**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**

**КАФЕДРА** **КОМП’ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ**

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.В. Ільєнко

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 р.

На правах рукопису

УДК 004.056.5:510.22(043.3)

**МАГІСТЕРСЬКА АТЕСТАЦІЙНА РОБОТА**

**ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ**

**«МАГІСТР»**

**Тема**:Система криптографічного захисту інформації на основі ДСТУ ГОСТ 28147 на програмуючих логічних матрицях

|  |  |
| --- | --- |
| **Автор:** | Д.В. Суліма |
| **Науковий керівник:** к.т.н., доц. | О.М. Кулініч |
| **Нормоконтролер:** асист. | С.В. Єгоров |

**Київ 2020**

**НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ**

**Факультет:** Кібербезпеки, комп’ютерної та програмної інженерії

**Кафедра:** Компютеризованих систем захисту інформації

**Освітній ступінь:** Магістр

**Спеціальність:** 125 «Кібербезпека»

**Освітньо-професійна програма**: «Безпека інформаційних і комунікаційних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

\_\_\_\_\_\_\_\_\_\_\_\_ С.В. Казмірчук

«\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 р.

**ЗАВДАННЯ**

**на виконання магістерської атестаційної роботи**

**магістранта Суліму Дениса Вікторовича**

1. Тема: *Система криптографічного захисту інформації на основі ДСТУ ГОСТ 28147 на програмуючих логічних матрицях*

затверджена наказом ректора від 02.10 2019р. №2265/ст.

1. Термін виконання з 14.10.2019р. по 09.02.2020р.
2. Вихідні дані: проаналізувати існуючі системи та методики захисту інформації на програмуючих логічних матрицях; на основі аналізу виділити основні моменти розробки на ПЛМ, завдяки яким можливо покращити захист існуючих систем, виявлення їх переваг і недоліків; розробити методику, алгоритм ДСТУ ГОСТ 28147 в режимі простої заміни.
3. Зміст пояснювальної записки: аналітичний огляд літературних джерел з тематики диплому. Здійснення аналізу нормативної бази, аналізу алгоритмів використання програмуючих логічних матриць, розробка програмного забезпечення за допомогою програми MAX+PLUS II.

**КАЛЕНДАРНИЙ ПЛАН**

**виконання магістерської роботи**

|  |  |  |  |
| --- | --- | --- | --- |
| **№ п/п** | **Етапи виконання магістерської роботи** | **Термін виконання етапів** | **Примітка** |
|  | Уточнення постановки задачі | 16.10.2019 | *Виконано* |
|  | Аналіз літературних джерел | 16.10.2019- 22.10.2019 | *Виконано* |
|  | Обґрунтування вибору рішення | 22.10.2019- 24.10.2019 | *Виконано* |
|  | Збір інформації | 25.10.2019- 15.11.2019 | *Виконано* |
|  | Дослідження сучасних систем і методик аналізу та оцінки криптографічного захисту інформації | 16.11.2019- 27.11.2019 | *Виконано* |
|  | Розробка методики та збір відомостей про систему MAX+PLUS II | 28.11.2019- 03.12.2019 | *Виконано* |
|  | Розробка алгоритму ДСТУ ГОСТ 28147 в режимі простої заміни за допомогою програми MAX+PLUS II | 04.12.2019- 08.12.2019 | *Виконано* |
|  | Апробація розробки алгоритму ДСТУ ГОСТ 28147 за допомогою системи автоматизованого проектування MAX+Plus II | 09.12.2019- 12.12.2019 | *Виконано* |
|  | Перевірка на антиплагіат | 13.12.2019- 14.01.2020 | *Виконано* |
|  | Оформлення і друк пояснювальної записки | 14.01.2020- 15.01.2020 | *Виконано* |
|  | Оформлення презентації | 16.01.2020- 17.01.2020 | *Виконано* |
|  | Отримання рецензій від рецензента | 18.01.2020 | *Виконано* |
|  | Захист в ЕК | 19.01.2020 | *Виконано* |

Магістрант Д. Суліма

(підпис, дата)

Науковий керівник А. Ільєнко

(підпис, дата)

**РЕФЕРАТ**

Магістерська атестаційна робота складається зі вступу, чотирьох розділів, загальних висновків, списку використаних джерел і має 84 сторінки основного тексту, 32 рисунка, 4 сторінки додатків. Список використаних джерел містить 32 найменування і займає 3 сторінки. Загальний обсяг роботи 90 сторінок.

Метою роботи є розробка система криптографічного захисту інформації на програмуючих логічних матрицях.

В роботі був проведений загальний теоретичний аналіз системи автоматизованого проектування та вирішено задачу розробки системи криптографічного захисту інформації і оцінки ризиків інформаційної безпеки на основі програмуючих логічних матрицях до вимог ДСТУ ГОСТ 28147 та інших нормативно-правових документів.

В роботі реалізовано принципи проектування алгоритму ДСТУ ГОСТ 28147 на основі програмуючих логічних матрицях та програмне забезпечення в режимі простої заміни за допомогою програми MAX+PLUS II.

Розроблений метод та програмне забезпечення відносяться до галузі інформаційної безпеки і можуть бути використані для підвищення рівня захищеності програмуючих логічних матриць.

Можливі напрямки розвитку цієї роботи пов’язані із розширенням моделі і алгоритму програмного забезпечення захисту інформації відповідно до міжнародних стандартів.

Ключові слова: інформаційна безпека, інформаційно-комунікаційна система, загроза, програмована логічна матриця, запам’ятовуючий пристрій, програмована матрична логіка, функціональний блок.

ЗМІСТ

[ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 7](#_Toc30456868)

[ВСТУП 8](#_Toc30456869)

[РОЗДІЛ 1 10](#_Toc30456870)

[ТЕОРЕТИЧНІ ОСНОВИ БЕЗПЕКИ ІНФОРМАЦІЙНИХ РЕСУРСІВ 10](#_Toc30456871)

[1.1 Основні визначення 10](#_Toc30456872)

[1.2 Основні принципи захисту інформації 11](#_Toc30456873)

[1.3 Організація криптографічного захисту інформації 14](#_Toc30456874)

[1.3.1 Основи криптографічного захисту 14](#_Toc30456875)

[1.3.2 Криптографічні методи 17](#_Toc30456876)

[Висновки до розділу 29](#_Toc30456877)

[РОЗДІЛ 2 31](#_Toc30456878)

[СИСТЕМА ПРОЕКТУВАННЯ ПРОГРАМОВАНИХ ЛОГІЧНИХ ІНТЕГРАЛЬНИХ СХЕМ 31](#_Toc30456879)

[2.1 Проектування цифрових систем і комп’ютерів на основі програмованих логічних інтегральних схем (ПЛІС) 31](#_Toc30456880)

[2.1.1 Області застосування та класифікація ПЛІС 31](#_Toc30456881)

[2.1.2 Переваги використання ПЛІС 37](#_Toc30456882)

[2.2 Програмовані логічні матриці, програмована матрична логіка, базові матричні кристали 39](#_Toc30456883)

[2.2.1 Програмовані логічні матриці та програмована матрична логіка (ПЛМ і ПМЛ) 39](#_Toc30456884)

[2.2.2 Базові матричні кристали (вентильні матриці з програмуванням маскою) 47](#_Toc30456885)

[Висновки до розділу 51](#_Toc30456886)

[РОЗДІЛ 3 53](#_Toc30456887)

[ОПИС РЕАЛІЗАЦІЇ АЛГОРИТМУ ДСТУ ГОСТ 28147 НА ПЛІС 53](#_Toc30456888)

[3.1 Основні поняття про системи автоматизованого проектування 53](#_Toc30456889)

[3.1.1 Основні визначення та класифікація САПР 53](#_Toc30456890)

[3.1.2. Загальні відомості про систему MAX+PLUS II 56](#_Toc30456891)

[3.2 Приклад розробки алгоритму блокового шифрування на основі програмованих логічних матриць 57](#_Toc30456892)

[3.2.1. Структурна блок-схема пристрою, принципова схема суматора по модулю два 57](#_Toc30456893)

[3.2.2 Опис роботи блоку вводу ключів, блоку шифруючих перетворень, блоку шифруючи регістрів та та блоку управління 59](#_Toc30456894)

[3.2.3 Побудова S-боків підстановок та реалізація регістру зсуву 63](#_Toc30456895)

[3.3 Принципи проектування алгоритму ГОСТ 28147 на основі ПЛМ 65](#_Toc30456896)

[3.3.1 Етапи проектних процедур 65](#_Toc30456897)

[3.3.2 Етапи розробки нового проекту 70](#_Toc30456898)

[Висновки до розділу 70](#_Toc30456899)

[РОЗДІЛ 4 72](#_Toc30456900)

[РЕАЛІЗАЦІЯ АЛГОРИТМУ ДСТУ ГОСТ 28147 В РЕЖИМІ ПРОСТОЇ ЗАМІНИ ЗА ДОПОМОГОЮ ПРОГРАМИ MAX+PLUS II 72](#_Toc30456901)

[4.1 Структурна та функціональна схема ДСТУ ГОСТ 28147 в режимі простої заміни 72](#_Toc30456902)

[4.2. Програмний код пристрою та епюри функціонування 77](#_Toc30456903)

[Висновки до розділу 81](#_Toc30456904)

[ВИСНОВКИ 82](#_Toc30456905)

[СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 84](#_Toc30456906)

[Додаток А 87](#_Toc30456907)

# ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БМК – базовий матричний кристал

ВІС/НВІС – великі та надвеликі інтегральні схеми

ВК – вихідний каскад

ВО – внутрішня область

ЕВ – еквівалентний вентиль

ЗП – запам’ятовуючий пристрій

ІС – інтегральна схема

КМДН – комплементарний метал, діелектрик, напівпровідник

ПЛМ – програмована логічна матриця

ПМЛ – програмована матрична логіка

САПР – система автоматизованого проектування

ФБ – функціональний блок

ФК – функціональна комірка

ФП – функціональний перетворювач

# ВСТУП

Визначальною особливістю сучасності є потреба вирішення складних багатоаспектних завдань, що супроводжуються розширенням інформаційного обміну про найновітніші досягнення науки, сприянням впровадженню їх результатів у життя. Інформаційна епоха привела до докорінних змін у способі виконання функціональних обов’язків для численних професій. Сьогодні нетехнічний фахівець середнього рівня може виконувати роботу, яку раніше здійснював високо-кваліфікований програміст. Службовець у своєму розпорядженні ще ніколи не мав стільки точної й оперативної інформації як тепер. Водночас використання комп’ютерів і автоматизованих технологій спричиняє появу низки проблем для керівництва організацією. Доступ до величезної кількості найрізноманітніших даних надають комп’ютери, часто об’єднані в мережі. Тому, дбаючи про безпеку інформації, важливо усвідомлювати наявність ризику, зумовленого автоматизацією і наданням щораз більшого доступу до конфіденційних, персональних чи інших даних. Збільшується кількість комп’ютерних злочинів, що врешті решт може призвести до економічних втрат. Відтак очевидно, що інформація – це ресурс, який треба захищати. Те, що в 60- х роках називалося комп’ютерною безпекою, а в 70-х – безпекою даних, тепер точніше називають інформаційною безпекою. Інформаційна безпека зумовлена важливістю інформації в сучасному суспільстві, розумінням того, що інформація – це коштовний ресурс – щось більше, ніж окремі елементи даних. Інформаційною безпекою називають заходи для захисту інформації від неавторизованого доступу, руйнування, модифікації, розкриття і затримок у доступі. Інформаційна безпека гарантує досягнення таких цілей: конфіденційність критичної інформації; цілісність інформації і пов’язаних з нею процесів (створення, введення, обробки, виведення); доступність до інформації у разі потреби; облік усіх процесів, пов’язаних з інформацією. Деякі технології із захисту системи і забезпечення обліку всіх подій вбудовані в самий комп’ютер, інші – в програми. Деякі виконуються людьми і є реалізацією вказівок, що містяться у відповідних керівних документах. Використання технологій, що входять у поняття інформаційної безпеки – запорука успішної діяльності будь – якої організації і її управлінської ланки. Інформаційні процеси і діяльність, зумовлена ними, регламентуються стандартизованими нормами. Для спрощення обміну інформацією, захисту комерційної таємниці й авторських прав, статистичного аналізу, планування та ефективного керування на всіх ієрархічних рівнях глобальної інформаційної системи країни необхідне законодавче регулювання інформаційної діяльності організацій. Щодо захисту інформації, то хоча розглянуті нами засоби не завжди надійні, оскільки сьогодні швидкими темпами розвивається не тільки техніка (у нашому випадку – комп’ютерна), але й методи, що дають змогу цю інформацію здобувати, цими засобами не слід нехтувати. Нашу епоху часто називають інформаційною, і вона несе величезні можливості, пов’язані з економічним зростом, технологічними нововведеннями. Володіння електронними даними, що стають найбільшою цінністю інформаційної ери, покладає на своїх власників права й обов’язки контролю їхнього використання. Файли і повідомлення, збережені на дисках, і ті, що пересилаються по каналах зв’язку, мають іноді більшу цінність, ніж самі комп’ютери, диски. Відтак перспективи інформаційного століття можуть бути реалізовані тільки в тому випадку, якщо окремі особи, підприємства й інші підрозділи, які володіють інформацією, що дедалі частіше має конфіденційний характер чи є особливо важливою, зможуть належно захистити свою власність від будь – яких загроз, вибрати такий рівень захисту, що відповідатиме їхнім вимогам безпеки

# РОЗДІЛ 1

# ТЕОРЕТИЧНІ ОСНОВИ БЕЗПЕКИ ІНФОРМАЦІЙНИХ РЕСУРСІВ

## 1.1 Основні визначення

Інформаційна безпека – це стан захищеності потреб інформації особистістю, суспільством і державою, при якому забезпечується їхнє існування і прогресивний розвиток незалежно від наявності внутрішніх і зовнішніх інформаційних загроз. Слід відзначити, що задоволення потреб в інформації призводить до оволодіння відомостями про навколишній світ та процеси, що протікають в ньому, тобто інформованості особистості, суспільства та держави.

Інформаційна безпека – це одна із сторін розгляду інформаційних відносин у межах інформаційного законодавства з позицій захисту життєво важливих інтересів особистості, суспільства, держави та акцентування уваги на загрозах цим інтересам і на механізмах усунення або запобігання таким загрозам правовими методами [1, с. 28].

Інформаційна зброя інформаційно-комунікаційних систем та мереж (ІКСМ) – інформаційна зброя, за допомогою якої здійснюється вплив на інформацію, що зберігається, оброблюється і передається в інформаційно-комунікаційних системах та мережах і (або) порушуються інформаційні технології, що застосовуються в ІКСМ.

У складі інформаційної зброї ІКСМ можна виділити чотири основних види засобів інформаційних впливів:

- засоби порушення конфіденційності інформації;

- засоби порушення цілісності інформації;

- засоби порушення доступності інформації;

- засоби психологічного впливу на абонентів ІКСМ.

Успішне застосування інформаційної зброї забезпечення дозволяє здійснювати деструктивні впливи на інформацію, що зберігається, обробляється й передається в мережах обміну інформацією, з використанням інформаційної зброї ІКСМ.

Інформаційна боротьба – це боротьба з використанням спеціальних способів і засобів для впливу на інформаційну сферу (середовище) конфронтуючої сторони, а також для захисту власної інформаційної сфери в інтересах досягнення поставленої мети.

Інформаційна боротьба може бути як самостійним видом, так і складовою частиною будь-якого іншого різновиду боротьби (збройної, ідеологічної, економічної і т. ін.). Вона ведеться постійно, як у мирний, так і у воєнний час. Масштаби інформаційної боротьби настільки великі, що її підготовка і ведення повинні носити плановий, систематичний характер, заснований на глибоких знаннях законів і закономірностей інформаційної боротьби [2, с 112].

Оскільки основними завданнями інформаційної боротьби є ураження об’єктів інформаційного середовища противника та захист власної інформації, то структура теорії інформаційної боротьби повинна включати загальні основи теорії інформаційної боротьби, теорію ураження інформації і теорію захисту інформації.

## 1.2 Основні принципи захисту інформації

Захист інформації від НСД є складовою частиною загальної проблеми забезпечення захисту інформації в ІКСМ.

В загальному випадку комплекс програмно-технічних засобів та організаційних рішень по захисту інформації в ІКСМ реалізується в рамках системи захисту інформації від НСД, яка умовно складається з таких чотирьох підсистем [3, c. 24]:

- управління доступом до ІКСМ, до її послуг та ресурсів;

- реєстрація і облік користувачів, послуг, інформаційних ресурсів;

- криптографічного захисту;

- забезпечення цілісності інформаційних потоків, інформаційних ресурсів та програмного забезпечення.

Закриття каналів несанкціонованого отримання інформації повинно починатися з контролю доступу користувачів до ресурсів ІКСМ. Ця задача вирішується на основі ряду принципів:

Принцип виправданості доступу. Користувач повинен мати достатню «форму допуску» для отримання інформації того рівня конфіденційності, що він вимагає, і ця інформація дійсно необхідна йому для виконання його виробничих функцій.

Принцип достатньої глибини контролю доступу. Засоби захисту інформації повинні включати механізми контролю доступу до всіх видів інформаційних і програмних ресурсів ІКСМ, які у відповідності з принципом виправданості доступу слід розмежовувати між користувачами.

Принцип розмежування інформаційних потоків. Для попередження порушення інформаційної безпеки, яке, наприклад, може мати місце при запису секретної інформації на несекретні носії і в несекретні файли, її передачі програмам і процесам, які не призначені для обробки секретної інформації, а також при передачі секретної інформації по незахищених каналах зв’язку, необхідно здійснювати відповідне розмежування інформаційних потоків.

Принцип персональної відповідальності. Кожний користувач повинен нести персональну відповідальність за свою діяльність в системі, включаючи будь-які операції з конфіденційною інформацією і можливі порушення її захисту.

Принцип цілісності засобів захисту. Даний принцип передбачає, що засоби захисту інформації в ІКСМ повинні чітко виконувати свої функції у відповідності з переліченими принципами і бути ізольованими від користувачів, а для свого супроводу повинні включати спеціальний захищений інтерфейс для засобів контролю, сигналізації про спроби порушення захисту інформації і впливу на процеси в системі.

Реалізація перелічених принципів здійснюється за допомогою так званого «монітору звернень», який контролює будь-які запити до даних чи програм з боку користувачів (чи їх програм) за установленими для них видами доступу до цих даних і програм.

При проектуванні ІКСМ на її базі проводяться розробка і реалізація функціональних задач по розмежуванню і контролю доступу до апаратури і інформації як в рамках інформаційної системи в цілому, так і до відокремлених інформаційних ресурсів.



Рис. 1.1 – Структура монітору звернень

## 1.3 Організація криптографічного захисту інформації

### 1.3.1 Основи криптографічного захисту

«Криптографія» – грецьке слово, що походить від слів kryptos (таємний, схований) та graphy (запис) і містить методи та засоби забезпечення перетворення даних з метою маскування (шифрування) змісту інформації для гарантування конфіденційності та цілісності, а «криптоаналіз» орієнтований на зламування шифротекстів (шифрів). Шифротекст – це дані, які представлені у зашифрованій формі та мають прихований семантичний зміст, який утворюються після шифрування (криптографічного перетворення) відкритого тексту (з неприхованим семантичним змістом) [4, c. 56].

Варто зазначити, що грецький письменник і історик Полібій винайшов так званий полібіанський квадрат (II ст. до н. е.) розміром 5×5, заповнений алфавітом у випадковому порядку. Для шифрування на квадраті знаходили букву тексту і вставляли в шифротекст нижню від неї у тій самій колонці. Якщо буква була в нижньому рядку, то брали, верхню з тієї самої колонки. Наведемо спрощений приклад такого квадрату (рис. 1.2), для якого відкритому тексту ДЕРЖАВНИЙ буде відповідати шифротекст РГНКЖИЙГБ.

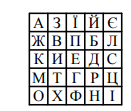


Рис. 1.2 – Приклад зашифрованого тексту полібіанським квадратом

Середньовічні вчені часто використовували магічний квадрат у вигляді рівносторонньої таблиці з вписаними в їх комірки послідовними натуральними числами від 1, що дають у сумі в кожній колонці, кожному рядку і кожній діагоналі те саме число. В такі квадрати вписувався відкритий текст відповідно до наведеної в них нумерації. Якщо потім виписати вміст таблиці колонками, то виходить шифротекст утворений перестановкою букв. Вважалося, що такі створені шифротексти охороняє не тільки ключ, але і магічна сила. На рис.1.3 наведено приклад магічного квадрата і його шифротексту.

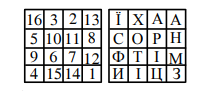


Рис. 1.3 – Приклад шифрування за допомогою магічного квадрата

Отриманий з відкритого тексту ЗАХИСТ ІНФОРМАЦІЇ, шифротекст ЇСФИХОТІАРІЦАНМЗ має досить вагомий вигляд. На перший погляд здається, начебто магічних квадратів дуже мало, але їх число дуже швидко зростає зі збільшенням розміру квадрату. Так, існує лише один магічний квадрат розміром 3×3 (не враховуючи його повороти), квадратів 4×4 уже 880, а 5×5 близько 250000, тому їх великі розміри стали вагомою основою для надійної системи шифрування того часу, оскільки ручний перебір всіх варіантів ключа для цього шифру був немислимий. Одна з відомих фігур ренесансної архітектури пов’язана з ім'ям Леона Батисту Альберта, який винайшов шифр поліалфавітної заміни, що є стійким до розкриття і докладно описав пристрої з коліс, що прокручуються, для його реалізації. Цей шифр можна описати спеціальною таблицею шифрування (рис. 1.4), яку іноді називають таблицею (квадратом) Віженера, названу ім’ям дипломата Блеза Віженера, який у XVI ст. розвивав і удосконалював криптографічні системи.

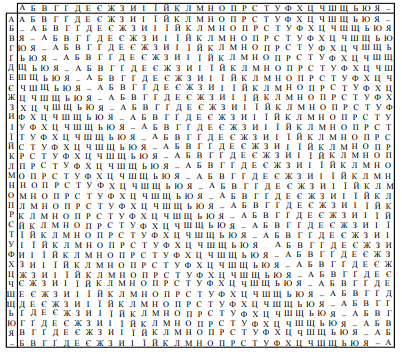


Рис. 1.4 – Приклад квадрату Віженера

Кожен рядок у цій таблиці відповідає одному типовому шифру Цезаря для алфавіту з проміжком. Повідомлення, що шифрується, виписують у рядок, а під ним розміщують ключ. Якщо він виявився коротший за відкритий текст, то його циклічно повторюють. Шифротекст формується шляхом знаходження символа у колонці таблиці за буквою тексту і рядка, що відповідає букві ключа.

Історики вважають Йогана Тритемія другим батьком сучасної криптографії, який у роботі «Стеганографія» першим систематично описав застосування таблиць шифрування, які заповнені алфавітом у випадковому порядку, а також зазначив, що можна шифрувати дві букви за один раз. Такі шифри, на відміну від раніше описаних (монограмних), були названі біграмними підстановочними шифрами і різко підсилили їхню стійкість до розкриття. Найбільш відомий шифр біграмами називається Play-fair (Чесна гра), винайдений у 1854 р. Він застосовувався Великобританією в Першу світову війну.

У XIX ст. почалася автоматизація процесу шифрування, наприклад, держсекретар Томас Джефферсон (згодом президент США) у 1790 р. винайшов шифруюче колесо, що реалізує поліалфавітну заміну тексту повідомлення за ключем, довжина періоду якого визначалася найменшим загальним кратним періодів обертів шифруючих коліс, наприклад, чотирьом колесам з періодом обертів 13, 15, 17 і 19 відповідала довжина періоду ключа 62985. Відомі також циліндр Базери (запропонований Етьєном Базери в 1891 р.), криптографічна машина (1917 р.) Жильбера Вернама та роторна машина Едварда Хеберна (1917 р.), яку згодом назвали Енігмою. У 1942 р. Аланом Тьюрингом була створена швидкодіюча ЕОМ під назвою «Колосс», спеціалізована для злому шифрів Енігми.

Завершує історію цього етапу криптографії математик Клод Елвуд Шеннон, який під час Другої світової війни, досліджуючи математичними методами надійність шифрування, включив криптологію в загальну теорію інформації.

### 1.3.2 Криптографічні методи

Суть криптографії – у забезпеченні конфіденційності даних (відкритого тексту, секретного ключа) шляхом їх шифрування [5. c. 41].

Крім забезпечення конфіденційності, криптографія застосовується для розв’язання таких задач, як:

- перевірка справжності (ідентифікація). Одержувач може встановити відправника, а зловмисник не може під нього маскуватися;

- цілісність. Отримувач може перевірити несанкціоновану модифікацію в тексті, а зловмисник не може видати підроблений текст за справжній;

- не заперечення авторства. Відправник не може в подальшому заперечувати відсилку даних.

Початкове повідомлення називають відкритим текстом P (plaintext), а процес приховування його семантичного змісту називають шифруванням і позначають E (encipher). У результаті шифрування утворюється шифротекст С (ciphertext), а одержання початкового відкритого тексту здійснюється в процесі розшифрування D (decipher) шифротексту (криптограми). Ця послідовність показана на рис.1.5. В математичній формі функція шифрування і розшифрування відповідно записується як E(P)=C i D(C)=P, а оскільки D(C) є відновлення P, то справедливо D(E(P))=P.

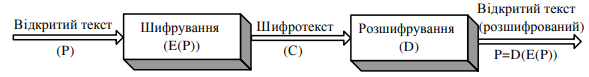


Рис. 1.5 – Шифрування і розшифрування

Сучасна криптографія містить у собі чотири базові розділи:

- симетричні криптосистеми (системи з секретним ключем або одноключові);

- асиметричні криптографічні системи (системи з відкритим ключем або двоключові);

- криптографічні протоколи;

- керування ключами.

Криптографічний алгоритм, також названий шифром – це математична функція, яка використовується для шифрування і дешифрування.

Сучасна криптографія за допомогою ключа K (key), що використовується в функціях E і D, може приймати будь-яке значення і бути обраним з великої множини, яку називають простором ключів. Ключовий простір розглядаються як потужність множини ключів, які використовують для даного шифру. Тоді відносно K функції E і D приймають вигляд Аk(P)=C, Dk(C)=P, при цьому справедлива рівність Dk(Ek(P))=P (рис. 1.6).

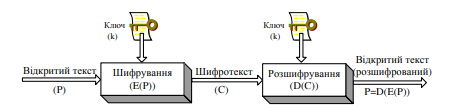


Рис. 1.6 – Шифрування і розшифрування за допомогою ключа

Для деяких алгоритмів при шифруванні і розшифруванні використовуються різні ключі K1 − для шифрування, K2 − для розшифрування, то у цьому випадку 1 Аk (P) = C, 2 Dk (C)=P, 2 Dk ( 1 Ek (P))=P (рис. 1.7).

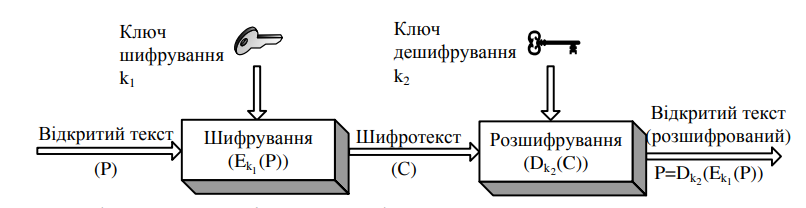


Рис. 1.7 – Шифрування і розшифрування за допомогою двох різних ключів

Безпека цих алгоритмів цілком заснована на ключах, а не на деталях алгоритмів. Це значить, що алгоритм може бути опублікований і проаналізований. Продукти, що використовують цей алгоритм, можуть широко тиражуватися. Не має значення, що зловмиснику відомий алгоритм, якщо йому не відомий конкретний ключ, то він не зможе прочитати шифротекст.

У криптографії існують дві базові методології з використанням ключів – симетрична (рис. 1.6) і асиметрична (рис. 1.7) і які відповідно називають із секретним ключем і з відкритим ключем. Кожна методологія використовує свої власні процедури, свої способи розподілу ключів, їх типи, алгоритми шифрування і розшифрування ключів.

Сучасні зміни полягають у тому, що алгоритми стали працювати з бітами, а не символами, що важливо хоча б з погляду розміру алфавіту (перехід з 26 елементів до двох). Більшість криптографічних алгоритмів комбінують підстановки і перестановки [5. c. 87].

Підстановочним називають такий шифр, у якому кожний символ відкритого тексту в шифротексту замінюється іншим символом. Одержувач інвертує підстановку шифротексту, відновлюючи відкритий текст. У криптографії використовують чотири типи підстановочних шифрів (рис. 1.8).

Знаменитий шифр Цезаря є простим підстановочним шифром і дійсно дуже простим, тому, що алфавіт шифротексту є зміщеним, а не випадково розподіленим. Прості підстановочні шифри легко розкриваються, оскільки шифр не ховає частоти використання різних символів у відкритому тексті і для відновлення, наприклад, англійського шифротексту, криптоаналітику необхідні тільки частоти використання 26 англійських символів.

Однозвучні підстановочні шифри використовувалися вже у 1401 р. в герцогстві Мантуа. Вони більш складні для розкриття, хоча і не ховають усіх статистичних властивостей мови відкритого тексту. На основі розкриття з відомим відкритим текстом ці шифри зламуються елементарно, а на основі тільки шифротексту − складніше, але використовуючи комп’ютер, це займе декілька секунд.



Рис. 1.8 – Класифікація підстановочних шифрів

Полігамні підстановочні шифри кодують відразу групи символів. Частинним випадком полігамного шифру є біграмний Play-fair.

Поліалфавітні підстановочні шифри були винайдені Ліном Баттістой (Lean Battista) у 1568 р. Вони використовувалися армією Сполучених Штатів у ході Громадянської війни в Америці. Шифр (квадрат) Віженера (Vigenere), який вперше опублікований у 1586 р. і шифр Бофора (Beaufort) також є прикладами поліалфавітних підстановочних шифрів.

У поліалфавітних підстановочних шифрах використовується множина однолітерних ключів для шифрування тільки одного символу відкритого тексту. Першим ключем шифрується перший символ відкритого тексту, другим − другий символ, і так далі. Після використання усіх ключів вони повторюються циклічно. Якщо застосовується 20 однолітерних ключів, то кожна двадцята літера шифрується тим же ключем. Цей параметр називається періодом шифру. Шифри з довгим періодом складніше розкрити, чим з коротким, але використання обчислювальної техніки значно спростило цю задачу.

Шифр зі змінним ключем (іноді названий книжковим шифром), що використовує один текст для шифрування іншого тексту і хоча період ключа дорівнює довжині тексту, він також може бути легко зламаний.

У перестановочному шифрі міняється не відкритий текст, а порядок символів. У простому колонковому перестановочному шифрі відкритий текст пишеться горизонтально на розграфленому листі фіксованої ширини, а шифротекст зчитується вертикально. Дешифрування є записом шифротексту вертикально на розграфленому листі фіксованої ширини і потім зчитування відкритого тексту горизонтально. Символи шифротексту ті ж самі, що й у відкритому тексті, і мають приблизно ті самі частотні характеристики, що і звичайно.

Відповідно до базових методологій використання ключів є два відповідних базових типів алгоритмів: симетричні і асиметричні.

Симетричні алгоритми, іноді називаються умовними алгоритмами, у яких ключ шифрування може бути розрахований за ключем дешифрування і навпаки. У більшості симетричних алгоритмів ключі шифрування і дешифрування ті самі. Ці алгоритми також називають алгоритмами із секретним ключем (передається секретним каналом) чи алгоритмами з одним ключем, вони вимагають, щоб відправник і одержувач погодили використовуваний ключ перед початком безпечної передачі повідомлень. Безпека симетричного алгоритму визначається ключем, знання якого дозволяє розшифрувати повідомлення. Тому, поки повідомлення є секретними, ключ повинен триматись у таємниці.

Симетричні алгоритми поділяються на дві категорії:

1) Алгоритми, що обробляють відкритий текст побітово (іноді побайтово) називаються потоковими чи алгоритмами з потоковими шифрами;

2) Алгоритми, що працюють з групами бітів, які складають блоки (наприклад, 64 біт) називаються блоковими чи алгоритмами з блоковими шифрами. До появи комп’ютерів алгоритми зазвичай обробляли відкритий текст посимвольно, що може розглядатися як потокове шифрування символів.

Алгоритми з відкритим ключем (асиметричні алгоритми) розроблені таким чином, що ключ, використовуваний для шифрування, відрізняється від ключа дешифрування. Більше того, ключ дешифрування не може бути (принаймні протягом певного інтервалу часу) розрахований за ключем шифрування. Алгоритми називаються «з відкритим ключем» тому, що ключ шифрування може бути відкритим і хто завгодно може використовувати його (отримуючи не секретним каналом) для шифрування відкритого тексту, але тільки власник відповідного ключа дешифрування може розшифрувати повідомлення. У цих системах ключ шифрування часто називається відкритим, а розшифрування − закритим.

Безпека алгоритмів. Різні алгоритми надають різні ступені безпеки залежно від того, наскільки важко зламати алгоритм. Якщо вартість злому алгоритму вище, ніж вартість зашифрованих даних, то ці дані швидше за все у безпеці. Якщо час злому алгоритму більше, ніж час, протягом якого зашифровані дані повинні зберігатися в секреті, то також швидше за все інформація у безпеці. Якщо обсяг даних, зашифрований одним ключем, менше, ніж обсяг даних, необхідний для злому алгоритму, і тоді, швидше за все, інформація у безпеці.

Криптографія більше цікавиться криптосистемами, які важко зламати обчислювальним способом.

Криптосистема це алгоритм плюс усі можливі відкриті тексти, шифротексти і ключі. Алгоритм вважається обчислювально безпечним (чи, як іноді називають, криптостійкими), якщо він не може бути зламаний з використанням доступних ресурсів зараз чи у майбутньому. Термін «доступні ресурси» є досить розпливчастим. Складність розкриття можна визначити різними способами (рис. 1.9).

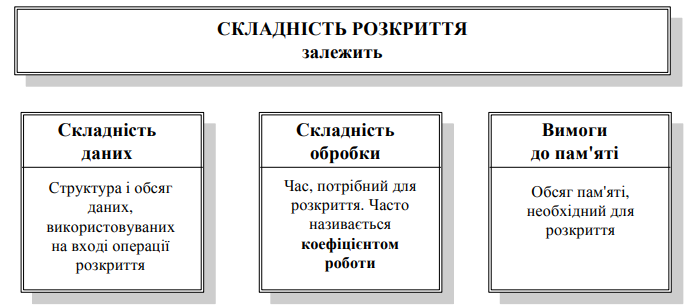


Рис. 1.9 – Характеристики складності розкриття даних

Складність розкриття визначається за максимальним з цих трьох коефіцієнтів. Ряд операцій розкриття припускають взаємозв’язок коефіцієнтів, наприклад, більш швидке розкриття можливе за рахунок збільшення вимог до пам’яті.

Складність виражається порядком величини. Якщо складність обробки для даного алгоритму складає , то операцій потрібно для розкриття алгоритму (ці операції можуть бути складними і тривалими). Так, якщо передбачається, що обчислювальні потужності здатні виконувати мільйон операцій за секунду, і для рішення задачі використовується мільйон паралельних процесорів, одержання ключа займе у понад 1019 років. У той час, як складність розкриття залишається постійною (поки будь-який криптоаналітик не придумає кращого способу розкриття), потужність комп’ютерів зростає. За останні піввіку обчислювальні потужності феноменально виросли і немає ніяких причин підозрювати, що ця тенденція не буде продовжена. Багато криптографічних зломів придатні для паралельних комп’ютерів. Задача розбивається на мільярди маленьких шматочків, рішення яких не вимагає міжпроцесорної взаємодії. Оголошення алгоритму безпечним просто тому, що його нелегко зламати, використовуючи сучасну техніку, у кращому випадку ненадійно. Якісні криптосистеми проектуються стійкими до злому з урахуванням розвитку обчислювальних засобів на багато років вперед.

Передача інформації з використанням симетричної криптографії.

У якісній криптосистемі безпека цілком залежить від знання ключа й абсолютно не залежить від знання алгоритму і тому керування ключами так важливо в криптографії. Ключ повинен залишатися в секреті перед, після і протягом роботи протоколу – доти, поки повинен залишатися в таємниці переданий шифротекст, у противному випадку він відразу буде розкритий [6].

Симетричні криптосистеми мають такі недоліки:

- розподіл ключів повинен проводитися в секреті. Ключі настільки ж важливі, як і всі шифротексти, зашифровані цими ключами, тому що знання ключа дозволяє розкрити криптограми. Розподіл ключів є дуже важливою задачею і часто кур’єри особисто доставляють ключі за призначенням.

- якщо ключ скомпрометований (украдений, розгаданий, випитаний, отриманий за хабар і т. ін.), то можна розшифрувати всі криптограми, зашифровані цим ключем.

- якщо кожна пара користувачів мережі використовує окремий ключ, то загальна їх кількість швидко росте із зростанням числа користувачів. Розв’язання цієї проблеми можливе шляхом зменшення кількості користувачів, але це не завжди можливо.

Поняття односпрямованої функції є центральним у криптографії з відкритими ключами. Односпрямовані функції відносно легко обчислюються, але інвертуються досить складно. Хорошим прикладом такої функції може бути розбита на тисячу малих шматочків ваза, яку нелегко знову скласти з цих шматочків.

Строгого математичного доказу існування односпрямованих функцій немає, як немає і реальних свідчень можливості їхньої побудови. Незважаючи на це, багато функцій виглядають у точності як односпрямовані і їх можна розрахувати, але досі, не має простого способу інвертувати їх. Наприклад, в обмеженій області легко обчислити , але набагато складніше .

У односпрямованої хеш-функції може бути низка імен: функція стискання, функція згортання, дайджест повідомлення, цифровий відбиток, характерна ознака, криптографічна контрольна сума, код цілісності повідомлення, код виявлення маніпуляцій та ін. Але як би не називалась ця функція, вона є центральною в сучасній криптографії і особливо важливе значення має при реалізації багатьох протоколів.

Хеш-функції, які довгий час використовувалися в комп’ютерних науках, є математичними чи іншими функціями, які одержують на вході рядок змінної довжини (прообраз) і перетворюють його в рядок фіксованої, зазвичай меншої, довжини (значення хеш-функції, згортка). Приклад такого згортання може бути реалізований за допомогою бінарної функції XOR шляхом одержання прообразу і повертання байта, утвореного виконанням операції побітового складання за модулем 2 (⊕) всіх бітів вхідних байтів, де

0⊕0=0, 0⊕1=1, 1⊕0=1, 1⊕1=0.

Суть хеш-функції полягає в одержанні цифрового відбитку прообразу, тобто утворення значення, за яким можна встановити ідентичність отриманого прообразу і дійсного. Оскільки хеш-функція є співвідношенням «багато до одного», то неможливо з усією визначеністю сказати, що два цифрові відбитки різних документів не збігаються, але хеш-функцію можна використовувати для оцінки з прийнятною точністю.

Односпрямована хеш-функція – це хеш-функція, що працює тільки в одному напрямку, тобто легко обчислити її значення за прообразом, але важко створити прообраз, значення хеш-функції якого дорівнює заданій величині. Зазначимо, що якісні односпрямовані хеш-функції – це хеш-функція без колізій (зіткнень), тобто достатньо важко створити два прообрази з однаковим значенням хеш-функції.

Хеш-функція є відкритою і секрету її розрахунку не існує. Безпека односпрямованої хеш-функції полягає саме в її односпрямованості. Зміна одного біта прообразу приводить до зміни, у середньому, половини бітів значення хеш-функції. Обчислювально неможливо знайти прообраз, що відповідає заданому значенню хеш-функції.

Організація зв’язку з використанням асиметричної криптографії.

У 1976 р. Уітфілд Діффі і Мартін Хеллман змінили парадигму криптографії (яка засновувалась на одноключових системах), описавши асиметричну криптографію, що базується на двох різних ключах − відкритому і закритому. Визначення закритого ключа за відкритим вимагає величезних обчислювальних ресурсів. Хто завгодно, використовуючи відкритий ключ, може зашифрувати повідомлення, але не розшифрувати його. Розшифрувати шифротекст може тільки власник закритого ключа. Це схоже на поштову скриньку, де процес шифрування з відкритим ключем можна порівняти з опусканням у її проріз листа, що може зробити кожний. Розшифрування закритим ключем нагадує витягування листа з поштової скриньки, що набагато складніше і може знадобитися зварювальний агрегат, але якщо знати секрет (код замка від поштової скриньки), можна легко дістати лист.

Математичною основою процесу є односпрямовані хеш-функції з секретом, за допомогою яких легко реалізується шифрування в прямому напрямку, а дешифрування − в зворотному і воно настільки важке, що, не знаючи секрету, навіть на комп’ютерах Cray за тисячі (і мільйони) років неможливо розшифрувати криптограму. Секретом (люком) служить закритий ключ, який робить розшифрування таке ж простим, як і шифрування.

Слід зазначити, що криптографія з відкритими ключами усуває проблему розподілу ключів, властиву симетричним криптосистемам.

Зазвичай ціла мережа користувачів приймає для використання певну асиметричну криптосистему. У кожного з них є відкритий і закритий ключ, при цьому відкриті ключі містяться в загальнодоступній базі даних.

Ось як на основі цієї криптографії відправник може переправити повідомлення одержувачу:

1) Відправник і одержувач домовляються використовувати асиметричну криптосистему.

2) Одержувач посилає відправнику свій відкритий ключ.

3) Відправник шифрує своє повідомлення, використовуючи відкритий ключ одержувача і відсилає його.

4) Одержувач приймає і розшифровує повідомлення своїм секретним ключем.

Змішані (гібридні) криптосистеми. Алгоритми з відкритими ключами не заміняють симетричні алгоритми і використовуються не для шифрування повідомлень, а для шифрування ключів за такими двома причинами:

- алгоритми з відкритими ключами працюють повільно. Симетричні алгоритми принаймні в 1000 разів швидше, ніж алгоритми з відкритими ключами. Так, комп'ютери стають усе швидше і швидше, і років через 15 криптографія з відкритими ключами досягне швидкостей, порівнянних із сьогоднішньою швидкістю симетричної криптографії. Але вимоги до обсягу переданої інформації також зростають, і завжди буде вимагатися шифрувати дані швидше, ніж це зможе зробити криптографія з відкритими ключами.

- Криптосистеми з відкритими ключами уразливі стосовно розкриття на основі підібраного відкритого тексту. Криптоаналітик не зможе розкрити k, але зможе визначити Р.

У більшості реалізацій криптографія з відкритими ключами використовується для засекречування і розповсюдження сеансових ключів, що використовуються симетричними алгоритмами для закриття потоку повідомлень. Іноді такі реалізації називаються змішаними (гібридними) криптосистемами.

Використання криптографії з відкритим ключем для розподілу секретних ключів вирішує дуже важливу проблему їх розподілу.

## Висновки до розділу

Отже, проаналізувавши роботи провідних вчених та закон України можна чітко визначити, що інформаційна безпека – стан захищеності особи, суспільства і держави, при якому досягається інформаційний розвиток (технічний, інтелектуальний, соціально-політичний, морально-етичний), за якого сторонні інформаційні впливи не завдають їм суттєвої шкоди.

Інформаційна безпека держави – це стан її захищеності та інформаційний розвиток, при якому акції інформаційного впливу, спеціальні інформаційні операції, інформаційний тероризм, незаконне зняття інформації за допомогою спеціальних технічних засобів та комп’ютерна злочинність не завдають особливої шкоди національним інтересам.

Інформаційна безпека є невід’ємною складовою національної безпеки, оскільки впливає на різні сфери життєдіяльності особи, суспільства та держави. Також інформаційна безпека повинна забезпечувати основні властивості інформації: конфіденційність – стан інформації, при якому доступ до неї здійснюють тільки суб'єкти, що мають на нього право; цілісність – уникнення несанкціонованої модифікації інформації; доступність – уникнення тимчасового або постійного заховання інформації від користувачів, що отримали права доступу.

Проаналізувавши загрози в інформаційній безпеці можна чітко сказати, що на сьогоднішній день актуальною проблемою є вплив ЗМІ на суспільство. Використовуючи різні засоби масової інформації, а також мережу інтернет відбуваються постійні інформаційні атаки, до яких сучасне суспільство практично не підготовлене. Україна зараз страждає від інформаційної війни. Вона використовує і управляє інформацією з метою набуття конкурентоздатної переваги над противником. Сьогодні інформація набирає матеріальної форми і володіння нею стає дуже жаданим. Відповідно і забезпечувати інформаційну безпеку стає все важче.

# РОЗДІЛ 2

# СИСТЕМА ПРОЕКТУВАННЯ ПРОГРАМОВАНИХ ЛОГІЧНИХ ІНТЕГРАЛЬНИХ СХЕМ

## 2.1 Проектування цифрових систем і комп’ютерів на основі програмованих логічних інтегральних схем (ПЛІС)

### 2.1.1 Області застосування та класифікація ПЛІС

Для побудови логічних керуючих автоматів з паралельною архітектурою використовують пристрої з регулярною архітектурою, які представляють собою набір стандартних елементів та вузлів, що інтегровані на одному кристалі. Існує багато різновидів таких пристроїв, але всі вони об’єднуються однією назвою – програмовані логічні інтегральні схеми [7, c. 85].

Настроювання ПЛІС на виконання заданих функцій здійснюються користувачем шляхом вільного компонування цих елементів і вузлів програмним способом. Завдяки зазначеним особливостям ПЛІС в останні роки успішно завойовують ринок і в ряді предметних областей становлять гідну конкуренцію мікропроцесорам і мікроконтролерам. Залежно від використовуваної архітектури ПЛІС можуть бути класифіковані таким чином (рис. 2.1)

Структури ПЛІС мають високий рівень регулярності: основу кристала ПЛІС становить матриця однотипних функціональних вузлів, на базі яких користувач може створювати цілі системи керування складними технологічними об'єктами. Завдяки цьому ПЛІС характеризуються високою швидкодією і надійністю, а також широкими можливостями в частині резервування і діагностики [7, c. 97].

Найбільш ефективними областями застосування ПЛІС є:

- синтез (моделювання) нетрадиційних архітектур керуючих автоматів (навіть якщо надалі планується використання іншої елементної бази);

- побудова надшвидкодіючих керуючих пристроїв;

- реалізація автономних систем керування критичного застосування, до яких пред'являються жорсткі вимоги в частині забезпечення надійності, безпеки, електромагнітної сумісності й т.ін.

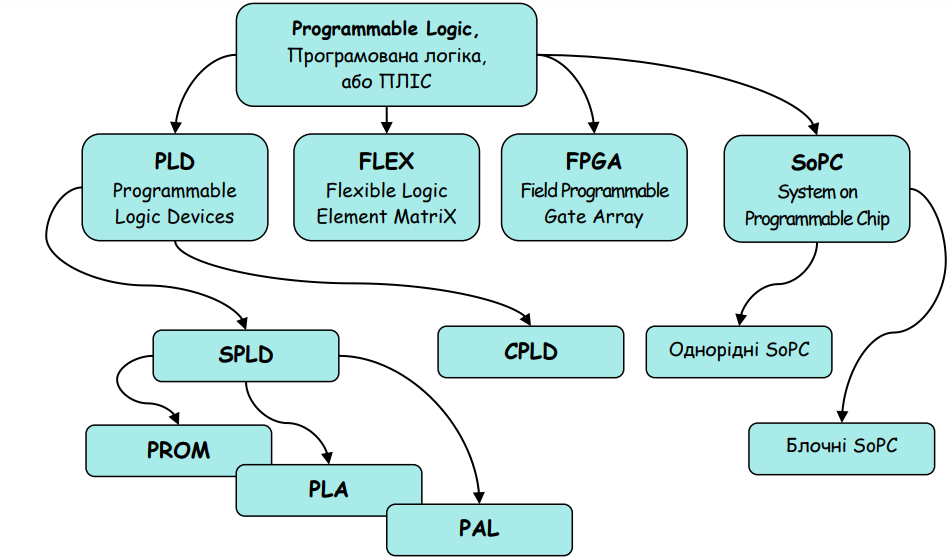


Рис. 2.1 – Класифікація ПЛІС

*Стандартні програмовані логічні пристрої (SPLD)*. До перших ПЛІС, які з'явилися на початку 70-х років, відносяться програмовані постійні запам'ятовуючі пристрої (ППЗП – Programmable Read Only Memory – PROM). Перший час PROM використовували винятково для зберігання інформації, пізніше їх стали застосовувати для реалізації логічних функцій. Структура PROM містить дві матриці: матриця DC, настроєна на реалізацію функції повного дешифратора, і програмована матриця «АБО» («OR»). PROM дозволяє реалізувати логічні функції, представлені в довершеній диз'юнктивній нормальній формі.

З 1971 р. стали випускатися програмовані логічні матриці (ПЛМ – Programmable Logіc Array – PLA), які містять дві програмовані матриці, одна з яких настроєна на реалізацію функції «І» («AND»), а інша – функції «АБО» («OR»).

Удосконалювання структури PLA привело до створення програмувованих матриць логіки (ПМЛ – Programmable Array Logіc – PAL), у яких, на відміну від PLA, програмується тільки матриця І, а матриця АБО має фіксовану настройку, при якій q проміжних шин зв'язується з одним виходом. Це дозволяє матрицю АБО реалізувати у вигляді сукупності q-входових дизъюнкторів. Вихідні буфери, що визначають архітектуру PAL, являють собою програмовані макроосередки, які можуть включати інвертор із трьома станами, тригери різних типів, елементи «виключне АБО» та ін.

Описані вище структури ПЛІС відносяться до стандартних програмованих логічних пристроїв (стандартних ПЛП, Standard Programmable Logіc Devіces, SPLD).

*Складні програмовані логічні пристрої (CPLD)*. Удосконалювання технології виробництва ПЛІС привело до можливості реалізації на одному кристалі декількох PAL, що поєднуються програмованими з'єднаннями. Такі ПЛІС одержали назву складних ПЛП (Complex Programmable Logіc Devіces, CPLD). Узагальнена структура CPLD містить матрицю функціональних блоків FB і програмовану матрицю перемикань (Swіtch Matrіx, SM).

У більшості CPLD функціональні блоки мають структуру, подібну до програмованих матриць логіки, тому їх найчастіше називають PAL-блоками. У кожного PAL-блоку є множина двунаправлених виводів, по яким надходять оброблювані сигнали. Основні логічні перетворення виконуються в PAL-блоках, а матриця перемикань служить лише для передачі сигналів між ними. Також у структурі CPLD присутні спеціалізовані входи, зв'язані як з матрицею перемикань, так і з усіма PAL-блоками. Ці входи звичайно використовують для передачі глобальних сигналів синхронізації і керування пристроєм.

*Матриці елементів гнучкої логіки (FLEX).* Подальший розвиток структура CPLD одержала в мікросхемах фірми Altera, названих матрицями елементів гнучкої логіки (Flexible Logic Element Matrix – FLEX). У таких ПЛІС відсутня звична PAL-структура, а є блоки логічних елементів, що поєднуються в LAB-модулі, зв'язок між якими здійснюються за допомогою горизонтальних і вертикальних каналів між з'єднань. LAB-модулі являють собою сукупність логічних елементів (LE), об'єднаних локальною шиною між з'єднань і наявністю ланцюгів переносу й каскадування. Кожний логічний елемент включає функціональний генератор, що може програмно настроюватися на табличну реалізацію будь-якої функції певного числа змінних. Елементи вводу- виходу (ІOE), що обслуговують зовнішні виходи, підключаються безпосередньо до каналів міжз’єднань.

*Вентильні матриці, що програмуються користувачем (FPGA)*. Якщо в FLEX-пристроях лінії каналів проходять через всю довжину або ширину кристала, то в програмованих користувачем вентильних матрицях (Field Programmable Gate Array – FPGA) лінії каналів складаються з окремих сегментів, що з'єднують програмованими матрицями перемикань, які дозволяють передавати сигнали в кожному із чотирьох напрямків.

Сегментовані міжз’єднання можуть утворювати складні розгалужені ланцюги, завдяки чому кожному з'єднанню може бути знайдено кілька альтернативних шляхів, при цьому ефективно використовуються металізовані лінії сегментів.

Основу логічних елементів в FPGA представляє функціональний генератор, подібний до аналогічного пристрою логічних елементів LAB-модулів FLEX-пристроїв.

Протягом перших років розвитку ПЛІС архітектури CPLD і FPGA були представлені в «чистому» вигляді, але кожна з них має свої переваги та недоліки. Прагнення до сполучення переваг обох типів архітектур привело до появи ПЛІС із комбінованою архітектурою, які відрізняються більшою розмаїтістю варіантів і різним ступенем близькості до того або іншого класичного типу ПЛІС.

Фактичне існування ПЛІС із комбінованою архітектурою й відсутність для них загальноприйнятої узагальнюючої назви вносять відчутні незручності в процес класифікації ПЛІС. Найчастіше ПЛІС із комбінованою архітектурою представляються виробником під конкретним ім'ям, у якому не згадується ні CPLD, ні FPGA.

*ПЛІС типу «система на кристалі».* Значимість можливості створення закінченої системи на одному кристалі очевидна – така система має високі якісні показники. Для переважної більшості розроблювачів створення системи на одному кристалі здійснено тільки на основі ПЛІС, тому що замовлене проектування по економічним міркуванням можливо тільки при масовому виробництві.

Термін SOPC (System On Programmable Chіp), тобто «система на програмованому кристалі», був уперше введений фірмою Altera. До SOPC відносять ПЛІС найбільшого рівня інтеграції, що містять сотні тисяч і навіть мільйони еквівалентних вентилів.

Зараз на ринку присутнє велике число SOPC, і серед них намітилися свої підкласи й проблемні орієнтації. Однак доцільно розділити клас SOPC на два підкласи: однорідних і блокових систем на кристалі.

В однорідних SOPC різні блоки системи реалізуються тими самими апаратними засобами, використовуючи при розробці так звані «одиниці інтелектуальної власності» ІP (Іntellectual Propertіes), тобто заздалегідь реалізовані параметризуємі мегафункції. Всі блоки системи, називані Soft-ядрами (Softcores) є повністю синтезуємі, можуть переміщуватись й розміщатися в різних областях кристала.

Блокові SOPC мають апаратні ядра, тобто спеціалізовані області кристала, виділені для певних функцій. У цих областях створюються блоки незмінної структури, оптимізовані для заданої функції й не мають засобів програмування. Такі блоки будемо називати hard-ядрами (Hardcores). Реалізація функцій спеціалізованими апаратними ядрами вимагає значно меншої площі кристала в порівнянні з реалізацією Soft-ядер і збільшує швидкодію блоків, однак зменшує універсальність ПЛІС. Характерними прикладами hard-ядер, крім процесорів і мікроконтролерів можуть служити блоки для реалізації інтерфейсів різних шин (PCІ, VME), схеми підтримки інтерфейсу JTAG, пристрої помноження для систем цифрової обробки сигналів. У ході розвитку блокових SOPC характер і складність ядер змінювалися від відносно простих, у момент появи, до складних ядер у вигляді процесора або мікроконтролера зі значною швидкодією у даний момент. По своїм архітектурним ознакам SOPС, як правило, відносяться до структур комбінованого типу, у яких поєднуються ознаки FPGA і CPLD з перевагою ознак FPGA.

Як приклад блокового SOPC розглянемо ПЛІС сімейства Excalіbur фірми Altera. Загальний план мікросхеми показує основні блоки кристала, у якому роль процесора грають ядра RAM або MІPS. Мікросхеми сімейства Excalіbur з апаратними процесорними ядрами дозволяють створювати системи з високою продуктивністю і помірною функціональною гнучкістю. Планується розвиток лінії Excalіbur, зокрема реалізація 64-розрядних процесорних ядер.

### 2.1.2 Переваги використання ПЛІС

Сьогодні кінцеву розробку приладу доцільно виконувати на базі ПЛІС. Ріст можливостей сучасних ПЛІС дозволяє розробникам реалізувати складні мультипроцесорні системи на одному кристалі, а ведучі компанії розробники ПЛІС в свою чергу пропонують можливості застосування програмованих ядер процесорів спеціально розроблених для використання в ПЛІС, та апаратних процесорних ядер. Крім того ПЛІС устатковані вбудованими блоками пам’яті, периферією та схемами зв’язку. Динамічна реконфігурація є однією з потужніших якостей FPGA-MPSoC, яка забезпечує гнучкість проектування та дозволяє мультипроцесорним системам адаптуватися до конкретних вимог вирішуваної задачі.

Слід визначити основний сенс реалізації мультипроцесорних систем за застосування реконфігурованих технологій ПЛІС. Головний недолік систем цього класу у порівнянні з технологією ASIC це зниження продуктивності, що обумовлено відмінностями використання внутрішніх ресурсів мікросхем. Розміщення логічних вентилів мікросхеми ПЛІС строго визначено і розташування логічної конфігурації розроблюваного пристрою та трасування внутрішніх з’єднань не завжди носить оптимальний характер, що відображається на швидкодії та споживаної потужності. Однак сучасні ПЛІС мають в першу чергу порівняні технічні характеристики з мікросхемами ASIC, та сама технологія FPGA-MPSoC має ряд переваг, що обумовлюють вибір саме цього способу реалізації убудованих мультипроцесорних систем. Далі наведені основні переваги технології FPGAMPSoC, що обумовлюють її сучасний прогрес.

*Гнучкість та реконфігурованість*. Кількість програмних процесорних ядер, що можуть бути включені в систему, обмежена тільки об’ємом внутрішніх ресурсів кристалу ПЛІС. Також розробник має можливість незалежно конфігурувати кожний процесор додаючи кеш-пам’ять та додаткові обчислювальні потужності, наприклад, блок обробки даних з плаваючою комою (FPU, Floating Point Unit) та різноманітні співпроцесори для спеціалізованих обчислень, або змінюючи функціональність будь яких блоків. В цьому контексті важливо те, що пристрій може програмуватися та модифікуватися внутрисистемно, тобто функціональність пристрою вже убудованого в електронну систему може бути легко змінено, одноразово або багаторазово, забезпечуючи динамічну реконфігурацію системи, так кажучи «на-льоту».

*Зменшення часу розробки*. Процес розробки не включає безпосередньо виробництво інтегральної схеми, що значно скорочує час розробки. Під розробкою пристрою мається на увазі програмування заздалегідь підготовленого кристалу. Крім того на скорочення строків виходу на ринок готового пристрою значно впливає просте внесення змін у запрограмовану конфігурацію, а саме можливість зручної та швидкої модифікації, налагоджування та тестування, як готового пристрою цілком, так і його окремих ланцюгів, без залучування технологічного процесу виробництва мікросхем.

*Низька вартість*. Зараз ПЛІС масового виробництва відносно дешеві, і можуть бути застосовані як для промислового виробництва електронних пристроїв, так і в лабораторних умовах невеличкими командами розробників, в тому числі науковцями. Звичайно відсутність технологічного процесу виробництва і вільнопоширювані та широкодоступні засоби автоматизації проектування від провідних виробників ПЛІС також відображаються на ціну готового пристрою.

*Масштабованість.* Збільшення кількості процесорів, об’єму пам’яті та периферії можливо настільки наскільки дозволяють ресурси кристалу.

## 2.2 Програмовані логічні матриці, програмована матрична логіка, базові матричні кристали

### 2.2.1 Програмовані логічні матриці та програмована матрична логіка (ПЛМ і ПМЛ)

*Програмовані логічні матриці* з’явились в середині 1970-х років. Їх основою слугують послідовно включені програмовані матриці елементів “І” та “АБО” (рис. 2.2). В ПЛМ також входять блоки вхідних і вихідних буферних каскадів (БВх та Бвих).

Схема ПЛМ складається з наступних логічних блоків:

1) деяка кількість вхідних виводів, кожний з яких оснащений буферним елементом, що має одночасно прямий і інверсний виходи;

2) програмована матриця логічних елементів “І” з численними входами, причому на кожен вхід кожного логічного елементу “І” може бути поданий через повторювач чи інвертор будь-який вхідний сигнал чи його доповнення через програмоване комутаційне поле;

3) програмована матриця логічних елементів “АБО” з визначеною кількістю входів, кожний з яких підключений до відповідного числа входів вентилів “І”;

4) кілька буферних схем, розміщені між виходами вентилів “АБО” і зовнішніми виводами ПЛМ;

5) декілька зворотних зв’язків, що повертають вихідні сигнали в програмоване комутаційне поле для того, щоб вони могли бути оброблені як вхідні сигнали.

Вхідні буфери, якщо не виконують більш складні дії, перетворюють однофазні вхідні сигнали в парафазні і формують сигнал необхідної потужності для живлення матриці елементів “І”.

Вихідні буфери забезпечують необхідну навантажувальну здатність виходів, дозволяють чи забороняють вихід ПЛМ на зовнішні шини за допомогою сигналу ОЕ, а іноді виконують і більш складні дії.

Основними параметрами ПЛМ є число входів *m*, число термів *l* і число виходів *n*.

Змінні ,…, подаються через вхідний буфер на входи елементів “І” (кон’юнкторів), і в матриці “І” утворюються *l* терми. Під термом розуміється кон’юнкція, що зв’язує вхідні змінні, представлені в прямій чи інверсній формі. Число сформованих термів дорівнює числу кон’юнкторів чи числу виходів матриці “І”.

Х Х

Вих. буфер

Матр.

“АБО”

Матр.

“І”

Вхідний буфер

Х

О

\_

ОЕ

Рис 2.2 – Базова структура ПЛМ

Терми подаються далі на входи матриці “АБО”, тобто на входи диз’юнкторів, що формують вихідні функції. Число диз’юнкторів дорівнює числу сформованих функцій *n*.

Таким чином, ПЛМ реалізує диз’юнктивну нормальну форму (ДНФ) відтворених фунцій (дворівневу логіку). ПЛМ здатна реалізувати ситему *n* логічних функцій від *m* аргументів, що містить не більш *l* термів. Відтворені функції є комбінаціями з будь-якого числа термів, сформованих матрицею “І”. Які саме терми будуть вироблені і які комбінації цих термів складуть вихідні функції, визначається програмуванням ПЛМ.

*Програмована матрична логіка*

Одне з важливих застосувань великих інтегральних схем (ВІС) програмованої логіки – зміна ІС малого і серенього рівня інтеграції при реалізації, так званої довільної логіки. У цих застосуваннях логічна потужність ПЛМ найчастіше використовується неповно. Це виявляється, зокрема, при відтворенні типових для практики систем перемикаючих функцій, що не мають великих перетинів одна з одною по однаковим термам. У таких випадках можливість використання виходів будь-яких кон’юнкторів будь-якими диз’юнкторами (як передбачено в ПЛМ) стає зайвим ускладненням. Відмова від цієї можливості означає відмову від програмування матриці “АБО” і приводить до структури ПМЛ.

У ПМЛ (рис. 2.3.) виходи елементів “І” (виходи першої матриці) жорстко розподілені між елементами (входами матриці “АБО”). У показаної ПМЛ *m* входів, *n* виходів і *4n* елементів “І”, оскільки кожному елементу “АБО” придається по чотири кон’юнктори.

У порівнянні з ПЛМ схеми ПМЛ мають меншу функціональну гнучкість, тому що в них матриця “АБО” фіксована, але їхнє виготовлення і використання простіше. Переваги ПМЛ особливо виявляються при проектуванні нескладних пристроїв.

Підготовка задач до рішення на ПМЛ має багато загального з підходом до рішення задачна ПЛМ, але є і розходження. Для ПМЛ важливо зменшити число елементів “І” для кожного виходу, але якщо для ПЛМ прагнуть знайти представлення функції з найбільшим числом загальних термів, то для ПМЛ це не потрібно, оскільки елементи “І” фіксовані по своїх виходах і не можуть бути використані іншими виходами (тобото для інших функцій).

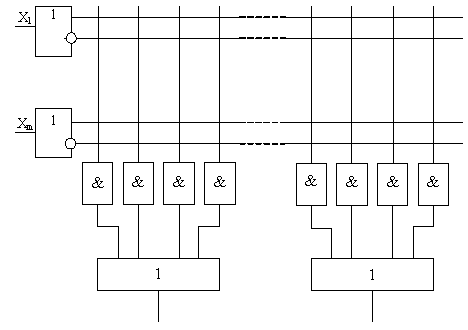


Рис. 2.3 – Базова структура ПМЛ

*Функціональні різновиди ПЛМ і ПМЛ*

Розглянуті структури ПЛМ і ПМЛ – базові, з яких почався розвиток цих напрямків. Надалі відбувалося збагачення функціональних можливостей ПЛМ і ПМЛ за допомогою ряду прийомів, у першу чергу наступних.

*Програмування вихідних буферів.* В схемах з програмованим вихідним буфером забезпечується можливість одержання вихідних функцій у прямому чи інверсному вигляді. У такій схемі (рис. 2.4) вироблені матрицями функції ,…, проходять через вихідний буфер, розрядні схеми якого виконані як суматори по модулю 2.

У показаній схемі другі входи суматорів одержують нульові сигнали від потенціалу “землі” через плавкі перемички (ПП). При цьому функції з виходу матриць передаються через буфер без змін. Якщо перепалити перемичку нижнього входу суматора, то він отримає сигнал логічної одиниці від джерела живлення через резистор R. Складаючись по модулю 2 з одиницею, функції будуть інвертуватися. Отже, у лініях з цілими перемичками функції проходять через буфер незмінними, а в лініях з відсутніми премичками – інвертуються.

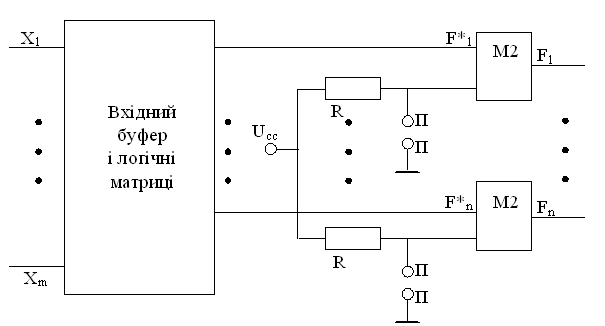


Рис. 2.4 – Схема програмованого вихідного буфера

Програмований буфер дає додаткові можливості для мінімізації кількості термів реалізованій системі. У вихідній системі можна заміняти функції їхніми інверсіями, якщо це приводить до зменшення числа термів. Повернення до вихідної системи буде забезпечено просто перепрограмуванням буфера.

*Схеми з двонаправленимим виходами.* Використовуючи елементи з трьома станами виходу, можна побудувати схему, в якій деякі виходи можна пристосовувати для роботи як входи чи виходи у залежності від програмування перемичок. У такій схемі один з кон’юнкторів призначений для керування елементом з трьома станами виходу (рис. 2.5). Вихід елемента зв’язаний з матрицею “І” як вхід.

Можливі чотири режими виходу Вх/Вих у залежності від того, як запрограмовані входи кон’юнктора:

1) Усі перемички – без змін. У цьому режимі на виході кон’юнктора буде нуль, буфер має третій стан виходу і вихід функціонує як вхід.

2) Усі перемички перепалені, на виході кон’юнктора одиниця, буфер активний, вихід працює як вхід (його сигнали не використовуються в матриці “І”).

3) Вихід зі зворотнім зв’язком. Цей режим відрізняється від попереднього тільки тим, що сигнали виходу використовуються в матриці “І”.

4) Керований вихід. Входи кон’юнктора програмуються.

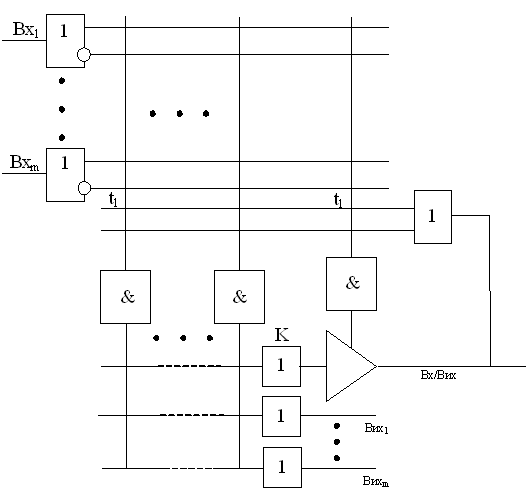


Рис. 2.5 – Схема з двонаправленим буфером

У схемі з деякою кількістю двонаправлених виходів можна змінювати співвідношення кількості входів-виходів. Якщо число входів дорівнює *m*, число виходів *n* і число двосторонніх виходів *p*, то можливо збільшити число входів від *m* до *m+p* і число виходів від *n* до *n+p* за умови, що сума числа входів і виходів не перевищує *m+n+p*.

*Схеми з пам’яттю.* Ці схеми дозволяюь будувати автомати найбільш зручним способом, тому що крім комбінаційної частини містять на кристалі тригери (регістри) типу D (рис. 2.6). ПЛМ із пам’яттю характеризуються чотирма параметрами. Крім трьох звичайних параметрів, вона має і параметр r – число елементів пам’яті (розрядів регістру).

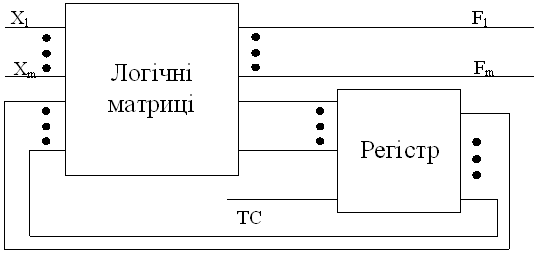


Рис. 2.6 – Структура ПЛМ з пам’яттю

Структура збігається з канонічною схемою автомата. Результат даного кроку обробки інформації залежить у ній від результатів попередніх кроків, що забезпечується зворотним зв’язком з регістра на вхід ПЛМ. Максимальне число внутрішніх станів автомата . Автомат розглядається як синхронний – петля зворотного зв’язку активізується тільки з дозволу тактових сигналів (ТС).

*ПЛМ з розподільчими кон’юнкторами*. Поряд з модифікаціями схем, які розглянуті вище, існують і специфічні модифікації, що відносяться тільки до ПЛМ. До них відноситься варіант із, так званими, розподільчими кон’юнкторами. Прийом “розподілу кон’юнкторів” полягає в наступному.

Для двох суміжних елементів “АБО” придається деяка кількість кон’юнкторів, що може бути довільно розділена між цими суміжними елементами. Інші елементи “АБО” використовувати даний набір кон’юнкторів не можуть. Повного програмування матриці “АБО” тут не виникає, але ця модифікація є кроком у напрямку до ПЛМ.

Варіант із розподільчими кон’юнкторами пом’якшує найбільш очевидне обмеження функціональних можливостей простих (твердих) ПЛМ – фіксоване число елементів “І” на входах елементів “АБО”, якого може не вистачити при відтворенні складних функцій. Маючи ПЛМ із розподільчими кон’юнкторами і розміщуючи складну функцію поруч з простою, можна запозичити частину загального набору кон’юнкторів у простої функції на користь складної.

Варіант схемотехнічної реалізації подільності кон’юнкторів показаних на рис. 2.7. У ПЛМ існує додатковий набір елементів “АБО” і додавання по модулю 2, за допомогою якого можна комбінувати сигнали обох виходів основних схем “АБО” для утворення залишкових значень функцій і . Виходи основних схем “АБО” можуть об’єднуватись по операціях диз’юнкції чи додавання по модулю 2 і розподіляться по основним виходам і .

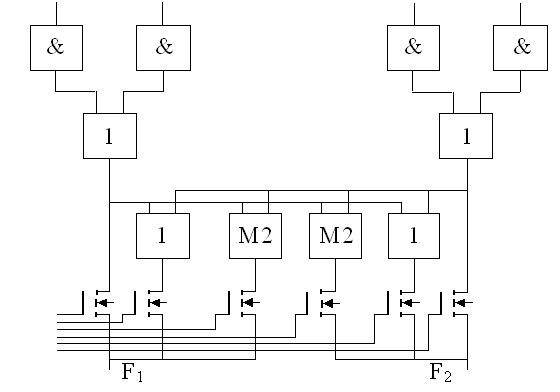


Рис. 2.7 – Розділення термів в ПЛМ

Операція додавання по модулю 2 дає додаткові функціональні можливості. Характер отриманих функцій залежить від того, який із трьох транзисторів у показаних двох групах буде провідним.

### 2.2.2 Базові матричні кристали (вентильні матриці з програмуванням маскою)

*Класифікація БМК*

Класифікація БМК показана на рис. 2.8. Першочерговою і класичною є структура канального БМК. У внутрішній (центральній) області такого БМК розміщена матриця базових комірок і канали для трасування.

Канали можуть бути вертикальними і горизонтальними, або лише вертикальними. Канальні БМК можуть мати більшу можливість по створенню зв’язків, але мають низьку щільність компонування через значні затрати площі кристалута області між’зєднань [8].

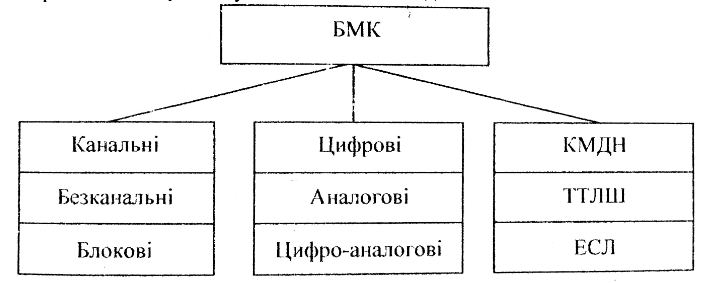


Рис. 2.8 – Класифікація базових матричних кристалів

Зростання рівня інтеграції БМК веде до швидкого зростання числа необхідних між’зєднань між базовими комірками, а значить і площі, для них. Пошук шляхів створення БМК високого рівня інтеграції з мінімізацією площі, під між’зєднання, привело до *безканальної архітектури* БМК. Внутрішня область такого БМК містить щільно упаковані ряди базових комірок і не має фіксованих каналів для трасування між’зєднань. В такому кристалі будь-яка область, в якій розміщені БК, може бути використана як для створення логічної схеми, так і для створення між’зєднань. Внаслідок більш раціонального розміщення зв’язків в безканальному БМК зменшується і затримка передачі сигналів, так як довжини і паразитні ємності між’зєднань зменшуються.

Безканальні БМК характерні для КМДН-схемотехніки, в якій компактність схемних елементів і мала потужність розсіювання БК при їх роботі на високих частотах дозволяє щільно упаковувати базові комірки.

Так як в безканальних БМК положення трасувальних каналів і комірок на робочому полі не є фіксованим і при проектуванні площа кристалу може перерозподілюватися між трасувальними каналами і функціональними комірками, то втрати площі кристалу знижуються. Наприклад в БМК з щільним розміщенням на робочому полі рядів транзисторів в деяких рядах реалізуються логічні елементи, а інші ряди використовуються під трасувальні канали, в них транзистори залишаються не зкомутованими і не використовуються ( над ними проходять траси). В залежності від завантаженості каналів, для них може бути відведене різне число рядів транзисторів.

В КМДН БМК використовуються також архітектура зі змінною довжиною комірок. При цьому кожний ряд представляє собою послідовне з’єднання пар *n-* і *p-* канальних транзисторів. Якщо в такому довгому ланцюзі розмістити в заданих місцях пари закритих транзисторів, то коло буде розділене на базові комірки заданої довжини. Внутрішня область (ВО) кристалу обмежена периферійною областю (ПО), розміщеною по краям прямокутної пластини БМК. В периферійній області розміщені спеціальні ПБК, набір схемних елементів, який орієнтований на вирішення задач вводу/виводу сигналів, а також контактні площадки (КП). Ріст рівня інтеграції веде до можливості реалізації на одному кристалі все більш складних пристроїв і систем. Це спричино появу блочних структур БМК, архітектура яких спрощує побудову комбінаційних пристроїв, що містять, як блоки логічної обробки даних, так і пам'ять чи інші спеціалізовані блоки. При цьому в БМК реалізуються декілька блоків-підматриць, кожний з яких має структуру БМК меншого розміру. Між блоками розміщуються трасувальні канали. На периферії блоків – внутрішні буферні каскади для формування достатньо потужних сигналів.

Тип сигналів, що оброблюються (цифрові, аналогові) впливає на якість і склад схемних елементів базових комірок. В зв’язку з цим БМК поділяються на цифрові, аналогові і цифроаналогові. Аналогові і цифроаналогові БМК з’явилися після цифрових і менш розповсюджені, мають склад базових комірок, які дозволяють отримати на їх основі такі схеми, як операційні підсилювачі, аналогові ключи, компаратори і т.д.

На рис. 2.9. представлений один із варіантів БК БМК типу КМДН. Схемними елементами таких БК служать тільки транзистори з *n-* і *p-* каналами.

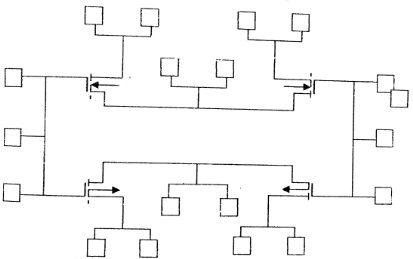


Рис. 2.9 – Варіант базової комірки БМК типу КМДН

Число транзисторів в комірці обирається по результатах аналізу частоти використання різних логічних елементів у пристроях заданого класу і вимогах по навантажувальній здатності, швидкодії. Високий коефіцієнт використання транзисторів дають кристали з числом транзисторів в комірці 4, 8 чи 10. На рис. 2.9 показана електрична схема комірки з 4 транзисторами. Квадратні елементи – контактні площадки до затворів і фіксовані до елементів комірок. Транзистори можна з’єднувати послідовно чи паралельно, тобто можна отримувати типові схеми логічних елементів І-НІ і АБО-НІ. В схемотехніці КМДН транзистори з протилежними по типу провідності каналами завжди використовуються попарно, тому пари транзисторів можуть мати загальний затвір.

Складність комірки досягається об’єднанням простих комірок в групу.

Параметри БМК можна розділити на 4 групи:

1) функціональні можливості (число еквівалентних вентилів, тип БК, число МБК і ПБК, склад бібліотеки функціональних комірок);

2) електричні параметри (рівні напруг, що кодують логічні сигнали, напруга живлення, затримки розповсюдження сигналів, максимальні частоти перемикань);

3) конструктивно-технологічні (тип корпусу, число виводів, число рівнів металізації, площа кристалу);

4) експлуатаційні характеристики.

## Висновки до розділу

В даному розділі розглянуто питання, яке стосується програмованих логічних матриць, програмованих матричних логік та базових кристалів. З наведеної інформації можна зробити висновок та вказати великі переваги використання ПЛМ, ПМЛ та БК в процесі програмування пристроїв.

Програмовані схеми з матричною структурою являють собою набір елементів і вузлів, не пов’язаних між собою. Для створення конкретного пристрою елементи з’єднуються на заключному етапі виготовлення – відповідно до замовлення, або програмуються самим користувачем. Фрагменти схеми з’єднуються або створенням, або усуненням перемичок, так само, як це робиться при програмуванні ЗУ.

Використання ПМЛ значно полегшує процес заміни ІС малого і середнього рівнів інтеграції при реалізації довільних логічних функцій.

Підготовка завдань до вирішення на ПМЛ має багато спільного з підходом до вирішення завдань на ПЛМ, але є й відмінності. Для ПМЛ логічну функцію потрібно представити таким чином, щоб число термів було не більше числа входів вихідних елементів «АБО».

 Базові матричні кристали (БМК) стали засобом, на основі якого при порівняно низьких витратах стало можливим виконання спеціалізованих пристроїв у вигляді БІС / НВІС. БМК являють собою сукупність регулярно розташованих на напівпровідниковому кристалі осередків, кожен з яких містить типовий набір елементів – транзисторів, резисторів. Такий кристал є напівфабрикатом, який виробляється в масових кількостях без орієнтації на конкретного споживача. Щоб на його основі було отримано конкретний пристрій, необхідно виконати необхідні з’єднання елементів і осередків. Виконання з’єднань здійснюється на заключних етапах виготовлення, а проектування пристрою, таким чином, зводиться до створення малюнка міжз’єднань. Для спрощення процесу проектування є бібліотеки готових рішень. Кожне готове рішення пропонує певний варіант міжз’єднань елементів усередині осередку, що дозволяє на базі однієї або кількох осередків отримати готовий функціональний вузол – логічний елемент, тригер і т.п.

# РОЗДІЛ 3

# ОПИС РЕАЛІЗАЦІЇ АЛГОРИТМУ ДСТУ ГОСТ 28147 НА ПЛІС

## 3.1 Основні поняття про системи автоматизованого проектування

### 3.1.1 Основні визначення та класифікація САПР

Система автоматизованого проектування (САПР) – автоматизована система, призначена для автоматизації технологічного процесу проектування виробу, кінцевим результатом якого є комплект проектно-конструкторської документації, достатньої для виготовлення та подальшої експлуатації об'єкта проектування. Процес автоматизованого проектування реалізується на базі спеціального програмного забезпечення, автоматизованих банків даних, широкого набору периферійних пристроїв [9].

САПР забезпечує створення, зберігання і обробку моделей геометричних об'єктів і їх графічне зображення за допомогою комп'ютера.

Автоматизоване проектування (computer-aided design – CAD) є технологією суть якої полягає у використанні комп’ютерних систем для полегшення створення, змін, аналізу і оптимізації проектів. Таким чином, будь-яка програма, що працює з комп’ютерною графікою, так само як і будь-який додаток використовуваний в інженерних розрахунках, відноситься до систем автоматизованого проектування.

Автоматизоване виробництво (computer-aided manufacturing – САМ) – це технологія, що полягає у використанні комп’ютерних систем для планування, управління і контролю операцій виробництва через прямий або непрямий інтерфейс з виробничими ресурсами підприємства. Одним з найбільш широко застосовуваних підходів до автоматизації виробництва є числове програмне управління (ЧПУ, numerical control – NC).

Автоматизоване конструювання (computer-aided engineering – САЕ) – полягає у використанні комп’ютерних систем для аналізу геометрії CAD, моделювання і вивчення поведінки виробу для удосконалення і оптимізації його конструкції. Засоби САЕ можуть здійснювати багато різних варіантів аналізу. Програми для кінематичних розрахунків, здатні визначати траєкторії руху і швидкості ланок в механізмах. Програми динамічного аналізу можуть використовуватися для визначення навантажень і зсувів в складних пристроях типу автомобілів. Програми верифікації і аналізу логіки і синхронізації імітують роботу складних електронних ланцюгів.

Перевагами методів аналізу і оптимізації конструкцій є те, що вони дозволяють конструктору побачити поведінку кінцевого виробу і виявити можливі помилки до створення і тестування реальних прототипів, уникнувши певних витрат. Оскільки вартість конструювання на останніх стадіях розробки і виробництва продукту є значною, а це призводить до скорочення термінів і вартості розробки.

Класифікувати САПР можна за такими ознаками:

а) за ступенем формалізації вирішуваних задач:

1) повністю формалізованих задач(застосовуються тільки для вирішення простих задач проектування);

2) частково формалізованих задач(придатні для вирішення завдань у багатьох галузях промислового виробництва);

3) не формалізованих задач(знаходяться у стадії розробки та дослідження,не застосовуються).

б) за функціональним призначенням:

1) розрахунково-оптимізаційні;

2) графічні;

3) графоаналітичні;

4) інформаційні і т.п.

в) по спеціалізації:

1) спеціалізовані;

2) інваріантні.

г) за технічною організацією:

1) однорівневі;

2) багаторівневі.

САПР складається з проектуючої і обслуговуючої підсистем.

Проектуючі підсистеми безпосередньо виконують проектні процедури. Прикладами проектуючи підсистем можуть слугувати підсистеми геометричного тривимірного моделювання механічних об'єктів, виготовлення конструкторської документації, схемотехнічного аналізу, трасування з'єднань у друкованих платах.

Обслуговуючі підсистеми забезпечують функціонування проектуючи підсистем, їхню сукупність часто називають системним середовищем (або оболонкою) САПР. Типовими обслуговуючими підсистемами є підсистеми керування проектними даними (PDM – Product Data Management), керування процесом проектування (DesPM – Desіgn Process Management), користувацького інтерфейсу для зв'язку розробників з ЕОМ, CASE (Computer Aіded Software Engіneerіng) для розробки та супроводу програмного забезпечення САПР, навчальні підсистеми для освоєння користувачами технологій, реалізованих у САПР.

### 3.1.2. Загальні відомості про систему MAX+PLUS II

Назва системи MAX+PLUS II є абревіатурою від *Multiple Array Matrix Programmable Logic User System* (система програмування користувачем логіки впорядкованих структур).

Система MAX+PLUS II пропонує повний спектр можливостей логічного дизайну: різноманітні засоби опису проектів з ієрархічною структурою, могутній логічний синтез, компіляцію з заданими часовими параметрами, поділ на функціональне і часове тестування (симуляцію), тестування декількох зв’язаних пристроїв, аналіз часових параметрів системи, автоматичну локалізацію помилок, а також програмування і верифікацію пристроїв. У програмі MAX+PLUS II можна як читати, так і записувати файли мовою AHDL і файли трасування у форматі EDIF, файли на мовах орису *Verilog* HDL, а також схемні файли OrCAD. Крім того, система MAX+PLUS II читає файли трасування, створені за допомогою ПО *Xilinx,* і записує файли затримок у форматі SDF зручності взаємодії з пакетами, що працюють з іншими промисловими стандартами.

Верифікація проекту (Project verification) виконується за допомогою симулятора, результати роботи якого зручно переглянути в сигнальному редакторі *Waveform Editor*. Тестові послідовності створюються також у сигнальному редакторі.

Компіляція проекту, включаючи витяг списку з’єднань (Netlist Extractor), бази даних проекту (Data Base Builder), логічний синтез (logic synthesis), часові, функціональні параметри проекту (SNF Extractor), розбивку на частини (Partitioner), трасування (Fitter) і формування файлу програмування чи завантаження (Assembler) виконуються за допомогою компілятора системи.

Безпосереднє програмування чи завантаження конфігурації з використанням відповідного апаратного забезпечення виконується з використанням модуля програмування (Programmer).

Багато характерних команд – такі як відкриття файлів, уведення призначень пристроїв, виводів і логічних елементів, компіляція поточного проекту – схожі для багатьох додатків системи MAX+PLUS II. Можна легко комбінувати різні типи файлів проекту в ієрархічному проекті, вибираючи для кожного функціонального блоку той формат опису проекту, що більше підходить. Можна одночасно працювати з різними додатками системи MAX+PLUS II.

Основою системи MAX+PLUS II є компілятор, що забезпечує могутні засоби обробки проекту, при цьому можна задавати потрібні режими роботи компілятора. Автоматична локалізація помилки, видача повідомлень і велика документація про помилки прискорюють і полегшують проведення змін у дизайні. Можна створювати вихідні файли в різних форматах для різних цілей, таких, як робота функцій, часових параметрів і зв’язку декількох пристроїв; аналізу часових параметрів; програмування пристрою [10 c.116].

## 3.2 Приклад розробки алгоритму блокового шифрування на основі програмованих логічних матриць

### 3.2.1. Структурна блок-схема пристрою, принципова схема суматора по модулю два

*Структурна блок-схема пристрою*

В даному розділі проводиться проектування пристрою на ПЛМ, який реалізує алгоритм блокового шифрування ДСТУ ГОСТ 28147.

Згідно з загальними методами проектування, для спрощення процесу розробки пристрою, доцільно виділити в ньому блок управління та операційні блоки. Спрощена структурна схема пристрою зображена на рис. 3.1.

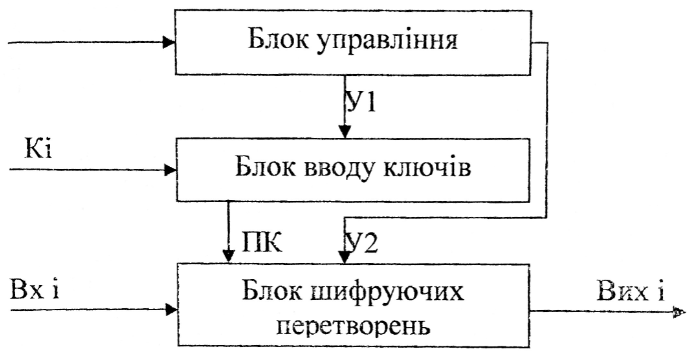


Рис.3.1 – Структурна блок-схема пристрою

Як видно зі схеми, пристрій складається з трьох основних блоків, які мають наступне призначення:

1) блок управління – забезпечує керування та синхронну роботу всіх блоків пристрою;

2) блок вводу ключів – забезпечує запис, зберігання ключової інформації та видачу на блок шифруючи перетворень вибраної частини ключової інформації;

3) блок шифруючи перетворень – призначений для здійснення криптографічних перетворень блоків вхідної інформації.

*Принципова схема суматора по модулю два*

Даний блок здійснює додавання за модулем 2. Принципова схема блоку зображена на рис. 3.2.

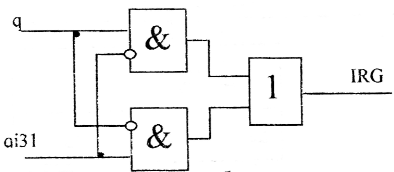


Рис. 3.2 – Принципова схема блоку суматора

### 3.2.2 Опис роботи блоку вводу ключів, блоку шифруючих перетворень, блоку шифруючи регістрів та та блоку управління

*Опис роботи блоку вводу ключів*

Даний блок повинен забезпечувати запис від зовнішнього джерела послідовно 256 бітів ключової інформації, їх збереження та видачу певного 32 бітового блоку ключової інформації по керуючому сигналу від блоку управління.

Для збереження ключової інформації доцільно використовувати регістри зсуву, а для вибору сегменту ключа – мультиплексор.

Даний блок здійснює послідовний запис 256 бітів ключової інформації у вісім 32 бітних регістрів, які з’єднані послідовно. Запис в регістри здійснюється по передньому фронту зовнішнього тактуючого сигналу, завдяки цьому зникає необхідність в синхронізації роботи зовнішнього джерела сигналу та даного блоку.

Лічильник СТ 256 після 256 імпульсів тактуючого сигналу (необхідних для запису 256 бітів ключової інформації) видає сигнал дозвіл вводу ключа (ДВК), який свідчить про запис ключа та забороняє подальший запис інформації в регістри. Сигнал ДВК надходить до блоку управління. У разі необхідності зміни ключової інформації необхідно подати імпульс на вивід ResK, який здійснює скидання лічильника (що дозволяє запис у регістри).

Вивід необхідних 32 бітів ключової інформації здійснюється через мультиплексом, керування яким здійснюється по сигналу ВК[0…2] з блоку управління. Сегмент ключа виводиться на виводи К[0…31].

*Опис роботи блоку шифруючи перетворень*

Даний блок здійснює криптографічні перетворення над блоками вхідних даних і має складну структуру. Тому доцільно його поділити на ряд блоків. Структурна схема блоку зображена на рис. 3.3

Призначення блоків:

1) блок шифруючих регістрів – призначений для вводу/виводу та обробки 64 бітних блоків інформації;

2) S-BOX – перетворює проміжну інформацію, шляхом заміщення восьми чотирьох бітних блоків вхідної інформації згідно таблиць заміщеня;

3) регістр зсуву – забезпечує зсув інформації на 11 розрядів вліво та виводить результат у послідовному вигляді;

4) суматор – здійснює додавання за модулем два.

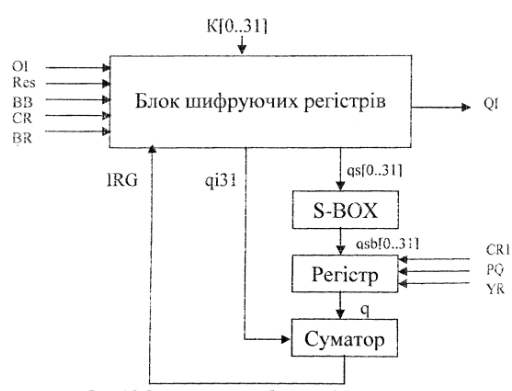


Рис. 3.3 – Структурна схема блоку шифруючи перетворень.

*Блок шифруючих регістрів*

Даний блок повинен забезпечувати послідовний ввід/вивід 64 бітних блоків даних, також збереження проміжних результатів під час їх обробки та додавання 32 бітний блок даних з 32 бітним блоком ключа.

Для збереження даних доцільно використовувати два 32 бітних регістри зсуву з’єднаних послідовно.

В даному блоці здійснюється послідовний запис вхідного 64 бітного блоку даних у два 32 бітні регістри зсуву, які з’єднані послідовно. Запис здійснюється по передньому фронту зовнішнього тактуючого сигналу CLK. Одночасно з записом блоку даних проходить послідовний вивід обробленого попередньо блоку по тактовому сигналу CLK.

Лічильник СТ здійснює підрахунок кількості зовнішніх тактових імпульсів та після 64-того імпульсу видає сигнал заборони подальшого запису/виводу даних. Мультиплексори здійснюють вибір необхідних сигналів для наступної обробки, по сигналу BRG з блоку управління (в залежності від номера інтерації). На вхід ІRG поступає послідовно 32 біта обробленої інформації після чергової інтерації. По входу CRG здійснюється тактування регістрів під час обробки даних (сигнал поступає з блоку управління). Після 32 тактів інтерацій блок управління подає сигнал Res, який скидає лічильник, дозволяючи цим ввід/вивід інформації. Можлива зовнішня подача цього сигналу для припинення процесу обробки інформації. Виводи qs[0…31] та qi31 призначені для видачі проміжної інформації, для наступної обробки.

*Описання роботи блоку управління*

Блок управління призначений для керування роботою операційних блоків і забезпечення синхронної роботи всіх елементів пристрою. Видача сигналів управління здійснюється в залежності від загального тактового сигналу з входу С. Функціональна схема блоку зображена на рис. 3.4.

Даний блок складається з двох лічильників, з’єднаних послідовно, які підраховують кількість тактових імпульсів і згідно їх кількості видають відповідні керуючі сигнали. Для кращого усвідомлення роботи даного блоку необхідно описати призначення входів і виходів.

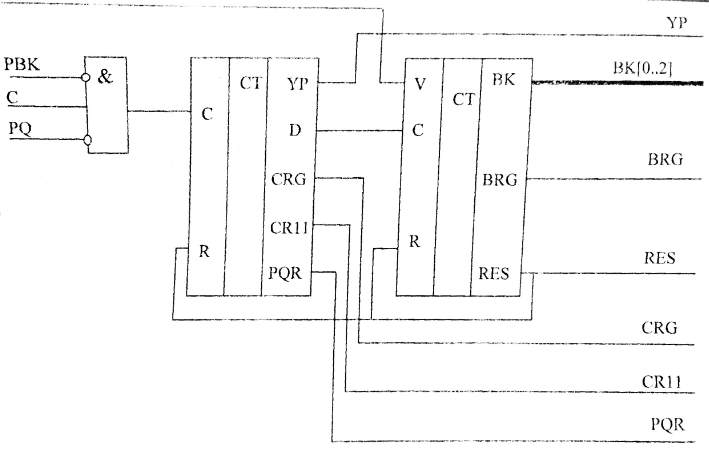


Рис. 3.4 – Функціональна схема блоку управління

Входи:

РВК – вхід, який забороняє обробку інформацію (сигналізує про відсутність ключових даних);

PQ – сигнал дозволу вводу/виводу інформації, що оброблюється (свідчить про закінчення процесу обробки інформації та готовності до обробки наступного блоку);

С – вхід тактових імпульсів;

V – сигал, по якому здійснюється вибір режиму роботи пристрою (зашифрування – розшифрування).

Виходи:

CRG – сигнал тактування регістрів в блоці шифруючи перетворень;

УР – сигнал управління регістром зсуву;

CR11 – сигнал тактування регістру зсуву;

PQR – дозвіл виходу інформації з регістру зсуву;

ВК[0…2] – управління вибору частини ключа шифрування;

BRG – вибір режиму роботи регістрів в блоці шифруючи перетворень;

RES – скидання лічильникав блоці шифруючи перетворень, даж дозвіл на ввід/вивід інформації [10, c. 227-235].

### 3.2.3 Побудова S-боків підстановок та реалізація регістру зсуву

*Побудова S-боків підстановок*

Функціональна схема блоку зображена на рис.3.5. Даний блок розбиває вхідний блок даних на вісім 4-х бітних сегментів та здійснює їхню заміну згідно таблиці підстановок, записаній у мікросхемі пам’яті. Мікросхеми пам’яті організовані 16\*4 біт. Виклик необхідної підстановки здійснюється по 4-х бітній адресі, якою виступає вхідний сегмент. Взагалі блок нараховує 8 мікросхем пам’яті, по одній на кожен сегмент. На виході сегменти складаються у вихідний 32 бітний блок. Дані надходять на вхід qs[0…31] та знімаються з виходу qsb[0…31].

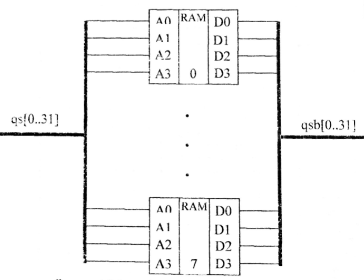


Рис. 3.5 – Функціональна схема S-BOX

В даному блоці реалізується додавання за модулем 2 сигналів з входів q та qi31та видача результату на вихід IRG.

*Реалізація регістру зсуву*

Даний регістр здійснює зсув вхідного 32-х бітного блоку даних на 11 позицій вліво. Функціональна схема регістру зображена на рис. 3.6.

Як видно зі схеми регістр складається з 32-х послідовно з’єднаних D тригерів. Запис даних здійснюється паралельно з входів qsb[0…31], по відсутності сигналу УР з блоку управління (який забороняє подачу вхідних даних на вхід тригерів) та наявності тактуючого сигналу СR11 з блоку управління. Далі по сигналу УР дозволяється зсув даних в регістрі та послідовною подачею 11 тактуючих імпульсів сигналу СR11, здійснюється зсув. По сигналу PQR з блоку управління дозволяється послідовний вивід даних з регістру на вихід q [10, c. 233-236].

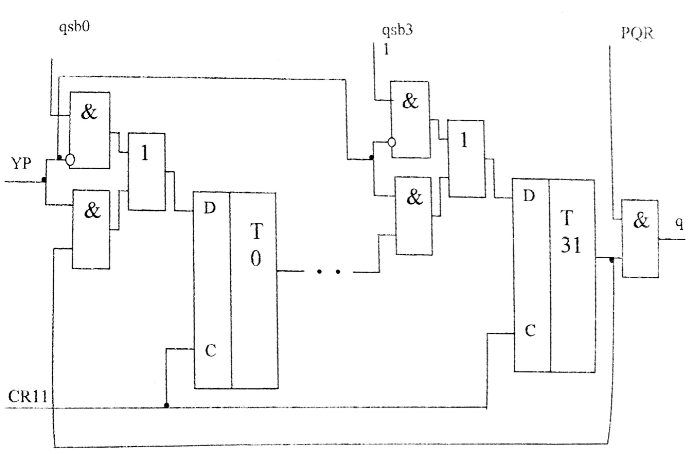


Рис. 3.6 – Функціональна схема регістру зсуву

## 3.3 Принципи проектування алгоритму ГОСТ 28147 на основі ПЛМ

### 3.3.1 Етапи проектних процедур

Розглянемо проектування з використанням САПР . Розробка виконується в наступному порядку:

*Специфікація проекту*

Складання змістовної граф-схеми чи алгоритму функціональної блок-схеми пристрою. Перша задача – перехід від технічного завдання до формалізованого опису проектованого пристрою. Сама форма технічного завдання може визначити проектувальнику ті чи інші засоби, хоча не виключено, що більш ефективним міг би бути інший метод опису проекту чи його фрагментів.

Декомпозиція може зводитися до складання схем алгоритмів функціонування фрагментів або до функціональної схеми пристрою і його частин. Можливим варіантом для досить складних систем буде розумне сполучення поведінкової і структурної розбивки проекту. Розбивка здійснюється не тільки в рамках одного рівня ієрархії. Для більшості проектів виробляється і розбивка на ієрархічно організовані рівні. Істотною задачею, що розв’язується на цьому етапі, є уточнення й узгодження з замовником інтерфейсних функцій проекту.

*Розробка загальної структури проекту*

Основна задача – вибір припустимих для реалізації кожного рівня ієрархії елементів, визначення зв’язків між ними, і якщо параметри елементів настроюються, - то їх настроювання. Ряд моментів для етапу є визначальними: з одного боку, - це джерело набору припустимих елементів, з іншого – засіб опису з’єднань елементів між собою, а при необхідності, і можливість опису нових (специфічних для цього проекту) елементів [11].

*Змістовний опис проекту і його частин*

Впровадження САПР дозволяє створювати ефективнф керовані і контрольовані описи проектів і їхніх частин. Причому той самий пристрій може бути описаний за допомогою різних засобів САПР. Більш того, більшість САПР дозволяють трансформувати один вид опису в інший. В даний час до найбільш розповсюдженого, універсального способу опису проекту, що застосовується для будь-якого рівня його ієрархії, відносяться графічний і текстовий. Графічне представлення проекту в сучасних САПР може створюватись як у базисі графічних символів проектувальника, так і в базисі припустимих для обраної САПР бібліотечних елементів. Основні переваги графічного способу – його традиційність і наочність.

*Розробка загальної структури операційного блоку*

Архітектурно-структурний опис операційного блоку базується на завданні структури з’єднанні окремих елементів. Традиційний графічний спосіб представлення структури з’єднань, залишаючись найбільш наочним способом, у сучасних САПР супроводжується текстовим способом опису. Утиліти, що входять до складу САПР забезпечують автоматичне пряме і зворотне перетворення описів.

Як правило, на будь-якому рівні ієрархії базовий набір елементів операційного блоку (цього рівня) доповнюється необхідним для його функціонування набором регістрів, логічних схем (багатофункціональних і керованих), буферних схем і зв’язків, що комутуються, між ними. Важливо, щоб на більш низьких ієрархічних рівнях опису проекту було однозначне трактування функціонування всіх елементів ОБ.

*Опис роботи керуючого блоку*

На цьому етапі визначається функціонування КБ, що забезпечує необхідну взаємодію елементів ОБ. Варто підкреслити, що два останніх етапи сильно взаємозалежні, і, якщо не розробляються паралельно, то виконуються шляхом ітерації.

Форми і засоби опису автомата різноманітні. Сучасна тенденція складається в переході від запису логічних виразів, обмежених правилами ТЗ, до графічної форми. Опис у вигляді граф-схем переходів (діаграми станів) стає одним з найпоширеніших варіантів завдання автоматів. Графічні редактори для створення автоматів включаються до складу засобів завдання вихідних проектів сучасних САПР.

*Компіляція проекту*

Після складання проекту і всіх його частин можна приступати до самого відповідального етапу проектування – компіляції проекту. Копміляції може підлягати, як увесь проект, так і його складові частини. Компіляція окремих фрагментів з однієї сторони спрощує проектування, оскільки зменшує розмір проблеми, що аналізується, а з іншої сторони вимагає враховувати різноманітні способи функціонування внутрішніх ресурсів і зовнішніх елементів. При компіляції виявляються всі сховані помилки. Компіляція розбивається на ряд послідовних підетапів: збір бази даних проекту, контроль з’єднань, логічна мінімізація проекту, формування завантажувального (конфігураційного) файлу й ін. На будь-якому етапі можуть виникати помилки, які потребують повторної компіляції після їх корекції [12].

*Верифікація проекту*

Тестування розробленоо пристрою або його фрагментів – один із найважливіших етапів проектування, оскільки практично не буває бездефектних проектів. Знаходження дефектів проекту – складна задача, її швидкість та якість залежить від досвіду розробника.

В сучасних САПР найбільш розповсюджене тестування шляхом роботи з редактором часових діаграм. Ці редактори поділяються на компілюючи та інтерпретуючі. Редактори інтерпретуючого типу дозволяють спростити процедуру відладки проектів і знайти їх дефекти, зв’язані з неправильною трактовкою розробником структурної чи поведінкової реалізації системи, чи особливостей реалізації елементної бази, що використовується.

Перевагами компілюючих систем моделювання є мінімізація часових затрат. При цьому проблемним залишається встановлення відповідності між рядками тексту і станами окремих сигналів [13].

*Визначення часових характеристик розробленого пристрою*

Сучасні САПР мають всередині себе повну інформацію про структуру проектованого пристрою і часових параметрів всіх компонентів – це дозволяє автоматизувати процес обчислення різноманітних часових характеристик проекту.

*Організація натурних експериментів*

Одним із заключних етапів проектування є етап експериментальної перевірки спроектованого пристрою. При всій старанності виконання попередніх етапів завжди існує далеко не нульова ймовірність того, що в проекті маються дефекти, що можуть проявитися на етапі впровадження чи навіть штатного використання пристрою і спричинити за собою тяжкі наслідки.

*Підготовка до виробничого випуску*

Після успішного завершення натурних експериментів із прототипними або макетними зразком проектувальник повинен забезпечити випуск дослідної партії розробленого виробу. Найважливішою задачею при цьому є забезпечення якісного супроводу продукції, що випускається у формі ІС ПЛ.

### 3.3.2 Етапи розробки нового проекту

Процедуру розробки нового проекту від концепції до завершення можна представити в такий спосіб:

* Створення нового файлу проекту чи ієрархічної структури декількох файлів проекту з використанням різних редакторів розробки проекту в системі MAX+PLUS II, тобто графічного, текстового і сигнального редакторів;
* Задавання імені файлу проекту верхнього рівня як імені проекту;
* Призначення сімейства ПЛІС для реалізації проекту. Користувач може сам призначити конкретний пристрій чи надати це компілятору, для того щоб оцінити необхідні ресурси;
* Відкриття вікна компілятора і його запуск. За бажанням користувача можна підключити модуль відтворення часових затримок для створення файлу розведення, використовуваного при тестуванні часових параметрів;
* У випадку успішної компіляції можливі тестування і часовий аналіз;
* Програмування чи завантаження конфігурації виконується шляхом запуску модуля програмування з наступною вставкою пристрою в програмуючий адаптер програматора MPU.
* Вибір кнопки *Program* для програмування пристроїв з пам’яттю типу EPROM, або вибір кнопки *Configure* для завантаження конфігурації пристрою з пам’яттю типу SRAM (FLEX).

## Висновки до розділу

Усі сучасні методики проектування ЦП на базі складних програмувальних ВІС/НВІС основуються на застосуванні САПР. Правильний вибір САПР – найважливіша умова ефективного проектування і прискорення випуску родукції.

САПР забезпечує створення, зберігання і обробку моделей геометричних об'єктів і їх графічне зображення за допомогою комп'ютера.

Застосування САПР вимагає ефективних, наочних, керованих і контрольованих засобів опису проекту. Описати проектований пристрій можна різними способами, причому звичайно застосовуючи спосіб, придатний для опису проекту в цілому.

САПР «MAX + PLUS II» являє собою інтегроване середовище для розробки цифрових пристроїв на базі програмованих логічних інтегральних схем (ПЛІС) фірми «Altera» ( «Альтера») і забезпечує виконання всіх етапів, необхідних для випуску готових виробів:

* створення проектів пристроїв;
* синтез структур і трасування внутрішніх зв'язків ПЛІС;
* підготовку даних для програмування або конфігурації ПЛІС (компіляцію);
* верифікацію проектів (функціональне моделювання і часовий аналіз);  програмування або конфігурацію ПЛІС

# РОЗДІЛ 4

# РЕАЛІЗАЦІЯ АЛГОРИТМУ ДСТУ ГОСТ 28147 В РЕЖИМІ ПРОСТОЇ ЗАМІНИ ЗА ДОПОМОГОЮ ПРОГРАМИ MAX+PLUS II

## 4.1 Структурна та функціональна схема ДСТУ ГОСТ 28147 в режимі простої заміни

ДСТУ ГОСТ 28147-89 – стандарт симетричного шифрування, який введений в 1990 році, також є стандартом СНД. Повна назва – “ДСТУ ГОСТ 28147-89 Системи обробки інформації. Криптографічний захист. Алгоритм криптографічного перетворення.”

Алгоритм криптографічного перетворення призначений для апаратної або програмної реалізації, задовольняє криптографічним вимогам і за своїми можливостями не накладає обмежень на степінь секретності інформації, яка захищається.

ДСТУ ГОСТ 28147-89 є блочним шифром – різновид симетричного шифру. Особливість блочного шифру – обробка блоку декількох байт за одну ітерацію (як правило, 8 або 16). Як і більшість блочних шифрів, ДСТУ ГОСТ оснований на мережі Фейстеля. Мережа являє собою визначену багаторазово повторюючу (ітеровану) структуру, яка називається коміркою Фейстеля. При переході від однієї комірки до іншої змінюється ключ, причому вибір ключа залежить від конкретного алгоритму. Структурна схема алгоритму зображена на рис. 4.1.

Блок суматорів

вхід вихід

Блок накопичувачів

Мережа Фейстеля

КЗУ

Рис. 4.1 – Структурна схема алгоритму ДСТУ ГОСТ 28147

Операції шифрування та розшифрування на кожному етапі дуже прості і при визначеному допрацюванні співпадають, потребуючи тільки зворотнього порядку використаних ключів. Шифрування за допомогою даної конструкції легко реалізується як на програмному рівні, так і на апаратному, що забезпечує широкі можливості застосування. Функціональна схема алгоритму зображена на рисунку 4.2.

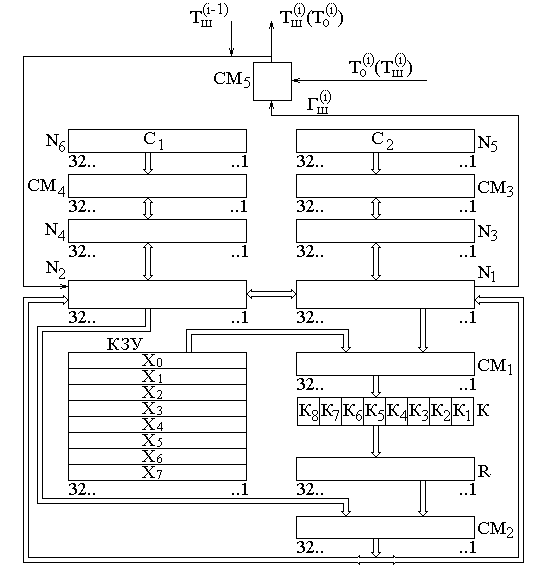


Рис. 3.1 – Функціональна схема алгоритму ДСТУ ГОСТ 28147

1) Кожен блок розбивається на два підблока (лівий і правий відповідно).

2) Вихідне заповнення правого блоку записується в лівий блок на виході.

3) Над правим блоком проводиться криптографічне перетворення із застосуванням ключових даних.

4) Лівий (вихідний) і правий (перетворений) блоки сумуються по модулю 2.

5) Отримана комбінація записується в правий блок на виході.

6) Така процедура повторюється декілька разів.

В ГОСТі ключова інформація складається з двох структур даних. Крім власного ключа, який необхідний для всіх шифрів, вона містить ще і таблицю замін. Нижче наведені основні характеристики ключових структур ГОСТу:

1. Таблиця замін К складається з восьми вузлів заміни , , , , , , , з пам’ятю на 64 біти кожний. 32-розрядний вектор, який поступає на блок підстановки, розбивається на вісім послідовно слідуючих 4-розрядних векторів, кожен з яких перетворюється в 4-розрядний вектор відповідним вузлом заміни, який представляє собою таблицю з шістнадцяти рядків, які містять по чотири біти заповнення в рядку. Вхідний вектор визначає адресу рядка в таблиці, заповнення даного рядка є вихідним вектором. Потім 4-розрядні вихідні вектори послідовно з’єднуються в 32-розрядний вектор.

Таким чином загальний об’єм таблиці замін дорівнює 512 біт (64 байта).

2. При додаванні і циклічному зсуві двійкових векторів старшими розрядами вважаються розряди накопичувачів з більшими номерами.

3. При записі ключа ( , в ключовий запам’ятовуючий пристрій (КЗП) значення вводиться в 1-й розряд накопичувача , значення вводиться в 2-й розряд накопичувача ,…, значення вводиться в 32-й розряд накопичувача , значення вводиться в 1-й розряд накопичувача і т.д. значення вводиться в 32-й розряд накопичувача .

4. При перезаписі інформації зміст *p*-го розряду одного накопичувача (суматора) переписується в *p*-й розряд другого накопичувача (суматора).

5. Ключі, що визначають заповнення КЗП і таблиць блоку підстановки К, є секретними елементами і поставляються в установленому порядку.

Ключ є масивом з восьми 32-бітних елементів коду в ГОСТі, елементи ключа використовуються як 32-розрядні цілі числа без знаку. Таким чином, розмір ключа складає 256 біт (32 байти). Ключ повинен бути масивом статично незалежних бітів, які приймають з однаковою імовірністю значення 0 та 1. При цьому деякі конкретні значення ключа можуть опинитися “слабкими”, тобто шифр може не забезпечувати заданий рівень стійкості у випадку їх використання. Однак імовірно частина таких значень в загальній масі всіх можливих ключів дуже мала. Тому ключі, вироблені за допомогою деякого датчика істинно випадкових чисел, будуть якісними з імовірністю від одиниці на дуже малу величину [14].

*Опис роботи алгоритму в режимі простої заміни*

Криптосхема, що реалізує алгоритм за шифрування в режимі простої заміни, повинна мати такий вигляд, як вказано на рис. 4.1.

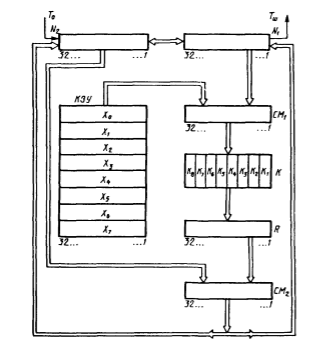


Рис. 4.1 – Схема роботи алгоритму ДСТУ ГОСТ 28147 в режимі простої заміни

*Процес зашифрування*

1) Відкриті дані розбиваються на блоки по 64 біта.

2) Далі выдбувається введення першого блоку в накопичувачі N1 і N2. При цьому біти відкритої інформації вводяться наступним чином: 1-й біт відкритої інформації - в 1-й розряд накопичувача N1, ..., 32-й - в 32-й розряд накопичувача N1, 33-й - в 1-й розряд накопичувача N2 і так далі, поки 64-й біт відкритої інформації не буде введений в 32-й розряд накопичувача N2.

3) В КЗУ вводиться ключ довжиною 256 біт.

*Проводиться зашифрування відкритих даних в режимі простої заміни (в 32 цикли):*

1) У першому циклі вміст регістра N1 підсумовується з заповненням з КЗУ по модулю в суматорі СМ1;

2) У блоці підстановки K проводиться заміна 32 біт інформації, що надійшла з сумматорf СМ1;

3) У регістрі зсуву R здійснюється циклічний зсув на 11 в сторону старшого розряду.

4) Інформація з регістра зсуву R і накопичувача N2 підсумовується по модулю 2 в суматорі СМ2.

5) Старе заповнення накопичувача N1 переписується в накопичувач N2.

6) Результат з виходу суматора СМ2 переписується в накопичувач N1.

Перший цикл закінчується.

Наступні цикли аналогічні першому, з тією лише відмінністю, що в 2-му циклі вводиться ключ , в 8-му -, в 9-м - і так далі в тому ж порядку до 24 циклу. З 25 по 32 цикл ключ вводиться в зворотному порядку: - в 25-му, - в 32-му.

Після 32-го циклу в N1 інформація зберігається, а ось результат з виходу суматора СМ2 переписується в N2.

Заповнення N1 і N2 і є перший блок зашифрованих даних.

Наступні блоки зашифровуються аналогічно.

## 4.2. Програмний код пристрою та епюри функціонування

Програмний код реалізації пристрою криптографічного перетворення написане мовою AHDL.

AHDL ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0) *Altera Hardware Definition Language*) — [комп'ютерна мова](https://uk.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B2%D0%B8_%D0%BE%D0%BF%D0%B8%D1%81%D1%83_%D0%B0%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D1%83%D1%80%D0%B8) опису [апаратних засобів](https://uk.wikipedia.org/wiki/%D0%90%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F), розроблена компанією [Altera](https://uk.wikipedia.org/wiki/Altera), що призначена для опису [комбінаційних логічних пристроїв](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B1%D1%96%D0%BD%D0%B0%D1%86%D1%96%D0%B9%D0%BD%D0%B0_%D0%BB%D0%BE%D0%B3%D1%96%D0%BA%D0%B0), [цифрових автоматів](https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D1%96%D1%8F_%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%D0%B0%D1%82%D1%96%D0%B2) і [таблиць істинності](https://uk.wikipedia.org/wiki/%D0%A2%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D1%96_%D1%96%D1%81%D1%82%D0%B8%D0%BD%D0%BD%D0%BE%D1%81%D1%82%D1%96) з врахуванням архітектурних особливостей [ПЛІС](https://uk.wikipedia.org/wiki/%D0%9F%D0%9B%D0%86%D0%A1) цієї фірми. Мова має [Ada](https://uk.wikipedia.org/wiki/Ada)-подібний [синтаксис](https://uk.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BD%D1%82%D0%B0%D0%BA%D1%81%D0%B8%D1%81_%D0%BC%D0%BE%D0%B2%D0%B8_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F).

*Subdesign Section* (опис модуля) – задає ім’я модуля й перераховує його порти. Починається з ключового слова SUBDESIGN, за яким слідує назва модуля та список виводів в дужках.

*Variable Section* (розділ змінних) – задає внутрішні зміни. Починається з ключового слова VARIABLE, за яким слідують імена однотипних змінних, а далі тип змінних.

*Logic Section* (опис логіки) – описує функціонал модуля. Починається з ключового слова BEGIN, закінчується ключовим словом END [15 c. 160].

*Множина символів мови*

До [множини](https://uk.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B6%D0%B8%D0%BD%D0%B0) символів мови відносять букви, цифри та спеціальні символи, призначення яких наведено в таблиці нижче.

|  |  |
| --- | --- |
| -- | Коментар в стилі [VHDL](https://uk.wikipedia.org/wiki/VHDL), що продовжується до кінця рядка |
| % | Ставляться з обох боків коментаря в стилі AHDL |
| () | Використовуються для підвищення пріоритету в операціях, оточують порти в іменах модуля та списки параметрів |
| [] | Використовуються для створення діапазону значень |
| '' | Використовуються для створення символічних імен |
| "" | Використовуються в заголовку, для вказування файлу для включення, оточують цифри в недесяткових константах |
| . | Відділяє ім'я файлу від [розширення](https://uk.wikipedia.org/wiki/%D0%A0%D0%BE%D0%B7%D1%88%D0%B8%D1%80%D0%B5%D0%BD%D0%BD%D1%8F_%D1%84%D0%B0%D0%B9%D0%BB%D1%83) |
| .. | Розділяє найбільші та найменші значення в діапазонах |
| ; | Закінчує розділи і секції мови |
| , | Розділяє елементи списків та груп |
| : | Відділяє символьні імена від типів |
| = | Присвоює значення в операціях |
| => | Відділяє входи і виходи в таблицях істинності та операторі CASE |

*Код програми реалізації алгоритму ДСТУ ГОСТ 28147*

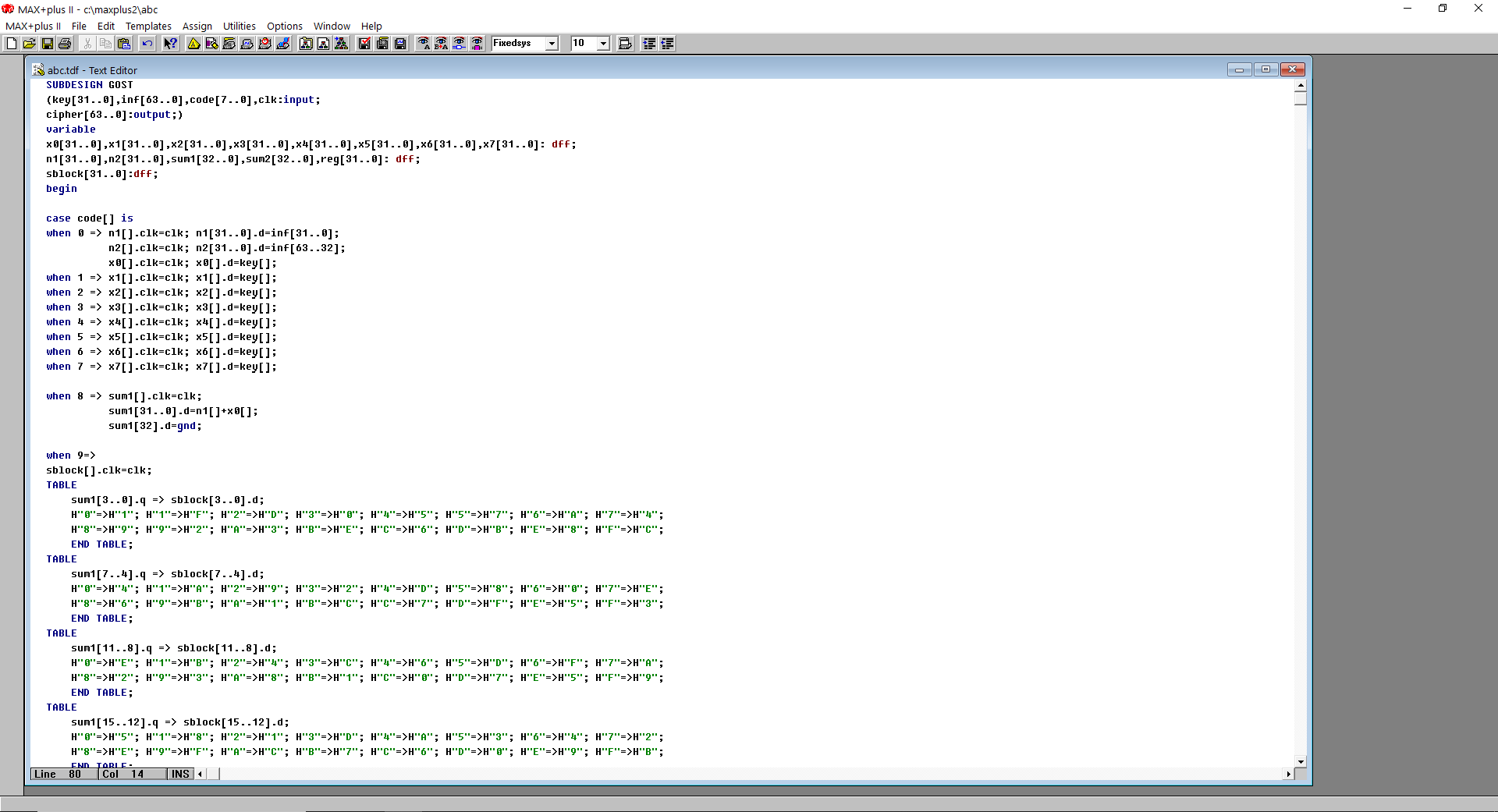


Рис. 4.2 – Приклад програмного вікна MAX+PLUS II з кодом.

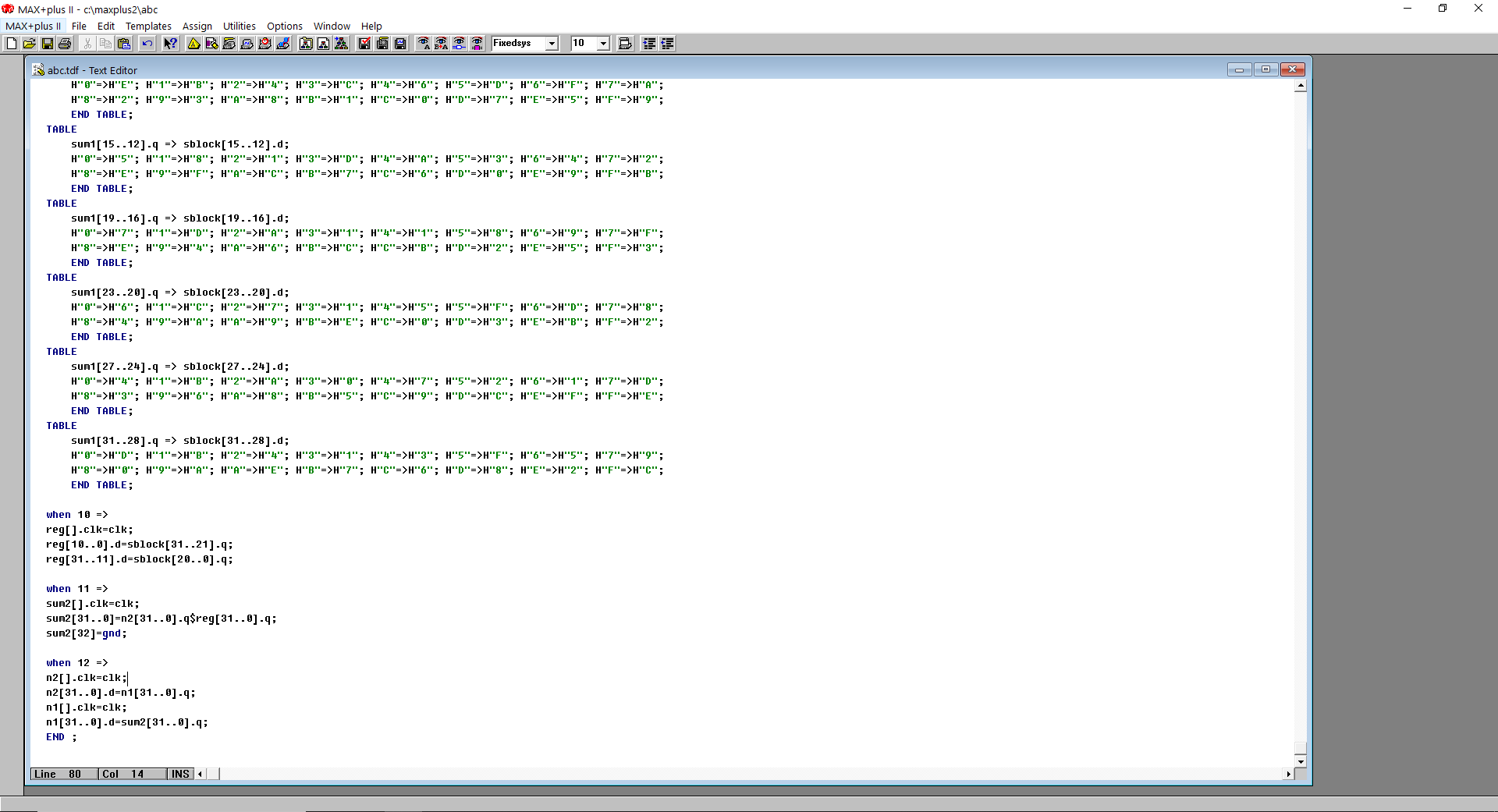


Рис. 4.3 – Приклад програмного вікна MAX+PLUS II з кодом.

Вище наведений програмний код лише для перших восьми циклів зашифрування, так як всі інші цикли виконуються по аналогії.

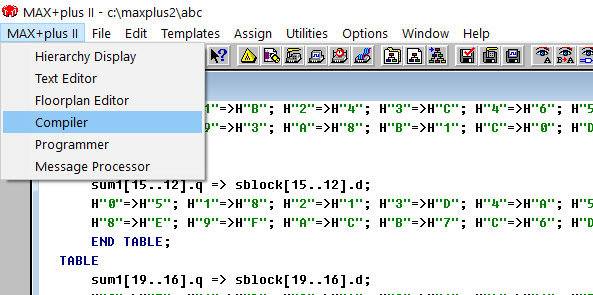


Рис. 4.4 – Режим компіляції.

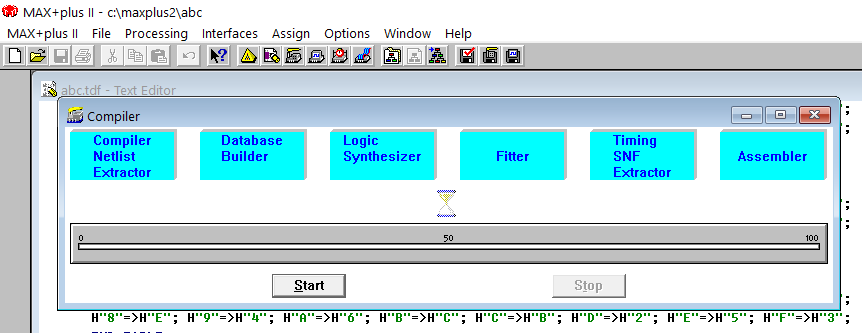


Рис. 4.5 – Статус компіляції.

## Висновки до розділу

Відзначимо переваги і недоліки режиму простої заміни алгоритму ДСТУ ГОСТ 28147-89.

***Переваги:***

1)Виключається вплив перекриття шифру на якість шифрування.

2)Простота синхронізації.

***Недолік:***

1)Одна помилка типу «заміна знака» в криптограмі веде за собою повне руйнування блоку при розшифрування.

# ВИСНОВКИ

Проаналізувавши загрози в інформаційній безпеці можна чітко сказати, що на сьогоднішній день актуальною проблемою є вплив ЗМІ на суспільство. Використовуючи різні засоби масової інформації, а також мережу інтернет, відбуваються постійні інформаційні атаки, до яких сучасне суспільство практично не підготовлене. Україна зараз страждає від інформаційної війни. Вона використовує і управляє інформацією з метою набуття конкурентоздатної переваги над противником. Сьогодні інформація набирає матеріальної форми і володіння нею стає дуже жаданим. Відповідно і забезпечувати інформаційну безпеку стає все важче.

Для забезпечення належного рівня захищеності інформації потрібна криптографічна система (криптосистема) сукупність засобів криптографічного захисту, необхідної ключової, нормативної, експлуатаційної, а також іншої документації (зокрема й такої, що визначає заходи безпеки).

В Україні для шифрування даних, які складають військову або комерційну таємницю, дозволений алгоритм ДСТУ ГОСТ 28147-89. Цей алгоритм достатньо надійний, так як використовується він протягом багатьох років і не був взламаний, а також даний алгоритм використовує ключ довжиною 256 біт, що означає, що варіант добору ключа перебором майже неможливий, так як це займе дуже багато часу.

В силу того, що техніка зв’язку, яка в теперішній час знаходиться на озброєнні має багато застарілого обладнання, яке не зручне у використанні через велику масогабаритність та через те, що вона часто виходить з ладу, тематика дипломної роботи є актуальною.

Необхідно створювати власне обладнання, що відповідає криптографічним та технічним вимогам із захисту інформації, але, нажаль, власна елементна база відсутня. Тому доцільним є реалізація систем криптографічних перетворень на закордонній елементній базі, а саме на програмованих логічних інтегральних схемах, які на цей час дуже актуальні у всьому світі.

Даний напрямок актуальний в наш час, так як швидкості передачі інформації постійно зростають, а необхідність в її зашифруванні залишається, системи шифрування, реалізовані програмно з використанням процесорів або мікропроцесорів часто не справляються з таким навантаженням.

В ході виконання завдання до кваліфікаційної роботи був проведений загальний теоретичний аналіз системи автоматизованого проектування MAX+PLUS II, проведено аналіз алгоритму шифрування ДСТУ ГОСТ 28147.

В розділах дипломної роботи вказані загальні та основні відомості по роботі з програмованими логічними матрицями, середовищем автоматизованого проектування MAX+PLUS II, в якому реалізовано алгоритм шифрування відповідно ДСТУ ГОСТ 8147:89 для подальшого їх програмування на ПЛМ.

# СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Богуш В. М., Кусин А. М. Інформаційна безпека від А до Я. – К.: МОУ, 1999.–456 с.

2. Алишов Н.И. Организации безопасности информационных ресурсов в системах телекоммуникаций // Праці 4-ї Міжнародної науково-техничної конференції по телекомунікаціям ―Телеком-99. – Одеса , 1999. – С. 112-115.

3. Левин В.К., Гайкович В.Ю., Дорошкевич П.В. и др. Информационная безопасность компьютерных сетей.// Технологии электронных коммуникаций. – М. 1993. т.5.–128 с.

4. Алферов А.П., Зубов А.Ю., Кузьмин А.С., Черемушкин А.В. Основы криптографии: Учеб. пособие. – М.: Гелиос АРВ, 2002. – 480 с.

5. Бабаш А. В., Шанкин Г. П. Криптография. - М.: СОЛОН-Р, 2002. - 512 с.

6. Бабичев С.Г., Гончаров В.В., Серов Р.Е. Основы современной криптографии: Учеб. курс. – М.: Горячая линия – Телеком, 2002. – 175 с.

7. Соловьев В.В., Васильев А.Г. Программируемые логические интегральные схемы и их применение. – Мн.: Беларуская наука, 1998. – 270 с.

8. Домрачев В.Г., Мальцев П.П., Новаченко И.В., Пономарьев С.Н. Базовые матричные кристалы и матричные ВИС. – М.: Энергоиздат, 1992. – 224с.

9. Норенков И.П., Маничев В.Б. Основы теории и проектирования САПР: Учеб. Для вузов по спец. “Вычислительные маш., компл., сист. и сети” – М.: Высш. Шк., 1990. – 335 с.

10. Скрипник Л.В., Корнейко О.В., Кулініч О.М., Ярманов С.С. Основи проектування засобів захисту інформації на основі інтегральних схем зі структурою, що програмується: Навчальний посібник / – К.: 2005, – 252с., 57 іл.

11. Горбенко І.Д. Прикладна криптологія: Теорія. Практика. Застосування:

монографія / І.Д. Горбенко, Ю.І. Горбенко. – Харків.: Вид-во «Форт», 2012. – 880 с.

12. Грушвицкий Р.И., Михайлов М.М. Проектирование в условиях временных ограничений: отладка проектов//Компоненты и технологии, 2007. - № 6, 8, 9.

13. Грушвицкий Р.И., Шашкин П.М. Проектирование в условиях временных ограничений: компиляция проектов//Компоненты и технологии, 2011. - № 1-3.

14. Грушвицкий Р.И., Шашкин П.М. Проектирование в условиях временных ограничений: верификация проектов//Компоненты и технологии, 2008. - № 3, 5, 6, 8.

15. ГОСТ 28147–89 / Группа П85 / Государственный стандарт Союза ССР / Системы обработки информации. Защита криптографическая / Алгоритм криптографического преобразования / ОКП 40 4000 / Дата введения: 07.01.1990

16. Антонов А.П. Язык описания цифровых устройств AlteraHDL. Практический курс. – М.:ИП РадиоСофт, 2001. – 224с.

17. А.Л. ЧМОРА. Криптография. Москва. «Гелиос АРВ», 2002. – 244 с.

18. А.А. Болотов, С.Б. Гашков, А.Б. Фролов, А.А. Часовских, "Алгоритмические основы эллиптической криптографии". Учебное пособие. М.: Изд-во МЭИ. 2000

19. Овчинников С.А., Гришин С.Е. Формирование культуры кибербезопасности в обществе – актуальная задача современности //Вестник СГСЭУ. 2011. № 3 (37). 206;

20. Цуканова О.А., Смирнов С.Б. Экономика защиты информации: Учебное пособие. – СПб.: СПб ГУИТМО, 2007. – 59 с.;

21. Астахов А.М. Искусство управления информационными рисками / Астахов А.М. – М.: ДМК Пресс,. 2010. – 312 с.;

22. Щербаков, А.Ю. Современная компьютерная безопасность. Теоретические основы. Практические аспекты. М.: Книжный мир, 2009. -352 с.;

23. Малюк А.А. Информационная безопасность: концептуальные и методологические основы защиты информации. Учеб. Пособие для вузов.-М.: Горячая линия-Телеком. -2008.- 280 с.;

24. Шаньгин, В.Ф. Защита компьютерной информации. Эффективные методы и средства. М.: ДМК Пресс, 2008. - 544 с.;

25. Гладиш С. В. Формування вимог щодо безпеки державних інформаційних ресурсів у телекомунікаційній мережі загального користування // Правове, нормативне та метрологічне забезпечення системи захисту інформації в Україні. — 2012. — № 14. — С. 33– 40.;

26. НД ТЗІ 2.5-004- 99 «Критерії оцінки захищеності інформації в комп’ютерних системах від несанкціонованого доступу»;

27. Хорев А.А. Защита информации от утечки по техническим каналам. Часть 1. Технические каналы утечки информации. Учебное пособие. М.: Гостехкомиссия России, 2010. - 320 с.;

29. Дубов Д. В. Кібербезпека : світові тенденції та виклики для України / Д. В. Дубов, М. А. Ожеван. – К. : НІСД, 2011. – 30 с

30. Фурашев В.М. Кіберпростір та інформаційний простір, кібербезпека та інформаційна безпека: сутність, визначення, відмінності //Інформація і право. – 2012. – № 2. – С. 162-169;

31. Сакович Л.М., Політов В.І. Використання системи підтримки прийняття рішення під час експлуатації та ремонту засобів і комплексів зв’язку // Зв’язок. — 2012. — № 5. — С. 37–39;

32. Юдін О.К. Кодування в інформаційно-комунікаційних мережах: – Монографія. – К.: НАУ, 2007. – 308с.

# Додаток А

SUBDESIGN GOST

(key[31..0],inf[63..0],code[7..0],clk:input;

cipher[63..0]:output;)

variable

x0[31..0],x1[31..0],x2[31..0],x3[31..0],x4[31..0],x5[31..0],x6[31..0],x7[31..0]: dff;

n1[31..0],n2[31..0],sum1[32..0],sum2[32..0],reg[31..0]: dff;

sblock[31..0]:dff;

begin

case code[] is

when 0 => n1[].clk=clk; n1[31..0].d=inf[31..0];

n2[].clk=clk; n2[31..0].d=inf[63..32];

x0[].clk=clk; x0[].d=key[];

when 1 => x1[].clk=clk; x1[].d=key[];

when 2 => x2[].clk=clk; x2[].d=key[];

when 3 => x3[].clk=clk; x3[].d=key[];

when 4 => x4[].clk=clk; x4[].d=key[];

when 5 => x5[].clk=clk; x5[].d=key[];

when 6 => x6[].clk=clk; x6[].d=key[];

when 7 => x7[].clk=clk; x7[].d=key[];

when 8 => sum1[].clk=clk;

sum1[31..0].d=n1[]+x0[];

sum1[32].d=gnd;

when 9=>

sblock[].clk=clk;

TABLE

sum1[3..0].q => sblock[3..0].d;

H"0"=>H"1"; H"1"=>H"F"; H"2"=>H"D"; H"3"=>H"0"; H"4"=>H"5"; H"5"=>H"7"; H"6"=>H"A"; H"7"=>H"4";

H"8"=>H"9"; H"9"=>H"2"; H"A"=>H"3"; H"B"=>H"E"; H"C"=>H"6"; H"D"=>H"B"; H"E"=>H"8"; H"F"=>H"C";

END TABLE;

TABLE

sum1[7..4].q => sblock[7..4].d;

H"0"=>H"4"; H"1"=>H"A"; H"2"=>H"9"; H"3"=>H"2"; H"4"=>H"D"; H"5"=>H"8"; H"6"=>H"0"; H"7"=>H"E";

H"8"=>H"6"; H"9"=>H"B"; H"A"=>H"1"; H"B"=>H"C"; H"C"=>H"7"; H"D"=>H"F"; H"E"=>H"5"; H"F"=>H"3";

END TABLE;

TABLE

sum1[11..8].q => sblock[11..8].d;

H"0"=>H"E"; H"1"=>H"B"; H"2"=>H"4"; H"3"=>H"C"; H"4"=>H"6"; H"5"=>H"D"; H"6"=>H"F"; H"7"=>H"A";

H"8"=>H"2"; H"9"=>H"3"; H"A"=>H"8"; H"B"=>H"1"; H"C"=>H"0"; H"D"=>H"7"; H"E"=>H"5"; H"F"=>H"9";

END TABLE;

TABLE

sum1[15..12].q => sblock[15..12].d;

H"0"=>H"5"; H"1"=>H"8"; H"2"=>H"1"; H"3"=>H"D"; H"4"=>H"A"; H"5"=>H"3"; H"6"=>H"4"; H"7"=>H"2";

H"8"=>H"E"; H"9"=>H"F"; H"A"=>H"C"; H"B"=>H"7"; H"C"=>H"6"; H"D"=>H"0"; H"E"=>H"9"; H"F"=>H"B";

END TABLE;

TABLE

sum1[19..16].q => sblock[19..16].d;

H"0"=>H"7"; H"1"=>H"D"; H"2"=>H"A"; H"3"=>H"1"; H"4"=>H"1"; H"5"=>H"8"; H"6"=>H"9"; H"7"=>H"F";

H"8"=>H"E"; H"9"=>H"4"; H"A"=>H"6"; H"B"=>H"C"; H"C"=>H"B"; H"D"=>H"2"; H"E"=>H"5"; H"F"=>H"3";

END TABLE;

TABLE

sum1[23..20].q => sblock[23..20].d;

H"0"=>H"6"; H"1"=>H"C"; H"2"=>H"7"; H"3"=>H"1"; H"4"=>H"5"; H"5"=>H"F"; H"6"=>H"D"; H"7"=>H"8";

H"8"=>H"4"; H"9"=>H"A"; H"A"=>H"9"; H"B"=>H"E"; H"C"=>H"0"; H"D"=>H"3"; H"E"=>H"B"; H"F"=>H"2";

END TABLE;

TABLE

sum1[27..24].q => sblock[27..24].d;

H"0"=>H"4"; H"1"=>H"B"; H"2"=>H"A"; H"3"=>H"0"; H"4"=>H"7"; H"5"=>H"2"; H"6"=>H"1"; H"7"=>H"D";

H"8"=>H"3"; H"9"=>H"6"; H"A"=>H"8"; H"B"=>H"5"; H"C"=>H"9"; H"D"=>H"C"; H"E"=>H"F"; H"F"=>H"E";

END TABLE;

TABLE

sum1[31..28].q => sblock[31..28].d;

H"0"=>H"D"; H"1"=>H"B"; H"2"=>H"4"; H"3"=>H"1"; H"4"=>H"3"; H"5"=>H"F"; H"6"=>H"5"; H"7"=>H"9";

H"8"=>H"0"; H"9"=>H"A"; H"A"=>H"E"; H"B"=>H"7"; H"C"=>H"6"; H"D"=>H"8"; H"E"=>H"2"; H"F"=>H"C";

END TABLE;

when 10 =>

reg[].clk=clk;

reg[10..0].d=sblock[31..21].q;

reg[31..11].d=sblock[20..0].q;

when 11 =>

sum2[].clk=clk;

sum2[31..0]=n2[31..0].q$reg[31..0].q;

sum2[32]=gnd;

when 12 =>

n2[].clk=clk;

n2[31..0].d=n1[31..0].q;

n1[].clk=clk;

n1[31..0].d=sum2[31..0].q;