**МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ І СПОРТУ УКРАЇНИ**
**ТЕРНОПІЛЬСЬКИЙ ДЕРЖАВНИЙ ЕКОНОМІЧНИЙ УНІВЕРСИТЕТ**
**ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

# ОПОРНИЙ КОНСПЕКТ ЛЕКЦІЙ

з курсу

# "СХЕМОТЕХНІКА СПЕЦПРОЦЕСОРІВ В КОМП'ЮТЕРНИХ СИСТЕМАХ"

*Для студентів спеціальностей:*

*Спеціалізовані комп'ютерні системи*

**ТЕРНОПІЛЬ – 2011**

Опорний конспект лекцій з курсу **“Схемотехніка спецпроцесорів в комп’ютерних системах”** для студентів спеціальностей “Спеціалізовані комп’ютерні системи” та “Комп’ютерні системи та мережі” / Укл.: Яцків В. В. – Тернопіль: Економічна думка, 2011. – 52 с.

Відповідальний за випуск: д.т.н., професор Николайчук Я.М.

Рецензенти: к.т.н., доцент Кочан В.В.;

к.т.н. доцент Сегін А.І.

Методичні вказівки розглянуті та схвалені на засіданні кафедри Спеціалізованих комп’ютерних систем. Протокол № 2

ЗМІСТ

Вступ 3

1. Теоретичні основи системи залишкових класів 5

1.1 Представлення чисел в системі залишкових класів 5

1.2 Методи переведення чисел з 10 в СЗК та їх апаратна реалізація 6

1.3 Метод переведення чисел з СЗК в десяткову систему числення 13

2. Методи і алгоритми обробки даних в системі залишкових класів 15

2.1 Арифметичні операції в СЗК 15

2.2 Коректуючі коди в системі залишкових класів 18

2.3 Алгоритм шифрування даних в СЗК 22

3. Апаратна та програмна реалізація методів та алгоритмів формування та оброблення даних в СЗК 23

3.1 Спецпроцесор виконання арифметичних операцій в СЗК 23

3.2 Програмна реалізація методу виявлення і виправлення помилок 29

3.3 Реалізація алгоритму шифрування даних в СЗК 30

Висновки 33

Список використаної література 34

# ВСТУП

Ресурси обчислювальної техніки, що функціонує в позиційній системі числення, постійно удосконалюються і збільшуються, але вони не можуть бути безмежними у принципі. Для вирішення багатьох наукових, технічних і промислових завдань не вистачає потужності сучасних комп'ютерів. Пошуки нових шляхів підвищення ефективності обчислень привели дослідників до об'єктивного висновку, що в рамках звичайної позиційної системи числення стрибкоподібного прискорення виконання операцій добитися неможливо. Окремі вдосконалення алгоритмів виконання операцій, розвитку архітектурних особливостей і ін. сприяють збільшенню продуктивності, але залишають її в рамках фон-неймановської можливості. Причина полягає в тому, що позиційні системи числення, в яких представляється і обробляється інформація в сучасних ЕОМ, мають важливий недолік – наявність міжрозрядних зв’язків. Вихід з цієї ситуації знайдений в залученні нових ідей у області організації і функціонування ЕОМ на основі залишкової арифметики, яка володіє специфічними характеристиками, що забезпечує ряд переваг.

З огляду на сучасний рівень розвитку обчислювальних засобів використання непозиційних систем числення дозволяє збільшити надійність та швидкість цифрової обробки даних, ввести методи контролю за правильністю виконання операцій без подальшого ускладнення апаратної частини та забезпечувати необхідну точність обчислень без збільшення розрядності шини. Сучасні обчислювальні потужності дозволяють розв’язувати задачі оптимального вибору модулів системи та розрахунку відповідних вагових коефіцієнтів і базисних чисел, що відкриває нові можливості застосування непозиційних систем числення.

Значний внесок в розвиток теоретичних основ системи залишкових класів СЗК) зробили Акушський І. Я. , Юдицкий Д. І., Варіченко Л.В. [1, 2, 3] методів цілочисельно перетворення Николайчук Я.М. [5].

# 1. ТЕОРЕТИЧНІ ОСНОВИ СИСТЕМІ ЗАЛИШКОВИХ КЛАСІВ

## 1.1 Представлення чисел в системі залишкових класів

Нехай задано набір із $k$ взаємопростих ( взиємопрості числа – це такі числа, які діляться лише на самих себе і на одиницю) натуральних чисел $p_i \in N$, $i = \overline{1,k}$, тоді під СЗК будемо розуміти таку систему, в якій ціле число представляється у вигляді невід'ємних залишків по вибраних модулях $p_i$ [1-4]:

$$b_i = res\, N \,(\mathrm{mod}\, p_i),\ i = \overline{1,k}. \tag{1.1}$$

Даний вираз відповідає системі діофантових рівнянь:

$$N = c_i \cdot p_i + b_i,\ i = \overline{1,k}, \tag{1.2}$$

де $N$ – вихідна величина;

$p_i$ – набір модулів;

$b_i$ – набір залишків по відповідних модулях;

$c_i$ – ранг числа $N$ по модулю $p_i$.

Діапазон чисел, що може бути представлений за допомогою набору модулів ( $p_1$, $p_2$, …, $p_{k-1}$ $p_k$ ) становить [0, $\wp$] з формули:

$$\wp = \prod_{i=1}^{k} p_i . \tag{1.3}$$

Розглянемо представлення числа N, яке задане в позиційній системі числення з основою d, в СЗК з набором модулів ( $p_1$, $p_2$, …, $p_{k-1}$ $p_k$ ). Згідно означення СЗК та рівняння (1.1) число N буде представлено у вигляді залишків $b_i$, $i = \overline{1,k}$. Використання методу є доцільним при умові здійснення перетворення людиною, оскільки операція визначення залишку по заданому модулю триває не довше, ніж, наприклад, запис результатів обчислення. При використанні автоматизованих обчислювальних засобів операція ділення є найбільш довготривалою, в порівнянні з усіма іншими операціями (збереження даних, додавання, множення та інші).

## 1.2 Методи переведення чисел з 10 в СЗК та їх апаратна реалізація

Методи переводу чисел з десяткової системи числення в систему залишкових класів є різними, однак в даній роботі ми покажемо алгоритми і апаратну реалізацію методів переводу чисел з 10 системи числення в СЗК, а саме:

переведення чисел з 10 в СЗК за допомогою лічильника;

переведення чисел з 10 в СЗК методом віднімання;

переведення чисел з 10 в СЗК методом булевих функцій;

переведення чисел з 10 в СЗК методом булевих функцій і відніманням універсальним методом.

Перший метод переводу полягає у використанні десяткового декриментуючого лічильника. Алгоритм роботи схеми переводу чисел наступний. Спочатку дані представленні в десятковій системі числення подаються на інформаційний вхід декриментуючий синхронний лічильник і при подачі на вхід завантаження логічної одиниці – дані загружаються в лічильник. На вхід синхронізації подаються імпульси з тактового генератора. Коли вхід завантаження лічильника скидається в нуль (загрузка даних в лічильник відбулася), то на вихід дозволу лічильника подається логічна одиниця, що дозволяє подачу тактових імпульсів на лічильники по відповідному модулю. При кожному наступному імпульсі вміст лічильника зменшується на одиницю, а вміст лічильників по відповідних модулях збільшується на одиницю.

Принцип роботи лічильника, наприклад по модулю 5, полягає в наступному. При кожному наступному надходженні на вхід синхронізації тактового імпульсу його вміст збільшується на одиницю, коли вміст лічильника стає рівним 5, то він скидається в нуль. В результаті, на виході лічильника буде залишок кількості вхідних тактових імпульсів по модулю 5.

По даному алгоритму працюють і всі інші лічильники.

Функціональна схема пристрою переводу чисел даним методом представлена на рисунку 1.1.

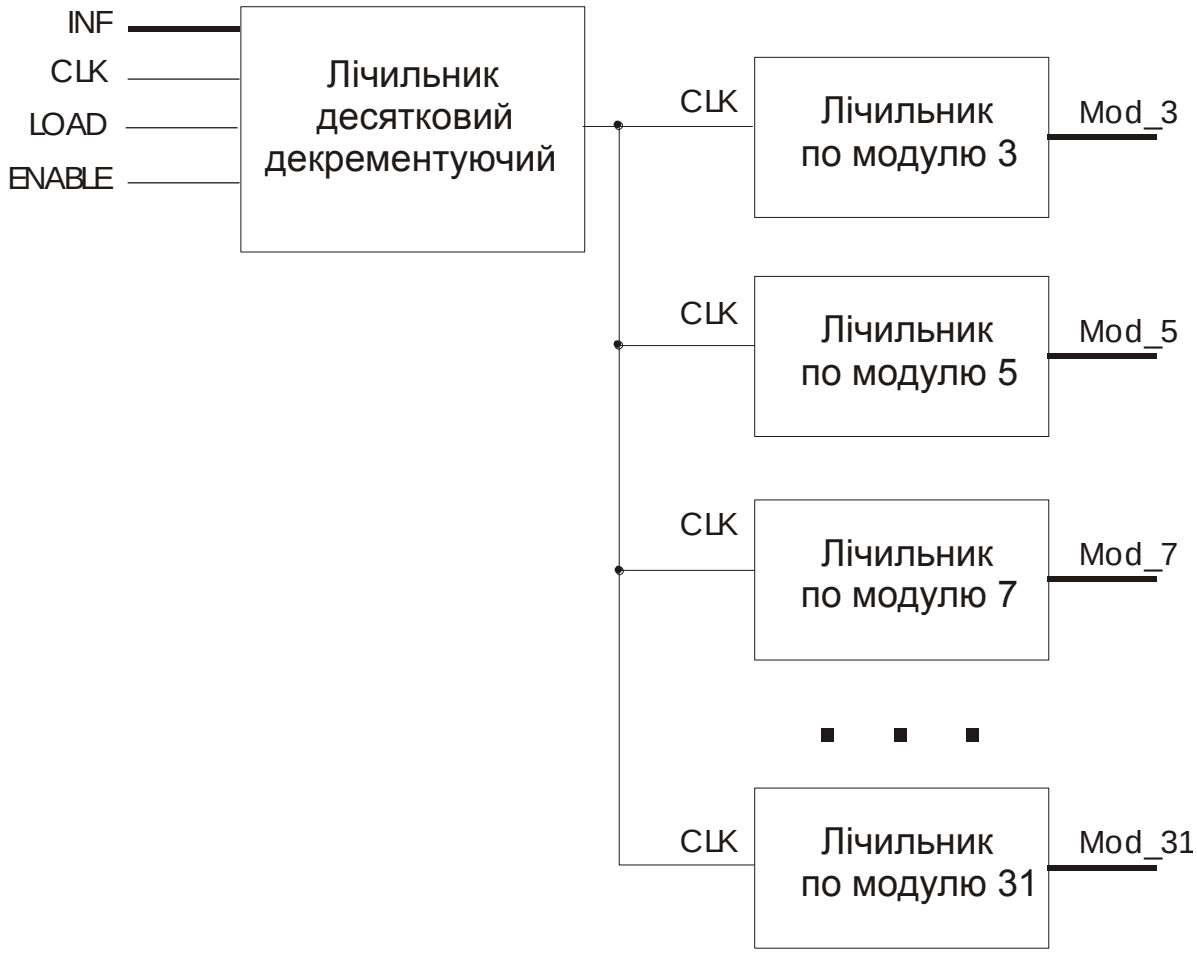


Рис 1.1 – Функціональна схема пристрою переведення чисел з 10 в СЗК з використанням лічильника.

Отже, коли вміст дикрементуючого лічильника стає рівним нулю (кількість тактів поданих на вхід синхронізації рівний десятковому представленню вхідного числа) то припиняється подача імпульсів на входи синхронізації інкрементуючих лічильників по відповідних модулях, а кількість імпульсів становитиме числу поданому на вхід дикрементуючого лічильника. На виході схеми буде вхідне число, представлене в системі залишкових класів.

Отже, даний метод є дуже простим в апаратній реалізації, оскільки потрібно лише один дикреметнуючий лічильник і стільки інкреметуючих лічильників, скільки основ в СЗК.

Недоліком даного методу переводу чисел з десяткової системи числення в систему залишкових класів є те, що сама процедура переводу здійснюється досить довго.

Для прикладу для переводу числа 10 000 нам потрібно 10 000 імпульсів тактового генератора. Отже, даний метод  не є найкращим, однак він має право на існування через свою простоту реалізації.

Другий метод базується на використанні дикриментуючих лічильників по відповідних модулях.

Алгоритм роботи схеми переводу чисел з десяткової системи числення в систему залишкових класів полягає в наступному. Вхідні дані, представленні в десятковій системі числення, паралельно подаються на інформаційні входи синхронних дикриментуючих лічильників по відповідних модулях. При поданні на вхід завантаження логічної одиниці дані завантажуються в лічильники. Після того, як вхід завантаження скидається в нуль (дані завантаженні в лічильники) і на вхід синхронізації подається імпульс з тактового генератора, вміст лічильників зменшується на відповідний модуль. Коли вміст лічильника стає меншим за даний модуль, то на вихід подається результат віднімання (залишок від вхідного числа по даному модулю) і на вихід готовності подається логічна одиниця, що сигналізує про закінчення операції переводу числа по даному модулю.

Перевід числа з десяткової системи числення в систему залишкових класів завершене тоді, коли на виходах готовності всіх модулів буде логічна одиниця.

Функціональна схема спецпроцесора переведення чисел представлена на рисунку 1.2.

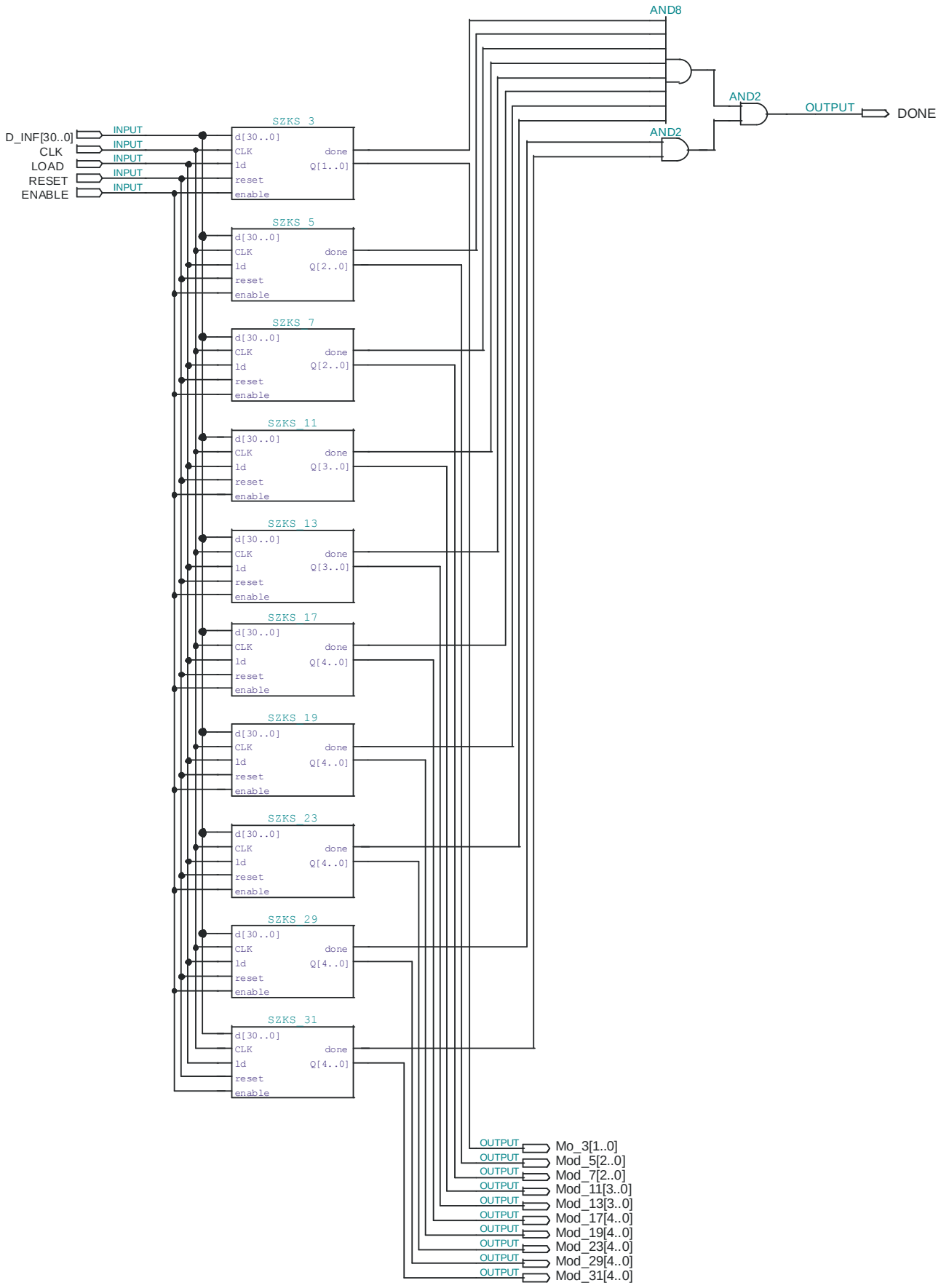


Рис 1.2 Функціональна схема спецпроцесора переведення чисел з 10 в СЗК методом віднімання.

Перевагою даного методу переводу є простота апаратної реалізації, оскільки потрібно лише блоки віднімання по відповідних модулях (які легко реалізувати на ПЛМ). Також перевагою даного методу над методом розглянутим вище є те, що потрібно менше часу для здійснення операції переводу.

Для прикладу, для переводу того ж таки числа 10 000 з десяткової системи числення в систему залишкових класів нам потрібно не 10 000 тактів, а

$$\frac{10000}{m}+1,$$

де $m$ – найменший модуль системи залишкових класів.

Третій метод переводу чисел з 10 в СЗК полягає у використанні системи булевих функцій.

При подачі на вхід завантаження позитивного імпульсу дані загружаються в модулі, в яких відбувається перевід вхідних чисел з 10 системи числення в систему залишкових класів, який здійснюється за один такт.

Даний метод має перевагу в тому, що перетворення здійснюється дуже швидко, за один такт, однак недоліком є те, що він є досить великим в плані пам’яті.

Суть методу переводу чисел з 10 в СЗК за допомогою булевих функцій полягає в наступному.

Для кожного модуля складається таблиця (в даному прикладі – для модуля 3):

Таблиця 1.1 Таблиця переводу чисел по модулю 3

| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 | 2 | 0 | 1 |

В даній таблиці перший рядок – це діапазон вхідних чисел (в даному випадку від 0 до 13), а другий – залишки по відповідному модулю вхідних чисел.

Далі дану таблицю представимо в двійковій системі числення:

Таблиця 1.2 Таблиця переводу чисел в двійковій системі числення

| 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 00 | 01 | 10 | 00 | 01 | 10 | 00 | 01 | 10 | 00 | 01 | 10 | 00 | 01 |

Наступним кроком є розбиття таблиці на стільки таблиць, скільки розрядів у даному, модулю по кожному розряду (в нашому випадку на дві таблиці).

Таблиця 1.3 Таблиця переводу чисел 0-й розряд

| 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |

З даної таблиці записуємо рівняння кон'юнкції для одиниць для першого розряду виходу:

$$y0 = x0 \wedge \overline{x1} \wedge \overline{x2} \wedge \overline{x3} \vee \overline{x0} \wedge \overline{x1} \wedge \overline{x2} \wedge x3 \vee x0 \wedge x1 \wedge x2 \wedge \overline{x3} \vee \overline{x0} \wedge x1 \wedge \overline{x2} \wedge x3 \vee$$
$$\vee x0 \wedge \overline{x1} \wedge x2 \wedge x3$$

Таблиця для другого розряду виходу буде наступною:

Таблиця 1.4 Таблиця переводу чисел 1-й розряд

| 0000 | 0001 | 0010 | 0011 | 0100 | 0101 | 0110 | 0111 | 1000 | 1001 | 1010 | 1011 | 1100 | 1101 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |

З даної таблиці записуємо рівняння кон'юнкції для одиниць для другого розряду виходу:

$$y1 = \overline{x0} \wedge x1 \wedge \overline{x2} \wedge \overline{x3} \vee x0 \wedge \overline{x1} \wedge x2 \wedge \overline{x3} \vee \overline{x0} \wedge \overline{x1} \wedge \overline{x2} \wedge x3 \vee x0 \wedge x1 \wedge \overline{x2} \wedge x3$$

Як бачимо, складання системи булевих функцій є досить клопіткою роботою (оскільки, чим більший діапазон представлення чисел і чим більший модуль тим більше буде рівнянь і тим більше пам'яті буде зайнято в ПЛМ) і потребує великої концентрації, щоб не припуститись помилки. Однак, для полегшення роботи було складено програму на мові програмування C++ for DOS, яка автоматично генерує файл з розширення *.vhd, який надалі потрібно лише скомпілювати.

Четвертий метод переводу чисел з 10 в СЗК дає змогу набагато швидше здійснювати переведення.

Суть даного методу полягає в наступному.

Для прикладу здійснимо перевід десяткового числа 1985497 в СЗК за основою 7. спочатку беремо перше число (1) і знаходимо залишок за основою 7 (залишок становитиме 1), далі даний залишок домножуємо на 10 і додаємо наступне число (9), отримаємо число 19, залишок за основою 7 від даного числа становитиме 5. Наступне число 58, залишок – 2, далі – 25, залишок – 4, число 44, залишок – 2, число 29, залишок – 1, число 17, залишок – 3.

Отже, залишок від числа 1985497 за основою 7 становитиме 3. І дійсно, перевіривши, ми дійшли висновку, що цей метод є правильним і залишок становить 3.

Сама схема переводу чисел з десяткової системи числення в систему залишкових класів даним методом складається з одного блоку керування, в якому здійснюється підготовка числа до наступного етапу переведення (тобто почерговий вибір наступної цифри двійково-десяткового представлення вхідного числа і додавання його до домноженого на 10 залишку від попередньої операції). Функціональна схема блоку переводу чисел з 10 в СЗК за допомогою булевих функцій представлена на рис.1.3.

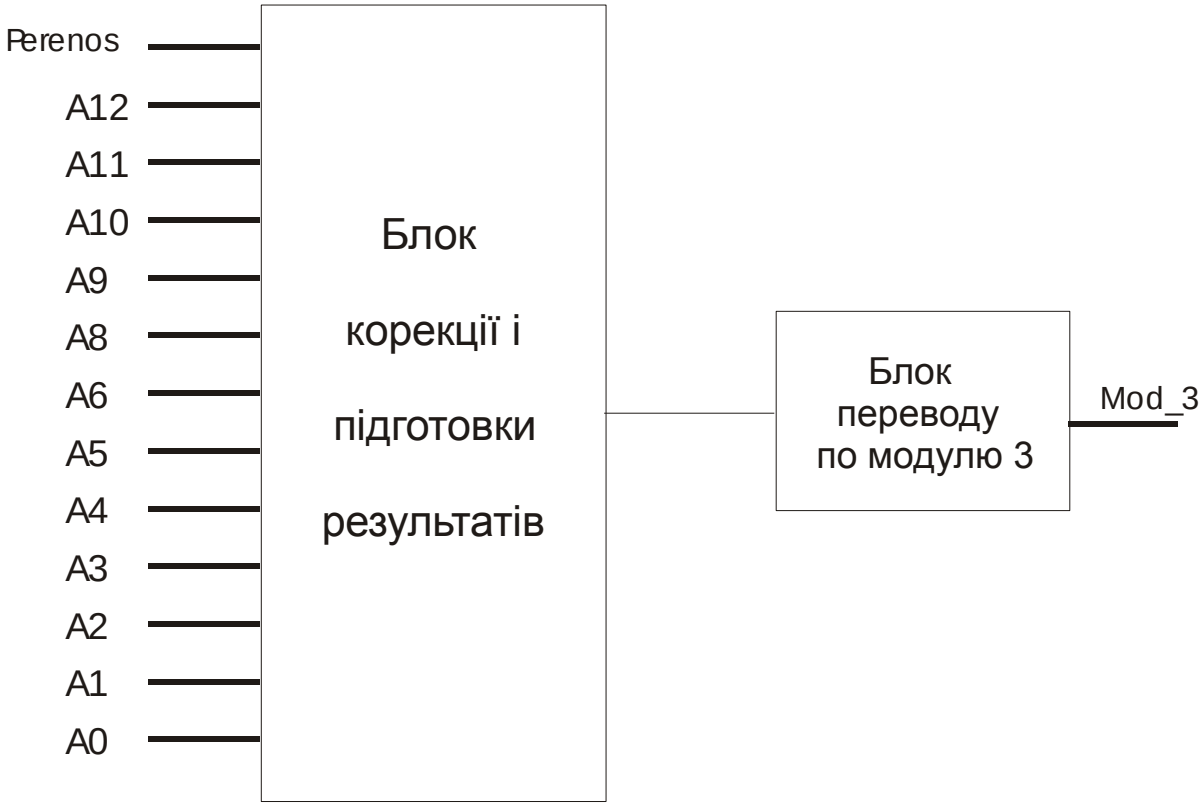


Рис 1.3 Функціональна схема блоку переводу чисел з 10 в СЗК за допомогою булевих функцій по модулю 3

Вагомою перевагою даного методу є те, що операція переводу числа з 10 системи числення в систему залишкових класів відбувається набагато швидше, ніж при використанні інших методів переводу. Для переводу будь-якого числа нам потрібно 13 тактів тактового генератора.

Недоліком є те, що вхідне число повинно бути представленим в двійково-десятковому форматі, а також, якщо число є невеликим, наприклад, 130, то на його перевід в СЗК нам необхідно все одно 13 тактів.

## 1.3 Метод переведення чисел з СЗК в десяткову систему числення

Процес переводу числа з системи залишкових класів в десяткову систему числення працює за наступним алгоритмом:

Залишки по відповідним основам поступають на відповідні входи. Далі, при подавання на вхід синхронізації позитивного імпульсу, береться залишок по

найбільшому модулю і прирівнюється до залишку другого по величині модуля. Якщо залишок по більшому модулю більший ніж залишок по меншому, тоді до даного залишку (по меншому модулю) додається його модуль. Якщо залишок по більшому модулю менший ніж залишок по меншому, тоді до залишку по більшому модулю додається більший модуль. Якщо вони рівні, то відбувається перевірка між найбільшим модулем і третім по величині модулем. Так відбувається доти, доки всі залишки по всіх модулях не будуть рівними між собою, а це буде лише тоді, коли вони будуть рівними десятковому представленню числа.

Для прикладу здійснимо перевід числа (14,10,4,2) по модулях 19, 23, 29, 31.

Оскільки 2<4, то до числа 2 додається модуль 31 і отримуємо число 33.

Оскільки 4<33, то до числа 4 додаємо модуль 29 і отримуємо число 33.

Оскільки дані два числа (33 і 33) рівні між собою, то переходимо до порівняння наступного залишку.

10<33, то до числа 10 додаємо 23 і отримаємо число 33.

Оскільки воно рівне 33, переходимо до наступного числа.

Оскільки 14<33, то до нього додаємо число 19 і отримаємо 33.

Всі чотири числа рівні між собою, тоді десяткове представлення числа (14, 10, 4, 2) по основах 19, 23, 29, 31 становить 33, що є вірним рішенням.

Розроблений метод переводу чисел з системи залишкових класів в десяткову систему числення дозволяє зменшити затрати часу та спростити апаратну реалізацію.

# 2. МЕТОДИ ТА АЛГОРИТМИ ОБРОБЛЕННЯ ДАНИХ В СЗК

## 2.1 Арифметичні операції в СЗК

Розглянемо правила виконання операцій додавання і множення в СЗК при умові, що обидва числа і результат операції знаходяться в діапазоні $[0, \wp]$.

Нехай операнди A і B представлені відповідно залишками $\alpha_i$ і $\beta_i$ по модулю $P_i$ при $i = 1, 2, \ldots, n$.

Результат операцій додавання визначається за формулою:

$$A + B \,(\mathrm{mod}\, P) = \begin{cases} \alpha_i + \beta_i, & \text{якщо} \quad \alpha_i + \beta_i < P_i; \\ \alpha_i + \beta_i - P, & \text{якщо} \quad \alpha_i + \beta_i \geq P_i. \end{cases}$$

Для множення:

$$\delta_i = A \cdot B - [\frac{A \cdot B}{P_i}] \cdot P_i .$$

Приклад: нехай основою системи є $P_1 = 3$, $P_2 = 5$, $P_3 = 7$.

Діапазон представлення чисел за допомогою вибраних модулів визначається, як $\wp = P_1 \cdot P_2 \cdot P_3 = 105$.

Додати числа A=17 і B=63. По вибраних модулях числа A і B в системі залишкових класів будуть представлені як:

$$A = 17 = (2, 2, 3)_{(3,5,7)}, \; B = 63 = (0, 3, 0)_{(3,5,7)}.$$

В відповідності з (2.1) отримаємо

$$A + B = (2, 0, 3)_{(3,5,7)}.$$

Легко перевірити, що число $(2, 0, 3)_{(3,5,7)}$ в десятковій системі числення є 80 і дорівнює сумі операндів.

Приклад 2. Помножити число A=17 на число B=6.

В СЗК числа A і B будуть представлені як

$$A = 17 = (2, 2, 3)_{(3,5,7)}, \; B = 6 = (0, 1, 6)_{(3,5,7)}.$$

В відповідності з (2.2) отримаємо $A \cdot B = (0, 2, 4)_{(3,5,7)}$.

Легко перевірити, що число $(0, 2, 4)_{(3,5,7)}$ в СЗК дорівнює десятковому числу 102 в десятковій системі числення і рівне добутку операндів.

Правила виконання операції віднімання в СЗК в випадку, якщо два числа і результат операції знаходяться в діапазоні $[0, \wp]$.

Нехай операнди A і B представлені відповідними залишками $\alpha_i$ і $\beta_i$ по модулях $P_i$ при $i = 1, 2, \ldots, n$.

Результат операції віднімання A-B представлений відповідними залишками $\gamma_i$ по тих же модулях $P_i$.

Тобто

$$A = (\alpha_1, \alpha_2, .., \alpha_n),\ B = (\beta_1, \beta_2, .., \beta_n),$$

$$A - B = (\gamma_1, \gamma_2, .., \gamma_n),$$

і при цьому виконуються умови:

$$A < \wp,\ B < \wp,\ 0 \le A - B < \wp.$$

Операція віднімання в тих випадках, коли її результат додатній, виконується відніманням відповідних цифр розрядів, при цьому завжди в результаті приводиться найменший додатній залишок, так як це випливає із визначення СЗК. Якщо різниця цифр від'ємна, то береться її доповнення до відповідного модуля.

Тобто

$$A - B \,(\mathrm{mod}\, P) = \begin{cases} \alpha_i - \beta_i, & \text{якщо} \quad \alpha_i - \beta_i \ge 0; \\ \alpha_i - \beta_i + P, & \text{якщо} \quad \alpha_i - \beta_i < 0. \end{cases}$$

Приклад віднімання двох чисел в СЗК. C=A-B.

$A = 17 = (2, 2, 3)_{(3,5,7)}$, $B = 6 = (0, 1, 6)_{(3,5,7)}$,

$C = (2-0, 2-1, 3-6+7) = (2, 1, 4)_{(3,5,7)}$, $C = 11 = (2, 1, 4)_{(3,5,7)}$.

Перевагою виконання математичних операцій в СЗК є можливість паралельної оброки

Розглянемо можливе застосування СЗК в схемотехнічній реалізації комп'ютерів. Нехай ми маємо два комп'ютери: один із двійковою системою числення, другий на основі СЗК, що зберігає числа й оперує ними по описаним вище методах.

Визначимо скільки звичайному комп'ютеру необхідний виконати мікрооперацій для додавання, наприклад, чисел A=23576 і B=31820. Обидва

числа є 16-ти розрядними, отже, потрібно послідовно виконати 16 мікрооперацій C=A+B:

A=0101110000011000 (23576);

B=0111110001001100 (31820);

C=1101100001100100 (55396).

Спочатку складаються останні розряди обох чисел, потім передостанні з урахуванням можливого переносу від останніх, потім перед-передостанні і т.д. Сумарні витрати часу на додавання двох 16-ти розрядних чисел - це 16 умовних одиниць часу (одна одиниця часу необхідна на виконання однієї елементарної однорозрядної операції). Аналогічні міркування відносяться також до 32-м і 64-х розрядних чисел.

Тепер визначимо наскільки швидко міг би виконати це ж додавання комп'ютер на основі СІК. Нехай СЗК має модулі 5, 7, 11, 13 і 17. Тоді:

A={ 1, 0, 3, 7, 14} (23576);

B={ 0, 5, 8, 9, 13} (31820);

C={1, 5, 0, 3, 10} (55396);

C=A+B.

Основна відмінність полягає у відсутності переносів між розрядами і можливості не послідовного, а паралельного виконання додавання в різних розрядах чисел. Тобто сумарний час додавання визначається найбільшим часом виконання додавання лише в одному розряді. Якщо додавання "всередині" розрядів здійснювати заздалегідь прошитою матрицею для кожного модуля, що цілком можливо при відносно невеликих значеннях модулів, то цей час буде дорівнювати часу виконання елементарної однобітної операції.

Таким чином, використовуючи СЗК, швидкодію виконання операції додавання, що виконується в процесорі усього за один такт, можна було б для 64-х розрядних чисел збільшити в 64 рази. Однак особливу актуальність СЗК має для операції множення, що є однією із самих повільних операцій у процесорах (вимагає кілька десятків тактів). Це значно підвищило б загальну продуктивність комп'ютера.

Однією з основних переваг цілочисельного перетворення СЗК є незалежність розрядів числа, що створює можливості для розпаралелення обробки інформації і підвищення загальної продуктивності обчислювальних засобів.

Ще однією важливою перевагою СЗК є можливість виявляти та виправляти помилки як в процесі виконання арифметичних операцій, так і в процесі передавання даних.

## 2.2 Коректуючі коди в системі залишкових класів

В даний час в зв'язку з розробкою машинної арифметики в системі залишкових класів виникла можливість побудови непозиційних кодів, які виявляють і виправляють помилки, і разом з тим повністю арифметичних кодів, де інформаційна і контрольна частини повністю рівноправні відносно будь-якої операції.

Розглянемо систему з основами $p_1, p_2, \ldots, p_n$ і діапазоном $\wp = p_1 \cdot p_2 \cdot \ldots \cdot p_n$. В подальшому діапазон $\wp$ будемо називати робочим діапазоном. Введемо основу $p_{n+1}$ взаємопросту з будь-якою із прийнятих основ і будемо представляти числа в системі із основами $n+1$. Це означає, що будемо передавати числа і виконувати операції над числами, які знаходяться в діапазоні $[0, \wp]$ в більш широкому діапазоні $[0, P]$, де $P = \wp \cdot p_{n+1}$.

В подальшому діапазон $P$ будемо називати повним діапазоном системи з однією контрольною основою.

Так як всі числа з якими працює обчислювальна машина, повинні знаходитись в діапазоні $[0, \wp]$, то зрозуміло, якщо в результаті будь-якої операції або при передаванні числа вийшло, що одержано число А, більше $\wp$, це означає, що при виконанні операції була допущена помилка.

Отже, числа менші $\wp$ будемо називати правильними, а більші $\wp$ – неправильні [1].

Нехай основи $p_1, p_2, \ldots, p_n, p_{n+1}$ системи залишкових класів задовольняють умові

$$p_i < p_{n+1}, \quad i = 1, 2, \ldots, n,$$

і нехай $A = (\alpha_1, \alpha_2, \ldots, \alpha_i, \ldots, \alpha_n, \alpha_{n+1})$ – правильне число.

Тоді число $\tilde{A} = \left( \alpha_1, \alpha_2, \ldots, \tilde{\alpha}_i \neq \alpha_i, \ldots, \alpha_n, \alpha_{n+1} \right)$, де $i = 1, 2, \ldots, n, n+1$ є неправильне число.

Правильність числа А визначається із виразу

$$A < \frac{P}{p_{n+1}},$$

але так як

$$\frac{P}{p_i} \geq \frac{P}{p_{n+1}}, \quad i = 1, 2, \ldots, n, n+1,$$

то відповідно

$$A < \frac{P}{p_i}.$$

Так як $\tilde{\alpha} \neq \alpha_i$, число $\tilde{A}$ не може знаходитись в діапазоні $\left[ 0, \frac{P}{p_i} \right]$,

а відповідно має місце

$$\tilde{A} > \frac{P}{p_{n+1}},$$

тобто $\tilde{A}$ є неправильним числом.

Приведені залежності доводять можливість побудови коректуючих кодів в системі залишкових класів. Отже, будь-яке спотворення цифри по одному розряді перетворює це число в неправильне і відповідно дозволяє виявити наявність помилки. Більше того, існує тільки одно значення цієї цифри, яке може перетворити неправильне число в правильне.

Контрольна основа $p_{n+1}$ повинна бути більшою за будь-яку іншу основу системи.

Порівняння коректуючих властивостей кодів СЗК проведемо з кодами Хемінга (табл. 2.1) та кодами Боуза – Чоудхурі – Хоквінгема (табл. 2.2).

Таблиці 2.1 – Параметри кодів Хемінга

| Кількість бітів даних | Виправлення 1-ї помилки | | Виправлення 1-ї помилки; виявлення 2-х помилок | |
|---|---|---|---|---|
| | Кількість контрольних бітів | Збільшення блоку в % | Кількість контрольних бітів | Збільшення блоку в % |
| 8 | 4 | 50 | 5 | 62,5 |
| 16 | 5 | 31,25 | 6 | 37,5 |
| 32 | 6 | 18,75 | 7 | 21,875 |
| 64 | 7 | 10,94 | 8 | 12,5 |
| 128 | 8 | 6,25 | 9 | 7,03 |
| 256 | 9 | 3,52 | 10 | 3,91 |

Таблиці 2.2 – Параметри кодів Боуза – Чоудхурі – Хоквінгема

| n | k | t | n | k | t | n | k | t | n | k | t | n | k | t |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 7 | 4 | 1 | 63 | 30 | 6 | 127 | 64 | 10 | 255 | 207 | 6 | 255 | 99 | 23 |
| 15 | 11 | 1 | | 24 | 7 | | 57 | 11 | | 199 | 7 | | 91 | 25 |
| | 7 | 2 | | 18 | 10 | | 50 | 13 | | 191 | 8 | | 87 | 26 |
| | 5 | 3 | | 16 | 11 | | 43 | 14 | | 187 | 9 | | 79 | 27 |
| 31 | 26 | 1 | | 10 | 13 | | 36 | 15 | | 179 | 10 | | 71 | 29 |
| | 21 | 2 | | 7 | 15 | | 29 | 2 | | 171 | 11 | | 63 | 30 |
| | 16 | 3 | 127 | 120 | 1 | | 22 | 23 | | 163 | 12 | | 55 | 31 |
| | 11 | 5 | | 113 | 2 | | 15 | 27 | | 155 | 13 | | 47 | 42 |
| 63 | 6 | 7 | | 106 | 3 | | 8 | 31 | | 147 | 14 | | 45 | 43 |
| | 57 | 1 | | 99 | 4 | 255 | 247 | 1 | | 139 | 15 | | 37 | 45 |
| | 51 | 2 | | 92 | 5 | | 239 | 2 | | 131 | 18 | | 29 | 47 |

На рисунку 2.1 і рисунку 2.2 приведені графічні залежності порівняння завадостійких кодів здатних виявляти і виправляти однократні помилки.

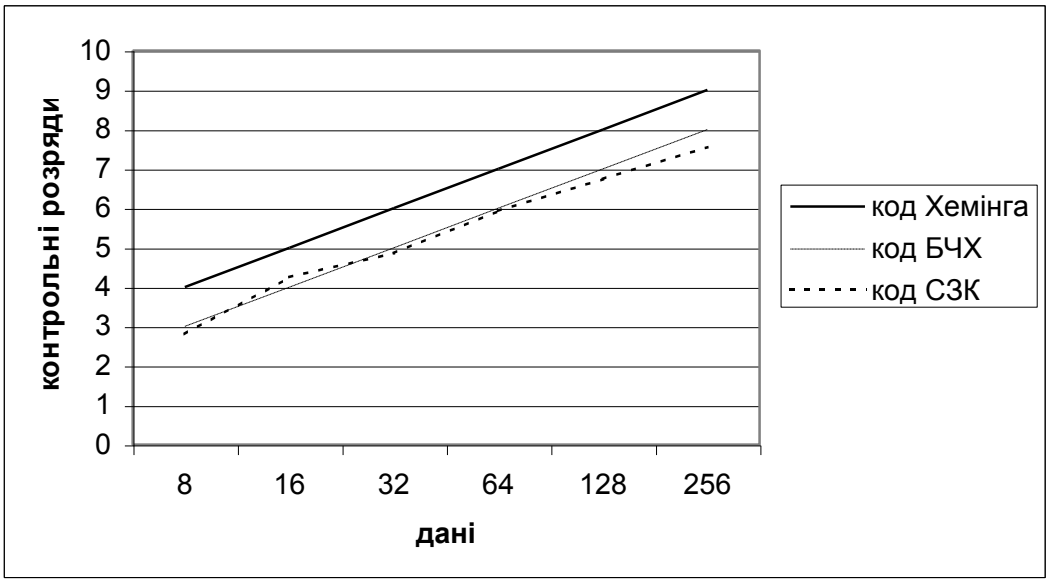


Рисунок 2.1 – Залежність кількості контрольних розрядів від розрядності блоку даних.

Як видно із представлених графічних залежностей коди системи залишкових класів мають кращі параметри від кодів Хемінга і практично однакові з кодами Боуза – Чоудхурі – Хоквінгема, які відносяться до кращих циклічних кодів.

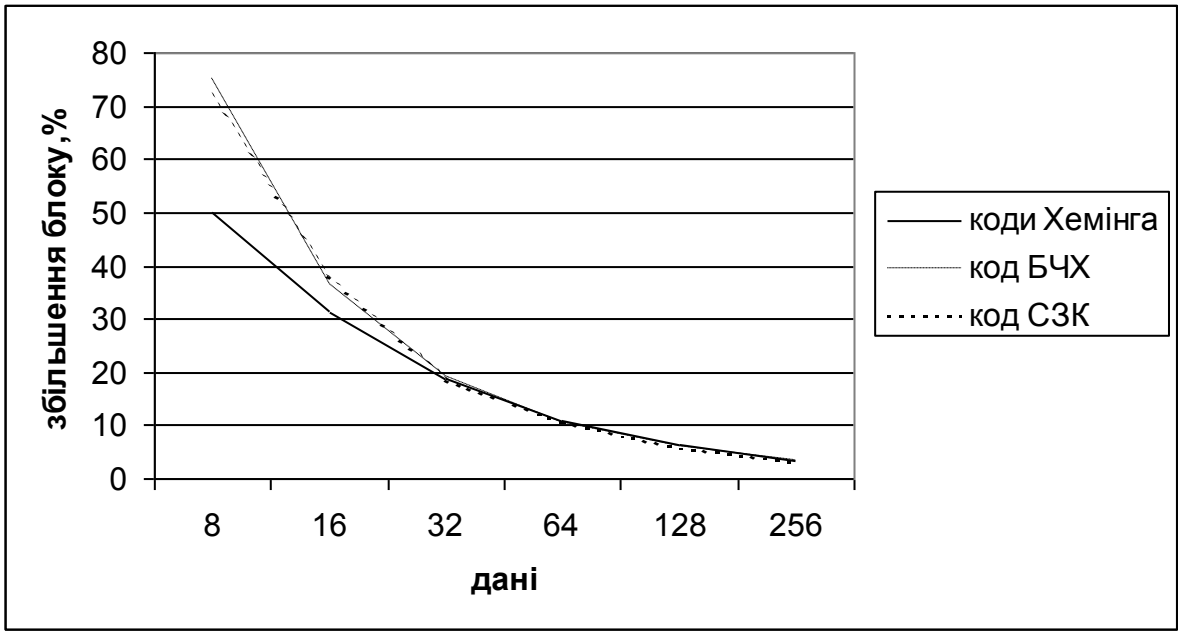


Рисунок 2.2 – Збільшення розрядності блоку в залежності від розрядності даних.

Важливою перевагою кодів системи залишкових класів є їх арифметичність, тобто здатність контролювати правильність виконання будь-яких математичних операцій.

## 2.3 Алгоритми шифрування даних в СЗК

При зберіганні, передачі і обміні електронною інформацією в інформаційних системах і мережах виникає проблема забезпечення її конфіденційності (захист від атак), встановлення аудентифікації (достовірності) автора і її цілісності (відсутності змін в отриманому електронному повідомленні). Конфіденційність може бути забезпечена застосуванням криптографічних методів (шифрування). Задачу встановлення цілісності повідомлення і правдивості автора дозволяє ефективно вирішувати електронний цифровий підпис – відносно коротка додаткова інформація, яка передається разом з підписаним текстом.

Відомі методи шифрування, схеми формування електронного цифрового підпису і стандарти розроблені для позиційної системи числення. Істотно підвищити криптостійкість алгоритмів шифрування, а також скоротити довжину електронного цифрового підпису дозволяють нетрадиційні методи криптографії на основі непозиційної системи числення.

Пропонуються алгоритми шифрування тексту повідомлення і формування електронного цифрового підпису в непозиційній поліомінальній системі, в якій криптостійкість залежить не лише від довжини ключа, але і від вибраної системи поліномінальних основ, а також їх розподілу (порядку слідування).

Розглянемо метод шифрування даних в системі залишкових класів.

Якщо розглядати повідомлення як таке, що складається з байтів, то потрібно вибрати систему таких взаємопростих основ, щоб кожна з них була більшою за 255 (оскільки діапазон представлення ASCII символів становить від 0 до 255). Надалі обробляти будемо байти даних, тому вхідні дані можна розглядати як залишки по відповідним основам системи залишкових класів.

Далі, відповідно до обраних основ системи залишкових класів, їх кількості і порядком слідування, переходимо з системи залишкових класів до десяткової системи числення. В результаті чого ми отримаємо десяткове число, яке і є закодованою інформацією.

# 3. АПАРАТНА ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ ТА АЛГОРИТМІВ ФОРМУВАННЯ ТА ОБРОБЛЕННЯ ДАНИХ В СЗК

## 3.1 Спецпроцесор виконання арифметичних операцій в СЗК

Однією із важливих переваг модулярної арифметики є мала розрядність операндів і результату операції. Ця обставина дозволяє застосовувати табличні методи, при яких бінарні операції перетворюються на однотактові, які здійснюються простою вибіркою з таблиць.

Недолік даного підходу, який перешкоджає його впровадження в практику, є істотне зростання апаратних затрат при збільшенні розрядності операндів. Отже актуальною є задача зменшення обсягу таблиць виконання арифметичних операцій.

Для реалізації арифметичного пристрою виберемо систему модулів: 3, 5, 7, 11, 13, 19, 23, 29, 31. Вибрані модулі забезпечують діапазон оброблення даних від 0 до 842 691 135.

Функціональна схема спецпроцесора виконання арифметичних операцій реалізована на ПЛІС фірми ALTERA серії MAX3000 і складається з п’яти функціональних блоків:

- два блоки SZK_MOD, в яких відбувається перевід числа в систему залишкових класів (рис. 3.1);
- блок KONTROL, який керує роботою схеми;
- блок SUM_SZK, в якому відбувається арифметична операція додавання над числами в системі залишкових класів (рис. 3.2);
- блок SUB_SZK, в якому відбувається арифметична операція віднімання над числами в системі залишкових класів (рис. 3.3);
- блок MUL_SZK, в якому відбувається арифметична операція множення над числами в системі залишкових класів (рис. 3.4).

Розробку спецпроцесора виконано з використанням САПР MAX+PLUS II, яке дозволяє на рівні часових діаграм оцінити поведінку пристрою до програмування його в ПЛІС. Часові діаграми відображають як логіку

функціонування, так і реальні часові відношення сигналів. Моделювання здійснюється з високим ступенем адекватності, що значно спрощує процес відлагодження пристрою.

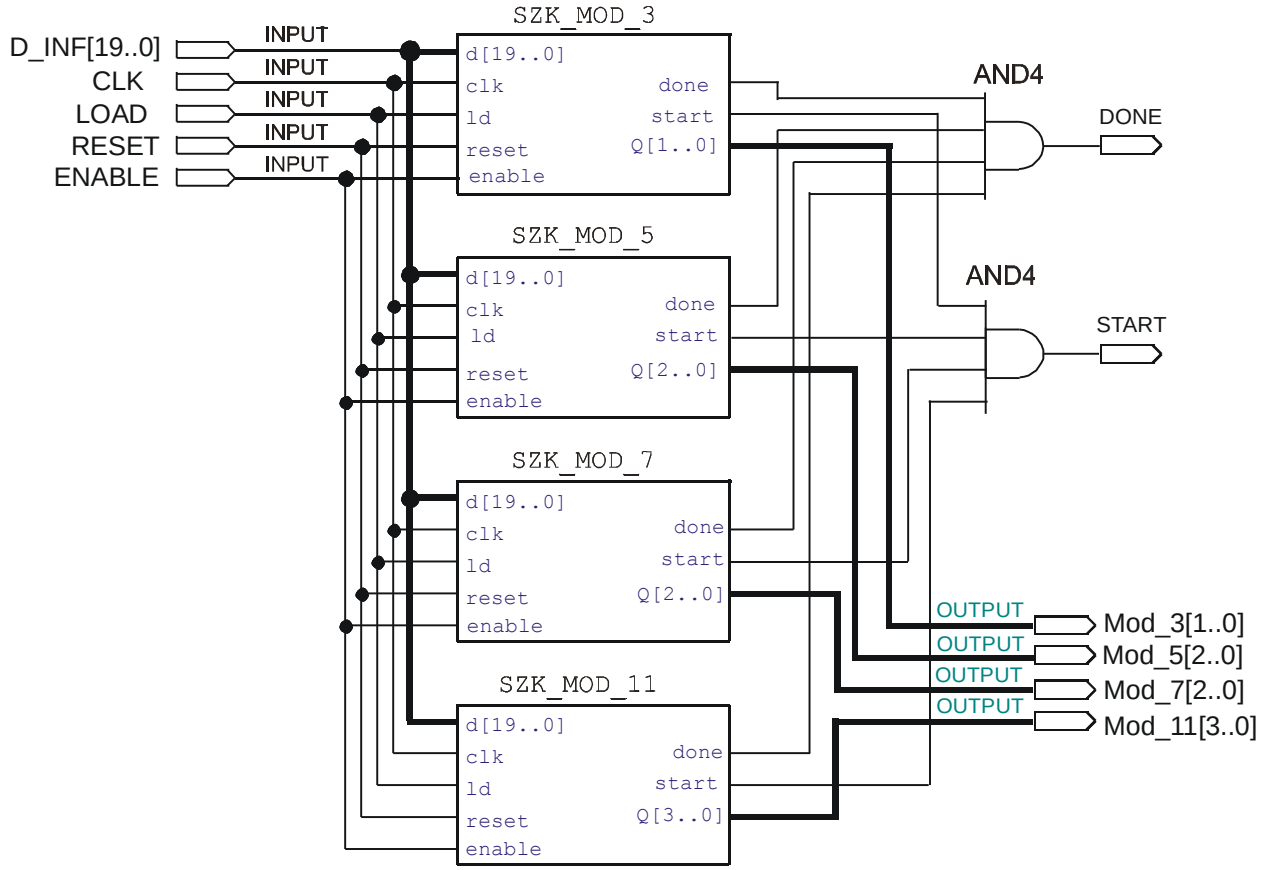


Рисунок 3.1 - Функціональна схема блоку переведення чисел в СЗК

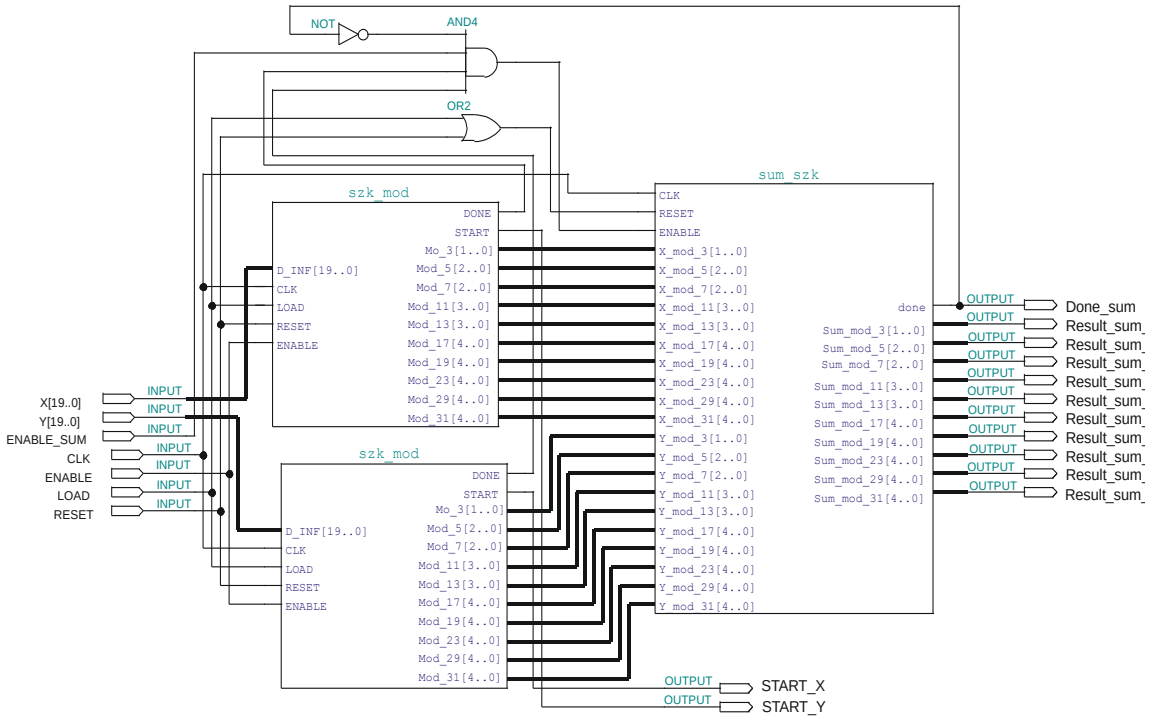


Рисунок 3.2 - Функціональна схема блоку додавання в СЗК

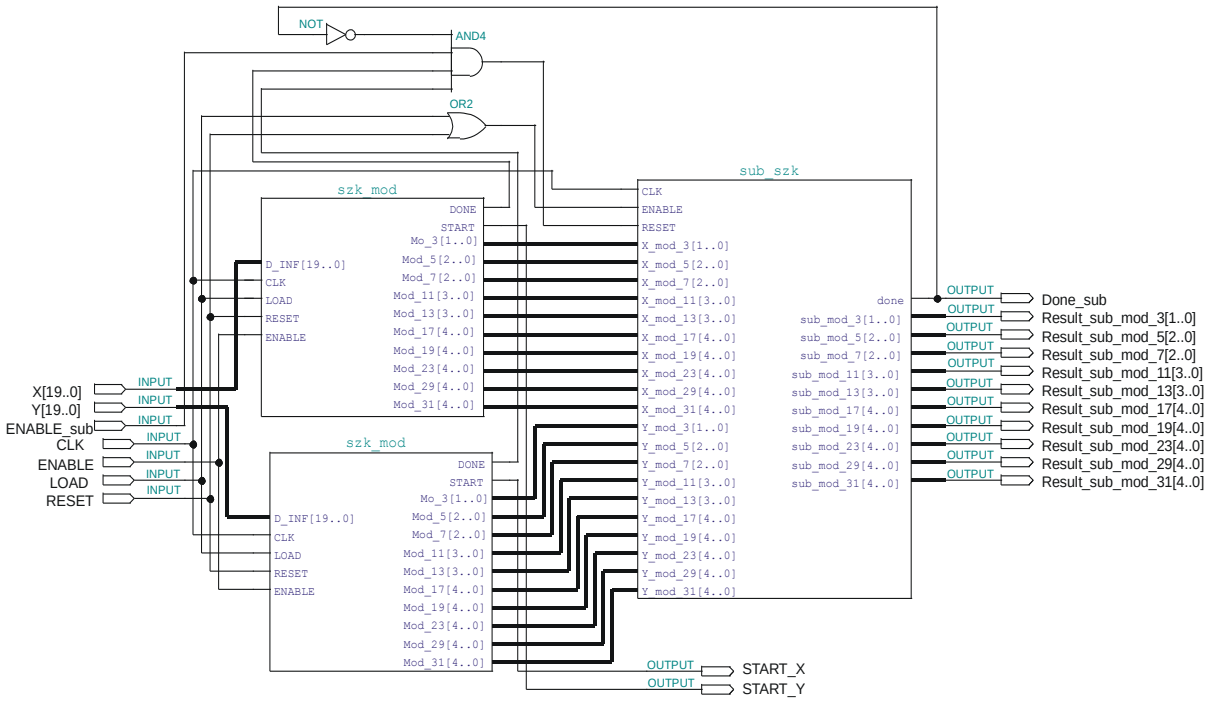


Рисунок 3.3 - Функціональна схема блоку віднімання в СЗК

Детальніше опишемо принцип роботи блоку виконання операції множення MUL_SZK (рис. 3.4). Функціональна схема блоку складається з десяти функціональних блоків, які являються блоками множення в системі залишкових класів по відповідному модулю.

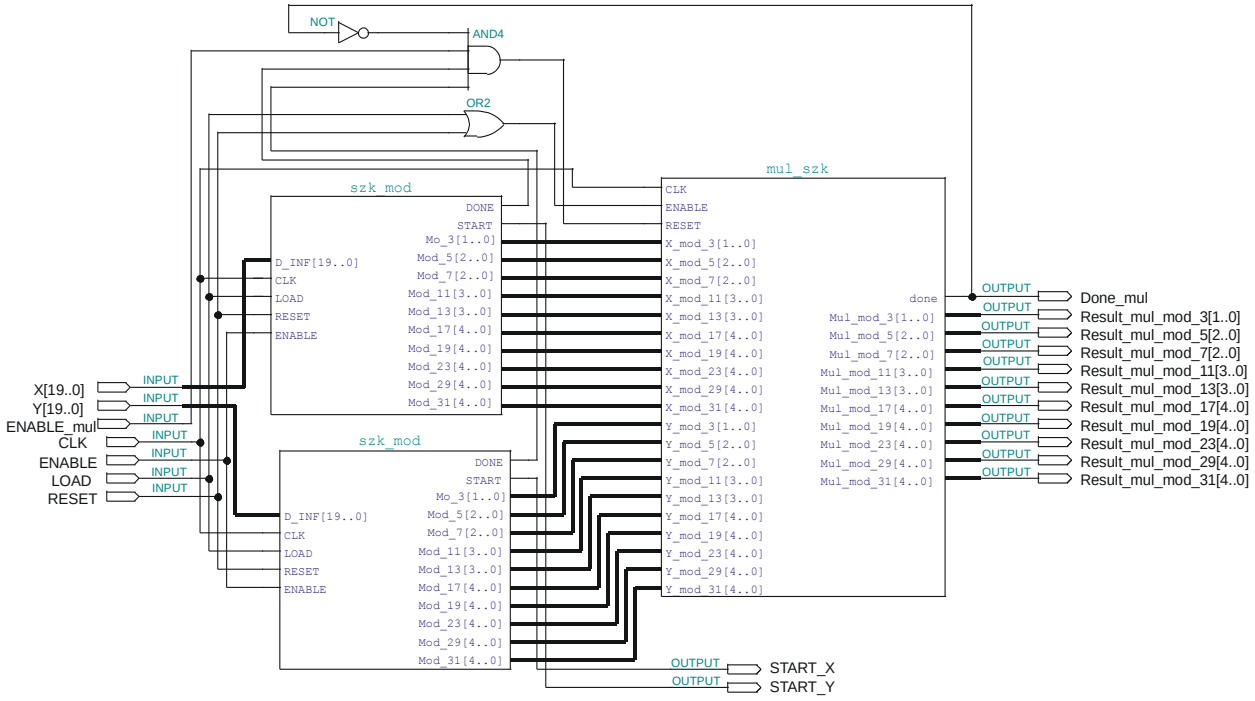


Рисунок 3.4 - Функціональна схема блоку множення в СЗК

Множення в даному модулі відбувається табличним методом. Використання табличного методу виконання арифметичних операцій додавання, віднімання і множення дає помітне збільшення швидкодії. Однак даний метод потребує значних затрат пам'яті, оскільки потрібно перерахувати всі варіанти вхідних даних.

Як видно із табл. 3.1 для модуля 11 існує 121 комбінація вхідних даних, тому актуальною задачею є зменшення розмірів таблиці 3.1 [2].

Таблиця 3.1 – Таблиця множення в СЗК по модулю 11

| | X | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Y | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| | 1 | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| | 2 | 0 | 2 | 4 | 6 | 8 | 10 | 1 | 3 | 5 | 7 | 9 |
| | 3 | 0 | 3 | 6 | 9 | 1 | 4 | 7 | 10 | 2 | 5 | 7 |
| | 4 | 0 | 4 | 8 | 1 | 5 | 9 | 2 | 6 | 10 | 3 | 7 |
| | 5 | 0 | 5 | 10 | 4 | 9 | 3 | 8 | 2 | 7 | 1 | 6 |
| | 6 | 0 | 6 | 1 | 7 | 2 | 8 | 3 | 9 | 4 | 10 | 3 |
| | 7 | 0 | 7 | 3 | 10 | 6 | 2 | 9 | 5 | 1 | 8 | 4 |
| | 8 | 0 | 8 | 5 | 2 | 10 | 7 | 4 | 1 | 9 | 6 | 3 |
| | 9 | 0 | 9 | 7 | 5 | 3 | 1 | 10 | 8 | 6 | 4 | 2 |
| | 10 | 0 | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

Як видно із таблиці 3.1 вона є симетричною відносно діагоналей, а це означає, що ми можемо її скоротити і отримувати правильні результати, тобто ми можемо перейти до наступної таблиці 3.2.

Щоб зменшити розміри таблиці на половину, нам потрібно виконати одну умову, а саме, якщо, наприклад, вхідна дана X більша за Y, то нам потрібно поміняти їх місцями і по таблиці знайти результат множення.

Подальше зменшення обсягу таблиці досягається за рахунок видалення крайнього правого стовпця (табл.3.1), так як при множенні будь-якого числа на нуль ми отримаємо нуль (таблиця 3.2).

Таблиця 3.2 – Скорочена на половину таблиця множення в СЗК по модулю 11 без нульових елементів.

| | X | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Y | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| | 0 | | | | | | | | | | | |
| | 1 | | 1 | | | | | | | | | |
| | 2 | | 2 | 4 | | | | | | | | |
| | 3 | | 3 | 6 | 9 | | | | | | | |
| | 4 | | 4 | 8 | 1 | 5 | | | | | | |
| | 5 | | 5 | 10 | 4 | 9 | 3 | | | | | |
| | 6 | | 6 | 1 | 7 | 2 | 8 | 3 | | | | |
| | 7 | | 7 | 3 | 10 | 6 | 2 | 9 | 5 | | | |
| | 8 | | 8 | 5 | 2 | 10 | 7 | 4 | 1 | 9 | | |
| | 9 | | 9 | 7 | 5 | 3 | 1 | 10 | 8 | 6 | 4 | |
| | 10 | | 10 | 9 | 8 | 7 | 6 | 5 | 4 | 3 | 2 | 1 |

Дана таблиця є симетричною відносно своєї другої діагоналі, отже можна перейти до таблиці наступного виду (табл.3.3).

Для реалізації даного методу без втрат даних необхідно виконати наступне: нехай Y рівний 9, а X – 5, їх сума рівна 14 – це на три більше ніж модуль, нам необхідно відняти від Y і X трійку, в результаті ми отримаємо такі нові значення X = 2, Y = 6. З таблиці знаходимо результат множення, він рівний 1, що є вірним.

Однак, дану таблицю можна скоротити ще наполовину (табл. 3.4).

Для правильно реалізації даної таблиці необхідно виконати наступну умову: якщо число $k$ менше за Y, то Y присвоюється величина $m - Y$,

де $k = \frac{m-1}{2}$; $m$ – модуль СЗК.

Таблиця 3.3 – 1\4 таблиці множення в СЗК по модулю 11.

| | | X | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Y | 0 | | | | | | | | | | | |
| | 1 | | 1 | | | | | | | | | |
| | 2 | | 2 | 4 | | | | | | | | |
| | 3 | | 3 | 6 | 9 | | | | | | | |
| | 4 | | 4 | 8 | 1 | 5 | | | | | | |
| | 5 | | 5 | 10 | 4 | 9 | 3 | | | | | |
| | 6 | | 6 | 1 | 7 | 2 | 8 | | | | | |
| | 7 | | 7 | 3 | 10 | 6 | | | | | | |
| | 8 | | 8 | 5 | 2 | | | | | | | |
| | 9 | | 9 | 7 | | | | | | | | |
| | 10 | | 10 | | | | | | | | | |

Таблиця 3.4 – 1\8 таблиці множення в СЗК по модулю 11.

| | | X | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Y | 0 | | | | | | | | | | | |
| | 1 | | 1 | | | | | | | | | |
| | 2 | | 2 | 4 | | | | | | | | |
| | 3 | | 3 | 6 | 9 | | | | | | | |
| | 4 | | 4 | 8 | 1 | 5 | | | | | | |
| | 5 | | 5 | 10 | 4 | 9 | 3 | | | | | |
| | 6 | | | | | | | | | | | |
| | 7 | | | | | | | | | | | |
| | 8 | | | | | | | | | | | |
| | 9 | | | | | | | | | | | |
| | 10 | | | | | | | | | | | |

Тоді по таблиці відшукуємо результат множення і віднімаємо його від модуля, в результаті чого ми отримаємо коректний добуток.

Для прикладу візьмемо наступну комбінацію вхідних даних:

X = 2, Y = 7, оскільки Y більше за $k$ (в даному випадку $k = 5$), то йому присвоюємо нове значення, а саме – 4. Отже, ми отримуємо нову комбінацію вхідних даних – X = 2, Y = 4. З таблиці ми отримуємо проміжний результат – число 8, далі, щоб отримати остаточний результат, нам необхідно від 11 відняти 8 і ми отримаємо 3. Дійсно добуток чисел 2 і 7 по модулю 11 становить 3.

Отже, використавши всі ці методи ми можемо скоротити розміри таблиць арифметичних операцій у 8 разів.

Код програми блоку множення по модулю 5, в якому використані дані методи, представлений в додатку Б.

Однак, дослідивши кількість зайнятого на ПЛМ місця, було встановлено, що перший метод (коли таблиця зменшується на половину) є ефективнішим, тому що в другому методі використовується багато логічних умов.

## 3.2 Програмна реалізація методу виявлення і виправлення помилок

Блок схема алгоритму виявлення і виправлення помилок представлена в додатку В. В програмі використовуються наступні функції:

- *int Bazus (int *moduls, int leng)* – функція пошуку базисних чисел;
- *int Find_A(int * Num,int num)* – функція пошуку десяткового числа;
- *int Find_RO (int *moduls, int num)* – функція пошуку діапазону представлення чисел.

При завантаженні програми на екран виводиться повідомлення про те, що необхідно ввести кількість модулів (не менше 3), після чого користувач повинен почергово ввести значення модулів, потім число, яке передається (залишки по кожному з модулів), а також число, яке було прийняте.

Після виконання вказаних вище дій, в програмі відбувається обчислення діапазону представлення чисел. Після того в функції *int Bazus (int *moduls, int*

*leng)* обраховуються базисні числа (для числа, яке передається), які в подальшому виводяться, для зручності слідкування за етапом виявлення і виправлення помилок на екран.

Наступними етапами роботи програми є почергове обчислення базисних чисел і десяткового числа А при вилученні одного з модулів, тобто спочатку обчислюються базисні числа, діапазон представлення чисел (інформаційний), коли вилучається перший модуль, потім – коли вилучається другий і так далі. Якщо на якомусь із даних етапів число А виходить менше за інформаційний діапазон представлення чисел, тоді по даному модулю (модулю який вилучили) відбулась помилка.

Програма є простою і зручною у користуванні і дає візуальну можливість спостерігати за алгоритмом виявлення і виправлення помилок. Лістинг програми представлений в додатку Г.

## 3.3 Реалізація алгоритму шифрування даних в СЗК

Блок схема роботи програми шифрування-дешифрування даних представлена в додатку Д. В програмі є наступні функції:

- *int Find_Mod(ul *moduls_all, ul *moduls, ul index, ul count, ul ind, ul length)* – функція пошуку взаємо простих чисел;
- *int Compare (ul *moduls, ul *work_modul, ul Num_Mod)* – функція провірки правильності вводу основ СЗК користувачем;
- *int Bazus (ul *moduls, int Num_Mod)* – функція пошуку базисних чисел;
- *int Find_RO (ul *moduls, int Num_Mod)* – функція пошуку діапазону чисел;
- *ui Find_A(int * buffer, ui *bazus, int Num_Mod)* – функція пошуку десяткового числа А;
- *int File_Open()* – функція відкриття вхідного і вихідного файлів.

Програма працює за наступним алгоритмом.

При завантаженні програми на дисплей виводиться меню користувача, за допомогою якого користувач може здійснити шифрування, дешифрування файлу або завершити роботу програми.

Коли користувач обирає пункт меню „Шифрувати дані” (додаток Е), то на екран виводиться запит на введення вхідного і вихідного файлів, тобто повне ім’я фалу, який користувач хоче закодувати (даний файл повинен існувати, в разі помилки виводиться повідомлення про те, що файл відкрити неможливо) і ім’я файлу, де будуть зберігатися закодовані дані.

Наступним кроком є виведення на екран взаємо простих чисел, серед яких користувач може вибрати декілька основ (не менше двох), на основі яких і відбуватиметься шифрування даних.

Після перевірки правильності введених користувачем чисел (вони повинні бути рівними одним з чисел, виведеними на екран) в функції *Find_RO* відбувається пошук діапазону представлення чисел, необхідного для подальшого пошуку числа А.

Після знаходження базисних чисел, в циклі відбувається зчитування одразу *k* байт з вхідного файлу, де *k* – кількість введених користувачем основ СЗК, і подальше обчислення числа А (тобто відбувається перевід чисел з СЗК в десяткову систему числення). Далі, щоб записати дане число в вихідний файл його потрібно розбити *k* раз на 100, оскільки для коректних результатів в вихідний файл потрібно вводити числа в діапазоні від 0 до 255.

Якщо користувач вибере другий пункт меню, тобто „Дешифрувати дані” (додаток Ж) то подальші дії відбуваються в наступному порядку.

Спочатку з вхідного файлу зчитується *k* байт і знаходиться число А (в даному випадку помножуємо перший зчитаний байт на 1, другий на 100 і т.д.).

Далі знаходимо залишки по введених користувачем основам і почергово виводимо їх у вихідний файл, в результаті чого отримаємо початкову інформацію.

В даному методі захист інформації відбувається за рахунок того, що людина, яка хоче перехопити і розкодувати дані повинна не лише знати кількість основ і їх значення, але й їх порядок. Якщо дана людина не знає хоча

б один з даних аспектів, то вона отримає лише набір незрозумілих символів, або неробочий файл.

Перша версія даної програми могла лише загодовувати/розкодовувати текстові файли і закодований файл був приблизно в 2 рази більший за розміром, оскільки число А записувалося у закодований файл напряму, тобто під кожен символ даного числа відводилося 1 байт.

Недоліком першої версії програми було і те, що вона не могла обробляти файли іншого формату. При спробі переробити програму для роботи з файлами з будь-яким розширення виникла проблема, яка полягала в тому, що в текстовому режимі байти, які зчитувалися або записувалися несли якесь логічне значення, тобто деякі з них трактувалися як символи керування (символ кінця файлу і т.д.), що призводило до того, що не можливо було або причитати всі байти, або записати їх. Вирішенням даної проблеми стало те, що в другій версії програми файли відкриваються не в текстовому а в двійковому режимі, при якому зчитувані/записувані байти не несуть ніякого логічно значення. Лістиг програми шифрування/дешифрування даних поданий в додатку З.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРА


1. Акушский И.Я., Юдицкий Д.И. Машинная арифметика в остаточных классах. - М.: Сов. радио. - 1968. – 460 с.

2. Торгашев В.А. Система остаточных классов и надёжность ЦВМ. – М.: Сов. радио. - 1973.

3. Вариченко Л.В., Лабунец В. Г., Раков М. А. / Абстрактные алгебраические системы и цифровая обработка сигналов – К.: Наук. думка, 1986. - 248 с.

4. Н.И.Червяков, П.А.Сахнюк, А.В.Шапошников, А.И.Макоха. Нейрокомпьютеры в остаточных классах. – М: Радиотехника, 2003. – 272 с.

5. Николайчук Я.М., Крикун З.Н., Божнев В.П., Представление измерительной информации в нормализованной системе исчисления остаточных классов. Известия ВУЗов «Нефть и газ». – 1976. – №6.

6. Яцків Н. Г., Король Р. І., Яцків В. В., Федчишин Т. Г. Спецпроцесор обробки даних на основі перетворення Крестенсона – Галуа // Вісник Технологічного університету Поділля. – 2003. –Т1, №3. – С. 105 – 108.

7. Муттер В.М. Основы помехоустойчивой телепередачи информации. – Л.: Энергоатомиздат. Ленингр. отд-ние, 1990. – 268 с.

8. Мак-Вильямс Ф. Дж., Слоэн Н. Дж. А. Теория кодов исправляющих ошибки.: Пер. с англ. – М.: Связь, 1979. –744 с., ил.

9. Блейхут Р. Теория и практика кодов контролирующих ошибки. Пер. с англ. – М.: Мир, 1986. – 576с., ил.

10. Стешенко В. Школа разработки аппаратуры цифровой обработки сигналов на ПЛИС. // Chip News. – 1999. – №8.

11. Стешенко В. ПЛИС фирмы ALTERA: проектирование устройств обработки сигналов – М.: «Додека», 2000. – 224 с.

12. ByteBlaster MV Parallel Port Download Cable, Data Sheet, Altera corporation, ver.1, April, 1998.

13. Угрюмов Е. П. Цифровая схемотехника. – СПб.: БХВ – Санкт-Петербург, 2002. – 528 с.