**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ КАФЕДРА КОМП’ЮТЕРИЗОВАНИХ СИСТЕМ ЗАХИСТУ ІНФОРМАЦІЇ**

ДОПУСТИТИ ДО ЗАХИСТУ

Завідувач кафедри

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

«\_\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_ 20\_\_ р.

На правах рукопису

УДК 004.056.5:510.22(043.3)

**МАГІСТЕРСЬКА АТЕСТАЦІЙНА РОБОТА**

**ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ**

**«МАГІСТР»**

**Тема: Програмно-апаратна реалізація алгоритму блокового шифрування ДСТУ 7624 : 2014**

**Автор: А.О. Гордієнко**

**Науковий керівник: к.т.н., доц. О.М. Кулініч**

**Нормоконтролер: асист.**

**Київ 2020**

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ 6

ВСТУП 7

РОЗДІЛ 1 ЗАГАЛЬНІ ВІДОМОСТІ ПРО ІНФОРМАЦІЮ ТА ЇЇ ЗАХИСТ 12

1.1 Основні поняття та визначення 12

1.2 Технічний захист інформації 18

1.3 Криптографічний захист інформації та його методи 19

1.4 Висновки до розділу 23

РОЗДІЛ 2 ПРОЕКТУВАННЯ ЦИФРОВИХ ПРИСТРОЇВ НА ПРОГРАМУЮЧИХ ЛОГІЧНИХ МАТРИЦЯХ В СИСТЕМІ MAX+PLUS II 24

2.1 Області застосування та класифікація ПЛІС 24

2.2 Основні поняття про системи автоматизованого проектування 28

2.3 Загальні відомості про систему MAX+PLUS II 35

2.4 Функціонування редакторів системи MAX + PLUS II 31

2.5 Розробка проектів цифрових пристроїв на мові AHDL 37

2.6 Висновки до розділу 38

РОЗДІЛ 3 АНАЛІЗ АЛГОРИТМУ ШИФРУВАННЯ ДСТУ 7624:2014 39

3.1 Означення алгоритму блокового шифрування 39

3.2 Опис блокового шифру ДСТУ 7624:2014 “Калина” 42

3.3 Криптографічна стійкість блокового шифру “Калина” 62

3.4 Висновки до розділу 66

РОЗДІЛ 4 РЕАЛІЗАЦІЯ АЛГОРИТМУ ДСТУ 7624:2014 НА ПЛІС 68

4.1 Принципова схема та початкове заповнення шифра 68

4.2 Перетворення XORRoundKey та AddRoundKey для 128 бітного блоку 70

4.3 Перетворення SubBytes 72

4.4 Перетворення ShiftRows 76

4.5 Перетворення MixColumns 78

4.6 Висновки до розділу 88

ВИСНОВКИ 89

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 90

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

АС – автоматизована система

БСШ – блоковий симетричний шифр

ЗІ – захист інформації

ЗП – запам’ятовуючий пристрій

ІзОД – інформація з обмеженим доступом

ІБ – інформаційна безпека

ІС – інтегральна схема

ІТС – інформаційно-телекомунікаційна система

КЗІ – криптографічний захист інформації

КТЗІ – комплекс технічного захисту інформації

ОІД – об’єкт інформаційної діяльності

ПЛІС – програмована логічна інтегральна схема

ПЛМ – програмована логічна матриця

ПМЛ – програмована матрична логіка

САПР – система автоматизованого проектування

ТЗІ – технічний захист інформації

# ВСТУП

Проблема національної безпеки України є особливо актуальною як сьогодні, так і в контексті подальшого загального цивілізаційного розвитку країни. Тому забезпечення безпеки інформаційно-телекомунікаційних систем стало однією з пріоритетних завдань сучасного світу. Необхідною умовою для цього є застосування криптографічного захисту. Користувачі глобальних та локальних мереж потребують простих і водночас потужних засобів захисту інформації. Сучасні стандарти шифрування розроблені з урахуванням особливостей середовища, у якому вони мають застосовуватися. Звідси постає актуальне питання аналізу використовуваних алгоритмів для визначення найбільш криптостійких або тих, що мають найвищу швидкодію та забезпечують високий рівень безпеки інформаційних систем.

**Актуальність:** за результатами проведення відкритого конкурсу симетричних блокових криптографічних алгоритмів ДССЗЗІ України ініціювала створення вітчизняного алгоритму шифрування, який дозволяє забезпечувати високий рівень стійкості із необхідним запасом для застосування протягом декількох десятків років, але постала проблема реалізації даного алгоритму. Враховуючи це, в даній роботі описано новий та перспективний алгоритм шифрування, який здатний прийти на зміну ГОСТ 28147:2009 та передбачає можливість одночасного забезпечення конфіденційності та цілісності повідомлення шляхом послідовного застосування відповідних перетворень, що забезпечують більший рівень криптографічної стійкості, високу швидкодію та якісний рівень захисту від атак. Даний стандарт розроблено з урахуванням існуючих та потенційних загроз, подальшого інтенсивного розвитку інформаційних технологій та необхідності активного використання протягом кількох наступних десятиліть. Основними перевагами шифру “Калина” порівняно з іншими міжнародними аналогами є можливість застосовувати блок даних і ключ шифрування розміром до 512 біт та принципово нова схема створення підключів, що забезпечують захист від усіх відомих атак на схеми їх створення. Тому введення в дію даного стандарту дозволить суттєво удосконалити показники ефективності систем захисту, засобів і протоколів криптографічного захисту інформації, які розробляються в Україні, а в деяких випадках поліпшити їх порівняно з існуючими та перспективними світовими практиками.

З появою нового національного стандарту ДСТУ 7624:2014 актуальною задачею є його оцінка з точки зору перспективи реалізації на найпоширеніших структурах з програмуємою логікою.

**Мета роботи:** показати можливість реалізації криптографічного перетворення ДСТУ 7624:2014 на сучасній елементній базі.

Для досягнення поставленої мети в роботі вирішуються наступні завдання:

* розглянути систему захисту інформації та методи криптографічного захисту інформації (РОЗДІЛ 1);
* дослідити середовище проектування MAX + plus II, яке необхідне для реалізації практичного завдання (РОЗДІЛ 2);
* розглянути вітчизняний алгоритм шифрування ДСТУ 7624:2014, його вимоги до розробки та особливості побудови (РОЗДІЛ 3);
* реалізувати основні перетворення SubBytes, ShiftRows, MixColumns, XoRRoundKey, AddRoundKey (РОЗДІЛ 4).

**Практична цінність:** даної роботи полягає в тому, що завдяки результатам, які отримані при реалізації ДСТУ 7624:2014 з’являється можливість і надалі застосовувати ці дані в переході до більш сучасних засобів криптографічного захисту інформації. Даний алгоритм був розроблений Адміністрацією Держспецзв’язку та провідними науковцями нашої держави задля забезпечення інформаційної безпеки України, тому проектування даного БСШ дає змогу вийти Україні на вищий рівень у сфері забезпечення захисту інформації.

**Галузь застосування**. можливості реалізації в програмно-апаратному вигляді блокового алгоритму шифрування “Калина”

**Об’єкт дослідження**- процес криптографічного перетворення на основі ДСТУ 7624:2014.

**Предмет дослідження**– спосіб реалізації ДСТУ “Калина” на програмованих логічних матрицях (ПЛМ).

**Методи дослідження**  Програмно-апаратнареалізація алгоритму блокового шифрування ДСТУ 7624 : 2014задопомогую MAX+PLUS II

**Новизна одержаних результатів полягає в наступному:** На сьогоднішній день відсутня інформація щодо реалізації ДСТУ в апаратному чи програмно-апаратному виді. Тому новизна заключається у використанні програмованих Логічних матриць (ПЛМ) для реалізації ДСТУ “Калина”, та намагання реалізувати на практиці ДСТУ “Калина” у програмовано-апаратному вигляді в сучасних умовах та в демонстрації можливості використання програмованих логічних матриць (ПЛМ) для даного алгоритму.

# 

# РОЗДІЛ 1

# ЗАГАЛЬНІ ВІДОМОСТІ ПРО ІНФОРМАЦІЮ ТА ЇЇ ЗАХИСТ

В даному розділі розглянуто особливості технічного та криптографічного захисту інформації, наведено основні поняття на які опирається і на чому базується захист інформації. Також розмежовано поняття симетричної та асиметричної криптографії.

* 1. Основні поняття та визначення

Термін інформація походить від information, що означає роз’яснення, повідомлення, виклад. З огляду філософії інформація є відображення реального світу за допомогою відомостей (повідомлень).

Існує велика кількість визначень інформації, але будемо використовувати наступні:

Інформація (згідно Закону України “Про інформацію”) – це документовані або публічно оголошені відомості про події та явища, що відбуваються у суспільстві, державі та навколишньому природному середовищі [1].

Стаття 7. **(**ЗАКОН УКРАЇНИ Про інформацію**).**Охорона права на інформацію

1. Право на інформацію охороняється законом. Держава гарантує всім суб'єктам інформаційних відносин рівні права і можливості доступу до інформації [1].

2. Ніхто не може обмежувати права особи у виборі форм і джерел одержання інформації, за винятком випадків, передбачених законом [1].

Суб'єкт інформаційних відносин може вимагати усунення будь-яких порушень його права на інформацію [1].

3. Забороняється вилучення і знищення друкованих видань, експонатів, інформаційних банків, документів з архівних, бібліотечних, музейних фондів, крім встановлених законом випадків або на підставі рішення суду [1].

4. Право на інформацію, створену в процесі діяльності фізичної чи юридичної особи, суб'єкта владних повноважень або за рахунок фізичної чи юридичної особи, Державного бюджету України, місцевого бюджету, охороняється в порядку, визначеному законом [1].

Стаття 15**.**Науково-технічна інформація

1. Науково-технічна інформація - будь-які відомості та/або дані про вітчизняні та зарубіжні досягнення науки, техніки і виробництва, одержані в ході науково-дослідної, дослідно-конструкторської, проектно-технологічної, виробничої та громадської діяльності, які можуть бути збережені на матеріальних носіях або відображені в електронному вигляді [1].

2. Правовий режим науково-технічної інформації визначається Законом України  "Про науково-технічну інформацію", іншими законами та міжнародними договорами України, згода на обов'язковість яких надана Верховною Радою України [1].

3. Науково-технічна інформація є відкритою за режимом доступу, якщо інше не встановлено законами України [1].

Стаття 21**.** Інформація з обмеженим доступом

1. Інформацією з обмеженим доступом є конфіденційна, таємна та службова інформація [1].

2. Конфіденційною є інформація про фізичну особу, а також інформація, доступ до якої обмежено фізичною або юридичною особою, крім суб'єктів владних повноважень. Конфіденційна інформація може поширюватися за бажанням (згодою) відповідної особи у визначеному нею порядку відповідно до передбачених нею умов, а також в інших випадках, визначених законом [1].

У науковій літературі використовуємо інше визначення. Інформація – це відомості про осіб, предмети, технології, засоби, ресурси, події та явища, що відбуваються в усіх сферах діяльності держави, життя суспільства та в довкіллі, незалежно від форми їх надання. Відомості можуть бути представлені у вигляді сигналів, знаків, зображень чи в інший спосіб [2].

До основних властивостей інформації належать:

* конфіденційність – властивість інформації бути захищеною від несанкціонованого ознайомлення;
* цілісність – властивість інформації бути захищеною від несанкціонованого спотворення, руйнування або знищення;
* доступність – здатність інформації бути захищеною від несанкціонованого блокування;
* достовірність – здатність інформації об’єктивно відображати процеси та явища, що відбуваються в навколишньому світі.

Всі сфери життєдіяльності людини напряму залежать від інформаційного ресурсу, який має свою цінність та вимагає захисту від різних впливів, що можуть призвести до знищення або порушення цілісності. В наш час інформація перебуває у постійному та неухильному зростанні, що потребує стабільного та постійного захисту.

Стаття 17. Захист суверенітету і територіальної цілісності України, забезпечення її економічної та інформаційної безпеки є найважливішими функціями держави, справою всього українського народу.

Загроза

Об’єкт

(суб’єкт)

Захист

Безпека об’єкту

Рисунок 1.1 - Модель безпеки об’єкту

Сучасне [суспільство](http://ua-referat.com/%D0%A1%D1%83%D1%81%D0%BF%D1%96%D0%BB%D1%8C%D1%81%D1%82%D0%B2%D0%BE) все більше стає інформаційно-обумовленим, успіх будь-якого виду діяльності все сильніше залежить від володіння певними відомостями і від відсутності їх у конкурентів. І чим сильніше проявляється зазначений ефект, тим більше потенційні [збитки](http://ua-referat.com/%D0%97%D0%B1%D0%B8%D1%82%D0%BA%D0%B8) від зловживань в [інформаційній](http://ua-referat.com/%D0%86%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D1%8F) сфері, і тим більше потреба в захисті інформації. Проблема національної безпеки України є особливо актуальною як сьогодні, так і в контексті подальшого загально цивілізаційного розвитку країни. Забезпечення національної безпеки є однією з найголовніших функцій будь-якої держави.

Національна безпека – захищеність життєво важливих інтересів людини і громадянина, суспільства і держави, за якої забезпечується сталий розвиток суспільства, своєчасне виявлення, запобігання і нейтралізація реальних та потенційних загроз національним інтересам.

Закон України “Про основи національної безпеки України” визначає інформаційну безпеку важливою складовою національної безпеки. Важливість інформаційної безпеки пояснюється тим, що, по-перше, національні інтереси, загрози їм, управління цими загрозами реалізуються тільки через інформаційну сферу; по-друге, людина та її права, інформація та інформаційні системи та права на них – це основні об’єкти не тільки національної безпеки в інформаційній сфері, але й основні елементи всіх об’єктів безпеки в усіх галузях; по-третє, інформаційна складова є притаманною будь-якій сфері життєдіяльності суспільства. Вона забезпечується безпекою інформації .

Інформаційна безпека – це стан захищеності інформаційного середовища суспільства, який забезпечує його формування, використання і розвиток в інтересах громадян, організацій, держави [3].

Об’єктами інформаційної безпеки є людина, суспільство та держава, забезпечення їхніх інтересів є завданням інформаційної безпеки.

У відповідності законами України “Про інформацію”(1992р.), “Про державну таємницю” (1990р.) та “Про власність” інформація має наступну класифікацію (див. табл. 1.1).

У відповідності щодо приведеної класифікації в таблицею 1.1 за режимом доступу інформація поділяється на відкриту інформацію та інформацію з обмеженим доступом.

Інформація з обмеженим доступом (ІзОД) – інформація, що становить державну або іншу передбачену законодавством України таємницю, конфіденційна інформація, що є державною власністю чи передана державі у володіння, користування, розпорядження [4]. Інформація з обмеженим доступом в свою чергу за своїм правовим режимом поділяється на конфіденційну і таємну.

Конфіденційна інформація – це відомості, які знаходяться у володінні, користуванні або розпорядженні окремих фізичних чи юридичних осіб і поширюються за їх бажанням відповідно до передбачених ними умов.[4].

Таблиця 1.1 – Класифікація інформації

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Ознаки класифікації інформації | | За доступом | | |
| Інформація з обмеженим доступом (ІзОД) | | Відкрита інформація з вільним доступом до неї |
| Таємна інформація | Конфіденційна  (службова)  інформація |
| За формою власності | Інформація державної власності | +  Секретна інформація або державна таємниця та інша встановлена законом таємниця | +  Службова  інформація | + |
| Інформація приватної та колективної власності | –  (законом не передбачено) | +  Конфіденційна  інформація | + |

Державна таємниця (секретна інформація) – вид таємної інформації, що охоплює відомості у сфері оборони, економіки, науки і техніки, зовнішніх відносин, державної безпеки та охорони правопорядку, розголошення яких може завдати шкоди національній безпеці України та які визнані у порядку, встановленому цим Законом, державною таємницею і підлягають охороні державою [22].

Охорона державної таємниці - комплекс організаційно-правових, інженерно-технічних, криптографічних та оперативно-розшукових заходів, спрямованих на запобігання розголошенню секретної інформації та втратам її матеріальних носіїв [22].

Режим секретності - встановлений згідно з вимогами цього Закону та інших виданих відповідно до нього нормативно-правових актів єдиний порядок забезпечення охорони державної таємниці [22].

Секретна інформація в свою чергу має три ступеня секретності: "особливої важливості", "цілком таємно", "таємно" – категорії, які характеризують важливість секретної інформації, ступінь обмеження доступу та рівень її охорони державою [5].

* 1. Технічний захист інформації

Одним з напрямків захисту інформації в ІС є технічний захист інформації (ТЗІ). Технічний захист інформації в АС забезпечується комплексом конструкторських, інженерних, програмних і технічних заходів на всіх етапах їх створення та експлуатації [6].

Технічний захист інформації (ТЗІ) – діяльність, спрямована на забезпечення інженерно-технічними заходами конфіденційності, цілісності та доступності інформації з обмеженим доступом, а також цілісності та доступності відкритої інформації, важливої для особи, суспільства і держави.

Мета ТЗІ – запобігання витоку або порушенню цілісності ІзОД. Мета ТЗІ досягається створенням системи ТЗІ.

Система ТЗІ – сукупність об’єктів, об’єднаних цілями та завданнями захисту інформації інженерно-технічними заходами, нормативно-правова та матеріально-технічна база.

Етапи побудови системи захисту інформації:

1. етап – визначення й аналіз загроз;
2. етап – розробка системи захисту інформації;
3. етап – реалізація плану захисту інформації;
4. етап – контроль функціонування та керування системою захисту інформації.

Технічний канал витоку інформації (ТКВ) – сукупність джерела інформації (небезпечного сигналу), фізичного середовища, в якому розповсюджується носій небезпечного сигналу, завад та технічних засобів розвідки (ТЗР) противника [6].

* 1. Криптографічний захист інформації та його методи

Швидкий розвиток інформаційних технологій привів до нових досягнень в сфері безпеки інформації, яка є дуже важливою для сучасного суспільства. Питання розроблення та впровадження методів ЗІ є актуальними не лише для криптографії та стеганографії, а й для майже всіх галузей науки, враховуючи високу автоматизацію різних сфер людської діяльності.Серед усього спектру методів захисту даних від несанкціонованого доступу особливе місце займають криптографічні методи. На відміну від інших методів, вони спираються лише на властивості самої інформації і не використовують властивості її [матеріальних](http://ua-referat.com/%D0%9C%D0%B0%D1%82%D0%B5%D1%80%D1%96%D0%B0%D0%BB%D0%B8) носіїв, особливості вузлів її обробки, передачі та зберігання.

Криптографічний захист – вид захисту, що реалізується за   
допомогою перетворень інформації з використанням спеціальних даних   
(ключових даних) з метою приховування (або відновлення) змісту   
інформації, підтвердження її справжності, цілісності, авторства   
тощо [7].

Засіб криптографічного захисту інформації – програмний,   
апаратно-програмний, апаратний або інший засіб, призначений для   
криптографічного захисту інформації.

Система криптографічного захисту інформації – сукупність  
органів, підрозділів, груп, діяльність яких спрямована на   
забезпечення криптографічного захисту інформації, та підприємств,   
установ і організацій, що розробляють, виробляють, експлуатують та   
(або) розповсюджують криптосистеми і засоби криптографічного   
захисту інформації.

Суть криптографії – у забезпеченні конфіденційності даних (відкритого тексту, секретного ключа) шляхом їх шифрування [8]. 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.

Одним із видів такого перетворення є шифрування, яке забезпечує практичну неможливість читання або модифікації інформації зловмисниками.

Методи шифрування повинні мати мінімум 2 властивості:

* законний власник зможе виконати зворотне перетворення й розшифрувати повідомлення;
* криптоаналітик ворога, що перехопив повідомлення, не зможе відновити по ньому вихідного повідомлення без таких затрат часу та засобів, які зроблять дану роботу недоцільною.

Криптосистема – сукупність вибраних за допомогою ключа оборотних перетворень, які перетворюють відкриту інформацію, що підлягає захисту в шифрограму й назад у відкритий вигляд.

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. Ця послідовність показана на рис.1.2. 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 (див. рис.1.2.).

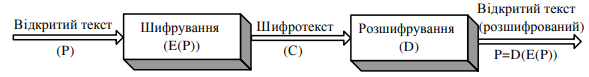


Рисунок 1.2 – Шифрування і розшифрування

Сучасна криптографія містить у собі чотири базові розділи:

* симетричні криптосистеми (системи з секретним ключем або одноключові);
* асиметричні криптосистеми (системи з відкритим ключем або двоключові);
* криптографічні протоколи;
* керування ключами.

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 (див. рис. 1.3).

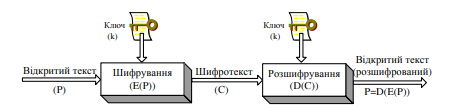


Рисунок 1.3 – Шифрування і розшифрування за допомогою ключа (симетрична)

Для деяких алгоритмів при шифруванні і розшифруванні використовуються різні ключі K1 − для шифрування, K2 − для розшифрування, то у цьому випадку C = А*k*1(P), P = D*k*2(C), P = D*k*2(E*k*1(P)) (див. рис.1.4).

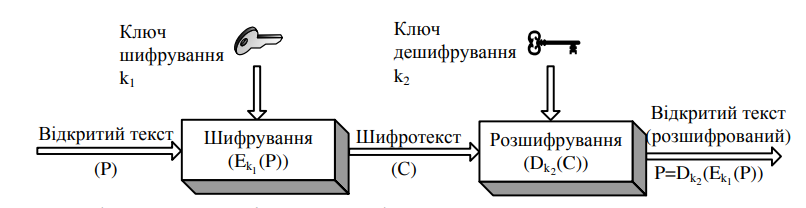


Рисунок 1.4 – Шифрування і розшифрування за допомогою двох різних ключів (асиметрична)

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

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 узгодили 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 [7].

Симетричні алгоритми поділяються на дві категорії:

1) алгоритми, що обробляють відкритий текст побітово (іноді побайтово) називаються потоковими;

2) алгоритми, що працюють з групами бітів, які складають блоки (наприклад, 64 біт) називаються блоковими.

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 (протягом певного інтервалу часу) розрахований за ключем шифрування. Алгоритми «з відкритим ключем» називаються 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 таких системах ключ шифрування буде відкритим, а розшифрування − закритим.

## 1.4 Висновки до розділу

Інформація (згідно Закону України “Про інформацію”) – це документовані або публічно оголошені відомості про події та явища, що відбуваються у суспільстві, державі та навколишньому природному середовищі.

До основних властивостей інформації належать: конфіденційність, цілісність, доступність та достовірність. Для досягнення належного рівня захисту інформації недостатньо використання лише одного методу захисту, наприклад технічного. Такого рівня можна досягнути лише за умови використання у тому числі сучасних криптографіних методів шифрування.

В даному розділі було розглянуто поняття інформаційної безпеки, яке є багатоаспектним та багатовимірним, наведено класифікацію інформації та розглянуто криптографічний захист інформації.

На сьогоднішній день для криптографічного захисту інформації, що є власністю держави, використовуються лише програмно-апаратні засоби, а їх реалізація можлива лише на сучасній елементній базі (мікроконтролери, мікропроцесори та програмовані логічні матриці). Нобхідною умовою правильного проектування програмно-апаратних засобів криптографічних перетворень є застосування систем автоматизованого проектування.

# РОЗДІЛ 2

# ПРОЕКТУВАННЯ ЦИФРОВИХ ПРИСТРОЇВ НА ПРОГРАМУЮЧИХ ЛОГІЧНИХ МАТРИЦЯХ В СИСТЕМІ MAX+PLUS II

В даному розділі основну увагу буде звернено на області застосування програмованих логічних інтегральних схем (ПЛІС) та особливості систем автоматизованого проектування (САПР). Розглянено віртуальне середовище MAX+plus II, його можливості та переваги використання. Без САПР неможливо виробляти сучасну техніку, що є дуже складною та вимагає надзвичайної точності виготовлення, тому дані відомості необхідні для подальшого вивчення теми дипломної роботи та можливості практичної реалізації.

2.1 Області застосування та класифікація ПЛІС

ПЛІС - програмовані логічні інтегральні схеми, які використовуються для розробників цифрових пристроїв на теперішній час. Це пристрої з регулярною архітектурою, які представляють собою набір стандартних елементів та вузлів, що інтегровані на одному кристалі, побудови логічних керуючих автоматів з паралельною архітектурою. [9]. ПЛІС останнім часом стають все більш розповсюдженою та звичною елементною базою для розробників цифрових пристроїв.

За рахунок вільного компонування елементів і вузлів програмним способом налаштування функцій ПЛІС здійснюється користувачем досить просто. Завдяки цим особливостям ПЛІС в сучасних умовах та в ряді предметних областей завойовує ринок і становить гідну конкуренцію мікропроцесорам і мікроконтролерам. : (див. рис. 2.1).

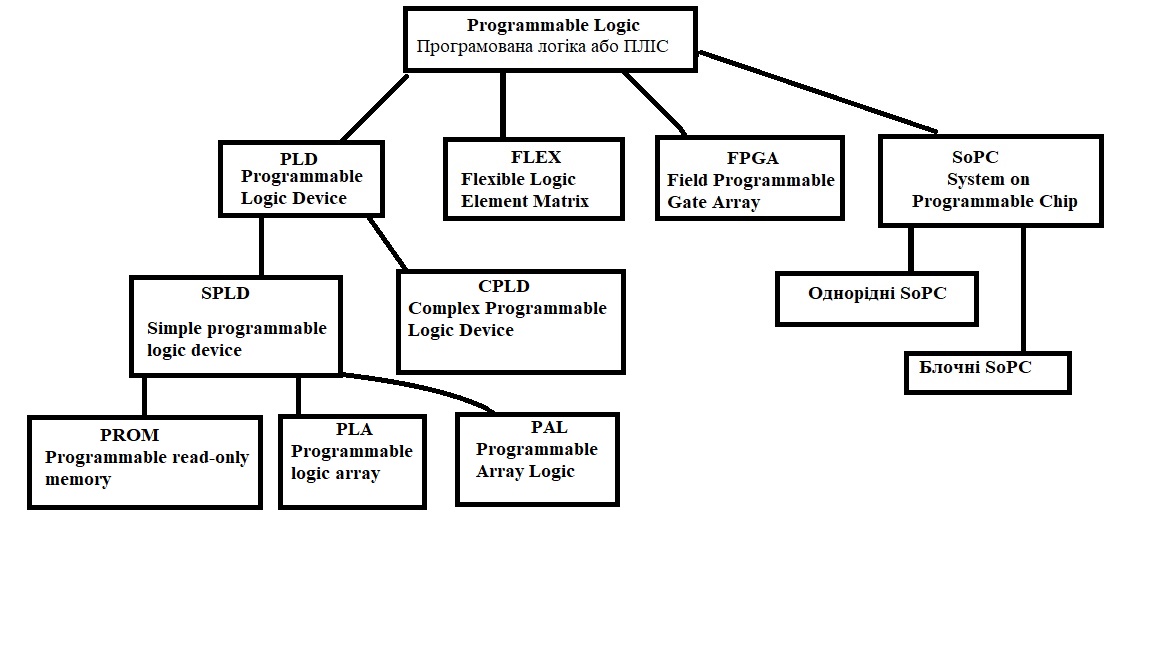


Рисунок 2.1 – Класифікація ПЛІС

Логічна ємність і продуктивність матриці FPGA останніми роками значно збільшується завдяки наступним факторам: переходу на більш високий технологічний рівень, підвищення ступеня інтеграції кристалів, поява більш високих послідовних інтерфейсів і протоколів зв'язку.

З розвитком технологій FPGA знайдено широке застосування у вбудованих обчислювальних системах військового і аерокосмічного призначення, для яких характерні жорсткі обмеження по потужності потреб, розміру і вазі елементів. Матриці FPGA добре підходять для таких пристроїв, як радіолокаційні установки, системи радіотехнічної розвідки, системи обробки зображень, пристрої обробки сигналів. Тобто вони призначені в першу чергу для тих пристроїв, які виробляють обробку сигналів і векторних або матричних обчислень. У таких додатках головним критерієм є не вартість, а характеристики пристрою, особливо його швидкодія.

Під програмованістю будемо розуміти можливість програмування мікросхем користувачем ПЛІС можна класифікувати за багатьма ознаками, а саме:

* рівень інтеграції;
* архітектура;
* число допустимих циклів програмування;
* тип пам’яті конфігурації;
* системні властивості;
* схемотехнологія;
* однорідність або гібридність;

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 [9, 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.
* проектування пристроїв, що виконують криптографічні операції, систем захисту інформації;
* проектування спецпроцесорів та мікроконтролерів;

Найбільш широке застосування ПЛІС знаходять при побудові периферійних модулів комп'ютерів, у тому числі:

* інтерфейсних засобів (адаптерів) – пристроїв, які узгоджують сигнали між різними системами, з різною логікою та напругою живлення;
* пристроїв з великою кількістю портів вводу-виводу;
* засобів сполучення з апаратурою управляючих комплексів і систем;
* засобів захисту інформації;
* апаратури підтримки телекомунікацій (смартфони, КПК та ін.);
* апаратних прискорювачів (наприклад, для реалістичної графіки, обробки зображень, обробки сигналів);
* засоби кодування і декодування інформації в системах зв'язку.

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 убудованими 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.

До основних переваг ПЛІС як компонента при проектуванні цифрової та аналого-цифрової апаратури слід віднести:

* різке скорочення термінів та затрат на проектування, можливість модифікації і налагодження апаратури, а також емуляція схем, які надалі реалізуються на іншій елементній базі;
* зниження вартості первісної розробки;
* висока рентабельність випуску малотиражної продукції;
* “домашнє проектування” (in hause) – весь цикл проектування виконується на одному робочому місці;
* можливість програмування структури;
* висока швидкодія;
* можливість програмування або зміна конфігурації в самій системі;
* висока надійність;
* зменшене енергоспоживання;
* сумісмність при переводі алгоритмів на рівні мов опису апаратури (VHDL, AHDL, Verilog);
* можливість зберігати проект і використовувати його як частину в інших проектах.

До недоліків ПЛІС потрібно віднести:

* відносно не високі частоти близько 1 ГГц у порівнянні з іншими технологіями;
* невисока густина (розміри кристалу перевищують розміри по іншим технологіям).

2.2 Основні поняття про системи автоматизованого проектування

Одним із важливих компонентів сучасного виробництва є системи автоматизованого проектування (САПР).

Система автоматизованого проектування – 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.

Комп’ютерна графіка, будучи підсистемою САПР, розв’язує найбільш трудомістку і важливу задачу САПР: автоматизацію розробки і виконання конструкторської діяльності. Вона забезпечує створення, зберігання і обробку моделей геометричних об’єктів і їх графічне зображення за допомогою комп’ютера.

Процес автоматизованого проектування реалізується на базі спеціального програмного забезпечення, автоматизованих банків даних, широкого набору периферійних пристроїв [10].

Автоматизоване проектування (computer-aided design – CAD) – проектування об’єкта та (або) алгоритму його функціонування або алгоритму процесу, а також описи різноманітними мовами здійснюється взамодією людини і комп’ютера. Це технологія, 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.

Автоматизація проектування включає в себе багато сучасних інформаційних технологій. Так, технічне забезпечення систем автоматизованого проектування (САПР) базується на використанні обчислювальних мереж і телекомунікаційних технологій. В САПР використовуються персональні комп’ютери та робочі станції. Програмні комплекси САПР відносяться до найбільш складних програмних систем, що базуються на операційних системах Unix, Windows-95/NT,мовах програмування С, С++,Java та інших, сучасних CASE- технологіях, стандартах відкритих систем і обміну даними в комп’ютерних середовищах. Установи, що ведуть розробки без САПР є неконкурентноспроможними як через великі матеріальні та часові витрати, так і через невисоку якість проектів. Нині створену величезну кількість програмно- методичних комплексів для САПР із різним ступенем спеціалізації та прикладної орієнтації.

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, то це призводить до скорочення термінів і вартості розробки [10].

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) за спеціалізацією: спеціалізовані та інваріантні;

г) за технічною організацією: однорівневі та багаторівневі.

САПР складається з проектуючої і обслуговуючої підсистем.

Проектуючі підсистеми безпосередньо виконують проектні процедури. Процес проектування ділиться на етапи:

* етап проектування – частина процесу проектування, що включає формування усіх потрібних описів об’єкта;
* проектні процедури – складові частини етапу проектування, виконання яких закінчується отриманням проектного рішення;
* проектна операція – дрібніша складова частина процесу проектування, що входить до складу проектної процедури.

Обслуговуючі підсистеми забезпечують функціонування проектуючих 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.

Застосування САПР у певній діяльності дає змогу впровадити стратегію накопичення та передавання знань і досвіду конструкторів, що забезпечує високу живучість та розв’язання нових більш складних задач. САПР дозволяють проектувати технологічні процеси з меншими витратами часу та засобів, зі збільшенням точності спроектованих процесів і програм обробки, що скорочує витрати матеріалів та час обробки.

2.3 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). САПР MAX + PLUS II являє собою інтегроване середовище для розробки цифрових пристроїв на базі програмованих логічних інтегральних схем фірми Altera 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.

Програмне забезпечення системи MAX+PLUS II представляє собою середовище логічного проектування і допомагає досягти максимальній ефективності і продуктивності [11].

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 опису *Verilog* HDL, а також схемні файли OrCAD.

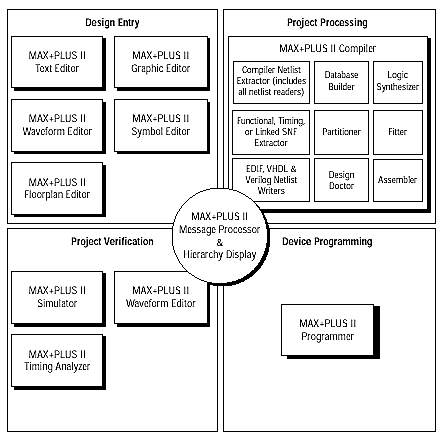


Рисунок 2.2 – Додатки в системі MAX+PLUS II

Система MAX+PLUS II пропонує користувачу багатий графічний інтерфейс, доповнений ілюстрованою оперативною довідковою системою. У повну систему MAX+PLUS II входять 11 впроваджених у систему додатків.

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. Для верифікації проектів використовуються такі додатки як [12]:

* Simulator - програма, яка спільно з редактором тимчасових діаграм призначена для функціонального моделювання проекту з метою перевірки правильності логіки його функціонування. Вона дозволяє розробнику моделювати проект перш, ніж він буде реалізований в пристрої, що істотно скорочує час роботи проекту.
* Timing Analyzer - додаток, що забезпечує розрахунок тимчасових затримок від кожного входу до кожного логічно пов'язаного з ним виходу. Цей часовий аналізатор використовується для аналізу виконання проекту.

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 [11].

Існує ряд програм, що входять в пакет компілятора і призначені для синтезу структури, трасування зв'язків, перевірки коректності проекту і локалізації помилок, формування файлів програмування або конфігурації ПЛІС, таких як [13]:

* Compiler Netlist Extractor - додаток, що забезпечує витяг списку зєднань з вихідного файлу подання проекту, створеного при введенні проекту.
* Database Builder - додаток, призначений для побудови бази даних проекту.
* Logic Synthesizer - додаток, що забезпечує перевірку коректності проекту за формальними правилами і синтез оптимальної структури проекту.
* Partitioner - додаток, що забезпечує розбиття проекту на частини в тих випадках, коли ресурсів одного кристала (мікросхеми) недостатньо для реалізації проекту.
* Fitter - трасувальник внутрішніх зв'язків, що забезпечує реалізацію синтезованої структури.
* Timing SNF Extractor - додаток, що забезпечує витяг параметрів проекту, необхідних для функціонального моделювання та часового аналізу.
* Assembler - додаток, що створює з скомпільованого проекту файли, необхідні для програмування і конфігурації ПЛІС через програматор.

Безпосереднє програмування чи завантаження конфігурації з використанням відповідного апаратного забезпечення виконується з використанням модуля програмування (Programmer), що також дозволяє проводити верифікацію і тестувати ПЛІС фірми Altera [13].

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.

Проект складається зі всіх файлів ієрархічної структури проекту: допоміжних і вихідних файлів. Ім’ям проекту є ім’я файлу проекту верхнього рівня без розширення. Система MAX+plus II виконує компіляцію, тестування, аналіз синхронізації і програмування відразу цілого проекту, хоча користувач може в цей час редагувати фали цього проекту в рамках іншого проекту.

Для видачі на екран повідомлень про помилки, застережливих і інформаційних повідомлень, в системі MAX + PLUS II використовується генератор повідомлень (Message Processor). Він дозволяє визначати і коректувати помилки, одержувати застережливі повідомлення і підказки у будь-який момент часу.

2.4 Функціонування редакторів системи MAX + PLUS II

До складу пакету MAX + PLUS II входять наступні пов'язані між собою додатки, що реалізують всі перераховані вище етапи розробки цифрових пристроїв на ПЛІС фірми Альтера.

Додатки для введення проектів (редактори проектів):

* Graphic Editor - графічний редактор, призначений для введення проекту у вигляді схеми з'єднань символів елементів, які вилучаються з стандартних бібліотек пакету або з бібліотеки користувача. Графічний редактор забезпечує багато можливостей. Можна збільшити чи зменшити масштаб відображення на екрані і побачити дизайн або яку-небудь його деталь. Можна копіювати, вирізувати, вставляти і видаляти обрані фрагменти, одержувати дзеркальне відображення, повертати виділені фрагменти на 90,180 чи 270 градусів, задавати розмір, розміщення поточного листа схеми по вертикалі чи горизонталі. Підтримує наступні формати файлів: .gdf і .sch.
* Waveform Editor - редактор часових діаграм (сигнальний редактор), який дозволяє вводити тестові вектори і проглядати результати тестування. Користувач може створювати сигнальні файли проекту (*.wdf*), що містять логіку дизайну проекту, а також файли каналів тестування (*.scf*), що містять вхідні вектори для тестування і функціонального налагодження. Цей редактор є альтернативою графічного або текстового редакторів. Створений файл формату *.wdf* містить послідовні входи і виходи цифрових автоматів, лічильників та регістрів.
* Text Editor - текстовий редактор, призначений для створення і редагування текстових файлів, що містять опис логіки проекту на мові опису пристроїв AHDL (розширення .tdf) або на близьких до нього мов типу VHDL (розширення .vhd), Verilog HDL. При цьому введення початкових даних про проект здійснюється не у вигляді схеми, виконаної у графічному редакторі, а у вигляді текстового опису проекту, що дозволяє наблизити розробку проекту до процесу програмування. Цей редактор дає можливість робити установки для компілятора, програми моделювання і тимчасового аналізатора.
* Symbol Editor - символьний редактор, що дозволяє редагувати існуючі символи і створювати нові. За допомогою символьного редактора системи MAX + PLUS II можна переглядати, створювати і редагувати символ, що представляє собою логічну схему. Символьний файл має те ж ім’я, що і проект, з розширенням .sym.
* Floorplan Editor - редактор зв'язків (порівневий планувальник), який на плані розташування основних логічних елементів дозволяє вручну розподіляти виводи ПЛІС (закріплювати виводи за конкретними вхідними та вихідними сигналами) і перерозподіляти деякі внутрішні ресурси ПЛІС.

Редактори MAX + PLUS II та редактори створення файлу проекту (графічний, текстовий і сигнальний) мають загальні функції, такі, як, наприклад, збереження і виклик файлу. Крім того, додатки редактора MAX + PLUS II мають наступні загальні функції:

* створення файлів символів і файлів із прототипами функцій;
* пошук вузлів;
* траверз ієрархічного дерева;
* спливаючі вікна меню, що залежать від контексту;
* часовий аналіз;
* пошук і заміна фрагментів тексту;
* скасування останнього кроку редагування, його повернення, вирізка, копіювання, вставка і видалення обраних фрагментів, обмін фрагментами між додатками MAX + PLUS II, додатками Windows.

2.5 Розробка проектів цифрових пристроїв на мові 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). 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). AHDL є високорівневою, модульною мовою, цілком інтегрованою у систему MAX + PLUS II. Вона особливо добре підходить для проектування складної комбінаційної логіки, шин, кінцевих автоматів, таблиць істинності параметричної логіки з врахуванням архітектурних особливостей [ПЛІС](https://uk.wikipedia.org/wiki/%D0%9F%D0%9B%D0%86%D0%A1) цієї фірми. Оператори і елементи мови AHDL є досить потужним і універсальним засобом опису алгоритмів функціонування цифрових пристроїв, зручним у використанні [14].

Можливо використовувати текстові редактори системи MAX + PLUS II або будь-якої іншої для створення текстових файлів проектів (AHDL Text Design Files (.tdf)). Потім можна відкомпілювати .tdf файли для одержання вихідних файлів, придатних для наступного моделювання, часового аналізу і програмування пристрою. Крім того компілятор системи MAX + PLUS II може створювати тестові файли експортування (AHDL Text Design Export Files (.tdx)) і тестові вихідні файли (Text Design Output Files (.tdo), які можна зберегти як файли .tdf і повторно використовувати як файли проекту.

Крім того, система MAX + PLUS II читає файли тестування, створені за допомогою ПО Xilinux і записує файли затримок у форматі SDF, зручно взаємодії з пакетами, що працюють з іншими промисловими стандартами. Система MAX + PLUS II пропонує користувачу багатий графічний інтерфейс, доповнений ілюстрованою оперативною довідковою системою.

## 

## 2.6 Висновки до розділу

В другому розділі було розглянено можливості САПР, їх види та особливості. Варто звернути увагу, що для виготовлення сучасної техніки вкрай необхідним є використання САПР, що дозволяє створювати складну сучасну техніку з високою точністю виготовлення.

САПР «MAX + PLUS II» є інтегрованим середовищем, яке необхідне для розробки цифрових пристроїв на базі ПЛІС фірми Altera і забезпечує виконання всіх вимог підготовки виробу до його випуск на ринок. Система пропонує повний спектр можливостей логічного дизайну, різноманітні засоби опису проекту для створення проекту з ієрархічною структурою, потужний логічний синтез, компіляцію із заданими часовими параметрами, функціональне й часове тестування, аналіз часових параметрів системи, автоматичну локалізацію помилок, а також програмування і верифікацію пристроїв. Для правильного функціонування редакторів цієї системи доцільно використовувати мову опису апаратних засобів AHDL. Саме ця мова найбільш підходить для реалізації поставленого завдання дипломної роботи.

Таким чином, на підставі отриманих результатів можна сказати, що створення засобів КЗІ на базі ПЛІС в САПР MAX + PLUS II надає можливість створення широкого спектру засобів для КЗІ. Для доведення цього в роботі буде продемонстрована можливість реалізації ДСТУ “Калина” на програмуємих структурах. Тому, розгляд САПР MAX + PLUS II є особливо важливим в даній роботі, так як практична реалізація алгоритму шифрування “Калина” буде здійснена саме за допомогою цієї системи.

# РОЗДІЛ 3

# АНАЛІЗ АЛГОРИТМУ ШИФРУВАННЯ ДСТУ 7624:2014

В даному розділі наведено особливості блокового шифрування, розглянено вітчизняний алгоритм шифрування ДСТУ 7624:2014 “Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного блокового перетворення”, побудований на основі SP-мережі. Дані відомості необхідні для правильного проектування шифру, що є основним завданням даної роботи.

3.1 Означення алгоритму блокового шифрування

Алгоритм шифрування являє собою множину зворотних перетворень форми повідомлень з метою його захисту від несанкціонованого доступу з його змістом. Вихідне повідомлення, яке піддається перетворенню, називається відкритим текстом, а результат, отриманий за допомогою застосування алгоритму шифрування, називається шифртекстом. Перехід від відкритого тексту до шифртексту називається защифруванням, а зворотній перехід – розшифруванням. Необхідною умовою виконання як прямого так і зворотнього криптографічного перетворення є наявність секретного ключа.

Блоковим алгоритмом шифрування називається такий алгоритм шифрування, в якому як зашифрування, так і розшифрування виконуються над блоками фіксованої довжини [15].

Крім необхідного рівня криптографічної стійкості, до БСШ ставиться вимога забезпечення високого рівня продуктивності (складності зашифрування, розшифрування та розгортання ключа). Зважаючи на значні обсяги інформації, що обробляються в ІТС, ця вимога є надзвичайно важливою та критичною для ефективного функціонування всієї ІТС. Також під час розроблення систем із застосуванням БСШ, крім перерахованих вимог, необхідно враховувати вартість реалізації засобів шифрування та експлуатації відповідних засобів.

Необхідність захисту інформації у вбудованих системах (ВС) привела до інтенсивних досліджень шляхів ефективної реалізації криптоалгоритмів, за умови обмежень, які накладаються цими системами. Ресурси вбудованих систем обмежені продуктивністю процесорного ядра, споживаною потужністю, розміром доступної постійної та оперативної пам’яті . Розмір коду програми безпосередньо впливає на вартість мікропроцесора, який переважно є найдорожчим компонентом системи. Час виконання є критичним з огляду на енергоспоживання, оскільки, як правило, у ВС центральний процесор більшу частину часу знаходиться в режимі пониженого енергоспоживання, виходячи з нього лише на короткий час для збирання, обробки та передачі інформації. Відповідно, час виконання криптографічного алгоритму прямо пропорційний споживаній потужності пристрою [15].

Перевагою блокового шифру є те, що він здатний зашифрувати одним ключем одне або кілька повідомлень, сумарною довжиною більше, ніж довжина ключа. Передача малого в порівнянні з повідомленням ключа по зашифрованому каналу – завдання більш просте і швидке, ніж передача самого повідомлення або ключа такої ж довжини, що робить можливим його повсякденне використання.

Процес зашифрування має наступні властивості:

* різні блоки відкритого тексту відображаються в різні блоки шифртексту;
* при розшифруванні відповідність зберігається;
* пряме перетворення можна розглядати як перестановку на множині повідомлень з фіксованим розміром блоку;
* сукупність даних, що визначають конкретне перетворення шифру з множини можливих, називається ключем.

Ідея, що лежить в основі блокових шифрів, полягає в побудові криптостійких систем шляхом багаторазового застосування відносно простих криптографічних перетворень в якості яких Шеннон запропонував використовувати перетворення підстановки (substitution) і перестановки (permutation), схеми, що реалізують ці перетворення, називаються SP-мережами.

Багаторазове використання цих перетворень дозволяє забезпечити дві властивості, які повинні бути притаманні стійким шифрам: розсіювання (diffusion) і перемішування (confusion). Розсіювання передбачає поширення впливу одного знаку відкритого тексту, а також одного знаку ключа на значну кількість знаків шифротексту. Наявність у шифру цієї властивості дозволяє:

* приховати статистичну залежність між знаками відкритого тексту, інакше кажучи перерозподілити надмірність вихідної мови за допомогою розповсюдження її на весь текст;
* не дозволяє відновлювати невідомий ключ по частинах.

Мета перемішування – зробити якомога складнішою залежність між ключем і шифртекстом. Криптоаналітик на основі статистичного аналізу перемішаного тексту не повинен отримати ніякої інформації про використані ключі.

Застосування розсіювання і перемішування порізно не забезпечує необхідну стійкість, надійна криптосистема виходить тільки в результаті їх спільного використання.

У сучасних блокових криптосистемах раундові шифри будуються в основному з використанням операцій заміни двійкових кодів невеликої розрядності (схеми, що реалізують цю нелінійну операцію, називаються S-блоками; як правило, саме від їх властивостей в першу чергу залежить стійкість всієї системи), перестановки елементів двійкових кодів, арифметичних і логічних операцій над двійковими кодами [16].

Одним з основних прийомів при розробці криптографічних перетворень є багаторазова, яка складається з багатьох циклів, обробка одного блоку відкритого тексту. Цей прийом ще іноді називають ітеруванням. Під час кожного циклу, дані піддаються спеціальному перетворенню за участю допоміжного ключа, отриманого з заданого секретного ключа. Вибір числа циклів визначається вимогами крипостійкості та ефективності реалізації блокового шифру. Як правило, чим більше циклів, тим більша криптостійкість і менша ефективність реалізації (більша затримка при шифруванні/розшифруванні) блокового шифру і навпаки.

Загалом можна сформулювати також вимоги відносно стійкості сучасних БСШ:

* забезпечення стійкості до силових атак, наприклад за часовим або просторовим критерієм стосовно зберігання проміжних результатів обсягу пам'яті;
* відсутність способів побудови або розв’язання системи рівнянь, що зв'язує відкритий текст, криптограму й ключ шифрування;
* неможливість реалізації відомих аналітичних атак на шифр або їхня складність повинна бути вищою від складності реалізації силових атак (наприклад, один з таких критеріїв: потужність необхідної множини відкритих/зашифрованих повідомлень; необхідна кількість операцій шифрування; обсяг пам'яті, потрібний для зберігання проміжних результатів);
* наявність “запасу стійкості” шифру (додаткових циклів шифрування), що забезпечує безпечне використання алгоритму у випадку вдосконалювання криптоаналітичних атак;
* стійкість спрощеного варіанта шифру, у якому деякі операції вилучені або замінені простішими;
* забезпечення “належних” статистичних властивостей вихідної послідовності шифру (криптограми або гами, що шифрує), за яких криптограми й гами шифрування практично не відрізняються за властивостями від випадкової послідовності.

Блокові симетричні шифри є одним із основних криптографічних механізмів забезпечення конфіденційності, цілісності та доступності інформації та інформаційних ресурсів у сучасних інформаційно-телекомунікаційних системах. Вони використовуються при реалізації послуг захисту інформації як самостійно, так і в сукупності з іншими криптографічними механізмами безпеки. Особливо важливим застосування БСШ є тоді, коли вимагається середня чи висока швидкодія шифрування чи автентифікації інформації, а також швидкісна реалізація режимів роботи БСШ. Іншою важливою властивістю БСШ є можливість забезпечення достатньо високої швидкодії шифрування та розгортання ключа. Також до переваг блокових шифрів відносять схожість процедур шифрування та розшифрування, які, як правило, відрізняються лише порядком дій. Це спрощує створення пристроїв шифрування, так як дозволяє використовувати одні й ті ж блоки в ланцюгах шифрування та розшифрування.

Розрізняють два види побудови блокових алгоритмів шифрування. Одні з них будуються на основі схеми Фейстеля. До таких алгоритмів відносяться, наприклад, алгоритм DES і ГОСТ 28147-89. Інші будуються за образом SP-мережі (Substitution-Permutation Network). До числа таких алгоритмів належать AES, Калина, “Кузнєчік”. Далі ми детальніше розглянемо принцип побудови алгоритмів блокового шифрування на основі ДСТУ 7624:2014.

3.2. Опис блокового шифру ДСТУ 7624:2014 “Калина”

В Україні з 1990 року в якості основного шифру використовувався ГОСТ 28147-89. Хоча він все ще забезпечує практичну стійкість, для нього вже відомі теоретичні методи криптоаналізу, зі складністю істотно меншою, ніж повний перебір ключів. Особливістю ГОСТ 28147:2009 є використання стандартного довготривалого ключового елемента (ДКЕ), який, як зазначено в стандарті, поставляється в установленому порядку.

З точки зору продуктивності, ГОСТ 28147-89 істотно поступається сучасним аналогам, таким, як AES, що призводить до ускладнення і подорожчання засобів криптографічного захисту при інших рівних характеристиках. Необхідно відзначити, що вказаний, по суті, регіональний шифр ГОСТ 28147:2009 вже виведений з дії в Білорусії і, судячи з результатів наукових конференцій, він планується до заміни і в Російській Федерації у вигляді алгоритму “Кузнечик” [17].

Таким чином, в Україні існує проблема розробки і введення в дію нового сучасного стандарту шифрування, здатного стати основою для створення ефективних засобів криптографічного захисту наступних поколінь.

Враховуючи сучасні міжнародні тенденції та позитивний досвід у галузі розробки перспективних криптографічних перетворень, Державна служба спеціального зв’язку та захисту інформації України успішно провела національний відкритий конкурс симетричних блокових криптографічних алгоритмів. За результатами конкурсу, був відзначений алгоритм “Калина”, на базі якого був розроблений стандарт ДСТУ 7624:2014 “Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного блокового перетворення” [18]. Цей стандарт є результатом багаторічної плідної співпраці Державної служби спеціального зв’язку та захисту інформації України та провідних українських науковців і враховує досвід та результати проведення міжнародних і відкритого національного конкурсів криптографічних алгоритмів. Він призначений для поступової заміни міждержавного стандарту [ДСТУ ГОСТ 28147-2009](http://www.wikiwand.com/uk/%D0%93%D0%9E%D0%A1%D0%A2_28147-89). Порівняно з відомим міжнародним стандартом AES (ISO/IEC 18033-3:2010), алгоритм ДСТУ 7624:2014 забезпечує вищий рівень криптографічної стійкості (із можливістю застосування блока та ключа шифрування включно до 512 бітів) і аналогічну або вищу швидкодію на сучасних і перспективних програмних і програмно-апаратних платформах.

При розробці ДСТУ 7624:2014 було прийнято рішення забезпечити прозорість проектування і використовувати консервативний підхід із застосуванням добре досліджених конструкцій, що забезпечують запас стійкості для безпечного застосування алгоритму в умовах істотного прогресу криптоаналітичних технік і засобів обробки даних.

До блокового шифру “Калина” ставляться такі вимоги: високий рівень криптографічної стійкості з достатнім запасом у разі появи нових атак протягом тривалого часу; висока швидкодія програмної реалізації на сучасних та перспективних платформах; компактність програмної та програмно-апаратної реалізації; можливість ефективної інтеграції декількох алгоритмів в одному засобі криптографічного захисту; прозорість проектування, консервативний підхід до забезпечення стійкості; вища (або однакова) ефективність порівняно з найкращими світовими рішеннями.

Окрім блокового шифру, ДСТУ 7624:2014 визначає режими роботи для забезпечення конфіденційності та цілісності, значення для перевірки, надає рекомендації щодо реалізації криптографічного перетворення і обмежень на обсяг інформації, яка захищається на даному ключі ( див табл. 3.1). ДСТУ 7624:2014 визначає десять різних режимів роботи (застосування), які широко поширені відповідно до міжнародного стандарту ISO/IEC 10116:2006. Це спрямовано на забезпечення широкої застосовності, у тому числі для захисту інформації, що передається комп’ютерними мережами (Інтернет), прозорого шифрування жорстких дисків та знімних носіїв, електронних документів, ключових даних тощо. Наявність такої кількості режимів роботи надає можливість ефективної реалізації систем, засобів і протоколів криптографічного захисту інформації в інформаційно-телекомунікаційних системах різноманітного призначення.

Таблиця 3.1 – Режими роботи алгоритму

|  |  |  |  |
| --- | --- | --- | --- |
| № режиму | Назва режиму | Позна-чення | Послуга безпеки |
| 1 | Проста заміна (базове перетворення) | ECB | Конфіденційність |
| 2 | Гамування | CTR | Конфіденційність |
| 3 | Гамування зі зворотнім зв’язком по шифртексту | CFB | Конфіденційність |
| 4 | Вироблення імітовставки | CMAC | Цілісність |
| 5 | Зчеплення шифр-блоків | CBC | Конфіденційність |
| 6 | Гамування зі зворотнім зв’язком за шифргамою | OFB | Конфіденційність |
| 7 | Вибіркове гамування із прискореним виробленням імітовставки | GCM, GMAC | Конфіденційність і цілісність (GCM), тільки цілісність (GMAC) |
| 8 | Вироблення імітовставки і гамування | CCM | Конфіденційність і цілісність |
| 9 | Індексована заміна | XTS | Конфіденційність |
| 10 | Захист ключових даних | KW | Конфіденційність і цілісність |

Новий національний стандарт здатний забезпечити:

* надвисоку стійкість - коли довжини блоку інформації та довжина вихідного ключа не менше за 512 бітів;
* високу стійкість - коли довжина блоку інформації та довжина ключа не менші, ніж 256 бітів;
* нормальну стійкість, коли довжина блоку інформації та довжина ключа не менше ніж 128 бітів;
* задовільну стійкість, коли довжина блоку інформації не менш ніж 64 бітів, а довжина ключа – не менше ніж 128 бітів.

Наразі це єдиний в світі стандарт блочного шифрування, що підтримує 512- бітові симетричні ключі. Різні варіанти забезпечують гнучкість вибору параметрів для розробників систем криптографічного захисту, що дозволяє отримати як найвищий рівень швидкодії, так і найбільший запас стійкості перетворення.

Принципи побудови алгоритму:

* здатність протистояти відомим методам криптографічного аналізу та мати запас стійкості з урахуванням тенденцій розвитку засобів електронної обчислювальної техніки та криптографічної науки;
* криптографічні перетворення, що застосовуються, повинні ґрунтуватись на надійній та прозорій математичній базі та не мати вбудованих лазівок;
* швидкодія криптоалгоритму повинна бути не меншою, ніж швидкодія чинного державного стандарту шифрування.

Реалізація криптоалгоритму:

* криптоалгоритм повинен бути орієнтованим для можливості реалізації на 32- або 64-розрядних процесорах;
* зазначені в криптоалгоритмі операції повинні мати ефективну програмну та апаратну реалізацію;
* необхідний для роботи обсяг пам’яті має враховувати можливість реалізації криптоалгоритму у мікропристроях;
* давати змогу паралельного виконання декількох операцій.

Ключова система:

* наявність ключа сеансу;
* криптоалгоритм може передбачати наявність довгострокового ключа;
* довжина синхронізуючої посилки – не менше ніж 64 бітів.

Високорівнева конструкція використовує добре досліджену Square-подібну SPN-структуру, яка застосовується в алгоритмах AES/Rijndael, Whirlpool, “Стрибог”, “Коник” і багато інших. Циклове перетворення побудовано на базі таблиць підстановки (S-блоків) і множення на МДР- матрицю над кінцевим полем, забезпечуючи необхідні криптографічні властивості. Застосування саме такої конструкції дозволяє забезпечити доказову стійкість до диференціального, лінійного та іншим видам криптоаналізу, одночасно забезпечуючи ефективну реалізацію на широкому спектрі програмних і програмно-апаратних платформ. При виборі розміру МДР-матриці був прийнятий до уваги розмір кеша L1 сучасних і перспективних процесорів, що дозволило оптимізувати швидкодію програмної реалізації шифру.

Альтернативний варіант, який розглядався при розробці циклової функції, - ARX перетворення (Addition-Rotation-XOR). Цей підхід реалізований в шифрі SPEC (розроблений Агентством національної безпеки США і переданий у відкритий доступ), в блокових алгоритмах, на основі яких побудовано сімейство хеш-функцій SHA-0,1,2 та інших. Перевагою підходу є компактність і швидкодія перетворення. Але, в той же час, є і істотний недолік, пов'язаний з відсутністю методів, що дозволяють виконати суворе аналітичне обґрунтування криптографічної стійкості таких рішень. Навіть з найпотужнішими в світі можливостями для аналізу, США кілька разів були змушені модифікувати свої стандарти хешування через знайдені вразливості: з 1993 по 1995 рр. діяв SHA-0, з 1995 по 2001 рр. застосовувався SHA-1, з тих пір використовується SHA-2. Через проблеми які виникають до стійкості цієї версії, в США з 2008 по 2012 рр. був проведений міжнародний конкурс SHA-3. До теперішнього часу розроблений проект стандарту FIPS 202,що описує нову криптографічну хеш-функцію [18].

Таким чином, консервативний і прозорий підхід до проектування нового національного стандарту України зумовив вибір добре перевіреної конструкції на базі S-блоків і лінійного перетворення, для якої можливо забезпечити доказову стійкість до різних видів криптоалгоритму.

При порівнянні характеристик S-блоків та інших симетричних перетворень, в тому числі нових білоруських і російських стандартів, можна відзначити, що саме національний стандарт України забезпечує найбільшу нелінійність булевих функцій, що дає додатковий запас стійкості для взаємо-однозначної підстановки можна забезпечити, застосовуючи, наприклад, афінно-еквівалентні статичні функції в кінцевому полі, але такі перетворення, використані в AES, Camellia та інших алгоритмах, ставлять шифр під загрозу реалізації алгебраїчної атаки (цей метод криптоаналізу був успішно застосований проти шифру Keeloq, що використовується в системах автомобільної безпеки).

В якості схеми розгортання ключів була запропонована нова конструкція з наступними властивостями:

* забезпечення криптографічної стійкості до інших відомих методів аналізу, відсутність “слабких” ключів, які можуть погіршити властивості перетворення;
* зручність програмної і програмно-апаратної реалізації (для формування циклових ключів застосовуються тільки операції, що використовуються при шифруванні);
* висока обчислювальна складність відновлення ключа шифрування по одному чи декільком циклових ключів.

Остання властивість забезпечує додатковий захист до атак на реалізацію, коли зловмисник намагається атакувати інженерні рішення (вимірюючи споживаний пристроєм струм, навмисно викликаючи збої в роботі через навмисний перегрів шифратора та інші). Ця особливість є суттєвою перевагою для ряду додатків, зокрема, при реалізації шифрування на смарт-картках, USB-токенах та інших, коли ключ прошитий в пристрої і повинен бути захищений від зовнішнього доступу (наприклад, в модулях доступу до платних цифрових ТБ каналах та інші).

У порівнянні з іншими алгоритмами на основі Square- подібної SPN- структури, блоковий шифр “Калина” має суттєві конструктивні відмінності:

* початкове і кінцеве забілювання з використанням модульного складання () для підвищення складності криптоаналітичних атак;
* застосування чотирьох різних S-блоків (замість одного) з властивостями для захисту від алгебраїчних атак, і при порівнянні з іншими блоковими і потоковими шифрами забезпечують найбільшу нелінійність булевих функцій, що дає додатковий запас стійкості перетворення;
* збільшений розмір МДР-перетворення, що покращує криптографічні властивості і дозволяє оптимізувати швидкодію на сучасних 64-бітових платформах;
* нову односпрямовану схему формування циклових ключів, що забезпечує захист від атак, ефективність програмної і програмно-апаратної реалізації, разом з додатковою стійкістю до методів аналізу спеціального виду.

Оцінка криптографічної стійкості до диференціального, лінійного, алгебраїчного, інтегрального та іншим методам аналізу (практичний критерій) показала, що шифр є стійким при 6 циклах для 128-бітового блоку, 7 циклах для 256-бітового і 9 циклах для 512-бітового (кожен додатковий цикл забезпечує експоненціальне зростання складності криптоаналізу). Таким чином, шифр, що містить 10,14 і 18 циклів для блоку 128,256,512 бітів відповідно, забезпечує захист від розглянутих видів аналізу і має суттєвий запас стійкості.

Тобто проведений в Україні відкритий конкурс криптографічних алгоритмів дозволив визначити перспективний блоковий шифр, на основі якого і був розроблений новий національний криптографічний стандарт блочного перетворення ДСТУ 7624: 2014. Даний алгоритм в подальшому забезпечить надання в державі безпечних електронних довірчих послуг з використанням засобів криптографічного захисту інформації, у тому числі надійних засобів електронного цифрового підпису, з підтвердженою відповідністю, що відповідає сучасним і перспективним вимогам Європейського Союзу. Згідно чинних змін до наказу [Держспецзв'язку](http://www.wikiwand.com/uk/%D0%94%D0%B5%D1%80%D0%B6%D0%B0%D0%B2%D0%BD%D0%B0_%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%B0_%D1%81%D0%BF%D0%B5%D1%86%D1%96%D0%B0%D0%BB%D1%8C%D0%BD%D0%BE%D0%B3%D0%BE_%D0%B7%D0%B2%27%D1%8F%D0%B7%D0%BA%D1%83_%D1%82%D0%B0_%D0%B7%D0%B0%D1%85%D0%B8%D1%81%D1%82%D1%83_%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D1%97_%D0%A3%D0%BA%D1%80%D0%B0%D1%97%D0%BD%D0%B8) від [20 серпня](http://www.wikiwand.com/uk/20_%D1%81%D0%B5%D1%80%D0%BF%D0%BD%D1%8F) [2012](http://www.wikiwand.com/uk/2012)року №1236/5/453після [1 січня](http://www.wikiwand.com/uk/1_%D1%81%D1%96%D1%87%D0%BD%D1%8F) [2022](http://www.wikiwand.com/uk/2022) року разом з функцією гешування [ДСТУ 7564­-2014](http://www.wikiwand.com/uk/%D0%9A%D1%83%D0%BF%D0%B8%D0%BD%D0%B0_(%D0%B3%D0%B5%D1%88-%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D1%8F)) (функція гешування “Купина”) є обов'язковим для використання при накладанні та перевірці електронного цифрового підпису за [ДСТУ 4145-2002](http://www.wikiwand.com/uk/%D0%94%D0%A1%D0%A2%D0%A3_4145-2002) “Інформаційні технології. Криптографічний захист інформації. Цифровий підпис, що ґрунтується на еліптичних кривих. Формування та перевірка” замість криптографічного перетворення за [ДСТУ ГОСТ 28147-2009](http://www.wikiwand.com/uk/%D0%93%D0%9E%D0%A1%D0%A2_28147-89).

Допустимі комбінації розміру блока та довжини ключа шифрування наведено у таблиці 3.2.

Таблиця 3.2 – Параметри алгоритму “Калина”

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № з/п | Розмір блоку (*l*) | Довжина ключа (*k*) | Кількість ітерацій перетворення (t) | Кількість стовпців в матриці (*с*) |
| 1 | 128 | 128 | 10 | 2 |
| 2 | 256 | 14 |
| 3 | 256 | 256 | 14 | 4 |
| 4 | 512 | 18 |
| 5 | 512 | 512 | 18 | 8 |

Алгоритм шифрування є процедурою, що складається з попереднього і прикінцевого забілювання та ітеративного циклового перетворення. На вхід кожного циклового перетворення подається поточний стан, а також необхідна кількість ключової інформації (цикловий ключ). Відкритий текст копіюється в поточний стан перед початком зашифрування, а після його завершення в поточному стані знаходиться шифртекст. Кількість циклів шифрування (N r ) залежить від довжини ключа; її наведено у таблиці 3.2. На початку та у кінці процедури зашифрування/розшифрування виконуються додаткові операції забілювання [19].

У стандарті визначені обсяги повідомлень, після обробки яких потрібна обов’язкова зміна ключа. Крім того, наводяться рекомендації розробникам, що звертають увагу на необхідність запобігання атак з використанням особливостей реалізації засобів шифрування. Зокрема, враховані особливості, які дозволяли організацію атак BEAST і CRIME / BREACH в протоколах SSL/TLS, повторний прийом повідомлень, відновлення конфіденційних параметрів на основі залежності часу шифрування від даних (через промахи кешу при табличній реалізації) та інших.

До вхідних та вихідних даних алгоритму “Калина” належать відкритий текст та шифртекст відповідно. Ці параметри представляються у вигляді рядків заданої довжини 8 × *Nb* байт (64 × *Nb* біт). Додатковим вхідним параметром є ключ, розмір якого дорівняє 8 × *Nk* байт (64× *Nk* біт). Байтовий рядок довжиною n = 8 × *Nb* байт представляється у наступній формі: *B0 B1 B2 … Bn – 1*.

До початку зашифрування відкритий текст копіюється в поточний стан шифру. Після завершення процедури зашифрування шифртекст копіюється з поточного стану [19].

Відкритий текст представлено байтовим рядком *in0, in1,… in31.* Отриманий шифртекст представлено як послідовність байт *out0,out1, …out31.*

Заповнення початкового стану шифру перед початком зашифрування та після його закінчення для блоку 128 біт представлено на рис.3.1.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Вхідна послідовність | |  | Поточний стан | |  | Вихідна послідовність | |
| ***in0*** | ***in8*** |  | ***s0,0*** | ***s0,1*** |  | ***out0*** | ***out8*** |
| ***in1*** | ***in9*** |  | ***s1,0*** | ***s1,1*** |  | ***out1*** | ***out9*** |
| ***in2*** | ***in10*** |  | ***s2,0*** | ***s2,1*** |  | ***out2*** | ***out10*** |
| ***in3*** | ***in11*** | **→** | ***s3,0*** | ***s3,1*** | **→** | ***out3*** | ***out11*** |
| ***in4*** | ***in12*** |  | ***s4,0*** | ***s4,1*** |  | ***out4*** | ***out12*** |
| ***in5*** | ***in13*** |  | ***s5,0*** | ***s5,1*** |  | ***out5*** | ***out13*** |
| ***in6*** | ***in14*** |  | ***s6,0*** | ***s6,1*** |  | ***out6*** | ***out14*** |
| ***in7*** | ***in15*** |  | ***s7,0*** | ***s7,1*** |  | ***out7*** | ***out15*** |

Рисунок 3.1 – Заповнення початкового стану для довжини блоку 128 біт

Аналогічно, при розшифруванні шифртекст позначається байтовою послідовністю *in0, in1,… in31* ; отриманий відкритий текст представлено послідовністю байт *out0 , out1 , …out31* ; заповнення відповідає рис.3.1.

Операція лінійного розсіювання (перемішування в стовбці) алгоритму “Калина” використовує поліноміальне представлення байт в полі GF(28), сформованому незвідним поліномом. Для алгоритму шифрування “Калина” у якості незвідного полінома використовується *f(x) = x 8 + x 4 + x 3 + x 2 + 1*.

Зашифрування у режимі простої заміни передбачає те, що на вхід процедури подається відкритий текст та циклові ключі. На початку зашифрування відкритий текст представляється у вигляді блоку даних, що описує поточний стан шифру. Після закінчення зашифрування отриманий шифртекст формується у вигляді байтової послідовності [20].

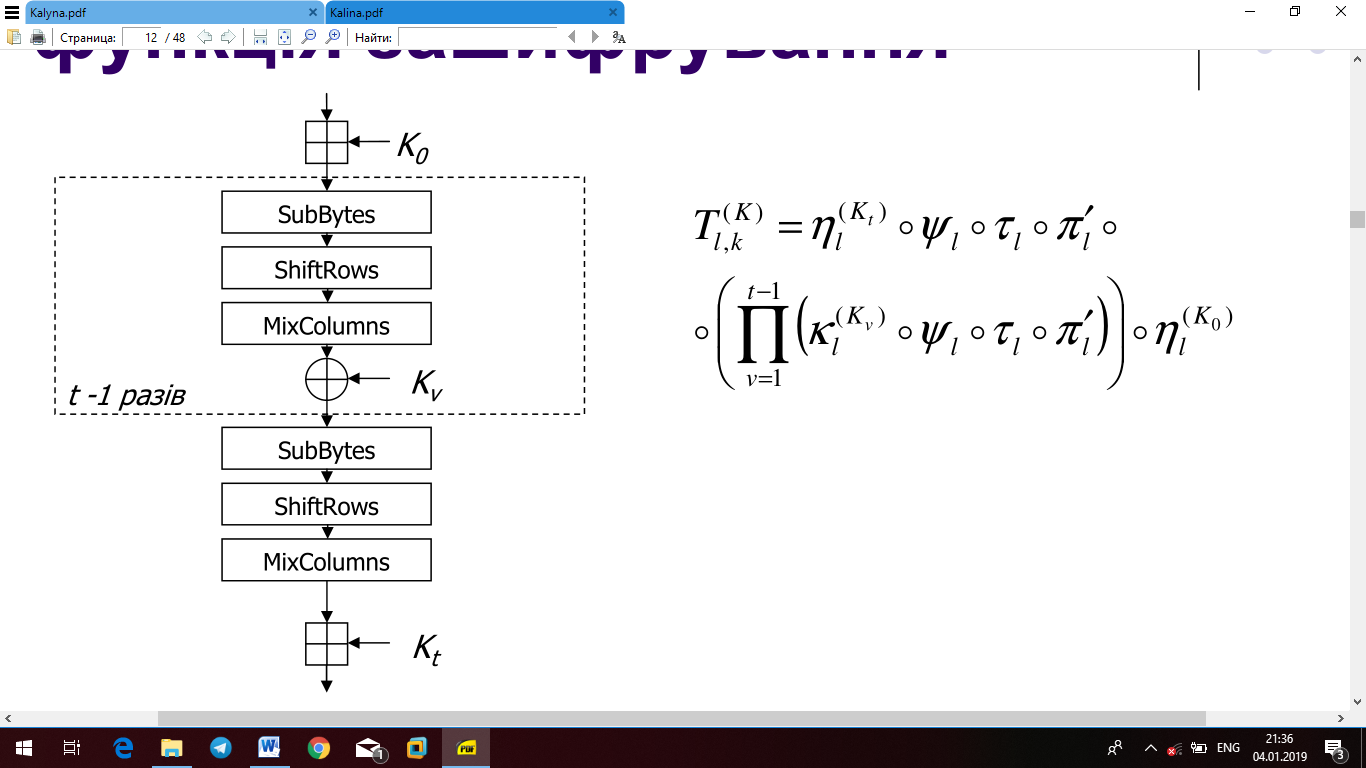


Рисунок 3.2 – Алгоритм шифрування “Калина”

При здійсненні перетворення XORRoundKey (функція в ДСТУ 7624:2014) виконується побітове складання за модулем 2 (XOR) поточного стану та циклового ключа. Після виконання операції результат записується на місце першого аргументу.

Представлення циклового ключа як матриці віповідного розміру є аналогічним представленню відкритого тексту у вигляді поточного стану шифра.

Для поточного стану *А=(ai,j)* та циклового ключа *k=*(*ki,j*) результат перетворення *B=(bi,j)= A*⊕*k* обчислюється за формулою *bi,j = ai,j* ⊕ *ki,j*, *0≤ i <8*, *0≤ j <Nb*. Приклад здійснення перетворення для розміру блока 128 біт представлено на рис. 3.3.

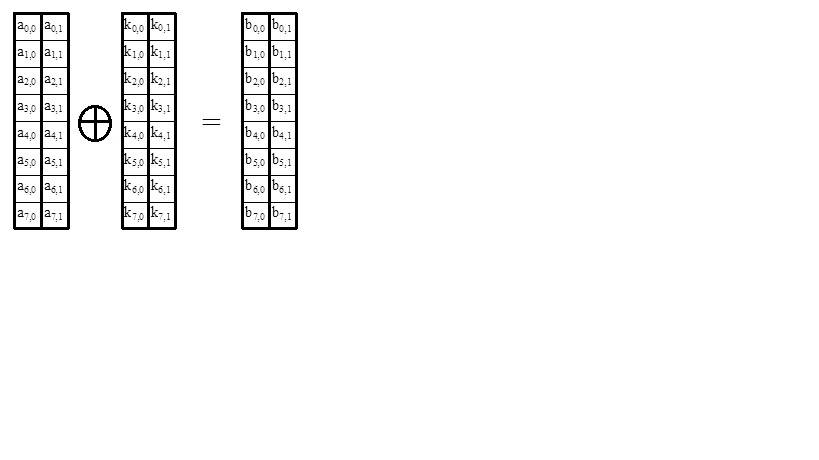


Рисунок 3.3 – Перетворення XORRoundKey для розміру блока 128 біт

При здійсненні Add64RoundKey виконується складання за модулем 64-бітних слів поточного стану та циклового ключа. Після виконання операції результат записується на місце першого аргументу [19] (рис 3.4).



Рисунок 3.4 – Перетворення Add64RoundKey для розміру блока 128 біт

Представлення циклового ключа як матриці віповідного розміру є аналогічним представленню відкритого тексту у вигляді поточного стану шифру. При розбитті поточного стану та циклового ключа на 64-бітні блоки молодшим бітом 64-бітного блоку буде молодший біт байту рядка з найменшим номером, старшим бітом 64-бітного блоку – старший біт байта рядка з найбільшим номером (формат little endian). [18].

Перетворення Kalyna\_S\_boxes виконує заміну кожного байта поточного стану у відповідності з заданою таблицею підстановки. Використовуються 4 різні підстановки “байт-в-байт”, причому для байтів одного рядка поточного стану шифру використовується одна і та ж підстановка:

для байт 0-го рядка ( елементи s 0,i ) – підстановка S0;

для байт 1-го рядка ( елементи s 1,i ) – підстановка S1;

для байт 2-го рядка ( елементи s 2,i ) – підстановка S2;

для байт 3-го рядка ( елементи s 3,i ) – підстановка S3;

для байт 4-го рядка ( елементи s 4,i ) – підстановка S0;

для байт 5-го рядка ( елементи s 5,i ) – підстановка S1;

для байт 6-го рядка ( елементи s 6,i ) – підстановка S2;

для байт 7-го рядка ( елементи s 7,i ) – підстановка S3.



Рисунок 3.5 – Приклад підстановки байта для розміра блоку 256 біт

Заміна одного байту полягає у виборі з таблиці підстановки нового значення за адресою із зсувом, що задає поточне значення байту. Нове вибране значення і є результатом здійснення підстановки для одного байту [20].

Приклад підстановки для таблиці S0 байта з шістнадцятковим значенням 5А наведено у табл. 3.3.

Таблиця 3.3 – Приклад підстановки для таблиці S0

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | A | B | C | D | E | F |
| 0 | A8 | 43 | 5F | 06 | 6B | 75 | 6C | 59 | 71 | DF | 87 | 95 | 17 | F0 | D8 | 09 |
| 1 | 6D | F3 | 1D | CB | C9 | 4D | 2C | AF | 79 | E0 | 97 | FD | 6F | 4B | 45 | 39 |
| 2 | 3E | DD | A3 | 4F | B4 | B6 | 9A | 0E | 1F | BF | 15 | E1 | 49 | D2 | 93 | C6 |
| 3 | 92 | 72 | 9E | 61 | D1 | 63 | FA | EE | F4 | 19 | D5 | AD | 58 | A4 | BB | A1 |
| 4 | DC | F2 | 83 | 37 | 42 | E4 | 7A | 32 | 9C | CC | AB | 4A | 8F | 6E | 04 | 27 |
| 5 | 2E | E7 | E2 | 5A | 96 | 16 | 23 | 2B | C2 | 65 | 66 | 0F | BC | A9 | 47 | 41 |
| 6 | 34 | 48 | FC | B7 | 6A | 88 | A5 | 53 | 86 | F9 | 5B | DB | 38 | 7B | C3 | 1E |
| 7 | 22 | 33 | 24 | 28 | 36 | C7 | B2 | 3B | 8E | 77 | BA | F5 | 14 | 9F | 08 | 55 |
| 8 | 9B | 4C | FE | 60 | 5C | DA | 18 | 46 | CD | 7D | 21 | B0 | 3F | 1B | 89 | FF |
| 9 | EB | 84 | 69 | 3A | 9D | D7 | D3 | 70 | 67 | 40 | B5 | DE | 5D | 30 | 91 | B1 |
| A | 78 | 11 | 01 | E5 | 00 | 68 | 98 | A0 | C5 | 02 | A6 | 74 | 2D | 0B | A2 | 76 |
| B | B3 | BE | CE | BD | AE | E9 | 8A | 31 | 1C | EC | F1 | 99 | 94 | AA | F6 | 26 |
| C | 2F | EF | E8 | 8C | 35 | 03 | D4 | 7F | FB | 05 | C1 | 5E | 90 | 20 | 3D | 82 |
| D | F7 | EA | 0A | 0D | 7E | F8 | 50 | 1ª | C4 | 07 | 57 | B8 | 3C | 62 | E3 | C8 |
| E | AC | 52 | 64 | 10 | D0 | D9 | 13 | 0C | 12 | 29 | 51 | B9 | CF | D6 | 73 | 8D |
| F | 81 | 54 | C0 | ED | 4E | 44 | A7 | 2ª | 85 | 25 | E6 | CA | 7C | 8B | 56 | 80 |

Старші 4 біти визначають рядок, молодші 4 біти – стовбець. Результат підстановки для значення 5А – це число в шістнадцятковому представленні 66, що знаходиться в таблиці на перетині 6-го рядка (з індексом 5) та 11-го стовбця.

Під час виконання перетворення ShiftRows (перестановка елементів в ДСТУ 7624:2014) здійснюється рівномірне розподілення байт кожного 64-бітного стовбця серед інших стовбців. Це досягається шляхом циклічного зсуву рядків стану вправо на різну кількість байт. Значення зсувів залежать від розміру блока шифрування і представлені у табл. 3.4. Рис. 3.6 пояснює порядок виконання перетворення ShiftRows для різних розмірів блоку [20].

Таблиця 3.4 – Значення циклічних зсувів рядків для різних розмірів блоку

|  |  |  |  |
| --- | --- | --- | --- |
| Номер рядка | Значення зсуву, байти | | |
| Довжина блоку 128 біт | Довжина блоку 256 біт | Довжина блоку 512 біт |
| 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 |
| 2 | 0 | 1 | 2 |
| 3 | 0 | 1 | 3 |
| 4 | 1 | 2 | 4 |
| 5 | 1 | 2 | 5 |
| 6 | 1 | 3 | 6 |
| 7 | 1 | 3 | 7 |

а) 128-бітний блок б) 256-бітний блок

Рисунок 3.6 – Порядок розподілення байт першого стовбця при здійсненні перетворення ShiftRows

Під час здійснення перетворення MixColumns (лінійне перетворення в ДСТУ 7624:2014) виконується послідовна обробка всіх стовбців поточного стану. Кожний 8- байтний стовбець розглядається як поліном над полем GF(28), що складається з суми 8 одночленів (кожний байт представляється у вигляді елемента поля GF(28) і є коефіцієнтом перед змінною степеня, що дорівнює індексу байту у стовбці). При перетворенні виконується множення цього полінома за модулем x8+1 на фіксований поліном С(*x*), де

C(x) = {01}x7 +{05}x6 +{01}x5 +{08}x4 +{06}x3 +{07}x2 +{04}x+{01}.

Ця операція еквівалентна матричному множенню над GF(28) початкового 8-байтового вектору на фіксовану матрицю, результат зберігається у 8 байтний вектор b (див. рис. 3.7).

Порядок обчислення елементів результуючого вектору *b* пояснюється на рис. 3.8 при цьому всі операції множення на байт виконуються у полі GF(28).

На рис. 3.9 пояснюється порядок виконання перетворення MixColumns для поточного стану шифру [20].



Рисунок 3.7 – Матричне представлення перемішування у стовбці



Рисунок 3.8 – Порядок обчислення байт при виконанні перемішування



Рисунок 3.9 – Порядок виконання перетворення Mix Columns для поточного стану шифру з розміром блоку 256 біт

Процедура розшифрування є зворотною зашифруванню. На вхід подається шифртекст та циклові ключі. Після закінчення розшифрування отриманий відкритий текст формується у вигляді байтового рядка.

Операція XORRoundKey є зворотною до себе (подвійне застосування дає початкове значення), відповідно, якщо до стану *B=А⊕ k* додати цикловий ключ *k* , то буде отримано початковий стан A .

При зашифруванні та розшифруванні використовується одна й та сама операція XORRoundKey.

Зворотне для Add64RoundKey перетворення Sub64RoundKey полягає в аналогічному розбитті стану та циклового ключа на 64-бітні блоки та відніманні за модулем 264 від блоків стану *B =* (*bi*) відповідних блоків циклового ключа *K =* (*ki*) для отримання стану (результату).

Після виконання операції результат записується на місце першого параметра. Правила переходу від байт поточного стану до 64-бітних блоків та навпаки повністю ідентичні з перетвореннями, що виконуються під час Add64RoundKey [20].

Позначення в алгоритмі

0x – префікс шістнадцяткових літералів

  поле Галуа для поліному 

сума по модулю 2 (XOR) ціла частина числа  довжина бінарної послідовності  функція, якка повертає  менш значимих бітів бінарної послідовності довжиною 

 функція, що повертає  більш значимих бітів бінарної

послідовності довжиною 

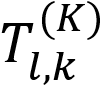
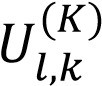
|  |  |
| --- | --- |
| >> | логічний зсув праворуч |
|  |  |
| << | логічний зсув ліворуч |
| >>> | циклічний зсув праворуч |
| <<< | циклічний зсув ліворуч |

+ додавання по модулю 

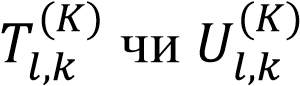
|  |  |
| --- | --- |
| Ⓧ | скалярний добуток векторів |
|  |  |

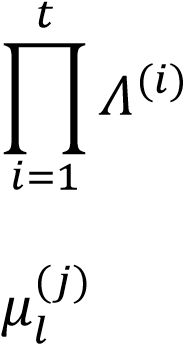
 розмір блоку (128, 256, 512 біт)  розмір ключа (128, 256, 512 біт) (або )  кількість рядків в матриці стану

 вектор розміру 

 базове перетворення при шифруванні  базове перетворення при розшифруванні

 - Конкатенація бінарних послідовностей  послідовне виконання операцій (справа наліво)

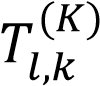
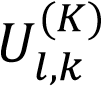
t - число перетворень при виконанні 

 послідовне виконання операцій , , … , ,

починаючи з

представлення невід’ємного цілого числа  у вигляді

бінарної послідовності з  бітів

 виконання  чи  для розміру блоку  та розміру ключа 

## 3.3 Криптографічна стійкість блокового шифру “Калина”

При розробці національного стандарту була виконана оцінка стійкості перспективного криптографічного перетворення щодо різних видів криптоаналітичних атак . Складність найбільш ефективних криптоаналітичних атак при зменшеної кількості циклів (послабленому алгоритмі) і необхідна кількість циклів для забезпечення стійкості наведені у табл.3.5- 3.7 для шифру з розміром блока 128, 256 і 512 бітів [21].

Таблиця 3.5

Результати аналізу стійкості шифру «Калина» із розміром блоку 128 бітів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод криптоаналізу | Найменша кількість циклів, для якої шифр є  стійким |  | Показники атак |  |
| Макс. кілкість циклів | Обчисл. складність, екв. оп. шифрув. | Пам’ять, байтів |
| Диференційний | 5 | 4 | 255 |  |
| Лінійний | 5 | 3 | 252,8 |  |
| Усіч. диференц. | 4 | 3 |  |  |
| Інтегральний | 6 | 5 | 297 | 233+4 |
| Нездійсн. дифер. | 6 | 5 | 262 | 266 |
| Бумеранг | 5 | 4 | 2120 |  |

Таблиця 3.6 Результати аналізу стійкості шифру «Калина» із розміром блоку 256 бітів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод криптоаналізу | Найменша кількість циклів, для якої шифр є  стійким |  | Показники атак |  |
| Макс. кілкість циклів | Обчисл. складність, екв. оп. шифрув. | Пам’ять, байтів |
| Диференційний | 7 | 6 | 2230 |  |
| Лінійний | 7 | 5 | 2220,8 |  |
| Усіч. диференц. | 4 | 3 |  |  |
| Інтегральний | 7 | 6 | 2145 | 264+5 |
| Нездійсн. дифер. | 6 | 5 | 261 | 266 |
| Бумеранг | 6 | 5 | 2220 |  |

Таблиця 3.7

Результати аналізу стійкості шифру «Калина» із розміром блоку 512 бітів

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Метод криптоаналізу | Найменша кількість циклів, для якої шифр є  стійким |  | Показники атак |  |
| Макс. кілкість циклів | Обчисл. складність, екв. оп. шифрув. | Пам’ять, байтів |
| Диференційний | 9 | 8 | 2490 |  |
| Лінійний | 9 | 7 | 2470,4 |  |
| Усіч. диференц. | 4 | 3 |  |  |
| Інтегральний | 7 | 6 | 2137 | 264+5 |
| Нездійсн. дифер. | 6 | 5 | 260 | 266 |
| Бумеранг | 7 | 6 | 2340 |  |

Відповідно, криптографічне перетворення є стійким при 6 циклах для 128-бітового блоку, 7 циклах для 256-бітового і 9 циклах для 512- бітового. Таким чином, шифр, який містить 10, 14 і 18 циклів для розміру блоку 128, 256 і 512 біт відповідно (див. табл. 3.4 – 3.7), забезпечує захист від розглянутих методів криптоаналізу і має достатній запас стійкості.

ШВИДКОДІЯ КРИПТОГРАФІЧНОГО ПЕРЕТВОРЕННЯ.

Тестування було спрямоване на моделювання особливостей роботи засобів криптографічного захисту, що потребують високої швидкодії перетворень (захист IP-трафіку та ін.). Для виключення впливу дискової підсистеми всі дані були розміщені в ОЗП (RAM). Для попередження використання даних, що розміщені тільки в кеш-пам’яті процесору був виділений блок розміром 1 ГБ, який гарантовано у багато разів більший порівняно із наявним доступом кеш-пам’яті будь-якого сучасного процесора, що призведе до необхідності здійснення звернень до основного ОЗП. Для зниження впливу переключення контексту процесора, цей блок пам’яті був перешифрований декілька разів [21]. Вимірювання швидкодії через шифрування однакового обсягу відкритих текстів (режим простої заміни, ECB) виконувалось для блокового шифру „Калина” (всі комбінації розміру блоку і довжини ключа), AES-128, AES-256, ГОСТ 28147-89, СТБ 34.101.31-2011 («БелТ», національний стандарт Білорусії) і шифру «Кузнечик» (проект державного стандарту РФ) в однакових умовах в рамках роботи одного інтерактивного процесу користувача операційної системи. Для отримання найбільшої швидкодії апаратно-незалежної реалізації була обрана мова програмування С++, використаний компілятор gcc version 4.9.2 (Ubuntu 4.9.2-0ubuntu1~12.04, 30- Oct-2014), тестування виконувалось на комп’ютері під управлінням 64-бітової ОС Linux (Ubuntu) з процесором Intel Core i5- 4670@3.40GHz. Результати тестування швидкодії програмної реалізації [21] для версій із найкращою оптимізацією компілятора (-O3 –m64) та порівняння результатів наведені у таблиці 3.8 та на рис. 3.10 [21].

Таблиця 3.8

Швидкодія оптимізованих версій програмної реалізації блокових шифрів

|  |  |  |
| --- | --- | --- |
| № з/п | Блоковий шифр | Швидкодія, Мбіт/с |
| 1 | Kalyna-128/128 | 2611.77 |
| 2 | Kalyna-128/256 | 1779.52 |
| 3 | Kalyna-256/256 | 2017.97 |
| 4 | Kalyna-256/512 | 1560.89 |
| 5 | Kalyna-512/512 | 1386.46 |
| 6 | AES-128 | 2525.89 |
| 7 | AES-256 | 1993.53 |
| 8 | GOST 28147-89 | 639.18 |
| 9 | STB 34.101.31-2011(BelT) | 1055.92 |
| 10 | Kuznyechik | 1081.08 |

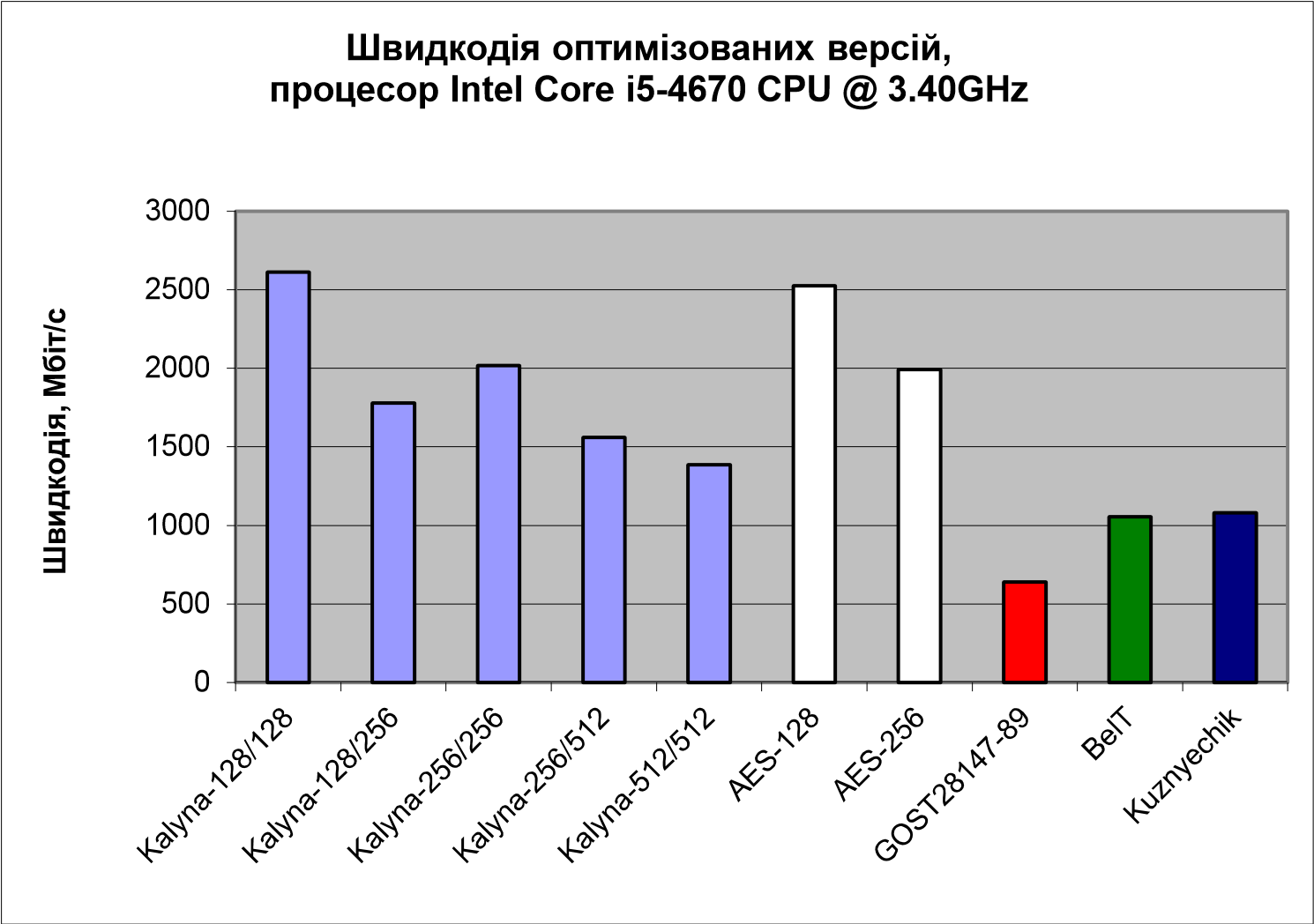


Рис 3.10 - Швидкодія оптимізованих версій програмної реалізації блокових шифрів

## Таким чином, для оптимізованих версій на 64-бітовій платформі: – для 128-бітової довжини ключа швидкодія «Калини» вища за AES на 3% (86 Мбіт/с); – для 256-бітової довжини ключа швидкодія «Калини» повільніша за AES на 10% (для 128- бітового блоку) та швидше на 1% (для 256- бітового блоку) [21].;

– швидкодія «Калини» при відповідній довжині ключа вища за ГОСТ 28147-89 у 2,8 рази (для 128-бітового блоку) і 3,16 рази (для 256- бітового блоку), і приблизно у 2 рази вища, ніж у нових стандартів шифрування Білорусії і Росії. Без фіксації розміру блоку Калина безумовно забезпечує більш швидке перетворення на тієї ж самої довжині ключа порівняно із AES. Крім того, швидкісні показники та їх співвідношення для різних шифрів можуть достатньо суттєво змінюватися в залежності від версії компілятора (наприклад, використання нової версії компілятора gcc 4.9.2 замість 4.8.2 дозволило підвищити швидкість шифрування майже на 80 Мб/с. Для порівняння програмних реалізації на інших платформах (мікроконтролерах, смарткартах та ін.), де компілятори не реалізують надскладний модуль оптимізації, було виконано порівняння із відімкненими відповідними параметрами. Результати, що отримаються в цьому випадку, є, зазвичай, повільнішими ніж при включеній оптимізації. Водночас, такі результати корисні при оцінці співвідношення швидкодії при реалізації на асемблері та відповідній розробці апаратного-програмного модулю. В цьому випадку перевага швидкодії «Калини» над AES складає від 12,5% до 27% в залежності від розміру блока та довжини ключа. У цих же умовах Калина виконує обробку швидше ніж ГОСТ 28147-89 від 3,17 до 3,72 разів. Додатково слід зазначити, що «Калина» забезпечує суттєво більш високий запас стійкости до криптоаналітичних атак, ніж AES [21].

## 3.4 Висновки до розділу

В даному розділі було наведено поняття блокового алгоритму шифрування та розглянено алгоритм роботи вітчизняного алгоритму шифрування. Звернено увагу на переваги використання БСШ, які є основним криптографічним механізмом забезпечення конфіденційності та цілісності, а також захисту інформації та інформаційних ресурсів від НСД.

Безумовно, прийняття національного стандарту БСШ сприятиме підвищенню рівня інформаційної безпеки України. Серед переваг прийняття національного стандарту – впровадження в Україні сучасного криптографічного БСШ, застосування якого допоможе забезпечити необхідну захищеність інформації та інформаційних ресурсів у частині послуг конфіденційності та цілісності, зокрема з обмеженим доступом. Введення в дію нового національного стандарту ДСТУ 7624:2014 дозволить суттєво вдосконалити показники ефективності захисту систем (більше ніж у 3 рази порівняно із старим гостом ГОСТ 28147-89), засобів і протоколів криптографічного захисту інформації, які розробляються в Україні, а в деяких випадках зробити їх суттєво кращими ніж наявні та перспективні світові рішення.

.В алгоритмі реалізована нова односпрямована конструкція схеми розгортання ключів, що забезпечує стійкість і використовує циклове перетворення до переборних атак та відомих методів аналізу, які орієнтовані на схему розгортання ключів, так і додатковий захист від низки методів криптоаналізу, спрямованого, в тому числі, і на апаратну або програмну реалізацію перетворення. Обчислювальна і просторова складність формування циклових ключів однакова.

# РОЗДІЛ 4

# РЕАЛІЗАЦІЯ АЛГОРИТМУ ДСТУ 7624:2014 НА ПЛІС

У четвертому розділі показано можливість програмно-апаратної реалізації даного алгоритму шифрування для блоку 128 біт та епюри виконуваного коду, написаного мовою AHDL. Також використано інтегроване програмне середовище MAX+PLUS II.

Алгоритм криптографічного перетворення ДСТУ 7624:2014 призначений для програмно-апаратної реалізації, задовольняє криптографічним вимогам і за своїми можливостями не накладає обмежень на ступінь секретності інформації.

* 1. Принципова схема та початкове заповнення шифра

В даній роботі будуть реалізовані основні перетворення на яких базується алгоритм ДСТУ 7624:2014. Такі перетворення як XORRoundKey та AddRoundKey, перетворення SubBytes, ShiftRows, MixColumns. Ці перетворення складають ітеративне циклове перетворення та завдяки ним забезпечуються високі криптографічні властивості. Принципова схема перетворень, які надалі будуть реалізовані зображена на рис. 4.1.

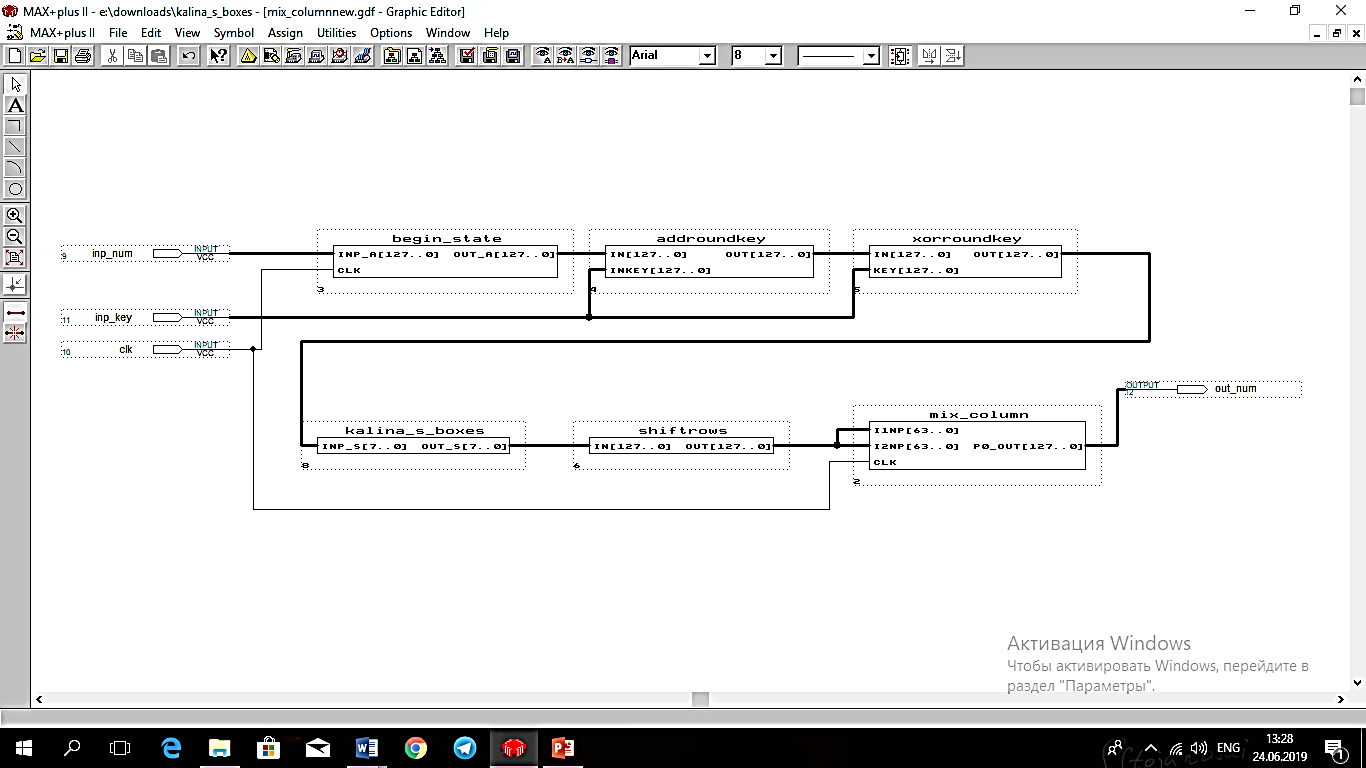


Рисунок 4.1 – Принципова схема основних перетворень

При виконанні зашифрування або розшифрування операції виконуються над двомірним масивом байт (поточним станом шифра). Поточний стан можна представити у вигляді матриці розмірністю *8 ×* *Nb* байт (вісім рядків довжиною *Nb* байт). Кожний байт в поточному стані має два індекси: номер рядка (*r*, 0 ≤ *r* < 8) і номер стовбця (*c*, 0 ≤ *с* < *Nb*). Байт адресується як *sr,c* або *s*[*r,c*].

Крім того, у ряді операцій поточний стан шифру  розглядається як послідовність 64-бітних слів . У цьому випадку кожний стовбець з номером , , розглядається як окреме слово, що має значення . Відповідно, молодшим бітом 64-бітного блока є молодший біт байта рядка з найменшим номером, старшим бітом 64-бітного блока – старший біт байта із рядка з найбільшим номером.

До початку процедури зашифрування відкритий текст копіюється в поточний стан шифра, а після завершення зашифрування шифртекст копіюється з поточного стану. На рисунку 4.2. показано заповнення однієї ячейки поточного стану. Повне заповнення 128 біт зображено на рис.3.3.

SUBDESIGN begin\_state

( inp\_a[127..0],clk:INPUT;

out\_a[127..0]:OUTPUT; )

VARIABLE

ff[127..0]:dff;

BEGIN

ff[127..0].d = inp\_a[127..0];

out\_a[127..0] = ff[127..0].q;

ff[127..0].clk = clk;

END;

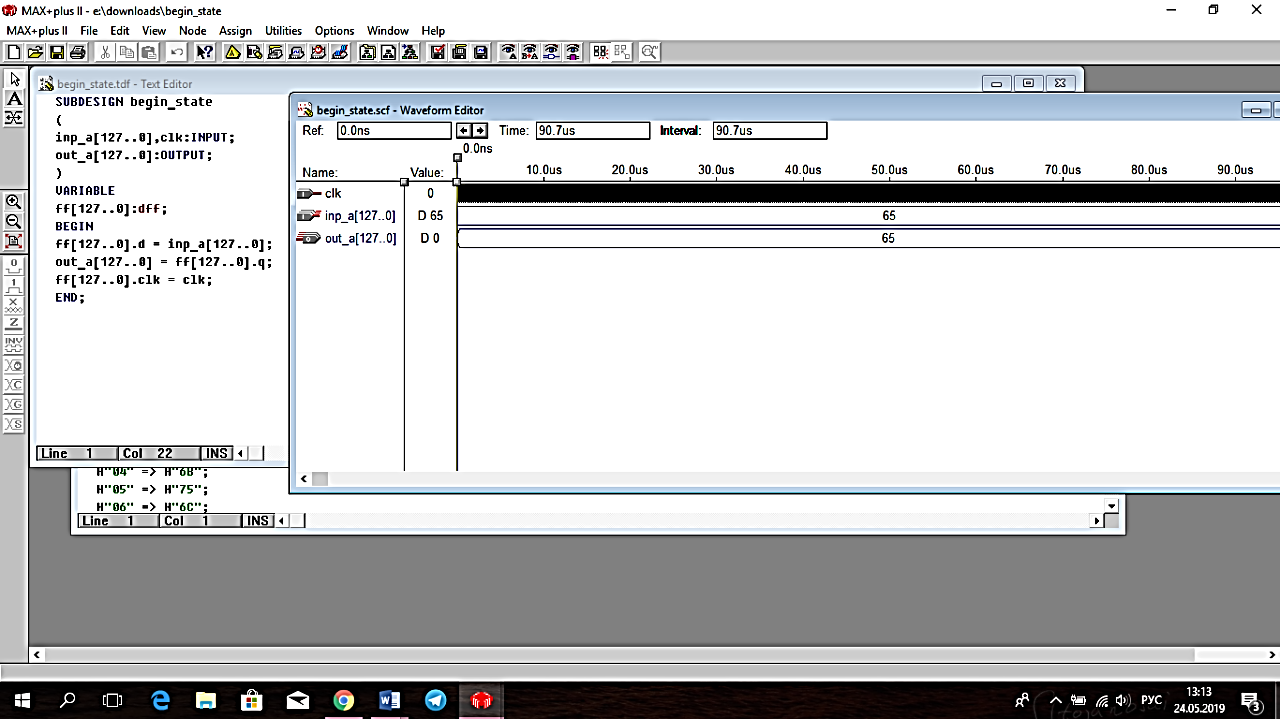


Рисунок 4.2 – Заповнення поточного стану

* 1. Перетворення XORRoundKey та AddRoundKey для 128 бітного блоку

Перетворення XORRoundKey виконує побітове складання за модулем 2 (XOR) поточного стану та циклового ключа. Після виконання операції результат записується на місце першого аргументу. На рис.4.3 зображені епюри виконання даного програмного коду.

subdesign XORroundKey

( in[127..0], clk, sum, upr :input;

out[127..0]: output; )

variable

reg1[127..0] : dff;

reg2[127..0] : dff;

begin

reg1[].d=in[]; reg2[].d=in[];

if upr&sum then

reg1[].clk=clk;

elsif upr then

reg2[].clk=clk;

ELSE

out[] = reg1[].q $ reg2[].q;

end if;

end;

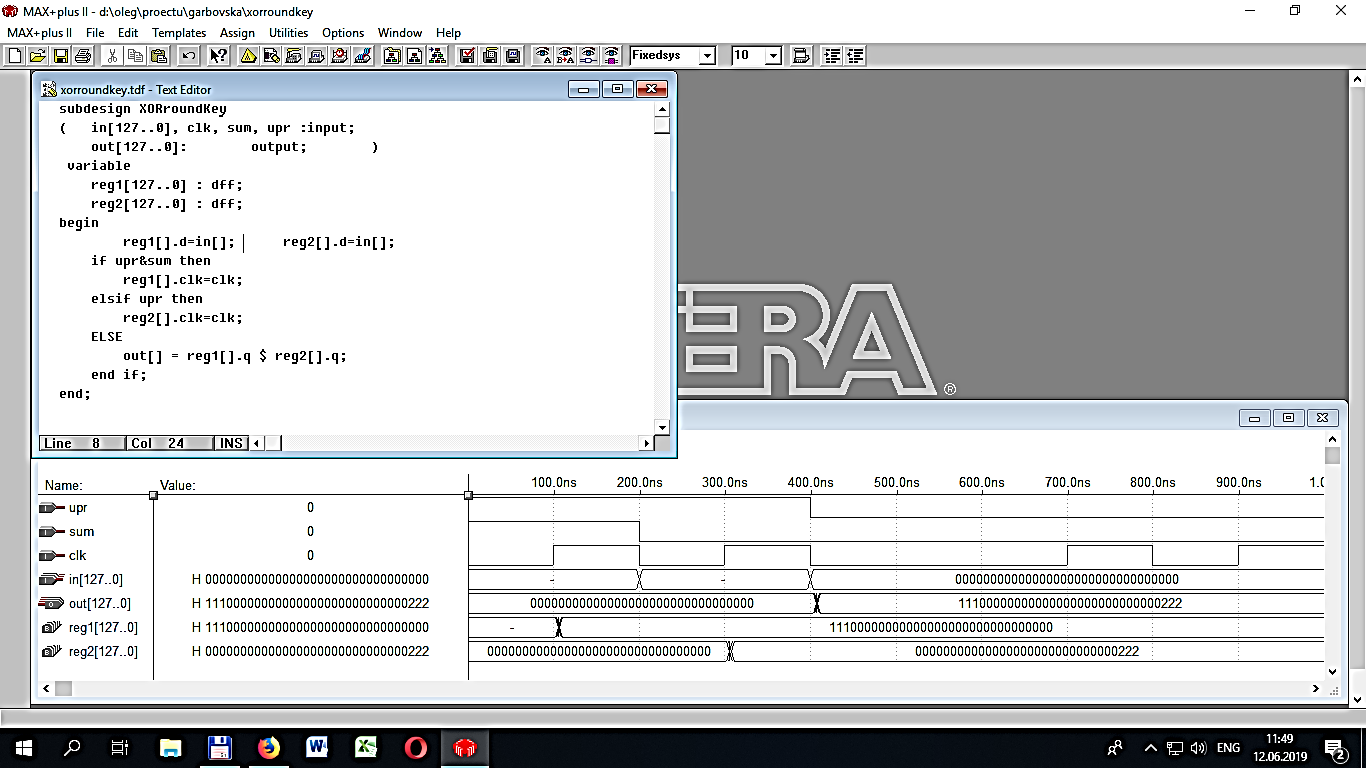


Рисунок 4.3 – Епюри виконання XORRoundKey

Перетворення AddRoundKey виконує побітове складання за модулем 264 (Add) 64-бітних слів поточного стану та циклового ключа. Після виконання операції результат записується на місце першого аргументу. Схема розбиття на 64-бітні блоки 128-бітного стану та циклового ключа такого ж розміру з наступним перетворенням зображено на рис.3.4. На рис.4.4 зображені епюри виконання даного програмного коду.

subdesign Addroundkey

( in[127..0], inkey[127..0]:input;

out[127..0]:output; )

begin

out[63..0] = in[63..0] + inkey[63..0];

out[127..64] = in[127..64] + inkey[127..64];

end;

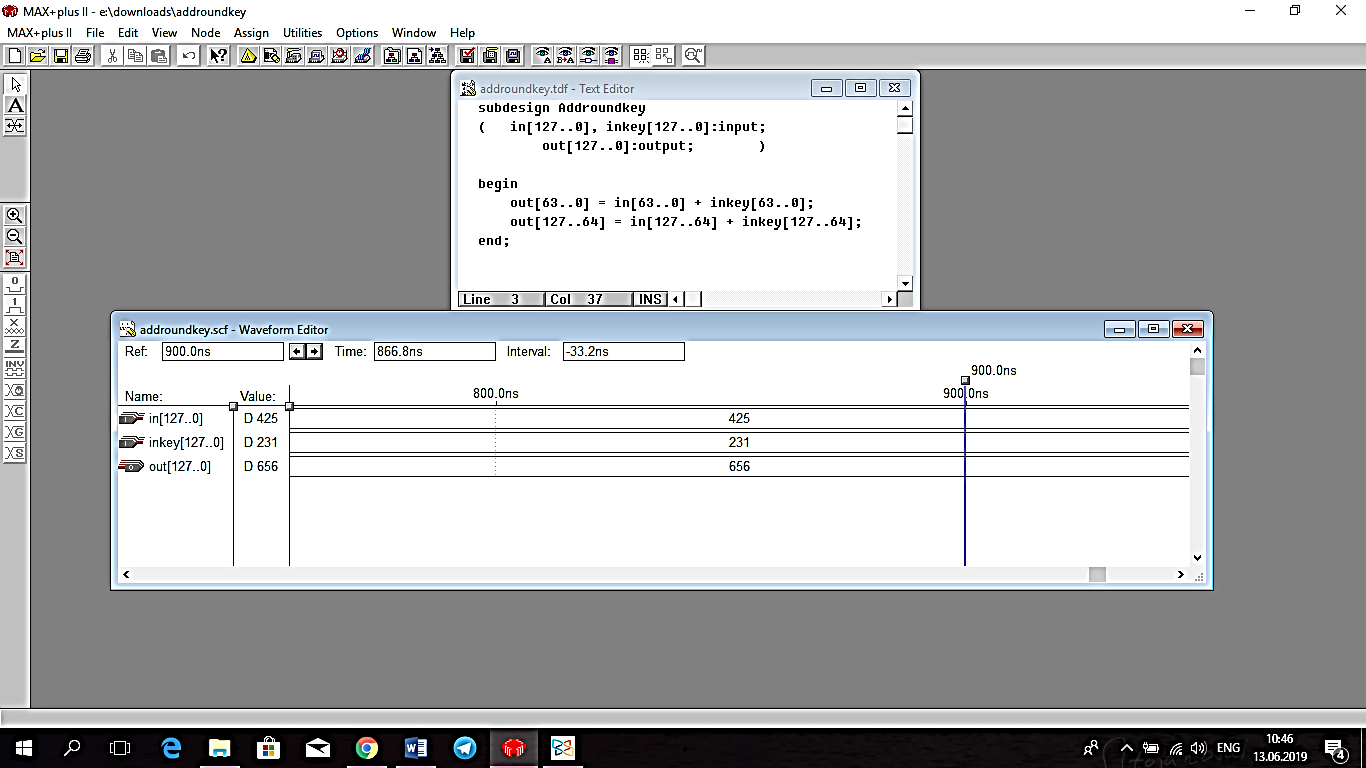


Рисунок 4.4 – Епюри виконання AddRoundKey

* 1. Перетворення SubBytes

Дане перетворення дозволяє виконувати заміну кожного байта поточного стану у відповідності з заданою таблицею підстановки. Існує 4 різні таблиці підстановки S0, S1, S2, S3, але для байтів одного рядка поточного стану використовується одна і та ж підстановка. Програмний код даного перетворення представлено нижче. На рис.4.5 показано приклад підстановки для таблиці S0, що зображена у табл. 3.3.

SUBDESIGN kalina\_S\_QQ\_boxes( inp\_S[7..0]:INPUT; out\_S[7..0]:OUTPUT; )

BEGIN

TABLE

inp\_S[7..0] => out\_S[7..0];

H"00" => H"A8"; H"01" => H"43"; H"02" => H"5F"; H"03" => H"06";

H"04" => H"6B"; H"05" => H"75"; H"06" => H"6C"; H"07" => H"59";

H"08" => H"71"; H"09" => H"DF"; H"0A" => H"87"; H"0B" => H"95";

H"0C" => H"17"; H"0D" => H"F0"; H"0E" => H"D8"; H"0F" => H"09";

H"10" => H"6D"; H"11" => H"F3"; H"12" => H"1D"; H"13" => H"CB";

H"14" => H"C9"; H"15" => H"4D"; H"16" => H"2C"; H"17" => H"AF";

H"18" => H"79"; H"19" => H"E0"; H"1A" => H"97"; H"1B" => H"FD";

H"1C" => H"6F"; H"1D" => H"4B"; H"1E" => H"45"; H"1F" => H"39";

H"20" => H"3E"; H"21" => H"DD"; H"22" => H"A3"; H"23" => H"4F";

H"24" => H"B4"; H"25" => H"B6"; H"26" => H"9A"; H"27" => H"0E";

H"28" => H"1F"; H"29" => H"BF"; H"2A" => H"15"; H"2B" => H"E1";

H"2C" => H"49"; H"2D" => H"D2"; H"2E" => H"93"; H"2F" => H"C6";

H"30" => H"92"; H"31" => H"72"; H"32" => H"9E"; H"33" => H"61";

H"34" => H"D1"; H"35" => H"63"; H"36" => H"FA"; H"37" => H"EE";

H"38" => H"F4"; H"39" => H"19"; H"3A" => H"D5"; H"3B" => H"AD";

H"3C" => H"58"; H"3D" => H"A4"; H"3E" => H"BB"; H"3F" => H"A1";

H"40" => H"DC"; H"41" => H"F2"; H"42" => H"83"; H"43" => H"37";

H"44" => H"42"; H"45" => H"E2"; H"46" => H"7A"; H"47" => H"32";

H"48" => H"9C"; H"49" => H"CC"; H"4A" => H"AB"; H"4B" => H"4A";

H"4C" => H"8F"; H"4D" => H"6E"; H"4E" => H"04"; H"4F" => H"27";

H"50" => H"2E"; H"51" => H"E7"; H"52" => H"E2"; H"53" => H"5A";

H"54" => H"96"; H"55" => H"16"; H"56" => H"23"; H"57" => H"2B";

H"58" => H"C2"; H"59" => H"65"; H"5A" => H"66"; H"5B" => H"0F";

H"5C" => H"BC"; H"5D" => H"A9"; H"5E" => H"47"; H"5F" => H"41";

H"60" => H"34"; H"61" => H"48"; H"62" => H"FC"; H"63" => H"B7";

H"64" => H"6A"; H"65" => H"88"; H"66" => H"A5"; H"67" => H"53";

H"68" => H"86"; H"69" => H"F9"; H"6A" => H"5B"; H"6B" => H"DB";

H"6C" => H"38"; H"6D" => H"7B"; H"6E" => H"C3"; H"6F" => H"E1";

H"70" => H"22"; H"71" => H"33"; H"72" => H"24"; H"73" => H"28";

H"74" => H"36"; H"75" => H"C7"; H"76" => H"B2"; H"77" => H"3B";

H"78" => H"8E"; H"79" => H"77"; H"7A" => H"BA"; H"7B" => H"F5";

H"7C" => H"14"; H"7D" => H"9F"; H"7E" => H"08"; H"7F" => H"55";

H"80" => H"9B"; H"81" => H"4C"; H"82" => H"FE"; H"83" => H"60";

H"84" => H"5C"; H"85" => H"DA"; H"86" => H"18"; H"87" => H"46";

H"88" => H"CD"; H"89" => H"7D"; H"8A" => H"21"; H"8B" => H"B0";

H"8C" => H"3F"; H"8D" => H"1B"; H"8E" => H"89"; H"8F" => H"FF";

H"90" => H"EB"; H"91" => H"84"; H"92" => H"69"; H"93" => H"3A";

H"94" => H"9D"; H"95" => H"D7"; H"96" => H"D3"; H"97" => H"70";

H"98" => H"67"; H"99" => H"40"; H"9A" => H"B5"; H"9B" => H"DE";

H"9C" => H"5D"; H"9D" => H"30"; H"9E" => H"91"; H"9F" => H"B1";

H"A0" => H"78"; H"A1" => H"11"; H"A2" => H"01"; H"A3" => H"E5";

H"A4" => H"00"; H"A5" => H"68"; H"A6" => H"98"; H"A7" => H"A0";

H"A8" => H"C5"; H"A9" => H"02"; H"AA" => H"A6"; H"AB" => H"74";

H"AC" => H"2D"; H"AD" => H"0B"; H"AE" => H"A2"; H"AF" => H"76";

H"B0" => H"B3"; H"B1" => H"BE"; H"B2" => H"CE"; H"B3" => H"BD";

H"B4" => H"AE"; H"B5" => H"E9"; H"B6" => H"8A"; H"B7" => H"31";

H"B8" => H"1C"; H"B9" => H"EC"; H"BA" => H"F1"; H"BB" => H"99";

H"BC" => H"94"; H"BD" => H"AA"; H"BE" => H"F6"; H"BF" => H"26";

H"C0" => H"2F"; H"C1" => H"EF"; H"C2" => H"E8"; H"C3" => H"8C";

H"C4" => H"35"; H"C5" => H"03"; H"C6" => H"D4"; H"C7" => H"7F";

H"C8" => H"FB"; H"C9" => H"05"; H"CA" => H"C1"; H"CB" => H"5E";

H"CC" => H"90"; H"CD" => H"20"; H"CE" => H"3D"; H"CF" => H"82";

H"D0" => H"F7"; H"D1" => H"EA"; H"D2" => H"0A"; H"D3" => H"0D";

H"D4" => H"7E"; H"D5" => H"F8"; H"D6" => H"50"; H"D7" => H"1A";

H"D8" => H"1A"; H"D9" => H"C4"; H"DA" => H"07"; H"DB" => H"B8";

H"DC" => H"3C"; H"DD" => H"62"; H"DE" => H"E2"; H"DF" => H"C8";

H"E0" => H"AC"; H"E1" => H"52"; H"E2" => H"64"; H"E3" => H"10";

H"E4" => H"D0"; H"E5" => H"D9"; H"E6" => H"13"; H"E7" => H"0C";

H"E8" => H"12"; H"E9" => H"29"; H"EA" => H"51"; H"EB" => H"B9";

H"EC" => H"CF"; H"ED" => H"D6"; H"EE" => H"73"; H"EF" => H"8D";

H"F0" => H"81"; H"F1" => H"54"; H"F2" => H"C0"; H"F3" => H"ED";

H"F4" => H"4E"; H"F5" => H"44"; H"F6" => H"A7"; H"F7" => H"2A";

H"F8" => H"85"; H"F9" => H"25"; H"FA" => H"E6"; H"FB" => H"CA";

H"FC" => H"7C"; H"FD" => H"8B"; H"FE" => H"56"; H"FF" => H"80";

END TABLE;

END;

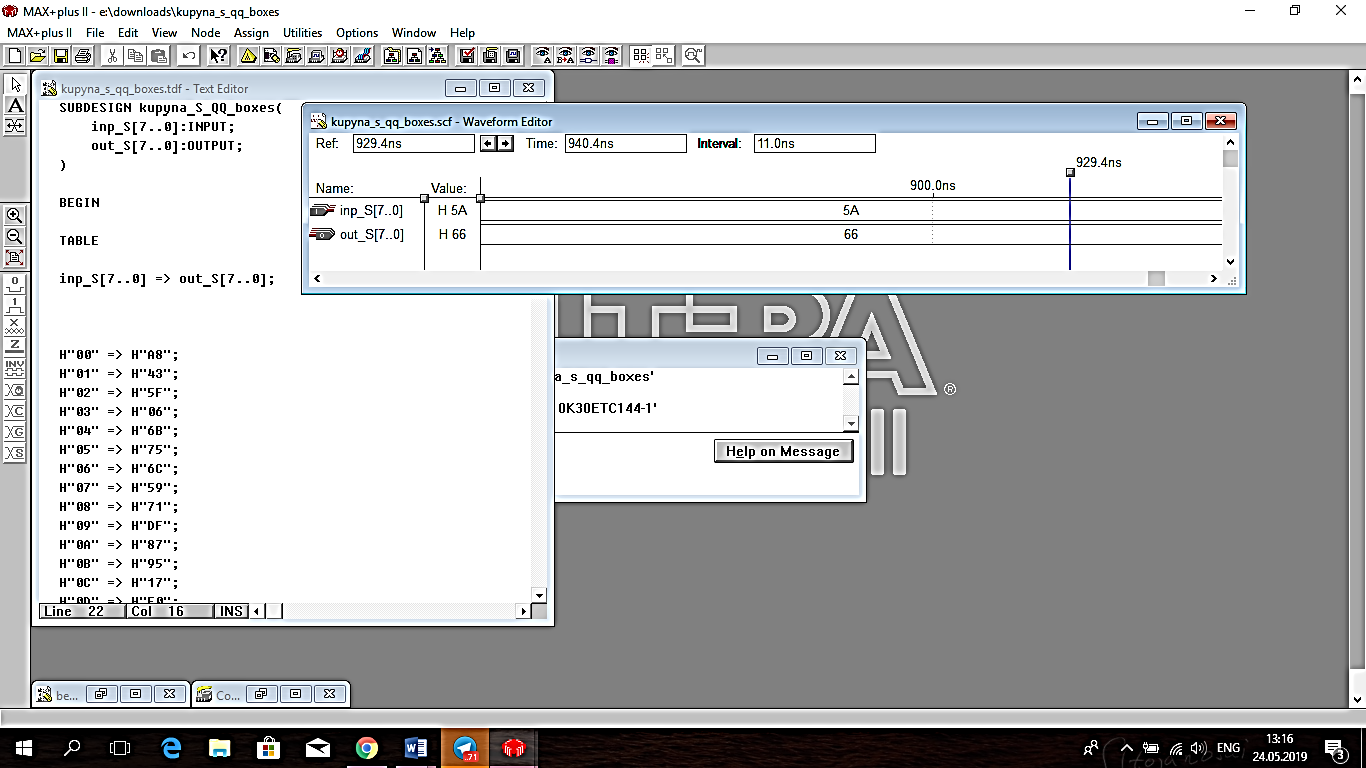


Рисунок 4.5 – Епюри перетворення Kalyna\_S\_boxes

* 1. Перетворення ShiftRows

Перетворення ShiftRows досягається шляхом зсуву рядків на різну кількість байт. Значення зсуву залежить від розміру блока шифрування. В даний роботі програмний код дозволяє виконати перетворення для 128-бітного блоку. Сам алгоритм зсуву для даного блоку представлений на рис. 3.9(а). На рис. 4.6 зображені епюри виконуваного коду.

**subdesign shiftrows**

( in[127..0], clk, load :input;

out[127..0]:output; )

variable reg[127..0]: dff;

begin

reg[].clk=clk; reg[].d=in[];

out[96..127]=reg[96..127].q&load;

out[64..95]=reg[0..31].q&load;

out[32..63]=reg[32..63].q&load;

out[0..31]=reg[64..95].q&load;

end

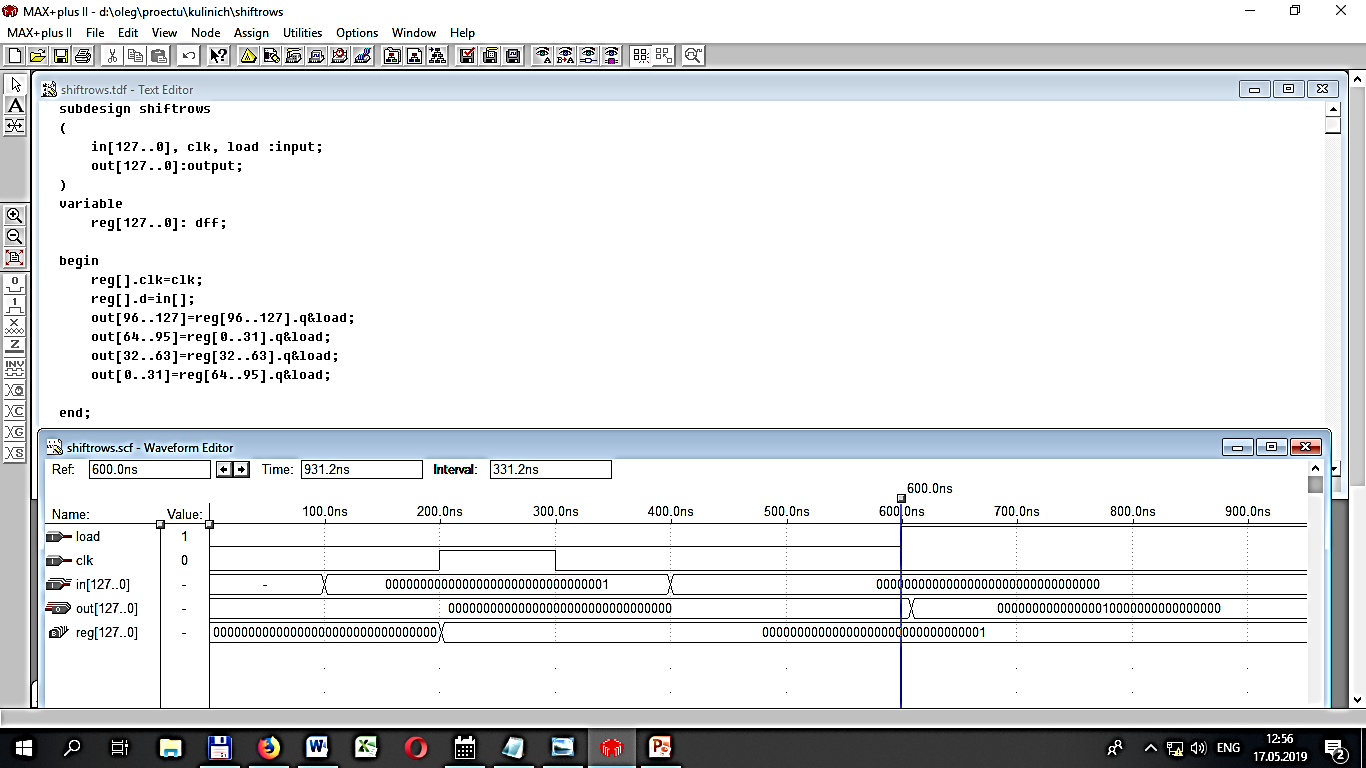


Рисунок 4.6 – Епюри перетворення ShiftRows

* 1. Перетворення MixColumns

Для реалізації блоку лінійного розсіювання було обране множення на МДВ-матрицю як найбільш ефективний метод реалізації впливу кожного вхідного символу на кожний вихідний завдяки отриманню найбільшого індексу галуження (branch number) відображення. Для блокового шифру МДВ-перетворення може бути реалізовано двома способами: - розмір вектора, який умножається на МДВ матрицю, співпадає з розміром блоку шифру і проміжних значень між циклами шифрування; - значення, що обробляється, розділяється на декілька блоків одного розміру, кожен з яких інтерпретується як вектор стовпець, що множиться на МДВ-матрицю розміру, меншого розміру блоку шифру. Підвищення розміру МДВ-матриці призводить до покращення криптографічних властивостей циклового перетворення. Найбільш ефективною є реалізація цієї операції у вигляді табличного перетворення (look-up tables), і всі таблиці, що використовуються при шифруванні, повинні уміщуватися до кешу L1 сучасних процесорів [19]. Таким чином, при виборі розміру МДВ матриці необхідно враховувати не тільки кількість операцій, що забезпечують достатні криптографічні властивості, але й обсяг пам’яті, необхідний для швидкісної програмної реалізації. Для блокового шифра “Калина” була обрана МДВ-матриця розміром 64х64 біта (8x8 над полем GF(28)) як така, що забезпечує необхідні криптографічні властивості і вимоги щодо швидкодії на сучасних програмних 64-бітових архітектурах, – настольних і серверних системах та мобільних пристроях. На рис. 4.7 зображені епюри виконуваного коду. Далі зображений програмний код даного перетворення:

**SUBDESIGN timer\_mix**

( clk\_T:INPUT;

OUTTIME[9..0]:OUTPUT; )

VARIABLE

ff1[9..0]:tff;

BEGIN

ff1[9..0].t = VCC;

ff1[0].clk = clk\_T;

ff1[9..1].clk = ff1[8..0].q;

OUTTIME[9..0] = !ff1[9..0].q;

end;

**SUBDESIGN multi**

include "plus";

( multi\_a[7..0],multi\_b[7..0]:INPUT; multi\_c[15..0]:OUTPUT; )

VARIABLE

plusik[6..0]:plus;

BEGIN

plusik[6..0].vhodplus\_b = 0; plusik[6..0].vhodplus\_a = 0;

plusik[6..1].vhodplus\_a[15..0] = plusik[5..0].vihodplus\_c[15..0];

plusik0.vhodplus\_a[7..0] = multi\_b0 & multi\_a[7..0];

plusik0.vhodplus\_b[8..1] = multi\_b1 & multi\_a[7..0];

plusik1.vhodplus\_b[9..2] = multi\_b2 & multi\_a[7..0];

plusik2.vhodplus\_b[10..3] = multi\_b3 & multi\_a[7..0];

plusik3.vhodplus\_b[11..4] = multi\_b4 & multi\_a[7..0];

plusik4.vhodplus\_b[12..5] = multi\_b5 & multi\_a[7..0];

plusik5.vhodplus\_b[13..6] = multi\_b6 & multi\_a[7..0];

plusik6.vhodplus\_b[14..7] = multi\_b7 & multi\_a[7..0];

multi\_c[15..0] = plusik6.vihodplus\_c[15..0];

END;

**Plus**

include "napivsuma ";

include "povnasuma";

SUBDESIGN plus

( vhodplus\_a[15..0], vhodplus\_b[15..0]:INPUT;

vihodplus\_c[15..0]:OUTPUT; )

VARIABLE

sumat[14..0]:povnasuma;

napiv:napivsuma;

BEGIN

napiv.napivhod\_a = vhodplus\_a0; napiv.napivhod\_b = vhodplus\_b0;

sumat[0].vhod\_peren = napiv.perennapiv\_ab;

sumat[14..1].vhod\_peren = sumat[13..0].perenos\_ab;

sumat[14..0].vhod\_a = vhodplus\_a[15..1];

sumat[14..0].vhod\_b = vhodplus\_b[15..1];

vihodplus\_c0 = napiv.vihodnapiv\_c;

vihodplus\_c[15..1] = sumat[14..0].vihod\_c;

END;

**SUBDESIGN napivsuma**

( napivhod\_a,napivhod\_b:INPUT; vihodnapiv\_c,perennapiv\_ab:OUTPUT; )

BEGIN

vihodnapiv\_c = napivhod\_a & !napivhod\_b # !napivhod\_a & napivhod\_b;

perennapiv\_ab = napivhod\_b & napivhod\_a;

END;

**SUBDESIGN povnasuma**

( vhod\_peren,vhod\_a,vhod\_b:INPUT; vihod\_c,perenos\_ab:OUTPUT; )

BEGIN

vihod\_c = !vhod\_peren & !vhod\_a & vhod\_b # !vhod\_peren & vhod\_a & !vhod\_b # vhod\_peren & !vhod\_a & !vhod\_b # vhod\_peren & vhod\_a & vhod\_b; perenos\_ab = !vhod\_peren & vhod\_a & vhod\_b # vhod\_peren & !vhod\_a & vhod\_b # vhod\_peren & vhod\_a & !vhod\_b # vhod\_peren & vhod\_a & vhod\_b;

END;

**MixColumn**

INCLUDE "multi"; INCLUDE "timer\_mix";

CONSTANT H\_4 = H"04";

CONSTANT H\_5 = H"05";

CONSTANT H\_6 = H"06";

CONSTANT H\_7 = H"07";

CONSTANT H\_8 = H"08";

SUBDESIGN mix\_column ( i1np[63..0],i2np[63..0],clk:INPUT;

P0\_out[127..0]:OUTPUT; )

VARIABLE

t:timer\_mix; m[19..0]:multi; reg[63..0]:dff; re1g[63..0]:dff;

BEGIN

t.clk\_T = clk;

if t.OUTTIME[9..0] == 1 then

m0.multi\_a[7..0] = i1np[23..16]; m0.multi\_b[7..0] = H\_5;

m1.multi\_a[7..0] = i1np[39..32]; m1.multi\_b[7..0] = H\_8;

m2.multi\_a[7..0] = i1np[47..40]; m2.multi\_b[7..0] = H\_5;

m3.multi\_a[7..0] = i1np[55..48]; m3.multi\_b[7..0] = H\_7;

m4.multi\_a[7..0] = i1np[63..56]; m4.multi\_b[7..0] = H\_4;

reg[7..0] = i1np[7..0] $ i1np[15..8] $ m0.multi\_c[7..0] $ i1np[31..24] $

m1.multi\_c[7..0] $ m2.multi\_c[7..0] $ m3.multi\_c[7..0] $ m4.multi\_c[7..0];

m5.multi\_a[7..0] = i1np[7..0]; m5.multi\_b[7..0] = H\_4;

m6.multi\_a[7..0] = i1np[31..24]; m6.multi\_b[7..0] = H\_5;

m7.multi\_a[7..0] = i1np[47..40]; m7.multi\_b[7..0] = H\_8;

m8.multi\_a[7..0] = i1np[55..48]; m8.multi\_b[7..0] = H\_6;

m9.multi\_a[7..0] = i1np[63..56]; m9.multi\_b[7..0] = H\_7;

reg[15..8] = m5.multi\_c[7..0] $ i1np[15..8] $ i1np[23..16] $ m6.multi\_c[7..0] $

i1np[39..32] $ m7.multi\_c[7..0] $ m8.multi\_c[7..0] $ m9.multi\_c[7..0];

m10.multi\_a[7..0] = i1np[7..0]; m10.multi\_b[7..0] = H\_7;

m11.multi\_a[7..0] = i1np[15..8]; m11.multi\_b[7..0] = H\_4;

m12.multi\_a[7..0] = i1np[39..32]; m12.multi\_b[7..0] = H\_5;

m13.multi\_a[7..0] = i1np[55..48]; m13.multi\_b[7..0] = H\_8;

m14.multi\_a[7..0] = i1np[63..56]; m14.multi\_b[7..0] = H\_6;

reg[23..16] = m10.multi\_c[7..0] $ m11.multi\_c[7..0] $ i1np[23..16] $ i1np[31..24] $ m12.multi\_c[7..0] $ i1np[47..40] $ m13.multi\_c[7..0] $ m14.multi\_c[7..0];

m15.multi\_a[7..0] = i1np[7..0]; m15.multi\_b[7..0] = H\_6;

m16.multi\_a[7..0] = i1np[15..8]; m16.multi\_b[7..0] = H\_7;

m17.multi\_a[7..0] = i1np[23..16]; m17.multi\_b[7..0] = H\_4;

m18.multi\_a[7..0] = i1np[47..40]; m18.multi\_b[7..0] = H\_5;

m19.multi\_a[7..0] = i1np[63..56]; m19.multi\_b[7..0] = H\_8;

reg[31..24] = m15.multi\_c[7..0] $ m16.multi\_c[7..0] $ m17.multi\_c[7..0] $ i1np[31..24] $ i1np[39..32] $ m18.multi\_c[7..0] $ i1np[55..48] $ m19.multi\_c[7..0];

reg[31..0].clk = clk;

elsif t.OUTTIME[9..0] == 2 then

m0.multi\_a[7..0] = i1np[7..0]; m0.multi\_b[7..0] = H\_8;

m1.multi\_a[7..0] = i1np[15..8]; m1.multi\_b[7..0] = H\_6;

m2.multi\_a[7..0] = i1np[23..16]; m2.multi\_b[7..0] = H\_7;

m3.multi\_a[7..0] = i1np[31..24]; m3.multi\_b[7..0] = H\_4;

m4.multi\_a[7..0] = i1np[47..40]; m4.multi\_b[7..0] = H\_5;

reg[39..32] = m0.multi\_c[7..0] $ m1.multi\_c[7..0] $ m2.multi\_c[7..0] $ m3.multi\_c[7..0] $ i1np[39..32] $ i1np[47..40] $ m4.multi\_c[7..0] $ i1np[63..56];

m5.multi\_a[7..0] = i1np[15..8]; m5.multi\_b[7..0] = H\_8;

m6.multi\_a[7..0] = i1np[23..16]; m6.multi\_b[7..0] = H\_6;

m7.multi\_a[7..0] = i1np[31..24]; m7.multi\_b[7..0] = H\_7;

m8.multi\_a[7..0] = i1np[39..32]; m8.multi\_b[7..0] = H\_4;

m9.multi\_a[7..0] = i1np[63..56]; m9.multi\_b[7..0] = H\_5;

reg[47..40] = i1np[7..0] $ m5.multi\_c[7..0] $ m6.multi\_c[7..0] $ m7.multi\_c[7..0] $ m8.multi\_c[7..0] $ i1np[47..40] $ i1np[55..48] $ m9.multi\_c[7..0];

m10.multi\_a[7..0] = i1np[7..0]; m10.multi\_b[7..0] = H\_5;

m11.multi\_a[7..0] = i1np[23..16]; m11.multi\_b[7..0] = H\_8;

m12.multi\_a[7..0] = i1np[31..24]; m12.multi\_b[7..0] = H\_6;

m13.multi\_a[7..0] = i1np[39..32]; m13.multi\_b[7..0] = H\_7;

m14.multi\_a[7..0] = i1np[47..40]; m14.multi\_b[7..0] = H\_4;

reg[55..48] = m10.multi\_c[7..0] $ i1np[15..8] $ m11.multi\_c[7..0] $ m12.multi\_c[7..0] $ m13.multi\_c[7..0] $ m14.multi\_c[7..0] $ i1np[55..48] $ i1np[63..56];

m15.multi\_a[7..0] = i1np[15..8]; m15.multi\_b[7..0] = H\_5;

m16.multi\_a[7..0] = i1np[31..24]; m16.multi\_b[7..0] = H\_8;

m17.multi\_a[7..0] = i1np[35..32]; m17.multi\_b[7..0] = H\_6;

m18.multi\_a[7..0] = i1np[47..40]; m18.multi\_b[7..0] = H\_7;

m19.multi\_a[7..0] = i1np[55..48]; m19.multi\_b[7..0] = H\_4;

reg[63..56] = i1np[7..0] $ m15.multi\_c[7..0] $ i1np[23..16] $ m16.multi\_c[7..0] $ m17.multi\_c[7..0] $ m18.multi\_c[7..0] $ m19.multi\_c[7..0] $ i1np[63..56];

reg[63..32].clk = clk; elsif t.OUTTIME[9..0] == 3 then

m0.multi\_a[7..0] = i2np[23..16]; m0.multi\_b[7..0] = H\_5;

m1.multi\_a[7..0] = i2np[39..32]; m1.multi\_b[7..0] = H\_8;

m2.multi\_a[7..0] = i2np[47..40]; m2.multi\_b[7..0] = H\_5;

m3.multi\_a[7..0] = i2np[55..48]; m3.multi\_b[7..0] = H\_7;

m4.multi\_a[7..0] = i2np[63..56]; m4.multi\_b[7..0] = H\_4;

re1g[7..0] = i2np[7..0] $ i2np[15..8] $ m0.multi\_c[7..0] $ i2np[31..24] $ m1.multi\_c[7..0] $ m2.multi\_c[7..0] $ m3.multi\_c[7..0] $ m4.multi\_c[7..0];

m5.multi\_a[7..0] = i2np[7..0]; m5.multi\_b[7..0] = H\_4;

m6.multi\_a[7..0] = i2np[31..24]; m6.multi\_b[7..0] = H\_5;

m7.multi\_a[7..0] = i2np[47..40]; m7.multi\_b[7..0] = H\_8;

m8.multi\_a[7..0] = i2np[55..48]; m8.multi\_b[7..0] = H\_6;

m9.multi\_a[7..0] = i2np[63..56]; m9.multi\_b[7..0] = H\_7;

re1g[15..8] = m5.multi\_c[7..0] $ i2np[15..8] $ i2np[23..16] $ m6.multi\_c[7..0] $ i2np[39..32] $ m7.multi\_c[7..0] $ m8.multi\_c[7..0] $ m9.multi\_c[7..0];

m10.multi\_a[7..0] = i2np[7..0]; m10.multi\_b[7..0] = H\_7;

m11.multi\_a[7..0] = i2np[15..8]; m11.multi\_b[7..0] = H\_4;

m12.multi\_a[7..0] = i2np[39..32]; m12.multi\_b[7..0] = H\_5;

m13.multi\_a[7..0] = i2np[55..48]; m13.multi\_b[7..0] = H\_8;

m14.multi\_a[7..0] = i2np[63..56]; m14.multi\_b[7..0] = H\_6;

re1g[23..16] = m10.multi\_c[7..0] $ m11.multi\_c[7..0] $ i2np[23..16] $ i2np[31..24] $ m12.multi\_c[7..0] $ i2np[47..40] $ m13.multi\_c[7..0] $ m14.multi\_c[7..0];

m15.multi\_a[7..0] = i2np[7..0]; m15.multi\_b[7..0] = H\_6;

m16.multi\_a[7..0] = i2np[15..8]; m16.multi\_b[7..0] = H\_7;

m17.multi\_a[7..0] = i2np[23..16]; m17.multi\_b[7..0] = H\_4;

m18.multi\_a[7..0] = i2np[47..40]; m18.multi\_b[7..0] = H\_5;

m19.multi\_a[7..0] = i2np[63..56]; m19.multi\_b[7..0] = H\_8;

re1g[31..24] = m15.multi\_c[7..0] $ m16.multi\_c[7..0] $ m17.multi\_c[7..0] $ i2np[31..24] $ i2np[39..32] $ m18.multi\_c[7..0] $ i2np[55..48] $ m19.multi\_c[7..0];

re1g[31..0].clk = clk;

elsif t.OUTTIME[9..0] == 4 then

m0.multi\_a[7..0] = i2np[7..0]; m0.multi\_b[7..0] = H\_8;

m1.multi\_a[7..0] = i2np[15..8]; m1.multi\_b[7..0] = H\_6;

m2.multi\_a[7..0] = i2np[23..16]; m2.multi\_b[7..0] = H\_7;

m3.multi\_a[7..0] = i2np[31..24]; m3.multi\_b[7..0] = H\_4;

m4.multi\_a[7..0] = i2np[47..40]; m4.multi\_b[7..0] = H\_5;

re1g[39..32] = m0.multi\_c[7..0] $ m1.multi\_c[7..0] $ m2.multi\_c[7..0] $

m3.multi\_c[7..0] $ i2np[39..32] $ i2np[47..40] $ m4.multi\_c[7..0] $ i2np[63..56];

m5.multi\_a[7..0] = i2np[15..8]; m5.multi\_b[7..0] = H\_8;

m6.multi\_a[7..0] = i2np[23..16]; m6.multi\_b[7..0] = H\_6;

m7.multi\_a[7..0] = i2np[31..24]; m7.multi\_b[7..0] = H\_7;

m8.multi\_a[7..0] = i2np[39..32]; m8.multi\_b[7..0] = H\_4;

m9.multi\_a[7..0] = i2np[63..56]; m9.multi\_b[7..0] = H\_5;

re1g[47..40] = i2np[7..0] $ m5.multi\_c[7..0] $ m6.multi\_c[7..0] $ m7.multi\_c[7..0] $ m8.multi\_c[7..0] $ i2np[47..40] $ i2np[55..48] $ m9.multi\_c[7..0];

m10.multi\_a[7..0] = i2np[7..0]; m10.multi\_b[7..0] = H\_5;

m11.multi\_a[7..0] = i2np[23..16]; m11.multi\_b[7..0] = H\_8;

m12.multi\_a[7..0] = i2np[31..24]; m12.multi\_b[7..0] = H\_6;

m13.multi\_a[7..0] = i2np[39..32]; m13.multi\_b[7..0] = H\_7;

m14.multi\_a[7..0] = i2np[47..40]; m14.multi\_b[7..0] = H\_4;

re1g[55..48] = m10.multi\_c[7..0] $ i2np[15..8] $ m11.multi\_c[7..0] $ m12.multi\_c[7..0] $ m13.multi\_c[7..0] $ m14.multi\_c[7..0] $ i2np[55..48] $ i2np[63..56];

m15.multi\_a[7..0] = i2np[15..8]; m15.multi\_b[7..0] = H\_5;

m16.multi\_a[7..0] = i2np[31..24]; m16.multi\_b[7..0] = H\_8;

m17.multi\_a[7..0] = i2np[35..32]; m17.multi\_b[7..0] = H\_6;

m18.multi\_a[7..0] = i2np[47..40]; m18.multi\_b[7..0] = H\_7;

m19.multi\_a[7..0] = i2np[55..48]; m19.multi\_b[7..0] = H\_4;

re1g[63..56] = i2np[7..0] $ m15.multi\_c[7..0] $ i2np[23..16] $

m16.multi\_c[7..0] $ m17.multi\_c[7..0] $ m18.multi\_c[7..0] $ m19.multi\_c[7..0] $ i2np[63..56];

re1g[63..32].clk = clk;

else

P0\_out[63..0] = reg[]; P0\_out[127..64] = re1g[];

end if;

END;

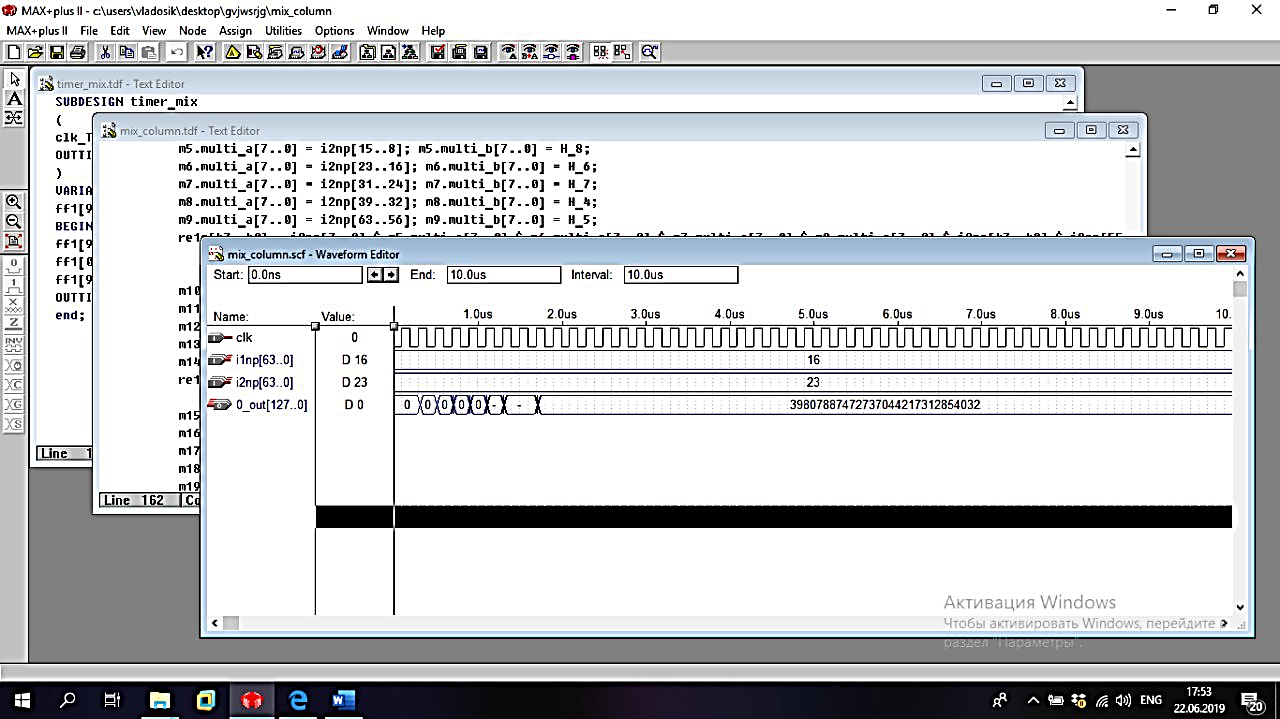


Рисунок 4.7 – Епюри виконання Mix Columns

4.6 Висновки до розділу

В даному розділі показана можливість програмно-апаратної реалізації вітчизняного алгоритму спроектованого в середовищі MAX + plus II при використанні мови AHDL, показано роботу редакторів середовища

MAX + plus II. Реалізовано операції SubBytes, ShiftRows, MixColumns, XoRRoundKey, AddRoundKey. Наочно показано програмно-апаратну реалізацію ДСТУ 7624:2014 “Калина” а також можливості вітчизняного стандарту блокового шифрування. Вище приведений матеріал показує, що ДСТУ 7624:2014 можливо реалізувати на програмованих структурах і використовувати для захисту інформації, що є власністю держави.

# ВИСНОВКИ

1. В дипломній роботі розглянуто основні поняття на які опирається і на чому базується захист інформації; звернено увагу на значимість ІБ, визначено понятійний апарат у сфері кібербезпеки; розмежовано поняття симетричної та асиметричної криптографії.
2. Описано класифікацію ПЛІС та особливості САПР, їх види та області застосування. Вказані загальні та основні відомості по роботі з ПЛМ, САПР MAX+PLUS II, в якому реалізовано алгоритм шифрування ДСТУ 7624:2014 для подальшого їх програмування на ПЛМ. Розглянуто принципи функціонування редакторів системи та розробки проектів цифрових пристроїв на мові AHDL.
3. Також розглянуто БСШ “Калина”, побудований на основі Rijndael-подібної структури, його вимоги до розробки та особливості алгоритму. Станом на 2015 рік це єдиний в світі національний стандарт, який визначає блоковий алгоритм шифрування із симетричним ключем довжиною 512 бітів. Алгоритм розроблено з урахуванням існуючих і потенційних загроз, подальшого інтенсивного розвитку ІС та необхідності активного використання протягом декількох наступних десятиліть.
4. В останньому розділі наведено реалізацію даного вітчизняного алгоритму шифрування завдяки наявних ресурсів і накопиченого світового досвіду.

Мету роботи досягнуто, основні завдання виконано: досліджено роботу САПР та ПЛМ; практично реалізовано основні блоки перетворення за допомогою САПР MAX+plus II. З огляду на відносну новизну шифру “Калина” невідомі публікації, які б стосувалися оцінки програмно-апаратної реалізації цього шифру для вбудованих систем, а отже, це питання потребує подальшого дослідження. Матеріал даної роботи можна застосовувати в подальшому для розробки засобів КЗІ та використання для в апаратурі шифрування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Про інформацію: Закон України Редакція від 21.12.2019 № 48. База даних Законодавство України / ВР України. URL: <https://zakon.rada.gov.ua/laws/show/2657-12> (дата звернення: 10.1.2020).
2. В.М. Петрик, М.М. Присяжнюк, Д.С. Мельник та ін. Інформаційна безпека держави: підручник. К: ІСЗЗІ КПІ ім. Ігоря Сікорського, 2016. Т 1. 264 c.
3. Богуш В. М., Кузин А. М. Інформаційна безпека від А до Я: 3000 термінів і понять: навчальний посібник. К : МОУ, 1999. 456 с.
4. Положення про технічний захист інформації в Україні: затв. Указом Президента України від 27.09.1999 р. № 1229. База даних Законодавство України / ВР України. URL: <https://zakon.rada.gov.ua/laws/show/1229/99> (дата звернення: 25.11.19).
5. Іванченко С.О. Інформація як об’єкт захисту та технічний захист інформації: методична розробка. К: ІСЗЗІ КПІ ім. Ігоря Сікорського, 2011. 5 с.
6. Іванченко С.О. Технічні канали витоку інформації: методична розробка. К: ІСЗЗІ КПІ ім. Ігоря Сікорського, 2011. 9 с.
7. Про захист інформації в автоматизованих системах: Закон України від 31.05.2005 № 26. База даних Законодавство України / ВР України. URL: <https://zakon.rada.gov.ua/laws/show/2594-15> (дата звернення: 12.12.2019).
8. Бабаш А. В., Шанкин Г. П. Криптография: науковий посібник. М: СОЛОН-ПРЕСС, 2002. 512 с.
9. Соловьев В.В., Васильев А.Г. Программируемые логические интегральные схемы и их применение. Мн: Беларуская наука, 1998. 270 с.
10. Норенков И.П., Маничев В.Б. Основы теории и проектирования САПР: Учеб. для вузов по спец. “Вычислительные машины, комплексы, системы и сети” Мн: Высшая школа, 1990. 335 с.
11. Скрипник Л.В., Корнейко О.В., Кулініч О.М. та ін. Основи проектування засобів захисту інформації на основі інтегральних схем зі структурою, що програмується: навчальний посібник. К: ІСЗЗІ КПІ ім. Ігоря Сікорського. 2005, 252 с.
12. Грушвицкий Р.И., Шашкин П.М. Проектирование в условиях временных ограничений: верификация проектов. СПб: БХВ-Петербург, 2008, 147 с.
13. Грушвицкий Р.И., Шашкин П.М. Проектирование в условиях временных ограничений: компиляция проектов. СПб.: БХВ-Петербург, 2011, 86 с.
14. Антонов А.П. Язык описания цифровых устройств AlteraHDL: практический курс. М: ИП РадиоСофт, 2001. 224 с.
15. Зенин О.С. Стандарт криптографической защиты AES. Конечные поля: учебное пособие. М: КУДИЦ, 2002. 176 с.
16. Бабенко Л.К. Современные алгоритмы блочного шифрования и методы их анализа: учебное пособие для студентов вузов, обучающихся по группе специальностей в обл. информ. безопасности. М: Гелиос АРВ, 2006. 376 с.
17. ДСТУ 7624:2014. Інформаційні технології. Криптографічний захист інформації. Алгоритм симетричного блокового перетворення. К: Мінекономрозвитку України, 2015. 35 с.
18. Горбенко І.Д. Симетричний блоковий шифр “Калина” – новий національний стандарт України. К: Інформаційні системи, 2015. 5-22 с.
19. Олійников Р.В, Горбенко І.Д, Казимиров О.О. та ін. Принципи побудови і основні властивості нового національного стандарту блокового шифрування України. Харків: нац. ун-т ім. В.Н. Каразіна, Приват. т-во “Ін‑т інформац. технологій”. 2015. 145 с.
20. Олійников Р.В, Горбенко І.Д, Казимиров О.О. та ін. МЕТОДЫ СИНТЕЗА И АНАЛИЗА СИММЕТРИЧНЫХ КРИПТОГРАФИЧЕСКИХ ПРЕОБРАЗОВАНИЙ ISSN 0485-8972 Радиотехника. 2015. Вып. 181 (або URl: <https://nure.ua/wp-content/uploads/2015/Scientific_editions/181/3.pdf> )
21. Олійников Р.В, Горбенко І.Д, Казимиров О.О. та ін. ПРИНЦИПИ ПОБУДОВИ І ОСНОВНІ ВЛАСТИВОСТІ НОВОГО НАЦІОНАЛЬНОГО СТАНДАРТУ БЛОКОВОГО ШИФРУВАННЯ УКРАЇНИ. ЗАХИСТ ІНФОРМАЦІЇ, ТОМ 17, №2, КВІТЕНЬ-ЧЕРВЕНЬ 2015 142-153 с.
22. Про державну таємницю: Закон України Редакція від 05.08.2018 № 16. База даних Законодавство України / ВР України. URL: <https://zakon.rada.gov.ua/laws/show/3855-12> (дата звернення: 10.1.2019).