ISSN 1992-6502 (Print) Вестник УГАТУ ISSN 2225-2789 (Online)
2015. Т. 19, № 4 (70). С. 138–148 http://journal.ugatu.ac.ru

УДК 004.056.55:004.272.25

# К ВОПРОСУ ОБ АППАРАТНОЙ РЕАЛИЗАЦИИ МОДУЛЯ ПОТОКОВОГО ШИФРОВАНИЯ ДЛЯ КОМПЛЕКСНОЙ СИСТЕМЫ ЗАЩИТЫ ИНФОРМАЦИИ

**Т. Е. Михайлюк[1], С. В. Жернаков[2]**

[1] realotoim@mail.ru, [2] zhsviit@mail.ru

ФГБОУ ВПО «Уфимский государственный авиационный технический университет» (УГАТУ)

*Поступила в редакцию 22.10.2015*

**Аннотация.** Рассматривается оптимизированная аппаратная реализация ГОСТ 28147-89 в режиме гаммирования на основе ПЛИС для средств высокопроизводительных комплексных систем защиты информации. Анализируются современные методы построения потоковых шифраторов. Приводятся результаты моделирования синтезированного IP-ядра в среде Xilinx ISE Design Suite. Оцениваются скоростные показатели архитектур потоковой обработки данных.

**Ключевые слова:** ГОСТ 28147-89; ПЛИС; потоковое шифрование.

## ВВЕДЕНИЕ

Одним из наиболее актуальных направлений защиты информации является криптография. Существует достаточно большое количество криптографических алгоритмов, однако не каждый из них способен обеспечить высокую эффективность защиты информации и скорость обработки. Наиболее высокая скорость шифрования достигается путем применения поточного шифрования. В качестве официального стандарта криптографической защиты на территории Российской Федерации принят стандарт ГОСТ 28147-89 [1]. На данный момент отечественный стандарт обладает высокой криптостойкостью к существующим потенциальным атакам и позволяет реализовать поточное шифрование в режиме гаммирования.

Алгоритм ГОСТ 28147-89 имеет иерархическую структуру и включает в себя три уровня. Базовой структурой алгоритма является основной шаг криптопреобразования. На следующем уровне используется понятие базовых циклов: цикл шифрования, цикл расшифрования и цикл выработки имитовставки. Практическое значение имеют алгоритмы верхнего уровня, построенные на базовых схемах. Уникальность ГОСТ 28147-89 состоит в том, что данный стандарт представляет собой семейство алгоритмов шифрования, предназначенных для определенных целей криптографии.

В настоящее время рынок отечественных средств криптографической защиты информации представлен рядом устройств, использующих данный стандарт шифрования. Зачастую такие средства представляют собой программно-аппаратные комплексы, в которых вычислительным ядром криптографической функции является процессор общего назначения либо специализированный вычислитель, не обладающий достаточными скоростными характеристиками. Лишь немногие устройства позволяют обрабатывать трафик с производительностью более 1 Гбит/с. Очевидно, что такая скорость шифрования является недостаточной. В связи с этим концентрация больших объемов информации различного назначения требует постоянного совершенствования аппаратных и программных средств ее передачи и обработки.

Несмотря на то, что в зарубежной [2–10] и отечественной [11–16] периодике вопросам построения специализированных вычислителей с высокой производительностью, ориентированных на шифрование больших объемов информации, уделяется значительное внимание, результаты их практического применения, с учетом специфики конкретной предметной области и этапов моделирования освещены недостаточно. Поэтому в данной работе в дальнейшем рассматриваются вопросы аппаратного построения алгоритмов шифрования, их структурная и функциональная реализации, с целью дальнейшего использования при решении комплексных задач в системах защиты информации.

## СУЩЕСТВУЮЩИЕ ПОДХОДЫ К РЕАЛИЗАЦИИ АЛГОРИТМА

Основой большинства существующих методов реализации криптографических протоколов является их ориентация на программные модули, оптимизированные под определенный тип процессора. У такого подхода есть свои преимущества и недостатки. Так, в работе [15] приводятся результаты программной реализации алгоритма в режиме простой замены на современных процессорах Intel и графических процессорах NVIDIA. Вместе с тем в современных процессорах архитектуры x86 аппаратное ускорение шифрования реализовано только для стандарта AES (набор инструкций AES NI). Этот стандарт базируется на вычислительной математике, жестко привязанной к его специфике, и ускорить другие стандарты шифрования можно, только если их выполнение связано с операциями алгоритма AES (например, Camellia) [15].

В качестве прототипа использован серверный компьютер использующий два процессора Intel Xeon E5-2697 v3 и ведеокарту Nvidia GeForce GTX 750 (AMD HD 7790 в одном из тестов). Для эффективного использования ресурсов процессора определенного типа предлагается многопоточная обработка. Таким образом достигается максимальная производительность шифрования. Полученные результаты приведены в табл. 1.

Таблица 1

**Результаты тестирования процессоров**

| Название процессора | Скорость шифрования, МБ/с |
|---|---|
| Intel HD 2500 | 295 |
| AMD HD 7790 | 1737 |
| Nvidia GTX 750 | 2588 |
| Intel Xeon (GPR) | 2682 |
| Intel Xeon (AVX) | 9425 |
| Intel Xeon (AVX2) | 13133 |

Несмотря на впечатляющие результаты, необходимо отметить моменты, которые могут оказать негативное влияние на характеристики конечного продукта. Во-первых, известно, что программные реализации являются уязвимыми к модификации кода, что требует мер по обеспечению безопасной среды шифрования. Во-вторых, использование серверного компьютера для функции шифрования может оказаться избыточной, в сравнении со специализированной, аппаратной платформой. Отсюда вытекают экономические затраты, увеличивается энергопотребление, а так же массо-габаритные показатели. В-третьих, для уменьшения вероятности удачных атак по побочным каналам необходимо усложнять программу, что приведет к потере скорости шифрования. В-четвертых, применение пространственного параллелизма может уменьшить суммарное ускорение в сетевых шифраторах в связи с последовательным поступлением данных, что говорит о возможном падении эффективности данного метода.

Альтернативным путем решения проблемы повышения производительности и защищенности системы криптографической защиты является возможность использования аппаратной платформы в виде законченного модуля на микросхемах программируемой логики (ПЛИС). Они являются мощным инструментом цифровой электроники. Использование таких кристаллов с большим количеством реконфигурируемых логических элементов позволяет реализовывать криптографические функции со значительным приростом производительности. Микросхемы ПЛИС представлены двумя основными классами: сложные программируемые логические устройства (СПЛУ) и программируемые пользователем вентильные матрицы (ППВМ) [17]. Микросхемы СПЛУ имеют меньшую степень интеграции, поэтому обладают ограниченными вычислительными ресурсами. Они больше приспособлены для реализации интерфейсной логики.

В работе [11] проводится анализ производительности алгоритма на графических процессорах и микросхеме ППВМ по методике RAT. На основе полученных результатов можно сделать вывод о том, что существует ряд особенностей при реализации режима простой замены на ПЛИС. Существующее количество логических блоков в ПЛИС является недостаточным для реализации многопоточного шифрования, как в графических процессорах. За счет реконфигурируемой архитектуры частота тактирования ПЛИС в несколько раз меньше, чем у графического процессора. Время обмена данными между вычислителем и памятью у ПЛИС более чем в два раза превышает аналогичный показатель у графического процессора. Однако, авторами получена не достаточно оптимизированная архитектура алгоритма, приводящая к уменьшению скорости тактирования всей микросхемы ППВМ. В табл. 2 приведены некоторые полученные результаты.

Одним из альтернативных методов ускорения является применение специализированных

криптографических процессоров. Так в [9] рассматривается криптографический процессор на основе ППВМ. Система представляет собой матричный процессор, имеющий 80 элементов обработки. Каждый элементарный процессор имеет специальный набор инструкций, предназначенных для криптографической обработки наиболее известных симметричных и некоторых ассиметричных алгоритмов. При анализе скоростных показателей данного процессора применительно к отечественному алгоритму шифрования в режимах гаммирования и гаммирования с обратной связью получены скорости 51,2 Гб/с и 2,67 Гб/с при количестве циклов обработки равном 96. Эта обработка потребовала задействовать все 4 вычислительных потока, доступных процессору. Таким образом, одному потоку соответствуют скорости 12,8 Гб/с и 683,5 Мб/с. Авторы отмечают, что реализация ГОСТ на данной архитектуре является менее оптимизированной относительно американского стандарта шифрования. Для сравнения приведем некоторые данные полученных результатов (табл. 3).

Видно, что сеть Фейстеля является неоптимизированной для данной реализации шифратора.

В работе [16] рассматриваются варианты реализации программного и аппаратного модулей шифрования по ГОСТ 28147-89 в режиме простой замены. В данной работе рассматривается циклическая аппаратная архитектура на ПЛИС. Особенности аппаратной реализации позволяют сделать вывод о том, что такая архитектура предназначена для низкоскоростных каналов передачи и мобильных приложений. Поэтому применение аппаратной реализации такого типа в высокопроизводительных системах является неприемлемым.

Для повышения производительности последовательных архитектур можно увеличить количество модулей шифрования. Так, в работе [12] получены варианты архитектур криптопреобразователей на основе ПЛИС для шифрования трафика в локальной сети Ethernet (DIX). Предложенные архитектуры отличаются применением параллельных блоков шифрования. Авторы предлагают два варианта ускорителей: параллельный и последовательный.

Параллельная архитектура отличается наличием одного управляющего автомата для каждого потока и одной таблицей замен, что, по сути, приводит к одному макропотоку. Это приводит к простоям ускорителя при несогласованности скоростей поступления данных из канала и их обработки. Для реализации последовательной архитектуры предлагается использовать параллельную архитектуру с подключаемыми блоками шифрования.

Таблица 2

**Сравнительные характеристики шифрования на графических процессорах NVIDIA и ППВМ**

| Показатель | Графический процессор | | | ППВМ |
|---|---|---|---|---|
| | NVIDIA GT 335M | NVIDIA GTX 260 | NVIDIA Tesla C1060 | Altera Arria II GX EP2AGX125 |
| Скорость чтения, Мбайт/с | 1684,1 | 3005,1 | 4265,3 | 748,0 |
| Скорость записи, Мбайт/с | 1595,0 | 2932,7 | 4603,4 | 748,0 |
| Время обмена данными, с | 1,25 | 0,69 | 0,46 | 2,74 |
| Количество операций над элементом данных | 2052 | 2052 | 2052 | 98 |
| Частота вычислителя, МГц | 450 | 576 | 602 | 125 |

Таблица 3

**Производительность специализированного процессора Cryptoraptor**

| Алгоритм шифрования | Число потоков | Число циклов | Скорость шифрования, Гб/с | |
|---|---|---|---|---|
| | | | Режим сцепления блоков шифротекста | Режим счетчика (гаммирования) |
| AES (подстановочно-перестановочная сеть) | 1 | 20 | 6,4 | 128 |
| DES (сеть Фейстеля) | 2 | 48 | 2,67 | 42,67 |
| ГОСТ 28147-89 (сеть Фейстеля) | 4 | 96 | 2,67 | 51,2 |

Несмотря на то что такой подход может помочь, при согласовании входного и обрабатываемого потоков присутствует значительное увеличение аппаратных затрат, так как каждый модуль является независимым, имеет свое ключевое пространство, таблицу замен и автомат управления. Применение такого типа конвейерной обработки является неоптимизированным как по аппаратным ресурсам, так и по согласованию скоростей поступления и обработки информации. Кроме того, предлагается использовать двухпроцессорную систему. При этом нахождение процессора в тракте обработки сетевых пакетов уменьшает пропускную способность всего устройства, а применение дополнительного процессора на кристалле ПЛИС для обработки данных, не включенных в тракт обработки сетевых пакетов, приводит к уменьшению потенциально достижимой скорости и к неоправданной потере ресурсов. Авторами получена скорость шифрования 280 МБ/с.

Анализ имеющихся источников показывает, что существующие методы повышения производительности криптографической обработки по ГОСТ 28147-89 являются недостаточно эффективными для потокового шифрования. Учитывая тот факт, что абсолютное большинство архитектур работают в режиме простой замены, возникает необходимость реализации быстродействующей, оптимальной, защищенной архитектуры для реализации алгоритма в режиме гаммирования.

## ОБОСНОВАНИЕ ПРИМЕНЕНИЯ АЛГОРИТМА ГАММИРОВАНИЯ

Режим гаммирования представляет собой процесс наложения блоков псевдослучайной последовательности на блоки открытых данных с помощью операции «исключающее ИЛИ». При этом для обеспечения максимальной защиты период повторения блоков гаммы должен быть как можно большим. Генерация криптографически стойкой гаммы связана со стойким ключем шифрования. Несмотря на то что рассмотренные архитектуры представлены в качестве поточных шифраторов, вопрос о дополнении блока обрабатываемых данных не кратного блоку шифротекста (использован режим простой замены) не освещен, что является серьезным недостатком. Покажем это.

Шифрование в режиме электронной кодовой книги (режим простой замены) можно представить как частный случай шифрования в режиме счетчика (гаммирования):

$$T_{\text{ш}} = E_k(T_{\text{о}}) = T_{\text{о}} \oplus E_k(Ctr) = 0 \oplus E_k(T_{\text{о}}). \quad (1)$$

Видно, что данный режим аналогичен шифрованию блока открытого текста, заполненного нулями, использующего в качестве псевдослучайного значения счетчика значение открытого блока данных. Однако известно, что в алгоритме для получения значения счетчика используется предварительный цикл шифрования псевдослучайного блока с использованием рекуррентного генератора последовательности чисел (РГПЧ). Блок-схема алгоритма генерации и наложения гаммы представлена на рис. 1.

Режим генерации гаммы шифра начинается с получения вектора инициализации IV. Также его называют синхропосылкой, однако стандарт использует в качестве синхропосылки блок зашифрованного вектора инициализации. Далее зашифрованная синхропосылка обрабатывается в РГПЧ, как показано на рисунке. И, наконец, для получения гаммы подготовленный блок закрывается в режиме электронной кодовой книги.

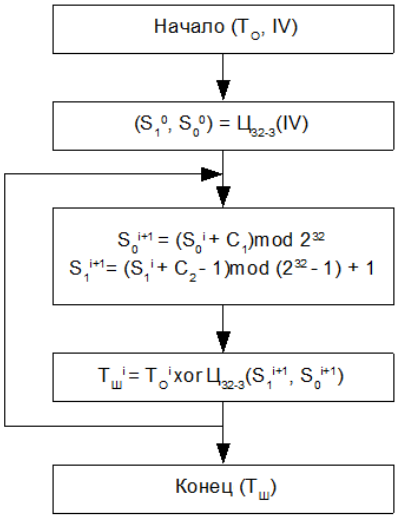


**Рис. 1.** Блок-схема алгоритма шифрования в режиме гаммирования

$$Ctr_0 = E_k(IV). \quad (2)$$

$$Ctr_{i+1} = Sum(Ctr_i). \quad (3)$$

$$T_{\text{ш}} = T_{\text{о}} \oplus E_k(Ctr_{i+1}). \quad (4)$$

Первичное шифрование блока инициализации приводит к криптографически защищенному блоку синхропосылки, позволяя РГПЧ генерировать последовательность псевдослучайных криптографически стойких синхропосылок за короткие промежутки времени, что приводит к увеличению производительности и защищенности обрабатываемых данных. В том случае, если злоумышленнику известен алгоритм шифрова-

ния, таблицы замен, значение вектора инициализации, а также блоков открытого и закрытого текстов, вероятность нахождения ключа методом полного перебора составит $1/2^{256}$. Для режима гаммирования значение синхропосылки при стойкой реализации алгоритма злоумышленнику неизвестно, поэтому вероятность угадывания снижается до значения $1/2^{320}$. Следовательно, режим гаммирования является более стойким к атакам прямого перебора и предпочтительным для потокового шифрования, в то время как стандарт предписывает использование режима простой замены исключительно для шифрования ключевой информации [13].

Так как режим гаммирования с обратной связью не дает явных преимуществ в криптостойкости по сравнению с обычным режимом гаммирования, в работе он не рассматривается. Предотвратить подмену битов блока позволяют алгоритмы аутентификации. Кроме того, этот режим предполагает использование значительных вычислительных ресурсов, многократно превышающих такие затраты для режима гаммирования, и, как следствие, требующих более мощных аппаратных средств. К тому же в данном режиме вычислительные затраты значительно превышают затраты в режиме гаммирования и требуют более мощных аппаратных средств.

## АНАЛИЗ ВАРИАНТОВ АППАРАТНЫХ АРХИТЕКТУР ДЛЯ ПОТОКОВОЙ ОБРАБОТКИ ДАННЫХ

Рассмотрим некоторые варианты аппаратной реализации цифровой схемы в виде конечного автомата. Оценим максимальную производительность подобных схем при поступлении на вход массива векторов $N$. Для обработки массива с максимальной пропускной способностью необходимо, чтобы скорость поступления информации соответствовала скорости ее обработки конкретной архитектурой. Обозначим длину вектора $|V|$ [бит], тогда длина всего сообщения в битах составит:

$$V = N \cdot |V|. \quad (5)$$

Также скорость обработки для случая максимальной производительности:

$$S = \frac{N \cdot |V|}{T}, \text{ бит/с}, \quad (6)$$

где $T$ – время обработки всего массива.

Учитывая, что алгоритм содержит $R$ раундов, получим время и скорость обработки массива $N$ в простейшей одноступенчатой циклической архитектуре (рис 2):

$$T = N \cdot R \cdot t_c, \quad (7)$$

где $t_c$ – время обработки одной ступени,

$$S = \frac{|V|}{R \cdot t_c}. \quad (8)$$

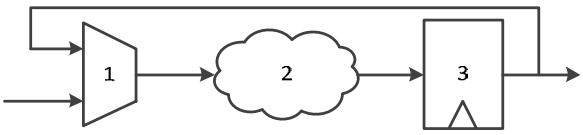


**Рис. 2.** Схема одноступенчатой циклической архитектуры: *1* – мультиплексор; *2* – комбинационная логика; *3* – регистр

Далее расширим данную схему до $M$ параллельных каналов (рис. 3).

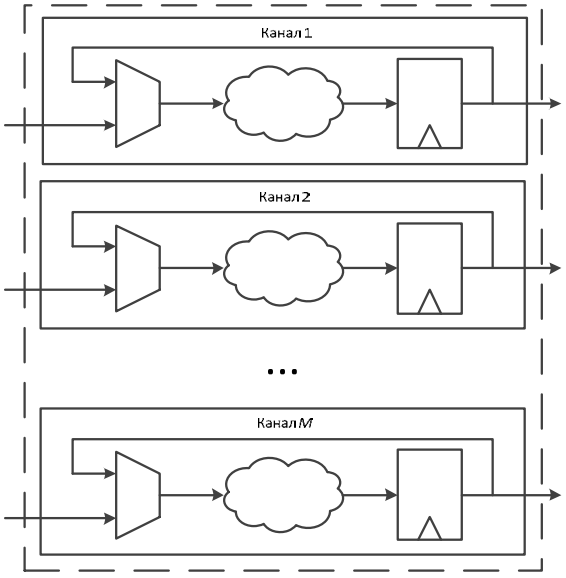


**Рис. 3.** Схема параллельной циклической архитектуры

Рассмотрим два режима работы. В первом режиме все блоки работают синхронно и находятся в одной фазе, следовательно, время обработки всего массива сократится в $M$ раз. Но из-за того что поток может содержать неполный $M$ блок, необходима дополнительная обработка в $R$ раундов:

$$T = O(N/M) \cdot R \cdot t_c, \quad (9)$$

при этом $O(N/M)$ есть округление до большего целого:

$$O(N/M) = 1 + \frac{(N-1)-(N-1)\ mod\ M}{M}. \quad (10)$$

Данное выражение следует из:

$$N = a \cdot M + b \quad (11)$$

где $a$ – количество полных $M$ блоков, $b$ – оставшиеся вектора, $b = N\ mod\ M$.

Скорость для большого массива составит:

$$S_a = M\frac{|V|}{R \cdot t_c}, \quad (12)$$

Второй режим осуществляется последовательным подключением каналов для дальнейшей обработки по мере поступления векторов. Для обеспечения максимальной производительности необходимо, чтобы на каждом такте работы схемы осуществлялась обработка $M$ векторов. Тогда время обработки массива:

$$T = O\left(\frac{N}{M}\right) \cdot R \cdot t_c + d \cdot t_c, \quad (13)$$

где $d$ определяет число дополнительных тактов после обработки первого вектора из текущего массива $M$ векторов. Простые логические рассуждения приводят к $d = (N–1)\ mod\ M$. Скорость для большого массива вычисляется аналогично первому режиму.

Применение в формулах $N$–1 вместо $N$ связано с тем, что количество входных векторов является натуральным числом.

Еще один вариант архитектуры можно рассматривать как раунд алгоритма с $k$ ступенями конвейеризации (рис. 4). В данном случае время обработки входного массива можно рассчитать как:

$$T = k \cdot R \cdot t_c + (N-1)t_c + ((N-1-(N-1)\ mod\ k) \cdot (R-1)t_c. \quad (14)$$

Первое слагаемое показывает время получения первого выходного вектора. Следующие $k$–1 векторов обрабатываются за время $t_c$, что отображено во втором слагаемом. Третье слагаемое рассчитывается из требования к отсутствию простаивания ступеней. А так как число ступеней задано, производится выборка $k$ векторов из входной последовательности, аналогично параллельной архитектуре. Скорость в данном случае описывается формулой, полученной для циклической одноступенчатой архитектуры.

$$S = \frac{|V|}{R \cdot t_c}. \quad (15)$$

В данном случае увеличение производительности заключается в повышении тактовой частоты в $k$ раз.

При полной конвейеризации алгоритма (рис. 5) с $R$ раундами полное время обработки составит:

$$T = k \cdot R \cdot t_c + (N-1) \cdot t_c, \quad (16)$$

а скорость:

$$S = \frac{|V|}{t_c}. \quad (17)$$

Полученные результаты занесем в таблицу (табл. 4). Из таблицы видно, что:

1. Циклическая архитектура является самой медленной.

2. При больших обрабатываемых массивах обе параллельные схемы обладают равными пропускными способностями.

3. При $k$=$M$ темп обработки параллельной архитектуры и циклической с конвейеризацией, на больших объемах данных совпадают.

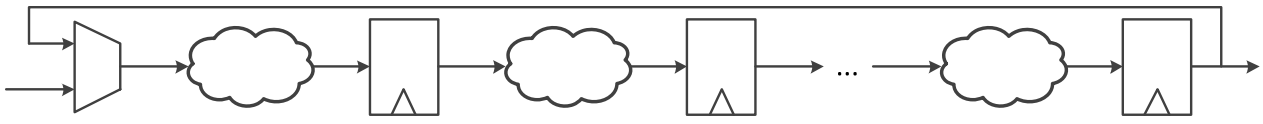


**Рис. 4.** Схема циклической архитектуры с конвейеризацией

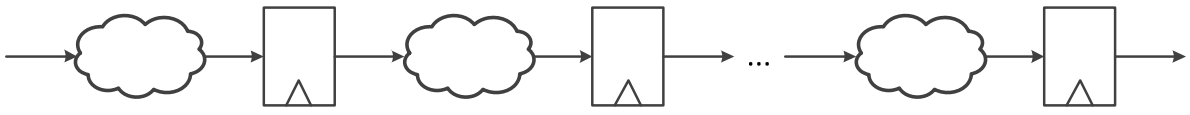


**Рис. 5.** Схема конвейеризированной архитектуры без обратной связи

4. Скорость параллельных архитектур увеличивается за счет введения дополнительных каналов обработки.

Таблица 4

**Сравнение производительности архитектур**

| Архитектура | Максимальная скорость обработки, S | Время обработки одной ступени, $t_c$ | Каналы обработки |
|---|---|---|---|
| Циклическая | $\frac{\lvert V\rvert}{R \cdot t}$ | $t$ | 1 |
| Параллельная синхронная | $\frac{M \cdot \lvert V\rvert}{R \cdot t}$ | $t$ | $M$ |
| Параллельно-последовательная | $\frac{M \cdot \lvert V\rvert}{R \cdot t}$ | $t$ | $M$ |
| Циклическая с конвейеризацией | $\frac{k \cdot \lvert V\rvert}{R \cdot t}$ | $\frac{t}{k}$ | 1 |
| Конвейеризированная без обратной связи | $\frac{k \cdot \lvert V\rvert}{t}$ | $\frac{t}{k}$ | 1 |

Таким образом, использование в качестве ускорителя параллельных и частично конвейеризированных циклических архитектур либо приводит к чрезмерному увеличению используемой поверхности кристалла, либо к недостаточному повышению производительности.

Существенное преимущество применения конвейерной архитектуры состоит в сокращении числа информационных каналов.

Теоретически, увеличение числа ступеней в $k$ раз приводит к уменьшению времени обработки одной ступени во столько же, но микросхема ППВМ имеет жесткую архитектуру на уровне базовых логических элементов [17], поэтому количество ступеней не может увеличиваться более, чем позволяет того отдельный элемент.

## ОСНОВНЫЕ ЭТАПЫ РАБОТЫ МОДУЛЯ ШИФРОВАНИЯ

В данной работе реализованы архитектуры для режимов простотой замены и гаммирования. На рис. 6 представлена общая структура криптографического модуля в режиме гаммирования. Такая структура, помимо шифрования и дешифрования данных из входного буфера, позволяет отправлять их напрямую в выходной канал, а также передавать сгенерированную синхропосылку для возможности последующего дешифрования.

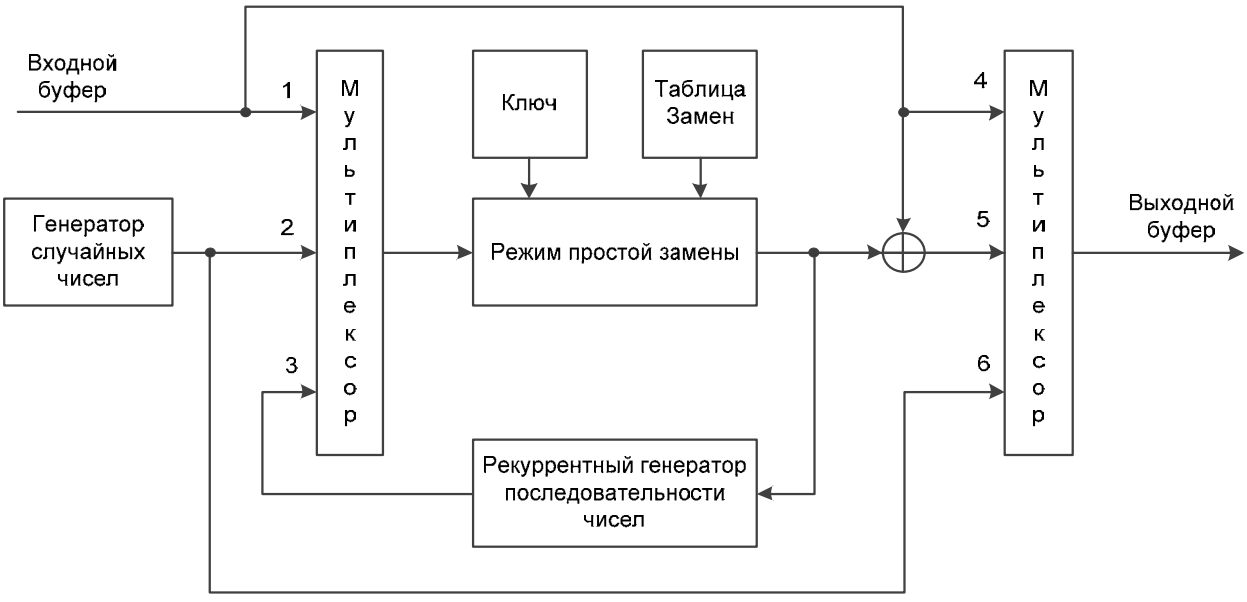


**Рис. 6.** Структурная схема криптографического модуля

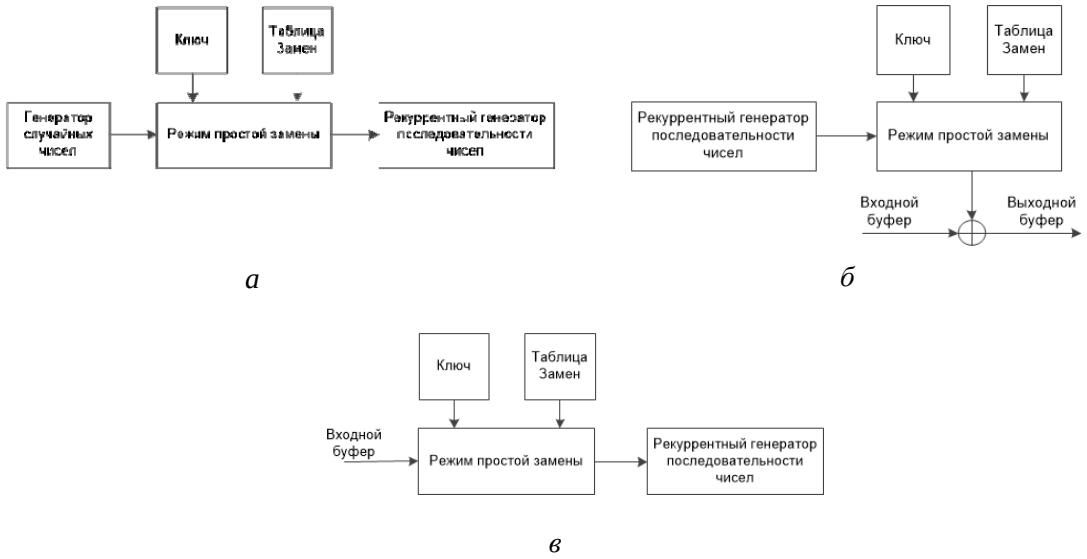


**Рис. 7.** Режимы работы генератора гаммы:
*а* – инициализация режима шифрования, *б* – режим шифрования/дешифрования, *в* – инициализация режима дешифрования

Рассмотрим режимы работы устройства (рис. 7). Режимы шифрования и дешифрования имеют два этапа — инициализация и обработка. Инициализация режима шифрования подразумевает этап получения первого выходного значения РГПЧ (рис. 7, *а*). Последующие значения вырабатываются за период работы основного сигнала тактирования. На этапе обработки последовательно соединенные блоки режима простой замены и РГПЧ объединяются в генератор гаммы (рис. 7, *б*). Тем самым достигается максимальное быстродействие с выработкой гаммы за один такт работы модуля. Режим дешифрования отличается тем, что вектор инициализации уже задан и требуется получить его из входного буфера с дешифруемым сообщением (рис. 7, *в*).

А на этапе обработки изменяется последовательность используемых подключей согласно стандарту. Режимы работы мультиплексоров приведены в табл. 5.

Таблица 5

**Состояния работы мультиплексоров**

| Режим работы | Состояние мультиплексора 1 | Состояние мультиплексора 2 |
|---|---|---|
| Инициализация шифрования | 2 | 3 |
| Гаммирование | 3 | 5 |
| Инициализация дешифрования | 1 | 3 |

Окончание табл. 5

| | | |
|---|---|---|
| Отправка синхропосылки | X | 6 |
| Байпас | X | 4 |

В связи со сказанным выше прямая передача шифротекста, полученного в ходе преобразования в режиме простой замены, в выходной буфер отсутствует.

## РЕЗУЛЬТАТЫ МОДЕЛИРОВАНИЯ

Для проведения вычислительного эксперимента и моделирования тестовых воздействий использовался язык описания аппаратуры VHDL. Моделирование проводилось в пакете Xilinx ISE Design Suite. Полученные результаты работы модуля шифрования занесены в таблицу (табл. 6). Из таблицы видно, что максимальная частота тактирования в режиме гаммирования ограничена значением 395 МГц. Это связано с введением в тракт обработки данных рекуррентного генератора последовательности чисел, работающего в однотактном режиме. Для сравнения полученных результатов на рис. 8 приведены характеристики режима простой замены и режима гаммирования. Также нарисована гипербола, показывающая минимальное времени обработки отдельной ступени конвейера при идеальных условиях. Для большей наглядности графика время обработки задано в логарифмическом масштабе. Видно, что при большом

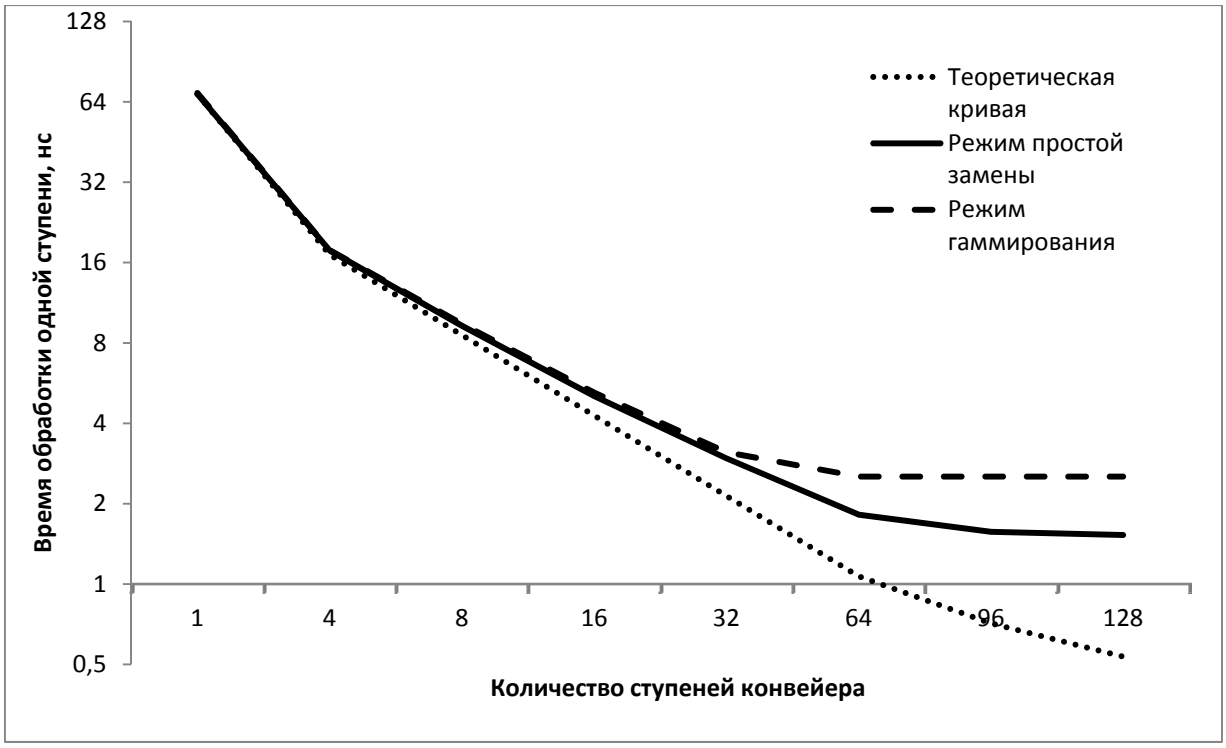


**Рис. 8.** Результаты моделирования алгоритма

количестве ступеней появляется эффект насыщения в росте производительности. На основе полученных моделей выбрано оптимальное количество ступеней для каждого шага преобразования, равное двум. Соответственно, для конвейеризированного алгоритма в режиме простой замены задержка получения первого блока составляет 64 такта (129 тактов для режима гаммирования). Данная конфигурация ядра позволяет получить максимальную скорость обработки 64 бит/такт на частоте 395 МГц, что соответствует пропускной способности в 23 Гбит/с.

Параметры использования ресурсов микросхемы xc6vlx760 приведены в табл. 7.

Таблица 6

Результаты моделирования режимов работы шифратора

| Число ступеней | Режим простой замены | | Режим гаммирования | |
|---|---|---|---|---|
| | Частота, МГц | Период, нс | Частота, МГц | Период, нс |
| 1 | 14,611 | 68,441 | 14,482 | 69,053 |
| 4 | 56,371 | 17,74 | 55,917 | 17,798 |
| 8 | 107,829 | 9,274 | 106 | 9,433 |
| 16 | 197,736 | 5,057 | 191,606 | 5,219 |
| 32 | 338,329 | 2,956 | 320,77 | 3,117 |
| 64 | 549,602 | 1,819 | 395,82 | 2,526 |
| 96 | 636,76 | 1,57 | 395,82 | 2,526 |
| 128 | 655,179 | 1,526 | 395,82 | 2,526 |

Таблица 7

Использование ресурсов микросхемы xc6vlx760

| Тип логики | Использовано | Доступно | Используемые ресурсы, % |
|---|---|---|---|
| Число регистров | 3954 | 948480 | 0 |
| Число LUT таблиц | 4445 | 474240 | 0 |
| Число LUT-FF пар | 3056 | 5343 | 57 |
| Блоки ввода-вывода | 301 | 1200 | 25 |
| Глобальные буферы | 3 | 32 | 9 |

## ЗАКЛЮЧЕНИЕ

Разработанный модуль для средств комплексной защиты позволяет значительно увеличить скорость обработки информации, что говорит о возможности его применения в узлах вычислительной сети класса 10 Gigabit Ethernet. Получена оптимальная архитектура блока шифрования с максимальной пропускной 23 Гб/с. Приведены показатели утилизации микросхемы ППВМ для полученного вычислителя. Обоснована целесообразность применения режима гаммирования для потоковой обработки данных. Оценена вероятность компрометации клю-

ча шифрования в рассматриваемом режиме. Разработанна методика работы модуля.

### СПИСОК ЛИТЕРАТУРЫ


1. **ГОСТ 28147-89.** Системы обработки информации. Защита криптографическая. Алгоритм криптографического преобразования. Введ. 01.07.90. М.: Государственный комитет СССР по стандартам: Изд-во стандартов. 1989. 28 с. **[** *Information Processing Systems - Cryptographic Security - Cryptographic Processing Algorithm,* (in Russian), Federal standard 28147-89, Moscow, Gosudarstvenny komitet SSSR po standartam: Izd-vo standartov, 1989. **]**

2. **Chodowiec P., Khuon P., Gaj K**. Fast implementations of secret-key block ciphers using mixed inner- and outer-round pipelining // Proc. 9th ACM/SIGDA International Symposium on Field Programmable Gate Arrays FPGA'01 (Monterey, CA, USA, Feb. 11-13 2001) ACM New York, 2001. P. 94-102

3. **Gaj K., Chodowiec P.** FPGA and ASIC Implementations of AES // Cryptographic Engineering. 2009. P. 235–294

4. **Hodjat, A.; Verbauwhede, I.** A 21.54 Gbits/s fully pipelined AES processor on FPGA // Proc. 12th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (Napa, California, Apr. 20-23 2004). IEEE Computer Society, 2004. P. 308–309

5. **Kotturi, D., Seong-Moo Yoo ; Blizzard, J.** An AES crypto chip using a high-speed parallel pipelined architecture // Proc. IEEE International Symposium on Circuits and Systems ISCAS 2005 (Kobe, Japan, May 23–26 2005). IEEE Microprocessors and Microsystems, 2005. Vol. 5. P. 4653–4656

6. **Prathyusha C., Rani P. S.** Implementation of Fast Pipelined AES Algorithm on Xilinx FPGA // Ijsrnet Editorial. 2013. Vol.2, N. 8. P. 377–381

7. **Qin H., Sasao T., Iguchi Y.** An FPGA Design of AES Encryption Circuit with 128-bit Keys // Proc. 15th ACM Great Lakes Symposium on VLSI GLSVLSI'05 (Chicago, Illinois, USA, Apr. 17–19 2005). ACM New York, 2005. P. 147–151

8. **Rajaram M., Vijaya J.** High Speed Pipelined AES with Mixcolumn Transform // European Journal of Scientific Research. 2011. Vol.61, N. 2. P. 255–264.

9. **Sayilar G., Chiou D.** Cryptoraptor: High throughput reconfigurable cryptographic processor // Proc. ACM International Conference on Computer-Aided Design ICCAD'2014 (San Jose, CA, Nov. 2–6 2014). IEEE Press Piscataway, 2014. P. 155–161

10. **Subashri T., Arunachalam R., Gokul Vinoth Kumar B., Vaidehi V.** Pipelining Architecture of AES Encryption and Key Generation with Search Based Memory //International journal of VLSI design & Communication Systems (VLSICS). 2010. Vol.1, N. 4. P. 48–60

11. **Андреев, А. Е., Силкин И. М., Шафран Ю. В.** Прогнозирование производительности при реализации алгоритмов на гибридных архитектурах с сопроцессорами [Электронный ресурс]. URL: http://www.science-education.ru/103-6389 (дата обращения 17.09.2015). **[** A. E. Andreev, I. M. Silkin, and Y. V. Shafran (2015, Sep. 17). *Predicting of performance in the implementation of algorithms on hybrid architectures using coprocessors* [Online]. Available: http://www.science-education.ru/103-6389 **]**

12. **Архангельский А. В., Торопченов Н. Ю.** Опыт аппаратной реализации функциональных модулей средств защиты информации на интегральных схемах программируемой логики // Программные продукты и системы, 2013. №2 (102). С. 108–113. **[** A. V. Arhangelsky, N. Y. Toropchenov "Experience of the hardware implementation of the information protection tools functional modules using integrated circuits of programmable logic," (in Russian), in *Programmnye produkty i sistemy*, no. 2 (102), pp. 108-113, 2013. **]**

13. **Винокуров А. Ю.** Алгоритм шифрования ГОСТ 28147-89, его использование и реализация для компьютеров платформы Intel x86 [Электронный ресурс]. URL: http://www.enlight.ru/crypto/articles/vinokurov/gost_i.htm (дата обращения 06.08.2015). **[** A. Y. Vinokurov (2015, Aug. 6). *The encryption algorithm GOST 28147-89, use and realization for computers with Intel x86 platform* [Online]. Available: http://www.enlight.ru/crypto/articles/vinokurov/gost_i.htm **]**

14. **Ильин В.Н, Гришин Р.А.** Анализ различных вариантов реализации алгоритма за/расшифрования с использованием двухуровневого макромоделирования // Вестник МАИ, 2010. №3. С. 200–205. **[** V. N. Ilyin, R. A. Grishin "An analysis of the various options for the implementation of the algorithm en/decryption using a two-tier macromodelling" (in Russian), in *Vestnik MAI*, no. 3, pp. 200-205, 2010. **]**

15. **Кролевецкий А. В.** Производительность ГОСТ шифрования на x86 и GPU процессорах // Научно-практическая конференция «РусКрипто'2014»: 16-я Междунар. конф.: тр. конф. М.: Код Безопасности, 2014. 17 с. **[** A. V. Krolevetcky, "Performance of GOST encryption on x86 and GPU processors", (in Russian), in *Proc. 16th Int. Workshop (RusKripto -2014)*, Moscow, Russia, 2014.**]**

16. **Логинов С.С., Голиков А.М.** Исследование алгоритма криптографической защиты ГОСТ 28147–89 и его аппаратно-программная реализация на ПЛИС // Электронные средства и системы управления: 7-я Междунар. конф. (Томск, 10–11 нояб. 2011): тр. конф. Томск: В-Спектр, 2011. С. 14–19 . **[** S. S. Loginov, A. M. Golikov "Research of algorithm for cryptographic protection of GOST 28147-89 and its hardware and software implementation on FPGA" (in Russian), in *Proc. 7th Int. Workshop (Electronic instrumentation and control systems-2014)*, Tomsk, Russia, 2011, pp. 14-19. **]**

17. **Грушвицкий Р.И., Мурсаев А.Х., Угрюмов Е.П.** Проектирование систем на микросхемах программируемой логики. СПб.: БХВ-Петербург, 2002. 608 с. **[** R. I. Grushvitcky, A. H. Mursaev, and E. P. Ugryumov, *System designing on programmable logic chips*, (in Russian). Saint-Petersburg: BKHV-Peterburg, 2002. **]**


### ОБ АВТОРАХ


**МИХАЙЛЮК Тарас Евгеньевич**, асп. каф. электроники и биомедицинских технол. М-р электроники и наноэлектроники (УГАТУ, 2014). Готовит дисс. в обл. комплексной защиты инф.

**ЖЕРНАКОВ Сергей Владимирович**, зав. каф. электроники и биомедицинских технол. Дипл. инж. по пром. электронике (УГАТУ, 1984). Д-р техн. наук по сист. анализу, упр. и обр. инф. (УГАТУ, 2005). Иссл. в обл. интел. систем.


### METADATA

**Title:** On the question of hardware implementation module streaming encryption for comprehensive information security system.


**Authors:** T. E. Mikhailyuk [1], S. V. Zhernakov[2]

**Affiliation:**

[1,2] Ufa State Aviation Technical University (UGATU), Russia.

**Email:** [1] realotoim@mail.ru, [2] zhsviit@mail.ru.

**Language:** Russian.

**Source:** Vestnik UGATU (scientific journal of Ufa State Aviation Technical University), vol. 19, no. 4 (70), pp. 138-148, 2015. ISSN 2225-2789 (Online), ISSN 1992-6502 (Print).

**Abstract:** We consider the optimized hardware implementation of GOST 28147-89 in a mode of XOR FPGA-based high-performance tools for complex information security systems. Analyzes modern methods of construction streaming encoders. The results of the simulation of the synthesized IP-core environment Xilinx ISE Design Suite. Estimated speed performance data streaming architectures.

**Key words:** GOST 28147-89; FPGA; streaming encryption.

**About authors:**

**MIKHAILYUK, Taras Evgenyevich**, Postgrad. (PhD) Student, Dept. of Electronics and Biomedical Technology. Master of Electronics & Nanoelectronics (UGATU, 2014).

**ZHERNAKOV, Sergey Vladimirovich,** Dr. (Habil.) Tech. Sci, Prof., Head, Dept. of Electronics and Biomedical Technology, Ufa State Aviation Technical University.