УДК 004.056.55

АППАРАТНАЯ РЕАЛИЗАЦИЯ ХЕШ-ФУНКЦИИ CBC MAC В СТАНДАРТЕ DES НА ОСНОВЕ МИКРОКОНТРОЛЛЕРА AT89c52

В.А. Абуздин[1], Р.М. Баринов[2], В.И. Глухих[3], Чан Хоай Бао[4]
Иркутский национальный исследовательский технический университет,
664074, Россия, г. Иркутск, ул. Лермонтова, 83.
Одним из аспектов функциональной безопасности и живучести информационных технологий является обеспечение достоверности информации, недопущение как предумышленного, так и непреднамеренного ее разрушения или искажения. Эта задача в современных информационных системах может решаться с помощью хэш-функций. В настоящее время большую значимость приобретают высокоскоростные криптографические хэш-функции. Такие хэш-функции могут быть весьма эффективно реализованы аппаратно, например, на программируемых логических интегральных схемах (ПЛИС). В данной работе была реализована хэш-функция CBCMAC для алгоритма шифрования данных DES на основе микроконтроллера AT89c52.
Ключевые слова: хэш-функция; DES; CBC MAC; микроконтроллер.

HARDWARE IMPLEMENTATION OF CBC MAC HASH FUNCTION USING DES STANDARD BASED ON AT89c52 MICROCONTROLLER

V. Abuzdin, R. Barinov, V. Glukhikh, Tran Hoai Bao
Irkutsk National Research Technical University,
83 Lermontov Str., Irkutsk, 664074, Russia.
One of the aspects of functional safety and durability of information technologies is ensuring the reliability of information, the prevention of both its intentional and unintentional destruction or distortion. This problem in modern information systems can be solved using hash functions. Currently, high-speed cryptographic hash functions are becoming increasingly important. Such hash functions can be quite effectively implemented in hardware, e.g. in programmable logic devices (PLD). In this paper CBC MAC hash function has been implemented for quite a DES encryption algorithm based on AT89c52 microcontroller.
Keywords: hash function; DES; CBC MAC; microcontroller.

Стремительное и повсеместное внедрение информационных технологий во все сферы человеческой деятельности привело к актуализации задач обеспечения функциональной безопасности и живучести информационных и информационно-управляющих систем. Отказы в работе таких систем в ряде областей их применения могут приводить к недопустимому ущербу, а подчас и к катастрофическим последствиям. Одним из аспектов их функциональной безопасности и живучести является обеспечение достоверности информации, недопущение как предумышленного, так и непреднамеренного ее разрушения или искажения. Эта задача в современных информационных системах может решаться с помощью хеш-функций [1].

В условиях повышения требований к защищённости информации, передаваемой по различным каналам связи и повышения скоростей передачи информации, большую важность приобретают высокоскоростные криптографические хэш-функции. Такие хэш-функции могут быть весьма эффективно реализованы аппаратно, например, на программируемых логических интегральных схемах (ПЛИС) [2]. Для реализации была выбрана хэш-функция CBC MAC довольно популярного алгоритма шифрования DES.

Алгоритм шифрования данных DES (Data Encryption Standard) был опубликован в 1977 г. Блочный симметричный алгоритм DES остается пока наиболее распространенным алгоритмом, используемым в системах защиты коммерческой информации.

[1] Абуздин Вячеслав Андреевич, студент, e-mail: seva_2222@istu.edu
Abuzdin Vyacheslav, student, e-mail: seva_2222@istu.edu
[2] Баринов Роман Максимович, студент, e-mail: baroman99@mail.ru
Barinov Roman, student, e-mail: baroman99@mail.ru
[3]Глухих Владимир Иванович, кандидат химических наук, доцент Института кибернетики им. Е.И. Попова, e-mail: augur@irk.ru
Glukhikh Vladimir, Doctor of Chemical Sciences, Associate Professor of the Department of Electrical and Computer Engineering, e-mail: augur@irk.ru
[4] Чан Хоай Бао, студент, e-mail: bao.tranhoai@mail.ru
Tran Hoai Bao, student, e-mail: bao.tranhoai@mail.ru

Алгоритм DES построен в соответствии с методологией сети Фейстеля и состоит из чередующейся последовательности перестановок и подстановок [4]. DES имеет блоки по 64 бит и 16 цикловую структуру сети Фейстеля, для шифрования использует ключ с длиной 56 бит. Алгоритм использует комбинирование нелинейного (S -блок) и линейного (перестановки E, IP, IP-1) преобразований.

В криптографии, CBC MAC является технологией построения аутентификационного кода сообщения из блочного шифра. Сообщение шифруется при помощи некоторого блочного алгоритма шифрования в режиме CBC, для создания цепочки блоков с правилом — каждый блок зависит от надлежащего (верного) шифрования предыдущего. Эта взаимозависимость гарантирует, что изменение в любом бите открытого текста приведёт к изменению конечного зашифрованного блока в сторону, которая не может быть предсказана или высчитана в случае, если ключ блочного шифра не известен [3].

Алгоритм CBC MAC является широко используемым методом для генерации имитовставок (имитовставка, англ. message authentication code – код аутентичности сообщения), основанных на блочных алгоритмах шифрования. На рис.1 приведена блок-схема алгоритма вычисления хеш-функции CBC MAC.

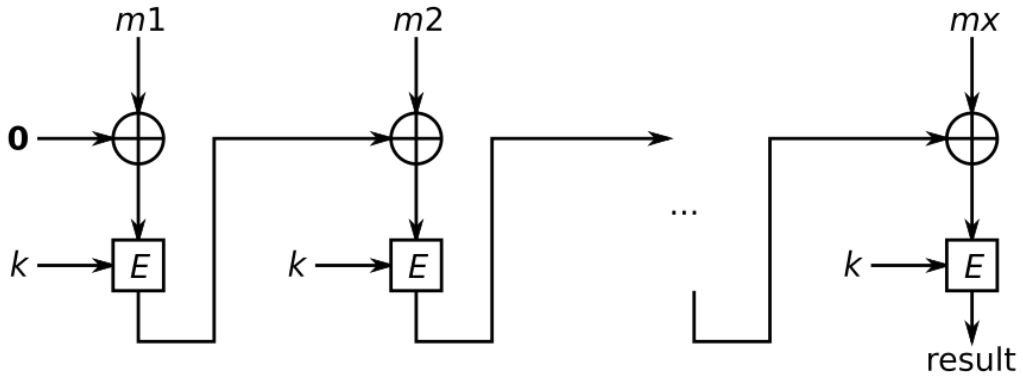

Рис. 1. Блок-схема вычисления хеш-функции CBC MAC

В стандартном варианте использования не предусмотрена процедура выравнивания блоков открытого текста, характерная для более поздних алгоритмов (MD5, SHA-1 и др.). В своей работе M. Bellare, J. Kilian и P. Rogaway [7] доказали применимость алгоритма при фиксированной длине блоков. В других случаях CBC MAC не является криптостойким. Таким образом, было предложено несколько вариантов алгоритма для варьируемой длины сообщения (EMAC, XCBC, TMAC, OMAC).

Несмотря на недостатки, рассматриваемый алгоритм является удобной моделью для использования битового процессора с ядром MCS51 для реализации сложных криптографических алгоритмов.

Цель, задачи работы, материалы и методы

Цель работы – аппаратно реализовать вычисление хеш-функции CBC MAC в стандарте DES на микроконтроллере с архитектурой MCS51.

Для этого использовались следующие материалы и методы: проектирование схемы проводилось в пакете прикладных программ Quartus II, который представляет собой автоматизированную систему сквозного проектирования цифровых устройств на кристаллах ПЛИС фирмы Altera. Для программирования применялся микроконтроллер AT89c52 компании Altera. Программирование микроконтроллера осуществлялось на специализированном языке Ассемблера для микроконтроллера AT89c52.

Реализация

В работе был выбран микроконтроллер компании Altera AT89c52 (AT89s52 – его более современная версия). Выбор этого микроконтроллера не случаен, так как AT89c52 относится к архитектуре МК семейства MCS51. Данная архитектура предусматривает использование битового процессора. В алгоритме шифрования DES используется большое количество как линейных, так и нелинейных перестановок битов, и битовый процессор приходится здесь как не зря кстати.

Битовый процессор можно рассматривать как независимый процессор побитовой обработки. Битовый процессор для выполнения своих команд использует побитово-адресуемую часть ОЗУ. Команды, оперирующие с битами, обеспечивают прямую адресацию 128 битов (0-127) в 16 ячейках

внутреннего ОЗУ (ячейки с адресами 20H-2FH) и прямую побитовую адресацию регистров специального назначения, адреса которых кратны 8.

Каждый из отдельно адресуемых битов может быть установлен в "1", сброшен в "0", инвертирован, проверен. Могут быть реализованы переходы: если бит установлен; если бит не установлен; переход, если бит установлен, со сбросом этого бита; бит может быть перезаписан в (из) разряда переноса. Между любым адресуемым битом и флагом переноса могут быть произведены логические операции "И", "ИЛИ", где результат заносится в разряд флага переноса. Команды побитовой обработки обеспечивают реализацию сложных функций комбинаторной логики и оптимизацию программ пользователя [5].

Битовый процессор может быть наиболее эффективным управляющим процессором для небольших систем управления или выполнять функции битового сопроцессора для процессора, оперирующего байтами или словами. Наиболее целесообразно применение данного управляющего процессора может быть в тех случаях, когда нет возможности применить стандартный микропроцессор с операционной системой реального времени или когда трудозатраты на программирование такой системы велики. Битовый процессор дает существенный выигрыш в быстродействии по сравнению со стандартным микропроцессором при одновременном упрощении программирования [6].

Аппаратное окружение микроконтроллера выполнено в пространстве программируемой логической интегральной схемы (ПЛИС) ep2c8q208n лабораторного стенда UNIL. Схема проекта представлена в Quartus II (рис. 2). Доступ к ресурсам ПЛИС (тактовый генератор, внешняя память и другие модули) осуществляется посредством внутрисхемной сети ПЛИС.

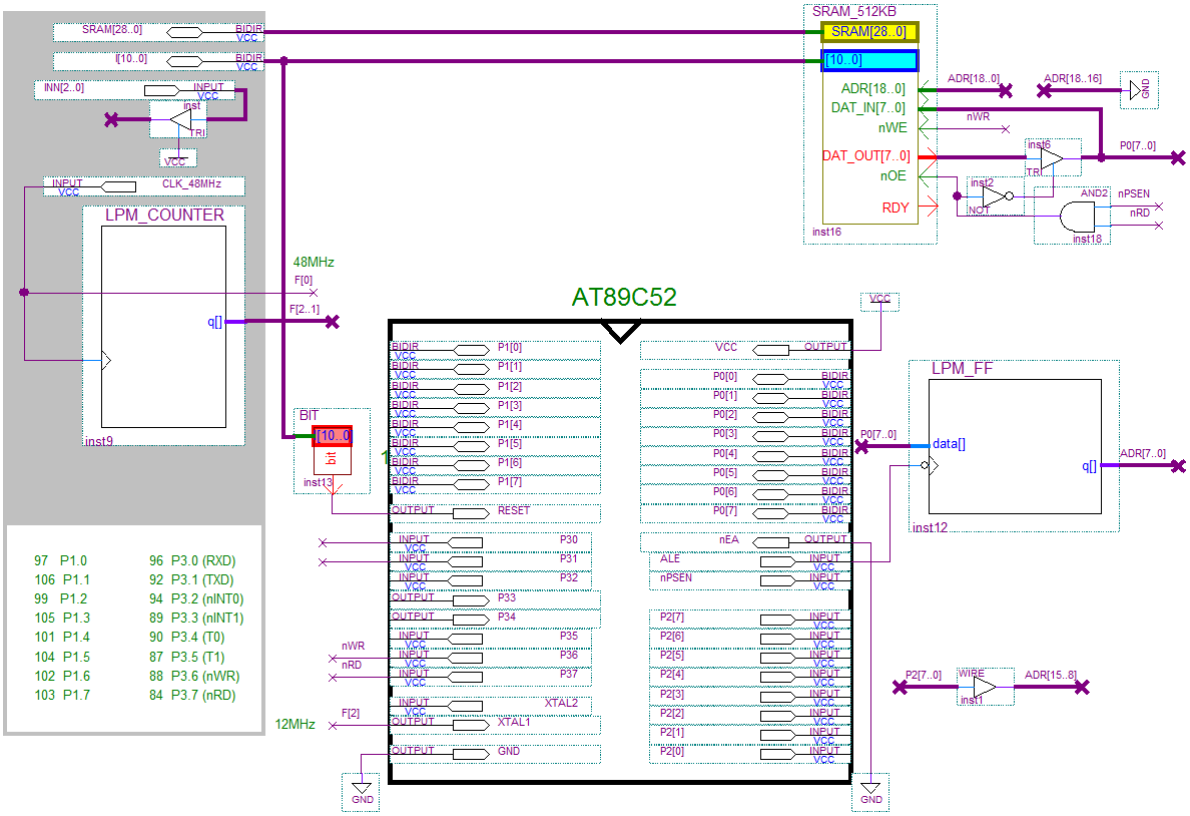

Рис. 2. Схема проекта в Quartus II

Элемент имеет следующие входы:

1. Шина ADR[18..0] – адрес запрашиваемого байта данных.
2. Шина DAT_IN[7..0] – байт данных для записи в файл.
3. nWE – значение логической единицы запрещает запись информации в память, логический ноль разрешает запись.
4. nOE – значение логической единицы запрещает чтение информации из памяти, логический ноль разрешает чтение.

Все без исключения регистры данного контроллера имеют размер 1 байт, что является причиной активного использования внешней по отношению к контроллеру памяти SRAM_512KB.

Запрашиваемый байт данных выводится из памяти через шину DAT_OUT[7..0]. Также элемент имеет две шины для взаимодействия с ПЛИС.

Данный элемент используется для хранения программы контроллера, а также для размещения промежуточных результатов и демонстрации вычисленного значения. Контроллер задаёт адрес ячейки памяти и вводит запрошенные данные посредством выводов порта P0.6 и 7 выводы порта подключены к входам управления записью и чтения nWE и nOE соответственно.

Имеется два типа команд, которые отличаются тем, что обеспечивают 8-битовый или 16-битовый косвенный адрес к внешнему ОЗУ данных. В первом случае содержимое R0 или R1 в текущем банке регистров обеспечивает 8-битовый адрес, который мультеплексируется с данными порта P0. Для расширения дешифрации ввода-вывода или адресации небольшого массива ОЗУ достаточно восьми бит адресации. Если применяются ОЗУ, немного больше чем 256 байт, то для фиксации старших битов адреса можно использовать любые другие выходы портов, которые переключаются командой, стоящей перед командой MOVX.

Во втором случае, при выполнении команды MOVX указатель данных DPTR генерирует 16-битовый адрес. Порт P2 выводит старшие 8 бит адреса (DPH), а порт P0 мультеплексирует младшие 8 бит адреса (DPL) с данными. Эта форма является эффективной при доступе к большим массивам данных (до 64К байт), так как для установки портов вывода не требуется дополнительных команд.

В соответствии с этими задачами адресное пространство памяти было поделено на следующие условные блоки:

1) 0003h – 0100h – блок размещения временных различных данных для подсчёта;
2) 0060h – 0067h – предыдущий зашифрованный пакет;
3) 0070h – 0077h – пакет для шифрования;
4) 0080h – 0086h – 7-байтный ключ;
5) 0101h – 1551h – рабочая программа, содержащая модули и функции на машинном языке;
6) 1FF8h – 1FFFh – подсчитанная хеш-функция DES CBC MAC;
7) 2000h – FFFFh – память, предназначенная для загрузки данных, для которых требуется подсчитать хеш-функцию (макс 56 кбайт). Подсчёт осуществляется либо до встречи нулевого байта (как принято в языках высокого уровня, либо до окончания памяти).

Для реализации хеш-функции CBC MAC (табл. 1, рис. 3) на языке Ассемблер были разработаны следующие функции.

Таблица 1

Функции программы на языке Ассемблер

Функция	Краткое описание
setNullRsBP	Сброс быстрых регистров микроконтроллера R0-R3 и регистров 20h-2Fh битового процессора
modBlock	Сложение по модулю предыдущего подсчитанного и нового пакета
DES	Шифрование нового пакета посредством алгоритма DES (рис. 4)
IP	Первоначальная перестановка битов входного пакета
E	Функция расширения E для пакета R шифрования на каждой итерации функции Фейстеля
Si	Преобразования S_i для пакета после функции расширения E и вывод 32 битного блока
P	Последняя перестановка битов для каждой итерации подсчёта функции Фейстеля
keyCODO	Получение ключа 56 бит и первоначальная перестановка его битов, деление ключа на два равных блока C_0 и D_0 (по 28 бит)
cycleKey	Циклический левый сдвиг битов частей ключа (C_0 и D_0) (рис. 5)
getKey	Конечная перестановка битов ключа и получение его 36-битного представления
EmodKey	Сложение по модулю пакета после расширения E с ключом и преобразование в 32 битное представление
IP-1	Конечная перестановка битов пакета
LmodF	Сложение по модулю пакета R(вторые 32 бита) и L (первые 32 бита), запись результатов для следующей итерации функции Фейстеля

Алгоритмы основных функций для подсчёта хеш-функции CBC MAC представлены на следующих блок-схемах (рис. 3–5).

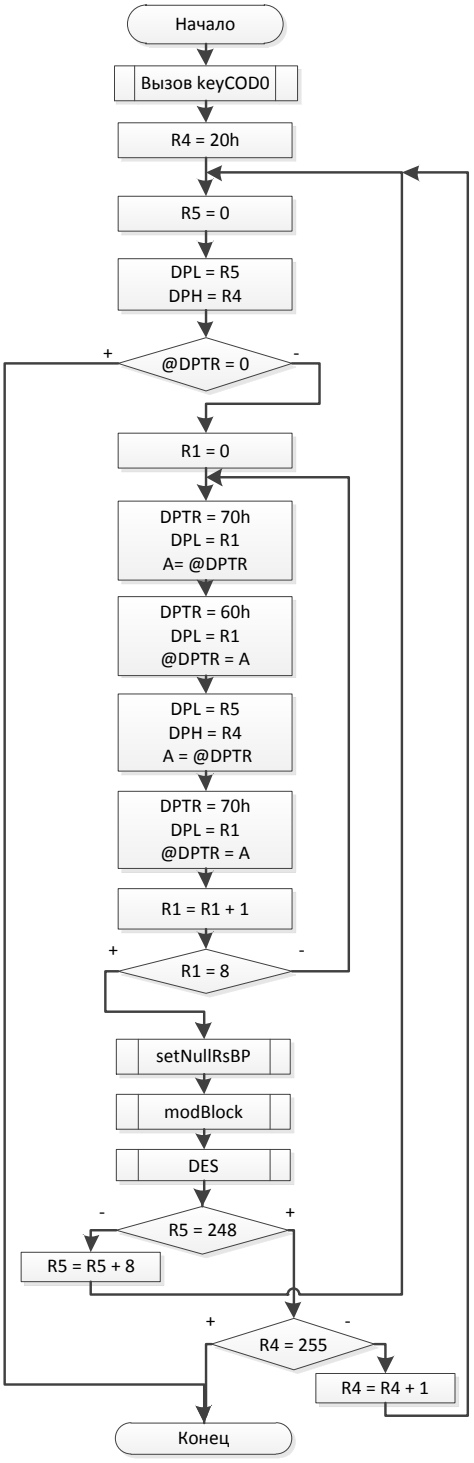

Рис. 3. Блок-схема алгоритма для подсчёта хеш-функции CBC MAC

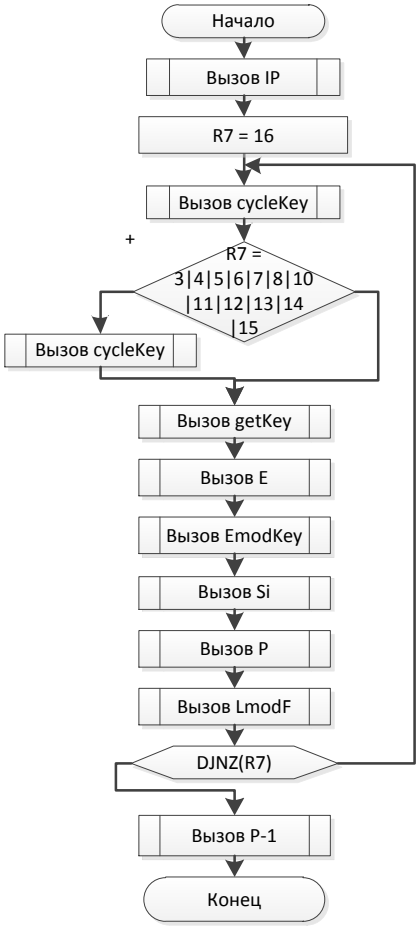

Рис. 4. Блок-схема алгоритма для подсчёта DES

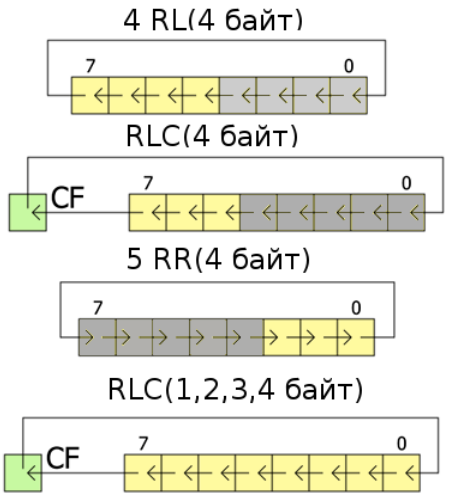

Рис. 5. Схема левого циклического сдвига блоков C_0 и D_0 для ключа

После прохождения всего цикла аппаратной реализации вычисления хеш-функции CBC MAC в стандарте DES на битовом процессоре с ядром MCS51 для тестирования и проверки результатов вычисления была написана программа на языке высокого уровня. Результаты вычисления хеш-функции для разных входных данных и разных ключах посредством программной и аппаратной реализации приведены в табл. 2.

Таблица 2

Результаты вычисления хеш-функции CBC MAC в стандарте DES посредством программной и аппаратной реализации

Данные, для которых подсчитывается хеш-функция (char)	Ключ (char)	Результат аппаратного вычисления хеш-функции (hex)	Результат программного вычисления хеш-функции (hex)
00000000	12345678	3D 75 95 A9 8B FF 80 9D	3D 75 95 A9 8B FF 80 9D
12345678	12345678	96 D0 02 88 78 D5 8C 89	96 D0 02 88 78 D5 8C 89
12345678	44556677	AF 53 7D 46 18 EF 32 4E	AF 53 7D 46 18 EF 32 4E
Знания для себя, достижения для отечества!	12345678	CB 99 03 40 37 E3 BB 29	CB 99 03 40 37 E3 BB 29

Результаты вычисления хеш-функции CBC MAC как в аппаратной, так и в программной реализации одинаковы. Следовательно, можно сделать вывод, что алгоритм реализован правильно.

Библиографический список

1. Yantao Li Parallel Hash function construction based on chaotic maps with changeable arameters [Text] / Li Yantao , Di Xiao, Shaojiang Deng, Qi Han, Gang Zhou // Neural Computing and Applications – 2011. Vol. 20, № 8. P. 1305–1312.

2. Klyucharev P.G. Kriptograficheskie khesh-funktsii, osnovannye na obobshchennykh kletochnykh avtomatakh [Cryptographic hash functions based on generalized cellular automata]. Nauka i obrazovanie MGTU im. N.E. Baumana [Science and Education of the Bauman MSTU], 2013, no. 1. DOI: 10.7463/0113.0534640.

3. E. Petrank and C. Rackoff. CBC MAC for real-time data sources. J.Cryptology, vol. 13, no. 3, pp. 315–338, Springer-Verlag, 2000.

4. Глухих В.И. Информационная безопасность и защита данных: учеб. пособие. Иркутск : Изд-во ИрГТУ, 2012. 246 с.

5. Команды битового процессора // Самый информированный сервер микроэлектроника, описания [Электронный ресурс]. URL: http://www.gaw.ru/html.cgi/txt/doc/micros/mcs51/asm/com_bit.htm (Дата обращения: 02.12.2015).

6. Каршенбойм И. Микропроцессор своими руками — Битовый процессор // Компоненты и технологии. 2003. С. 77.

7. Appears in Journal of Computer and System Sciences, Vol. 61, No. 3, Dec 2000, pp. 362 399. Preliminary version was in Advances in Cryptology – Crypto 94 Proceedings, Lecture Notes in Computer Science Vol. 839, Y. Desmedt ed., Springer-Verlag, 1994.