. Цей метод відомий як статичне кодування Хафмана [10].

Адаптивний метод. Адаптивний або динамічний метод (adaptive coder method) позбавлений пов'язаних із передачею параметрів моделі витрат. Його загальний принцип у тому, щоб змінювати схему кодування залежно від характеру змін вхідного потоку. Спочатку і кодер, і декодер надають собі модель рівноймовірністного джерела. Кодер використовує цю модель для

стиснення чергового символу, а декодер для розгортання. Потім вони обоє змінюють свої параметри моделі однаковим чином (наприклад, нарощуючи ймовірність символу, що розглядається). Наступний символ кодується та декодується на основі нової моделі, а потім знову змінює модель. Кодування продовжується аналогічним декодуванням чином, яке підтримує ідентичну модель за рахунок застосування такого ж алгоритму її зміни.

Імовірності можна оцінювати адаптивно за допомогою масиву лічильників по одному на кожен символ. Спочатку всі вони встановлюються в 1 (щоб уникнути проблеми нульової ймовірності), а після кодування символу значення відповідного лічильника збільшується на одиницю. Аналогічно при декодуванні відповідного символу декодер збільшує значення лічильника. Імовірність кожного символу визначається його відносною частотою. Ця проста адаптивна модель незмінно застосовується разом із кодуванням Хаффмана [10; 11].

Адаптивне кодування може дати більший ступінь стиснення, порівняно зі статичним, оскільки повніше враховуються зміни частот вхідного потоку. Такий підхід має однопрохідний алгоритм і не вимагає збереження інформації про використане кодування у явному вигляді. Цей метод відомий як динамічне кодування Хаффмана [11].

1.5 Постановка задачі проектування

Виходячи з поставленого завдання, необхідно спроектувати пристрій, який виконуватиме такі функції:

– формування сигналів-запитів із зовнішньою шиною управління та каналом зв'язку;

– перевірка наявності даних із зовнішньої шини управління;

– прийом від зовнішньої шини керування даних для подальшого стиснення;

– зберігання отриманих даних;

– проведення над отриманими даними математичних операцій, згідно з алгоритмом арифметичного кодування, закладеним у пам'яті контролера;

– зберігання одержаного результату обчислення закодованого повідомлення;

– підготовка до відправлення даних в канал зв'язку;

– передача до каналу зв'язку закодованого повідомлення.

Ця електронна система повинен мати такі технічні характеристики:

– мікропроцесорний комплект із центральним процесором та математичним співпроцесором з наявністю 16-розрядної шиною даних;

– наявність у мікропроцесорному комплекті пристроїв, здатних виконувати різноманітні операції вводу-виводу, зберігання та обробки цифрової інформації;

– інтерфейс підключення до зовнішньої шини управління – неповний перехідник на ISA шину (8-розрядний односпрямований порт введення даних; 1 вхід переривання);

– інтерфейс підключення до зовнішнього каналу зв'язку – один інтерфейс RS-232, що не комутується.

2 НАУКОВО-ДОСЛІДНА ЧАСТИНА.

Алгоритми кодування інформації в системах передачі та зберігання даних

Розглянемо існуючі алгоритми кодування та декодування інформації. У цьому можна використовувати таку класифікацію.

Алгоритми кодування можуть працювати в реальному масштабі часу і в потоці, якщо в них немає накопичення інформації і вони формують вихідний текст у міру надходження вхідного повідомлення (а не після).

Алгоритм зветься:

– однопрохідним, якщо для отримання стисненого тексту йому необхідно лише один раз "переглянути" вихідний текст;

– блоковим, якщо окремий вихідний код відповідає більш ніж одному вхідному символу.

Для роботи в реальному масштабі часу, очевидно, потрібно, щоб алгоритм був однопрохідним.

У [2; 5; 7] дана наступна класифікація методів архівації:

– статистичний, якщо передбачає відповідність вхідного потоку певної моделі сигналу та здійснює стиск на основі зібраної про текст статистичної інформації;

– інкрементальний, що здійснює стиск шляхом кодування відмінностей у послідовних записах;

– макро або текстової підстановки, що виконує стиснення шляхом пошуку рядків, що збігаються, і заміни їх на більш короткі коди.

***Кодування довжин серій* (*RLE-кодування*).** Алгоритм RLE (упаковка, кодування довжин серій, групове кодування), запропонований в [2; 5; 7], є найшвидшим, простим і зрозумілим алгоритмом стиснення даних і при цьому іноді виявляється дуже ефективним. Саме такий алгоритм використовується для стиснення зображень у файлах PCX. Він полягає в наступному: будь-якій послідовності вхідних символів, що повторюються, ставиться у відповідність набір з трьох вихідних символів: перший – байт префікса, що говорить про те, що зустрілася вхідна повторювана послідовність, другий - байт, що визначає довжину вхідної послідовності, третій – сам вхідний символ – $< prefix, length, symbol >$.

Якщо одиночні символи зустрічаються досить рідко, хорошою може бути модифікація алгоритму RLE взагалі без префікса, тільки $< length, symbol >$.

Можливий варіант алгоритму, коли замість довжини length кодується позиція щодо початку тексту distance першого символу, що відрізняється від попереднього.

Таблиця 2.1 – Параметри RLE-кодування

Алгоритм	Ступінь стиску	Швидкість	Пам’ять	Стиск без втрат	Проходи	РП	ЗН
RLE	2-3	10	10	Да	1	Ні	Можл.

Тут і далі прийняті такі позначення: РП – розповсюдження помилки, ЗН – зростання надмірності. Ступінь стиснення, швидкість та оперативна пам'ять будемо оцінювати за десятибальною системою. Чим більша величина, тим краще зазначений параметр (вища швидкість роботи, вище ступінь стиснення і менша споживана пам'ять).

Робота: у реальному масштабі часу та в потоці.

Основне застосування: РСГ, стиснення зображень.

Термін (у термінології PkZip): Packing.

Модель: Маркова 1 порядку.

Класифікація: блоковий, інкрементальний чи макро.

Ентропійне кодування із відомими характеристиками. Унарне кодування. Один з очевидних способів стиснення інформації, особливо текстової, полягає в тому, що можливість використання в тексті різних символів дуже різна. Зокрема, у російській мові найчастіше використовуються літери "*о*", "*а*", редко - "*ь*", "*щ*" і т.д. Тому пропонується використовувати для частих символів більш короткий код, для рідкісних - більш довгий. Труднощі у тому, що коди виходять змінної довжини й у загальному потоці вихідних даних важко знайти кінець одного коду і початок іншого.

Проте є коди [2; 5; 7], котрим завдання визначення довжини легко вирішується (так звані нерівномірні коди з властивістю однозначного декодування). Цей код дуже простий, але не ефективний. Однак, він може стати в нагоді для побудови кодів цілих змінної довжини (VLI codes). Саме ці коди успішно застосовуються в архіваторах ARJ і LHA. VLI-коди складаються

з двох частин [12]: спочатку слідує унарний код, що визначає групу чисел певної довжини, а потім саме число зазначеної довжини. Таким чином, унарний код виконує роль префікса, який вказує – якою довжиною буде таке число. Для запису VLI-кодів використовують три числа: $(start, step, stop)$, де $start$ визначає начальну довжину числа, $step$ – приріст довжини для наступної групи, $stop$ - кінцеву довжину.

Унарний код для n-ої групи – n одиниць, потім ноль; для останньої групи - тільки n одиниць, так як це не вносить невизначеності. Розмір числа з n-ої групи рівне $start + n \cdot step$. VLI-коди зручно застосовувати для кодування монотонних джерел, котрим ймовірність появи менших чисел вище, ніж великих. За відсутності кодування будь-яке число було б десятьма бітами.

Таблиця 2.2 – Параметри унарного кодування

Алгоритм	Ступінь стиску	Швидкість	Пам'ять	Стиск без втрат	Проходи	РП	ЗН
VLI	2-3	9	10	Да	1	Так	Можл.

Робота: в реальному масштабі часу і в потоці.

Модель: джерело Бернулі (монотонне).

Класифікація: неблочний, статистичний.

Ентропійне кодування. Алгоритм Хаффмана. Адаптивне кодування Хаффмана. Доведено, що найбільш ефективним кодом змінної довжини, в якому жодне слово не збігається з початком іншого (тобто префіксний код), є код Хаффмана [2; 5; 7].

Нехай $l_1,...,l_k$ - позитивні цілі числа $(k > 0)$. Щоб існував префіксний код, довжини слів якого рівні $l_1,...,l_k$, необхідно і достатньо виконання нерівності Крафта [7]:

$$\sum_{i=1}^{k} 2^{-li} \le 1. \qquad (2.1)$$

Усі префіксні коди є кодами з властивістю однозначного декодування, але не навпаки (наприклад, код, що однозначно декодується 0, 01, 011, 0111, ... не є префіксним). Надмірність кодування, що дешифрується, невід'ємна. Для коду Хаффмана (Шеннона) надмірність не перевищує 1, тобто $0 \le R \le 1$. Довжина коду Шеннона дорівнює:

$$|a_i| = \lceil -\log(p(a_i)) \rceil, \quad (2.2)$$

а довжина коду Хаффмана не перевищує величини $|a_i|$.

Звідси, зокрема слідує висновок, що чим більша довжина Т символів вхідного алфавіту (для якого будується код Хаффмана), тим менша надмірність вихідного тексту і тим вищий ступінь стиснення. Однак, як буде видно надалі, при цьому значно зростають вимоги до пам'яті даних і швидкодії програми, оскільки кількість кодів дорівнює кількості символів вихідних літер. Надмірність коду Хаффмана значною мірою залежить, як випливає з наведеної формули, від того, наскільки ймовірності появи символів близькі до негативних ступенів числа 2. Для двосимвольного алфавіту, наприклад код Хаффмана ніколи не зможе дати скорочення надмірності, нехай навіть ймовірності різняться на кілька порядків.

Інший алгоритм побудови оптимальних префіксних кодів, що дає аналогічний результат, було запропоновано Шенноном та Фано [7; 13].

Недоліком такого кодування є той факт, що разом із закодованим повідомленням необхідно передавати також побудовану таблицю кодів (дерево), що знижує величину стиснення. Однак існує динамічний алгоритм побудови дерева Хаффмана [2; 5; 7], при якому кодова таблиця оновлюється самим кодером та (синхронно та аналогічно) декодером у процесі роботи після отримання кожного чергового символу. Отримані при цьому коди виявляються квазіоптимальними, тому текст стискається дещо гірше. Більше того, зростають складність алгоритму та час роботи програми (хоча вона і стає однопрохідною), тому в даний час динамічний алгоритм майже повністю поступився місцем статичному.

Наступним варіантом аналізованого алгоритму є фіксований алгоритм Хаффмана, що поєднує у собі переваги двох попередніх – високу швидкість роботи, простоту та відсутність додаткового словника, що створює зайву надмірність. Ідея його в тому, щоб користуватися деяким усередненим за багатьма текстами деревом, однаковим для кодера та декодера. Таке дерево не треба будувати і передавати разом із текстом, а отже – відпадає необхідність першого проходу. Але, з іншого боку, усереднене фіксоване дерево виявляється ще неоптимальніше, ніж динамічне. Тому іноді може бути зручно мати кілька фіксованих дерев для різних видів інформації.

Варіантами дерева Хаффмана слід вважати дерева, отримані для монотонних джерел.

Ці джерела мають дві важливі для наших цілей властивості:

– немає необхідності обчислювати частоти появи символів вхідного тексту - як наслідок однопрохідний алгоритм;

– дерево Хаффмана для таких джерел може бути представлене у вигляді вектора з декількох байтів, кожен з яких означає кількість кодів дерева певної довжини.

Порівняно недавно з'явився ще один різновид адаптивного алгоритму Хаффмана, описаний Рябко, а потім Бентлі і названий, відповідно, алгоритмом стопки книг або "рухай вгору" ("MTF-Move To Front") [2; 5]. Кожному символу (літері) надається код залежно від його положення в алфавіті - чим ближче символ до початку алфавіту - тим коротший код (як код можна використовувати, наприклад, код дерева для монотонних джерел). Після кодування чергового символу ми поміщаємо його на початок алфавіту, зрушуючи решту літер на одну позицію вглиб. Через деякий час символи, що найчастіше зустрічаються, згрупуються на початку, що і потрібно для успішного кодування. Робота алгоритму, дійсно, нагадує перекладення найбільш потрібних книг із глибин бібліотеки ближче до верху, внаслідок чого потім їх буде легко знайти (знов аналогія з повсякденним життям).

Таблиця 2.3 – Параметри стиснення Хаффмана

Алгоритм	Ступінь стиску	Швид-кість	Пам'ять	Стиск без втрат	Проходи	РП	ЗН
Хаффман статич.	5-6	8	8	Да	2	Да	Рідко
Хаффман динаміч.	4-5	6	7	Да	1	Да	Можл.
Хаффман фіксир.	3-6	9	9	Да	1	Да	Можл.
Стопка книг	3-5	7	8	Да	1	Да	Можл.
Монотон.	3-4	8	9	Да	1	Да	Можл.

Робота: у реальному масштабі часу та в потоці (крім статичного).

Основне застосування: універсальний, стиснення текстів та бінарної інформації. Динамічний алгоритм побудови коду Хаффмана використовується в архіваторі ICE, статичний в LHA, ARJ. Статичний алгоритм Шеннона-Фано використовується архіватором PKZIP. Застосовується для стиснення зображень (JPEG).

Термін: Squashing.

Модель: Бернуллі.

Класифікація: неблоковий, статистичний.

Ентропійне кодування. Алгоритм арифметичного кодування. Адаптивне арифметичне кодування. Арифметичне кодування [2; 5; 7; 14] є методом, що дозволяє упаковувати символи вхідного алфавіту без втрат за умови, що відомий розподіл частот цих символів. Арифметичне кодування є оптимальним, досягаючи теоретичної межі ступеня стиснення. Проте, коди Хаффмана, ми також називали оптимальними, як узгодити цей парадокс? Відповідь ось у чому: коди Хаффмана є неблочними, тобто. кожній літері вхідного алфавіту ставиться у відповідність певний вихідний код. Арифметичне кодування - блокове та вихідний код є унікальним для кожного з можливих вхідних повідомлень, його не можна розбити на коди окремих символів.

Текст, стиснутий арифметичним кодером, сприймається як деяка двійкова дріб з інтервалу . Результат стиснення можна подати як послідовність двійкових цифр із запису цього дробу. Кожен символ вихідного тексту представляється відрізком на числовій осі з довжиною, що дорівнює ймовірності його появи і початком, що збігається з кінцем відрізка символу, що передує йому в алфавіті. Сума всіх відрізків, очевидно, повинна дорівнювати одиниці. Якщо розглядати на кожному кроці поточний інтервал як ціле, то кожен знову вхідний символ «вирізає» з нього підінтервал пропорційно своїй довжині і положенню (див. рисунок 1.1).

Пояснимо роботу методу з прикладу. Нехай алфавіт складається із двох символів: "*a*" і "*b*" з ймовірностями відповідно $3/4$ і $1/4$. Розглянемо (відкритий праворуч) інтервал $[0,1)$. Розіб'ємо його на частини, довжина яких пропорційна ймовірностям символів. У нашому випадку це $[0,3/4)$ і $[3/4,1)$. Суть алгоритму в наступному: кожному слову у вхідному алфавіті відповідає

деякий подінтервал з $[0,1)$. Порожньому слову відповідає весь інтервал $[0,1)$. Після отримання кожного чергового символу арифметичний кодер зменшує інтервал, вибираючи ту його частину, яка відповідає символу, що знову надійшов. Кодом повідомлення є інтервал, виділений після обробки всіх його символів, точніше число мінімальної довжини, що входить в цей інтервал. Довжина отриманого інтервалу пропорційна ймовірності появи тексту, що кодується.

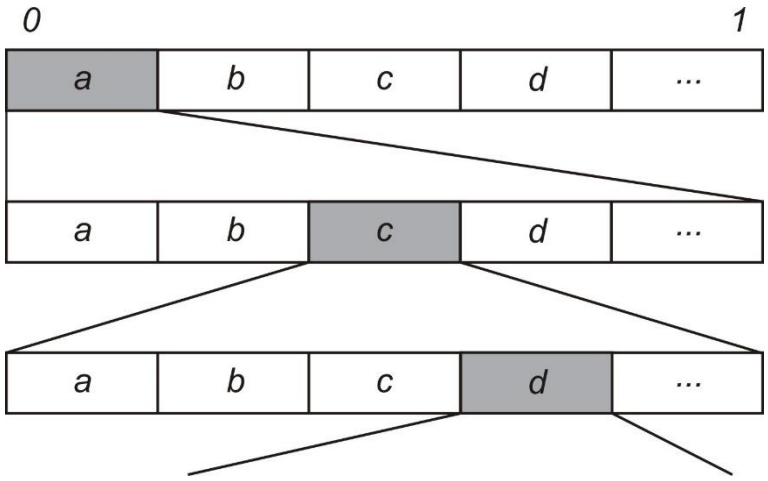


Рисунок 2.1 – Побудова інтервалу для повідомлення «acd…»

Виконаємо алгоритм для ланцюжка "*aaba*" (див. таблицю 1.4):

Таблиця 2.4 – Алгоритм ланцюжка кодування

Крок	Переглянутий ланцюжок	Інтервал
0.	""	$[0,1)=[0,1)$
1.	"*a*"	$[0,3/4)=[0,0.11)$
2.	"*aa*"	$[0,9/16)=[0,0.1001)$
3.	"*aab*"	$[27/64,36/64)=[0.011011,0.100100)$
4.	"*aaba*"	$[108/256,135/256)=[0.01101100,0.10000111)$

На першому кроці ми беремо перші $3/4$ інтервала, відповідні символу "*a*", потім залишаємо від нього ще тільки $3/4$. Після третього кроку від попереднього інтервалу залишиться його права чверть відповідно до положення та ймовірності символу "*b*". І, нарешті, на четвертому кроці ми залишаємо лише перші $3/4$ від результату. Це і інтервал, якому належить вихідне повідомлення.

Як код можна взяти будь-яке число з діапазону, отриманого на кроці 4. Вихідний текст можна було закодувати чотирма бітами, а ми отримали 8-бітовий інтервал. Справа в тому, що як код ми можемо вибрати, наприклад, найкоротший код з інтервалу, що дорівнює 0,1 і отримати чотириразове скорочення обсягів тексту. Для порівняння, код Хаффмана не зміг би стиснути подібне повідомлення, однак, на практиці виграш зазвичай невеликий і перевага надається більш простому та швидкому алгоритму.

Арифметичний декодер працює синхронно з кодером: почавши з інтервалу $[0,1)$, він послідовно визначає символи вхідного ланцюжка. Зокрема, у нашому випадку він спочатку розділить (пропорційно до частот символів) інтервал $[0,1)$ на $[0,0.11)$ і $[0.11,1)$. Оскільки число 0.1 (переданий кодером код ланцюжка "*aaba*") знаходиться в першому з них, можна отримати перший символ "*a*". Потім ділимо перший підінтервал $[0,0.11)$ на $[0,0.1001)$ і $[0.1001,0.1100)$ (пропорційно до частот символів). Знову обираємо перший, тому що $0 < 0.1 < 0.1001$. Продовжуючи цей процес, ми однозначно декодуємо всі чотири символи.

При розгляді цього виникають дві проблеми: по-перше, необхідна речовинна арифметика, взагалі кажучи, необмеженої точності, і по-друге, результат кодування стає відомий лише при закінченні вхідного потоку. Подальші дослідження, однак, показали, що можна практично без втрат обійтися цілою арифметикою невеликої точності (16-32 розряду), а також досягти інкрементальної роботи алгоритму: цифри коду можуть видаватися послідовно в міру читання вхідного потоку.

Подібно до алгоритму Хаффмана, арифметичний кодер також є двопрохідним і вимагає передачі разом із закодованим текстом ще й таблиці частот символів. Взагалі, ці алгоритми дуже схожі і легко взаємозаміняться. Отже, існує адаптивний алгоритм арифметичного кодування з усіма перевагами та недоліками. Його основна відмінність від статичного в тому, що нові інтервали ймовірності перераховуються після отримання кожного наступного символу вхідного потоку.

Таблиця 2.5 – Параметри арифметичного кодування

Алгоритм	Ступінь стиску	Швидкість	Пам'ять	Стиск без втрат	Проходи	РП	ЗН
Арифмет. статич.	5-6	8	8	Так	2	Так	Рідко
Арифмет. динаміч.	4-5	6	7	Так	1	Так	Рідко

Робота: у реальному масштабі часу та в потоці (крім статичного).

Основне застосування: універсальний, стискування текстів. Використовується в архіваторі LZARI для стиснення зображень (JPEG).

Модель: джерело Бернуллі.

Класифікація: блоковий, статичний.

Імовірнісне стиснення. Надзвичайно цікавий і найшвидший із відомих (поряд з RLE) методів стиснення інформації, що дає, до того ж непогані результати. Імовірнісне стиснення [2; 5; 7] багато в чому нагадує ворожіння на кавовій гущі або прогноз погоди, тільки більш ефективно.

Працює алгоритм у такий спосіб: є досить велика, динамічно оновлюється таблиця передбачень, у якій кожної можливої пари послідовних вхідних символів вказується наступний (третій) символ. Якщо символ передбачений правильно – генерується код у вигляді однобітного префікса, що дорівнює 1. Якщо ж ми не вгадали – видається код у вигляді префікса, що дорівнює 0 і невгаданого символу. У цьому невгаданий символ заміщає у таблиці колишній там колись, забезпечуючи постійне оновлення статистичної інформації.

Алгоритм використовує модель сигналу, що є моделлю Маркова 2 порядку. Алгоритм є адаптивним і тому він однопрохідний і вимагає зберігання таблиці разом із стислим текстом.

Декомпресор працює за аналогічною програмою, підтримуючи та оновлюючи таблицю синхронно з компресором. Якщо надійшов префікс 1, чергова вихідна буква витягується з таблиці індексу, отриманому з двох попередніх букв, інакше - просто копіюється код з вхідного потоку у вихідний. Для невеликого підвищення ступеня стиснення рекомендується ініціалізувати

таблицю перед початком роботи якими-небудь найпоширенішими буквосполученнями.

Схожий алгоритм, заснований на моделі Маркова порядку 1, використовувався в архіваторі PKZIP Ver.1.5 під ім'ям Reducing [2; 5; 7]. Він є майже за всіма параметрами гірше, ніж Fin (двопрохідний, вимагає передачі таблиці разом з текстом), проте гідний згадки. Таблиця програми Reducing містить масив символів $\vec{v}[256][32]$, тобто. для кожної вхідної літери на першому проході знаходяться не більше, ніж 32 (але може бути і менше) наступних літери, що найбільш часто зустрічаються. На другому проході справа подібна до Fin - якщо черговий символ b, наступний за символом a знаходиться серед 32 очікуваних, генерується код $< prefix, position >$, $prefix = 1$, $position$ дорівнює положенню символу b в масиві $\vec{v}[a][]$ з 32 букв. Інакше видається префікс 0 та сам символ b. Алгоритм статичний, тому таблиця не оновлюється у процесі роботи.

Таблиця 2.6 – Параметри імовірнісного стиснення

Алгоритм	Ступінь стиску	Швид-кість	Пам'ять	Стиск без втрат	Проходи	РП	ЗН
Reducing	4-5	8	7	Да	2	Да	Рідко
Імовірн.	5-6	9	6	Да	1	Да	Можл.

Робота: у реальному масштабі часу та в потоці.

Основне застосування: універсальний.

Термін: Reducing.

Модель: Маркова 1 або 2 порядку.

Класифікація: неблоковий, статистичний.

3 РОЗРОБКА АЛГОРИТМУ ФУНКЦІОНУВАННЯ СТРУКТУРНОЇ СХЕМИ ЕЛЕКТРОННОЇ СИСТЕМИ

3.1 Обґрунтування алгоритму функціонування

Перш ніж розпочати синтез структурної схеми проектованого пристрою, необхідно чітко визначити, за яким алгоритмом він повинен працювати. Алгоритмом називається послідовність дій (операцій), що призводить до досягнення поставленої мети. Алгоритм повинен мати універсальність у межах заданого класу умов. Так, алгоритм керуючої системи повинен призводити до необхідного результату за будь-яких допустимих збурень. Алгоритм також повинен бути детермінованим, тобто при конкретних керуючих та обурювальних впливах наказувати виконання однозначно певних дій, і кінцевим, тобто бажана мета повинна досягатися за кінцеве число кроків. У цьому має забезпечуватися стійкість процесів управління. Алгоритм не повинен приводити систему в нестійкий стан і до некоректних впливів керуючих при виході за межі робочого діапазону вхідних сигналів.

По відношенню до алгоритму арифметичного кодування нетривіальним є питання, кого вважати його автором. Ідею приписують Елаєсу, посилаючись на книгу Абрамсона [2; 5], опубліковану 1963 р. До кінця 70-х ця ідея не була затребувана. Декілька статей, опублікованих на межі 70-х і 80-х років носили скоріше теоретичний характер. Велику роль у розумінні як ідеї алгоритму, так і можливості його використання для вирішення практичних завдань зіграла стаття Віттена, Нілі та Клірі [15]. Мабуть, саме цих авторів слід вважати авторами алгоритму у його нинішній формі. В той же час, коли став відомий сам алгоритм, стало зрозуміло, що він є майже тривіальним узагальненням коду Шеннона на послідовності. Можливо, тому у книзі Кавера та Томаса [16] арифметичне кодування називається алгоритмом Шеннона-Фано-Елайєса.

Розглянемо для простоти дискретне постійне джерело, що вибирає повідомлення з множини $X = \{1,...,M\}$, з ймовірностями $\{p_1,..., p_M\}$. Позначимо через $\{q_1,...,q_M\}$ кумулятивні ймовірності повідомлень. Наше завдання полягає в кодуванні послідовностей множини $X^n = \{\vec{x}\}$. При описі

алгоритму кодування ми будемо використовувати позначення $\vec{x}_i^{\,j}$ для короткого запису підпослідовності $(x_i,...,x_j)$ в послідовності $\vec{x}=(x_1,...,x_n)$.

Ми хотіли б застосувати до ансамблю $X^n=\{\vec{x}\}$ досить простий і ефективний літерний код. Спрощення полягає в тому, ні кодер ні декодер не зберігають і не будують всієї множини з $|X^n|$ кодових слі. Натомість при передачі конкретної послідовності $\vec{x}$ кодером обчислюється кодове слово $\vec{c}(\vec{x})$ тільки для даної послідовності $\vec{x}$. Правило кодування, звичайно, відоме декодеру і він відновлює $\vec{x}$ по $\vec{c}(\vec{x})$, без повного списку кодових слів.

Можливими кандидатами на використання у такій схемі можна розглядати код Шеннона та код Гільберта-Мура. Однак використання коду Шеннона передбачає впорядкованість повідомлень зі спадання ймовірностей. За великих n складність упорядкування виявиться неприпустимо великою, тому єдиним претендентом залишається код Гілберта-Мура.

Відповідно до правила побудови коду Гілберта-Мура кодове слова формується за ймовірністю $p(\vec{x})$ і кумулятивної ймовірності $q(\vec{x})$ як перших

$$l(\vec{x})=\left\lceil -\log p(\vec{x})+1\right\rceil \qquad (3.1)$$

розрядів після крапки в двійковому запису числа

$$\sigma(\vec{x})=q(\vec{x})+p(\vec{x})/2. \qquad (3.2)$$

Для того, щоб обчислити $q(\vec{x})$, треба домовитися про деяку нумерацію послідовностей з X^n. Найбільш природний спосіб нумерації послідовностей – використання лексикографічної впорядкованості. Лексикографічний порядок на послідовностях позначатиметься знаком "$\prec$". Запис $\vec{y}\prec\vec{x}$ буде значити, що $\vec{y}$ лексикографічно передує $\vec{x}$.

Поняття лексикографічного порядку визначається в такий спосіб.

Для послідовностей довжини 1 (для окремих повідомлень з X) ми вважаємо, що повідомлення з меншим номером передують повідомленню з

великим номером. Якщо, наприклад, елементи X - числа, то $x \prec x^{/}$, якщо $x < x^{/}$, $x, x^{/} \in X$.

Для двох послідовностей $\vec{x} = (x_1,...,x_n)$, $\vec{y} = (y_1,...,y_n)$ позначимо через i найменший індекс такий, що $x_i \neq y_i$. Тоді $\vec{y} \prec \vec{x}$, якщо $y_i \prec x_i$.

Неважко бачити, що лексикографічний порядок – це порядок, який зазвичай використовується при складанні словників.

Отже, основне завдання полягає у обчисленні кумулятивної ймовірності:

$$q(\vec{x}) = \sum_{y \prec x} p(\vec{y}), \tag{3.3}$$

оскільки для джерела без пам'яті ймовірності послідовностей $p(\vec{x})$ обчислюються досить просто за такою формулою:

$$p(\vec{x}) = \prod_{i=1}^{n} p(x_i). \tag{3.4}$$

Виведемо рекурентну формулу для обчислення $q(\vec{x})$. Для цього висловимо ймовірність $q(\vec{x}_1^{\,n})$ через $q(\vec{x}_1^{\,n-1})$. Наступний ланцюжок елементарних перетворень показує, як це можна зробити:

$$q(\vec{x}_1^{\,n}) = \sum_{\vec{y}_1^{\,n} \prec \vec{x}_1^{\,n}} p(\vec{y}_1^{\,n}) = \sum_{\vec{y}_1^{\,n-1} \prec \vec{x}_1^{\,n-1}} \sum_{y_n} p(\vec{y}_1^{\,n-1}, y_n) + \sum_{\vec{y}_1^{\,n-1} = \vec{x}_1^{\,n-1}} \sum_{y_n \prec x_n} p(\vec{y}_1^{\,n-1}, y_n) =$$

$$= \sum_{\vec{y}_1^{\,n-1} \prec \vec{x}_1^{\,n-1}} p(\vec{y}_1^{\,n-1}) + \sum_{\vec{y}_1^{\,n-1} = \vec{x}_1^{\,n-1}} p(\vec{y}_1^{\,n-1}) \cdot \sum_{y_n \prec x_n} p(y_n) = q(\vec{x}_1^{\,n-1}) + p(\vec{x}_1^{\,n-1}) \cdot q(x_n), \tag{3.5}$$

де $q(x_n)$ – означає кумулятивну ймовірність символу x_n.

Підсумуємо наші викладки у вигляді рекурентних формул:

$$q(\vec{x}_1^{\,n}) = q(\vec{x}_1^{\,n-1}) + p(\vec{x}_1^{\,n-1}) \cdot q(x_n), \tag{3.6}$$

$$p(\vec{x}_1^{\,n}) = p(\vec{x}_1^{\,n-1}) \cdot p(x_n). \tag{3.7}$$

У цій рекурсії кожна пара значень $(q(\vec{x}_1^{\,i}), p(\vec{x}_1^{\,i}))$ використовується рівно на одному кроці при обчисленні наступної пари $(q(\vec{x}_1^{\,i+1}), p(\vec{x}_1^{\,i+1}))$. Тому при

реалізації арифметичного кодування, знову обчислені значення записуються в ті ж осередки пам'яті, в яких були попередні значення. У наведеному нижче алгоритмі кумулятивні ймовірності $q(\vec{x}_1^{\,i})$, $i = 1,2,...$ зберігаються у вигляді змінної F, ймовірності послідовностей $p(\vec{x}_1^{\,i})$, $i = 1,2,...$ – у вигляді змінної G.

3.2 Алгоритм арифметичного кодування

Остаточне формулювання алгоритму може бути записано у такому вигляді:

1. Ініціалізація. По ймовірностям $\{p_1,...,p_M\}$ повідомлень джерела обчислюються кумулятивні ймовірності:

$$\begin{gathered} q_1 = 0, \ для \ j = 1 \\ q_j = q_{j-1} + p_{j-1}, \ для \ j = 2,...,M. \end{gathered} \tag{3.8}$$

Встановлюються початкові значення допоміжних змінних:

$$F = 0, \ G = 1. \tag{3.9}$$

Приймається від джерела послідовність повідомлень:

$$\vec{x} = (x_1,...,x_n). \tag{3.10}$$

2. Для $i = 1,...,n$ виконуються такі обчислення:

$$F \leftarrow F + q(x_i) \cdot G, \tag{3.11}$$

$$G \leftarrow p(x_i) \cdot G. \tag{3.12}$$

3. Кодове слово для $\vec{x}$ формується як перші:

$$Довжина \ кодового \ слова = \lceil -\log_2 G + 1 \rceil \tag{3.12}$$

розрядів після коми в двійковому запису числа:

$$Кодове \ слово = (F + G/2). \tag{3.13}$$

Відповідно до виразів (3.8)-(3.13) розглянемо практичний приклад. Розглянемо джерело: $X = \{a,b,c\}$, розподіл ймовірностей $p_a = 0{,}1$, $p_b = 0{,}6$, $p_c = 0{,}3$. Обчислення, що виконуються арифметичним кодером під час кодування послідовності $\vec{x} = (bcbab)$ довжини $n = 5$, приведені в таблиці 3.1.

В цій таблиці через $\hat{F}$ позначено число $(F+G/2)$ округлене вниз з точністю до $\lceil -\log_2 G+1 \rceil = 9$ двійкових розрядів.

Таблиця 3.1 – Кодування послідовності арифметичним кодом

Крок i	x_i	$p(x_i)$	$q(x_i)$	F	G
0	–	–	–	0,0000	1,0000
1	b	0,6	0,1	0,1000	0,6000
2	c	0,3	0,7	0,5200	0,1800
3	b	0,6	0,1	0,5380	0,1080
4	a	0,1	0,0	0,5380	0,0108
5	b	0,6	0,1	0,5391	0,0065
6	Довжина кодового слова		Кодове слово		
	$\lceil -\log_2 G+1 \rceil = 9$		$\hat{F} = F + G/2 = 0{,}5423\ldots._{10} = 0{,}100010101_{\ 2}$		

Графічна інтерпретація процесу кодування аналогічна інтерпретації кодів Шеннона та Гілберта-Мура показана на рисунку 3.1.

Як показано на малюнку, на кожному кроці кодування перераховується початкова точка F і довжина відрізка G, в якому буде розташовано число, відповідне кодової послідовності для заданої послідовності повідомлень. Так, після першого кроку ($x_1 = b$) ми знаємо, що точка буде розміщена у відрізку [0.1, 0.7). Більш детально цей відрізок показано на рис. 1.3 б. Оскільки друга літера $x_2 = c$, початкова точка переміщається у значення 0,52, а довжина інтервалу зменшується до 0,18 і т.д. Після п'ятого кроку $F = 0{,}541$. Тим самим вся послідовність повідомлень відображається в одну точку інтервалу [0,1). Ця точка позначена кружком на малюнках 3.1 а-3.1 д. Як було показано вище, 9 розрядів достатньо, щоб послідовність була відновлена однозначно, тобто. найближча можлива точка, що відповідає іншій послідовності повідомлень, віддалена від $\hat{F}$ на відстань не менше $1/2^9 = 1/512$.

Коротко обговоримо питання про складність кодування. З опису алгоритму слід, що у кожному кроці кодування виконується одне додавання і 2 множення. Звідси легко зробити неправильний висновок у тому, що складність кодування послідовності з повідомлень пропорційна n. Це не так, оскільки кожному етапі лінійно зростає складність виконання самих операцій складання і множення, так як наростає число двійкових розрядів, необхідні записи операндів.

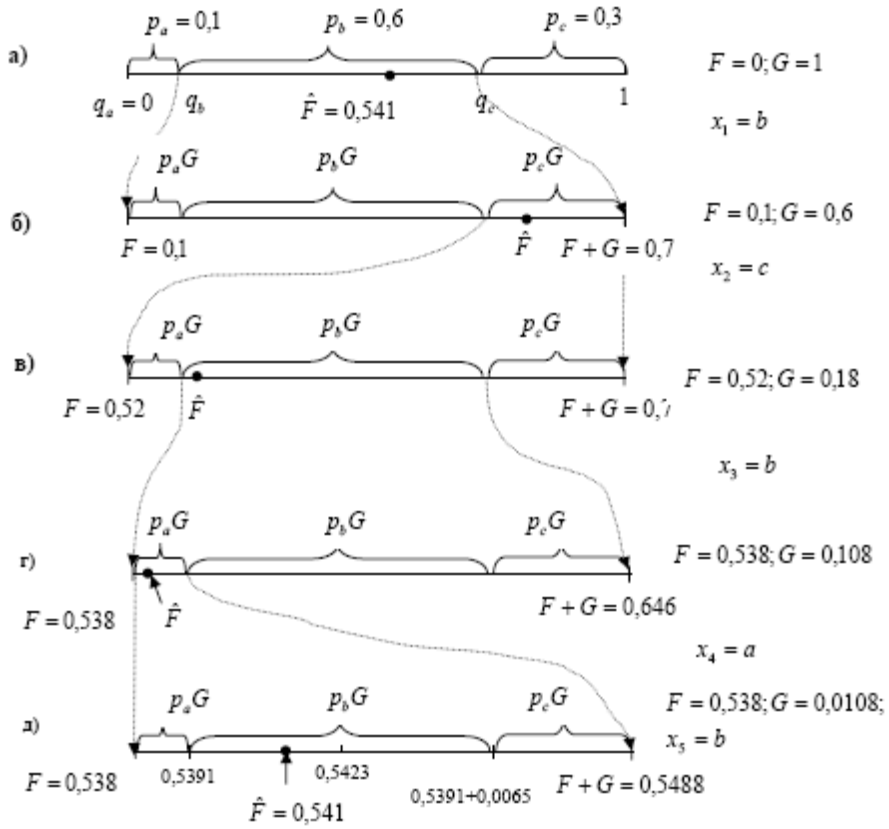


Рисунок 3.1 – Графічна інтерпретація арифметичного кодування

Припустимо, що для подання ймовірностей $p_1,...,p_M$ використані числа розрядності d. Після першого кроку кодування точне уявлення F і G потребує $2d$ розрядів, ..., після n кроків кодування кодер та декодер працюватимуть (у гіршому випадку) з числами розрядності nd, і, отже, сумарна складність має порядок

$$d + 2 \cdot d + ... + n \cdot d = \frac{n(n+1)}{2} \cdot d . \qquad (3.14)$$

Таким чином, можна говорити, що складність арифметичного кодування пропорційна n^2.

Відзначимо ще одну надзвичайно важливу особливість арифметичного кодування. Його легко адаптувати до джерел з пам'яттю. Якщо, наприклад, як модель джерела розглядається простий ланцюг Маркова, то алгоритм кодування залишається тим самим винятком, що замість одновимірних ймовірностей $p(x_i)$ і $q(x_i)$ потрібно використовувати умовні ймовірності $p(x_i \mid x_{i-1})$ и $q(x_i \mid x_{i-1}) = \sum_{y \prec x_i} p(x_i \mid x_{i-1})$.

3.3 Практична реалізація алгоритму кодування

Як ясно з опису арифметичного кодування, при його практичній реалізації для кодування послідовностей великої довжини виникають такі проблеми:

– арифметичне кодування вимагає великої (у межі - нескінченної) точності обчислень, що веде до неприпустимо високої складності реалізації;

– для формування кодового слова формально необхідна вся послідовність повідомлень, що призводить до неприпустимо великої затримки кодування, яка дорівнює довжині послідовності повідомлень, що кодується.

Для реалізації алгоритму кодування необхідно провести такі операції:

1. Створюється список, що складається з елементів множини літер вхідного алфавіту, всі ймовірності $\{p_1,...,p_M\}$ символів, що зустрічаються у вхідному тексті, який необхідно заздалегідь записати.

2. Встановлення початкового значення кумулятивних ймовірностей $q_1 = 0$, та допоміжних параметрів $F = 0,\ G = 1$.

3. Виконується цикл розрахунку таблиці кумулятивних ймовірностей $q_j = q_{j-1} + p_{j-1},\ для\ j = 2,...,M$. Під час цього накопичення, доцільно використовувати два тимчасові значення для q_{j-1} та q_j, тому необхідним кроком є збереження чергового чергового q_j та заміна попереднього значення наступним $q_j \to q_{j-1}$ на кожній ітерації процесу.

4. Після завершення циклу обчислення кумулятивних ймовірностей (обчислення меж ймовірностей наших символів на відрізку $[0,1)$ майбутнього закодованого повідомлення) відбувається прийом інформаційного блоку, а саме від джерела послідовності повідомлень $\vec{x} = (x_1,...,x_n)$.

5. Кожен прийнятий символ (група з М біт) x_i узгоджується зі значеннями ймовірностей $(q(x_i), p(x_i))$ і відбувається робота зі значеннями ймовірностей.

6. За описаними раніше формулами алгоритму $F_{i+1} \leftarrow F_i + q(x_i) \cdot G_i$, $G_{i+1} \leftarrow p(x_i) \cdot G_i$, виробляються обчислення. Під час даних розрахунків,

доцільно використовувати чотири тимчасові значення для F_i, F_{i+1}, G_{i+1} і G_i, тому необхідним кроком є збереження чергових F_i, G_i значень допоміжних параметрів та заміна попередніх значень наступними $F_{i+1} \to F_i$, $G_{i+1} \to G_i$ на кожній ітерації процесу.

7. Після всіх ітерацій символів вхідної послідовності повідомлення, проводиться аналіз та обчислюється кодове слово для $\vec{x}$, яке формується як перші $\lceil \log_2 G + 1 \rceil$ розрядів після коми у двійковому записі числа $(F + G/2)$.

8. Далі відбувається передача закодованого повідомлення одержувачу та закінчення алгоритму арифметичного кодування.

3.4 Алгоритм роботи функціонування пристрою

Відповідно до виразів (3.1)-(3.14) розглянемо приклад, що стосується нашого пристрою. Виходячи із заданих заздалегідь умов, кількість символів алфавіту дорівнює 4, а довжина повідомлення дорівнює 7. Розглянемо джерело: $X = \{a, b, c, d\}$, можливий розподіл ймовірностей $p_a = 0{,}285714286$, $p_b = 0{,}142857143$, $p_c = 0{,}285714286$, $p_d = 0{,}285714286$. Обчислення, що виконуються арифметичним кодером під час кодування послідовності $\vec{x} = (abddcca)$ довжини $n = 7$, наведені в таблиці 3.2.

Таблиця 3.2 – Кодування послідовності арифметичним кодом

Крок i	x_i	$p(x_i)$	$q(x_i)$	F	G
0	–	–	–	0,0	1,0
1	a	0,285714286	0,0	0,0	0,285714286
2	b	0,142857143	0,285714286	0,081632653	0,040816327
3	d	0,285714286	0,714285714	0,110787172	0,011661808
4	d	0,285714286	0,0	0,110787172	0,003331945
5	c	0,285714286	0,428571429	0,112215148	0,000951984
6	c	0,285714286	0,428571429	0,112623142	0,000271996
7	a	0,285714286	0,0	0,112623142	0,000077713
8	Длина кодового слова		Кодовое слово		
	$\lceil -\log_2 G + 1 \rceil = 15$		$\hat{F} = F + G/2 = 0{,}112661998_{10} = 0{,}000111001\ 101011_2$		

Як видно, довжина кодового слова має максимальне значення 15. В інших варіантах значень ймовірностей p_a, p_b, p_c, p_d (іншої частоти символів a, b, c, d при кодуванні послідовності довжиною в $n=7$ символів), довжина кодового слова буде йти на зменшення, тому для нашого пристрою має достатньо вистачити 16 біт даних.

Враховуючи досить складну практичну реалізацію алгоритму арифметичного кодування, ми можемо зробити висновок, що реалізація апаратним методом даного алгоритму досить трудомістка, тому доцільніше вирішити це завдання програмним методом. У цьому нам допоможе мікропроцесорний комплект, який використовуватиме шину даних – 16 біт, а також вміти оперувати з речовими числами (наприклад, арифметичний співпроцесор). Загалом у нас має вийти контролер, який має реалізовувати раніше описаний алгоритм арифметичного кодування.

Визначимося з тим, які операції має виконувати проектований пристрій. Наш пристрій повинен вміти отримувати дані від джерела, їх обробляти за раніше описаним алгоритмом і передавати перетворені (закодовані) дані одержувачу. Також необхідно узгоджувати з джерелом готовність до прийому даних, запит на прийом інформаційного блоку, узгоджувати з одержувачем готовність до передачі закодованого повідомлення, запит на передачу закодованого повідомлення, очікування наступної порції повідомлення від джерела до одержувача.

Опишемо алгоритм роботи пристрою стиснення даних на основі арифметичного кодування.

1. На початку роботи пристрою, виконується скидання, початкова установка та ініціалізація пристрою.

2. Відбувається очікування прийому даних для стиснення джерела, назвемо – зовнішньої шиною, оскільки крім даних необхідно ще отримувати сукупність сигналів керувати пристроєм ззовні.

3. Проводиться умова перевірки на готовність пристрою до прийому даних, якщо пристрій не готовий для прийому даних, то він продовжує перебувати в режимі очікування для прийому даних.

4. Після підтвердження готовність до прийому даних проводиться запит на передачу, і навіть обмін керуючими сигналами.

5. Проводиться перевірка готовності джерела до передачі даних даних для стиснення. За відсутності такого, проводиться повторний запит на передачу та обмін сигналами, що управляють.

6. По зовнішній шині надходить сигнал переривання, який активує пристрій стиснення прийому інформаційного блоку.

7. Відбувається прийом інформаційного блоку зовнішньої шини. Оскільки для подання кожного символу нам буде достатньо 1 байта даних (8 біт), то цей процес виконує прийом байта даних, який повторюється $n = 7$ разів.

8. Після отримання інформаційного блоку, і занесення його в пам'ять пристрою, виконуються арифметичні операції згідно з раніше описаним алгоритмом арифметичного кодування. Внаслідок їх виконання ми отримуємо закодоване повідомлення, яке необхідно передати одержувачу.

9. У пристрої виконується очікування передачі даних канал зв'язку, з перевіркою готовності каналу зв'язку прийому. Якщо ж готовності немає, то продовжується очікування передачі в канал зв'язку.

10. При отриманні підтвердження готовності виконується передача в канал зв'язку закодованого повідомлення, внаслідок чого відбувається процес очікування наступного повідомлення або закінчення роботи алгоритму пристрою.

3.5 Розробка структурної схеми електронної системи

Виходячи з, розглянутого раніше, алгоритму роботи проектованого пристрою стиснення даних на основі арифметичного кодування розробимо структурну схему пристрою.

Раніше ми дійшли висновку, що реалізація апаратним методом даного алгоритму досить трудомістка, тому доцільніше вирішити дану задачу програмним методом. У цьому нам допоможе мікропроцесорний комплект, який повинен вміти оперувати з речовими числами. У цілому, у нас повинен вийти мікроконтролер, який повинен реалізовувати раніше описаний алгоритм арифметичного кодування.

Всі функції пристрою стиснення даних на основі арифметичного кодування можна розділити на три великі групи: магістральні, обчислювальні

та канали зв'язку (одержувача). Магістральні функції реалізують пару пристрою з комп'ютером. Функції першої групи визначаються пристроєм комп'ютера, що підключається до пристрою, і не відрізняються великою різноманітністю, оскільки комп'ютерні інтерфейси для підключення різних пристроїв зазвичай стандартизовані і нечисленні. Обчислювальні функції другої групи реалізують процес обробки отриманих даних. Функції другої групи визначаються вибором центральних процесора та співпроцесора, трактом формування шин адреси, даних, управління, обмін даними з пам'яттю та характеризуються виконуваними операціями закладеного алгоритму у програмі. Функції третьої групи визначаються типом каналу зв'язку і можуть бути різними в залежності від типу середовища передачі даних, прийнятого протоколу управління та інших особливостей.

Під магістральними (шинними) функціями тут розуміється організація сполучення із системною магістраллю (шиною) персонального комп'ютера. Найбільшого поширення серед системних (зовнішніх) магістралей персональних комп'ютерів типу IBM PC набула ISA, що є 16-разрядную магістраль з роздільними шинами адреси та даних. Вона підтримує програмний обмін та обслуговування переривань. Ця магістраль використовується практично у всіх комп'ютерах.

Ми будемо використовувати деяку її подобу, використовуючи не повністю існуючі функції, зважаючи на обмежений набір необхідних функцій для пристрою, а також через те, що пристрій під'єднуватиметься через своєрідний перехідник з роз'єму вилки штирової в ISA.

Виходячи з розглянутого раніше алгоритму роботи майбутнього пристрою, нам потрібні такі структурні частини, з наступним поясненням кожної більш докладно:

– блок центрального управління;

– блок центрального управління і оперування дійсними числами;

– блок установки, скидання та генерування синхроімпульсів;

– блок формування шини адрес;

– блок формування шини даних;

– блок формування шини управління;

– блок дешифрації адреси та формування сигналів вибірки кристала;

– блок обробки переривань;

– блок буферізації даних;

– блок зберігання постійних даних;

– блок зберігання оперативних даних;

– блок паралельного введення даних;

– блок зовнішньої синхронізації;

– блок послідовного виведення даних;

– блок перетворення рівнів ТТЛ.

Блок центрального управління має бути розроблений на основі центрального процесора. Основою контролера є центральний процесор, який забезпечує обслуговування всіх внутрішніх і зовнішніх блоків контролера і інформації, що приймається від них під управлінням робочої програми, що зберігається в блоці зберігання постійних даних. Ініціатором дій виконуваних пристроєм є мікропроцесор, який виробляє сигнали управління та видає їх на блок формування шини управління, що діють на блоки пам'яті (блоки зберігання постійної та оперативної пам'яті), блоки введення-виведення (блоки паралельного введення даних, послідовного виведення даних, зовнішньої синхронізації) та блок обробки переривань. Блок центрального управління виробляє адреси для блоку формування шини адрес, які надалі задають адреси для обраних осередків пам'яті, а також адресні входи блоків дешифрації та вводу-виводу. Як і вказувалося раніше, ми будемо використовувати шину даних об'ємом в слово (16 біт), тому розрядність даного процесора повинна мати можливість оперувати 16-розрядними даними. Блок центрального управління виробляє дані, які надходять на блок формування шини даних, так і приходять від нього, за допомогою передачі даних від блоків пам'яті і блоків введення-виведення.

Блок центрального управління та оперування речовими числами повинен бути розроблений на основі математичного співпроцесора, який зможе виконувати арифметичні операції над десятковими, цілими та числами з плаваючою комою. Так само, як і блок центрального управління, він повинен використовувати 16-розрядну шину даних. Він працюватиме паралельно з блоком центрального управління, що дозволяють в цілому організувати конвеєрну обробку даних, що приймаються, і розширювати математичні можливості блоку центрального управління відповідно до необхідних вимог реалізації нашого алгоритму пристрою.

Блок установки, скидання та генерування синхроімпульсів управляє роботою блоку центрального управління та блоку центрального управління та оперування речовими числами. Генерує тактуючий сигнал, що забезпечує синхронізацію роботи блоків центрального управління та оперування з речовими числами та інших периферійних блоків пристрою в цілому. Цей блок повинен складатися з генератора та схеми скидання, яка керує генератором. Блок формує сигнал «READY» службовець для індикації моменту, коли встановилася частота сигналу, що генерується, і сигналу «RESET» службовця для скидання блоку центрального управління та інших блоків системи.

Блок формування шини адрес бере участь у створенні шин адрес і сприяє підвищенню здатності навантаження шин блоків центрального управління. Шини являють собою провідні лінії (провідники), що з'єднують між собою складові частини всієї системи пристрою, які мають розрядність (кількістю провідників в одній шині). Блок формування шини адрес використовується для вказівки адреси осередків під час звернення до блоків пам'яті; адреси порту при зверненні до блоків введення-виводу; адреси контролера переривань при обміні даними між блоком обробки переривань та блоком центрального керування. Розрядність даного блоку залежить в основному від можливостей центрального блоку управління і необхідних розмірів блоків зберігання постійних та оперативних даних.

Блок формування шини даних бере участь у створенні шини даних та сприяє підвищенню здатності навантаження шин блоків центрального управління. Блок формування шини даних служить передачі даних що йдуть з блоків пам'яті і блоків введення-виведення до блоків центрального управління, або з блоків центрального управління в блоки пам'ять і блоки введення-виведення, а також обмін даними між блоками центрального управління і обробки переривань. Блок формує двонаправлену шину даних.

Блок формування шини управління бере участь у створенні шини управління та сприяє підвищенню здатності навантаження шин блоків центрального управління. Служить передачі керуючих сигналів, таких як читання даних, запис даних, при адресації вибору портів блоків вводу-виводу і пам'яті. Передає сигнали управління від центрального блоку управління до інших блоків системи, а також передачі сигналів запиту переривання від блоків введення-виводу до центрального блоку управління.

Блок дешифрації адреси та формування сигналів вибірки кристала займається тим, що вибирає всі зовнішні блоки введення-виводу та пам'яті, а також блок обробки переривань. Функція даного блоку, що виконується пристроєм, що працює в режимі програмного обміну, - це розпізнавання (дешифрація) власної адреси на магістралі. Цю функцію виконує вузол, званий селектором (дешифратором) адреси, який повинен виробити сигнали, що відповідають виставленню на шині адреси магістралі коду адреси, що належить даному блоку, або коду адреси із зони адрес цього пристрою.

Блок обробки переривань забезпечує узгодження сигналів запиту переривання, що надходять від зовнішньої шини управління, блоків введення-виведення та блоку центрального управління та оперування речовими числами з блоком центрального управління. Всі переривання в пристрої радіальні, тобто для переведення блоку центрального управління в режим обробки переривання досить надіслати запит, як виступає позитивний фронт сигналу на одній з ліній IRQ. Адреса в системній таблиці, за якою знаходиться адреса початку програми обробки переривання, однозначно визначається номером лінії IRQ, що використовується однією з ліній IRQ.

Блок буферизації даних застосовується для електричного узгодження та виконує основні функції: передача сигналів у потрібному напрямку (для двонаправлених сигналів) та конвертування 16-розрядної шини даних у 8-розрядну та назад. Конвертування необхідно, оскільки всі блоки вводу-виводу мають 8-розрядні входи (виходи) шини даних, то нам необхідно узгодити її з 16-розрядною шиною даних.

Блок зберігання постійних даних необхідний для зберігання та читання цифрової інформації: містить програму та константи, які визначають всі операції, необхідні та вироблені для роботи пристроєм.

Блок зберігання оперативних даних необхідний для зберігання, читання та запису цифрової інформації: допомагає в організації стеку центрального блоку управління та зберігання оперативної інформації (містить тимчасові дані).

Блок паралельного введення даних служить для паралельного обміну інформацією та управління блоками периферії за трьома 8-розрядними каналами A, B, C. Комп'ютер передає дані байтами (8 біт), тому даний блок

приймає паралельний 8-розрядний код від зовнішньої шини управління, в якому містяться дані для подальшої операції стиснення пристроєм.

Блок зовнішньої синхронізації необхідний ведення внутрішнього відліку часу, і навіть синхронізації блоку послідовного вывода. Швидкість передачі інформації визначається програмою та формується за допомогою даного блоку зовнішньої синхронізації.

Блок послідовного виведення даних перетворює паралельний кодований код в послідовний, при подальшій передачі в канал зв'язку. Оскільки пристрій буде оперувати даними в байтах (8 біт) або в словах (16 біт), то канал зв'язку дані повинні передаватися послідовно біт за бітом, щоб можна було обмежитися єдиним кабелем. Оскільки паралельна передача, хоч і є швидшою, зажадала б різкого збільшення апаратурних витрат (приймач і передавач для кожної лінії зв'язку). Тому таке перетворення необхідне. Перетворення послідовного коду в паралельний і навпаки можна було б реалізувати і програмним шляхом, з використанням команд арифметичного зсуву центрального блоку управління, що входить до складу пристрою, проте значне уповільнення передачі даних робить такий підхід практично неприйнятним.

Блок перетворення рівнів ТТЛ необхідний перетворення рівнів сигналів з логічних ТТЛ в мережеві (канал зв'язку) при передачі і з мережевих в логічні при прийомі. Необхідність даного структурного блоку полягає у перетворенні електричних сигналів одного рівня або виду (оброблюваного пристроєм стиснення) на інший (переданий по середовищі) і навпаки. Наприклад, сигналів рівня ТТЛ в послідовний канал передачі стандарту RS-232, яким контролер обмінюється із зовнішнім каналом зв'язку або іншими пристроями.

4 РОЗРОБКА ФУНКЦІОНАЛЬНОЇ СХЕМИ ЕЛЕКТРОННОЇ СИСТЕМИ

На основі структурної схеми і блок схеми алгоритму функціонування пристрою, спроектуємо функціональну схему устрою, що розробляється. Вибір функціональних вузлів пристрою здійснимо, використовуючи наведений раніше перелік виконуваних функцій структурних блоків устрою.

Основним завданням функціональної схеми є показати функції, які виконує той чи інший блок логічної структури пристрою. Тому функціональна схема буде комбінацією структурної схеми пристрою та алгоритму роботи самого пристрою.

Функціональна схема має дозволити вирішити завдання, які задані та описані при проектуванні структурної схеми. А саме, необхідно розробити пристрій стиснення даних на основі арифметичного кодування, яке має виконувати такі функції: формування сигналів-запитів із зовнішньою шиною управління та каналом зв'язку; перевірка наявності даних із зовнішньої шини управління; прийом від зовнішньої шини керування даних для подальшого стиснення; зберігання даних; проведення над отриманими даними математичних операцій згідно з алгоритмом арифметичного кодування; зберігання одержаного результату обчислення закодованого повідомлення; підготовка до надсилання даних у канал зв'язку; передача до каналу зв'язку закодованого повідомлення.

Функціональна схема розроблюваного пристрою повинна включати основні підсистеми, описані раніше блоками в структурній схемі, опишем значення вхідних і вихідних функціональних висновків для деяких з них, а також для всіх значення вхідних і вихідних сигналів для них.

Для створення мікропроцесорного блоку пристрою стиснення даних на основі арифметичного кодування будемо використовувати мікропроцесорний комплект серії Intel 8086 (вітчизняний аналог КР1810).

Блок установки, скидання та генерування синхроімпульсів виконує функції управління роботою блоків центрального управління та периферійних пристроїв, а саме генерування тактових сигналів, сигналів готовності та скидання. Блок повинен складатися з генератора і схеми установки і скидання, котра управляє режимами генератора. Функції сигналів, що подаються на

входи блоку: $\overline{RES}$ сигнал установки, що йде із зовнішньої шини управління; $\overline{HD}$ сигнал установки, що йде із блоку зовнішньої синхронізації; $\overline{AEN}$ сигнал адреси готовності шини, що подається із зовнішньої шини управління; RDY сигнал готовності шини (на який постійно подається логічна одиниця, що також йде і на блок паралельного введення даних $PC1$). Сигнали, що формуються цим блоком: CLK тактовий сигнал, який подається для синхронізації блоків центрального керування та блоку формування шини керування; $RESET$ сигнал початкової установки для блоків центрального управління, а також всіх інших блоків, що потребують скидання у вихідний стан; $READY$ сигнал готовності, керуючий режимом очікування блоків центрального керування; $PCLK$ периферійний тактовий сигнал, призначений для синхронізації зовнішніх пристроїв, а саме для синхронізації із зовнішньою шиною керування та блоку зовнішньої синхронізації.

Блоки центрального управління та центрального управління та оперування речовими числами, представлені на рисунку 4.1, які повинні виконувати функції: керування всім пристроєм; забезпечувати прийом, видачу, зберігання та обробку даних, адрес; видавати сигнали стану, що діють на шину керування; узгоджувати спільну роботу з виконання заданої програми з алгоритму арифметичного кодування. Блоки складаються з 16-розрядного мікропроцесора (МП) CPU та арифметичного співпроцесора NPX.

Від блоку установки, скидання та генерування синхроімпульсів подаються сигнали CLK, $RESET$, $READY$ на входи блоків центрального управління виконуючі функції: синхроімпульсів CLK, початкової установки SR і готовності RDY. Виходячи з використання кількох процесорів, режим у якому контролер працюватиме – максимальний, тому логічний нуль підключений до функціонального входу $MN/\overline{MX}$. Для узгодження спільної роботи блоків центрального управління, функціональні входи та виходи з'єднані, а саме: $\overline{TEST}$ і $\overline{BUSY}$; $RQ/\overline{GT0}$ і $RQ/\overline{GT0}$; $QS0$, $QS1$ і $QS0$, $QS1$. Для обробки переривань, що маскуються, сигнал INT підключений від блоку обробки переривання до блоку центрального керування. У разі передачі роботи з обробки арифметичних даних від процесора до співпроцесора співпроцесор формує сигнал INT що йде до блоку обробки переривань, який генерує переривання $IRQ0$.

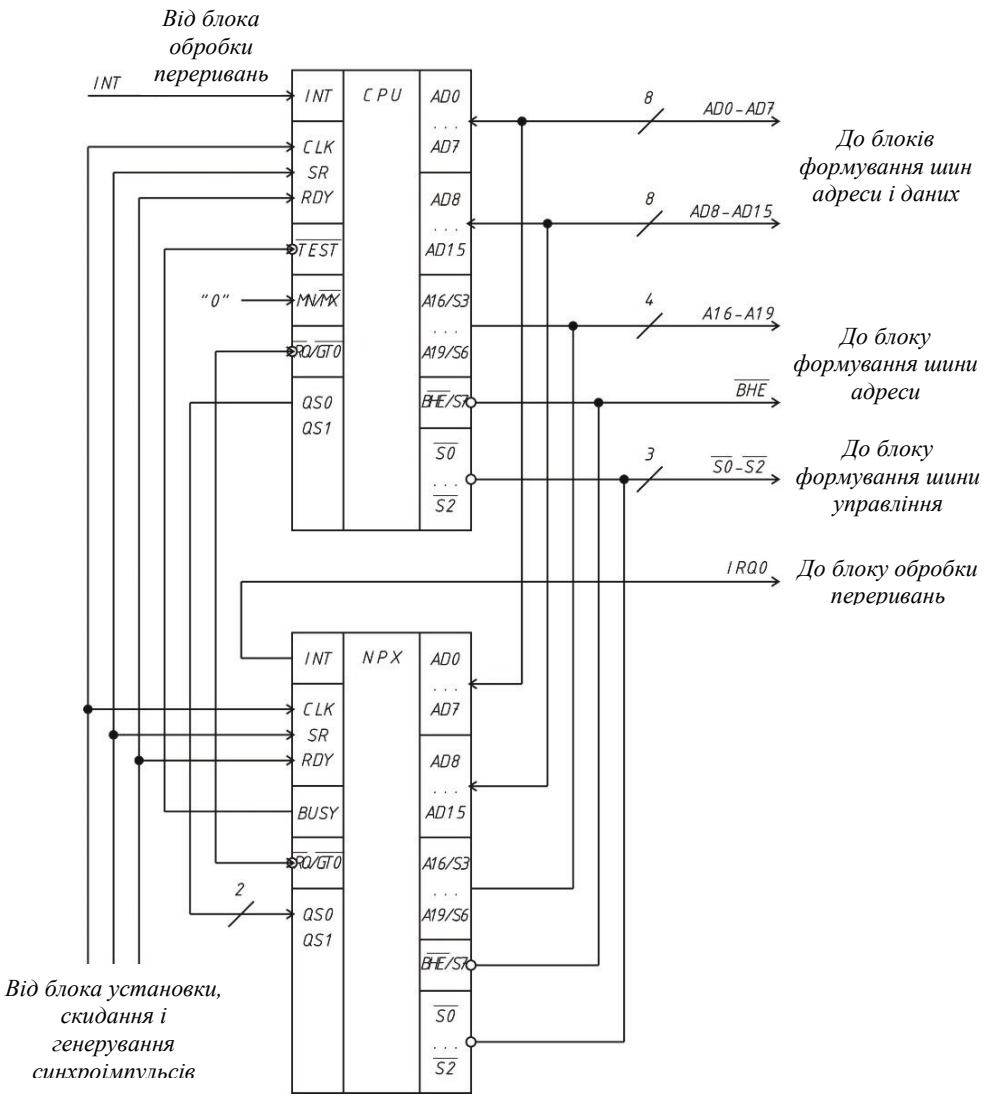


Рисунок 4.1 – Функціональна схема блоків центрального управління та центрального управління та оперування дійсними числами

Обидва блоки з'єднані однаковими входами/виходами в декілька сформованих шин: 8-розрядну шину адреси/даних $AD0-AD7$; 8-розрядну шину адреси/даних $AD8-AD15$; 4-розрядну шину адреси $A16-A19$; 3-розрядну шину стану сигналів процесорів $\overline{S0}-\overline{S2}$; сигналу вибору старшого банку пам'яті $\overline{BHE}$. Шини $AD0-AD7$ і $AD8-AD15$ йдуть (у різних тактах роботи процесорів) на блок формування шини адреси та блок формування шини даних. Шина $A16-A19$ і сигнал $\overline{BHE}$ блок формування шини адреси. Шина $\overline{S0}-\overline{S2}$ на блок формування шини керування.

Блок формування шини керування представлений на рисунку 4.2 і виконує функції формування 4-розрядної односпрямованої шини управління,

яка перетворює сигнали процесора в сигнали читання/запису пам'яті та зовнішніх пристроїв, а також блок виконує управлінням шинних формувачів адреси/даних та блоком обробки переривань. Блок складається із системного контролера *СВ*.

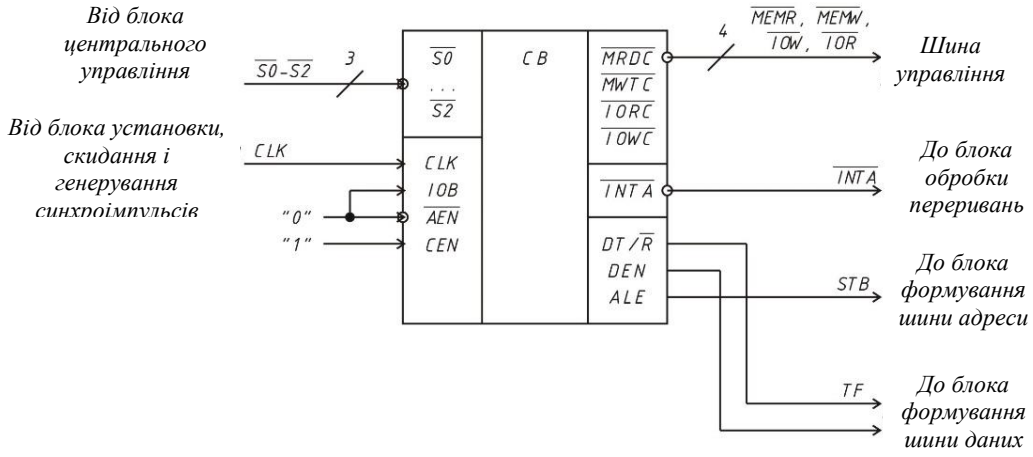


Рисунок 4.2 – Функціональна схема блоку формування шини керування

Функції сигналів на його входах: синхросигнал CLK; 3-розрядна шина стану сигналів процесорів $\overline{S0}-\overline{S2}$; логічний нуль на входи $\overline{AEN}$ (сигнал управління станом «вимкнено» командних виходів) та IOB (сигнал вибору режиму роботи з шиною вводу/виводу); логічна одиниця на CEN (сигнал управління станом командних виходів та контрольного виходу DEN). Сигнали, що формуються цим блоком: $\overline{MWTC}$ командний сигнал запису в пам'ять, $\overline{MRDC}$ командний сигнал зчитування з пам'яті, $\overline{IOWC}$ командний сигнал запису в пристрій введення/виводу, $\overline{IORC}$ командний сигнал зчитування з пристрою введення/виводу відповідають відповідно сигналам $\overline{MEMW}$, $\overline{MEMR}$, $\overline{IOW}$, $\overline{IOR}$, які формують шину керування пристрою; $\overline{INTA}$ сигнал підтвердження переривання йде до блоку обробки переривань; ALE стробуючий сигнал запису йде до блоку формування шини адреси; $DT/\overline{R}$ сигнал управління роботою та DEN сигнал управління станом "вимкнено" шинних формувачів подаються до блоку формування шини даних.

Блок формування шини даних, представлений на рисунку 4.3, виконує функції формування 16-розрядної двонаправленої шини даних. Блок

складається з двох 8-розрядних буферних формувачів та простої логічної схеми. Функції сигналів, що подаються на цей блок: $DT/\overline{R}$ и DEN (зазначені раніше) від блоку формування шини керування; $\overline{MS}/\overline{SV}/\overline{DE}$ сигнал дозволу даних від блоку обробки переривань. З блоків центрального управління за двома 8-розрядними шинами адреси/даних $AD0-AD7$ і $AD8-AD15$ відбувається обмін даними із блоками формування шини даних. Сигнали, що формуються цим блоком, є двома 8-розрядними шинами даних. $D0-D7$, $D8-D15$, які утворюють 16-розрядну двонаправлену шину даних $D0-D15$ циркулюючу по всьому пристрою.

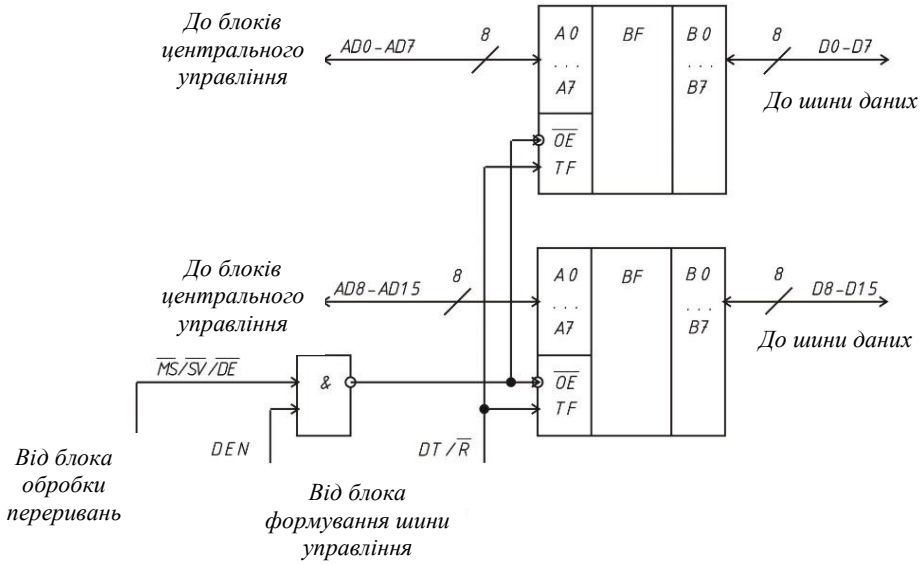


Рисунок 4.3 – Функціональна схема блоку формування шини даних

Блок формування шини адрес, представлений на рисунку 4.4 виконує функції формування 21-розрядної односпрямованої шини адреси. Блок складається з трьох 8-розрядних регістрів-засувок RG. Функції сигналів, що подаються на цей блок: ALE стробуючий сигнал запису від блоку формування шини управління на входи STB; логічний нуль на входи $\overline{OE}$.

На входи $DI0...DI7$ від блоків центрального управління подаються дві 8-розрядні шини адреси/даних $AD0-AD7$ і $AD8-AD15$, 4-розрядна $A16-A19$ та сигнал вибору старшого банку пам'яті $\overline{BHE}$. Сигнали, які формує цей блок, подаються з виходів $DO0...DO7$ та формують дві 8-розрядні $A0-A7$, $A8-A15$ шини адреси; 4-розрядну $A16-A19$ шину адреси і сигнал

вибору старшого банка пам'яті $\overline{BHE}$, разом утворюючу 21-розрядну шину адреси.

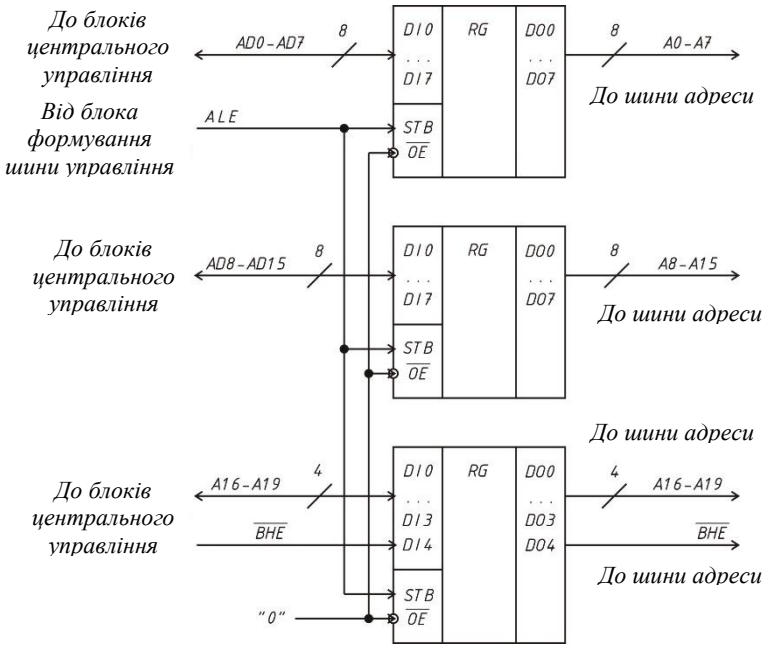


Рисунок 4.4 – Функціональна схема блоку формування шини адреси

Блок дешифрації адреси та формування сигналів вибірки кристала, виконує функції організації вибірки ВІС у контролері, а саме блоків пам'яті та блоків введення-виведення. Блок повинен складатися з кількох дешифраторів та кількох простих логічних схем. Функції сигналів, що подаються на цей блок: $A0$, $A5$, $A6$, $A14$, $A15$ адресні лінії шини адреси, за якими проводиться дешифрація; $\overline{BHE}$ сигнал вибору старшого банку пам'яті; $\overline{MEMW}$, $\overline{MEMR}$, $\overline{IOW}$, $\overline{IOR}$ сигнали керування шини керування. Сигнали, які формує цей блок: $\overline{ROMS}$ сигнал вибірки кристалів блоку зберігання постійних даних; $\overline{RAMS1}$, $RAMS2$ сигнали вибірки кристалів блоку зберігання оперативних даних; $\overline{PIC}$ сигнал вибірки кристала блоку обробки переривань; $\overline{PPI}$ сигнал вибірки кристала блоку паралельного введення даних; $\overline{PCT}$ сигнал вибірки кристала блоку зовнішньої синхронізації; $\overline{STD}$ сигнал вибірки кристала блоку послідовного виведення даних

Блок зберігання незмінних даних, представлений на рисунку 4.5 виконує функції зберігання коду програми та постійних констант пристрою, за якими

працює контролер. Блок складається з двох 8-розрядних ППЗП $EPROM$, кожна об'ємом 16 Кбайт. Функції сигналів, що подаються на цей блок: 13-розрядна шина адреси $A1-A13$, що йде одночасно на обидві ППЗП по адресним входам $A0-A12$; $\overline{ROMS}$ сигнал вибірки кристала, що приходить від блоку дешифрації адреси і подається на вхід $\overline{CS1}$ даного блока; $\overline{MEMR}$ сигнал із шини керування, призначений для дозволу читання з блоків пам'яті, що приходить на входи $\overline{OE}$; на входи $\overline{PR}$ блоку зберігання постійних даних подається логічна одиниця, оскільки ми не програмуватимемо дані модулі пам'яті (заздалегідь запрограмовані).

З виходів $D0...D7$ даного блока формуються дві 8-розрядні шини даних $D0-D7$ і $D8-D15$, що йдуть до шини даних.

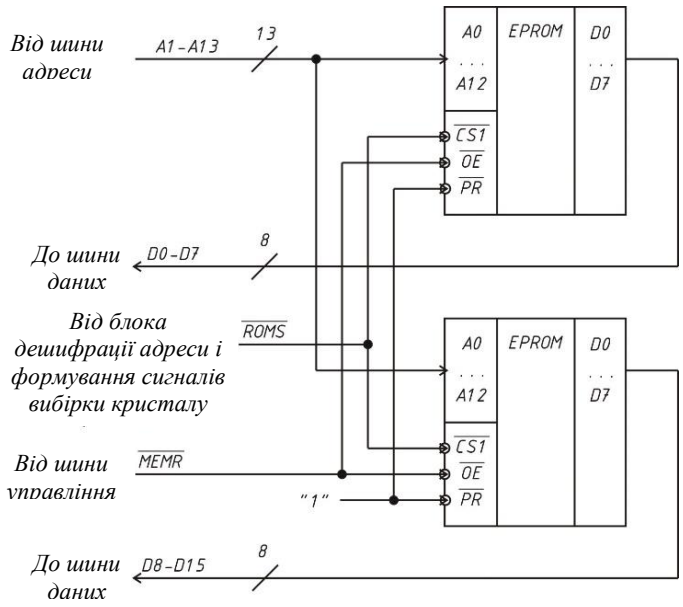


Рисунок 4.5 – Функціональна схема блоку зберігання постійних даних

Блок зберігання оперативних даних ілюстрований на рисунку 4.6, виконує функції зберігання, запису та видачі вхідних змінних, проміжних значень та результату обчислення. Блок складається з двох 8-розрядних ОЗП RAM, кожна обсягом 16 Кбайт. Функції сигналів, що подаються на блок: 13-розрядна шина адреси $A1-A13$, що йде одночасно на обидві ОЗП по адресних входах $A0-A12$; $\overline{RAMS1}$, $RAMS2$ сигнали вибірки кристала, що йдуть від блоку дешифрації адреси та під'єднані на входи ОЗУ $\overline{CS1}$ і $CS2$; $\overline{MEMR}$, $\overline{MEMW}$

сигнали з шини управління, для читання або запису з/в блок(а) пам'яті, що йдуть на входи $\overline{OE}$, $\overline{WR}$.

Сигнали, що формуються цим блоком, передаються та приймаються на входи/виходи $D0...D7$, що складаються із двох 8-розрядних шин даних, які обмінюються інформацією з 16-розрядною шиною даних.

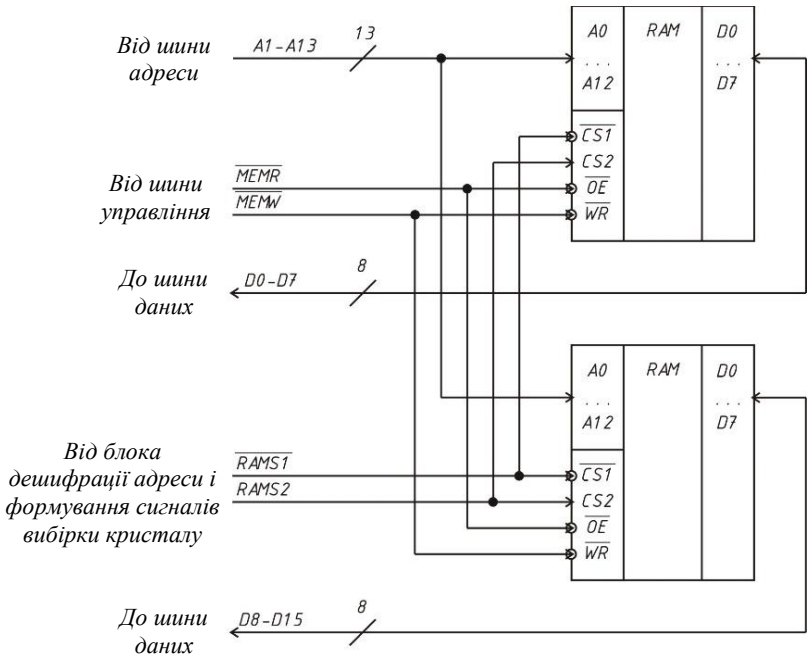


Рисунок 4.6 – Функціональна схема блоку зберігання оперативних даних

Блок обробки переривань показаний на рисунку 4.7 виконує функції переривання пристрою, що реалізують 8-рівневу векторну систему переривань з можливістю маскування та динамічної зміни дисципліни обслуговування, для організації обміну інформації в режимі переривання та програмно-керованого обміну.

Блок складається з 8-розрядного контролера переривання PIC. Функції сигналів, що подаються на цей блок: INT сигнал, що йде від арифметичного співпроцесора і викликає переривання $IRQ0$; $IRQ1$ сигнал, що йде від зовнішньої шини управління і викликає переривання $IRQ1$; $TxRDY$, $RxRDY$ сигнали, що йдуть від блоку послідовного виведення даних і викликають переривання $IRQ2$ і $IRQ3$; $IRQ4$ сигнал, що йде від блоку зовнішньої синхронізації і викликає переривання $IRQ4$; $\overline{PIC}$ сигнал вибірки кристала;

$\overline{IOW}$, $\overline{IOR}$ сигнали керування читання/запису пристроїв введення/виводу шини керування; $\overline{INTA}$ сигнал підтвердження переривання, що йде від блоку формування шини керування; $A1$ сигнал для адресного входу з адресної шини. Сигнали, що формуються цим блоком: 8-розрядна двонаправлена шина даних $D0-D7$, сполучена з 16-розрядною шиною даних (перетворення розрядності не потрібно); INT сигнал переривання, що звертається до центрального процесора; $\overline{MS}/\overline{SV}/\overline{DE}$ сигнал роздільної здатності даних, що йде до блоку формування шини даних.

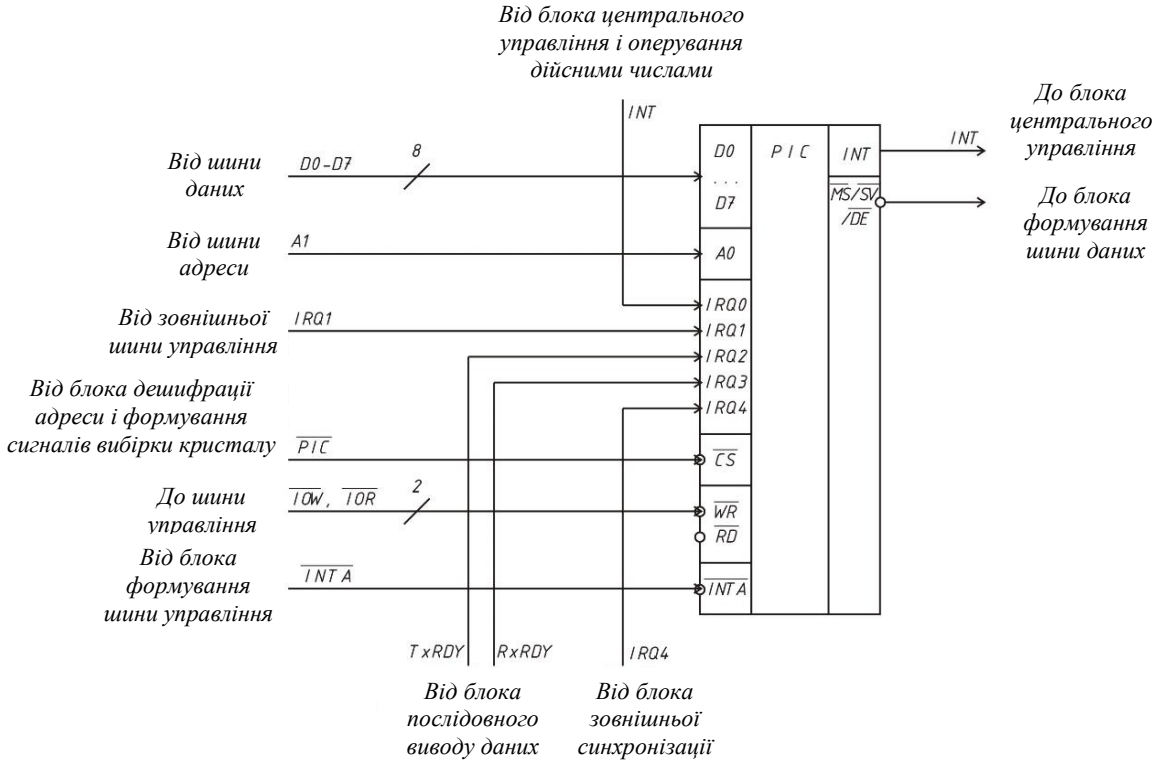


Рисунок 4.7 – Функціональна схема блоку обробки переривань

Блок паралельного введення даних представлений на рисунку 4.8 виконує функції введення/виведення паралельної інформації різного формату, для управління зовнішніми пристроями, а також для забезпечення необхідної швидкості введення коду в пам'ять.

Блок складається з одного програмованого паралельного інтерфейсу PPI, містить три 8-розрядних портів вводу/виводу. Функції сигналів, що подаються на блок: $\overline{IOW}$, $\overline{IOR}$ сигнали керування читання/запису пристроїв введення/виводу шини керування; $\overline{PPI}$ сигнал вибірки кристала; $RESET$

сигнал приведення пристрою в початковий стан, що йде від генератора синхроімпульсів; $A1$, $A2$ сигнали для адресного входів $A0$, $A1$ з шини адреси, що задають адресу порту; $PB0-PB7$ шина даних, по якій подається інформаційний код для стиснення від зовнішньої шини управління; RDY сигнал, що йде від генератора синхроімпульсів, установки входу $PC1$ з порту PC в одиничне становище.

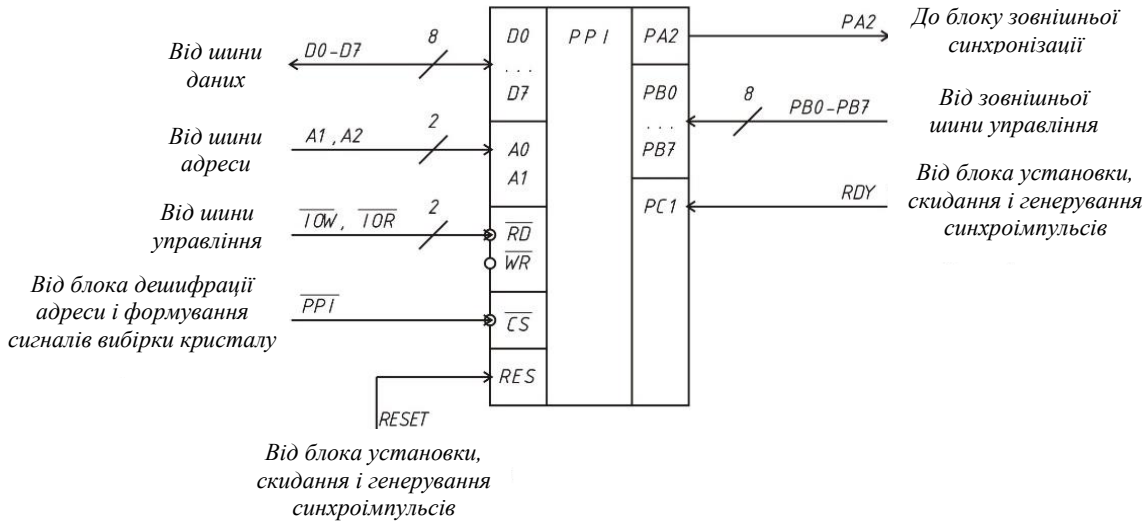


Рисунок 4.8 – Функціональна схема блоку паралельного введення даних

Блок складається з одного програмованого паралельного інтерфейсу PPI, містить три 8-розрядних портів вводу/виводу. Функції сигналів, що подаються на блок: $\overline{IOW}$, $\overline{IOR}$ сигнали керування читання/запису пристроїв введення/виводу шини керування; $\overline{PPI}$ сигнал вибірки кристала; $RESET$ сигнал приведення пристрою в початковий стан, що йде від генератора синхроімпульсів; $A1$, $A2$ сигнали для адресного входів $A0$, $A1$ з шини адреси, що задають адресу порту; $PB0-PB7$ шина даних, по якій подається інформаційний код для стиснення від зовнішньої шини управління; RDY сигнал, що йде від генератора синхроімпульсів, установки входу $PC1$ з порту PC в одиничне становище.

Сигнали, що формуються цим блоком: $PA2$ сигнал з порту PA, що йде для керування блоком зовнішньої синхронізації; 8-розрядна двонаправлена шина даних $D0-D7$, що йде на блок буферизації даних.

Блок зовнішньої синхронізації виконує функції генератора синхросигналів для блоку послідовного виведення даних та завдання ритму роботи для блоку установки скидання та генерування синхроімпульсів. Блок складається з триканального програмованого таймера та простої логічної схеми. На його вхід подаються сигнали: $\overline{PCT}$ сигнал вибірки кристала; $\overline{IOW}$, $\overline{IOR}$ сигнали керування читання/запису пристроїв введення/виводу шини керування; $A1$, $A2$ сигнали з шини адреси, що задають адресу вибраного таймера каналу; $PA2$ сигнал керування, що йде від блоку паралельного введення даних; HD сигнал керування, що йде від зовнішньої шини керування контролера; $PCLK$ периферійний сигнал, що йде від блоку установки скидання та генерування синхроімпульсів. Цей блок формує такі сигнали: CC сигнал синхронізації, що йде на входи блоку послідовного виведення даних, який задає швидкість, з якою дані будуть передаватися канал зв'язку; $\overline{HD}$ сигнал управління, що йде на блок установки, скидання та генерування синхроімпульсів; $IRQ4$ сигнал, що йде на блок обробки і викликає переривання $IRQ4$; 8-розрядна двонаправлена шина даних $D0-D7$, що йде на блок буферизації даних.

Блок послідовного виведення даних представлений на рисунку 4.9 і виконує функції організації обміну даних між МП і каналом зв'язку в послідовному форматі. Блок складається з універсального синхронно-асинхронного приймача. Функції сигналів, що подаються на цей блок: RxD дані (вхідні послідовні дані з каналу зв'язку); CC сигнали синхроімпульсів, що йдуть від блоку зовнішньої синхронізації на входи TxC і RxC, які необхідні тактування пристрою під час прийому/передачі послідовних сигналів; $PCLK$, $RESET$ периферійний тактовий сигнал і сигнал початкової установки блоку, що йдуть на входи CLK і $RESET$ відповідно від блоку установки, скидання та генерування синхроімпульсів; $A1$ сигнал адреси з шини адреси, що йдуть на вхід C/D та вибираючий режими роботи приймача; $\overline{IOW}$, $\overline{IOR}$ сигнали від шини керування, що йдуть на входи $\overline{WR}$, $\overline{RD}$ для роботи читання/запису даних; $\overline{STD}$ сигнал вибірки кристала, що йде від блоку дешифрації адреси на вхід $\overline{CS}$.

Сигнали, що формуються цим блоком: TxD дані, що передаються (вихідні послідовні дані в канал зв'язку), що подаються на блок перетворення

рівнів ТТЛ; *TxRDY*, *RxRDY* сигнали готовності до передачі або прийому даних, що йдуть до входів, блоку обробки переривань, що викликають переривання *IRQ*2 и *IRQ*3; $D0-D7$ 8-розрядна двонаправлена шина даних, що обмінюється даними з блоком буферизації даних, приєднана до входів/виходів $D0...D7$.

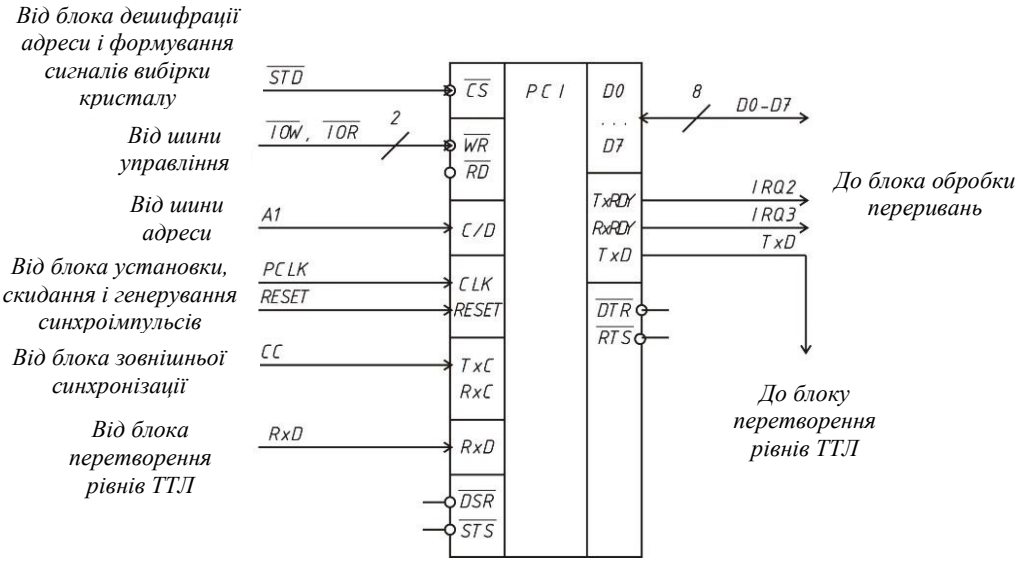


Рисунок 4.9 – Функціональна схема блоку послідовного виведення даних

Блок буферизації даних виконує функції двонаправленого перетворення 16-розрядної шини даних $D0-D15$ в 8-розрядну $D0-D7$, для забезпечення доступу 8-розрядних шин периферії до 16-розрядної шини даних процесора. Блок повинен складатися із шинних формувачів. Функції сигналів, що подаються на цей блок: $\overline{IOW}$ управління від шини даних; $A0$, $\overline{BHE}$ шини адрес; $\overline{PPI}$, $\overline{PCT}$, $\overline{STD}$ від блоку дешифрації адреси та формування сигналів вибірки кристала для активації блоку буферизації при використанні одного з пристрою периферії (крім контролера переривання). Всі сигнали, що подаються, сприяють виконанню функції передачі.

Блок перетворення рівнів ТТЛ виконує функцію узгодження рівнів сигналів пристрою, заснованих на логікі ТТЛ з каналом зв'язку заснованому на інтерфейсі RS-232, і може складатися з аналогових транзисторних каскадів. Функції сигналів, що подаються на цей блок: *TxD* сигнал, який надходить від блоку послідовного виведення даних, що містить дані для подальшого перетворення та передачі їх у канал зв'язку; *RRS* сигнал, отриманий з каналу

зв'язку, містить дані для подальшого перетворення сигнал рівнів ТТЛ і передачі їх на блок послідовного виведення даних. Сигнали, що формуються цим блоком: *RxD* сигнал, який надходить на блок послідовного виведення даних, що містить перетворені дані у рівнях ТТЛ з каналу зв'язку; *TRS* сигнал, що передається у канал зв'язку, що містить перетворені дані в рівні інтерфейсу RS-232.

5 РОЗРОБКА СХЕМИ ЕЛЕКТРИЧНОЇ ПРИНЦИПОВОЇ ЕЛЕКТРОННОЇ СИСТЕМИ

5.1 Вибір елементної бази

У зв'язку з поставленими умовами вирішення нашого завдання реалізації алгоритму арифметичного кодування, а також труднощами в процесі оперування числами з плаваючою комою, доцільним було б використовувати в якості елементної бази мікропроцесорний комплект серії Intel 8086, сумісний за командами і функціональністю з вітчизняним мікропроцесорним комплектом КР1810, заснований на n-МОП технології. Дане рішення використовувати саме цей комплект, зважаючи на його застарілу серію в порівнянні з іншими доступними в сучасному світі мікропроцесорними комплектами, а також наявністю великого обсягу можливих мікроконтролерів, обумовлено обмеженнями, які ставить наш алгоритм на пристрій у вигляді двох необхідних складових. Це велика розрядність чисел, що оперуються, і власне оперування речовими числами. Використовуючи обраний мікропроцесорний комплект, ми можемо використовувати центральний процесор, а також найнеобхіднішу частину реалізації алгоритму співпроцесора, який безпосередньо оперує роботою з речовими числами.

В якості елементної бази для побудови пристрою, згідно [17], були обрані комплекти мікросхем серії 74ALS (КР1533) і I8080 (радянський аналог КР580, зважаючи на успадковану сумісність), як сумісні рекомендовані комплекти з будь-яким із входів/виходів мікропроцесорного комплекту I8086. Виконані за ТТЛШ-технології, характеризуються архітектурною єдністю, що забезпечується автономністю та функціональною закінченістю окремих мікросхем, уніфікацією їх процесу, програмованістю мікросхем, їхньою логічною та електричною сумісністю.

Інтерфейсні елементи деяких мікросхем вибраних комплектів мають виходи з трьома станами, що є необхідним при побудові систем із шинною організацією. Вхідні та вихідні струми, а також тимчасові затримки сигналів мікросхем цих серії задовольняють вимогам до буферних елементів, що працюють з неповною магістраллю ISA. Також використані мікросхеми пам'яті серії КР537 та КР573, які також відповідають необхідним параметрам. Деякі параметри ІМС, що використовуються, наведені в таблиці 5.1 (значення

статичних параметрів, що відрізняються від зазначених, наведені в таблицях параметрів конкретних мікросхем).

Таблиця 5.1 – Деякі параметри застосовуваних мікросхем

Параметр, одиниця вимірювання	Значення			
Серія мікросхем	I8086, I8080 (КР1810, КР580), макс(мін)	74ALS (КР1533), макс(мін)	КР537, КР573, макс(мін)	
Напруга живлення, В (U_{CC})	5,25 (4,75)	5,5 (4,5)	5,5 (4,5)	
Вхідна напруга, В логічного нуля (U_{IL}) логічної одиниці (U_{IH})	0,8 (2,0)	0,8 (2,0)	0,8 (2,2)	0,8 (2,0)
Вихідна напруга, В логічного нуля (U_{OL}) логічної одиниці (U_{OH})	0,45 (2,4)	0,4 (2,4)	0,4 (2,4)	
Вихідний струм, мА логічного нуля (I_{OL}) логічної одиниці (I_{OH})	2,0 - 0,4	8,0 - 0,4	4,0 -1,0	2,1 - 0,4
Споживаний струм, мА (I_{CC}) логічного нуля логічної одиниці	230,0	3,0 0,85	60,0	30,0
Вхідний струм, мА логічного нуля (I_{IL}) логічної одиниці (I_{IH})	- 0,5 0,06	- 0,2 0,02	- 0,25 0,04	
Час перехода, нс 0, 1 1, 0	10,0 10,0	13,0 13,0	-	
Час вибірки, нс дозволу адреси	-	-	35,0 70,0	50,0 75,0
Емність навантаж., пФ (C_L)	100,0	200,0	100,0	

Комплект мікросхем серії I8086 з 16-розрядною організацією призначений для побудови засобів обчислювальної техніки різного призначення: від одноплатних мікро-ЕОМ, що управляють, до мультипроцесорних високопродуктивних систем. Широкі можливості мікропроцесорного комплекту (МПК) забезпечуються як досконалістю

архітектури центрального процесорного елемента, і набором функцій, виконуваних усіма мікросхемами, які входять у комплект (склад даних МПК серій представлений у таблиці 5.2). Ми будемо використовувати зарубіжний аналог – серії I8086, I8080 фірми Intel. Конструктивно мікросхеми серій I8086, I8080 виконані в 18-, 20-, 28-вивідних полімерних та 40-висновних металокерамічних корпусах типу 2104.18-5, 2140.20-2, 2121.28-5 та 2123.

Таблиця 5.2 – Склад МПК серій I8086 і I8080

Тип мікросхеми	Функціональне призначення	Тип корпуса	Технологія
I8086 (КР1810ВМ86)	Однокристальний 16-розрядний мікропроцесор	2123.40-6	n-МОП
I8087 (КР1810ВМ87)	Арифметичний сопроцесор	2123.40-6	n-МОП
I8284А (КР1810ГФ84)	Генератор тактових імпульсів	2104.18-5	ТТЛШ
I8286 (КР1810ВА86)	Шинний формувач без інверсії	2140.20-2	ТТЛШ
I8282 (КР1810ИР82)	Буферний регістр-защіпка без інверсії	2140.20-2	ТТЛШ
I8288 (КР1810ВГ88)	Контролер системної шини	2140.20-2	ТТЛШ
I8259А (КР1810ВН59А)	Програмований контролер переривань	2121.28-5	n-МОП
I8282 (КР1810ИР82)	Регістр- защіпка без інверсії	2140.20-2	ТТЛШ
I8251А (КР580ВВ51)	Програмований послідовний інтерфейс	2121.28-5	n-МОП
I8253А (КР580ВИ53)	Програмований інтервальний таймер	2123.40-2	n-МОП
I8255А (КР580ВВ55)	Програмований контролер паралельного введення-виведення	2123.40-2	n-МОП

Малопотужні швидкодіючі цифрові інтегральні мікросхеми серії КР1533 призначені для організації високошвидкісного обміну та обробки цифрової інформації, тимчасового та електричного узгодження сигналів у обчислювальних системах. Мають, порівняно з відомими серіями логічних ТТЛ мінімальним значенням твору швидкодії на розсіювану потужність. Ми

будемо використовувати закордонний аналог - серія SN74ALS фірми Texas Instruments. Конструктивно мікросхеми серії КР1533 виконані в 14- та 16-вивідних пластмасових корпусах типу 201.14-1 та 238.16-1 [19]. Для модулів пам'яті КР537 і КР573 ми будемо використовувати зарубіжні аналоги - UT6264 Utron [20] та M27C64A STmicroelectronics [21] (склад даних МПК серій представлений у таблиці 5.3).

Таблиця 5.3 – Склад МПК серій КР1533, КР537 і КР573

Тип мікросхеми	Функціональне призначення	Тип корпуса	Технологія
SN74ALS00AN (КР1533ЛА3)	Чотири логічних елементів 2І-НЕ	201.14-1	ТТЛШ
SN74ALS138N (КР1533ИД7)	Дешифратор/демультіплексор 3 на 8	238.16-1	ТТЛШ
SN74ALS02A (КР1533ЛЕ1)	Чотири логічних елементів 2АБО-НЕ	201.14-1	ТТЛШ
SN74ALS04B (КР1533ЛН1)	Шість логічних елементів НЕ	201.14-1	ТТЛШ
SN74ALS10AN (КР1533ЛА4)	Три логічних елементи 3И-НЕ	201.14-1	ТТЛШ
M27C64A (КР573РФ4)	Пам'ять ППЗП з УФ-стиранням 8К*8	2121.28-8	ЛИПЗ МОП
UT6264CPCL-70LL (КР537РУ17)	Пам'ять ОЗП 8К*8	2121.28-8	КМОП

5.2 Розробка основних електронних вузлів

Центральні блоки управління складаються з процесора І8086 (DD2) та співпроцесора І8087 (DD3).

Однокристальний високопродуктивний 16-розрядний МП І8086 [22] виконаний за n-МОП-технологією та має архітектуру, однотипну з вітчизняним МП КР1810ВМ86. Напруга живлення мікросхеми +5, струм споживання не більше 340 мА.

Мікропроцесор має високу швидкодію, забезпечує можливість прямої адресації пам'яті об'ємом до 1 МБайт, 65536 пристроїв введення та 65536 виведення. Для обчислення адрес операндів, розміщених у пам'яті,

використовується 24 режими адресації. Мікропроцесор має векторну структуру переривання та забезпечує обробку 256 запитів переривання трьох типів: зовнішніх, внутрішніх та програмних.

Архітектурною особливістю мікропроцесора I8086 є наявність апаратно-програмних засобів, що дозволяють спростити побудову мультипроцесорних систем на його основі. Ці засоби забезпечують синхронізацію роботи кількох незалежних (що виконують власні потоки команд) процесорів, що мають спільні ресурси, а також синхронізацію паралельної роботи мікропроцесора та співпроцесорів (спеціалізованих процесорів, які апаратно реалізують команди складних процедур).

В залежності від вимог, що пред'являються до системи, МП I8086 включається в мінімальному або максимальному режимі, які відрізняються способом формування сигналів обміну і можливостями систем, що реалізуються. Мінімальний режим призначений для застосування однопроцесорних систем, максимальний орієнтований на складні одно- та мультипроцесорні системи. Оскільки ми використовуємо кілька процесорів (центральний процесор I8086 та співпроцесор I8087), ми будемо використовувати максимальний режим. Під час роботи у цьому режимі змінюються функції деяких висновків МП. Для максимального режиму виведення підключається до логічного нуля. Умовне графічне позначення мікросхеми у максимальному режимі наведено на рисунку 5.1 призначення висновків, що використовуються в максимальному режимі, представлене в таблиці 5.4 [17; 22].

Усього в МП I8086 є 20 адресних ліній. За чотирма старшими адресними лініями в мультиплексному режимі передаються ще й сигнали стану системи. Лінії утворюють 16-розрядну мультиплексну шину адрес та даних.

Фізично область пам'яті МП I8086 організується як два банки (старший та молодший) по 512 Кбайт кожен. Для адресації осередків пам'яті кожному банку використовуються розряди шини адреси МП.

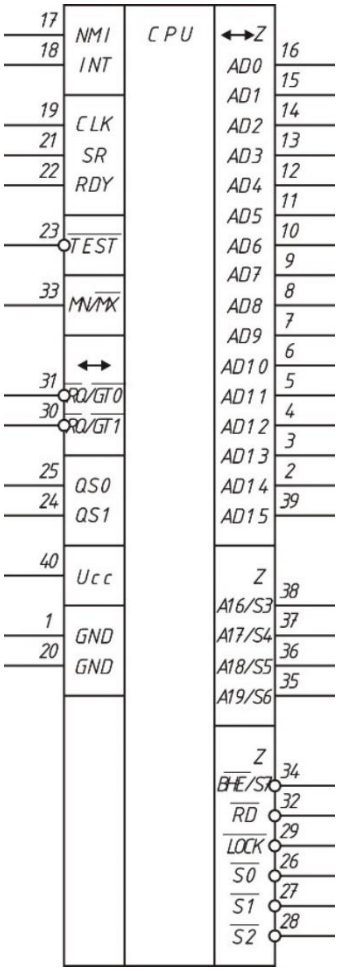


Рисунок 5.1 – Умовне графічне позначення МП I8086

Таблиця 5.4 – Функціональне призначення виводів мікропроцесора I8086 у максимальному режимі

Номер виводу	Позначення	Функціональне призначення виводів
1, 20	GND	Загальний
16-2, 39	$AD0-AD15$	Входи/виходи шини адреси/даних
17	NMI	Вхід запиту переривання, що не маскується.
18	INT	Вхід запиту переривання, що маскується.
19	CLK	Вхід тактових імпульсів
21	SR	Вхід сигналу "Установка" (скидання)
22	RDY	Вхід сигналу «Готовність»
23	$\overline{TEST}$	Вхід сигналу «Перевірка»
32	$\overline{RD}$	Вихід сигналу «Читання»
33	$MN/\overline{MX}$	Вхід мінімального/максимального режиму
34	$\overline{BHE}/S7$	Вихід дозволу старшого байта шини / сигнали стану
38-35	$A16/S3-A19/S6$	Виходи старших розрядів адреси / сигнали стану

Завершення таблиці 5.4

Номер виводу	Позначення	Функціональне призначення виводів
40	Ucc	Напруга живлення +5 В
25, 24	$QS0$, $QS1$	Виходи стану черги команд
26-28	$\overline{S0} - \overline{S2}$	Виходи типу циклів обміну
29	$\overline{LOCK}$	Вихід зайнятості каналу
31, 33	$\overline{RQ}/\overline{GT0}$, $\overline{RQ}/\overline{GT1}$	Входи/виходи запиту/дозвіл доступу до шини

МП формує сигнали $\overline{BHE}$ і $A0$, використовувані для вибору старшого чи молодшого банків пам'яті, відповідно. Дані (1 байт) з парною адресою пересилаються лініями $D0 - D7$, а дані (1 байт) з непарною адресою – по лініям $D8 - D15$ шини даних.

Простір адрес вводу/виводу МП адресується молодшими 16 розрядами 20-розрядної адреси. Фізично простір адрес вводу/виводу організується як 32К 16-розрядних слів. Пристрої вводу/виводу можуть розміщуватися в просторі адрес пам'яті МП, при цьому над вмістом можна здійснювати дії, аналогічні діям над вмістом пам'яті.

Сигнали МП $\overline{S0} - \overline{S2}$ видають інформацію про тип циклу шини мікропроцесора. У максимальному режимі роботи мікропроцесорної системи функції управління каналом перебирає системний контролер шини I8288, тому сигнали $\overline{S0} - \overline{S2}$ подаються на системний контролер шини, який дешифрує їх і формує розширений набір сигналів, що управляють (див. таблицю 5.5). У циклі читання з МП I8086 виробляється сигнал $\overline{RD}$, але, оскільки ми дешифруємо сигнали станів $\overline{S0} - \overline{S2}$, сигнал читання $\overline{RD}$ у максимальному режимі ми не будемо використовувати (необхідний для мінімального режиму).

Для запуску або встановлення МП у вихідний стан використовують сигнал SR, формується генератором тактових сигналів I8284A. Також генератор формує сигнали синхронізації для МП, що надходять на вхід CLK, сигнал початкової установки $RESET$ і сигнал готовності $READY$. Сигнал $READY$ керує режимом очікування МП і дозволяє забезпечити пару в часі роботи МП з роботою зовнішніх пристроїв, що мають меншу швидкодію.

Таблиця 5.5 – Кодування типу циклу каналу за допомогою сигналів стану

Сигнали стану			Тип канального циклу
$\overline{S2}$	$\overline{S1}$	$\overline{S0}$	
0	0	0	Підтвердженням переривання
0	0	1	Читання з пристрою введення/виводу
0	1	0	Запис у пристрій введення/виводу
0	1	1	Останов
1	0	0	Вибірка команди
1	0	1	Читання пам'яті
1	1	0	Запис в пам'ять
1	1	1	Пасивний стан (циклу шини немає)

Запити зовнішніх переривань надходять на МП на вход INT (ті, що маскується). Якщо переривання дозволено та отримано сигнал INT, то МП формує сигнал підтвердження $\overline{INTA}$ через системний контролер I8288.

Двонаправлені лінії $\overline{RQ}/\overline{GT0}$ і $\overline{RQ}/\overline{GT1}$ призначені в організацію захоплення системної шини іншими пристроями. Вхід запиту захоплення $\overline{RQ}/\overline{GT0}$ має більш високий пріоритет. Вивід $\overline{LOCK}$, так же як і розглянуті лінії $\overline{RQ}/\overline{GT0}$ і $\overline{RQ}/\overline{GT1}$, служить для організації взаємодії МП I8086 із зовнішніми пристроями. Активний сигнал $\overline{LOCK}=0$ (канал зайнятий) дозволяє у разі потреби заборонити захоплення шин МП зовнішніми пристроями, але ми його використовувати в нашому контролері не будемо. Сигнали $QS0$, $QS1$ дозволяють зовнішній стосовно МП системі опитати стан черги визначити поточну виконувану команду. Ці сигнали використовуються зовнішніми процесорами (у разі співпроцесором), кодування станів наведено в таблиці 5.6.

Ефективним з погляду підвищення арифметичної продуктивності МП-системи є використання поруч із основним МП I8086 допоміжного арифметичного співпроцесора I8087 [23]. Умовне графічне позначення мікросхеми наведено на рисунку 5.2 призначення висновків представлено в таблиці 5.7.

Таблиця 5.6 – Кодування стану черги команд

Сигнали стану		Стан черги
*QS*0	*QS*1	
0	0	Відсутність операції (в останньому такті з черги не було вибірки)
0	1	Вибірка першого байта (з черги було обрано перший байт команди)
1	0	Порожня черга (черга була скинута під час виконання команди передачі управління)
1	1	Вибір наступного байта команди (читання багатобайтових команд)

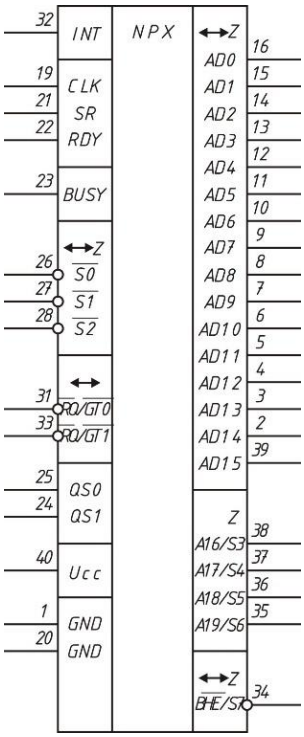


Рисунок 5.2 – Умовне графічне позначення співпроцесора I8087

Шістнадцятирозрядний математичний співпроцесор виконує операції над десятковими і цілими числами, а також над числами з плаваючою комою при довжині слова від двох до десяти байт (80 розрядів). У систему команд

співпроцесора входять не тільки операції складання та віднімання (час виконання не більше 14-18 мкс), множення (18 мкс при одинарній та 27 мкс при подвійній точності) та поділу (39 мкс), а й такі функції, як вилучення квадратного кореня (36 мкс), знаходження натурального логарифму (120 мкс) та інші.

Співпроцесор не має змоги самостійно вибирати команди і може працювати лише у парі з МП I8086, який має працювати у максимальному режимі. Він працює паралельно із основним процесором.

Таблиця 5.7 – Функціональне призначення висновків співпроцесора I8087

Номер виводу	Позначення	Функціональне призначення виводів
1, 20	*GND*	Загальний
16-2, 39	$AD0 - AD15$	Входи/виходи шини адреси/даних
19	*CLK*	Вхідний сигнал тактової частоти
21	*SR*	Вхідний сигнал «скидання» (установки)
22	*RDY*	Вхідний сигнал «готовності»
23	*BUSY*	Вихідний сигнал «зайнятості»
25, 24	$QS0$, $QS1$	Вхідні сигнали стану черги команд
26-28	$\overline{S0} - \overline{S2}$	Вихідні сигнали стану цикла
31, 33	$\overline{RQ}/\overline{GT0}$, $\overline{RQ}/\overline{GT1}$	Входи/виходи запиту/дозвіл доступу до шини
32	*INT*	Вихідний сигнал запиту переривання
34	$\overline{BHE}/S7$	Вихідний сигнал дозволу старшого байта шини даних/ сигнали стану
38-35	$A16/S3 - A19/S6$	Виходи старших розрядів адреси / Сигнали стану
40	*Ucc*	Напруга живлення +5 В

Як і МП I8086, співпроцесор можна умовно розділити на два незалежно працюючі пристрої, що дозволяють організувати конвеєрну обробку даних, що приймаються. У випадку I8087 можна як архітектурне розширення I8086. При цьому арифметичний співпроцесор додає свої вісім 80-розрядних

регістрів, а також регістри стану та управління до восьми регістрів загального призначення I8086. Набір 80-розрядних регістрів зазвичай використовується в режимі стека, що видає операнди в порядку, зворотному їх занесення. Дані подаються у стеку лише у форматі проміжного речовинного, які мають 64-розрядну мантису та 15-розрядний порядок. У цьому форматі проводяться всі внутрішні обчислення [23].

Зв'язок між I8086 та I8087 досягається простим з'єднанням однойменних висновків обох процесорів. Обидва процесори мають загальні генератори тактових імпульсів та системні шини контролера. Команди, що передаються лініями $AD0-AD15$, надходять на обидва процесори одночасно. Однак співпроцесор лише контролює стан МП по лініях $\overline{S0}-\overline{S2}$ та відстежує проходження його команд. Після того, як з'явиться команда, адресована співпроцесору, останній за сигналами $QS0$ и $QS1$ починає аналізувати стан черги команд Починаючи виконання команди, співпроцесор формує сигнал зайнятості $BUSY=1$, який подається на вхід $\overline{TEST}$ МП та затримує його у стані очікування на час виконання команди. Зазначимо, що, перебуваючи у стані очікування, МП аналізує запити переривання та за необхідності може їх обслужити. Закінчивши виконання команди, співпроцесор встановлює сигнал $BUSY=0$, роблячи активним сигнал на вході $\overline{TEST}$, що виводить МП із стану очікування. МП має переважне право використання шини. Коли у співпроцесора виникає необхідність використання шини, наприклад, при пересиланні результату операції в пам'ять, він надсилає запит на надання шини МП по лінії $\overline{RQ}/\overline{GT0}$. МП з цієї лінії видає сигнал дозволу використання шини сопроцессору. Після того, як співпроцесор виконає дії, пов'язані з використанням шини, він знову посилає сигнал $\overline{RQ}/\overline{GT0}$, що свідчить про закінчення використання.

Співпроцесор має ефективні апаратурно-програмні засоби захисту від помилок. Причому кожен із шести типів помилок індивідуально маскується від формування запиту переривання. Сигнал INT співпроцесора подається на один із входів контролера переривань, відповідно до обраного для цього випадку пріоритету [17]. У нашому випадку співпроцесор має найвищий пріоритет переривань, тому сигнал INT підключається до входу $IRQ0$ контролера переривань.

Центральна система управління всім контролером, забезпечує управління і синхронізацію роботи всього пристрою, забезпечує прийом, видачу, зберігання і обробку даних, що надходять, як по внутрішнім шинам даних, адреси, і управління, так і по зовнішній шині управління. Вміщує окрім центральних блоків управління: два двонаправлених шинних формувача I8286 (DD6.1, DD7, DD8); три односпрямованих буферних регістра-заскочки I8282 (DD10-DD12); генератор тактових імпульсів I8284A (DD1) із зовнішнім резонатором (ZQ1, R1, R2, C1) та схемою формування скидання та установки (R3, R4, C2, VD1-VD3); системний контролер I8288 (DD4); програмований контролер переривань I8259A (DD5).

Головною складовою блоку установки, скидання та генерування синхроімпульсів є мікросхема генератора тактових імпульсів I8284A, яка є пристроєм для МП I8086, співпроцесора I8087 і системного контролера шини I8288. Її призначення - формування тактових сигналів для мікропроцесора та периферійних пристроїв, а також сигналів "Установка" та "Готовність" [17; 22]. Умовне графічне позначення мікросхеми наведено на рисунку 5.3, призначення висновків у таблиці 5.8. Увімкнення мікросхеми у складі пристрою показано на рисунку 5.4.

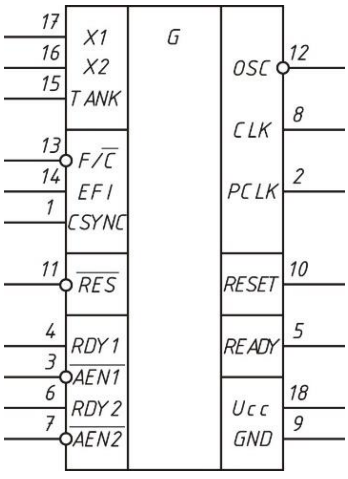


Рисунок 5.3 – Умовне графічне позначення генератора тактових імпульсів I8284A

Як джерело частоти в мікросхемі використовується кварцовий резонатор ZQ1, що підключається до висновків X1 та X2 мультивібратора. Вибране джерело має генерувати на триразовій частоті тактового сигналу CLK,

необхідної для МП. Оскільки наш процесор працює на частоті $f_{МП} = 5\ МГц$, то необхідна частота резонатора повинна дорівнювати:

$$f_{РЕЗ} = f_{МП} \cdot 3, \tag{5.1}$$

$$f_{РЕЗ} = 15\ МГц.$$

Таблиця 5.8 – Функціональне призначення виводів I8284A

Номер виводу	Позначення	Функціональне призначення виводів
1	*CSYNC*	Вхід синхронізації
2	*PCLK*	Вихід периферійного тактового сигналу ТТЛ
3, 7	$\overline{AEN1}$, $\overline{AEN2}$	Входи адреси готовності шин 1 і 2
4, 6	*RDY*1, *RDY*2	Вхідні сигнали готовності шин 1 і 2
5	*READY*	Вихідний сигнал «готовності»
8	*CLK*	Вихідний тактовий сигнал МОП
9	*GND*	Загальний
10	*RESET*	Вихідний сигнал «установки» (скидання)
11	$\overline{RES}$	Вхідний сигнал «установки» (скидання)
12	*OSC*	Вихідний сигнал мультивібратора
13	$F/\overline{C}$	Вхідний сигнал вибору джерела частоти
14	*EFI*	Вхідний сигнал зовнішньої частоти
15	*TANK*	Вивід підключення LC-контура
16, 17	*X*2, *X*1	Виводи підключення резонатора
18	*Ucc*	Напруга живлення +5 В

При використанні кварцового резонатора можлива робота на його гармоніках, у цьому випадку висновку *TANK* підключається резонансний LC-контур, проте стабільність тактової частоти забезпечується лише за роботі на основний частоті резонатора. Виберемо кварцовий резонатор типу РПК01 – НС-49U – 15.000 МГц, має металевий корпус та малі габаритні розміри. Відповідно до [24], послідовно з резонатором підключається конденсатор невеликої ємності. ($C_1 = 30\ нФ$), а паралельно резонатору підключається два резистори ($R_1 = R_2 = 510\ Ом$). Інші номінали розрахунку елементів наведені у

пункті 5.3 цього розділу пояснювальної записки. Задає частоту генератору, може також служити зовнішній генератор імпульсів, що підключається на вхід EFI. Вхід використовується для вибору генератора, що задає: $F/\overline{C}=0$ відповідає внутрішньому, а $F/\overline{C}=1$ – зовнішньому генератору.

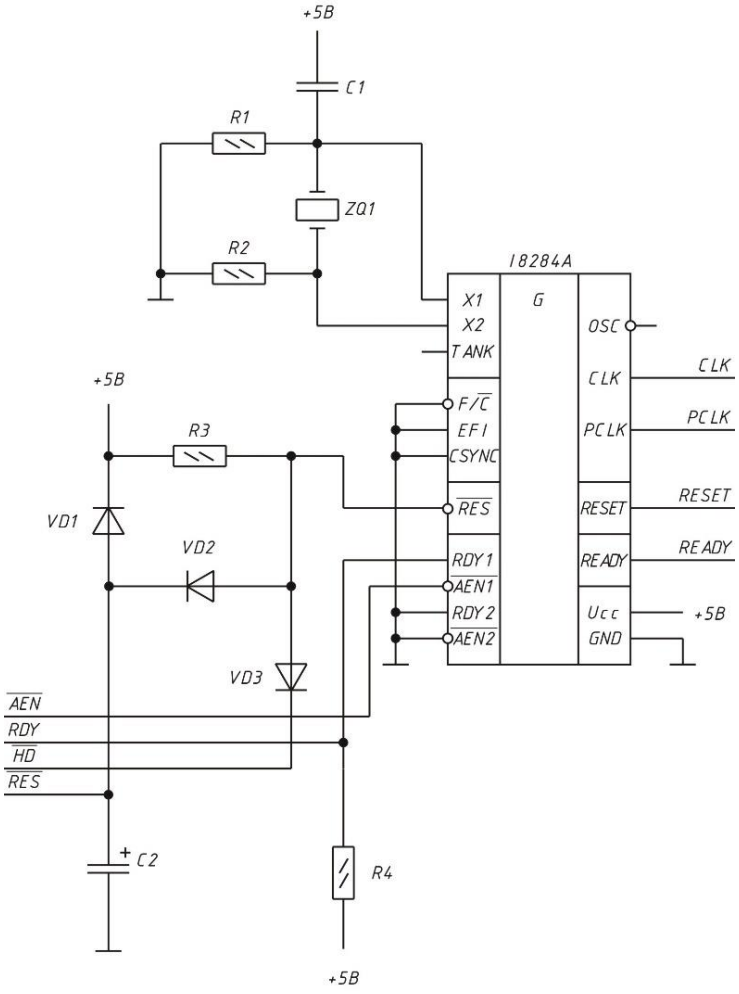


Рисунок 5.4 – Вмикання мікросхеми I8284A у складі пристрою

Вхід зовнішньої синхронізації використовується в системах із кількома генераторами, що працюють синхронно. Оскільки ми використовуємо внутрішній генератор імпульсів одного тактового генератора імпульсів, то входи EFI, $F/\overline{C}$ і $CSYNC$ підключаємо на логічний нуль.

У мікросхемі I8284A є три частотні виходи: $\overline{OSC}$ – мультивібратор, CLK – тактовий сигнал МОП і $PCLK$ – периферійний сигнал ТТЛ. Вихід $\overline{OSC}$ може підключатися до інших генераторів I8284A як джерело зовнішньої частоти, тому ми цей вихід використовувати не будемо. Вихід CLK формує

сигнали синхронізації для МП I8086, співпроцесора I8087 та контролера шин I8288. Шпаруватість імпульсу CLK дорівнює 3, амплітуда біля 4,5 В (при $U_{CC} = 5{,}0\ B$), тривалість фронту (зрізу) не більше 10 нс. Перелічені параметри тактового сигналу необхідні ефективного управління МП, та інші МОП- і ТТЛ-пристроями, безпосередньо з'єднаними з місцевою процесорною шиною [17; 22]. Периферійний тактовий сигнал $PCLK$ утворюється шляхом розподілу на 2 частоти сигналу CLK ($f_{CLK} = f_{МП} = 5\ МГц$):

$$f_{PCLK} = f_{CLK} / 2, \tag{5.2}$$

$$f_{PCLK} = 2{,}5\ МГц.$$

Сигнал призначений для синхронізації зовнішніх пристроїв, а саме для зовнішньої шини керування та пристрою послідовної передачі даних I8251A (DD18).

Генератор формує також сигнал $RESET$ початкової установки МП із зовнішнього сигналу $\overline{RES}$ і сигнал готовності $READY$. Сигнал $READY$ керує режимом очікування МП і дозволяє забезпечити пару в часі роботи МП з роботою зовнішніх пристроїв, що мають меншу швидкодію. У генераторі сигнал $READY$ формується тригером синхронізації готовності, який на входах $RDY1$, $\overline{AEN1}$ і $RDY2$, $\overline{AEN2}$ керується від двох зовнішніх пристроїв системи. Для отримання готовності сигналу $READY$, достатньо буде готовність однієї з пар:

$$RDY1 \cdot \overline{AEN1} + RDY2 \cdot \overline{AEN2} = 1. \tag{5.3}$$

Входи $RDY1$ и $RDY2$ визначають готовність цих пристроїв, але в входи $\overline{AEN1}$ и $\overline{AEN2}$ подаються сигнали дозволу аналізу відповідних сигналів готовності. Готовність системи здійснюється одночасно з тактовим сигналом. Входи $RDY2$ и $\overline{AEN2}$ підключаємо на логічний нуль, $RDY1$ підключаємо до логічної одиниці, яка також подається на контролер паралельного вводу/виводу I8255A (DD26) к виводу $PC1$. Сигнал готовності $\overline{AEN1}$ підключений до зовнішньої шини управління, яка задаватиме ритм роботи нашого контролера. Генератором управляє схема скидання, призначена для приведення процесора у вихідний стан та виведення процесора зі стану «зависання». Складається із вхідних сигналів: $\overline{RES}$ подається із зовнішньої

шини управління та $\overline{HD}$ задається програмованим таймером I8253A (DD21) (наприклад, для виведення процесора зі стану «зависання»), а також містить діоди $VD1-VD3$. У цій схемі будемо застосовувати високочастотні діоди типу КД522Б. Встановлення системи у вихідний стан може здійснюватися при включенні живлення за допомогою простого RC-ланцюга (R_3, $VD1$, C_2), що підключається до входу $\overline{RES}$ генератора. У нормальному стані конденсатор C_2 заряджений і вхід МП $\overline{RES}$ через резистор R_3 з'єднаний із джерелом +5 В, що зумовлює у ньому логічну одиницю. Після подачі живлення до контролера, конденсатор C_2 розряджається на корпус, після чого починається його розрядка, вхід МП $\overline{RES}$ виявляється у своїй замкнутим на корпус, що відповідає стану логічного нуля. Після закінчення зарядки на вході $\overline{RES}$ знову встановлюється рівень, що відповідає логічній одиниці.

5.3 Розрахунок елементів та функціональних вузлів схеми

У логічних системах перешкоди можуть наводитися від найрізноманітніших джерел і виявлятися в найрізноманітнішій формі. Перешкоди, які є причиною помилкового спрацьовування чутливих ланцюгів апаратури, поділяються на декілька видів:

- перехресні перешкоди, що наводяться одними сигнальними лініями в інших сигнальних лініях;
- зовнішні перешкоди, що проникають з навколишнього середовища в систему від різного роду електромагнітних сигналів, а також обумовлені дією електростатичних та електромагнітних полів;
- відображення в лініях зв'язку, при неузгодженні навантаження;
- струмові перешкоди ланцюга живлення, виникають в результаті викидів струму при комутаційних процесах.

Перші три типи перешкод, можливо, усунути на етапі проектування пристрою шляхом розведення провідників на друкованій платі, дотримання рекомендованих обмежень на довжину з'єднувальних провідників та розміщення елементів на друкованій платі, екранування найбільш сприйнятливих до перешкод та наведень ділянок схеми.

Останній тип перешкод, саме струмових перешкод, слід враховувати етапі розробки електричної принципової схеми. Цю проблему можна вирішити за допомогою включення конденсаторів розв'язки між шинами живлення та загальною, які є ефективним засобом захисту ІС від перешкод ланцюга живлення. У цьому випадку шина живлення розглядається як провідний постійний струм елемент, який має низький опір при протіканні струмів перехідних процесів на землю. Конденсатори, призначені для якісної розв'язки, повинні мати велику ємність для низьких частот та малу для високих. Зазвичай конденсатори розв'язки встановлюють окремо для блокування низькочастотних та високочастотних перешкод.

Низькочастотні перешкоди по ланцюгу живлення, що проникають в систему, повинні блокуватися за допомогою електролітичного конденсатора ємністю не менше 1,0 мкФ, з розрахунку, що на кожні п'ять-десять ІМС припадає один конденсатор. Електролітичні конденсатори слід встановлювати якомога ближче до контактів роз'ємів [25].

Таким чином, для процесорної плати контролера, що містить 12 корпусів ІМС, вибираємо 3 конденсатора ємністю 4,7 мкФ, а для плати введення-виведення контролера, що містить 14 корпусів ІМС, вибираємо 3 конденсатора аналогічної ємності 4,7 мкФ для ланцюга живлення +5.

Для зняття високочастотних перешкод, що розв'язують ємності, у загальному випадку повинні бути рівномірно розподілені по всій площі друкованої плати щодо ІМС. Кількість конденсаторів береться з розрахунку один конденсатор на групу не більше 10 ІМС, а ємність 0,001-0,01 мкФ на одну ІМС. При цьому необхідно, щоб лінія мала по можливості низький опір для протікання струмів перехідних процесів «на землю» і щоб довжина висновків конденсаторів, що розв'язували, в безпосередній близькості від ІВ була мінімальною. Ця вимога диктується необхідністю блокування високочастотної напруги, що викликаються сплесками струму, що виникають у ланцюзі живлення ІВ [25].

Таким чином, для процесорної плати контролера, що містить 12 корпусів ІМС, вибираємо 12 конденсатора ємністю 0,01 мкФ, а для плати введення-виведення контролера, що містить 14 корпусів ІМС, вибираємо 14 конденсатора аналогічної ємності 0,01 мкФ.

У схемі контролера застосовані резистори, які забезпечують подачу напруги логічної одиниці одночасно кілька входів мікросхем.

Оскільки струм, який споживається входом мікросхеми при подачі сигналу логічної одиниці, для мікросхеми становить $I_{IH}(I^1_{вх})$, то номінал резистора виберемо, виходячи з таких міркувань:

– струм, що проходить через резистор, до якого підключено кілька входів логічних елементів дорівнюватиме сумі струмів, споживаних кожним із входів;

– падіння напруги на резисторі при можливих змінах напруги живлення $U_{CC} = +5\ B$ в межах $\pm\ 0{,}25\ B$ повинно складати $U_{Пад}$ для забезпечення рівня логічної одиниці:

$$U_{Пад} = (U_{CC} - 0{,}25) - U^1_{вх}, \tag{5.6}$$

де $U^1_{вх}$ – вхідна напруга логічної одиниці;

$U_{Пад}$ – необхідне падіння напруги.

Таким чином, формула для розрахунку опору резистора матиме вигляд:

$$R = \frac{U_{Пад}}{n \cdot I^1_{вх}}, \tag{5.7}$$

де n – кількість об'єднаних входів логічних елементів;

$I^1_{вх}$ – струм, споживаний одним входом при подачі сигналу логічної одиниці.

Для резистора, що підключається до мікросхеми I8284A до входу $\overline{RES}$ і к діодам $VD2$, $VD3$ ($U^1_{вх} = 2{,}0\ B$, $I^1_{вх} = 0{,}5\ мА$, $n = 3$): $R = 1800\ Ом$. Значить номінал дорівнює: $R_3 = 1{,}8\ кОм$.

Максимальна потужність резистора, що розсіюється, складе:

$$P = (I^1_{вх})^2 \cdot R. \tag{5.8}$$

Тому $P = 0{,}45 \cdot 10^{-3}\ Вт$. Отже, можливе застосування резисторів МЛТ-0,125 або резисторів, виконаних у корпусі для поверхневого монтажу.

Ланцюг R_3C_2 служить на формування тривалості сигналу скидання, тобто. короткочасного формування імпульсу з негативним переднім фронтом

із тривалістю мінімум 50 мкс. Підберемо номінали ланцюжка R_3C_2, добуток яких повинен бути рівним або більше 50 мкс, маємо: $R_3 = 1{,}8\ кОм$, $C_2 = 0{,}47\ мкФ$.

Для резистора, що підключається до мікросхеми I8284A, до входу $RDY1$ і виводу RDY ($U^1_{вх} = 2{,}0\ В$, $I^1_{вх} = 0{,}5\ мА$, $n = 2$): $R = 2700\ Ом$. Значить номінал дорівнює: $R_4 = 2{,}7\ кОм$. Максимальна потужність, що розсіюється, складе: $P = 0{,}67 \cdot 10^{-3}\ Вт$, як розглянуто раніше виберемо резистор МЛТ-0,125.

Для резистора, що підключається до мікросхеми I8288, до входу CEN ($U^1_{вх} = 2{,}0\ В$, $I^1_{вх} = 0{,}7\ мА$, $n = 1$): $R = 4300\ Ом$. Значить номінал дорівнює: $R_5 = 4{,}3\ кОм$. Максимальна потужність, що розсіюється, складе: $P = 2{,}1 \cdot 10^{-3}\ Вт$, як розглянуто раніше, виберемо резистор МЛТ-0,125.

Для резистора, що підключається до мікросхеми I8253A, до входів $GATE0$, $GATE1$ ($U^1_{вх} = 2{,}2\ В$, $I^1_{вх} = 0{,}6\ мА$, $n = 2$): $R = 2200\ Ом$. Значить номінал дорівнює: $R_{13} = 2{,}2\ кОм$. Максимальна потужність, що розсіюється, складе: $P = 0{,}79 \cdot 10^{-3}\ Вт$, як розглянуто раніше, виберемо резистор МЛТ-0,125.

5.4 Розробка програмного забезпечення

Всі операції, що здійснюються контролером, визначаються програмою, що знаходиться в постійному пристрої. Весь код програми прошивається за допомогою спеціального програматора у ПЗП.

Оскільки всю програму в цій записці пояснення ми не зможемо надати, вона досить громіздка для цього. Представимо кілька фрагментів та підпрограм виконання програмного коду у Додатку А.

6 ТЕХНІКО-ЕКОНОМІЧНА ЧАСТИНА

6.1 Розрахунок собівартості

Собівартість вироби – це виражені у грошовій формі поточні витрати підприємства на його виробництво та збут [27; 28]. Витрати виробництво утворюють виробничу (заводську) собівартість, а Витрати виробництво і збут – повну собівартість. Розрахунок собівартості за статтями витрат називається калькуляцією. Калькулювання собівартості виробу здійснюється відповідно до «Типовим положенням щодо планування обліку та калькулювання собівартості продукції (робіт, послуг) у промисловості» [27; 28].

Застосовується таке угруповання за статтями калькуляції:

1) основна заробітна плата виробничих робітників;

2) додаткова заробітна плата виробничих робітників;

3) відрахування від заробітної плати;

4) сировину та матеріали, куплені комплектуючі та напівфабрикати;

5) цехові витрати;

6) загальнозаводські витрати;

7) адміністративні витрати.

Таке угруповання витрат дозволяє визначити собівартість виробу пристрою та надалі рівень ціни.

Для виробництва будь-якого виробу, у тому числі електронних пристроїв, необхідно витратити певні кошти. Але перш за все, необхідно мати обладнання, за допомогою якого пристрій буде виготовлятися. Купівля технологічного устаткування повного циклу виробництва електронних пристроїв одного типу дуже підвищить собівартість кінцевого продукту. Тому прийнято рішення скористатися послугами підприємств електронної промисловості, які мають необхідне обладнання та надають такі послуги. Таким чином, персонал організації-виробника пристрою, що розробляється, буде зайнятий лише збіркою, налаштуванням і подальшим обслуговуванням апаратури, що виготовляється.

Організація-розробник є невеликим підприємством, обладнаним апаратурою для ремонту та технічного обслуговування електронних пристроїв, що не заважає зайнятися виробництвом невеликої партії цифрових пристроїв, середнього ступеня складності, не виробляючи переобладнання

наявних приміщень.

Як виконавець з виробництва друкованих плат було обрано компанію «НДІ Комп'ютерних технологій» виготовлення партії зі ста друкованих плат необхідної конфігурації з урахуванням підготовки виробництва коштуватиме замовнику 15 грн. за штуку. Таким чином вартість партії друкованих плат у розмірі 100 шт. становитиме 1500 грн.

6.2 Розрахунок повної собівартості електронної системи, що розробляється

Нижче здійснено розрахунок собівартості партії комплектів обладнання за калькуляційними статтями [29]. Передбачається, що за весь термін роботи потрібно виготовити сто комплектів обладнання.

1. Витрати на фонд основної заробітної плати становитимуть:

$$З_о = \sum_{i=1}^{n} З_{oi}\,, \tag{6.1}$$

$$З_{oi} = О_i \cdot К_i \cdot М_i\,, \tag{6.2}$$

де n – кількість різних посад виробничих працівників;

$О$ – фіксований оклад виробничого працівника, грн.;

$К$ – кількість осіб однакової посади;

$М$ – кількість місяців для виконання всього замовлення;

$$З_о = 2100 \cdot 1 \cdot 5 + 2300 \cdot 1 \cdot 5 + 1800 \cdot 3 \cdot 5 = 49000{,}00 \ \ (грн).$$

Відомості про основну заробітну плату виробничих працівників підприємства наведено у таблиці 6.1.

Таблиця 6.1 – Основна заробітна плата виробничих працівників

№ п/п	Посада	Оклад, грн.	Кіл., чол.	Кіл., міс.	Сума, грн.
1	Інженер-конструктор І кат.	2100	1	5	10500,00
2	Інженер-програміст І кат.	2300	1	5	11500,00
3	Монтажник 4 розряду	1800	3	5	27000,00
	Всього:				49000,00

2. Додатковий фонд заробітної плати складає 20% від основного фонду

заробітної плати:

$$З_д = З_о \cdot \frac{К_д}{100\ \%}, \tag{6.3}$$

де $К_д$ – відсоток додаткової заробітної плати ($К_д = 20\ \%$);

$$З_д = 49000 \cdot \frac{20\ \%}{100\ \%} = 49000 \cdot 0{,}2 = 9800{,}00\ \ (грн).$$

3. Відрахування від заробітної плати згідно з чинним законодавством становить 37,2% (до пенсійного фонду – 33,2%, соціального страхування – 1,4%, фонду зайнятості – 1,6%, фонду страхування від нещасних випадків – 1,0%) від суми основної та додаткової заробітної плати:

$$С_{соц} = (З_о + З_д) \cdot \frac{37{,}2\ \%}{100\ \%}, \tag{6.4}$$

$$С_{соц} = (49000 + 9800) \cdot 0{,}372 = 21873{,}60\ \ (грн).$$

4. Підрахуємо основні матеріали, покупні комплектуючі та напівфабрикати. У таблиці 6.2 наведено відомості про вартість комплектів електронних компонентів, необхідних для збирання пристрою. Комплект обладнання складається із двох плат, на яких розташований контролер.

Таблиця 6.2 – Ціни електронних компонентів

№ п/п	Найменування компонента	Кіл.	Вартість, грн.	
			За один.	Загальна
1	2	3	4	5
1	Мікросхема I8284A	1	5,75	5,75
2	Мікросхема I8086	1	8,75	8,75
3	Мікросхема I8087	1	8,75	8,75
4	Мікросхема I8288	1	4,00	4,00
5	Мікросхема I8259A	1	9,75	9,75
6	Мікросхема SN74ALS00AN	2	4,25	8,50
7	Мікросхема I8286	4	2,59	10,36
8	Мікросхема SN74ALS138N	2	2,32	4,64
9	Мікросхема I8282	3	1,30	3,90
10	Мікросхема M27C64A	2	15,00	30,00
11	Мікросхема SN74ALS02A	1	6,85	6,85
12	Мікросхема I8251A	1	3,00	3,00
13	Мікросхема UT6264CPCL-70LL	2	12,50	25,00

Завершення таблиці 6.2

1	2	3	4	5
14	Мікросхема I8253A	1	15,00	15,00
15	Мікросхема SN74ALS04B	1	7,00	7,00
16	Мікросхема SN74ALS10AN	1	7,25	7,25
17	Мікросхема I8255A	1	13,75	13,75
18	Діод 1N4001	4	0,68	2,72
19	Транзистор КТ361	1	0,43	0,43
20	Транзистор КТ315	2	0,58	1,16
21	Резонатор кварцовий РПК01 НС-49U-15МГц-8ПС	1	2,75	2,75
22	Конденсатор керамічний КМ-5Б-М47-33пФ	1	3,25	3,25
23	Конденсатор електролітичний К50-35-10В-47мкФ	1	0,43	0,43
24	Конденсатор електролітичний К50-35-50В-4,7мкФ	6	0,38	2,28
25	Конденсатор керамічний К10-17Б-Y5V0805-50В-0.01мкФ	27	0,55	14,85
26	Резистор постійний SMD 0805 0,125Вт	13	0,08	1,04
27	Вилка штирова дворядна пряма PLD-20	2	3,25	6,50
28	Вилка штирова дворядна пряма PLD2-40	2	4,75	9,50
29	Вилка штирова дворядна пряма PLD-18	1	2,75	2,75
30	Вилка штирова дворядна пряма PLD-4	1	1,25	1,25
31	Допоміжні матеріали (припій, флюс)	1	3,80	3,80
32	Друкована плата	2	24,50	49,00
	Разом:			273,96
	Усього за партію 100 шт.:			27396

Підрахуємо витрати на матеріали та покупні вироби:

$$C_{ом} = k \cdot \sum_{i=1}^{n} C_{омі} \,, \tag{6.5}$$

$$C_{омі} = K_i \cdot Ц_i \,, \tag{6.6}$$

де n – кількість компонентів;

k – кількість штук у партії ($k = 100$);

$C_{омі}$ – ціна витрат на один тип компонента, грн.;

$К_i$ – кількість одного типу компонента;

$Ц_i$ – вартість компонента, грн.;

$$C_{ом} = 27396,00 \ \ грн.$$

5. Витрати утримання устаткування становлять 5% від основний зарплати:

$$C_{об} = З_о \cdot \frac{K_{об}}{100\ \%}, \tag{6.7}$$

де $К_{об}$ – відсоток витрат на утримання обладнання ($К_{об} = 5\ \%$);

$$C_{об} = 49000 \cdot 0,05 = 2450,00 \ \ (грн).$$

6. Цехові витрати на утримання приміщення становлять 8% від основної заробітної плати:

$$C_{общ} = З_о \cdot \frac{K_{общ}}{100\ \%}, \tag{6.8}$$

де $К_{общ}$ – відсоток витрат на утримання обладнання ($К_{общ} = 8\ \%$);

$$C_{общ} = 49000 \cdot 0,08 = 3920,00 \ \ (грн).$$

7. Цехова собівартість є витрати виробничого підрозділу підприємства на виробництво продукції:

$$C_{цех} = З_о + З_д + C_{соц} + C_{ом} + C_{об} + C_{общ}, \tag{6.9}$$

$$C_{цех} = 49000 + 9800 + 21873,60 + 27396 + 2450 + 3920 = 114439,60 \ \ (грн).$$

Загальнозаводські витрати включають цехові витрати, а також основну та додаткову зарплату управлінського персоналу та становлять 160 % від суми прямої заробітної плати, витрати на утримання приміщень та інші потреби:

$$C_{озр} = З_о \cdot \frac{K_{озр}}{100\ \%} + C_{общ}, \tag{6.10}$$

де $К_{озр}$ – відсоток витрат на загальнозаводські витрати ($К_{общ} = 160\ \%$);

$$C_{озр} = 49000 \cdot 1,6 + 3920 = 82320,00 \ \ (грн).$$

8. Заводська (виробнича) собівартість, крім витрат цехів, включає загальні по підприємству витрати:

$$C_{зав} = C_{цех} + C_{озр}, \quad (6.11)$$

де $C_{цех}$ – сума затрат розрахованих за формулою (3.9), грн.;

$C_{озр}$ – загальнозаводські витрати, грн.

Таким чином, заводська собівартість складає:

$$C_{зав} = 114439,60 + 82320 = 196759,60 \ (грн).$$

9. Оскільки підприємство-виробник планує постачання обладнання у комплектації для первинної установки, то витрати на упаковку будуть мінімальними. Сума позавиробничих витрат складе $C_{вн} = 1500,00 \ грн$.

10. Повна собівартість включає витрати на виробництво і реалізацію продукції:

$$C_{полн} = C_{зав} + C_{вн}, \quad (6.12)$$

де $C_{вн}$ – позавиробничі витрати, що включають витрати на транспортування, упаковку, маркування тощо, грн.

Таким чином, повна собівартість складе:

$$C_{полн} = 196759,60 + 1500 = 198259,60 \ (грн).$$

Підсумовуючи отримані результати, наведемо у таблиці 6.3 статті калькуляції та витрати на них.

Таблиця 6.3 – Статті калькуляції та витрати на них

№ п/п	Статті калькуляції	Витрати, грн.
1	Основна заробітна плата	49000,00
2	Додаткова заробітна плата	9800,00
3	Відрахування від заробітної плати	21873,60
4	Матеріали та покупні вироби	27396,00
5	Витрати на утримання обладнання	2450,00
6	Цехові витрати	3920,00
7	Загальнозаводські витрати	82320,00
8	Виробнича собівартість	196759,60
9	Позавиробничі витрати	1500,00
10	Повна собівартість	198259,60

6.3 Розрахунок ціни електронної системи

Відпускна ціна партії зі ста комплектів блоків обладнання:

$$Ц_{отп} = C_{полн} + П, \tag{6.13}$$

де $C_{полн}$ – повна собівартість комплектів обладнання, грн.;

$П$ – величина прибутку, грн.

Прибуток визначається виходячи з нормативу (показника) рентабельності виробництва продукції, що встановлюється підприємством:

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

де R – рентабельність продукту, яка приймається у розмірі ($R = 15\ \%$).

Підставимо значення величини прибутку з (3.14) до (3.13), отримаємо відпускну ціну партії:

$$Ц_{отп} = C_{полн} + \frac{R \cdot C_{полн}}{100\ \%} = C_{полн}(1 + \frac{R}{100\ \%}), \tag{6.15}$$

$$Ц_{отп} = 198259,60 \cdot (1 + 0,15) = 227998,54\ \ (грн).$$

Відпускна ціна одного комплекту обладнання контролера:

$$Ц_{отп.ед.} = \frac{227998,54}{100} \approx 2280,00\ \ (грн).$$

10. Ціна з ПДВ:

$$Ц_{отпНДС} = (\frac{K_{НДС}}{100\ \%} + 1) \cdot Ц_{отп}, \tag{6.16}$$

$$Ц_{отпедНДС} = (\frac{K_{НДС}}{100\ \%} + 1) \cdot Ц_{отпед}\ , \tag{6.17}$$

де $K_{НДС}$ – значення податку на додану вартість ($K_{НДС} = 20\ \%$);

$$Ц_{отпНДС} = 1,2 \cdot 227998,54 = 273598,24\ \ (грн),$$

$$Ц_{отпедНДС} = 1,2 \cdot 2280 = 2736,00\ \ (грн).$$

Розрахунок ціни товару з урахуванням ПДВ та без урахування наведено у таблиці 6.4.

Таблиця 6.4 – Розрахунок ціни товару

№ п/п	Найменування ціни	Вартість, грн.
1	Ціна відпускна за 100 шт.	227998,54
2	Відпускна ціна одиниці виробу	2280,00
3	Ціна із ПДВ за 100 шт.	273598,24
4	Ціна з ПДВ одиниці виробу	2736,00

Таким чином, відпускна ціна одиниці пристрою стиснення на основі арифметичного кодування становить 2280,00 грн., або з урахуванням ПДВ – 2736,00 грн.

Провівши аналіз ринку обладнання для пристроїв зі стиснення даних, показав, що вартість обладнання лише для одного пристрою знаходиться в межах 500-1000 доларів США. І враховуючи досить не малу різницю в ціні продуктів, можливо, припустити, що розроблений пристрій зможе зайняти свою необхідну нішу для виконання завдань зі стиснення даних.

ВИСНОВОК

Результатом виконання роботи є: розробка алгоритму арифметичного кодування; алгоритм роботи пристрою стиснення даних на основі арифметичного кодування; електричних схем (структурної, функціональної, принципової); написання програмного забезпечення під даний пристрій; розробка друкованої плати процесорного модуля контролера; розрахунок собівартості контролера за його серійному виробництві; оцінка потенційних небезпек та шкідливостей при виготовленні та експлуатації проектованого пристрою; розглянуто технологію виготовлення контролера та друкованих плат для нього.

Отриманий контролер має такі параметри. Тип процесора – I8086 фірми Intel. Тактова частота – 5 МГц. Конфігурація МПС – максимальна. Пам'ять: ємність ОЗП – 16 Кбайт, ємність ПЗП – 16 Кбайт. Введення-виведення: спосіб звернення до портів введення-виведення – за допомогою спеціальних команд; Метод організації введення-виведення – введення-виведення інформації в режимі переривання. Порт введення – паралельний, 8-розрядний односпрямований порт уведення, а також неповна ISA шина. Порт виведення – послідовний, один інтерфейс RS-232, асинхронний режим.

Контролер має такі характеристики. Живлення контролера здійснюється від джерел: +5 не більше 600 мА; +15 не більше 10 мА; -15 не більше 10 мА.

До переваг даного проектованого пристрою відноситься можливість реалізації на досить недорогій доступній імпортній та вітчизняній елементній базі. Високий ступінь стиснення, навіть у порівнянні з алгоритмом Хаффмана.

До недоліків даного проектованого пристрою є досить ресурсомістка апаратурна реалізації алгоритму, яка вимагає в ідеалі, майже необмеженої точності розрядності даних, що оперуються по роботі з речовими числами.

СПИСОК ЛІТЕРАТУРИ

1. Голь В. Д., Ірха М. С. Телекомунікаційні та інфокомунікаційні мережі. Навчальний посібник. Київ: ІСЗЗІ КПІ ім. Ігоря Сікорського, 2021. 250 с.
2. Sayood Kh. Introduction to Data Compression. Morgan Kaufmann, 2019. 5 edition. 790 p.
3. Голь В. Д., Ірха М. С. Системи передачі даних : конспект лекцій. Київ: ІСЗЗІ КПІ ім. Ігоря Сікорського, 2021. 126 с.
4. Погребняк Б. І., Булаєнко М. В. Операційні системи : навч. посібник. Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. Харків : ХНУМГ ім. О. М. Бекетова, 2018. 104 с.
5. McAnlis Colt, Haecky Aleks. Undestanding Compression. O'Reilly Media, Inc., 2018. 1st edition. 222 p.
6. Shannon C. E. A mathematical theory of communication // The Bell System Technical Journal. 1948. Vol. 27. P. 379–423, 623–656.
7. Sayood Kh. Lossless Compression Handbook. Academic Press, 2014. 488 p.
8. Gaines B. R. System identification, approximation and complexity // Int. J. General Syst. 2020. № 3. P.145-174.
9. Bell T. C., Moffat A. M. A note on the DMC data compression scheme // Computer. J. 32/1. 1989. P. 16-20.
10. Gottlieb D., Hagerth S. A., Lehot P. G. H. and Rabinowitz H. S. A classification of compression methods and their usefulness for a large data processing center // National Comput.Conf. 1975. № 44. P. 453-458.
11. Knuth D.E. Dynamic Huffman coding // J. Algorithms. № 6/2. P. 163–180.
12. Fiala E. R., Greene D. H. Data compression with finite windows // CACM-32, 4. 2019. P. 490–505.
13 Беркман Л. Н., Бондарчук А. П., Гайдур Г. І., Чумак Н. С. Кодування джерел інформації та каналів зв’язку. Частина 3: посібник. Київ: ННІТІ ДУТ, 2018. 91 с.
14. Rubin F. Arithmetic stream coding using fixed precision registers // IEEE Trans. Inf. Theory IT-25. 1979. № 6. P. 672-675.
15. Witten I. C., Neal R. M., Cleary J. G. Arithmetic coding for data compression // Communication of the ACM. 2019, v. 30, № 6. P. 520–540.

16. Cover T. M., Thomas J. A. Elements of Information Theory. John Wiley & Sons, New York. 1991. 187 p.
17. Brey B. B. The Intel microprocessors 8086/8088, 80186/80188, 80286, 80386, 80486, Pentium and Pentium Pro Processor. Architecture, programming and interfacing. New Jersey: Prentice Hall. 2013. 920 p.
18. Огородник К. В., Книш Б. П. Мікропроцесорна техніка: навчальний посібник. Вінниця : ВНТУ, 2018. 106 с.
19. The TTL Data Book. Texas Instruments, 2020. 724 p.
20. UT6264C. 8Kx8 Bit Low Power CMOS RAM: Utron Technology Inc. 2001. 33 p.
21. M27C64A. 8Kx8 Bit UV EPROM and OTP EPROM. STmicroelectronics company. 20014. 27 p.
22. 8086 16-bit HMOS Microprocessor: Intel Corporation, Copyright, 2009. 169 p.
23. 8087 Math Coprocessor: Intel Corporation, Copyright 2016. 108 p.
24. 8284A/8284A-1. Clock generator and driver for 8086, 8088 Processors: Intel Corporation, Copyright, 2019. 84 p.
25. http://www.texnic.com/data/ims-sprav.htm. Довідник мікросхем. Добірка довідкової документації на вітчизняні та зарубіжні, цифрові та аналогові мікросхеми (оновлено 2021 р.)
26. https://radiodetali.com.ua/ua/catalog. Довідник конденсаторів. Добірка довідкової документації на вітчизняні та зарубіжні конденсатори (оновлено 2022 р.)
27. Економіка підприємства: Підручник. – В 2 т. / За ред. С.Ф. Покропивного. – К.: Вид-во "Хвиля-Прес", Донецьк: МП "Пошук", 2015. – 280 с.
28. Економіка підприємства: Підручник / За заг. ред. д.е.н., проф. Л. Г. Мельника. – Суми: ВТД "Університетська книга", 2004. – 648 с.
29. Типове положення з планування, обліку і калькулювання собівартості продукції. Затверджено КМУ від 26 квітня 1996 № 473 // Бізнес. – № 32-35.

ДОДАТОК А
Лістинг програми
Ініціалізація визначень

```
; Ініціалізація адрес портів
KP0A        EQU 0000H       ; Контролер переривань I8259A
KP0B        EQU 0001H
PP0A        EQU 0020H       ; Інтерфейс паралельний I8255A
PP0B        EQU 0021H
PP0C        EQU 0022H
PP0R        EQU 0023H
PS0D        EQU 0040H       ; Інтерфейс послідовний I8251A
PS0R        EQU 0041H
T0C0        EQU 0060H       ; Таймер I8253A
T0C1        EQU 0061H
T0C2        EQU 0062H
T0CR        EQU 0063H
; Ініціалізація адрес пам'яті
SIZE_RAM    EQU 4000H       ; Розмір ОЗУ=16Кб (16384 байт)
SIZE_ROM    EQU 4000H       ; Розмір ПЗУ=16Кб (16384 байт)
SIZE_BNK    EQU 2000H       ; Розмір одного банка ОЗП і ПЗП 8Кб (8192
                            байт)
```

Підпрограма обчислення суми SUMM

Оскільки нам необхідно буде підсумовувати масиви даних з речовими числами, то нам знадобиться підпрограма за допомогою, якою буде зручно підсумовувати масиви кумулятивних ймовірностей, згідно з алгоритмом арифметичного кодування.

Підпрограма узагальненого підсумовування чисельного масиву, автоматичне перетворення вхідних даних у внутрішній речовий формат, а вихідних даних у формат одержувача. Тип оброблюваних даних передається як параметр і враховується в командах завантаження і запам'ятовування, а внутрішні обробка в співпроцесорі виявляється однаковою для всіх типів даних. Підпрограма обчислює суму масиву, елементами якого можуть бути цілі числа у форматі слова (2 байти), а також короткі (4 байти) або довгі (8байт) речові числа. Як параметри підпрограмі передаються масив ARRAY, ціла змінна TYPE, що визначає довжину елементів масиву (2, 4 або 8), змінна N, що задає число

елементів масиву, і змінна SUM в довгому речовинному форматі, через яку повертається накопичена сума.

```
CSEG    SEGMENT
        ASSUME  CS:CSEG
SUMM    PROC    FAR
        PUSH    BP              ; Формувати
        MOV     BP, SP          ; стековий кадр
        MOV     BX, [BP+12]     ; В регістрі BX адрес масива
        MOV     SI, [BP+10]     ; В регістрі SI адрес TYPE
        MOV     AX, [SI]        ; В регістрі AX тип елементів
        MOV     SI, [BP+8]      ; В регістрі SI адрес N
        MOV     CX, [SI]        ; В регістрі CX значення N
; Підготовка завершена
        FLDZ                    ; Включити 0.0 в стек
        CMP     CX, 0           ; Перевірити N
        JLE     DONE            ; Масива немає
LADD:   CMP     AX, 2           ; Формат цілого слова?
        JNE     NOINT           ; Ні, перейти
        FIADD   WORD PTR [BX]   ; Так, додати елемент
        JMP     NEXT            ;
NOINT:  CMP     AX, 4           ; Короткий дійсний формат
        JNE     NOSNG           ; Ні, перейти
        FADD    DWORD PTR [BX]  ; Так, додати елемент
        JMP     NEXT
NOSNG:  FADD    QWORD PTR [BX]  ; Довгий дійсний формат
NEXT:   ADD     BX, AX          ; Передвинути покажчик
        LOOP    LADD            ; При необхідності повторити
DONE:   MOV     DI, [BP+6]      ; Адрес суми в регістрі DI
        FSTP    QWORD PTR [DI]  ; Запам’ятати суму
        POP     BP              ; Відновити BP
        FWAIT                   ; Очікувати, поки сопроцесор
                                завершить
        RET     8               ; Повернення
SUMM    ENDP
CSEG    ENDS
        END
```

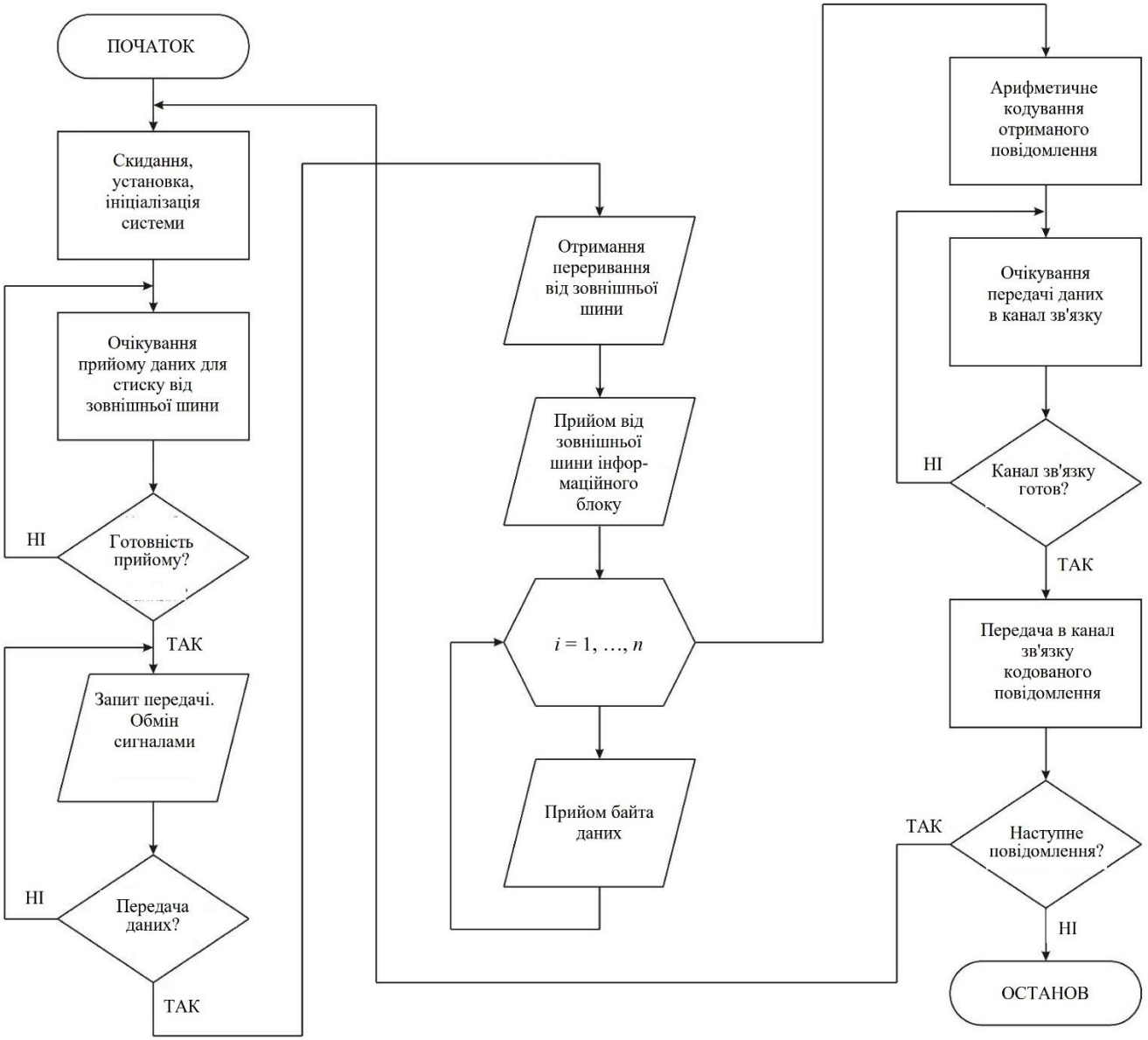
Поз. позн.	Найменування	Кол.	Примітки
	Мікросхеми		
DD1	I8284A	1	
DD2	I8086	1	
DD3	I8087	1	
DD4	I8288	1	
DD5	I8259A	1	
DD6,DD14	SN74ALS00AN	2	
DD7,DD8	I8286	2	
DD23,DD2	I8286	2	
DD9,DD13	SN74ALS138N	2	
DD10…DD	I8282	3	
DD15,DD1	M27C64A	2	
DD17	SN74ALS02A	1	
DD18	I8251A	1	
DD19,DD2	UT6264CPCL-70LL	2	
DD21	I8253A	1	
DD22	SN74ALS04B	1	
DD25	SN74ALS10AN	1	
DD26	I8255A	1	
	Резонатори		
ZQ1	РПК01 НС-49U - 15.000МГц - 8ПС	1	Резонатор кварцевий
	Діоди		
VD1…VD4	КД522Б	4	

					ЕлІТ 8.171.00.10.503 ПЕ3
Изм.	Лист	№ докум.	Подпись	Дата	
Розроб.		Матьякубов М. С.			Електронна система стиснення даних на основі арифметичного кодування. Перелік елементів
Перевір.		Кулик І. А.			
Реценз.					
Н. Контр.		Гапич В. М.			
Затв.		Опанасюк А. С.			

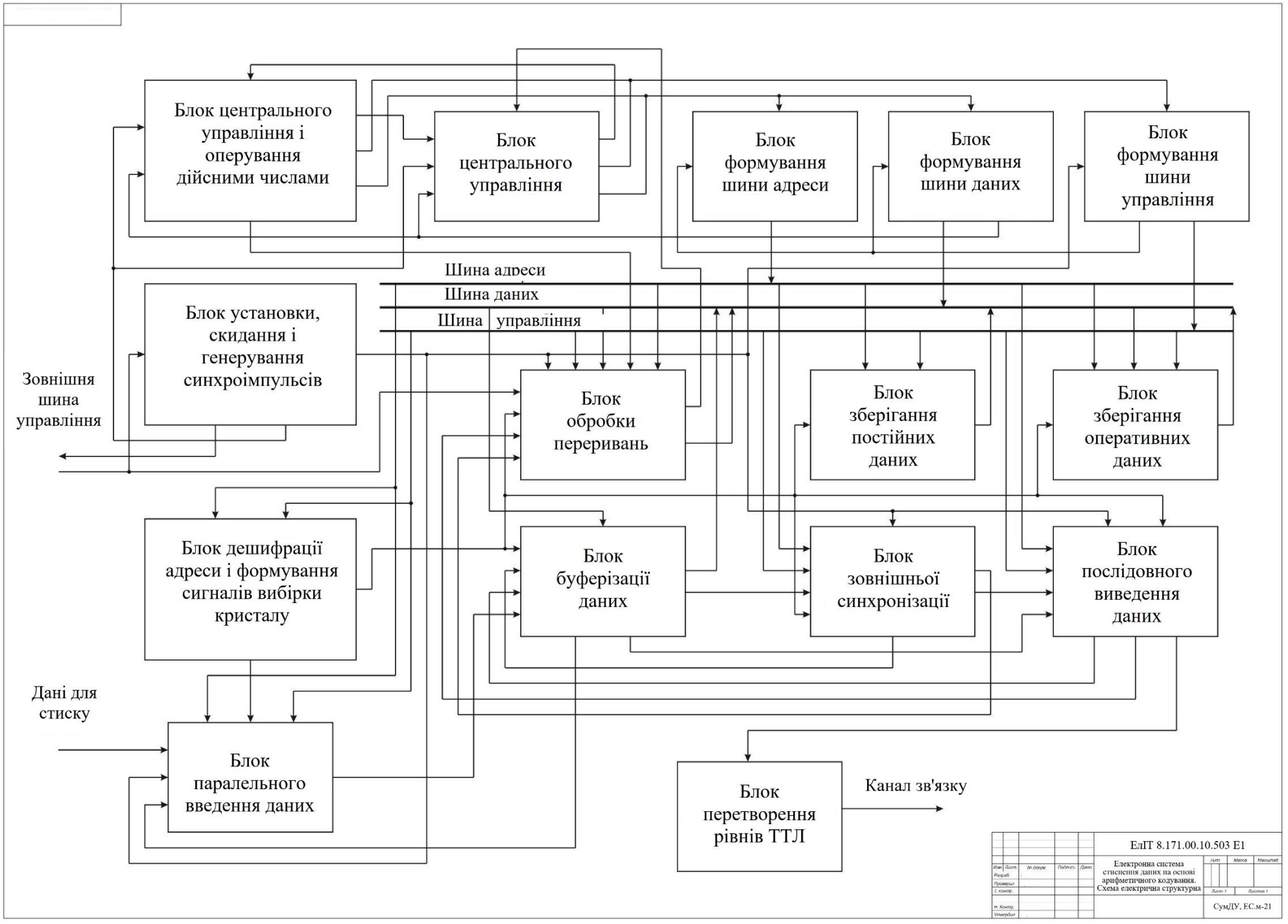
Лит.	Лист	Листов
	91	2

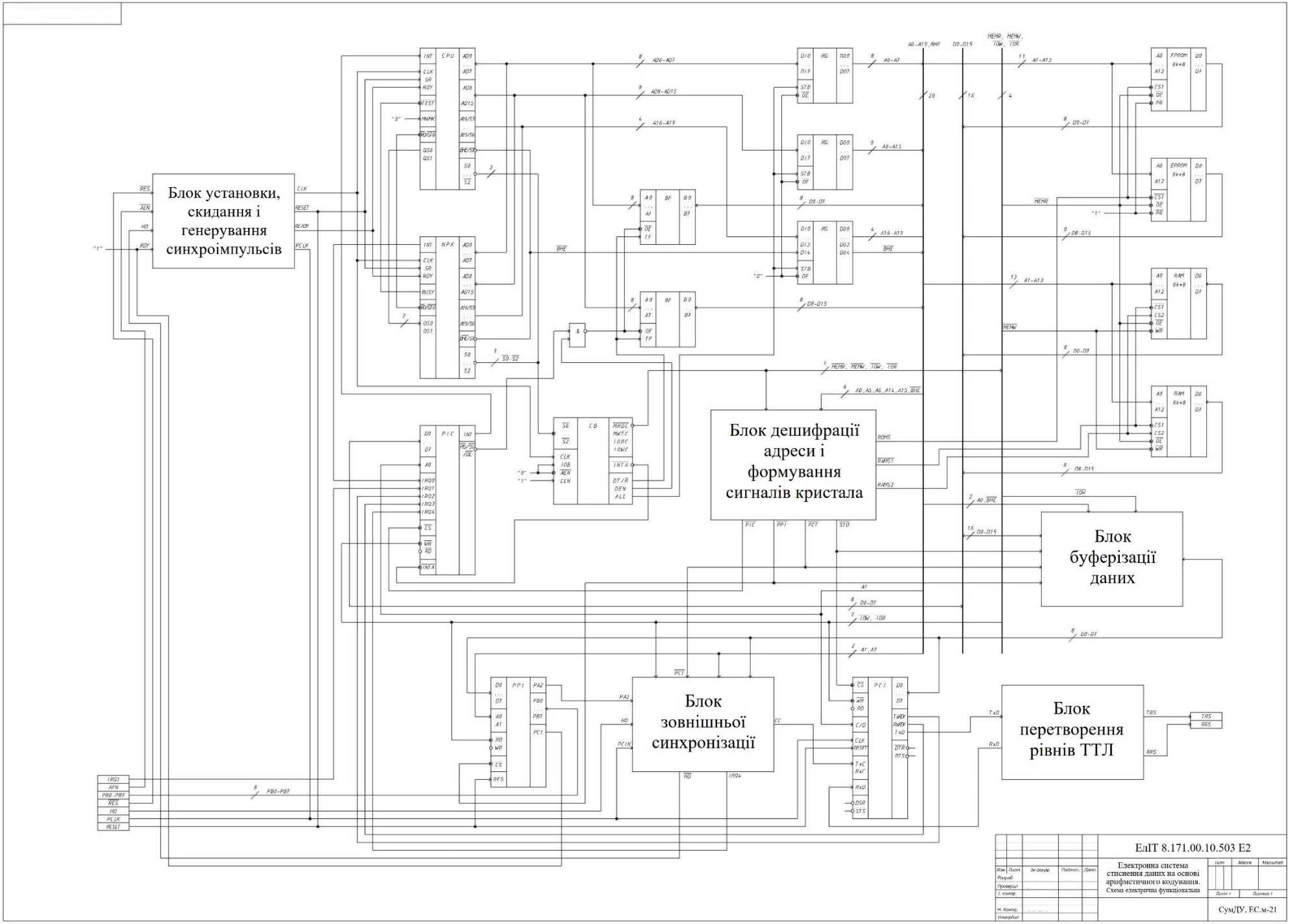
СумДУ, гр. ЕС.м-21

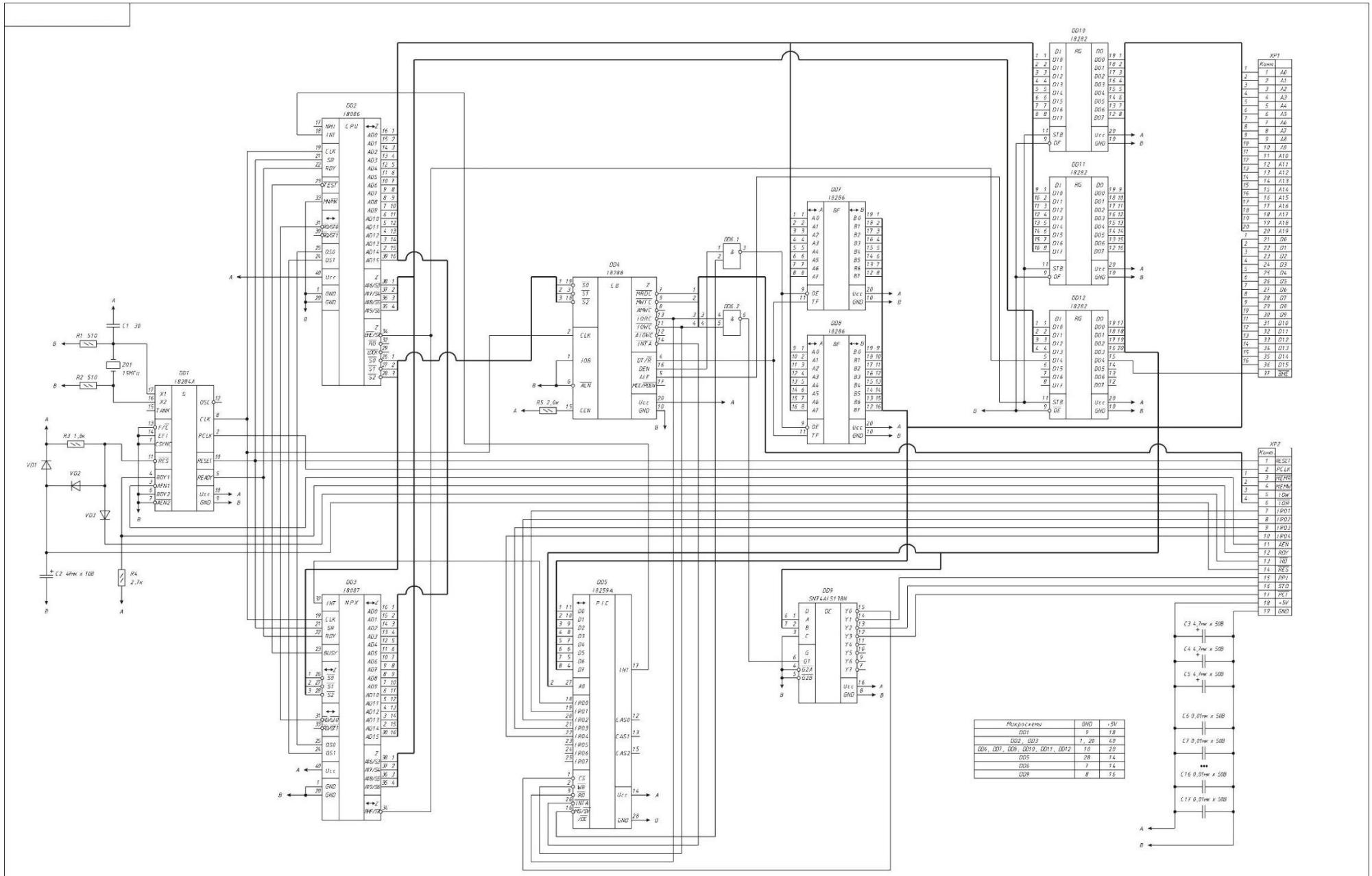
Поз. позн.	Найменування	Кол.	Примітки
	<u>Транзистори</u>		
VT1	КТ361	1	
VT2,VT3	КТ315	2	
	<u>Конденсатори</u>		
C1	КМ-5Б-М47-33пФ	1	
C2	К50-35-10В-47мкФ	1	
C3…C5	К50-35-50В-4,7мкФ	3	
C18…C20	К50-35-50В-4,7мкФ	3	
C6…C17	К10-17Б-Y5V0805-50В-0.01мкФ	12	
C21…C34	К10-17Б-Y5V0805-50В-0.01мкФ	15	
	<u>Резістори</u>		
R1,R2	ЧИП-резистор 0,125Вт-0805-5%-510Ом	2	
R3	ЧИП-резистор 0,125Вт-0805-5%-1,8кОм	1	
R4	ЧИП-резистор 0,125Вт-0805-5%-2,7кОм	1	
R5	ЧИП-резистор 0,125Вт-0805-5%-4,3кОм	1	
R6,R13	ЧИП-резистор 0,125Вт-0805-5%-2,2кОм	2	
R7,R9	ЧИП-резистор 0,125Вт-0805-5%-1кОм	2	
R10,R12	ЧИП-резистор 0,125Вт-0805-5%-1кОм	2	
R8,R11	ЧИП-резистор 0,125Вт-0805-5%-10кОм	2	
	<u>Роз’єми</u>		
XP1,XP3	PLD2-40	2	Вилка штирьова
XP2,XP4	PLD-20	2	Вилка штирьова
XP5	PLD-18	1	Вилка штирьова
XP6	PLD-4	1	Вилка штирьова



						ЕлІТ 8.171.00.10.503 СА	
Изм.	Лист.	№ докум.	Подпись	Дата	Електронна система стиснення даних на основі арифметичного кодування. Схема алгоритму	Лит.	Масса	Масштаб
Разраб.								
Проверил						Лист 1	Листов 1	
Т. контр.								
Н. Контр.						СумДУ, ЕС.м-21		
Утвердил								


Блок центрального управління і оперування дійсними числами
Блок центрального управління
Блок формування шини адреси
Блок формування шини даних
Блок формування шини управління
Шина адреси
Шина даних
Шина управління
Блок установки, скидання і генерування синхроімпульсів
Зовнішня шина управління
Блок обробки переривань
Блок зберігання постійних даних
Блок зберігання оперативних даних
Блок дешифрації адреси і формування сигналів вибірки кристалу
Блок буферізації даних
Блок зовнішньої синхронізації
Блок послідовного виведення даних
Дані для стиску
Блок паралельного введення даних
Блок перетворення рівнів ТТЛ
Канал зв'язку
ЕлІТ 8.171.00.10.503 Е1
Електронна система стиснення даних на основі арифметичного кодування. Схема електрична структурна
СумДУ, ЕС.м-21


Блок установки, скидання і генерування синхроімпульсів
Блок дешифрації адреси і формування сигналів кристала
Блок буферізації даних
Блок зовнішньої синхронізації
Блок перетворення рівнів ТТЛ
CPU
NPX
PIC
PPI
PCI
CB
EPROM 8к*8
RAM 8к*8
ЕлІТ 8.171.00.10.503 Е2
Електронна система стиснення даних на основі арифметичного кодування. Схема електрична функціональна
СумДУ, ЕС.м-21



Микросхемы	GND	+5V
DD1	9	18
DD2 , DD3	1, 20	40
DD4, DD7, DD8, DD10, DD11, DD12	10	20
DD5	28	14
DD6	7	14
DD9	8	16

ЕлІТ 8.171.00.10.503 Е3

Електронна система стиснення даних на основі арифметичного кодування.

Схема електрична принципова

Літ. | Маса | Масштаб

Аркуш 1 | Аркушів 2

СумДУ, ЕС.м-21