УДК 004.056

Л.В. ДЕРБУНОВИЧ, д-р техн. наук, проф. каф. АУТС НТУ «ХПИ»,
Д.Г. КАРАМАН, аспирант каф. АУТС НТУ «ХПИ»

АППАРАТНАЯ РЕАЛИЗАЦИЯ ПРЕОБРАЗОВАНИЯ SUBBYTES АЛГОРИТМА ШИФРОВАНИЯ RIJNDAEL (AES)

Проведено аналіз існуючих рішень апаратної реалізації підстановлюючого блоку S-Box, що є основою перетворення SuBytes алгоритму шифрування Rijndael. Виконано синтез, моделювання та верифікацію динамічних і функціональних характеристик найбільш поширених варіантів на основі їхніх описів мовою VHDL.

Analysis of existing solutions for substitution module S-Box hardware implementations which is the basic part of SubBytes transformation in Rijndael encryption algorithm is provided. Synthesis, modeling and dynamic and functional characteristics verification for the most popular options based on appropriate VHDL source codes are performed.

Введение. Криптографические методы играют важную роль в сохранении и передаче конфиденциальных данных. Потребность в защите информации отображается широким выбором алгоритмов и стандартов шифрования, которые можно разделить на две группы: асимметричные [1, 2] и симметричные[3, 4]. Симметричные алгоритмы по всем параметрам являются намного более простыми в реализации и быстрыми при функционировании, чем ассиметричные. Поэтому их используют в устройствах хранения и передачи данных с большими требованиями по скорости к потокам обрабатываемой информации[5, 6, 7].

Алгоритм *Rijndael* является блочным симметричным алгоритмом шифрования. Этот алгоритм был принят Национальным Институтом Стандартов и Технологий США (*NIST*) 26 ноября 2001 года как стандарт для использования в коммерческих или государственных структурах для защиты конфиденциальной информации (не сопряженной с государственными тайнами) [3]. С принятием его как стандарта (*Advanced Encryption Standard, AES*) с него были сняты все патентные ограничения, таким образом, его можно использовать в любых проектах и в любом виде без каких-либо отчислений.

Не смотря на то, что в качестве стандарта криптостойкого шифрования этот алгоритм был принят только в США, в наши дни он используется практически повсеместно. В процессе конкурсного отбора среди нескольких десятков претендентов, он был тщательно рассмотрен и одобрен ведущими специалистами и лабораториями по разработке криптографических средств компьютерной безопасности как один из самых надежных алгоритмов шифрования по крайней мере на ближайшие 30 лет.

Алгоритм *Rijndael* представляет собой цикл обработки исходного — открытого — текста, каждая итерация которого состоит из набора четырех по-

следовательных преобразований: *SubBytes*, *ShiftRows*, *MixColumns* и *AddRoundKey* (рис. 1).

Варианты аппаратной реализации преобразования *SubBytes* являются объектом анализа в данной статье.

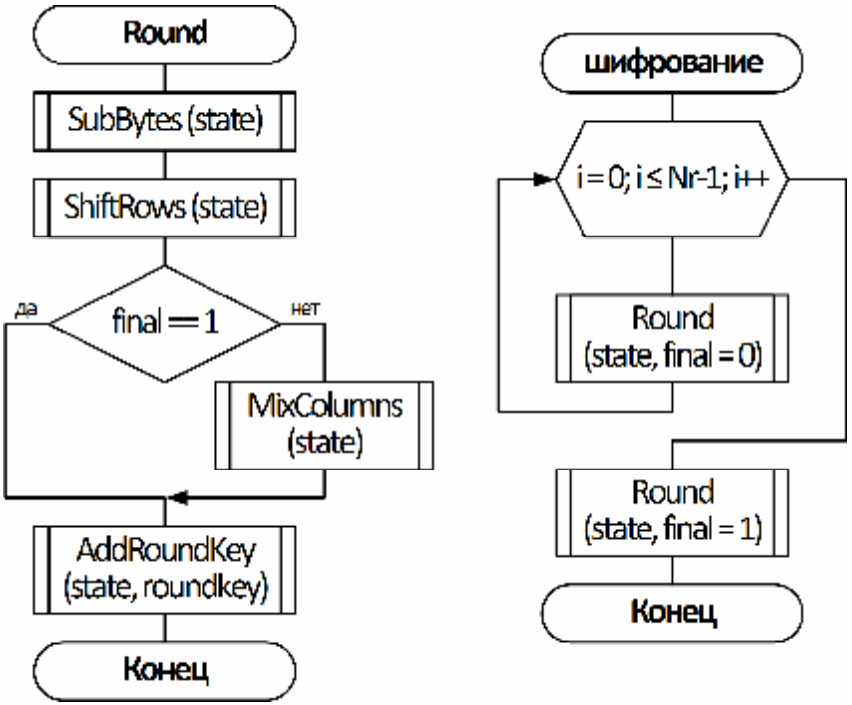

Рис. 1. Схема алгоритма *Rijndael*

Анализ структуры. Преобразованию *SubBytes* во многих публикациях уделено много внимания по той причине, что оно является наиболее ресурсоемким для аппаратных реализаций модулей шифрования на базе алгоритма *Rijndael*. По различным источникам это преобразование занимает от 84% [8] до 90%[9] от общих аппаратных затрат на реализацию всего модуля шифрования в зависимости от способа реализации как самого преобразования, так и алгоритма в целом.

Технически само преобразование *SubBytes* представляет собой замену исходного байта данных байтом, сгенерированным по определенному стандартом (описанием алгоритма) математическому закону, который обеспечивает наименьшую степень корреляции между значениями этих двух байт. Авторы алгоритма определили в качестве такого закона 2 операции: нахождение обратного по умножению в поле $GF(2^8)$ и аффинное преобразование.

При расшифровывании обе операции применяются в обратном порядке, только вместо прямого аффинного преобразования выполняется обратное аффинное преобразование (рис. 2).

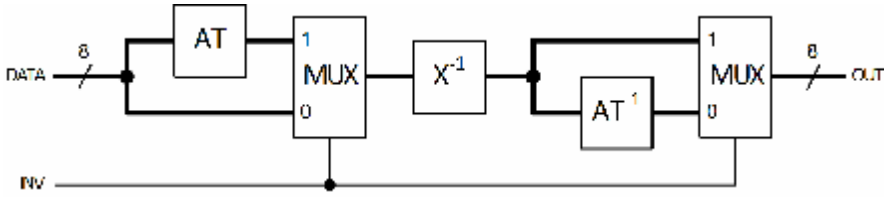

Рис. 2. Структура универсального блока *S-Box*

Структурно, преобразование, согласно авторам, выполнено в виде одного блока, который они назвали *S-Box* и который последовательно обрабатывает каждый байт исходного состояния (рис. 3).

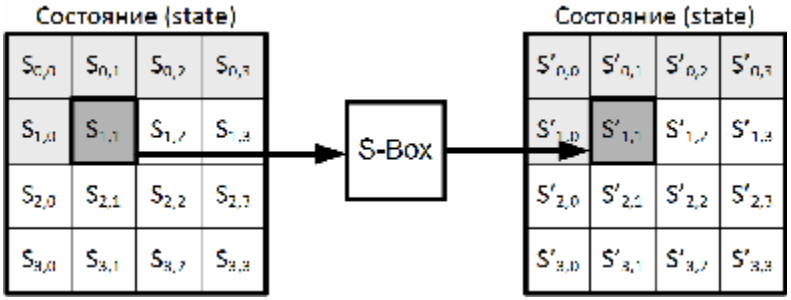

Рис. 3. Выполнение преобразования *SubBytes*

Цель статьи: анализ существующих подходов к аппаратной реализации блока *S-Box*, описание схемных решений на языке *VHDL*, моделирование и верификация их динамических и функциональных характеристик с целью выбора наиболее эффективных решений.

Реализация преобразования *SubBytes* сводится к синтезу блока *S-Box* и организации алгоритма замены: либо 1 *S-Box* последовательно обслуживает все 16 байт исходного состояния, либо 16 блоков, которые параллельно выполняют преобразование за один такт.

Существует два наиболее распространенных и проверенных способа реализации блока *S-Box*: в виде подстановочной таблицы на 256 байт, либо комбинационной схемы, тем или иным способом реализующей описанную в стандарте эквивалентную функцию.

Подстановочная таблица. В первом способе предполагается использование ячейки памяти для хранения всех возможных значений подстановочно-

го блока. Эти значения были рассчитаны заранее и приведены в виде таблицы в тексте стандарта (рис. 4).

S-Box	0	1	2	3	4	5	6	7	8	9	a	b	c	d	e	f
0	63	7c	77	7b	f2	6b	6f	c5	30	1	67	2b	fe	d7	ab	76
1	ca	82	c9	7d	fa	59	47	f0	ad	d4	a2	af	9c	a4	72	c0
2	b7	fd	93	26	36	3f	f7	cc	34	a5	e5	f1	71	d8	31	15
3	4	c7	23	c3	18	96	5	9a	7	12	80	e2	eb	27	b2	75
4	9	83	2c	1a	1b	6e	5a	a0	52	3b	d6	b3	29	e3	2f	84
5	53	d1	0	ed	20	fc	b1	5b	6a	cb	be	39	4a	4c	58	cf
6	d0	ef	aa	fb	43	4d	33	85	45	f9	2	7f	50	3c	9f	a8
7	51	a3	40	8f	92	9d	38	f5	bc	b6	da	21	10	ff	f3	d2
8	cd	0c	13	ec	5f	97	44	17	c4	a7	7e	3d	64	5d	19	73
9	60	81	4f	dc	22	2a	90	88	46	ee	b8	14	de	5e	0b	db
a	e0	32	3a	0a	49	6	24	5c	c2	d3	ac	62	91	95	e4	79
b	e7	c8	37	6d	8d	d5	4e	a9	6c	56	f4	ea	65	7a	ae	8
c	ba	78	25	2e	1c	a6	b4	c6	e8	dd	74	1f	4b	bd	8b	8a
d	70	3e	b5	66	48	3	f6	0e	61	35	57	b9	86	c1	1d	9e
e	e1	f8	98	11	69	d9	8e	94	9b	1e	87	e9	ce	55	28	df
f	8c	a1	89	0d	bf	e6	42	68	41	99	2d	0f	b0	54	bb	16

Рис. 4. Подстановочная таблица блока *S-Box* из стандарта *AES*

Эта таблица реализуется в виде модуля ПЗУ. На адресные входы подается исходный байт, а на выходной порт поступает соответствующее ему значение из приведенной выше таблицы. Основное достоинство данного решения – простота реализации, которое зачастую является решающим для срочных проектов, в которых к скорости работы не предъявляется жестких требований. Следует упомянуть, что для реализации всего преобразования *SubBytes* необходимо максимум 16 вышеописанных модулей памяти, или минимум один, но с дополнительной управляющей логической схемой, которая последовательно будет подавать по очереди все 16 байт исходного состояния.

Существуют определенные сложности при реализации этого варианта на ПЛИС. Формально, описание на *VHDL* сводится к объявлению нового типа данных – массива байт необходимой размерности – и константы этого типа. Однако не все синтезирующие пакеты в состоянии правильно обработать такие структуры и разработчики этих пакетов настоятельно рекомендуют использовать специализированные типовые компоненты из встроенных библиотек, поставляемых вместе с синтезатором [10]. Такое решение не очень благоприятно сказывается на универсальности проекта и влечет за собой привязку к конкретным средствам синтеза. Не смотря на это, реализация блока S-Box в виде модуля памяти весьма популярна не только в академических

исследованиях, таких как [6], [11], но и в конкретных проектах, находящих практическое применение , например, модулях беспроводных сетей [12].

Схема на логических элементах. Второй способ, реализация в виде комбинационной схемы, более популярен, чем первый. Он сводится к отысканию логической функции, которая описывала бы математические операции, приведенные в тексте стандарта: аффинное преобразование и нахождение обратного по умножению элемента в поле $GF(2^8)$. В свою очередь, этот способ имеет несколько вариантов решений в зависимости от методов получения результирующей логической функции.

Первый вариант: синтез логической функции по данным таблицы, приведенной выше. Существуют методы и соответствующее программное обеспечение, позволяющие получить требуемую логическую функцию в виде СДНФ или СКНФ. Применяя современные мощные средства синтеза логическую функцию любой сложности можно отобразить в оптимизированную структуру подстановочного блока *S-Box* для реализации на ПЛИС [8]. Стоит заметить, один из авторов алгоритма *Rijndael* Винсент Реймен (*Vincent Rijmen*) в своей статье [13] признает подобное решение наиболее оптимальным и приемлемым для аппаратных реализаций *AES*.

Второй подход: использование специальных методов, которые сводят вычисления в поле $GF(2^8)$ к вычислениям в конечных полях меньшего порядка для отображения операции нахождения обратного по умножению в логическую функцию. Подобная методика подробно рассмотрена в [14] и опробована в [5], [6] и [9]. Суть ее сводится к представлению элементов поля Галуа $GF(2^8)$ в виде элементов полей меньшего порядка, арифметические операции над которыми можно выполнять на порядок проще и быстрее, чем над элементами поля $GF(2^8)$.

В соответствии с [13], обратное по умножению может быть рассчитано по следующей формуле:

$$(bx+c)^{-1} = b(b^2B+bcA+c^2)^{-1}x+(c+bA)(b^2B+bcA+c^2)^{-1} \qquad (1)$$

где b – старший полубайт, а c – младший полубайт элемента поля $GF(2^8)$, A и B – коэффициенты характеристического неприводимого полинома x^2+Ax+B.

В [14] был использован неприводимый полином $x^2+x+\lambda$. Следовательно, $A = 1$ и $B = \lambda$, а формула (1) принимает следующий вид:

$$(bx+c)^{-1} = b(b^2\,l+c(b+c))^{-1}x+(c+b)(b^2\,l+c(b+c))^{-1} \qquad (2)$$

Согласно этой формуле нахождение обратного по умножению элемента поля $GF(2^8)$ сводится к арифметическим операциям над элементами поля

$GF(2^4)$, к которым относятся b и c. По формуле (2) в [14] была составлена следующая схема:

Рис. 5. Схема нахождения обратного по умножению элемента в поле $GF(2^8)$

Подстановочный блок *S-Box*, выполненный по этой схеме, является наиболее экономичным и быстродействующим решением. Кроме того, такой вариант больше всего подходит для конвейерной обработки данных, а так же реализации различных механизмов обнаружения и устранения сбоев и ошибок.

Представленные решения были описаны на языке *VHDL*, проведен их синтез и моделирование средствами пакета *Xilinx ISE* 10.1. Анализ особенностей полученных схем а так же временных параметров их функционирования показывает, что по абсолютным показателям реализация в виде комбинационной схемы в 1,4 раза выигрывает по аппаратным затратам, но в 2,1 раза проигрывает по быстродействию варианту в виде модуля памяти. Такое соотношение временных характеристик обусловлено тем, что современные ПЛИС типа *FPGA* фирмы *Xilinx* оснащены собственными оптимизированными ячейками памяти, которые при синтезе задействуются синтезатором. Однако в [6] приведены методики, которые позволяют за счет конвейеризации существенно поднять производительность решений на основе комбинационной логики при незначительном перерасходе аппаратных ресурсов кристалла. С учетом приведенных ранее других достоинств, вариант реализации подстановочного блока *S-Box* на основе комбинационной схемы является наиболее предпочтительным.

Выводы. В результате анализа существующих решений аппаратной реализации блока *S-Box* рассмотрены несколько наиболее распространенных вариантов исполнения этого подстановочного блока. На основе описания схемного решения блока *S-Box* на языке *VHDL* выполнены моделирование и верификация динамических и функциональных характеристик нескольких вариантов аппаратных схем и определены их достоинства и недостатки.

Список литературы: 1. *Коваленко И.Н., Кочубинский А.И.* Асимметричные криптографические алгоритмы // Кибернетика и системный анализ. – 2003. –№ 4. – с. 95-102. **2.** *Mazzeo A., Romano L., Saggese G.P., Mazzocca N.* "FPGA-Based Implementation of a Serial RSA Processor," date,pp.10582, Design, Automation and Test in Europe Conference and Exhibition (DATE'03), 2003. **3.** National Institute of Standards and Technology (NIST), " Federal Information Processing Standard 197, The Advanced Encryption standard (AES)", http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf , 2001. **4.** ГОСТ 28147-89 Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования. – Введ. 01.07.1990. – М.: ИПК Издательство стандартов, 1990. – 28 с. **5.** *Stefan Mangard, Manfred Aigner, Sandra Dominikus*. A Highly Regular and Scalable AES Hardware Architecture. // IEEE Transactions on Computers. – 2003. – Volume 52 , Issue 4. – p. 483-491. 6. *Alireza Hodjat, Ingrid Verbauwhede*. Minimum Area Cost for a 30 to 70 Gbits/s AES Processor // Proceedings of IEEE computer Society Annual Symposium on VLSI. – 2004. – ISVLSI'04. – p. 83-88. **7.** *Лившиц Н.В.* Криптографические методы защиты информации. // Безопасность информационных технологий. – 1998. – Т.1. – С. 61-62. **8.** *Song J. Park*. Analysis of AES Hardware Implementations. // ECE 679 Advanced Security and Cryptography. – 2003. – Oregon State University, USA; [электронный ресурс]. – Режим доступа: http://islab.oregonstate.edu/ koc/ece679/project/ 2003/park.pdf. **9.** *G. Di Natale, Flottes M. L., Rouzeyre B.* On-Line Self-Test of AES Hardware Implementations. // DSN 2007 Workshop on Dependable and Secure Nanocomputing In conjunction with the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks. – June 28, 2007. – Edinburgh International Conference Centre, UK. **10.** *Сергиенко А.М.* VHDL для проектирования вычислительных устройств. – К.: ЧП "Корнейчук", ООО "ТИД'ДС", 2003. – 208 с. **11.** *Nadia Nedjah, Luiza de Macedo Mourelle*. A Versatile Pipelined Hardware Implementation for Encryption and Decryption using AES // High Performance Computing for Computational Science. – 2007. – Volume: 4395/2007. – p. 249-259. **12.** *Arshad Aziz, Nassar Ikram*. Hardware Implementation of AES-CCM for Robust Secure Wireless Network. // ISSA 2005 New Knowledge Today Conference, 29 June – 1 July 2005. – Balalaika Hotel, Sandton, South Africa. **13.** *Vincent Rijmen*. Efficient Implementation of the Rijndael S-box. [Электронный ресурс] – Режим доступа: https://www.iaik.at/research/krypto/AES/old/~rijmen/rijndael/sbox.pdf. **14.** *Akashi Satoh, Sumio Morioka, Kohji Takano, Seiji Munetoh*. A Compact Rijndael Hardware Architecture with S-Box Optimization // Advances in Cryptology - ASIACRYPT 2001. –2001. – Volume 2248/2001. – p.239-254.

Поступила в редколлегию 17.12.08