Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра захисту інформації

КУРСОВИЙ ПРОЕКТ
з дисципліни "Спеціалізовані мікропроцесорні системи захисту інформації"
на тему: "Засіб захищеного зберігання паролів"
08-20.СМСЗІ.020.05.305 ПЗ

Студента 4 курсу групи 1БС-18мс
спеціальності 125 – Кібербезпек
ОПП Безпека інформаційних і комунікаційних систем
Жакун Геннадій Андрійович
(прізвище та ініціали)

Керівник: к. т. н., доц. каф. ЗІ
Баришев Ю. В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Національна шкала _____
Кількість балів: __ Оцінка: ECTS __

Члени комісії:

_____ _____
(підпис) (прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

_____ _____
(підпис) (прізвище та ініціали)

м. Вінниця – 2020 року

Підпис та дата
Інв. №
Нп. зам.
Підпис та дата
Інв. №

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
Зав. кафедри ЗІ, д. т. н., проф.
___В. А. Лужецький

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсовий проект з дисципліни "Спеціалізовані мікропроцесорні системи захисту інформації"
студенту групи БС-18 мс факультету ІТКІ Жакуну Геннадію Адрійовичу
Тема: Засіб захищеного зберігання паролів

1. Проаналізувати аналогічні відомі мікропроцесорні засоби та алгоритми їх роботи.
2. Розробити структуру мікропроцесорної системи та деталізувати її елементи. Результати роботи оформити у вигляді схеми електричної структурної.
3. Розробити алгоритм роботи мікропроцесорної системи захисту інформації та її структурних елементів. Результати роботи оформити у вигляді схеми роботи системи.
4. Виконати перевірку коректності роботи системи за допомогою середовища моделювання.

Вихідні дані:
мікроконтролер – ПЛІС;
розрядність ключа – 128;
клавіатура – не передбачено;
кількість цифрових давачів – не передбачено;
кількість аналогових давачів – не передбачено.
інтерфейс – SPI.

Дата видачі 02 вересня 2019 р.

Керівник _____ Ю. В. Баришев

Завдання отримав _____ Г. А. Жакун

АНОТАЦІЯ

Жакун Г. А. Засіб захищеного зберігання паролів. Курсовий проект. – Вінниця: ВНТУ, 2020. – 26 с.

Українською мовою. Рисунків – 21.

В курсовому проекті розроблено структуру засобу захищеного зберігання паролів. Здійснено аналіз існуючих засобів захищеного зберігання паролів, та досліджено процес обміну даними в інтерфейсі SPI, складено функціональну та структуру схему роботи мікропроцесорної системи. Розроблено програму для реалізації роботи пристрою для засобу захищеного зберігання паролів.

Програмне рішення реалізовано засобами ModelSim-Altera з використанням мови програмування VHDL.

ABSTRACT

Zhakun G. A. A tool for secure password storage. Course project. - Vinnytsia: VNTU, 2020. - 26 p.

In ukrainian. Figures - x, tables - x.

In the course project the structure of the secure password storage facility was developed. The analysis of the existing means of secure storage of passwords is carried out, the process of data exchange in the SPI interface is investigated, the functional and structure diagram of the microprocessor system operation is made. A program has been developed to implement the device for secure password storage.

The software solution is implemented by ModelSim-Altera using VHDL programming language.

ЗМІСТ

ВСТУП

Безпечне зберігання паролів потрібно для захисту паролів від їх викрадення.

Дедалі більше набувають популярності програмні реалізації зберігання паролів, однак вони не забезпечують захист та безпеку, та мають стійкість у порівнянні з апаратним рішенням, тому більш доцільніше використовувати саме спеціалізований засіб зберігання паролів.

Спеціалізований засіб зберігання паролів має свої переваги, але також є недоліки.

Переваги:

– засіб не потребує багато ресурсів процесора так як це окремий прилад а також немає необхідності підключення до інтернету;

– мало способів зламати пристрій або втрутитись в його роботу.

Недоліки:

– неможливо здійснити оновлення роботи пристрою кінцевому користувачу;

– матеріальні витрати на пристрій.

Об’єктом курсового проектування є процес зберігання паролів . Предметом – засіб зберігання пароля.

Метою даного курсового проекту є збільшення захисту в інформаційних систем під час зберігання паролів.

Для досягнення мети необхідно розв’язати такі задачі:

– аналіз відомих методів засобів зберігання паролів;

– огляд відомих засобів та пристрої;

– розробка структури засобу зберігання паролів;

– розробка алгоритмів роботи блоків даної структури;

– реалізація та тестування коректності роботи мікропроцесорної системи.

1 АНАЛІЗ ЗАСОБУ ЗАХИЩЕНОГО ЗБЕРІГАННЯ ПАРОЛІВ

1.1 Огляд існуючих засобів захисту паролів

Дивлячись на існуючі засоби зберігання пароля треба відзначити KeePass.

KeePass - це безкоштовний менеджер паролів з відкритим вихідним кодом, який безпечно керувати своїми паролями. З можливістю помістити всі свої паролі в одну базу даних, яка заблокована одним головним ключем або файлом ключа. Бази даних зашифровані з використанням кращих і найбільш безпечних алгоритмів шифрування, відомих в даний час (AES і Twofish) [1].

KeePass підтримує розширений стандарт шифрування (AES) і алгоритм Twofish для шифрування своїх баз паролів що забезпечує надійність.

Вся база даних зашифрована, а не тільки поля пароля. Таким чином, всі дані користувача в захищеному виді.

SHA-256 використовується для хешування компонентів головного ключа. SHA-256 - це 256-бітна криптографічна безпечна одностороння хеш-функція.

Захист пам'яті процесу: паролі зашифровані під час роботи KeePass, тому навіть коли операційна система вивантажує процес KeePass на диск, паролі не розкриваються [1].

Захищені потоки в пам'яті: при завантаженні внутрішнього формату XML паролі шифруються з використанням ключа сеансу. Так як використовується для кожного сеансу свій ключ то це забезпечує безпеку.

Елементи управління для редагування паролів з підвищеною безпекою: Більшість з доступних шпигунів для редагування пароля не працює проти елементів управління. Паролі, введені в ці елементи управління, навіть не видно в пам'яті процесу KeePass. Авто-тип також може бути захищений від клавіатурних шпигунів [1].

Під час роботи KeePass конфіденційні дані зберігаються в зашифрованому вигляді в пам'яті процесу. Це означає, що навіть якщо ви скинете пам'ять процесу KeePass на диск, ви не зможете знайти конфіденційні дані. З міркувань продуктивності захист пам'яті процесу застосовується тільки до конфіденційних даних; конфіденційні дані тут включають, наприклад, головний ключ і паролі входу, але не імена користувачів, замітки і вкладення файлів. Зверніть увагу, що це не має нічого спільного з шифруванням файлів бази даних; в файлах бази даних всі дані (включаючи імена користувачів і т. д.) зашифровані.

Отже, враховуючи, що дані засобів захисту паролів не містять потрібних інтерфейсів, а також необхідного блоку шифрування даних, тому реалізовується саме власна мікропроцесорна система, що містиме всі необхідні модулі.

1.2 Алгоритм шифрування паролів RC4

RC4- потоковий шифр, який широко застосовується в різних системах захисту інформації в комп'ютерних мережах.

Алгоритм RC4, як і будь-який потоковий шифр, будується на основі генератора псевдовипадкових бітів. На вхід генератора записується ключ, а на виході зчитаються псевдовипадкові біти. Довжина ключа може становити від 40 до 2048 біт. Генеруються біти мають рівномірний розподіл [2].

Основні переваги шифру:

- висока швидкість роботи;
- змінний розмір ключа.

RC4 досить уразливий, якщо:

- використовуються не випадкові або пов'язані ключі;
- один ключовий потік використовується двічі.

RC4 - фактично клас алгоритмів, що визначаються розміром блоку (надалі S-блоку). Параметр n є розміром слова для алгоритму і визначає довжину S-блоку. Зазвичай, n = 8, але в цілях аналізу можна зменшити його. Однак для підвищення безпеки необхідно збільшити цю величину. В алгоритмі немає протиріч на збільшення розміру S-блоку. При збільшенні n, припустимо, до 16 біт, елементів в S-блоці стає 65 536 і відповідно час початкової ітерації буде збільшено. Однак, швидкість шифрування зросте [2].

Внутрішній стан RC4 представляється у вигляді масиву розміром 2n і двох лічильників. Масив відомий як S-блок, і далі буде позначатися як S. Він завжди містить перестановку 2n можливих значень слова.

Для проведення досліджень протестована робота генератора псевдовипадкових чисел RC4 для поля GF (24) На повному безлічі ненульових ключових даних. Обчислювалися значення i, j, при яких довжина періоду була мінімальною.

В ході проведених досліджень було встановлено, що при будь-якому значенні S-блоку існує значення i, j, при яких довжина періоду становить 240 елементів псевдослучайной послідовності (для поля GF (24). Для проведення досліджень протестована робота генератора псевдовипадкових чисел RC4 для поля GF (24) На повному безлічі ненульових ключових даних. Обчислювалися значення i, j, при яких довжина періоду була мінімальною.

В ході проведених досліджень було встановлено, що при будь-якому значенні S-блоку існує значення i, j, при яких довжина періоду становить 240 елементів псевдослучайной послідовності (для поля GF (24).

1.3 Інтерфейс SPI

Схема зв'язку SPI - це двобічний канал передачі даних, що використовує чотири дроти. Пристрій ініціює транзакцію, потягнувши провід Slave Select (SS) на низький рівень. Лінія послідовних тактових імпульсів (SCLK), керована пристроєм, забезпечує синхронний джерело тактових імпульсів. Майстер передає дані через лінію Master Out, Slave In (MOSI) і отримує дані через лінію Master In, Slave Out (MISO) [3].

Інтерфейс може спілкуватися з декількома пристроями за допомогою різних методів. У найбільш поширеною конфігурації кожен пристрій має незалежну лінію SS, але спільно використовує лінії SCLK, MISO і MOSI з іншими пристроями. Кожен ведений ігнорує загальні лінії, коли його лінія SS не перебуває на низькому рівні.

SPI має чотири режими роботи, засновані на двох параметрах: полярність годин (CPOL) і фаза годин (CPHA). Ведучий і ведений повинні використовувати один і той же режим для обміну даними. Якщо CPOL дорівнює нулю, то SCLK зазвичай низький, і перший фронт тактового сигналу є наростаючим фронтом. Якщо CPOL дорівнює одиниці, SCLK зазвичай високий, і перший фронт тактового сигналу є падаючим фронтом. CPHA визначає вирівнювання даних. Якщо CPHA дорівнює нулю, то перший біт даних записується на задньому фронті SS і зчитується на першому фронті SCLK. Якщо CPHA дорівнює одиниці, дані записуються на першому ребрі SCLK і зчитаються на другому SCLK.

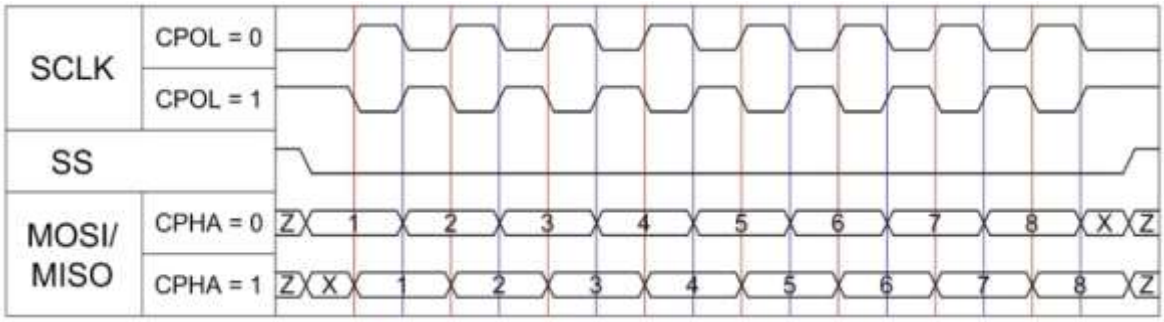


Рисунок 1.1 – Режими роботи SPI

Низький логічний рівень на зайнятому вихідному порту вказує, що компонент готовий прийняти команду. Компонент фіксує настройки, адресу і дані для транзакції за першим наростаючому фронті тактового сигналу, де підтверджується вхід дозволу. На наступних компонентах підтверджує сигнал зайнятості і починає виконувати транзакцію. Після завершення компонент виводить отримані дані через порт rx_data. Ці дані залишаються в порту до тих пір, поки компонент не отримає нові дані з подальшою транзакції. Компонент

встановлює низький рівень зайнятості, щоб повідомити користувача, коли дані доступні, і компонент негайно готовий до іншої інструкції [3].

Інтерфейс SPI є гнучким програмованим логічним компонентом, який забезпечує зв'язок з різними відомими пристроями через єдиний паралельний інтерфейс. Це дозволяє здійснювати зв'язок із зазначеним користувачем числом ведених пристроїв, що може зажадати незалежних режимів SPI, ширини даних і тактової частоти послідовного порту.

Всього, для повнодуплексного обміну, в інтерфейсі SPI використовуються 4 лінії: SCLK, MOSI, MISO і SS. (рис 1.2)

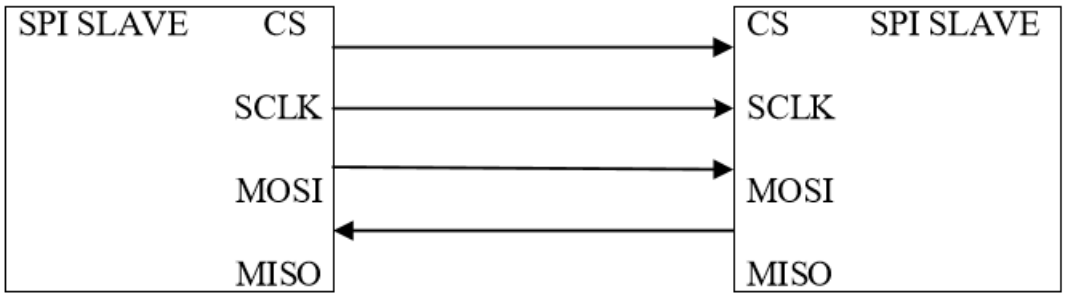


Рисунок 1.2 – Представлення повнодуплексного обміну

SCLK – шина тактування, на цій лінії майстер генерує синхроімпульси;

MOSI (Master Out, Slave In) – вихід ведучого, вхід веденого (по цій лінії майстер передає дані слейву)

MISO (Master In, Slave Out) – вхід ведучого, вихід веденого, по цій лінії майстер приймає дані від слейва;

SS (Slave Select) – вибір веденого, за допомогою цієї лінії майстер керує сеансами обміну. «1» і «0» кодуються рівнем напруги на шинах даних (MOSI, MISO) у звичайній позитивній логіці, тобто високий рівень напруги на шині відповідає «одиниці», а низький рівень відповідає «нулю».

Сигнал SS відзначає початок і кінець сеансу обміну. Цей сигнал зазвичай інверсний, тобто під час сеансу обміну даними майстер повинен встановлювати на лінії SS низький рівень, а при відсутності обміну – високий. Наявність сигналу SS дозволяє майстру організувати підключення до декількох веденних, використовуючи один і той же синхросигнал і одні і ті ж шини даних, без яких-небудь додаткових протоколів.

Тобто, ні за назвою ліній, ні за рівнями напруги на них, ні навіть по їх кількості, однозначно ідентифікувати SPI не можна, зате це відмінно можна зробити по самому методу передачі даних, по тому як відбувається їх установка на шину і зчитування.

2 СТРУКТУРА МІКРОПРОЦЕСОРНОЇ СИСТЕМИ ЗА ЗАСОБОМ ЗАХИЩЕНОГО ЗБЕРІГАННЯ ПАРОЛІВ

2.1 Узагальнена структура пристрою

Для моделювання поставленої задачі використано програмне середовище ModelSim та мову VHDL [4].

Загальна структурна схема роботи пристрою представлена на рисунку 2.1. До її складу входить інтерфейс SPI, що відповідають за отримання та пересилання даних користувачеві. Блок підтримання протоколу який відповідає за розподілення команд.

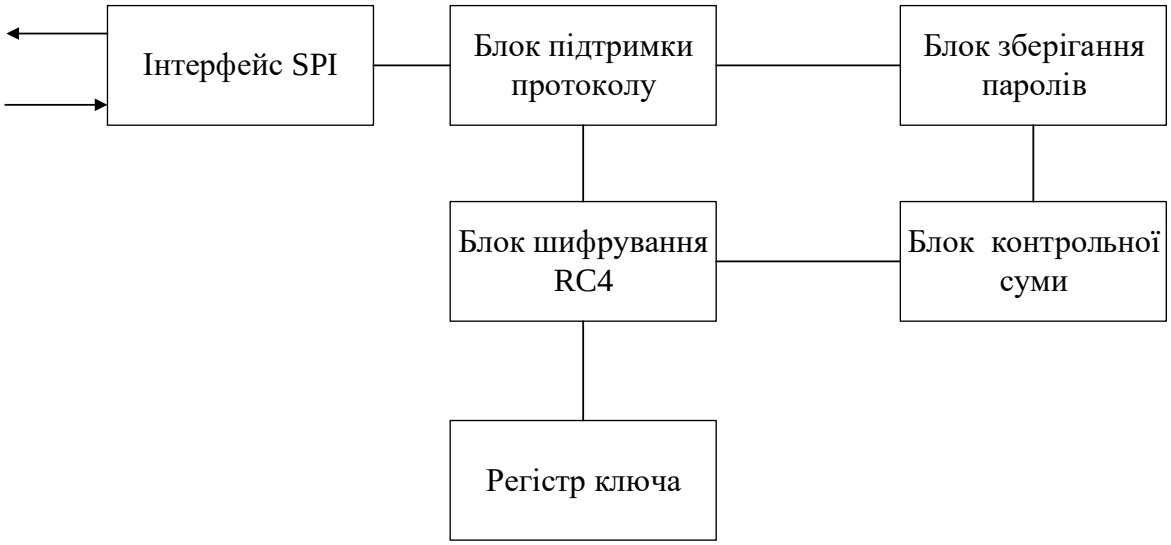


Рисунок 2.1 – Загальна структурна схема роботи

На інтерфейс SPI надходить запит на пароль тоді блок підтримки звертається до блоку зберігання паролю який видає зашифрований пароль з контрольною сумою блоку контрольної суми який її перевіряє після чого звертається до блоку шифрування який в свою чергу бере регістр ключа та розшифровує пароль та видає його блоку підтримки протоколу. Після чого блок підтримки протоколу відає пароль інтерфейсу SPI який передає пароль користувачеві.

Блок підтримання протоколу відповідає за обмін інформацій між блоками.

2.2 Структура блоку контрольної суми

Контрольна сума — деяке значення, розраховане на основі набору даних з використанням певного алгоритму, що використовується для перевірки цілісності даних при їх передачі або збереженні. Також контрольні суми можуть використовуватись для швидкого порівняння двох наборів даних на

нееквівалентність: з великою вірогідністю різні набори даних матимуть відмінні контрольні суми.

Використовується для обчислення контрольного коду — великої кількості біт всередині великого блока даних, наприклад, мережевого пакету або блоку комп'ютерного файлу, що використовується для виявлення помилок під час передачі або збереження інформації. Значення контрольної суми зазвичай додається до початку або кінця блоку даних безпосередньо перед початком передачі або запису даних на носій інформації. У майбутньому це значення перевіряється для підтвердження цілісності даних.

Блок контрольної суми відповідає за цілісність пароля на схемі нижче показана його робота

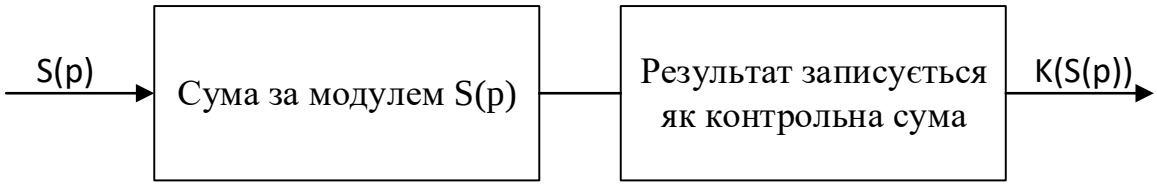


Рисунок 2.2 – Загальна структурна схема роботи

На блок контрольної суми надходить зашифрований пароль (S(p)). Пароль додається за модулем 256. Результат додається до пароля K(S(p)) як окрема переміна. Перевірка результат по сумі модулю два на пароль. Має вийти значення 0.

2.3 Структура блоку шифрування RC4

Rivest Cipher 4 – потоковий шифр, що широко застосовується в різних системах захисту інформації в комп’ютерних мережах (наприклад, в протоколах SSL і TLS, алгоритмі безпеки бездротових мереж WEP і WPA) [5].

Основні переваги шифру - висока швидкість роботи і змінний розмір ключа.RC4 досить уразливий, якщо використовуються не випадкові або пов’язані ключі, один ключовий потік використовується двічі. Ці фактори, а також спосіб використання можуть зробити криптосистему небезпечною (наприклад WEP).Головними факторами, що сприяли широкому застосуванню RC4, були простота його апаратної й програмної реалізації, а також висока швидкість роботи алгоритму в обох випадках [6].

Алгоритм генерації псевдовипадкових чисел k_i буде показаний нижче на рисунку 2.3 [7].

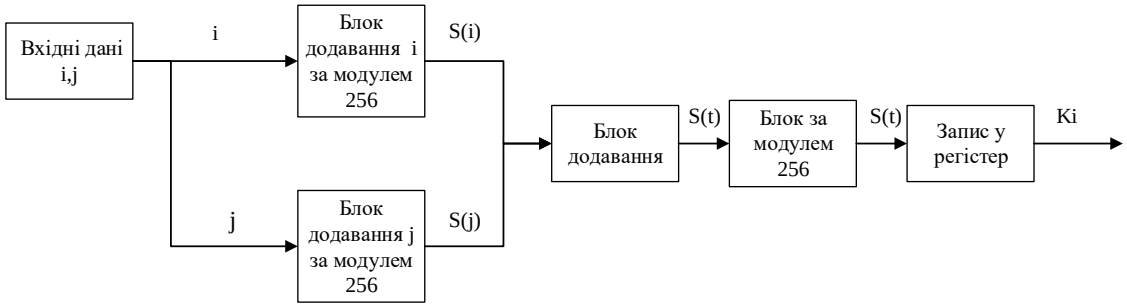


Рисунок 2.3 – Алгоритм генерацій послідовності бітів

В одному циклі RC4 визначається одне n-бітове слово K з ключового потоку [7].

Алгоритм шифрування.

- Функція генерує послідовність бітів k_i.
- Потім послідовність бітів за допомогою операції «підсумовування по модулю два» (xor) об'єднується з відкритим текстом m_i.
- В результаті виходить шифрограма c_i: $c_i = m_i \oplus k_i$

Алгоритм розшифрування.

- Повторно створюється (регенерується) потік бітів ключа (ключовий потік) k_i.
- Потік бітів ключа складається з шифрограмою c_i операцією «xor».
- В силу властивостей операції «xor» на виході виходить вихідний (незашифрований) текст m_i:

$m_i = \mathrm{c}_i \oplus k_i = (m_i \oplus k_i) \oplus k_i$

Блок шифрування даних складається із блоків, що позначають етапи роботи. (рис. 2.4).

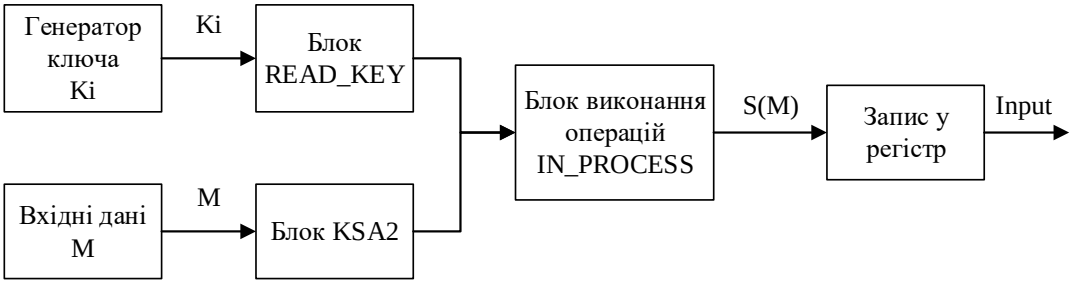


Рисунок 2.4 – Блок шифрування даних

На початку роботи програми на блок вхідних даних M надходить дані для зашифрування. З генератору ключа надходить ключ, який необхідно використовувати в поточній операції.

Після чого дані надходять до блоку KSA2 де дані присвоюється значенню input де присвоюються перемінній sblockInValue та перевіряється clock та після перевірки відправляє переміну sblockInValue де дані зчитуються та записуються у переміну sblockIndex. Після чого надсилаються на блок IN_PROCESS.

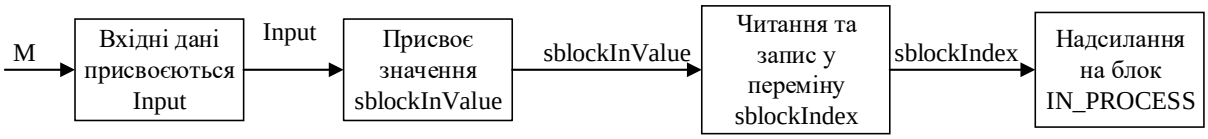


Рисунок 2.5 – Блок читання даних KSA2

Після чого ключ надходить до блоку READ_KEY де ключ присвоюється значенню input після чого присвоюються перемінній KeyInValue та перевіряється clock та після перевірки відправляє переміну KeyInValue де дані зчитуються та записуються у переміну KeyIndex. Після чого надсилаються на блок IN_PROCESS.

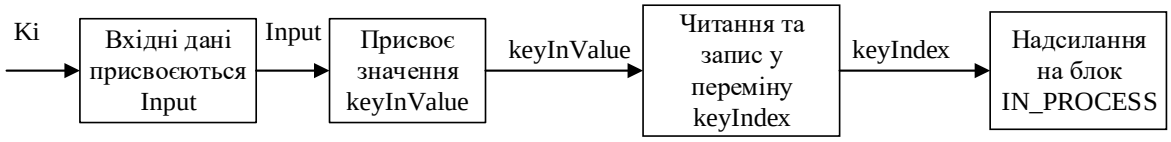


Рисунок 2.6 – Блок читання ключа READ_KEY

Коли переміні з даними надходять на блок IN_PROCESS. Далі кожна переміна окремо присвоюється. Переміна keyIndex присвоюється переміні S[i] аналогічно переміна sblockIndex присвоюється S[j] після чого виконується операція сума за модулем два і результат надсилається на регістр.

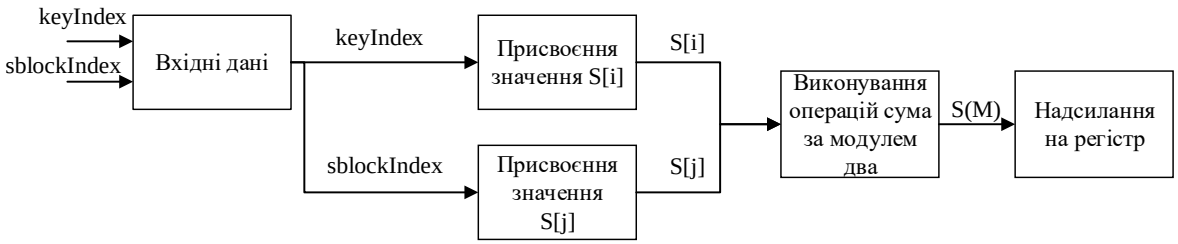


Рисунок 2.7 – Блок виконання операцій IN_PROCESS

Регістр записує результат у переміну Input і відправляє на вихід. При розшифруванні всі процедури зберігаються, проте проводяться в зворотному порядку.

2.4 Схема роботи блоку SPI

Для виконання завдання розроблено SPI інтерфейс, для входу Slave та виходу Master.

Схема інтерфейсу SPI Slave на рисунку 2.8.

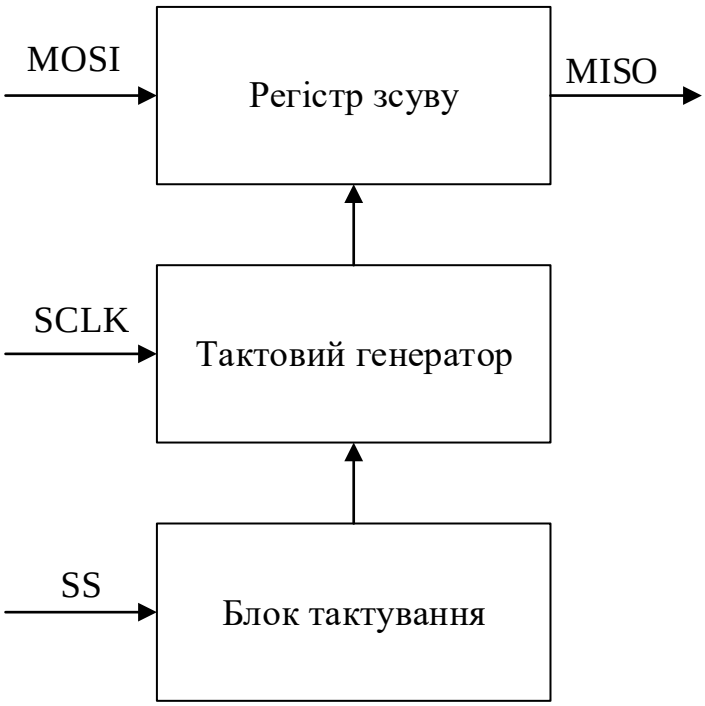


Рисунок 2.8 – Схема блоку інтерфейсу SPI Slave

Схема інтерфейсу SPI Master на рисунку 2.9.

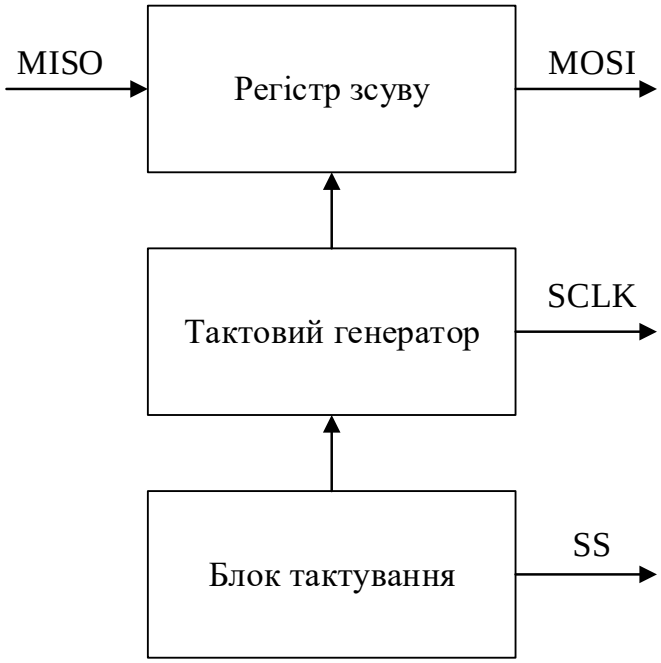


Рисунок 2.9 – Схема блоку інтерфейсу SPI Master

Для коректної роботи інтерфейсів необхідна наявність блоків сигналу керування та блоку команд. Для взаємодії інтерфейсів є лінії передачі бітів даних MOSI та MISO, а SCLK необхідний для передачі сигналу для синхронної роботи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Загальна схема роботи

Для програмної реалізації поставленої для курсового проекту задачі, розроблено загальний алгоритм функціонування, який представлено на рисунку 3.1.

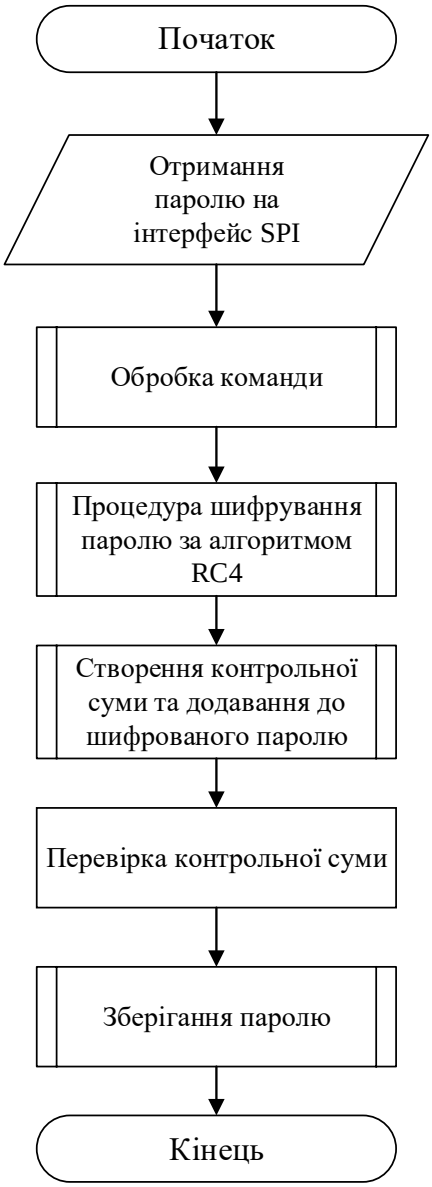


Рисунок 3.1 – Загальна робота схеми

Даний пристрій працює наступним чином: спочатку на інтерфейс находить пароль з командою, які потрібно виконати. Далі їде обробка команди де вирішується що буде виконувати пристрій. Після чого відбувається безпосередньо процедура зашифрування паролю за допомогою алгоритму RC4.

Після до зашифрованого паролю створюється контрольна сума та догадається до паролю та надсилається до блоку зберігання паролю. Після отримання зашифрованого пароля перевіряється контрольна сума. Якщо

результат контрольної суми відповідає необхідному значенню то пароль зберігається.

Для діставання збереженого паролю відбувається так само, але після потрапляння команди на інтерфейс вони переходять до процедури розшифрування за допомогою алгоритму RC4 і також виводиться за допомогою інтерфейсу SPI.

Отже, на виході інтерфейсу пристрою отрується збережений пароль або відкрити для читання.

3.2 Процедура шифрування за алгоритмом RC4

Реалізацію роботи шифрування паролю за алгоритмом RC4 розбито на реалізацію окремих блоків, які входять до складу алгоритму. На вхід блоку подаються дані з зовнішнього світу, які є відомі, а також генерується значення ключа. Основною задачею є отримання паролю та його зашифрування для подальшого використання. (рис 3.2).

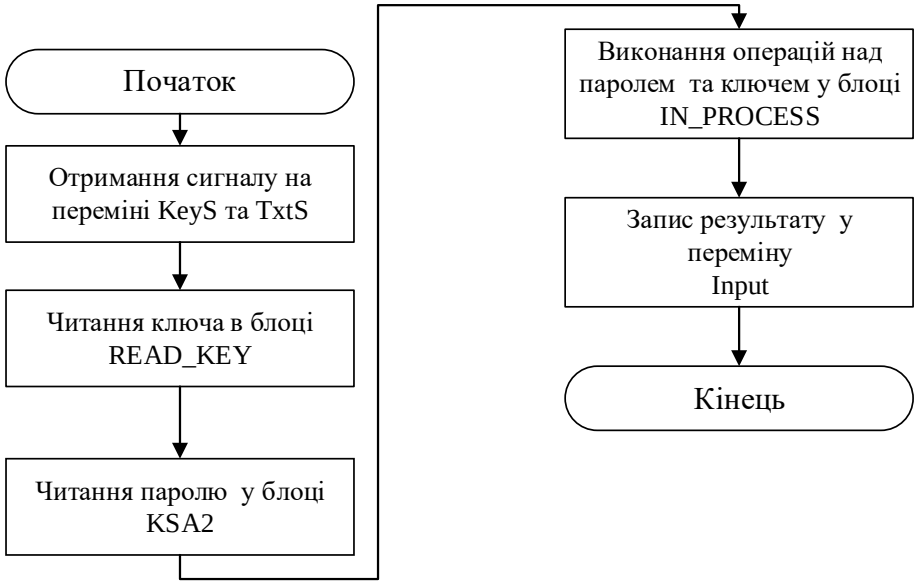


Рисунок 3.2 – Схема роботи блоку RC4

Отже, на алгоритму показано повну роботу блоку шифрування RC4. Який відповідає за зберігання паролю.

3.3 Схема роботи блоку контрольної суми

Реалізована контрольна сума по алгоритму операцій суми по модулю два. Спочатку пароль конвертується у двійковий формат. Після чого пароль сам на себе робить операцію сума по модулю два за модулем 256 результат записується

як контрольна сума. Якщо пароль був змінений або була втрата частини паролю при передачі то при перевірці результат буде 1.

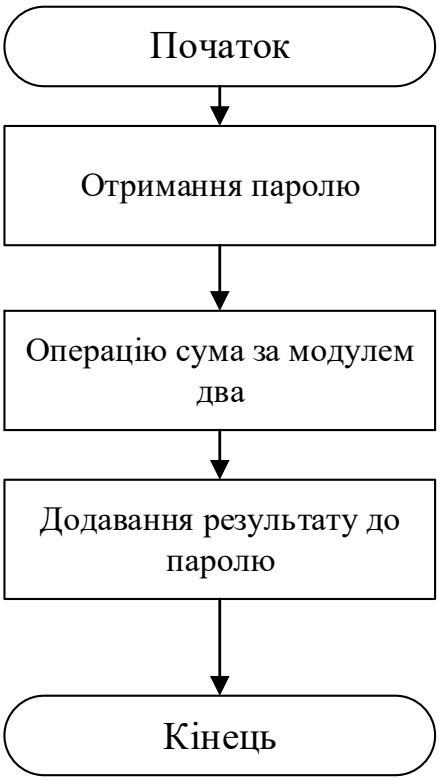


Рисунок 3.3 – Схема роботи блоку контрольної суми

Отже, на алгоритму показано повну роботу блоку контрольної суми. Який відповідає за цілісність паролю.

3.4 Тестування програмного засобу

Програмування мікропроцесорного пристрою було реалізоване за допомогою середовища ModelSim та мови програмування VHDL.

Спочатку було проведено компіляцію кожного блоку, що показано на рисунку 3.4.

Рисунок 3.4 – Тестування проекту у середовищі

На рисунку 3.5 зображено блок шифрування в середовищі програмування.

```
KeyS <= '1';
Input <= "01001011";--K
wait for clock_period*10;
Input <= "01100101";--e
wait for clock_period*10;
Input <= "01111001";--y
wait for clock_period*10;

KeyS<='0';

wait for clock_period*2500;

TxtS<='1';
Input <= "01010000";--P
wait for clock_period*22;
Input <= "01101100";--l
wait for clock_period*22;
Input <= "01100001";--a
wait for clock_period*22;
Input <= "01101001";--i
wait for clock_period*22;
Input <= "01101110";--n
wait for clock_period*22;
Input <= "01110100";--t
wait for clock_period*22;
Input <= "01100101";--e
wait for clock_period*22;
Input <= "01111000";--x
wait for clock_period*22;
Input <= "01110100";--t
wait for clock_period*22;
```

Рисунок 3.5 – Представлення блоку шифрування в середовищі програмування

Для тестування потрібно запустити симуляцію виконання та переконатись, що програмний засіб не містить помилки, та відповідає завданню.

Першим дією програми є створення ключа для шифрування даних для пароля.

Результат роботи першої дій програми представлено на рисунку 3.6.

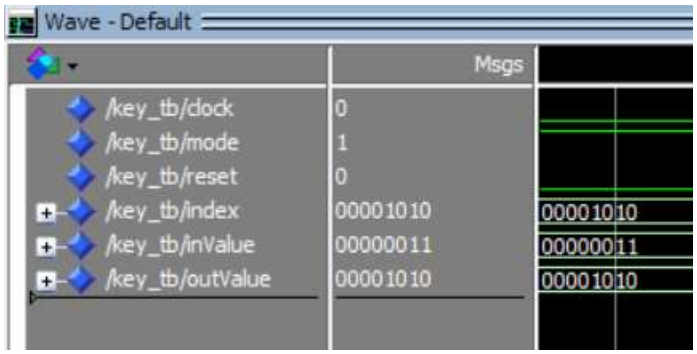


Рисунок 3.6 – Результат тестування генерування ключа

Далі потрібно здійснити перевірку, що відбувається коректне шифрування даних(рис 3.7).

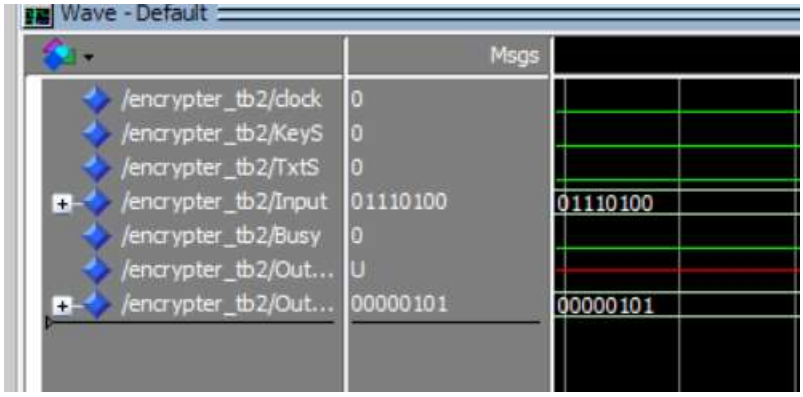


Рисунок 3.7 – Представлення зашифрованих даних

В результаті вийшов зашифрований пароль за алгоритмом та представлений у двійкові формі.

Далі було проведено тестування блоку контрольної суми.

При виборі даних для контрольної суми були вибрані різні комбінацій для перевірки алгоритму. Представлення успішного створення контрольної суми (рис 3.8).

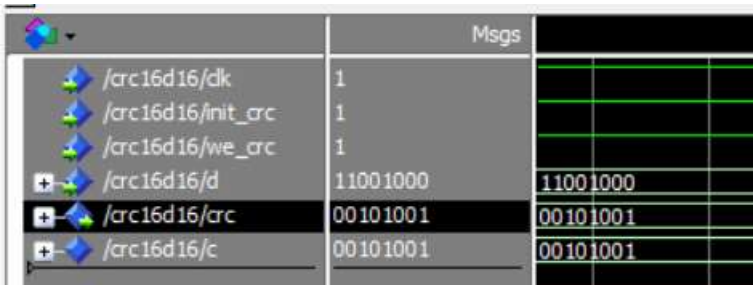


Рисунок 3.8 –Представлення контрольної суми.

Якщо під час передачі була зроблена зміна контрольної суми то результат представлений на рисунку 3.9.

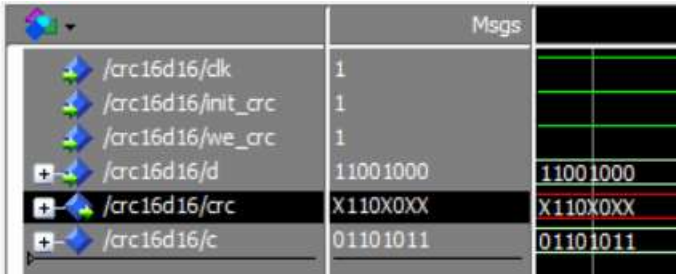


Рисунок 3.9 –Представлення зміни контрольної суми.

Наступною дією є здійснюється перевірка роботи інтерфейсу SPI для програми (рис 3.10).

Рисунок 3.10 – Представлення роботи інтерфейсу SPI

Отже після проведення тестування інтерфейс видав команду ready, що значить готовність виконання програми.

ВИСНОВОК

Під час виконання курсового проекту було розроблено засіб захищеного зберігання паролів.

У ході роботи над курсовим проектом було проведено аналіз існуючих засобів зберігання паролів. По результату аналізу обґрунтоване рішення, про розробку власного засобу зберігання, що має необхідні модулі. Виконано розробку структури мікропроцесорної системи та її складових компонентів.

Отже, пристрій складається з таких компонентів: блок SPI, блок підтримки протоколу, блок шифрування RC4, блок контрольної суми, блок зберігання паролю. Інтерфейси SPI Slave та Master розроблено з метою забезпечення зберігання паролю та його діставання з пристрою. Результати оформлено у вигляді схеми електричної структурної.

Для розв’язання задач курсового проекту досліджено засоби зберігання паролю, а також інтерфейс SPI. Мікропроцесорну систему, що реалізує даний алгоритм та передавання даних через інтерфейс SPI реалізовано за допомогою мови програмування VHDL в середовищі моделювання ModelSim.

На основі розробленої структури та її компонентів розроблено алгоритм роботи мікропроцесорної системи. Результати роботи оформлені у вигляді схеми роботи системи. Виконано перевірку коректності роботи системи за допомогою середовища моделювання, та проведено тестування програмного застосунку, що підтверджує коректність роботи алгоритму, а також роботи інтерфейсів.

Розроблена мікропроцесорна система може застосовуватися в різноманітних інформаційно-комунікаційних системах для забезпечення конфіденційності даних, що передаються каналом зв’язку.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Dominik Reichl. Setup – KeePass. Режим доступу: URL: https://keepass.info/news/n200120_2.44.html/
2. Klein A. Attacks on the RC4 stream cipher // Designs, codes and cryptography. — 2008 – 286 с.
3. Modern Embedded Computing Designing Connected, Pervasive, Media-Rich Systems Peter Barry Patrick Crowley 2012 Elsevier Inc – 545с.
4. VHDL: Programming by Example Douglas L. Perry Fourth Edition 2014 – 497с.
5. Standard cryptographic algorithm naming. Режим доступу: URL: http://www.users.zetnet.co.uk/hopwood/crypto/scan/cs.html#RC4-drop
6. Scott R. Fluhrer, Itsik Mantin, Adi Shamir. Weaknesses in the key scheduling algorithm of RC4 // Lecture notes in computer science. — 2001.
7. Souradyuti Paul, Bart Preneel. A New Weakness in the RC4 Keystream generator and an approach to improve the security of the cipher (англ.) // Lecture notes in computer science : journal. — 2004. — Vol. 3017. — P. 245—259.

ДОДАТКИ

Додаток А

Технічне завдання

Вінницький національний технічний університет
Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ
Керівник доцент кафедри ЗІ
_____ Баришев В. Ю.
_____ 2020 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на курсовий проект
з дисципліни "Спеціалізовані мікропроцесорні системи захисту інформації"
на тему: " Засіб захищеного зберігання паролів "
08-20.СМСЗІ.020.05.305 ТЗ

Вінниця 2020

1 Назва та галузь використання

Мікропроцесорна система захищеного зберігання паролів призначена для захисту паролів.

2 Основа для розробки

Робоча програма навчальної дисципліни «Спеціалізовані мікропроцесорні системи захисту інформації».

3 Мета та призначення розробки

Покращення надійності зберігання паролів комп'ютерної системи шляхом розробки спеціалізованої мікропроцесорної системи, що реалізує засіб захищеного зберігання паролів. Мікропроцесорна система призначена для шифрування паролів користувачів комп'ютерної системи. Мікропроцесорна система призначена для шифрування даних користувача комп'ютерної системи.

4 Джерела розробки

Modern Embedded Computing Designing Connected, Pervasive, Media-Rich Systems Peter Barry Patrick Crowley 2012 Elsevier Inc – 545c.

Klein A. Attacks on the RC4 stream cipher // Designs, codes and cryptography. — 2008 – 286 c.

5 Вимоги до системи

5.1 Параметри розроблюваної системи:

- мікроконтролер – ПЛІС;
- розрядність ключа – 128;
- клавіатура – не передбачено;
- кількість цифрових давачів – не передбачено;
- кількість аналогових давачів – не передбачено.
- інтерфейс – SPI.

5.2 Вимоги до апаратного і програмного забезпечення, на якому повинна працювати система:

- середовище моделювання – ModelSim;
- мова розробки – VHDL.

5.3 Вимоги щодо тестування. Тестування повинно проводитись методом комп'ютерного моделювання з використанням програмного пакету ModelSim.

6 Вимоги до супровідної документації

6.1 Графічна і текстова документація повинна відповідати діючим стандартам України.

6.2 Пристрій повинен супроводжуватись:

- текстом програми;

– результатами тестування роботи мікропроцесорної системи;

7 Стадії та етапи розробки

Робота з теми виконується у 7 етапів.

Етап	Зміст	Початок	Закінчення	Примітка
1	Аналіз засобів зберігання паролів	03.02.20	15.02.2020	Чорновий варіант розділу 1
2	Огляд засобів зберігання паролів	15.02.20	21.02.20	Чорновий варіант розділу 1
3	Розробка структури мікропроцесорної системи	22.02.20	29.02.20	Схема електрична структурна, чорновий варіант розділу 2
4	Розробка алгоритмів роботи мікропроцесорної системи	30.02.20	05.03.20	Схема роботи системи, чорновий варіант розділу 3
5	Розробка програми	08.03.20	12.03.20	Текст програми, чорновий варіант розділу 3
6	Тестування роботи мікропроцесорної системи	13.03.20	15.03.20	Результати тестування, чорновий варіант розділу 3
7	Оформлення пояснювальної записки.	16.03.20	18.03.20	Пояснювальна записка

8 Порядок контролю та прийому.

До приймання курсового проекту представляється :

– ПЗ до курсового проекту;
– робоча система для реалізації захисту;
– графічні частина.

Початок розробки 31.01.2020.

Крайній термін виконання курсового проекту 20.03.2020.

Розробив студент групи 1БС-18мс _____ Жакун Г. А.

Додаток Б

Текст програми

Блок RC4

```
        state_manager : process(clock, KeyS, TxtS, Input)
        begin
      if (rising_edge(clock)) then
        case c_state is

          when INIT =>
            if (KeyS = '1' and TxtS = '0') then
              next_state := READ_KEY;

            elsif (KeyS = '0' and TxtS = '1') then
              next_state := ARRRESET;

            else
              next_state := ARRRESET;
            end if;

          when READ_KEY =>
            if (KeyS = '1' and TxtS = '0') then
                if(clocks > io_period) then
                  clocks := 0;
                else
                  if (clocks = 0) then
                    keyMode <= '0';
                    keyIndex <= conv_std_logic_vector(iter, 8);
                    keyInValue <= Input;
                    iter := iter + 1;
                    Output <= conv_std_logic_vector(iter, 8);
                  end if;
                  clocks := clocks + 1;
                end if;
                if (iter = 256 ) then
                  next_state := KSA1;
                  clocks := 0;
                  busy_internal := True;
                else
                  next_state := READ_KEY;
                end if;
            elsif (KeyS = '0' and TxtS = '1') then
              busy_internal := True;
              next_state := KSA1;
              clocks := 0;

            elsif (KeYS = '0' and TxtS = '0') then
```

```
                busy_internal := True;
                next_state := KSA1;
                clocks := 0;
                Output <= conv_std_logic_vector(99, 8);
            else
                next_state := ARRRESET;
            end if;

        when KSA1 =>
            if (KeyS = '1' and TxtS = '1') then
                next_state := ARRRESET;
            else
                if (clocks < 256) then
                    if (clocks = 0) then
                        sblockMode <= '0';
                    end if;
                    sblockIndex <= conv_std_logic_vector(clocks, 8);
                    sblockInValue <= conv_std_logic_vector(clocks, 8);
                    clocks := clocks + 1;
                    next_state := KSA1;
                    Output <= "10101010";
                else
                    clocks := 0;
                    j := 0;
                    i := 0;
                    next_state := KSA2;
                end if;
            end if;


        when KSA2 =>
            if (KeyS = '1' and TxtS = '1') then
                next_state := ARRRESET;
            else
                    if (clocks < 2048) then
                    if ((clocks mod 8) = 0) then
                        sblockMode <= '1';
                        sblockIndex <= conv_std_logic_vector(i, 8);
                        keyMode <= '1';
                        keyIndex <= conv_std_logic_vector((i mod iter), 8);
                    elsif ((clocks mod 8) = 2) then
                        j := (j + conv_integer(unsigned(sblockOutValue)) +
conv_integer(unsigned(keyOutValue))) mod 256;
                        temp := sblockOutValue;
                    elsif ((clocks mod 8) = 3) then
                        sblockIndex <= conv_std_logic_vector(j, 8);
                    elsif ((clocks mod 8) = 4) then
                        sblockMode <= '0';
```

```
            sblockIndex <= conv_std_logic_vector(i, 8); -- S[i]
          elsif ((clocks mod 8) = 6) then
            sblockInValue <= sblockOutValue;
            elsif ((clocks mod 8) = 7) then
            sblockIndex <= conv_std_logic_vector(j, 8); --S[j]
            sblockInValue <= temp;
            i := i + 1;
            Output <= conv_std_logic_vector(i, 8);
          end if;

          clocks := clocks + 1;
          next_state := KSA2;
        else
          busy_internal := False;
          iter := 0;
          clocks := 0;
          i := 0;
          j := 0;
          next_state := IN_PROCESS;
        end if;
      end if;

    when IN_PROCESS =>
      if (KeyS = '1' and TxtS = '1') then
        next_state := ARRRESET;
      elsif (KeyS = '1' and TxtS = '0') then
        next_state := ARRRESET;
      elsif (KeyS = '0' and TxtS = '1') then
        if (clocks > (io_period+12)) then
          clocks := -1;
        elsif (clocks > 11) then
          if (clocks = 14) then
            OutReady <= '0';
          end if;
        elsif ((clocks mod 12) = 0) then
          i := (i + 1) mod 256;
          sblockMode <= '1';
          sblockIndex <= conv_std_logic_vector(i, 8);--S[i]
        elsif ((clocks mod 12) = 2) then
          j := (j + conv_integer(unsigned(sblockOutValue))) mod 256;--j = j + s[i] 0+2takty
        elsif ((clocks mod 12) = 3) then
          temp := sblockOutValue;  --temp := S[i] v
          sblockIndex <= conv_std_logic_vector(j, 8);  -- S[j] v
        elsif ((clocks mod 12) = 4) then
          sblockMode <= '0';
          sblockIndex <= conv_std_logic_vector(i, 8); -- S[i]
        elsif ((clocks mod 12) = 6) then
          sblockInValue <= sblockOutValue; -- S[i] <= S[j] 0+2 takty
```

```
                        elsif ((clocks mod 12) = 7) then
                           sblockIndex <= conv_std_logic_vector(j, 8); --S[j]
                           sblockInValue <= temp;  -- S[j] <= S[i] v
                        elsif ((clocks mod 12) = 9) then
                           sblockMode <= '1';
                           sblockIndex <= conv_std_logic_vector((conv_integer(unsigned(sblockOutValue)) +
conv_integer(unsigned(temp)) mod 256),8);
                           elsif ((clocks mod 12) = 11) then
                           Output <= (sblockOutValue xor Input);--XOR 0+2takty
                           OutReady <= '1';
                        end if;

                        clocks := clocks + 1;
                        next_state := IN_PROCESS;
                        next_state := IN_PROCESS;
                        busy_internal := False;
                     end if;

                  when ARRRESET =>
                     Output <= conv_std_logic_vector(66, 8);
                     iter := 0;
                     clocks := 0;
                     i := 0;
                     j := 0;

                     Output <= "00000101";
                     next_state := INIT;

                  when others =>
                    next_state := ARRRESET;

               end case;
               if (busy_internal = True) then
                  Busy <= '1';
               else
                  Busy <= '0';
               end if;
            end if;
               end process;
```

Блок Key

```
entity key is
   generic(
         maxLen : integer := 7;
         maxSize : integer := 255
   );
   port(
         clock : in std_logic;
```

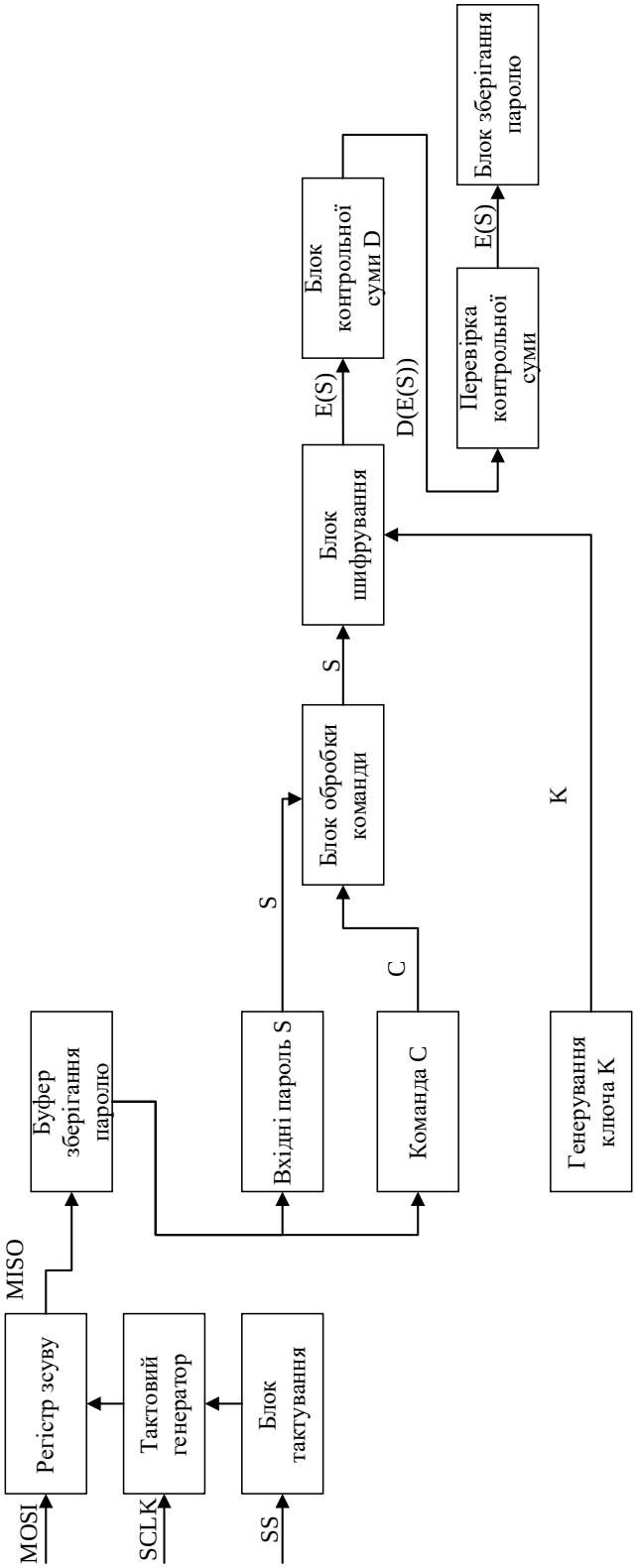
```
            mode : in std_logic;
            reset : in std_logic;
            index : in std_logic_vector(maxLen downto 0);
            inValue : in std_logic_vector(maxLen downto 0);

            outValue : out std_logic_vector(maxLen downto 0)
        );
    end key;

    architecture Behavioral of key is

    type Arr is array (255 downto 0) of std_logic_vector(maxLen downto 0);
    shared variable key : Arr:=(others =>(others => '0'));

    begin
        process (index, clock)
        begin
            if rising_edge(clock) then
                if (reset = '1') then
                    key := (others =>(others => '0'));
                elsif (mode = '0') then
                    key(conv_integer(unsigned(index))) := inValue;
                elsif (mode = '1') then
                    outValue <= key(conv_integer(unsigned(index)));
                end if;
            end if;
        end process;

    end Behavioral;
```

Блок CRC

```
function nextCRC8_D8
    ( Data:  std_logic_vector(7 downto 0);
      CRC:   std_logic_vector(7 downto 0) )
    return std_logic_vector is

    variable D: std_logic_vector(7 downto 0);
    variable C: std_logic_vector(7 downto 0);
    variable NewCRC: std_logic_vector(7 downto 0);

  begin

    D := “01100011”;
    C := “10001110”;
```

```
  NewCRC(0) := D(7) xor D(6) xor D(0) xor C(0) xor C(6) xor C(7);
  NewCRC(1) := D(6) xor D(1) xor D(0) xor C(0) xor C(1) xor C(6);
  NewCRC(2) := D(6) xor D(2) xor D(1) xor D(0) xor C(0) xor C(1) xor
            C(2) xor C(6);
  NewCRC(3) := D(7) xor D(3) xor D(2) xor D(1) xor C(1) xor C(2) xor
            C(3) xor C(7);
  NewCRC(4) := D(4) xor D(3) xor D(2) xor C(2) xor C(3) xor C(4);
  NewCRC(5) := D(5) xor D(4) xor D(3) xor C(3) xor C(4) xor C(5);
  NewCRC(6) := D(6) xor D(5) xor D(4) xor C(4) xor C(5) xor C(6);
  NewCRC(7) := D(7) xor D(6) xor D(5) xor C(5) xor C(6) xor C(7);

  return NewCRC;

end nextCRC8_D8;
```

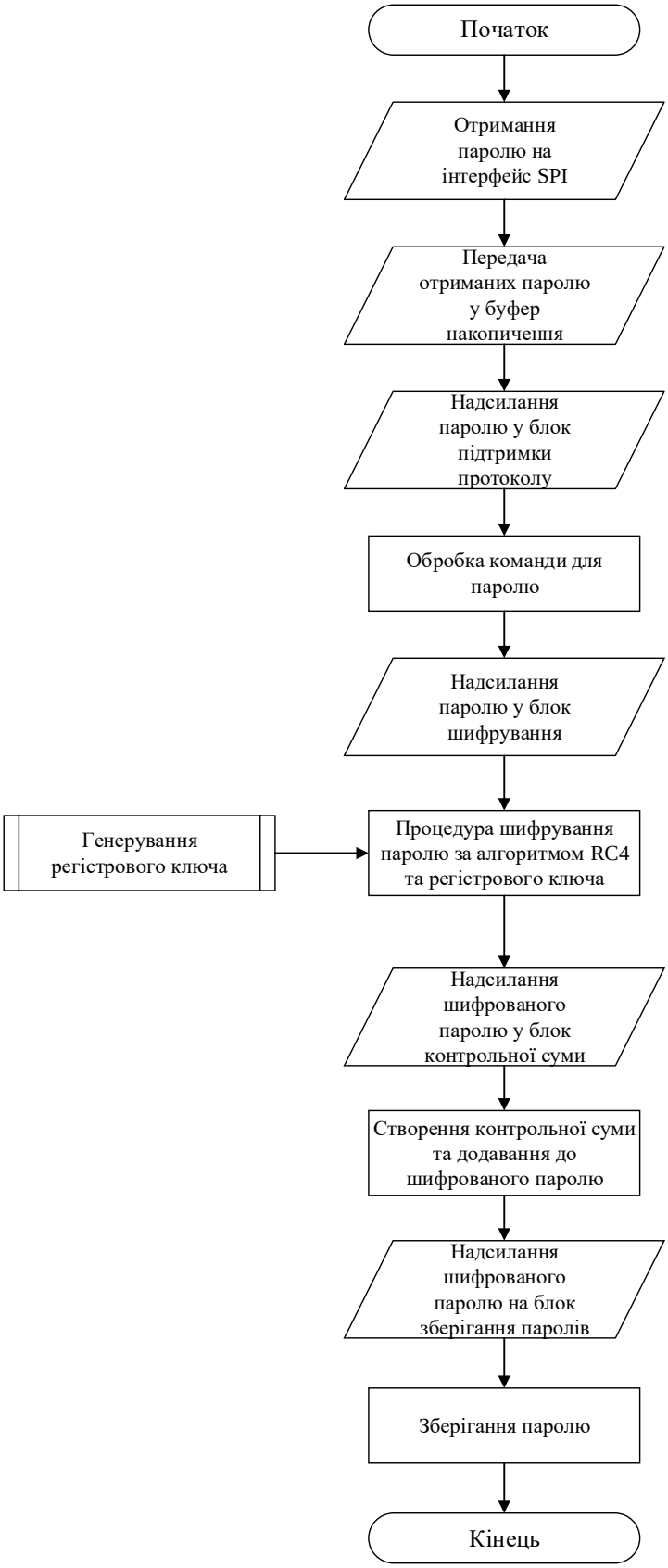
ГРАФІЧНА ЧАСТИНА



					08-20.СМСЗІ.020.05.305 Е1
Змн.	Арк.	№ докум.	Підпис	Дата	
Розроб.		Жакун Г. А.			Мікропроцесорна система засобу зберігання паролів Схема електричної структурної
Перевір.		Баришев В.Ю.			
Реценз.					
Н. Контр.		Баришев В.Ю.			
Затверд.		Лужецький В. А.			

Літ.	Арк.	Аркушів
	1	1

ВНТУ, гр. 1 БС-18мс



					08-20.СМСЗІ.020.05.305 А8	
Змн.	Арк.	№ докум.	Підпис	Дат		
Розроб.		Жакун Г. А.			Мікропроцесорна система засобу зберігання паролів Схема роботи системи	Літ. / Арк. / Аркушів: 1 / 1
Перевір.		Баришев В.Ю.				
Реценз.						ВНТУ, гр. 1 БС-18мс
Н. Контр.		Баришев В.Ю.				
Затверд.		Лужецький В. А..				